April 21, 2012

Age-Layered Expectation Maximization for Parameter Learning in Bayesian Networks

Avneesh Saluja, *Carnegie Mellon University*
Priya Sundararajan, *Carnegie Mellon University*
Ole J Mengshoel, *Carnegie Mellon University*

Age-Layered Expectation Maximization for Parameter Learning in Bayesian Networks

Avneesh Saluja
avneesh@cmu.edu
Carnegie Mellon University

Priya Krishnan Sundararajan
priya.sundararajan@sv.cmu.edu
Carnegie Mellon University

Ole J. Mengshoel
ole.mengshoel@sv.cmu.edu
Carnegie Mellon University

Abstract

The expectation maximization (EM) algorithm is a popular algorithm for parameter estimation in models with hidden variables. However, the algorithm has several non-trivial limitations, a significant one being variation in eventual solutions found, due to convergence to local optima. Several techniques have been proposed to allay this problem, for example initializing EM from multiple random starting points and selecting the highest likelihood out of all runs. In this work, we a) show that this method can be very expensive computationally for difficult Bayesian networks, and b) in response we propose an age-layered EM approach (ALEM) that efficiently discards less promising runs well before convergence. Our experiments show a significant reduction in the number of iterations, typically two- to four-fold, with minimal or no reduction in solution quality, indicating the potential for ALEM to streamline parameter estimation in Bayesian networks.

1 Introduction

The expectation maximization (EM) algorithm [Dempster et al., 1977, McLachlan & Krishnan, 2008] is one of the most established ways to perform parameter estimation with incomplete or hidden data (e.g., [Rabiner, 1989, Bauer et al., 1997]). The basic idea of the algorithm is to alternate between an expectation (E) step, wherein the algorithm uses current parameter estimates to generate a complete data likelihood, and a maximization (M) step, in which the parameters are modified with the goal of maximizing the data likelihood. These steps repeat until convergence. The algorithm and its variants and special cases are prevalent in machine learning, as one can view training algorithms as optimizers, where the overall aim is to estimate a set of parameters while maximizing a 'score', most often the data likelihood, on training data.

Despite its successful and widespread use, the EM algorithm has at least one severe limitation: it has a strong tendency to gravitate towards the locally optimal solution, depending primarily on the initialization of the algorithm. This phenomenon can in turn lead to other issues, for example poor generalization to unseen test data, and is especially exacerbated in parameter estimation for difficult Bayesian networks, where we attempt to estimate parameters for a joint distribution with many conditional dependencies. Another limitation is that the EM algorithm can be very time-consuming, due to its slow convergence speed, in terms of the number of iterations undergone.

In this work, we focus on speeding up the EM algorithm for difficult Bayesian networks with no or minimal degradation in solution quality.

- We present and analyze our approach, age layered EM (ALEM) (Section 3), where we incorporate an age-layered population structure heuristic in an attempt to reduce the average number of iterations in a multiple starting point EM setup.
- We highlight the local optima and slow convergence problems for a multiple starting points EM strategy for select difficult Bayesian networks (Section 4.2). Our experiments with these networks show that the main problem is the slow convergence of the EM algorithm rather than the log-likelihood shortfall of the local optima.
- We demonstrate ALEM's ability to significantly reduce the average number of iterations (Section 4.3), typically two- to four-fold, and for larger sample sizes the wall-clock time as well by the same magnitude.

2 Related Work

Several approaches have attempted to mitigate the local optima problem in the EM algorithm. One can use an upper bound on the eventual log likelihood of

Appearing in Proceedings of the 15^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume 22 of JMLR: W&CP 22.

a run at termination as a criterion for early dismissal of a run [Zhang et al., 2008]. Alternatively, the training data can be perturbed, rather than the hypothesis, to generate new directions for greedy hill-climbing ascent [Elidan et al., 2002]. The training data is then annealed back to its original state over time. A genetic algorithm approach has also been used [Jank, 2006]. Some work has looked at alternative stopping criteria that are less likely to converge to local optima [McLachlan & Peel, 2000].

A prevalent way to tackle local optima is the multiple starting point or multiple restart strategy [Mengshoel et al., 2011, Jacobson & Ycesan, 2004], wherein we initialize the EM algorithm from N random starting points. After the N EM runs have completed, we choose the run that resulted in the highest data log likelihood. This strategy allows us to search the parameter space more extensively for the optimal values, but can be extremely expensive, considering that we expend a significant amount of processing cycles on runs that are in no way guaranteed to be the global optimum or even close to it. We show this experimentally in Section 4.

The severity of the local optima problem in EM has been demonstrated [Wang & Zhang, 2006], but the authors restricted their analysis to a specific type of Bayesian network, the hierarchical latent class model. In Section 4, we perform a similar analysis on two networks that do not fall into the hierarchical latent class model, and show that the issue is prevalent in a wider class of networks but that the main problem is the variation in average number of iterations needed, and not so much log likelihoods of the local optima.

Additional work has looked at scaling up EM to very large datasets, for example incremental and lazy variants of EM [Thiesson et al., 2001]. Incremental EM partitions a dataset into blocks, with EM computed incrementally on these blocks, the main variable being optimal block size selection [Ng & McLachlan, 2003]. Lazy EM picks a significant subset of the dataset after a given number of iterations, and then proceeds with EM using only this subset. The emphasis of these techniques is how to scale EM to large databases, whereas our work focuses not necessarily on scale but on the orthogonal issue of how to speed up parameter estimation on *difficult* Bayesian networks. As such, the scalability methods and our approach can easily be combined. Simulated annealing [Lavielle & Moulines, 1997] and Tabu search methods [Nanda et al., 2005] have been incorporated into EM, but they serve primarily as an alternative to the multiple starting point strategy with the goal of avoiding local minima, and often require a re-implementation of the core algorithm.

Our approach is influenced by the age-layered population structure (ALPS) paradigm, originally implemented for genetic algorithms [Hornby, 2006]. ALPS assigns an *age* to each individual in the population, which is then used to sort the individuals into different age layers, each layer having its own age limit. Competition amongst individuals in the population is restricted to within age layers, and all layers evolve independently. This division of the population aims to create a more 'fair' competition between individuals, as older individuals in the population compete amongst themselves and not with substantially younger individuals. When an individual's age exceeds upper limit of its current layer, it is moved up to the next layer if it can replace a poor performing individual in the target layer, otherwise it is discarded.

3 Expectation Maximization Approach

We present and analyze our algorithm which, like ALPS, relies on an age-layered structure to efficiently remove underachieving runs early on, along with pseudocode for our algorithm.

3.1 ALEM: Age-Layered EM

The EM algorithm with multiple random starting points (henceforth referred to as 'traditional') can be viewed as a genetic algorithm: each starting point can be seen as an individual. For the age-layered paradigm, a natural measure of age would be the number of iterations an EM run has undergone. With these basic concepts, we developed ALEM (Figure 1), the Age-Layered Expectation Maximization algorithm.

In the *main* procedure of Figure 1, we have a set of L layers, $\Gamma = \{\Gamma_1, \Gamma_2, \Gamma_3, \ldots, \Gamma_L\}$, and a set of N EM runs, $\rho = \{\rho_1, \rho_2, \rho_3, \ldots, \rho_N\}$ where $\Gamma_i \subseteq \rho$. We denote the age limit or the maximum number of iterations for the i^{th} layer to be β_i, the minimum number of runs in the i^{th} layer to be M_i (these parameters control how runs move between layers and will be elaborated upon shortly), and the set of runs[1] in the i^{th} layer to be Γ_i. Note that $\Gamma_1 \cup \cdots \cup \Gamma_L = \rho$, and $\Gamma_i \cap \Gamma_j = \emptyset$, for $1 \leq i, j \leq L$, $i \neq j$. The number of iterations reached by the j^{th} EM run is given by $\eta(\rho_j)$ and its log likelihood is given by $LL(\rho_j)$.

Each EM run ρ_j is created by initializing its variables with randomly generated initial probabilities. We predefine the age limit β_i for each layer Γ_i, except for the last layer, through an exponential relationship between age limit and layer number: $\beta_i = a \cdot 2^{i-1}$, where β_i is the age limit for the i^{th} layer, and a is a predetermined constant (the age gap). Other relationships can also be used, but we chose this representation as it corresponded well empirically with the distribution of iterations in a multiple starting points strategy (Section 4.2). The last layer has no age limit, it is set to the maximum number of iterations ω that an EM run can undergo as specified for the EM algorithm: $\beta_L = \omega$.

In ALEM, there are three ways to terminate a run. First, no run can exceed ω, the maximum num-

[1] Γ_i refers to both the layer as well as the set of runs in that layer.

ber of iterations for a run. Second, there is a log likelihood difference tolerance ε (the relative difference in log likelihood between two successive iterations). These two termination criteria, as shown in ALEMCheck (Figure 1), are standard convergence criteria used for the traditional EM algorithm [Zhang et al., 2002, Carson et al., 1999] and are not specific to ALEM.[2] A run that terminates through these two termination criteria is called an *inactive* run and resides in the layer it was last in; all other non-terminated runs are called *active* runs. The third way, which we call *culling*, is ALEM-specific and amounts to a run failing the log likelihood comparison test. As shown in CheckRuns (Figure 1), if the age of an EM run ρ_j in layer Γ_i reaches β_i, then we remove it from layer Γ_i and try to insert it into the next layer Γ_{i+1}. An attempted insertion works as follows: each layer has an associated *minimum runs* parameter, M_i. We define the minimum run M_i for a layer Γ_i as the minimum number of runs that need to be in the layer before ALEM does a log likelihood comparison check. If the number of active runs in Γ_{i+1} is less than M_{i+1}, then we automatically insert the EM run from Γ_i into Γ_{i+1}. Otherwise, we perform a log likelihood comparison between the EM run to be shifted up, ρ_j, and runs, active or inactive, in Γ_{i+1}: if a run with worse log likelihood is found in Γ_{i+1}, then that run is *discarded*, i.e., it is completely removed from the age layers, and ρ_j takes its place. If both the minimum runs and the log likelihood comparison conditions fail, then ρ_j is discarded and we proceed. Discarding only takes place when a run fails the log likelihood comparison test.

We can intuitively view M_i as a parameter that can trade off computational efficiency and solution quality. The lower the value of this parameter for a given layer Γ_i, the more likely ALEM is to conduct a log likelihood check. On the other hand, inserting the run without a check will mean we do not get a chance to cull. For simplicity in the rest of this paper we assume all internal layers have the same minimum runs value ($M_2 = M_3 \cdots = M_{L-1}$), except for the first layer Γ_1 and the last layer Γ_L. In Γ_1, new runs are inserted when the number of active runs in that layer is below M_1, and thus M_1 controls the rate at which we introduce these new runs. The last layer can accommodate all N EM runs, and thus $M_L = N$.

The main assumption is that log likelihood comparisons between similarly aged runs are a reliable indicator of comparisons at convergence. Justification for this assumption is provided in Section 4.2, where we discuss empirical results on the distribution of log likelihoods and iterations. We note a significant difference in iterations between runs that achieve the highest log likelihood and runs that do not, which led us to posit that many runs are unsuccessful because they are initialized in bad areas of the search space.

[2]Despite the prevalence of these criteria, it has been noted [Lindstrom & Bates, 1988] that the difference in successive log likelihoods is a measure of lack of progress rather than convergence.

Algorithm 3.1: ALEM (N, L, M)

```
procedure CheckRuns(Γ_i, ρ_l)
 if |Γ_{i+1}| < M_{i+1}
   then { Γ_{i+1} ← Γ_{i+1} ∪ {ρ_l}
          inserted ← true
   else { comment: find if there is a worse run
          for each ρ_m ← Γ_{i+1}
            do { if LL(ρ_m) < LL(ρ_l)
                   then { Γ_{i+1} ← Γ_{i+1} − {ρ_m}
                          Discard EM run ρ_m
                          Γ_{i+1} ← Γ_{i+1} ∪ {ρ_l}
                          inserted ← true
 if inserted = false
   then {Discard EM run ρ_l
 Γ_i ← Γ_i − {ρ_l}

procedure ALEMCheck (N, L, M)
 comment: x denotes the number of terminated runs
 x ← 0
 while x ≤ N
   do { for i ← 1 to L
        do { if i = 1 and |Γ_i| < M_i
               then { k ← M_1 − |Γ_1|
                      Insert k EM runs
                      Γ_1 ← Γ_1 ∪ {ρ_j, ρ_{j+1} ··· ρ_k}
                      Start traditional EM algorithm
                      for each ρ_j ← Γ_1
                        do EM(ρ_j, ε, ω)
             for ρ_l ← Γ_i
               do { If EM run terminated based on ω or ε
                    if ρ_l.isTerminated
                      then x ← x + 1
                      else if η(ρ_l) = β_i
                      then CheckRuns(Γ_i, ρ_l)

main
 { Initialize layers with age limit β
 for i ← 1 to L − 1
   do β_i ← (a) · 2^{i−1}
 Set β_L for the top layer
 β_L ← ω
 Initialize layers with minimum runs M
 for i ← 2 to L − 1
   do M_i ← 2
 Set M for the bottom-most layer
 M_1 ← q
 Set M for the top layer
 M_L ← N
 Insert EM runs in the bottom-most layer
 Γ_1 ← {ρ_1, ρ_2 ··· ρ_q}
 Start traditional EM algorithm on each run
 for each ρ_q ← Γ_1
   do EM(ρ_q, ε, ω)
 ALEMCheck(Γ, ρ)
 Find EM run with max log likelihood
 for i ← 1 to L
   do LL ← arg max(Γ_i)
```

Figure 1: Pseudocode for ALEM algorithm. The values for β_i, M_i and a can be set depending on the nature of the Bayesian network. In our experiments, we have set $a = 5$, $\omega = 1000$, $N = 200$, $M_1 = 5$ for the bottom layer, $M_L = N$ for the top layer, $M_i = 2$ for $i < 2$ to $L-1$ and $\varepsilon = 0.00001$

3.2 Analysis using Poisson Processes

In traditional EM, runs mostly terminate because the algorithm fails to improve the log likelihood by a prespecified amount ϵ between successive iterations. In ALEM, a run terminates for the same reason, but in addition it can also be culled. In this section, we formalize this intuition by analyzing ALEM's rate of convergence as compared to that of traditional EM.

In a shared resource environment (e.g., all processes running on the same machine), the total time required to complete a multiple starting points experiment can be thought of as the product of the number of runs and the average time taken per run (or upper-bounded by the maximum time over all runs). If we take into account the fact that as certain runs terminate, computational resources free up and can be assigned to the runs that are still iterating, it then becomes a question of which strategy results in a higher probability of undesirable (runs that iterate for too long, runs that will converge to local optima, etc.) runs terminating early on. The main condition for ALEM's successful performance is that the probability of a young, poorly performing run turning into an older, strongly performing run is low, and can be made low by the ALEM parameters, i.e., β_i and M_i. We model the termination of runs stochastically by assuming that terminations in traditional EM and ALEM follow a Poisson process, motivated by the discrete, independent, and uniformly distributed nature of the events (terminations) occurring over a continuous time interval. Specifically, let $N(t)$ be a Poisson process that dictates the number of terminations in a given time interval, such that

$$P[(N(t+\tau) - N(t)) = k] = \frac{e^{\lambda\tau}(\lambda\tau)^k}{k!} \quad \text{for } k = 0, 1, \ldots \quad (1)$$

where k is the number of terminations. We can model ALEM terminations as the superposition of two independent Poisson processes: $N_{\text{EM}}(t)$ with Poisson parameter λ_{EM}, which is the process for terminations occurring through standard EM termination, and $N_{\text{cul}}(t)$ with Poisson parameter λ_{cul} which is the culling process. The resulting Poisson process $N_{\text{ALEM}}(t)$ has parameter $\lambda_{\text{ALEM}} = \lambda_{\text{EM}} + \lambda_{\text{cul}}$ [Kingman, 1993].

Lastly, if we let T^k_{ALEM} be the time taken for ALEM to reach k terminations, we can can say $T^k_{\text{ALEM}} = X^1_{\text{ALEM}} + X^2_{\text{ALEM}} + \cdots + X^k_{\text{ALEM}}$, where each element in the sequence X^i_{ALEM} is an i.i.d. exponential random variable with density function $f_{\text{ALEM}}(t) = \lambda_{\text{ALEM}} e^{-\lambda_{\text{ALEM}} t},\ t \geq 0$, and is the distribution of run i terminating before time t. We can similarly define T^k_{EM} for the traditional EM approach and also define it as a sum of i.i.d. exponential random variables with $\lambda = \lambda_{\text{EM}}$. Now let $Y_k = T^k_{\text{ALEM}} - T^k_{\text{EM}}$. Then,

$$P[T^k_{\text{ALEM}} < T^k_{\text{EM}}] = P[Y_1 + Y_2 + \ldots Y_k < 0] \quad (2)$$

noting that $Y_i, i = 1, \ldots, k$ are i.i.d. random variables with mean $\mu = \frac{1}{\lambda_{\text{ALEM}}} - \frac{1}{\lambda_{\text{EM}}}$ and variance $\sigma^2 = \frac{1}{\lambda^2_{\text{ALEM}}} + \frac{1}{\lambda^2_{\text{EM}}}$ (addition of independent exponential random variables). Thus,

$$\frac{\mu}{\sigma} = \frac{\lambda_{\text{EM}} - \lambda_{\text{ALEM}}}{\lambda_{\text{ALEM}}\lambda_{\text{EM}}} \frac{\lambda_{\text{ALEM}}\lambda_{\text{EM}}}{\sqrt{\lambda^2_{\text{ALEM}} + \lambda^2_{\text{EM}}}} = \frac{\lambda_{\text{EM}} - \lambda_{\text{ALEM}}}{\sqrt{\lambda^2_{\text{ALEM}} + \lambda^2_{\text{EM}}}} \quad (3)$$

Now let $T^k_Y = Y_1 + Y_2 \ldots Y_k$, and $Z_k = \frac{T^k_Y - k\mu}{\sigma\sqrt{k}}$. Then,

$$P[Y_1 + \ldots Y_k < 0] = P\left[\frac{T^k_Y - k\mu}{\sigma\sqrt{k}} < \frac{-k\mu}{\sigma\sqrt{k}}\right] = P\left[Z_k < \frac{\lambda_{\text{ALEM}} - \lambda_{\text{EM}}}{\sqrt{\lambda^2_{\text{ALEM}} + \lambda^2_{\text{EM}}}}\sqrt{k}\right] \quad (4)$$

From the central limit theorem, Z_k converges in distribution to the normal distribution with $\mu = 0, \sigma^2 = 1$. Therefore, if $\lambda_{\text{ALEM}} > \lambda_{\text{EM}}$, i.e., $\lambda_{\text{cul}} > 0$, then Equation 4 tends to 1 as $k \to \infty$. The analysis gives the probability of the time taken for ALEM to reach k terminations being less than the time taken for traditional EM to reach k runs. This probability approaches 1 as the number of terminations increases.

4 Experiments with Bayesian Networks

The objectives of the experiments are two-fold: to compare the average number of iterations undergone by all runs in traditional EM and in ALEM, and to analyze the solution quality of ALEM vis-à-vis traditional EM. In all sets of experiments, we set ω=1000, and $\varepsilon = 0.00001$, i.e., if the relative difference in log likelihood between two successive iterations is less than ε, then we deem the EM run to have converged or terminated and set its status to inactive. We also analyze the effects of varying the minimum runs parameter on the average iterations and the solution quality, and provide wall-clock times comparing the two strategies.

4.1 Datasets & Evaluation

We performed experiments on two Bayesian networks of different complexities, the Carstarts network, which contains 18 nodes (random variables), and the Alarm network, with 37 nodes (random variables).[3] The Carstarts network is based on the various operations in a car and the more complex Alarm network represents a real life situation of monitoring a patient in an intensive care unit.

For each of these networks, we generated $N = 200$ EM runs by randomly choosing starting points in the search space, in this case starting conditional probability distribution values, for each of the random variables in each run. We then varied two experimental

[3] http://www.cs.cmu.edu/∼javabayes/Examples/

parameters, namely the training set size and the number of hidden variables. Firstly, through a Gibbs sampler, we generated training sets consisting of 100, 250, 500, 1000, 2000, and 4000 samples. For each sample size, we generated n different hidden variable configurations, where n is the number of random variables in the network, and for each configuration, we chose m variables to be hidden, where m ranges from 0 to $n-1$. For example, for the Carstarts network, we had 18 different hidden variable configurations, ranging from 0 hidden variables to 17 hidden variables. In all our experiments, the EM runs are run in parallel on the same Linux machine, a 2.5 GHz Intel quad core processor with 8GB RAM. We used the libDAI graphical models library [Mooij, 2010], which contains an EM implementation suitable for Bayesian network parameter estimation.

For our first set of experiments (Section 4.2), we calculated the *average relative likelihood shortfall* for each hidden variable configuration-sample size pair as:

$$\mathrm{RLS}_{\mathrm{avg}} = \frac{l_{\mathrm{avg}} - l^*}{l^*} \tag{5}$$

where l^* is the maximum log likelihood from the 200 EM runs and is deemed the global maximum,[4] and l_{avg} is the average log likelihood across all 200 EM runs. We also computed the average number of iterations undergone by all runs. This exhaustive analysis allowed us to identify which hidden variable configurations and sample sizes took a large number of iterations to converge (we term these configurations as 'problematic'), as well as the distribution of log likelihood values at convergence.

For the second set of experiments (Section 4.3), we focused on the hidden variable configurations from the first set of experiments that we found problematic, and performed EM parameter estimation through ALEM on these troublesome hidden variable configurations. We compare average iterations and solution quality (through Equation 5) between the traditional and ALEM approaches. The third set of experiments (Section 4.4) focuses on varying the minimum runs parameter; we restrict ourselves to the hidden variables chosen for the second experiment. The fourth group of experiments (Section 4.5) compares the wall-clock time between the traditional EM and ALEM, once again restricted to the problematic hidden variables.

4.2 Slow Convergence in Traditional EM

In the first set of experiments, we used traditional EM and went through the 6 sample sizes and the n different hidden variable configurations ($n = 37$ for Alarm, $n = 18$ for Carstarts) for each of the 200 EM runs. Therefore, the total number of EM runs amounted to $6 \times 37 \times 200 = 44,400$ for Alarm, and $6 \times 18 \times 200 = 21,600$ for Carstarts. For each hidden variable configuration-sample size pair, we calculated the average RLS as per Equation 5.

Figures 2 and 3 show the average number of iterations for the networks as we vary hidden variables and sample size. We found that $\mathrm{RLS}_{\mathrm{avg}}$ never exceeds 1%, which means that on average the log likelihood values obtained from each of the 200 EM runs was never more than 1% away from the global maximum. In addition, the average RLS peaks and average iteration peaks in the figures tracked each other fairly well. We generally find that the highest peaks occur with the 100 sample size experiments, although one cannot establish a trend that a lower sample size leads to a higher average number of iterations.

We also looked at the runs that eventually reached the global maximum (successful runs), and calculated the average iterations for just those runs versus the average over all runs, and found significant differences between these two sets of runs. The average number of iterations for all runs exceeds the average number of iterations for successful runs by 17.6 iterations in Carstarts and 5.2 iterations in Alarm, on average. This thorough analysis allowed us to identify problematic hidden variable configurations and sample sizes which were more time-consuming. Therefore, we felt that focusing on reducing the average number of iterations would result in the most significant improvement in terms of the overall time taken to complete the runs and find the global maximum.

4.3 ALEM: Mitigating Slow Convergence

Our ALEM implementation consisted of seven layers ($L = 7$). Based on results from traditional EM (Section 4.2), we chose the age gap a to be 5, and so $\beta_1 = 5$, $\beta_2 = 10$, $\beta_3 = 20$, $\beta_4 = 40$, $\beta_5 = 80$, $\beta_6 = 160$, and $\beta_7 = 1000$, as the last layer has no age limit per se and is simply restricted by ω. For layers 2 to 6 we set $M_i = 2$, with $M_1 = 5$ and $M_7 = 200$.

For the second set of experiments, we run ALEM on the hidden variable configurations from Section 4.2 that we found problematic, which were 3, 4, 7, and 16 hidden variables for the Carstarts network, and 7, 13, 19, 28, and 33 hidden variables for the Alarm network. These configurations broadly correspond to the peaks in Figures 2 and 3. Using ALEM, we did not *exactly* achieve the global maximum in 4 out of 24 hidden variable-sample size configurations (6 sample sizes $\times$ 4 hidden variable configurations) in Carstarts, although on average we were within 0.007% of the global maximum for the ones we missed. For Alarm, we were off the global maximum for 12 out of the 30 experiments (6 sample sizes $\times$ 5 hidden variable configurations), but the average shortfall for the configurations where we did not achieve the global maximum was also 0.007%. However, both of these $\mathrm{RLS}_{\mathrm{avg}}$ values are orders of magnitude less than the $\mathrm{RLS}_{\mathrm{avg}}$ obtained in the traditional experiments, which were 0.13% and

[4] We use for simplicity the term 'global' maximum for the purposes of this paper, although we recognize that the maximum achieved from multiple starting points is not necessarily the true global maximum.

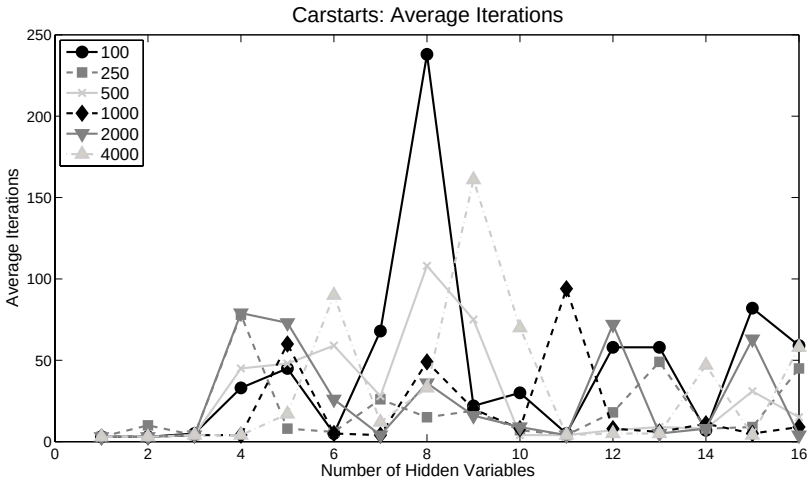


Figure 2: Carstarts network: average number of iterations for 200 EM runs, across all hidden variable and sample size configurations. Notice the peaks at particular hidden variable configurations.

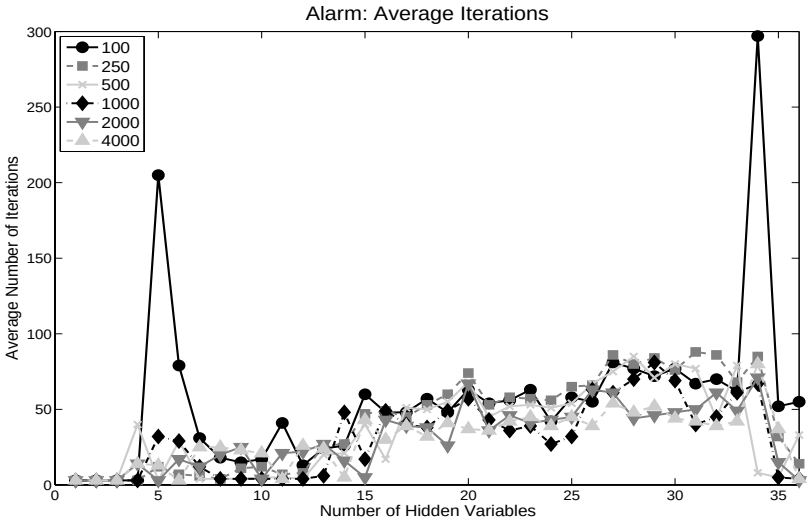


Figure 3: Alarm network: average number of iterations for 200 EM runs, across all hidden variable and sample size configurations. Notice the peaks at particular hidden variable configurations.

0.10% for Carstarts and Alarm respectively, indicating ALEM's effectiveness even on configurations where the global maximum was not found.

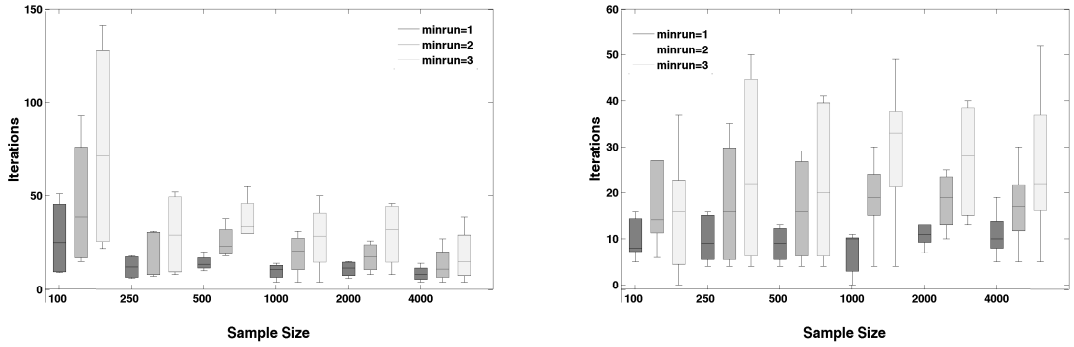
In no case did the average number of iterations ever increase with ALEM: mostly there is a significant decrease, especially on very hard (i.e., high iterations in traditional EM) instances. In some cases when the average number of iterations is already low there is no change. Table 1 contains the average number of iterations for the chosen hidden variable configurations in the ALEM approach, with the speedup $\frac{\text{traditional \# iterations}}{\text{ALEM \# iterations}}$ in parentheses. On average for Carstarts the decrease in average iterations is 51.4% and for Alarm it is 47.8%. The fact that we managed to reduce the average number of iterations for these problematic hidden variable configurations so significantly yet still achieve the global maximum in most instances (or very close to it in all instances) demonstrates to us how expensive the traditional strategy is. There are clearly some initial points that result in a very high number of iterations but tend not to find the global maximum due to the starting position in the search space. The ALEM approach effectively culls these iterations, thus saving processing cycles.

4.4 Parameter Variation

We performed a set of experiments where we varied the minimum runs parameter M_i to see its effects on the average number of iterations as well as the solution quality. Figure 4 is a box plot that shows how the average number of iterations (across the hidden variables that we tested ALEM for) varies as a function of the sample size and the minimum runs parameter. One can see a clear trend: as we reduce the minimum runs, the average iterations also decreases. In fact, some

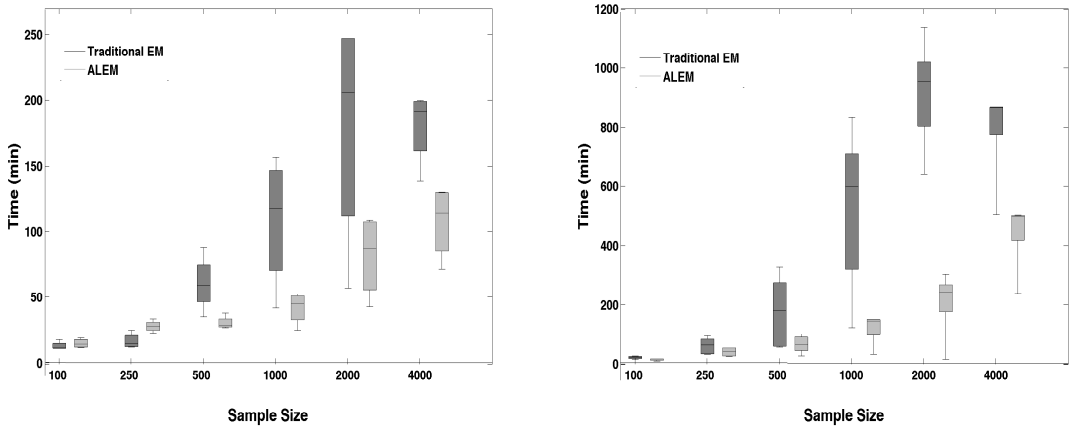
	Number of hidden variables								
	Carstarts Bayesian network				Alarm Bayesian network				
Sample size	**3**	**4**	**7**	**16**	**7**	**13**	**19**	**28**	**33**
100	15 (2.2)	19 (2.4)	93 (2.6)	**59 (3.2)**	14 (1.3)	13 (2.0)	27 (2.3)	27 (2.7)	**6 (49.5)**
250	31 (2.5)	7 (1.1)	9 (1.7)	**30 (3.1)**	4 (1.0)	16 (1.7)	28 (2.6)	35 (2.4)	**6 (14.2)**
500	26 (1.7)	20 (2.4)	38 (2.8)	**18 (3.1)**	4 (1.0)	4 (1.0)	29 (2.3)	**26 (2.7)**	7 (1.1)
1000	4 (1.0)	24 (2.5)	17 (2.9)	**31 (3.2)**	4 (1.0)	19 (2.5)	22 (2.6)	31 (2.7)	**19 (3.5)**
2000	26 (3.0)	**22 (3.3)**	13 (2.8)	8 (1.1)	10 (2.0)	14 (1.1)	**23 (2.9)**	19 (2.4)	25 (2.8)
4000	4 (1.0)	9 (1.9)	13 (2.5)	**27 (2.7)**	14 (1.8)	5 (1.0)	17 (2.2)	19 (2.7)	**30 (2.7)**

Table 1: Average number of iterations for chosen hidden variable configurations for the ALEM approach, with speedup $= \frac{\text{traditional iterations}}{\text{ALEM iterations}}$ in parentheses, for Carstarts & Alarm networks. Highest speedups are in bold. These results corroborate with the wall-clock time experiments shown in Figure 5.



(a) Carstarts Bayesian network (b) Alarm Bayesian network

Figure 4: Variation in the number of iterations ALEM runs undergo, on average, as a function of the minimum runs parameter M_i for both networks. The lower M_i is, the fewer iterations undergone. Darker shades of gray denote lower values of M_i.



(a) Carstarts Bayesian network (b) Alarm Bayesian network

Figure 5: Wall clock time comparison between traditional EM and ALEM. ALEM is, for larger sample sizes, significantly faster and the variation amongst runs tends to be much smaller. Traditional EM is in dark gray, ALEM is in light gray.

Bayesian network	Minimum runs(M_i)	Failures	RLS_{avg}
Carstarts	1	5	0.009%
Carstarts	2	4	0.007%
Carstarts	3	4	0.001%
Alarm	1	14	0.013%
Alarm	2	12	0.007%
Alarm	3	6	0.006%

Table 2: Minimum runs variation and solution quality: 'Failures' is a count of the number of hidden variable-sample size configurations (out of a total of 24 for each entry for Carstarts, and 30 for each entry for Alarm) where we fail to achieve the global maximum as we vary the minimum runs, and RLS_{avg} of *only those experiments* that did not achieve the global maximum.

of the average iteration reductions when setting the parameter to 1 were extremely significant (e.g., 10x reduction in iterations with no reduction in solution quality). However, generally the solution quality is also reduced; as we reduce M_i, there are some hidden variable-sample size configurations where the global maximum is not attained. Table 2 provides a count of the number of such hidden variable-sample size configurations. Each value in the 'Failures' column is out of a total of 24 configurations for Carstarts and 30 for Alarm (which corresponds to 6 sample size configurations $\times$ the number of hidden variable configurations chosen for the network). We also provide the RLS_{avg} of *only those experiments* that did not achieve the global maximum (the RLS for the others would be 0). As with average iterations, we can see a consistent variation of solution quality as we vary M_i to take values between 1 and 3.

M_i can be adjusted by methods like simulated annealing where a temperature parameter can be increased initially to allow less discarding of runs for exploration of the search space and then gradually decreased to allow more discarding of EM runs. Future work will concentrate on a method to tune this parameter.

4.5 Wall Clock Time Comparison

We recorded the time taken by traditional EM and ALEM when running the EM experiments across all six sample sizes for the problematic subset of hidden variables. Figures 5a and 5b show the box plots of the average time taken for both approaches. From these graphs we can clearly see a considerable decrease in time taken by ALEM for the Alarm network. For Carstarts, similar results are evident, but with sample sizes 500 and above. The variance in the number of iterations is also significantly smaller in ALEM. Thus, we can conclude that a reduction in average iterations, as reported in Table 1, translates well to a reduction in wall-clock time, especially for larger sample sizes.

5 Conclusion and Future Work

In this work, we present an age-layered influenced algorithm to mitigate the local optima problem in the EM algorithm. Specifically, our approach can be seen as a way to manage multiple EM runs, randomized with different initial starting points. We have shown that parameter estimation using EM on difficult Bayesian networks can be extremely computationally wasteful, underlying the need for an efficient method to restrict the overall average number of iterations undertaken for a given parameter estimation problem instance. We then show that ALEM manages to significantly decrease the average number of iterations required on two difficult Bayesian networks, but at the same still achieves the global optimum, or gets very close, in all instances.

Future work will look at robust methods to tune the minimum runs parameter as well as error bounds on using ALEM versus traditional EM. We are also intrigued by the relationship between the nature of the stopping criterion used and the convergence of EM, and will explore this aspect of the work too.

Acknowledgements: We would like to thank the anonymous reviewers for their helpful suggestions. This work is supported by DARPA contract D11PC20022 (Avneesh), and NSF awards CCF0937044 and ECCS0931978 (Priya and Ole).

References

[Bauer et al., 1997] Bauer, E., Koller, D. & Singer, Y. (1997). In Proc. Thirteenth Annual Conference on Uncertainty in AI (UAI) pp. 3–13,.

[Carson et al., 1999] Carson, C., Belongie, S., Greenspan, H. & Malik, J. (1999). IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 1026–1038.

[Dempster et al., 1977] Dempster, A., Laird, N., Rubin, D. & Others (1977). Journal of the Royal Statistical Society. Series B (Methodological) 39, 1–38.

[Elidan et al., 2002] Elidan, G., Ninio, M., Friedman, N. & Shuurmans, D. (2002). In Proceedings of the 18th National Conference on Artificial Intelligence pp. 132–139, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[Hornby, 2006] Hornby, G. S. (2006). In Proceedings of the 8th annual conference on Genetic and evolutionary computation: GECCO 2006.

[Jacobson & Ycesan, 2004] Jacobson, S. H. & Ycesan, E. (2004). Journal of Heuristics 10, 387–405.

[Jank, 2006] Jank, W. (2006). The EM algorithm, its stochastic implementation and global optimization: Some challenges and opportunities for OR, vol. 36,. Springer US.

[Kingman, 1993] Kingman, J. F. C. (1993). Poisson processes, vol. 3, of Oxford Studies in Probability. The Clarendon Press Oxford University Press, New York. Oxford Science Publications.

[Lavielle & Moulines, 1997] Lavielle, M. & Moulines, E. (1997). Statistics and Computing 7, 229–236.

[Lindstrom & Bates, 1988] Lindstrom, M. J. & Bates, D. M. (1988). Journal of the American Statistical Association 83, pp. 1014–1022.

[McLachlan & Peel, 2000] McLachlan, G. & Peel, D. (2000). Finite mixture models. Wiley series in probability and statistics: Applied probability and statistics, Wiley.

[McLachlan & Krishnan, 2008] McLachlan, G. J. & Krishnan, T. (2008). The EM Algorithm and Extensions (Wiley Series in Probability and Statistics). 2 edition, Wiley-Interscience.

[Mengshoel et al., 2011] Mengshoel, O. J., Wilkins, D. C. & Roth, D. (2011). IEEE Transactions on Knowledge and Data Engineering 23, 235–247.

[Mooij, 2010] Mooij, J. M. (2010). Journal of Machine Learning Research 11, 2169–2173.

[Nanda et al., 2005] Nanda, P., Patra, D. & Pradhan, A. (2005). In Proceedings of the 2nd Indian International Conference on Artificial Intelligence.

[Ng & McLachlan, 2003] Ng, S. K. & McLachlan, G. J. (2003). Statistics and Computing 13, 45–55.

[Rabiner, 1989] Rabiner, L. R. (1989). Proceedings of the IEEE 77, 257–286.

[Thiesson et al., 2001] Thiesson, B., Meek, C. & Heckerman, D. (2001). Machine Learning 45, 279–299.

[Wang & Zhang, 2006] Wang, Y. & Zhang, N. (2006). In Proceedings of the 3rd European Workshop on Probabilistic Graphical Models pp. 301–308,.

[Zhang et al., 2002] Zhang, Y., Xu, W. & Callan, J. (2002). In In Text Learning Workshop in International Conference on Machine Learning (ICML2002.

[Zhang et al., 2008] Zhang, Z., Dai, B. T. & Tung, A. K. H. (2008). In Proceedings of the 25th international conference on Machine learning ICML '08 pp. 1240–1247, ACM, New York, NY, USA.