International Journal on Artificial Intelligence Tools
© World Scientific Publishing Company

DENSE STRUCTURAL EXPECTATION MAXIMISATION WITH PARALLELISATION FOR EFFICIENT LARGE-NETWORK STRUCTURAL INFERENCE

CHRISTOPHER FOGELBERG

Department of Computer Science, University of Oxford,
Oxford, OX1 3QD, United Kingdom
cgf@syntilect.com

VASILE PALADE

Department of Computer Science, University of Oxford,
Oxford, OX1 3QD, United Kingdom
vasile.palade@cs.ox.ac.uk

Received (18th January 2012)
Revised (24th September 2012)
Accepted (22nd January 2013)

Research on networks is increasingly popular in a wide range of machine learning fields, and structural inference of networks is a key problem. Unfortunately, network structural inference is time consuming and there is an increasing need to infer the structure of ever-larger networks. This article presents the Dense Structural Expectation Maximisation (DSEM) algorithm, a novel extension of the well-known SEM algorithm. DSEM increases the efficiency of structural inference by using the time-expensive calculations required in each SEM iteration more efficiently, and can be $O(N)$ times faster than SEM, where N is the size of the network. The article has also combined DSEM with parallelisation and evaluated the impact of these improvements over SEM, individually and combined. The possibility of combining these novel approaches with other research on structural inference is also considered. The contributions also appear to be usable for all kinds of structural inference, and may greatly improve the range, variety and size of problems which can be tractably addressed. Code is freely available online at: `http://syntilect.com/cgf/pubs:software`.

Keywords: Bayesian networks; structural inference; SEM; parallelisation; large networks.

1. Introduction

Structural inference is the problem of inferring the set of dependencies (edges) in a graphical model using time series observations of the nodes. Accurate structural inference is needed to understand the functional relationship amongst the nodes and to predict their future states.

Bayesian networks (BNs) [18] and other statistical graphical models [21] have some advantages over other representations like *Boolean networks* [2] and *differential equations* [37]. These advantages include the capacity for modelling non-linear relationships, some causal inference of the underlying system [24,35] and robust inference in the face of missing and noisy data when using algorithms like *structural expectation maximisation* (SEM) [13,40].

On the other hand, the structural inference of BNs and other similar graphical models is NP hard [4,3], and this makes structural inference of networks with $N > 100$ nodes very difficult. This article presents *Dense SEM*, a general machine learning research contribution which extends the SEM algorithm to help address this problem, and it uses *genetic regulatory networks* (GRNs) to show the value of DSEM over SEM. The article also briefly presents a parallelisation of the proposed algorithm, and shows how the benefits of DSEM and parallelisation combine multiplicatively to allow for the successful inference of larger networks.

This article is structured as follows. Section 2 introduces GRNs and BNs, and presents previous research relevant to this article. Section 3 goes on to describe the major novel contribution of this article, DSEM. The resulting algorithm is up to $O(N)$ times more efficient than SEM, where N is the size of the network being inferred. In combination with the parallelisation summarised in Section 5, DSEM makes structural inference of larger networks more tractable, and some analysis on this is presented as well. Finally, Section 6 summarises this article, its contributions and future research. Access details for the implementations of the novel approaches described in this paper are also provided.

2. Background

Subsection 2.1 introduces BNs and Subsection 2.2 describes how BNs can be used to represent GRNs. Subsection 2.3 describes relevant previous research.

2.1. *Bayesian Networks*

BNs are comprehensively introduced in other publications (e.g., [21]) and will not be fully detailed in this article. However, note that BNs have also been extensively used in other fields, including natural language processing [5,34], machine translation [41], medical diagnosis [21] and technological fault diagnosis [25]. This means that the novel contributions illustrated using biological examples in this article may generalise easily to research in other fields as well. Figure 1 is the structure of an example BN.

Many real-world networks, including GRNs, are cyclic. However classic BNs must be acyclic because they use conditional probability distributions [10]. Fortunately, this problem can be addressed by representing each node's state at time t and time $t+1$ using two different variables.

Unrolling a BN in this way is analogous to unrolling a Markov chain over time. The resulting structure is known as a *dynamic Bayesian network* (DBN) [14,15,40] and can represent cyclic networks. Figure 2(a) shows a cyclical network which must be represented using a DBN, and Figure 2(b) shows the network structure if a DBN were used to model this network.

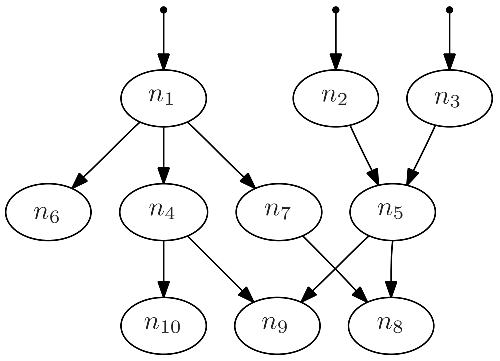

Fig. 1. A Bayesian network. The distributions for the nodes with no parents (i.e., n_1, n_2, n_3) are not conditional on any other node and are denoted by an incoming parentless edge.

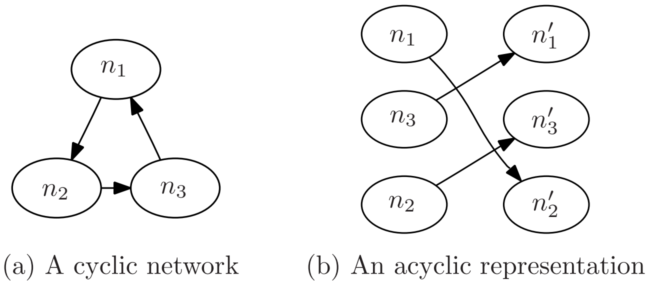

Fig. 2. A cyclic network structure (impossible to factorise) and the same dependencies unrolled over time, as they would be represented in a DBN.

Valid DBN structures are restricted so that edges cannot point “backwards” or “sideways” in time, but only forwards, and so a DBN is necessarily acyclic. Because GRNs are often cyclic, this article will only use DBNs.

2.2. *Genetic Regulatory Networks*

As well as having direct phenotypic effects, genes also turn each other on and off, and these relationships can be represented using a directed graphical model (e.g., Figure 3). Understanding what these regulatory relationships are is important for a range of reasons [10], and the importance of genome-scale (i.e., large network) structural inference is also increasingly appreciated [16,17,19,31,32,33,29].

Genome-scale GRNs often contain 6000 or more genes, making them 1 to 2 orders of magnitude larger than is tractable with SEM. Similarly, techniques using other graphical representations like Boolean networks are also intractable on genome-scale networks [22]. This illustrates and motivates the need for novel approaches to structural inference.

A gene’s *expression level* is a real-valued measurement of how strongly the gene

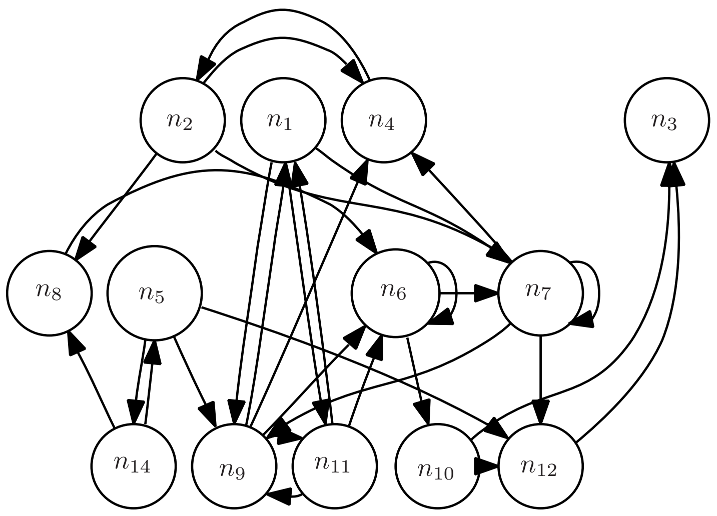

Fig. 3. A hypothetical genetic regulatory network. A genetic regulatory network is a directed graph with many loops. This figure shows one with 13 nodes, and the edges represent regulatory relationships. For example, n_6 is regulated by n_8, n_9, n_{11} and itself. Note that genome-scale GRNs are much larger than the network shown in this figure. For example, *Saccharomyces cerevisiae* is commonly used as a model organism and has more than 6000 genes.

is turned on (or not) at any particular point in time, and time series observations of these expression levels are used for the network structural inference. More detailed information on GRNs and their network or functional characteristics can be found in literature surveys like [7] and [10].

2.3. *Previous Research on Efficiency for Bayesian Networks*

Likelihood calculation is a fundamental task for using BNs and is necessary for structural inference, but it is NP-hard. Thus, efficient likelihood calculation has been the focus of much BN research, and algorithms like Pearl's [27] message passing algorithm are widely used. Other efficiency research has also considered structural restrictions, parameter approximations and dimensional reduction, and these techniques may be combined with the research presented in this article to make structural inference even more tractable.

2.3.1. *Structural Restrictions*

In a BN, the size of a conditional probability distribution grows exponentially as the node's in-degree grows. One efficiency technique is to limit the maximum in-degree of a single node [19]. However, because the in-degree of a GRN follows a power law [10], restricting the structure may affect the accuracy of the inferred structure.

2.3.2. *Parameter Approximation*

Classically, θ in a BN is a set of conditional probability distributions. Another way of addressing the time complexity of likelihood calculation is by approximating these distributions in some way.

For example, [14] used *noisy-or*, which models the impact of each parent node as independent of the other parents. Such an approximation has only linearly as many parameters as there are parents, but it restricts the functions which can be represented, potentially including many of the non-linearities that are key to GRNs and other networks. Hence, since GRNs can be highly non-linear, using noisy-or may limit the value of any GRN structural inference.

Langseth and Nielsen [23] use *parameter tying*, grouping nodes together using object-oriented programming principles and prior knowledge. Parameter tying has also been used for other problems, including speech recognition (e.g., [42] and [43]), handwriting recognition [28], and in bioinformatics (e.g., [6,20] and [26]).

2.3.3. *Dimensional Reduction*

Dimensional reduction by clustering nodes together has also been used [9,15,30,31,36]. Because co-expressed nodes in a GRN can be co-dependent, clustering may usefully reduce the dimensionality without introducing too much false information or significantly inflating the maximum in-degree. On the other hand, it may also limit the accuracy of the inferred network in the same way that structural restrictions and parameter approximation may.

3. Dense SEM

SEM is a structural inference algorithm for BNs that works by calculating the parameters and the score of the current structure, estimating the score of all structures that can be reached after applying one edge-change operation and then applying the operation which improves the score by the most. It iterates this process until no operation is estimated to improve the score, terminating and returning the highest scoring network found.

Dense SEM is a modified version of this algorithm that can apply more than one operation per iteration, and it is called Dense SEM because the applied operations are more densely concentrated amongst fewer iterations (illustrated conceptually in Figure 4). Each iteration of SEM requires time-expensive calculations. DSEM can use each set of calculations to make multiple changes and this is why it can be faster than SEM. While applying multiple operations is relatively simple at a conceptual level, and hinted at in [21], actual development and implementation turns out to be quite subtle. Figure 5 presents the DSEM algorithm in detail.

In each iteration of DSEM, up to c operations may be applied, and these are selected one after the other using up to c inner iterations. In each one of these, the best candidate operation that can still be applied is o_{best}. So long as it is estimated to

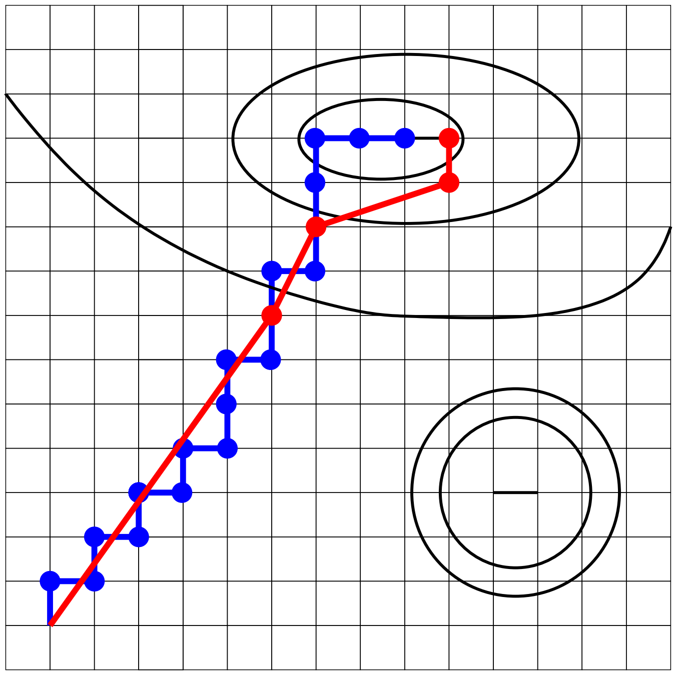

Fig. 4. A visual conceptualisation of the difference between SEM and DSEM. The network search space is represented by the grid, where points represent structures, and lines amongst them possible operations. The contours show the relative quality of each structure. A local maxima spanning three structures (e.g., an equivalence class) exists near the top-right corner of the figure, and another local maxima is located in the bottom right hand corner of the figure. The search path for SEM (blue, more dots) and DSEM (red, fewer dots) start at the same place in the lower left. The points in the space where SEM and DSEM infer θ and S, compute the ESS (expected sufficient statistics) and estimate the ΔS are denoted by small circles on their path.

improve the score, then this is applied, and then removed from the set of candidate operations.

However, additionally, the estimated ΔS (change in score) of some other operations will also be affected by the application of o_{best}. In general, o_{best} and any other operation are one of the following types of operations:

- Strongly dependent
- Weakly dependent
- Independent

Subsection 3.1 describes strongly dependent operations which greatly affect each others' ΔS, and Subsection 3.2 describes weakly dependent and independent operations. Subsection 3.1 also describes how the set of operations which are strongly dependent on o_{best}, $O_{invalid}$, are identified and removed from O.

```
begin
    repeat
        Infer $\theta$ and calculate the score $S$ for current structure $G_{cur}$
        Identify $O$, the set of valid operations from $G_{cur}$
        Calculate the ESS (Expected Sufficient Statistics) for each $o \in O$
        Use the ESS to estimate $\Delta S(o)$ for each $o \in O$
        for no more than $c$ operations do
            $o_{best} = \arg\max_{o \in O} \Delta S(o)$
            if $\Delta S(o_{best}) > \epsilon$ then
                $G_{cur} = o_{best}(G_{cur})$
                $O = O - o_{best} - O_{invalid}(o_{best})$
            else
                // Best operation decreases the score

                break
            end
        end
    until $G_{cur}$ is at a local maxima
end
```

Fig. 5. The DSEM algorithm, where $\Delta S(o)$ is the estimated change in score from operation o, $o(G)$ applies o to structure G and $O_{invalid}(o)$ are the operations in O made invalid by the application of o.

3.1. *Strongly Dependent Operations*

Two operations (o_1 and o_2) are strongly dependent if both change the parent set of the same node, n_i. This is because BNs are scored using a penalised likelihood function and the likelihood of the structure as a whole is just the product of the likelihood of each node, where each node's likelihood is conditional on its parents. Thus, if two operations both affect a node's parents, then both will affect its likelihood and each other's ΔS.

For example, o_1 ("add $n_j \rightarrow n_i$") and o_2 ("add $n_k \rightarrow n_i$") are strongly dependent because both are parents of n_i and the likelihood of n_i changes non-linearly as parents are added and removed, i.e., $\Delta S(o_1, o_2) \neq \Delta S(o_1) + \Delta S(o_2)$.

One approach to this problem is to calculate $\Delta S(o_x, o_y)$ for all possible 2nd-order operations that are strongly dependent, $\langle o_x, o_y \rangle \in O$. However, if each pair of operations affecting each of the N nodes were considered (N^2 pairs of operations per node), then $O(N \times N^2) = O(N^3)$ possible 2nd-order operations must be considered and their ΔS calculated, as well as ΔS for each of the $O(N^2)$ 1st-order operations (see Appendix Appendix A). Similarly, if 3rd-order operations were considered, then another $O(N^4)$ operations must be considered, and if all possible higher order operations were considered, then ΔS would need to be estimated for $O(N^N)$ operations, which is intractable.

Additionally, note that the operations' ΔS are only estimated and may sometimes be wrong to a greater or lesser extent. Thus, attempting to estimate ΔS for higher order operations whose component operations are strongly dependent on each other may be vulnerable to multiplicatively growing estimation errors. Such errors could lead to noisy and inefficient structural inference.

Therefore, for conceptual and practical reasons, DSEM does not allow directly dependent operations to both be performed in the same iteration, and all operations which are strongly dependent on o_{best} are included in $O_{invalid}$ and removed from consideration for that iteration. These operations are also easily detected because it is clear from the operation which nodes' parents are affected. Because these temporarily invalidated operations are still possible in the next iteration and will be considered then with their ΔS re-calculated, this restriction does not restrict the search.

3.2. *Weakly Dependent Operations*

Most operations whose ΔS are not strongly dependent on the application or non-application of o_{best}'s will be independent of o_{best}. However, some operations may also be weakly dependent, and this is due to the structure, interactions amongst the operations and the patterns of missing (hidden) data. For example, consider Figure 6.

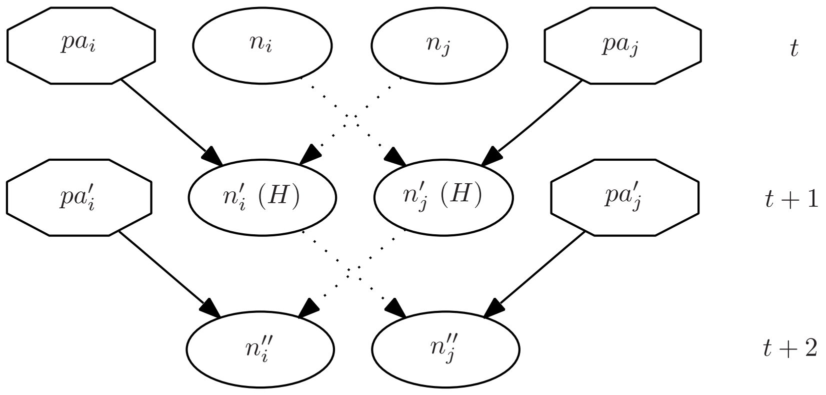

Fig. 6. An example of two weakly dependent operations in a DBN. The two operations which are weakly dependent are "add $n_i \rightarrow n_j$" and "add $n_j \rightarrow n_i$" (dotted lines), and this figure shows the state of n_i, n_j and their parents at times t, $t+1$ and $t+2$. It does not matter in this example if pa_i and pa_j are disjoint, but (for the sake of the example) this figure assumes that $n_i \notin pa_j$ and $n_j \notin pa_i$ before any operations are applied to the current structure. Note that n_i and n_j are hidden at time $t+1$. The two operations are weakly dependent because of this.

In this figure, the two operations shown are:

- Add an edge from n_i to n_j
- Add an edge from n_j to n_i

And they are weakly dependent because of the pattern of hidden data, even though the two operations do not directly affect the same node's parents. Consider: If the edge from n_i to n_j is added then it will change the uncertain distribution over n_j at $t+1$, which would in turn affect the likelihood of n_i at time $t+2$ if $n_j \rightarrow n_i$ were also added, and thus affecting ΔS for $n_j \rightarrow n_i$. Similarly, if the edge from n_j to n_i were added then it would change the uncertain distribution over n_i at $t+1$, which would in turn affect the likelihood of n_j at time $t+2$ if $n_i \rightarrow n_j$ were also added, and thus affecting ΔS for $n_i \rightarrow n_j$.

Because weakly dependent operations are rare, complex to detect and because exploratory experiments show that assuming independence in their case does not have a detectable impact on the number of iterations necessary for structural inference or the inferred networks' quality, DSEM assumes independent ΔS amongst all operations which are not strongly dependent.

Furthermore, even in the worst case, incorrectly assuming independence when two operations are weakly dependent could only alter the local maxima that DSEM identified. This is because DSEM will continue to iterate until no individual operation in an iteration has an estimated score that is better than the current structure's score.

4. Parameterising DSEM and comparing it to SEM

This section describes experiments used to help identify the best parameterisation of DSEM and to compare its efficiency and the inferred network quality with SEM. Subsection 4.1 describes the experimental configuration, Subsection 4.2 presents the results and Subsection 4.3 summarises the relative efficiency of DSEM and SEM.

4.1. *Experimental Configuration*

10 different parameterisations of DSEM were used and evaluated in these experiments, and their performance was evaluated using 5 different 100-gene networks (50 structural inference experiments in total, requiring more than 500 hours of CPU time).

Before execution of these experiments, another 80 exploratory inferences with different training set sizes, noise and missing data configurations on other networks were also used to help design this experimental configuration. These exploratory experiments required a similar quantity of CPU time and are referred to where necessary in this article, but are not fully detailed here to limit the length.

4.1.1. *Networks and data*

Full knowledge of the underlying network is needed to evaluate novel approaches for structural inference, and very few biological GRNs are well known[a]. Because of this, the 5 different GRNs used in these experiments were simulated using GreenSim [11]. GreenSim is a modern GRN simulator that can automatically generate a biologically plausible structure, and it simulates gene expression levels in a continuous fashion using non-linear regulatory relationships.

Although the quality of the inferred network and time required is strongly dependent on the quantity and quality of training data, the exploratory experiments also showed that the same network was inferred in all cases using any of the 10 different parameterisations, regardless of the quantity of training data, noise and hidden data values. Based on the DSEM and SEM algorithms, this similar variation is to be expected.

Because the purpose of these experiments is to compare the different parameterisations, and not evaluate their absolute performance, it also allows a much simpler and concise experimental configuration. In particular, it means that only a single training data set needed to be generated for each network. Each of these training data sets consisted of 25 independently generated time series, and each time series consisted of 60 consecutive observations with no missing data or noise, (i.e., 1500 observations per gene). Using the same training data set with each parameterisation also helps ensure the results are comparable across parameterisations.

Bayesian networks with discrete nodes were also used to help make the inferred networks more comprehensible, and the continuous gene expression levels are discretised into four buckets [39]. Empirical tests were also performed on the simulated networks and time series data to ensure that they exhibited reasonable characteristics for inference (e.g., no genes permanently at a maximum or nil expression level).

4.1.2. *Parameterisations*

DSEM can be parameterised by specifying the maximum number of operations allowed per iteration and Table 1 shows the 10 parameterisations that were considered. Each parameterisation may potentially be an exponentially decaying maximum number of operations per iteration according to Equation 1, where $c^{max}(i)$ is the maximum number of operations per iteration allowed in the i'th iteration, δ_{init} is the number of operations at the end of the first SEM iteration, and τ is the exponential decay rate of δ_{init}.

$$c^{max}(i) = \lceil \delta_{init} \times \tau^{i-1} \rceil \tag{1}$$

Using an exponentially decaying maximum number of operations (as in param-

[a]Indeed, this is one reason it is such a vibrant and central networks topic.

eterisations 8–10) may force DSEM to take smaller and smaller steps as it gets closer and closer to a maxima, and the utility of this forced reduction in step size is analysed along with the other results in the next subsection. Regardless of the parameterisation used, inference of each network was run for as many iterations as needed until no possible operation had a positive ΔS.

NB: If no more than 1 change is allowed per iteration then DSEM is SEM. This is the left-most parameterisation in Table 1 and was used to compare the time efficiency and inferred network quality of DSEM with SEM.

Parameterisation:	1 (SEM)	2	3	4	5	6	7	8	9	10
δ_{init}:	1	3	5	7	11	20	100	11	11	11
τ:	1	1	1	1	1	1	1	0.92	0.9	0.85

The *Akaike information criterion* [1,12] is appropriate to be used for inferring GRNs [38] and was used in this article. Exploratory experiments also found that an empty initial structure without any edges led to inference of better networks, so this initial structure was used for all presented experiments in this article.

4.2. *Experimental Results*

As Figure 7 shows, the fastest parameterisation for DSEM was parameterisation 7, with $\delta_{init} = 100$ and $\tau = 1$ (i.e., $c = N$ for all iterations). Because the same network structure was inferred for each network regardless of the parameterisation that was used (and similarly, in the other exploratory experiments), these results also suggest that the best parameterisation for DSEM in other situations is $c = \delta_{init} = N$ and $\tau = 1$.

Conversely, SEM (parameterisation 1) was the slowest parameterisation, taking between 167 and 200 iterations on the 5 networks considered. Since parameterisation 7 took only 4 iterations on each of the networks and the time taken per iteration does not vary with the number of operations applied, this means that DSEM was 41-50 times faster than SEM on these networks.

Further analysis of these experiments also supports in two ways the earlier conceptual analysis for not considering the interaction of weakly dependent operations.

Firstly, the variance in the results for each parameterisation are due to the same number of operations for each network being applied across more or fewer iterations. However, if weak dependence was significant, then interactions amongst weakly dependent operations would mean that extra iterations would be needed in the parameterisations with greater average c. In that case, the relative variance across networks would be greater, not the same as is shown in Figure 7.

Secondly, consider Figure 8. This figure shows the changes made per iteration by parameterisation 5 and 8, where 8 enforced a declining number of maximum changes per iteration. As this (and other similar figures, not shown here to limit the length)

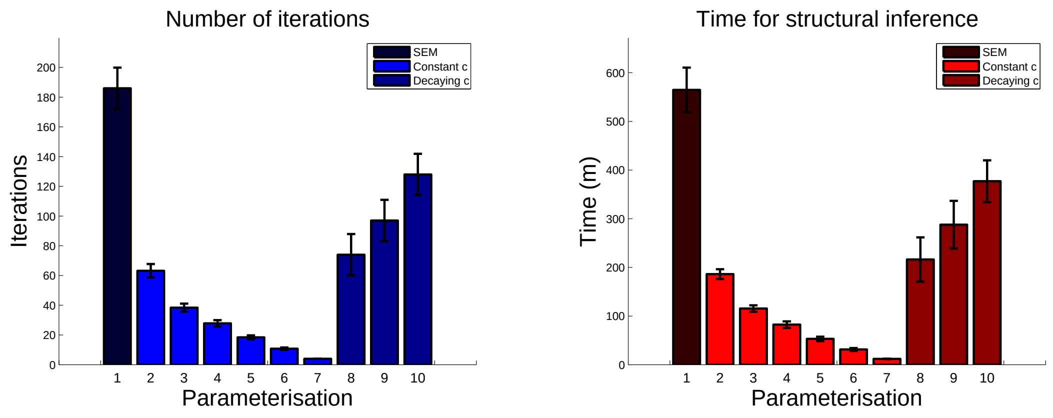

Fig. 7. The mean number of iterations (LHS) and time (RHS) for each configuration (numbered as in Table 1). Error bars are one standard deviation of the iterations or time necessary from network-to-network and show variation amongst the networks used.

make clear, artificially limiting the number of operations to less than N leads to poorer performance. However if weak dependency had a greater impact, then there would be a more gradual decrease in the number of operations per iteration.

4.3. *Relative Efficiency Summary*

Using the best identified parameterisation, the experiments in Subsection 4.2 showed that DSEM was up to 50 times faster than SEM on the five 100-gene GRNs being considered (see Figure 9).

In the best case situation, DSEM may be up to N times faster than SEM, where N is the size of the network which must be inferred. On the other hand, in the worst case, DSEM may be very slightly slower due to the additional overhead in the algorithm (see Figure 5).

Unfortunately, it is difficult to calculate the relative speed advantage of DSEM in the average case because the number of iterations is what drives the advantage, and this is determined by a complex range of factors that are extremely difficult to quantify, including the intrinsic likelihood of the data and its quantity, the pattern of missing values in the data, and the (presumably unknown) dependencies amongst the nodes in the graph. Fogelberg [8] presents some initial analysis, and thorough analysis of this question is one area for further research.

5. Parellelising Structural Inference

Implementing and evaluating parallelised DSEM (and SEM) is the second novel contribution that this article makes. Like DSEM, the parallelisation greatly increases the size, range and variety of networks which can be structurally inferred.

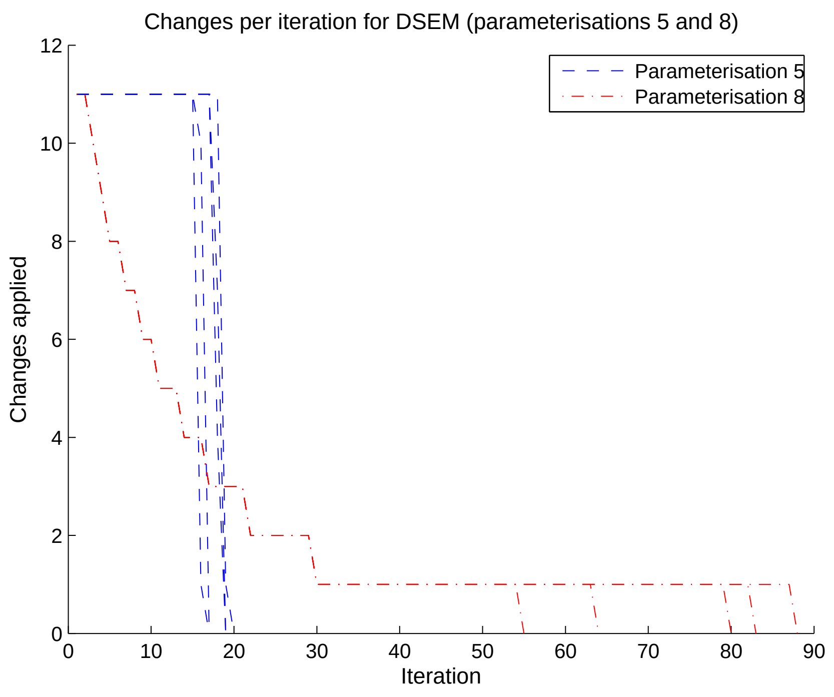

Fig. 8. The number of operations applied by DSEM with parameterisation 5 ($\delta_{init} = 11$, $\tau = 1$; i.e., no decay) and parameterisation 8 ($\delta_{init} = 11$, $\tau = 0.92$; i.e., decay). Each of the ten lines represents the inference of one network with one parameterisation (5 networks, once for each parameterisation).

5.1. *Parallel Implementation*

Parallelisation of the DSEM codebase was constrained by intrinsic limitations of distributing computing. In particular, disk IO is not possible without the overhead of using a distributed file system and the inputs and outputs to each loop iteration must be independent of each other, and this can drive some of the changes needed.

For example, rather than accumulating a total over several loop iterations, the subtotals from each individual loop iteration must be separately stored into a distributed data structure and then reduced after the parallel loop has completed.

With these factors in mind, the following three stages of DSEM were parallelised:

- Likelihood calculations (for calculating S) and θ inference
- ESS computation
- Estimating ΔS for each $o \in O$

Each of these three steps in the DSEM algorithm is very time consuming, with loops that have many iterations and where each individual iteration can be long. Focusing on loops like this maximise the benefits of parallelisation.

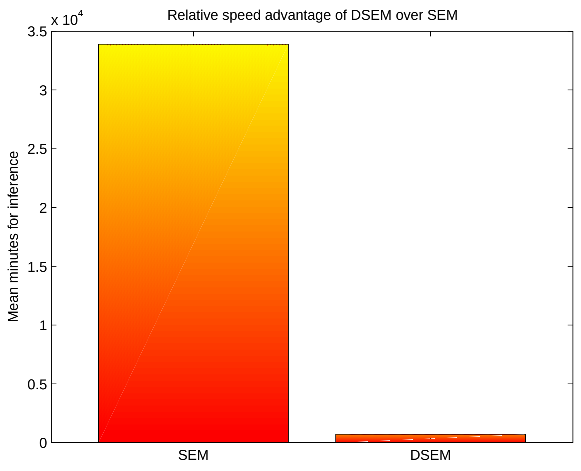

Fig. 9. The time required for inference using SEM and DSEM. Time is in minutes.

All code was executed using the Oxford e-Research Centre Windows Compute Cluster (OeRC WCC) parallel Matlab cluster. In this cluster, the head node is a quadruple dual-core 2.66GHz 64 bit Intel Xeon with 32GB of RAM, and each of the 16 worker machines is a Dell PE 1950 with 2 dual-core 2.66GHz 64 bit processors and 8GB RAM. A pool of 12 worker cores (automatically chosen by the head node) was opened for the experiments in this section. Note that, because others were using other cores in the cluster simultaneously, some variation in the time taken can be observed.

5.2. *Experimental Configuration and Results*

A conceptual analysis of parallelisation suggests that its benefit will be approximately linear in the number of worker cores the inference is divided amongst. These experiments considered 6 levels of parallelisation: 10 additional worker cores, 8, 6, 4, 2 and 0 (i.e., no parallelisation).

In total 10 experiments for parallelisation were conducted, using five 100-gene GRNs (like in Subsection 4) and 2 different training data sets for each. Each training data set consisted of 25 independently generated time series with 30 consecutive observations. Because the objective of these experiments was to evaluate the relative impact on the time required by different levels of parallelisation, it means that varying the network size, quantity of training data or level of noise and missing data is not insightful and no such experiments are reported in this article.

As expected, the same search path was followed for each of the 10 experiments and the same network was inferred, regardless of the degree of parallelisation. Figure 10 presents the mean number of minutes required for structural inference.

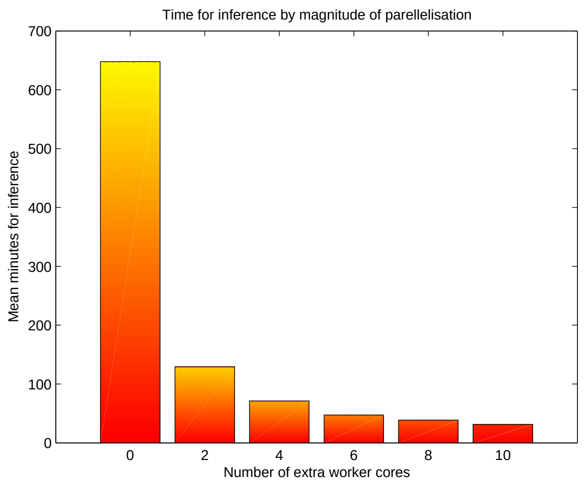

Fig. 10. Inference time with varying degree of parallelisation. The number of extra worker cores (10, 8, 6, 4, 2 or 0; i.e., no parallelisation) are shown as bars along the x-axis, and the mean number of minutes necessary to infer the structure of each network is shown on the y-axis. Results are noisy due to the cluster also being used by others while the experiments were being conducted.

Firstly, this figure shows that parallelisation is essential to the practical and tractable structural inference of large networks. This is because, even when using DSEM, the mean time necessary to infer one 100-gene network without any parallelisation is more than 10 hours. Further, this figure grows super-linearly as larger networks [8] are considered.

Secondly, the analysis of the time needed for inference using between 2 and 10 worker cores shows the expected linear speed up as more parallelisation is used. However, inference without any parallelisation was much slower than this linear analysis would suggest. The experiment was re-run several times using an alternative master computer, and the same pattern was observed in each case. Since structural inference is also very memory intensive, we hypothesise that this may be a memory and paging issue, and a further motivation for parallelisation.

6. Discussion and Conclusions

Network studies is an increasingly important research topic that spans a number of fields. In general, network structural inference is an extremely difficult and very time consuming machine learning problem, but it is also a very important one, including for large biological networks like genome-size GRNs. This article has presented DSEM, a novel extension of the SEM algorithm that increases the efficiency of structural inference by using the time-expensive calculations required in each SEM iteration more efficiently. The article has also combined DSEM with parallelisation and evaluated their impact, individually and combined, and this impact is summarised in Figure 11.

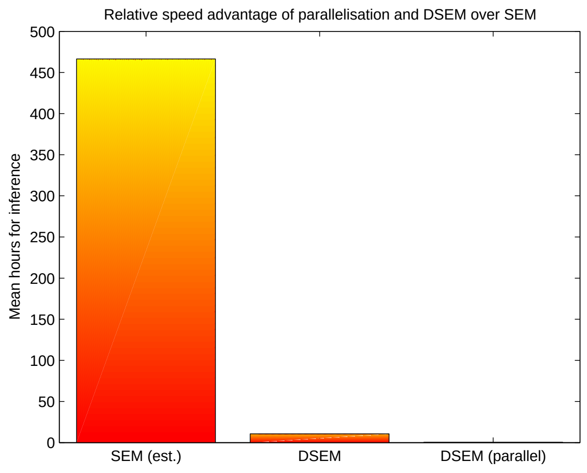

Fig. 11. The relative speed advantage of DSEM and parallelisation over SEM. DSEM (parallel) results are using 10 cores for parallelisation. Y-axis results are in hours. Inference time for SEM (without parallelisation) is estimated by calculating the mean time per iteration and multiplying by the number of operations applied. Graph calculated using data from experiments in Section 5 and error bars were insignificant and are not shown.

Both contributions are publicly available online (`http://syntilect.com/cgf/pubs:software`), and further research and analysis of the DSEM and its parallelisation combined with other techniques may prove fruitful. In particular, more thorough analysis of Fogelberg's [8] average case efficiency of DSEM versus SEM would be valuable. Further validation of the DSEM algorithm in other problem domains would also be useful, as would be its combination with the three other structural inference efficiency levers presented in Subsection 2.3.

7. Acknowledgements

Christopher Fogelberg gratefully acknowledges the financial and logistical support of the Commonwealth Scholarship Commission in the United Kingdom and the Oxford-Man Institute of Quantitative Finance, without whose support this research would not have been possible. We would also like to thank the Oxford e-Research Centre (OeRC) for access to their Microsoft Cluster on which our experiments were run.

Appendix A. Bounding the Number of Operations

The number of possible operations per iteration in a DBN is polynomially bound by the size of the network.

Consider: A DBN has N nodes, and in some structure G, each $n_i \in N$ is the sink for e_i^{in} edges and the source of e_i^{out} edges. SEM and DSEM use three types of operation, and they are:

- Edge addition
- Edge deletion
- Edge reversal

Therefore, for some $n_i \in N$, the number of possible operations of each type are as follows:

- Possible edge additions: $o_i^{add} = N - e_i^{out}$
- Possible edge deletions: $o_i^{del} = e_i^{out}$
- Possible edge reversals: $o_i^{rev} = e_i^{out} - f_i$

where $f_i = |n_j \in N : \exists n_j \to n_i|$ and is the size of the set of all nodes with edges to n_i that n_i also has edges to, and its inclusion is justified because reversing an edge between n_i and n_j, if there is already an edge from n_j to n_i, is akin to deleting the edge between n_i and n_j and, thus, is already counted in the set of possible edge deletions.

Therefore, since $o_i^{add} + o_i^{del} = N$ and $o_i^{rev} \le o_i^{del}$, there may be no more than:

$$\sum_{x \in \{add, del, rev\}} o_i^x \le 2N \tag{A.1}$$

operations for all $n_i \in N$. Hence the total number of possible operations per iteration in a DBN with N nodes is upper bound by $N \times 2N = O(N^2)$.

References

1. H. Akaike. Information theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Csaki, editors, *2nd International Symposium on Information Theory*, pages 267–281, 1973.
2. Y. Cao, P. P. Wang, and A. Tokuta. Reverse engineering of NK boolean network and its extensions — fuzzy logic network (FLN). *New Mathematics and Natural Computation*, 3(1):68–87, 2007.
3. X. Chen, G. Anantha, and X. Wang. An effective structure learning method for constructing gene networks. *Bioinformatics*, 22(11):1367–1374, 2006.
4. D. M. Chickering. Learning Bayesian networks is NP-Complete. In D. Fisher and H. J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, 1996.
5. D. M. Chickering and T. Paek. Personalizing influence diagrams: applying online learning strategies to dialogue management. *User Model. User-Adapt. Interact.*, 17(1-2):71–91, 2007.
6. R. D. Dowell and S. R. Eddy. Efficient pairwise RNA structure prediction and alignment using sequence alignment constraints. *BMC Bioinformatics*, 7:400, 2006.
7. M. E. Driscoll and T. S. Gardner. Identification and control of gene networks in living organisms via supervised and unsupervised learning. *Journal of Process Control*, 16(3):303–311, March 2006.

8. C. Fogelberg. *The Structural Inference of Large Regulatory Networks*. PhD thesis, Computing Laboratory, Oxford University, September 2010.
9. C. Fogelberg and V. Palade. Evaluating clustering algorithms for genetic regulatory network structural inference. In *Research and Development in Intelligent Systems XXVI: Proceedings of AI-2009, The Twenty-Ninth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence (AI2009)*, volume 29, pages 137–150, Cambridge, UK, December 2009. Springer-Verlag.
10. C. Fogelberg and V. Palade. Genetic regulatory networks: A review and a roadmap. In A. Abraham, A.-E. Hassanien, A. Vasilakos, W. Pedrycz, F. Herrera, P. Siarry, A. de Carvalho, and A. P. Engelbrecht, editors, *Foundations of Computational Intelligence*, chapter 1:1. Springer-Verlag, 2009.
11. C. Fogelberg and V. Palade. Greensim: A network simulator for comprehensively validating and evaluating new machine learning techniques for network structural inference. In *IEEE ICTAI2010*, volume 2, pages 225–230, Arras, France, October 2010. IEEE Computer Society.
12. M. Forster and E. Sober. How to tell when simpler, more unified, or less ad hoc theories will provide more accurate predictions. *The British Journal for the Philosophy of Science*, 45(1):1–35, 1994.
13. N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In D. H. Fisher, editor, *ICML*, pages 125–133. Morgan Kaufmann, 1997.
14. N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, volume 14, pages 139–147, San Francisco, CA, 1998. Morgan Kaufmann.
15. A. M. Gholami and K. Fellenberg. Cross-species common regulatory network inference without requirement for prior gene affiliation. *Bioinformatics*, 26(8):1082–1090, 2010.
16. R. N. Gutenkunst, J. J. Waterfall, F. P. Casey, K. S. Brown, C. R. Myers, and J. P. Sethna. Universally sloppy parameter sensitivities in systems biology models. *PLoS Compututational Biology*, 3(10):e189, October 2007.
17. B. Hayete, T. S. Gardner, and J. J. Collins. Size matters: Network inference tackles the genome scale. *Molecular Systems Biology*, 3(77):1–3, February 2007.
18. D. Heckerman. A tutorial on learning with Bayesian networks. Technical report, Microsoft Research, Redmond, Washington, 1995.
19. Z. Huang, J. Li, H. Su, G. S. Watts, and H. Chen. Large-scale regulatory network analysis from microarray data: modified Bayesian network learning and association rule mining. *Decision Support Systems*, 43(4):1207–1225, 2007.
20. R. Jurgelenaite and P. J. F. Lucas. Exploiting causal independence in large Bayesian networks. *Knowledge-Based Systems*, 18(4–5):153–162, 2005.
21. D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.
22. H. Lähdesmäki, I. Shmulevich, and O. Yli-Harja. On learning gene regulatory networks under the Boolean network model. *Machine Learning*, 52(1–2):147–167, 2003.
23. H. Langseth and T. D. Nielsen. Fusion of domain knowledge with data for structural learning in object oriented domains. *Journal of Machine Learning Research*, 4:339–368, 2003.
24. S. Mani, P. Spirtes, and G. Cooper. A theoretical study of Y structures for causal discovery. In *Proceedings of the Twenty-Second Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*, pages 314–323, Arlington, Virginia, 2006. AUAI Press.
25. O. J. Mengshoel, S. Poll, and T. Kurtoglu. Developing large-scale Bayesian networks

by composition: Fault diagnosis of electrical power systems in aircraft and spacecraft. In *Proc. of the IJCAI-09 Workshop on Self-⋆ and Autonomous Systems (SAS): Reasoning and Integration Challenges*, 2009.

26. S. Natarajan, I. Ong, D. Haight, D. Page, and V. S. Costa. Modeling temporal biomedical data by SRL. In J. Ramon, F. Costa, C. Costa, and J. Kok, editors, *ECML08 Workshop on Statistical and Relational Learning in Bioinformatics (StReBio)*, Antwerp, Belgium, September 2008.
27. J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, September 1988.
28. F. Pernkopf and M. Wohlmayr. On discriminative parameter learning of Bayesian network classifiers. In W. L. Buntine, M. Grobelnik, D. Mladenic, and J. Shawe-Taylor, editors, *ECML/PKDD (2)*, volume 5782 of *Lecture Notes in Computer Science*, pages 221–237. Springer, 2009.
29. M. Pritsker, Y.-C. Liu, M. A. Beer, and S. Tavazoie. Whole-genome discovery of transcription factor binding sites by network-level conservation. *Genome Research*, 14(1):99–108, January 2004.
30. D. Reiss, N. Baliga, and R. Bonneau. Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks. *BMC Bioinformatics*, 7(1):280, 2006.
31. S. Roy, S. Plis, M. Werner-Washburne, and T. Lane. Scalable learning of large networks. *Systems Biology, IET*, 3(5):404–413, September 2009.
32. T. Schlitt and A. Brazma. Modelling gene networks at different organisational levels. *FEBS Letters*, 579:1859–1866, March 2005.
33. T. Schlitt and A. Brazma. Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, 8 Suppl 6, 2007.
34. M. M. Sondhi, J. Benesty, and Y. Huang, editors. *Springer Handbook of Speech Processing*. Springer-Verlag, January 2010.
35. P. Spirtes. Introduction to causal inference. *Journal of Machine Learning Research*, 11:1643–1662, 2010.
36. M. Stetter, G. Deco, and M. Dejori. Large-scale computational modeling of genetic regulatory networks. *Artificial Intelligence Review*, 20(1–2):75–93, October 2003.
37. J. Tegner, M. K. Yeung, J. Hasty, and J. J. Collins. Reverse engineering gene networks: integrating genetic perturbations with dynamical modeling. *Proceedings of the National Academy of Sciences, USA*, 100(10):5944–5949, May 2003.
38. J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink, and E. D. Jarvis. Using Bayesian network inference algorithms to recover molecular genetic regulatory networks. In *International Conference on Systems Biology (ICSB02)*, December 2002.
39. J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink, and E. D. Jarvis. Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20(18):3594–3603, 2004.
40. Y. Zhang, Z. Deng, H. Jiang, and P. Jia. Dynamic Bayesian network (DBN) with structure expectation maximization (SEM) for modeling of gene network from time series gene expression data. In H. R. Arabnia and H. Valafar, editors, *BIOCOMP*, pages 41–47. CSREA Press, 2006.
41. A. Zollmann, A. Venugopal, F. J. Och, and J. M. Ponte. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In D. Scott and H. Uszkoreit, editors, *COLING*, pages 1145–1152, 2008.
42. G. G. Zweig. *Speech Recognition with Dynamic Bayesian Networks*. PhD thesis, Computer Science, University of California Berkeley, April/May 1998.
43. G. G. Zweig. Bayesian network structures and inference techniques for automatic

speech recognition. *Computer Speech and Language*, 17:173–193, 2003.