6.825 Project 3
Bayes' Net Learning: Parameter Estimation and Structure Search

Eric Mumpower
Dan RK Ports

December 8, 2005

Contents

1 Overview

In this project, we explore various algorithms and issues pertaining to Bayes' Net learning. Structure search and a pair of search-scoring metrics are discussed, and we discuss the difference in results achieved by these scoring metrics. Two different search algorithms were implemented, and a variety of experiments were run to compare and contrast the performance and results provided by these algorithms.

2 Parameter Estimation and Scoring

2.1 Parameter Estimation

We began by implementing maximum likelihood estimation to determine CPT parameters, for a given network structure. We initially did so by computing each entry in the CPT as the observed number of samples satisfying both the appropriate assignment to the node and its parents, divided by the number of samples satisfying the appropriate assignment to the parents. We later refined this by adding the Laplacian correction to provide a prior estimation of the probability of node values that depend on parent assignments that never occur in the data.

The Bayes'-net parameters for the provided dataset B, given the provided network graph, were estimated by our Parameter Estimation implementation to be:

Estimated CPT for **A**	
P(A=0)	**P(A=1)**
0.4932	0.5068

Estimated CPT for **B**		
A	**P(B=0)**	**P(B=1)**
0	0.273317	0.726683
1	0.271113	0.728887

Estimated CPT for **C**		
A	**P(C=0)**	**P(C=1)**
0	0.370235	0.629765
1	0.356354	0.643646

Estimated CPT for **D**			
B	**C**	**P(D=0)**	**P(D=1)**
0	**0**	0.159265	0.840735
0	**1**	0.163842	0.836158
1	**0**	0.196045	0.803955
1	**1**	0.221729	0.778271

Estimated CPT for **E**	
P(E=0)	**P(E=1)**
0.686	0.314

Estimated CPT for **F**		
E	**P(F=0)**	**P(F=1)**
0	0.546064	0.453936
1	0.535669	0.464331

Estimated CPT for **G**		
F	**P(G=0)**	**P(G=1)**
0	0.277819	0.722181
1	0.265529	0.734471

Estimated CPT for **H**		
G	**P(H=0)**	**P(H=1)**
0	0.155033	0.844967
1	0.149766	0.850234

2.2 Structure Scoring Metrics

Bayes' Net learning involves the comparison of various possible Bayes' Network graph structures. As such, a scoring metric is required; we experimented with both the Bayesian Information Criterion (BIC) and the Akaike Information Criterion (AIC). These scoring mechanisms are discussed in the following sections.

2.2.1 The Bayesian Information Criterion

We used the Bayesian Information Criterion (BIC) as a basis for evaluating the quality of network structures. The BIC is defined as follows:

$$\mathsf{BIC}(G : D) = \ell\left(\hat{\theta}_G : D\right) - \frac{\log_2 M}{2}\mathrm{Dim}[G] \tag{1}$$

The BIC score consists of the log-likelihood of the data given the MLE estimation, reduced by a structural penalty of $\frac{\log_2 M}{2}\mathrm{Dim}[G]$, where $\mathrm{Dim}[G]$ is the dimension of the network. The network dimension is the number of parameters required to specify the CPTs in the network: the sum over all nodes of the number of possible assignments for the parents (number of rows in the CPT) multiplied by the number of possible assignments minus 1 (number of columns in the CPT). We subtract 1 from the number of columns in the CPT because we use the observation that the last column can be derived from the other ones. The effect of this structure penalty is to prefer graphs with less connections, since the resulting networks are more efficient to work with and avoid overfitting the data.

Here are the components of the BIC score we calcuated for our parameter estimation of dataset "B" given the provided network graph:

$\log_2$-Likelihood	$l(\hat{\theta}_G : D)$	-34285.24
Structural Penalty	$\frac{\log_2 M}{2}\text{Dim}[G]$	98.30
BIC Score	$\hat{S}(G : D)$	-34383.54

In the above, the magnitude of the log-likelihood of our data (given the graph) is much larger than was computed for dataset "A", but this is due to the greater number of data points. This difference in magnitude may be plausibly explained as follows:

If one makes the (unreasonable) assumption that all 5000 samples in dataset "B" were equally likely, then they each have probability $2^{-34285/5000} \approx 0.0086$. Given that dataset "B" has 8 variables, there are $2^8 = 256$ possible combinations of node values. If every possible outcome were equally likely, they would each have probability $2^{-8} \approx 0.0039$, which is not so different from the "average" likelihood achieved by our fit to the data. Thus, it's reasonable that such a large log-likelihood does not necessarily indicate a poor fit.

However, as we will see in Section 3.3.1, this is a relatively poor score. In particular, none of our Data-B search attempts achieved a BIC score worse than -31653. Additionally, the BIC scores we achieved through structure search all had standard deviations smaller than 19, suggesting that a score difference of 2600 points is a rather significant difference in fit for this data.

2.2.2 The Akaike Information Criterion

We chose to also implement the Akaike Information Criterion[1] (AIC) for comparison with BIC. The AIC metric seemed appropriate for our purposes, as it has been widely cited in comparison with the BIC metric for use in scoring Bayes networks.[2]

The AIC is calculated very similarly to the BIC:

$$\text{AIC}(G : D) = \ell\left(\hat{\theta}_G : D\right) - \text{Dim}[G] \tag{2}$$

That is, the AIC score of a particular graph structure is the log likelihood of the data under the current estimation of parameters for the graph, minus a structure penalty given simply by the dimension of the graph. This is identical to the BIC score, except that the AIC structure penalty does not have a factor of $\frac{\log_2 M}{2}$. Hence, the AIC score does not penalize the complexity of the network structure as much as the BIC score, particularly for very large data sets.

As a result, we expect structure search based on the AIC metric to produce networks with more dependencies than if the BIC network is used. Though the resulting networks may produce higher likelihood of the data, they do so at the cost of a more complex network: this is undesirable both because the more complex network requires a longer description and leads to increased runtime for inference algorithms, and because it can overfit the data. As we describe in Section 2.3, we did indeed observe more complex networks when performing structure search using the AIC metric.

[1] Akaike, H. (1974). A new look at statistical model identification. Transaction on Automatic Control 19,716-723.

[2] See, e.g. `http://www.cs.cmu.edu/~awm/15781/slides/Param_Struct_Learning05v1.pdf`

2.3 Structure Scoring Results and Comparison

Overall, we observed that AIC tends to yield more complex graphs than those found when using BIC scoring. Qualitatively, we observed that the graphs resulting from AIC are much more highly connected than those resulting from BIC. Quantitatively, one can observe this difference in "graph complexity" by comparing the dimension of the graphs resulting from use of both scoring metrics.

To better compare our various search and scoring options, we performed the following experiment: for each of the datasets under consideration ("A" and "B"), we generated 200 randomly-connected initial graph structures. We then separately applied Greedy+AIC, Greedy+BIC, First-Ascent+AIC, and First-Ascent+BIC to each of these initial graph structures, recording various details about each application of all these search variants.

As can be seen in Table 1, the dimension of graphs produced from the above experiment vary consistently between AIC and BIC. In particular, the average dimension yielded by all the AIC data/search combinations are consistently larger than those found with BIC; similarly, the standard deviations of the graph dimension are larger for AIC than BIC. Most tellingly, in all but one case, the smallest graph dimension yielded by AIC is larger than the largest graph dimension yielded by BIC[3]. Thus, the bulk of the graphs produced by AIC have a higher dimension than any of the graphs produced when using BIC.

One may also compare the runtime of the various searches, and the number of search steps taken before finding a result in each case. On average, AIC also takes more time to yield a result, and uses more search steps, than BIC (as can be seen in Table 2). This is because it requires more steps to build up the complex resulting structures: AIC will continue to add new edges to the graph beyond the point at which the BIC algorithm concludes that additional edges do not provide enough of a likelihood improvement to offset the increased complexity of the structure, and considering and applying the new edges requires time.

From these comparisons, one might conclude that AIC tends to overfit the given data, by producing Bayes' Net structures which are unjustifiably complex. Furthermore, searching with AIC suffers the additional cost of slightly lower efficiency than when using BIC. In contrast, on the datasets under consideration, BIC tends to produce relatively streamlined graphs; it seems likely that the structures found by BIC provide more useful approximations of the underlying Bayesian relationships which generated the data in the first place.

Table 1: Resulting graph dimensions, AIC vs. BIC

	AIC dim				**BIC** dim			
	μ	σ	min	max	μ	σ	min	max
Data A, First-Ascent	8.674	0.469	8.000	9.000	7	0	7.000	7.000
Data A, Greedy	8.421	0.494	8.000	9.000	7	0	7.000	7.000
Data B, First-Ascent	51.768	11.264	28.000	82.000	21.311	2.239	19.000	36.000
Data B, Greedy	51.953	10.768	31.000	80.000	20.647	1.226	19.000	24.000

[3]Furthermore, in each case, the mean BIC dimension is at least 2.7 of AIC's standard devs below the mean AIC; by Chebyshev's Theorem, we have (as a lower bound) that at least $1 - \frac{1}{2.7^2} \approx 86\%$ of AIC's dimensions are larger than the average BIC dimension.

Table 2: Runtime and search steps, AIC vs. BIC

	AIC		BIC	
	μ_{time}	μ_{steps}	μ_{time}	μ_{steps}
Data A, First-Ascent	40.922	5.118	36.049	4.755
Data A, Greedy	71.049	4.029	69.127	3.794
Data B, First-Ascent	34920.358	24.907	27181.069	21.539
Data B, Greedy	93357.363	15.623	78283.201	14.225

Table 3: Result quality, AIC vs. BIC

	AIC score				BIC score			
	μ	σ	min	max	μ	σ	min	max
A, FA	-201.587	0.403	-202.450	-201.236	-220.545	1.476	-222.823	-219.493
A, Gr	-201.391	0.233	-202.450	-201.236	-220.692	1.582	-222.823	-219.493
B, FA	-31439.454	3.710	-31450.532	-31433.896	-31562.090	18.520	-31652.759	-31547.159
B, Gr	-31437.906	3.030	-31446.883	-31433.836	-31557.216	12.737	-31613.307	-31547.159

3 Structure Search

3.1 Structure Search: Initial Networks

Before we can proceed to implement and test greedy structure search, we must generate initial networks. We tested our greedy search algorithm using the following initial networks:

- **Given:** the initial network provided in the data files.
- **Unconnected:** an unconnected network — all variables independent
- **Connected:** a fully connected network — each node is a child of every node before it in the topological ordering
- **Random:** a randomly generated network, produced as follows: on each iteration, one parent and one child node are chosen, and an edge is added between them if valid. It is valid to add an edge if the edge does not already exist, and the child is not an ancestor of the parent; these conditions are necessary and sufficient for avoiding directed cycles[4] This process is repeated until there are n edges in the network, where n is the number of nodes in the network.

The initial networks used in our experiments can be found in the first graph of Figures 1 and 7 for Given networks, Figures 2 and 8 for Unconnected networks, Figures 3 and 9 for Connected networks, and Figures 4– 6 and 10– 12 for Random networks.

[4]This is a subset of the rules for which operations are valid, described in Task 2, since here we are only adding edges and there we also consider reversing edges.

3.2 Implementation Details

In this section, we discuss the implementation and results of our local structure search algorithms.

3.2.1 Structure-search Framework

In planning our structure-search implementation, we made the observation that all our search algorithms worked via a process of iterative refinement: at any given point there is a single "current network structure," which may be paired with a "best known operator" (if one has been found). Given the potential difficulty and inefficiency of creating deep copies of the provided BayesNet objects, we decided to implement our structure search based on a paradigm where there was a single "working copy" of the BayesNet object, which could be temporarily transformed in order to determine the score which would result from various strucure-search operators.

To this end, we decided to implement a BNOp (Bayes Net Operation) object class which could be used to represent any given operation. When created, a BNOp object specifies the structure-operator (add, delete, reverse), a source node, and a target node. We also implemented an operator-validity test method which, given a current network structure, evaluates the legality of the network which would result from the application of a given BNOp.

In order to test operation validity without actually modifying the network structure, we introduced a couple of concepts of node-relationship: Ancestors and GrandAncestors. A is an Ancestor of B if there is a directed route from A to B. A is a GrandAncestor of B iff there is a directed route from A to B which contains more than one edge[5].

Operation validity was determined as follows:

- **Delete** Any existing graph edge could be deleted[6].
- **Add** An edge could be added if both (1) it did not already exist and (2) it did not create any directed cycles in the graph[7].
- **Reverse** Any existing graph edge could be reversed if it did not create any directed cycles in the graph[8].

One may note that our Reverse-edge operator is directional – if Reverse(A, B) is valid, Reverse(B, A) is never valid; we deliberately chose to make this adjustment so there would be no redundancy among the valid operations[9].

It can be shown that this formulation of operation validity is closed over the set of legal network structures. (I.e., given a legal network structure, one cannot generate a nonlegal network from the application of BNOps which meet the above conditions for validity.)

[5] I.e. regardless of whether A is a *direct* parent of B, is A an *indirect* ancestor of B?

[6] delete(a, b) is valid IFF a is a parent of b

[7] add(a, b) is valid IFF (a is not a parent of b)$\wedge$(b is not an Ancestor of a)

[8] reverse(a, b) is valid IFF (a is a parent of b)$\wedge$(a is not a GrandAncestor of b)

[9] Redundancy among the structure-search operations causes wasted work when one is exhaustively calculating the scores of all operations. Given that scoring is relatively slow, we decided to optimize by scoring each reversal only once. Note that this optimization is functionally equivalent to the non-optimized form, aside from reducing the probability of randomly selecting a Reverse operation.

Finally, we implemented an apply/unroll mechanism for BNOps, such that they could be applied (and optionally later unrolled in the order of original application) to a given BayesNet object. Thus, given a current network structure, the BIC score of a given BNOp X could be calculated idempotently by applying X, calculating the MLE-based BIC score, and then unrolling X.

3.2.2 Greedy Hill-Climbing Implementation

With this framework of Bayes Net operations, it is straightforward to implement the greedy structure search algorithm. We begin with one of the initial networks described in Task 1. At each step, we generate all possible valid operations, compute the CPT values that best fit the data using maximum likelihood estimation, and determine the BIC score. We choose the operation that leads to the highest BIC score; ties are broken randomly. If this operation improves the BIC score, it is applied and the process is repeated; if no operation leads to a better BIC score, we have reached a local maximum, and stop.

Note that the random tiebreaking process means that even with the same initial network, running the greedy structure search algorithm can lead to two different final networks. Indeed, even though the randomization happens only when breaking a tie, the final BIC score can be different since the different resulting networks can lead down different paths of operations. We did not, however, observe any nondeterministic results given a nonrandom initialization method; repeating the procedure with the same initialization method gave the same results[10]. From this we may conclude that our search process most likely did not encounter many ties in BIC scoring. On the other hand, as is described in Section 3.3.1, many different final networks are possible when starting from a random initial network.

3.2.3 First-Ascent Hill-Climbing Implementation

Given the above implementation of Greedy Structure Search, it is fairly straightforward to adjust the algorithm to perform First-Ascent Hill-Climbing structure search. In first-ascent search, the currently-valid operations are examined (in random order) until one is found which results in an BIC-score improvement over the current graph state. That operation is applied to yield a new "current graph state," and first-ascent is applied iteratively to the resulting graph. When the graph reaches a state where no operation would improve the BIC score, we have reached a local maximum, and stop.

Note that, at any given state, there are likely to be multiple operations which would cause a scoring improvement. Unlike greedy search, first-ascent search picks randomly among these operations; thus, when attempting to find an optimal graph (given a particular initial search state), first-ascent search is much more likely than greedy search to climb to different local maxima if invoked multiple times. Hence, nondeterministic behavior is relatively likely in first-ascent search.

[10] Accordingly, we show only one set of results for the non-random initialization methods

3.3 Results

3.3.1 Greedy Structure Search Results

In the case of the "Data A" network, where the supplied network structure was presumably similar to that used in generating the data, it is unsurprising that the very fastest search occurred when starting with the given network configuration. Obviously, if one deliberately starts out very close to a (local) maximum, it doesn't take many steps to reach that maximum.

Aside from the above case, the fastest search was performed when starting with an initially-unconnected network (regardless of which dataset was used), as can be seen in Table 2. Similarly, the slowest search occurred when using the fully-connected initialization method. These results should not be surprising either; the dominating factor in our search times are the MLE/BIC calculations, and these are fastest for small CPTs (as in the fully/mostly disconnected case) and slowest for high-dimensional CPTs (as occur in the fully-connected case). Furthermore, the fully-connected initialization is also penalized by the fact that it has to make $O(n^2)$ decisions about which edges to delete; it starts with a heavily overspecified structure and must take many more steps to reach a local maximum than in the case of an initially-unconnected network. In the two experiments we ran, the fully-connected initial network led to the best final structures, though it is not clear whether this is just a property of the data sets we were considering, or applied more generally. However, it took much longer than the other methods to find a local maximum, so this extra computation time may not be worthwhile.

In the case of the random initialization, we observed one of a variety of different results each time we ran the search algorithm. This is because the randomly-generated starting structures cause the greedy hill-climbing search to begin at randomly selected points in the search space, and there exist multiple local maxima towards which the greedy search ascended. It is noteworthy that, for both networks, the random initialization led in some cases to the same best final result that we obtained from starting with a fully-connected network. However, it did so much faster, requiring fewer steps and less time to reach the local maximum. This suggests that starting from a random initial configuration is a worthwhile approach for learning a network structure, though the nondeterminism implies that it may be best to run the greedy search procedure more than once on different random networks.

3.3.2 First-Ascent Search Results

As can be seen in Table 1, first-ascent search (with AIC or BIC) yielded graphs of comparable complexity (i.e. dimension) to greedy search performed with the same metric (respectively). However, as shown by Table 2, we can see that first-ascent was typically significantly faster than greedy search. Furthermore, Table 3 demonstrates that first-ascent search typically achieved metric scores which were only slightly worse than those for greedy search. However, the standard deviation of scores were substantially greater for first-ascent search, due to the greater degree of nondeterminism involved.

4 Conclusion

We implemented and studied a variety of Bayes'-network search algorithms and scoring metrics. Our experiments allowed the comparison of their relative performance, and assessment of the relative quality of the results achieved by the search algorithm variants. We observed that first-ascent structure search provides approximately the same result quality as greedy search with considerably faster runtime, but has increased variance of result quality due to the nondeterminism of the algorithm. We found that the BIC scoring metric was preferable to the AIC metric for structure search, since its larger structural penalty led to less complex network structures. Additionally, analysis of our experimental results led to a deeper appreciation for the issues involved in structure search, and the tradeoffs which must be made when choosing a scoring metric and a search algorithm.

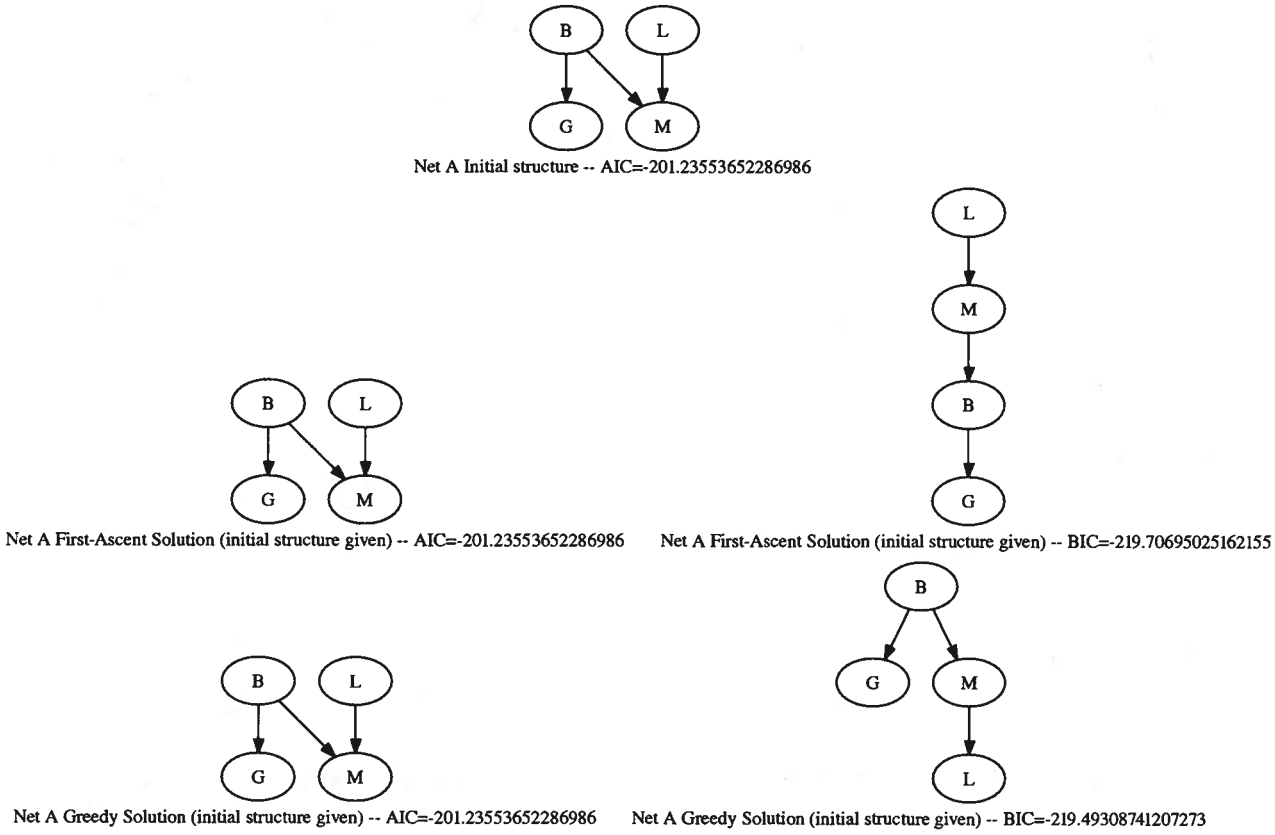


Figure 1: Initial and final structures for Net A, Given initial network

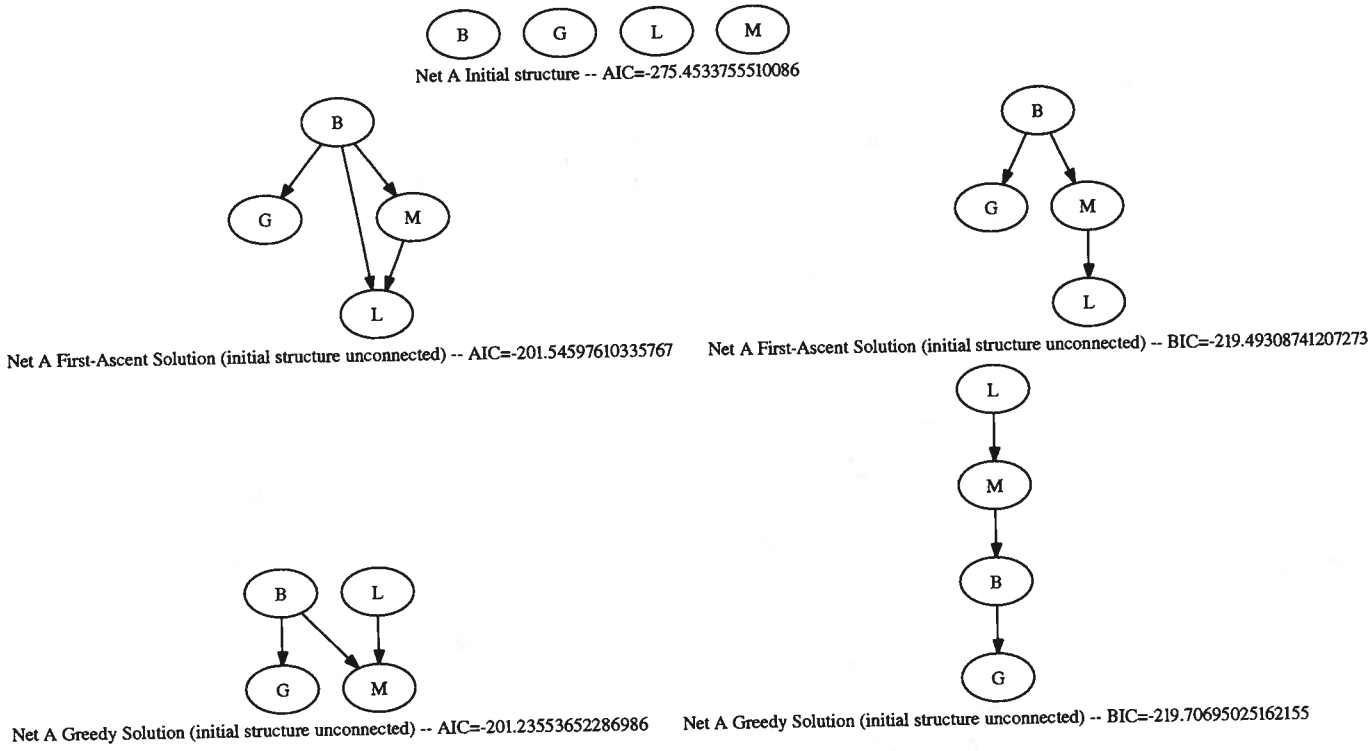


Figure 2: Initial and final structures for Net A, Unconnected initial network

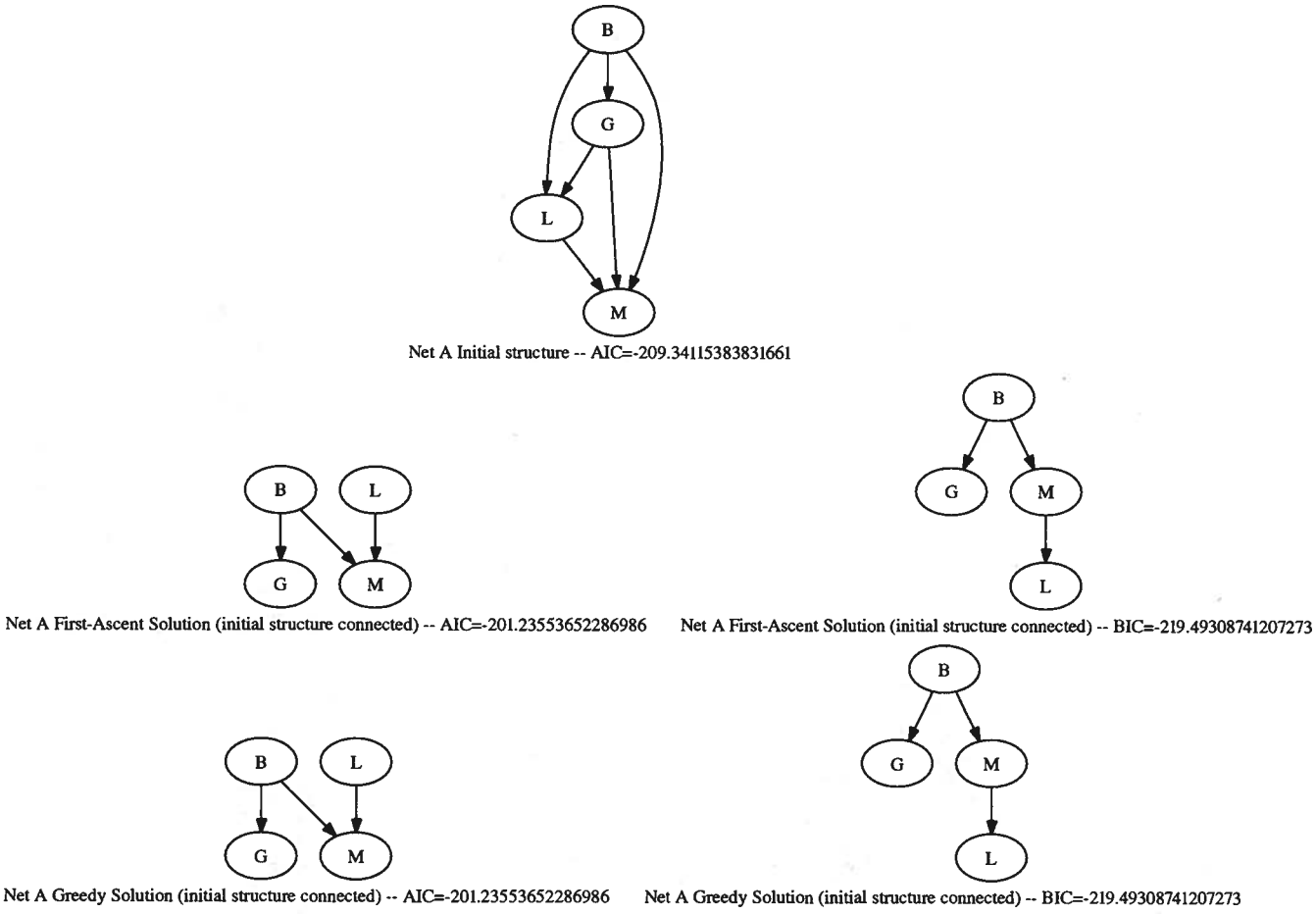
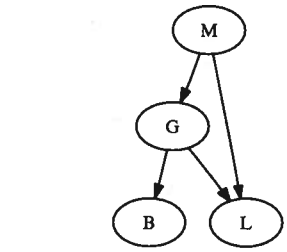
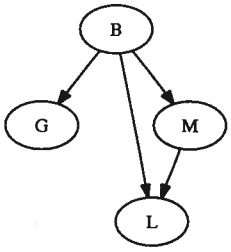


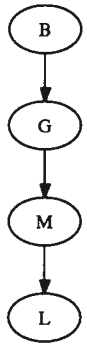
Figure 3: Initial and final structures for Net A, Connected initial network



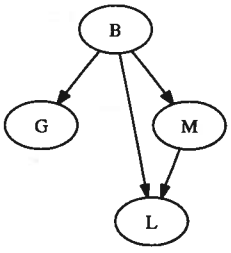
Net A Initial structure -- AIC=-207.79722681871766



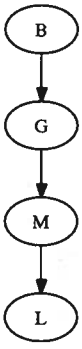
Net A First-Ascent Solution (initial structure random1) -- AIC=-201.54597610335767



Net A First-Ascent Solution (initial structure random1) -- BIC=-222.8233764786591

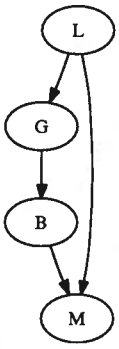


Net A Greedy Solution (initial structure random1) -- AIC=-201.54597610335767

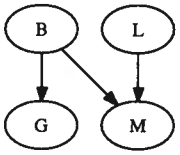


Net A Greedy Solution (initial structure random1) -- BIC=-222.8233764786591

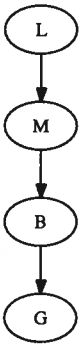
Figure 4: Initial and final structures for Net A, Random1 initial network



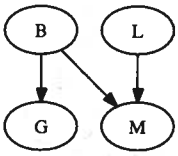
Net A Initial structure -- AIC=-202.38854474616986



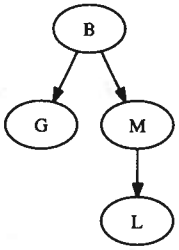
Net A First-Ascent Solution (initial structure random2) -- AIC=-201.23553652286986



Net A First-Ascent Solution (initial structure random2) -- BIC=-219.70695025162155

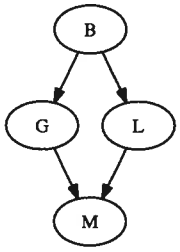


Net A Greedy Solution (initial structure random2) -- AIC=-201.23553652286986

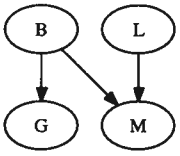


Net A Greedy Solution (initial structure random2) -- BIC=-219.49308741207273

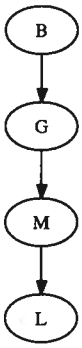
Figure 5: Initial and final structures for Net A, Random2 initial network



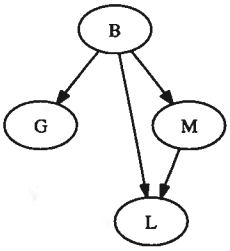
Net A Initial structure -- AIC=-207.7148981200892



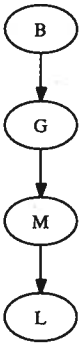
Net A First-Ascent Solution (initial structure random3) -- AIC=-201.23553652286986



Net A First-Ascent Solution (initial structure random3) -- BIC=-222.8233764786591

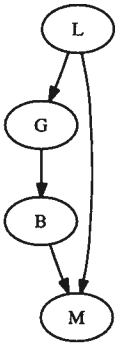


Net A Greedy Solution (initial structure random3) -- AIC=-201.54597610335767

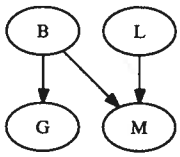


Net A Greedy Solution (initial structure random3) -- BIC=-222.8233764786591

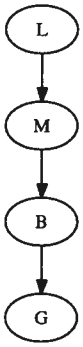
Figure 6: Initial and final structures for Net A, Random3 initial network



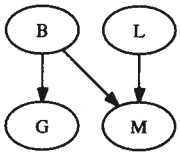
Net A Initial structure -- AIC=-202.38854474616986



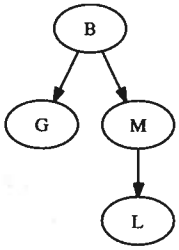
Net A First-Ascent Solution (initial structure random2) -- AIC=-201.23553652286986



Net A First-Ascent Solution (initial structure random2) -- BIC=-219.70695025162155

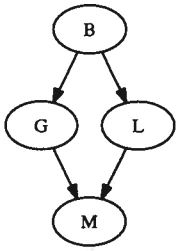


Net A Greedy Solution (initial structure random2) -- AIC=-201.23553652286986

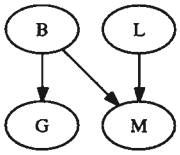


Net A Greedy Solution (initial structure random2) -- BIC=-219.49308741207273

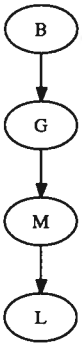
Figure 5: Initial and final structures for Net A, Random2 initial network



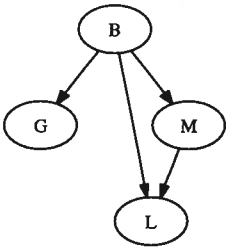
Net A Initial structure -- AIC=-207.7148981200892



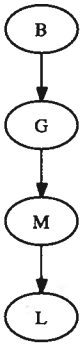
Net A First-Ascent Solution (initial structure random3) -- AIC=-201.23553652286986



Net A First-Ascent Solution (initial structure random3) -- BIC=-222.8233764786591

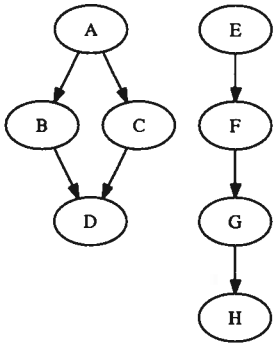


Net A Greedy Solution (initial structure random3) -- AIC=-201.54597610335767

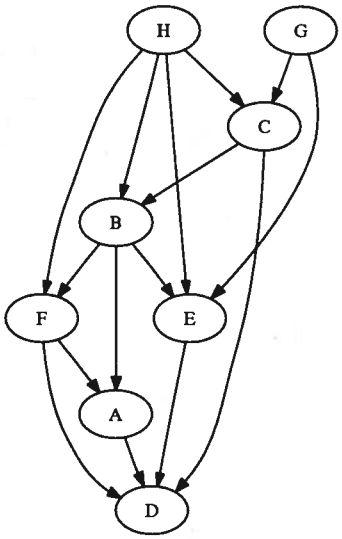


Net A Greedy Solution (initial structure random3) -- BIC=-222.8233764786591

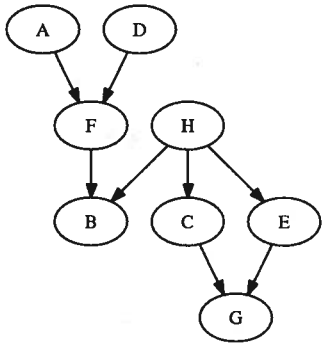
Figure 6: Initial and final structures for Net A, Random3 initial network



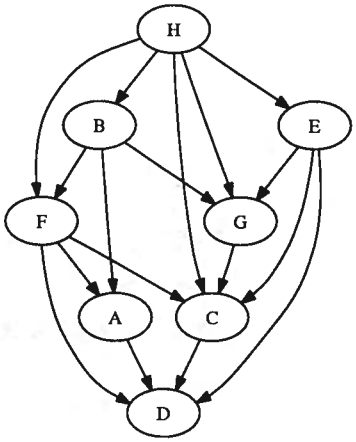
Net B Initial structure -- AIC=-34301.24887441866



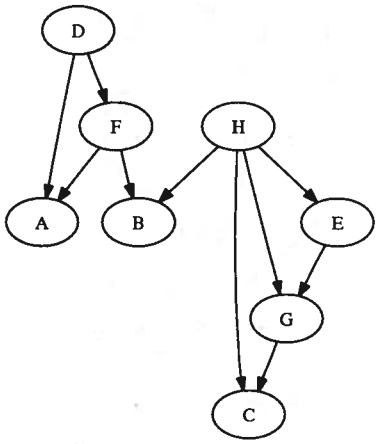
Net B First-Ascent Solution (initial structure given) -- AIC=-31437.093389999754



Net B First-Ascent Solution (initial structure given) -- BIC=-31547.158888303864



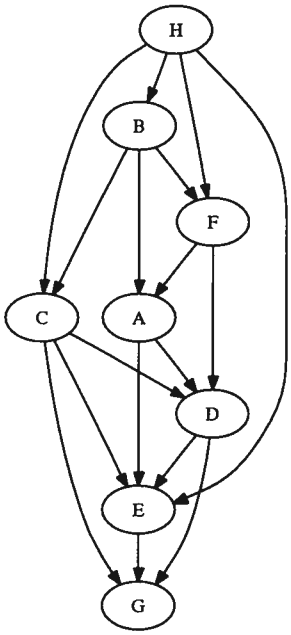
Net B Greedy Solution (initial structure given) -- AIC=-31436.24419271924



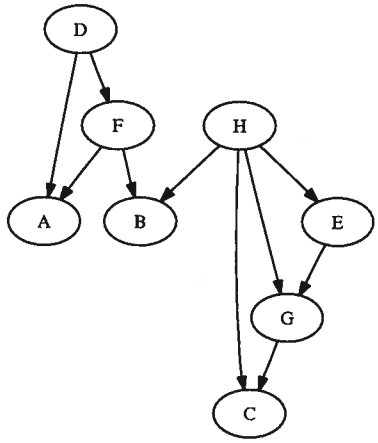
Net B Greedy Solution (initial structure given) -- BIC=-31563.285559390988

Figure 7: Initial and final structures for Net B, Given initial network

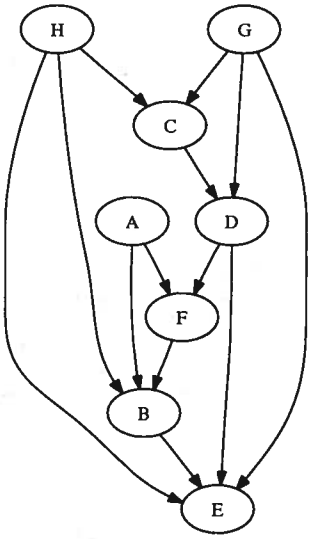
Net B Initial structure -- AIC=-34309.94209133534



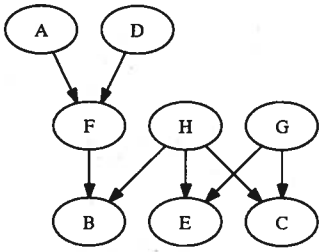
Net B First-Ascent Solution (initial structure unconnected) -- AIC=-31438.6169736581



Net B First-Ascent Solution (initial structure unconnected) -- BIC=-31563.285559390988

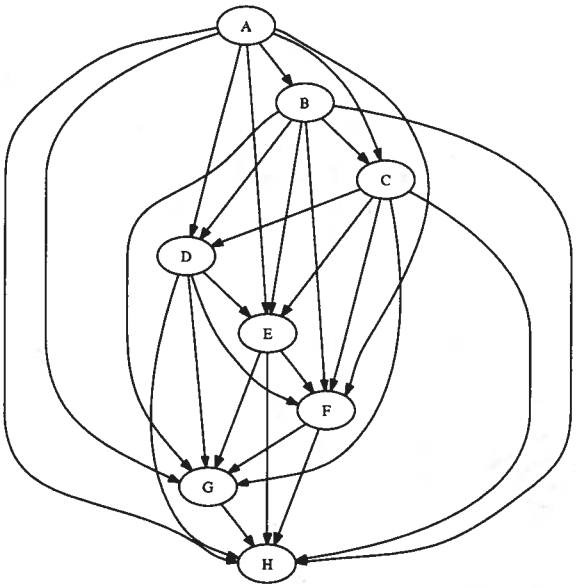


Net B Greedy Solution (initial structure unconnected) -- AIC=-31438.881021413647

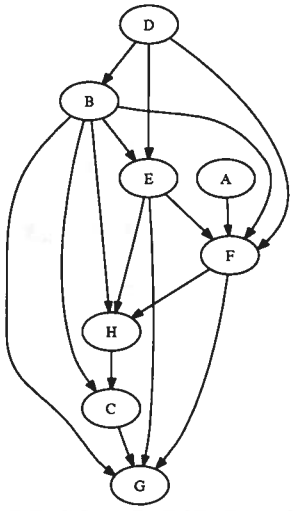


Net B Greedy Solution (initial structure unconnected) -- BIC=-31551.472898514498

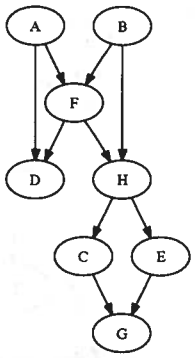
Figure 8: Initial and final structures for Net B, Unconnected initial network



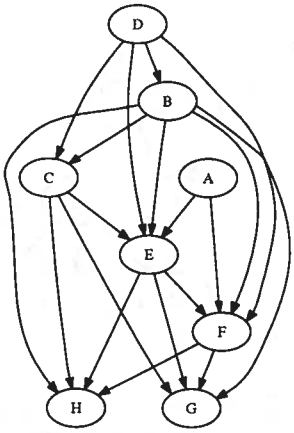
Net B Initial structure -- AIC=-31566.37426546428



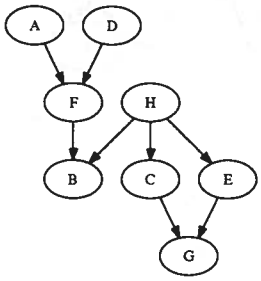
Net B First-Ascent Solution (initial structure connected) -- AIC=-31443.13767361484



Net B First-Ascent Solution (initial structure connected) -- BIC=-31555.476581343046

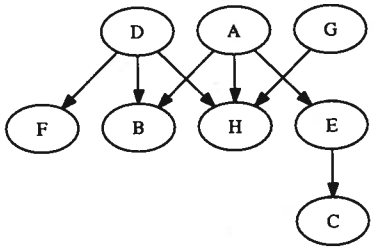


Net B Greedy Solution (initial structure connected) -- AIC=-31446.95203171633

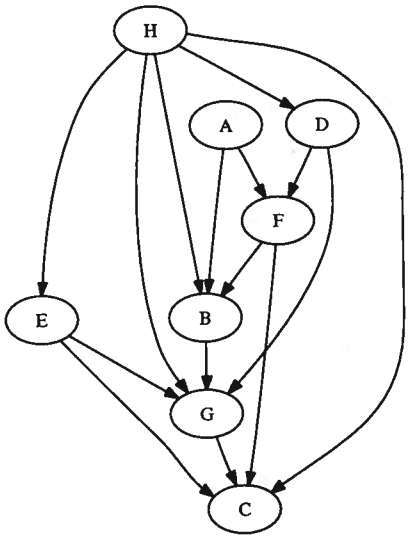


Net B Greedy Solution (initial structure connected) -- BIC=-31547.158888303864

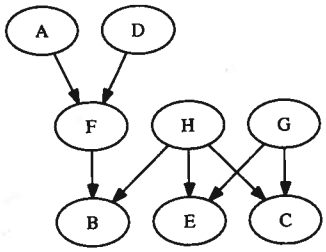
Figure 9: Initial and final structures for Net B, Connected initial network



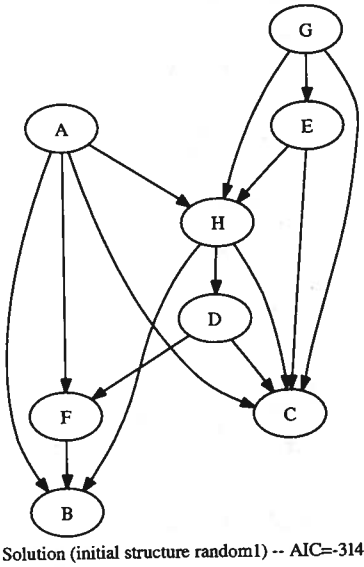
Net B Initial structure -- AIC=-33841.13307743078



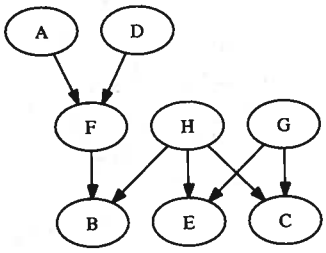
Net B First-Ascent Solution (initial structure random1) -- AIC=-31440.379986152344



Net B First-Ascent Solution (initial structure random1) -- BIC=-31551.472898514498

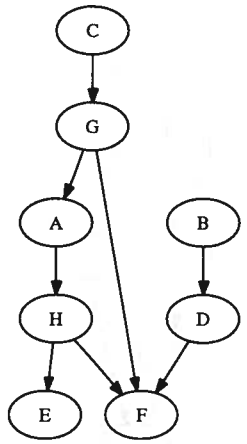


Net B Greedy Solution (initial structure random1) -- AIC=-31440.49469940629

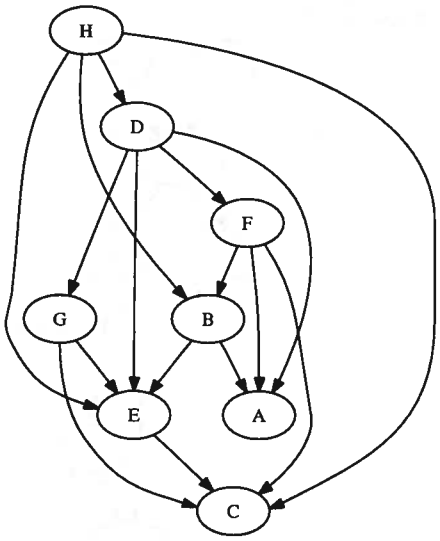


Net B Greedy Solution (initial structure random1) -- BIC=-31551.472898514498

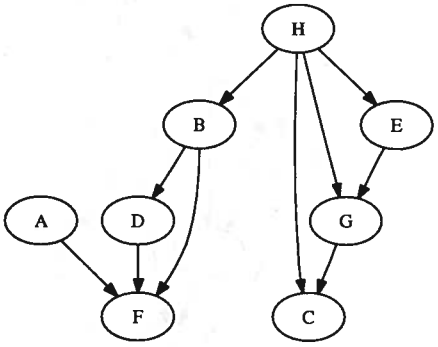
Figure 10: Initial and final structures for Net B, Random1 initial network



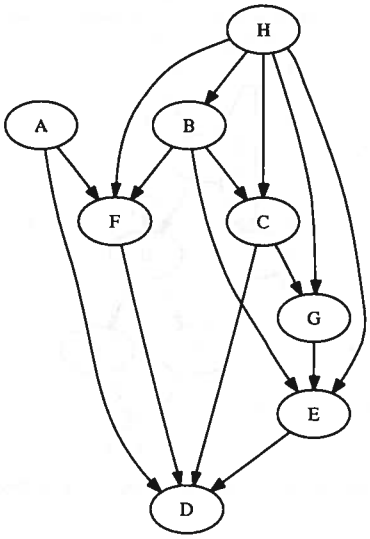
Net B Initial structure -- AIC=-33218.952443116104



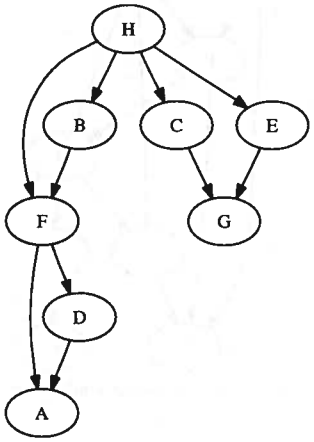
Net B First-Ascent Solution (initial structure random2) -- AIC=-31439.70425079518



Net B First-Ascent Solution (initial structure random2) -- BIC=-31613.307106263546

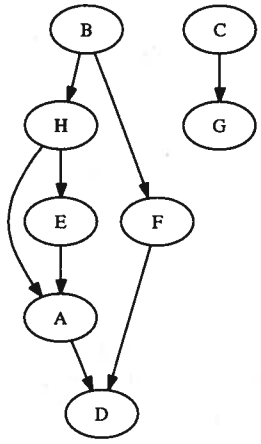


Net B Greedy Solution (initial structure random2) -- AIC=-31438.87858754794

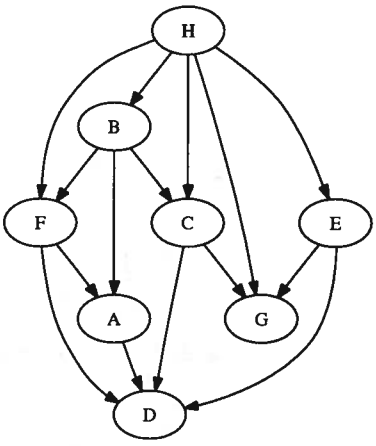


Net B Greedy Solution (initial structure random2) -- BIC=-31557.904070156135

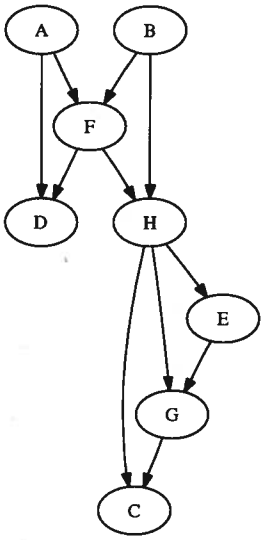
Figure 11: Initial and final structures for Net B, Random2 initial network



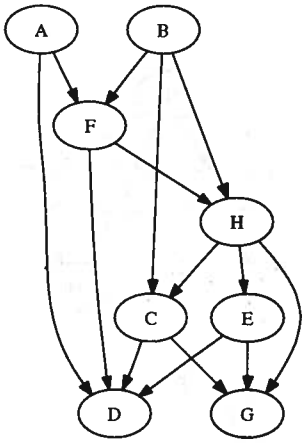
Net B Initial structure -- AIC=-32099.666610150893



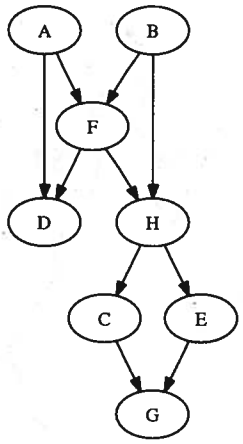
Net B First-Ascent Solution (initial structure random3) -- AIC=-31437.0038495327



Net B First-Ascent Solution (initial structure random3) -- BIC=-31565.78041889262



Net B Greedy Solution (initial structure random3) -- AIC=-31436.044621914443



Net B Greedy Solution (initial structure random3) -- BIC=-31555.476581343046

Figure 12: Initial and final structures for Net B, Random3 initial network