ASSESSING THE USE OF VOTING METHODS TO IMPROVE BAYESIAN NETWORK STRUCTURE LEARNING

A Thesis
Presented to
The Academic Faculty

by

Khaldoon Abu-Hakmeh

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Chemical & Biomolecular Engineering

Georgia Institute of Technology
December 2012

ASSESSING THE USE OF VOTING METHODS TO IMPROVE BAYESIAN NETWORK STRUCTURE LEARNING

Approved by:

Dr. Mark Styczynski, Advisor
School of Chemical & Biomolecular Engineering
Georgia Institute of Technology

Dr. Matthew Realff
School of Chemical & Biomolecular Engineering
Georgia Institute of Technology

Dr. Martha Grover
School of Chemical & Biomolecular Engineering
Georgia Institute of Technology

Date Approved: August 24, 2012

To my family, for their encouragement and support.

ACKNOWLEDGEMENTS

I wish to thank Dr. Mark Styczynski, Assistant Professor in Chemical & Biomolecular Engineering, for his guidance in research and completing this thesis. I would also like to thank my committee members, Dr. Matthew Realff and Dr. Martha Grover for their helpful comments, feedback, and questions.

I thank my colleagues in the Styczynski Group for the stimulating discussions and friendship over the past two years.

Finally, I thank my parents, Emad Abu-Hakmeh and Amal Ileiwat, my wife, Sabrina Oukil, and brother and best friend, Ahmad Abu-Hakmeh, for their unyielding love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Structure inference in learning Bayesian networks remains an active interest in machine learning due to the breadth of its applications across numerous disciplines. As newer algorithms emerge to better handle the task of inferring network structures from observational data, network and experiment sizes heavily impact the performance of these algorithms. Specifically difficult is the task of accurately learning networks of large size under a limited number of observations, as often encountered in biological experiments. This study evaluates the performance of several leading structure learning algorithms on large networks. The selected algorithms then serve as a committee, which then votes on the final network structure. The result is a more selective final network, containing few false positives, with compromised ability to detect all network features.

CHAPTER 1

INTRODUCTION

Bayesian networks have become a recurrent tool for studying complex systems due to their intuitive representation of relationships. Learning network structure is particularly interesting in domains where complex interactions exist between variables. Bayesian network structure learning has been successfully applied to address problems in bioinformatics[1, 2], decision support systems[3], and information retrieval[4] among others. This study aims to address the use of Bayesian network structure learning to study large-scale networks observed under small sample size. This is motivated by an attempt to develop methods applicable datasets such as those found in bioinformatics, where data is available for many variables, but insufficient to capture all relationships due to experimental limitations. Since no single algorithm to date has proven consistent performance in detecting true positives and avoiding false negatives, existing algorithms are used to create an ensemble and vote on relationships learned in the data. A brief discussion of basic probability and the theory behind the construction and learning of Bayesian networks is helpful prior to presenting detailed methods and results used in this study.

Probability

The probability of an event occurring can be viewed as the limit of the relative frequency of an event in an arbitrarily large number of random experiments or trials. Probabilities of observing multiple events can be represented jointly. For example, the probability of observing two events, A and B, is represented as $P(A \cap B)$. If events are independent, then their joint probability is expressed as the product of each probability occurring. The probability of two independent events occurring together is represented by $P(A \cap B) = P(A)P(B)$. Where events are not independent, the notion of conditional probability becomes relevant. The conditional probability of an event *A* given that an event *B* has occurred is given by

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}.$$

It follows from the definition of joint probability that

$$P(A \cap B) = P(A \mid B)P(B) = P(B \mid A)P(A).$$

This result gives the definition of Bayes' theorem:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

In the application of Bayes' theorem, P(A|B) is the posterior probability of *A* given *B*, or the degree of belief having accounted for *B*. P(B|A)/P(B) represents the support *B* provides for *A*, and P(A) is the prior, or initial degree of belief in *A*. In cases where *A* and *B* are independent, their conditional probabilities are equal to their respective prior probabilities, $P(A \mid B) = P(A)$ and $P(B \mid A) = P(B)$.

Drawing from the definitions of conditional probability and independence, conditional independence of two variables, *A* and *B*, can be written if they are independent with respect to a third variable, *C*: $P(A \mid B \cap C) = P(A \mid C)$ and $P(B \mid A \cap C) = P(B \mid C)$. Conditional independence is an important characteristic in the representation of Bayesian networks. For the example above, conditional independence can be represented as $I(A; B \mid C)$ and $I(B; A \mid C)$.[1, 5, 6]

Bayesian Networks

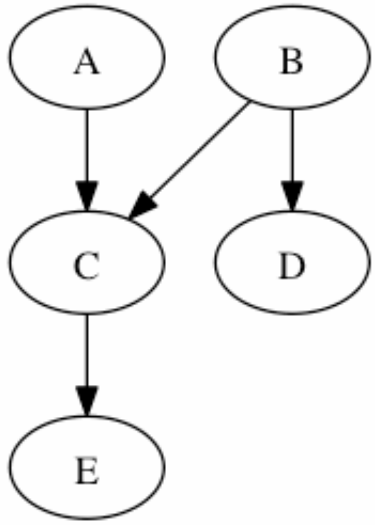


Figure 1: A simple Bayesian network

Probability Representation

Bayesian networks are graphical models representing probabilistic relationships among a set of variables. Bayesian networks are acyclic, such that no connected path of edges returns to a node along the path. A graph $G = \langle \mathrm{X}, \mathrm{E} \rangle$ is a Directed Acyclic Graph consisting of variables belonging to the set $\mathrm{X} = \{X_i, ..., X_n\}$ connected by a set of edges, E. When the network is discretely distributed, the graph is denoted as $\langle G, P \rangle$, with P encoding a discrete joint probability distribution of the variables. Random variables are represented as nodes and are connected by directed arrows called edges indicating conditional dependence. Nodes that are not connected by an edge are conditionally independent of one another in this framework. Considering a specific node, A, all nodes with edges directed towards A are called the parents of A. Conversely, all nodes accepting edges originating from A are called children of A. The graph is called a Bayesian network if it satisfies the Markov condition, which states a node is conditionally independent of its non-descendants, given its parents.[5-7]

For example, consider constructing a network representing the operation of a sprinkler.[5] If grass is observed to be wet, that implies one of two possible causes. Wet grass can be a result of either rain or a sprinkler. The operation of the sprinkler or rainfall is also related to whether or not the sky is cloudy. This description allows for construction of a Bayesian network, representing the believed interactions between the four variables considered. This Bayesian network is depicted in Figure 2. The four nodes represent the variables. The edges represent the probabilistic relationships between the variables. Further to the graphical representation of the network, there is a probability distribution describing how the variables interact. In the case of the sprinkler network, a discrete probability distribution is considered. Nodes in this example assume binary values of either "True" if present or "False" if absent. The number of values that a node can assume is called the Node Size.

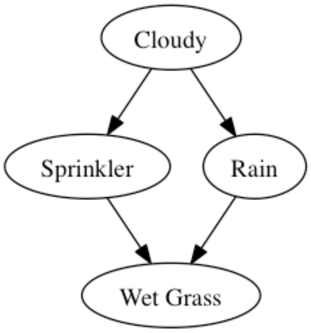


Figure 2: A Bayesian network representing a sprinkler system

Bayesian networks can be used to perform inference given observational data. An observation is defined as the state of all variables at a single point in time. Collecting multiple observations illustrates the different states that nodes can assume. With sufficient observational data, patterns between the variable's values begin to emerge. These patterns indicate potential relationships between the data. Table 1 contains data from five observations of the sprinkler network from Figure 2. The values contained in the table are representative of the underlying probability distribution of the sprinkler network.

Table 1: Observational Data for the Sprinkler Network

Observation\Variable	Cloudy	Sprinkler	Rain	Wet Grass
Observation 1	True	False	True	True
Observation 2	False	True	False	False
Observation 3	False	False	False	False
Observation 4	False	False	False	False
Observation 5	False	False	True	True

In this case, all nodes additional to the sprinkler node itself are within the Markov blanket of the sprinkler. For a given node, the Markov blanket is defined as the set of all nodes that are parents, children, or the other parents of its children.[5] Representing the dependencies in the neighborhood of the variable of interest allows for complete analysis of the system, in this case, the sprinkler. In our example, we might say that the probability of it being cloudy is 0.5, the probability of it raining given that it is cloudy is 0.6, and the probability of it raining given that it is not cloudy is 0.05. We can similarly define such "probability distributions" for each of the nodes, which when combined with the topology of the graph provides a complete Bayesian network that can be used to characterize and model the system.

Essential to the construction of Bayesian networks are concepts from probability theory, conditional independence, and Bayes' rule discussed above. Since Bayesian networks represent probabilistic relationships, the probability of observing of a node (random variable) is conditioned on its parent nodes. The distribution of the entire network can be represented as the product of the probabilities of all constituent nodes due to the rules of independence in probability, given that all nodes are independent of their non-descendants conditioned on their parents. The distribution of the network, can be factored as follows:

$$P(\mathrm{X}) = P(X_1, \ldots, X_n) = \prod_{X_i \in \mathrm{X}} P(X_i \mid \mathrm{P}a_i^G),$$

Where $\mathrm{P}a_i^G$ is the set of all variables on which X_i is conditioned.

Conditional independence allows for a compact representation of each variable in the factored distribution. In absence of the property, the storage of values for the set would be exponential with the number of variables. The distribution would quickly become computationally intractable.[8, 9] The chain rule of probability and conditional independence of variables therefore reduces the problem of constructing the distribution over all variables to a much simpler problem. Learning the conditional probabilities of each variable individually is an easier task. The resulting distribution can be expressed as the product of the smaller probabilities.

Equivalence Classes

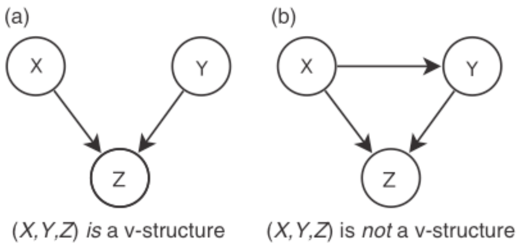


Figure 3: Comparison of a v-structure to a similar, but not equivalent set of edges[10]

Two Bayesian network graphs are said to be equivalent if they encode the same set of independencies.[5, 11] For example, $A \rightarrow B \rightarrow C$, $A \leftarrow B \rightarrow C$, and $A \leftarrow B \leftarrow C$ are all equivalent structures. However, $A \rightarrow B \leftarrow C$ is not considered equivalent to the three structures above. To understand equivalence, consider the parameters required for each of the models. For the first case, the required parameters are P(A), P(B|A) and P(C|B). Similarly, the parameters required for the second case are P(A|B), P(B), and P(C|B). By the earlier definition of conditional probability, these two cases are equivalent. The same applies for the third case, where P(C), P(B|C), and P(A|B) are required. The final case, where $A \rightarrow B \leftarrow C$, is represented by P(B|A,C), P(A), and P(C). Since P(B|A,C) cannot be determined from the probabilities of the other structures, this structure is not equivalent. The latter example is called a V-structure.[12] For any two directed acyclic graphs to be equivalent, they must share the same underlying undirected graph and V-structures. Equivalence structures can be represented as partially directed graphs, where V-structures are conserved, but edges where both $A \rightarrow B$ and $A \leftarrow B$ can exist are represented by an undirected edge, $A - B$.[11]

CHAPTER 2

STRUCTURE LEARNING

In studying complex systems, data is often available for variables, yet the nature of interactions between them remains unclear. Knowledge of structure is essential to understanding and characterizing emergent properties of systems. Problems in classification and prediction can addressed when the underlying network structure is known.[13] Given observational data, structure learning of Bayesian networks is a problem of selecting a probabilistic model to explain the data. While experts can construct networks, the task becomes difficult when the domain is too large or complex. The problem becomes more difficult if there are “hidden variables”, those whose measurements are not taken or known when collecting the data. Machine learning approaches are useful in addressing these problems.

Learning Bayesian networks from data is an NP-hard problem.[14, 15] There are two prevalent approaches to learning Bayesian network structure. In the constraint-based approach, data is subjected to conditional independence tests to determine the presence of a relationship. The other predominant approach is a search-and-score technique, where directed acyclic graphs are generated, scored on the probability of observing the network given the data, then modified to improve the score. Learning the probability distributions represented by edges in the network is called Parameter Learning. Learning the parameters is a sub-problem of structure learning. However, the details of parameter learning are beyond the scope of this study.

Constraint Based Methods

Learning Bayesian network structure by constraint-based methods begins a search with a completely connected graph.[16] Edges are removed according to statistical tests to measure conditional independencies in the data. A drawback of the constraint-based approach is a loss of statistical power due to repetitive independence tests. Independence is determined by testing the association between two variables, given a set of conditioning variables. Generally, these methods do not return a completely directed graph. Instead, a partially directed graph, potentially equivalent to several Bayesian networks, is the result. A prominent example of a constraint-based algorithm is the PC algorithm, introduced in Chapter 4.

An example of a statistical test used by constraint-based algorithms is the G^2 test.[9]

$$G^2 = 2\sum_{a,b,\mathbf{c}} S_{ijk}^{ab\mathbf{c}} \ln \frac{S_{ijk}^{ab\mathbf{c}} S_k^{\mathbf{c}}}{S_{ik}^{a\mathbf{c}} S_{jk}^{b\mathbf{c}}}$$

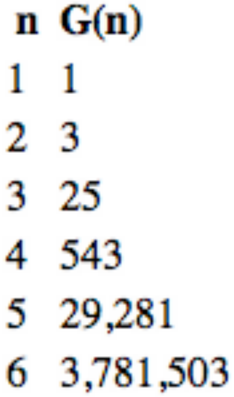
Shown above, the test measures the strength of association between two variables, conditioned on a set of neighboring variables, where S_{ijk}^{abc} represents the number of times in the data $X_i = a$, $X_j = b$ and $\mathbf{X}_k = \mathbf{c}$. G^2 is asymptotically distributed as X^2, returning a p-value corresponding to the probability of falsely rejecting the null hypothesis.[9]

Search and Score

A more popular approach than the constraint-based method is the search-and-score. This approach considers the space of all directed acyclic graphs, returning the best, or candidate set of graphs best fitting the data. However, searching the space of all DAGs is impossible. For a network of *n* nodes, the number of possible graphs is super-exponential in *n*.[17]

$$G(n)=\sum_{k=1}^{n}(-1)^{k+1}\binom{n}{k}2^{k(n-k)}G(n-k)$$

Table 2: Number of Possible Directed Acyclic Graphs Compared to Number of Variables[16]



n	G(n)
1	1
2	3
3	25
4	543
5	29,281
6	3,781,503

Quickly, the number of graphs in the DAG space becomes too large to consider exhaustively. Instead, algorithms resort to global or local search algorithms. Common among search-and-score algorithms are mechanisms to determine the state or phase of the search, mechanism to move between states in the search space, and a scoring function, necessary for comparing states and determining which graph best fits the data.

For example, given two nodes A and B, three cases can be considered. No edge exists, A can be a parent of B, or B can be a parent of A. Given a set of observations described in Chapter 1, a scoring function is used to evaluate how well the structure matches the data. Below, an example of how scoring functions are used is presented.

Scoring Functions

The Bayesian Dirichlet scoring metric is one of many scoring functions used to evaluate Bayesian network structure.[18]

$$BD(B\mid D)=P(B)\prod_{i=1}^{l}\prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\prod_{k=1}^{r_i}N_{ijk}!$$

Where there are l variables are q_i parent configurations of variable i. r_i is the node size of variable i, N_{ijk} is the number of times variable i took on the value k with parent configuration j, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$.[10] By inspection, P(B) is the prior probability of observing structure B. The remainder of the expression is the likelihood of observing the structure of interest, given the available data.

CHAPTER 3

SOFTWARE

There is an abundance of software available for learning structure of Bayesian networks from data.[16] However, most are restricted in their capability to learn networks with a large number of variables. Another issue is that authors of new algorithms present their work as a separate software package. The use of different languages and notation makes direct comparison of network structures learned by multiple algorithms a more difficult task. For this study, packages are selected for the availability of multiple structure learning algorithms. The Matlab language is also preferred due to ease of numerical computations in pre and post-processing of data.

Bayes Net Toolbox

The Bayes Net Toolbox (BNT) presented by Kevin Murphy is an open-source Matlab package for directed graphical models.[16] BNT contains tools for both structure and parameter learning, along with algorithms for inference using Bayesian networks. Additionally, BNT contains useful general functions in preparing and analyzing network data and structure. In this study, the structure learning implementations in BNT were abandoned early on; however, the toolbox is mentioned due to the useful functions available for building, modifying, and sampling data from networks.

BNT Structure Learning Package

The BNT Structure Learning Package (SLP) introduced by Philippe Leray adds an entire suite of structure learning algorithms to interface with Bayes Net Toolbox, using the existing framework as a backbone for structure-specific learning.[17] The algorithms, while useful, are restricted similar to those found in BNT in their ability to infer larger networks. Again, SLP is mentioned due to native functions used in this work to evaluate algorithm performance.

Causal Explorer

The Causal Explorer software is a structure learning toolkit for large-scale network reconstruction presented by Aliferis et al.[19] The package contains a group of established structure learning algorithms in addition to several new algorithms presented by the authors. The package is closed-source, but its Matlab implementation allows for the algorithms to be called by non-native scripts. Causal Explorer's set of algorithms can also be used in conjunction with the useful functions available in BNT and SLP. For these reasons, along with the ability to learn structure of large-scale networks, Causal Explorer is the primary software used in this study.

CHAPTER 4

METHODOLOGY

As discussed in Chapter 3, there are many useful tools for Bayesian network inference. Tools for learning network structure are more limited than those used for inferring and updating probabilistic relationships, but are still adequately present in the machine learning community. The choice of tools is complicated due to the limitations and domain specificity of both the software packages and algorithms. Notably, structure inference software capable of handling larger network sizes are a more recent contribution to the field.[9] Poor implementations of algorithms theoretically proven capable of handling large structure learning render them computationally expensive, and thus, fail to converge to a final structure. To form a robust method suited for learning diverse networks, algorithms are carefully selected with the specific criteria of convergence for networks ranging from tens to hundreds of nodes, with a similar number of edges. The ability to converge given a small number of observations is also considered when selecting the algorithms. While accurate performance is desired, it is not expected under such rigid conditions.

The result is a committee of four algorithms. Each algorithm is tasked with independently learning the network structure fitting the same dataset for a group of well-defined synthetic networks presented in Chapter 5. Following the structure-learning step, the learned graphs are pre-processed, then new graphs are formed using ensemble methods. Both the original graphs and those formed by the ensemble are assessed for accuracy and reasonableness.

Committee

Machine learning literature contains a number of ensemble methods, such as Bootstrap Aggregating (Bagging), which use voting to improve the performance of weak classifiers.[20-22] In Bagging, smaller subsets are randomly sampled with replacement from a dataset. A classification algorithm is then used to train the model. Models learned from each subset of the data are combined to a final model by voting. The resulting model is an improvement from using the unstable algorithm to train the entirety of the dataset. To our knowledge, the only use of a voting committee to assist in learning Bayesian network structure is presented by Mwebaze and Quinn.[23] Following these examples, we apply the use of a committee of algorithms to improve learning of Bayesian network structure.

As described earlier, a committee of four algorithms is used to independently learn network structure. The Max-Min Hill-Climbing, Sparse Candidate, PC, and Three-Phase Dependency Analysis algorithms are selected for the task. The application of each algorithm is discussed in more detail below.

Algorithms

PC

The PC algorithm is a commonly known prototypical constraint-based algorithm developed by Spirtes et al.[7, 24] PC tests for conditional independence using statistical tests, and returns a partially directed graph, equivalent to several possible directed structures as discussed in Chapter 1. Initial development of PC was aimed for causal inference. It is useful for our application, where causality is not strictly considered, since the conditional independence tests theoretically return a low number of false positives. Since the implementation present in Bayes Net Toolbox was incapable of handling large networks, the version available in Causal Explorer was used instead.

The number of edges learned by PC is sufficient to avoid a need to adjust parameters. Since the goal is to provide a large number of edges to be used by the ensemble, and the default parameters of PC achieve this end, they are used. PC takes observational data, node sizes, statistical test, and threshold as input. The result, mentioned earlier, is a partially directed acyclic graph. While the desired goal is to identify relationships between variables by locating edges, the representation of undirected edges in the adjacency matrix is problematic, where the adjacency matrix is a binary representation of the existence or non-existence of an edge in the graph. An undirected edge is presented as two entries in the adjacency matrix. The presence of double entries for a single relationship causes confusion in evaluating the performance by metrics discussed in Chapter 5. To address this problem, the resulting undirected graph is converted to a DAG if it admits extension. A partially directed graph is said to admit extension if it can be converted to a DAG containing the same V-structures and edge location.[25, 26] In cases where extension to a DAG is not possible, one of the two entries in the adjacency matrix is deleted for undirected edges, and the edge is directed from the node with the lower arbitrary order towards the node with the higher arbitrary order. The order is used to identify a node's position in the adjacency matrix. This method is arbitrarily chosen, and while it affects the apparent performance, further processing discussed later in this chapter addresses the issue of directionality.

Sparse Candidate

The Sparse Candidate algorithm was developed by Friedman et al. as one of the first structure learning algorithms suited for large-network inference.[8] Sparse Candidate is a hybrid algorithm that begins by identifying a candidate set of parents, whose size is determined by the user, searching until an acceptable set is found. The algorithm uses a mutual information statistic to test dependence between a variable and its candidate parents. This is called the “restrict” step of the algorithm. Once a satisfactory parent set is determined for each variable, the algorithm proceeds to a maximization step. In this phase, network structures existing in the DAG space equivalent to the candidate sets are scored. The best scoring network is retuned as a directed graph.

The implementation of Sparse Candidate present in the Causal Explorer toolbox takes observational data, node sizes, test statistic, prior (probability value) type, and candidate parent set size as input, returning a DAG. Since the algorithm is used to test networks of varying size and connectivity, the parent size is selected to be sufficiently large for to accommodate all trials, without having to be adjusted. This is done at the expense of longer computational time. Of the remaining input parameters, the type of test statistic and prior type are shown to have the greatest effect on the number of edges learned. Therefore, they are varied in an algorithm to maximize the number of edges. Since the goal is to use the resulting DAG in an ensemble vote, the presence of extra edges is not considered problematic. The goal of maximizing edges is to improve the recall (or sensitivity) of the Sparse Candidate.

Max-Min Hill-Climbing

The Max-Min Hill-Climbing algorithm (MMHC) was first described by Tsamardinos et al.[9] MMHC is considered a hybrid method of the constraint-based and search-and-score approaches described in Chapter 3. A two-phase method is implemented, where an undirected skeleton is learned using constraint-based learning, followed by a scored search of the DAG space to orient edges directionally. MMHC is similar to SC, without the constraint of parent count. The candidate set of parents determined by searching within a restricted space is replaced by the local variables sharing edges with the variable of interest. The argued benefit of this approach is a sounder identification of parents, without user constraints affecting algorithm performance. The constraint-based skeleton-learning step is similar to that employed by the PC algorithm, but using fewer statistical tests to optimize computational resources.

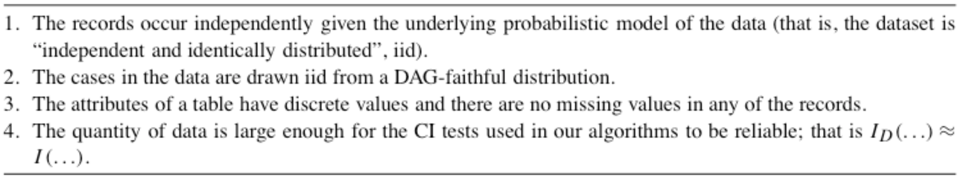
The implementation of MMHC presented in the Causal Explorer package takes several parameters as input, returning an adjacency matrix corresponding to a DAG. Parameters can be tuned to optimize performance provided the user has some prior or expert knowledge of the system of interest. For purposes of developing a robust approach, no knowledge of the system, (besides the number of nodes and edges described in Chapter 5) is assumed prior to learning the structure. Instead, parameters are "optimized" to maximize the number of edges learned. While this is obviously a potential increase in false positive and reduction of accuracy, the ensemble approach is relied upon to amend this issue.

MMHC takes observational data, node sizes, statistical threshold, and prior type as input. Based on testing results, threshold value and prior type are found to have the most significant effect on the number of edges learned. They are therefore used in a maximization algorithm, in which MMHC is used to learn the structure of the associated dataset, and the parameter set maximizing the number of edges, and their corresponding graphs, are kept for use in the ensemble step.

TPDA

The Three Phase Dependency Analysis algorithm uses an information-theoretic approach to learning Bayesian network structure from data.[27] The algorithm requires a polynomial number of conditional independence tests, reducing the number typical of constraint-based approaches. As indicated by the name, the algorithm operates in three phases. The algorithm begins by "Drafting" a network of edges for any nodes with a mutual information statistic above a set threshold. The graph then undergoes "Thickening" to add edges to connect areas of the graph where no relationship is found by the drafting step. Finally, "Thinning" removes redundant arcs between nodes if a path between them already exists.

Table 3: Assumptions of Three-Phase Dependency Analysis[27]



1. The records occur independently given the underlying probabilistic model of the data (that is, the dataset is "independent and identically distributed", iid).
2. The cases in the data are drawn iid from a DAG-faithful distribution.
3. The attributes of a table have discrete values and there are no missing values in any of the records.
4. The quantity of data is large enough for the CI tests used in our algorithms to be reliable; that is $I_D(\ldots) \approx I(\ldots)$.

TPDA takes as input observational data, node sizes, test statistic, and threshold, returning a partially directed graph. Since the number of edges learned by TPDA is typically large in comparison to MMHC and SCA, maximizing the number of learned edges is not a concern. The resulting partially directed graph is directed by the same arbitrary method described above for PC. Any issues in assessing the algorithm's performance are resolved by the methods described in the discussion of directionality below.

Ensemble approaches

As stated above, the objective of selecting a committee and maximizing the sensitivity of the constituent algorithms, specifically MMHC and SCA, was to ultimately improve the ability to identify true positive edges. To achieve this a voting method was implemented. In this approach, each algorithm in the committee received an equal vote on whether each edge should be in the final graph. From the counted votes, four new graphs are made. The "union" graph contains all edges learned by the committee. This graph is presumed to be highly sensitive to detecting true positives, with compromised ability to avoid false positives. A "half" graph contains edges learned by two or more committee members. The expectation of the half graph is an improvement in selectivity from the union, but still too imprecise for confident use. The "majority" graph consists of edges appearing in three or more committee results, and the "intersection" contains only edges learned by all four algorithms. The latter two graphs are expected to display a higher precision than the others.

The hypothesis is that the committee graphs will demonstrate a distinct improvement from any single algorithm. To test the hypothesis, the committee graphs are evaluated using the same metrics as the individual algorithms, and compared in Chapter 5. It is important to note that the graphs resulting from the voting method likely deviate from the DAG space. This concession is accepted, since the goal is to identify the position of true relationships in the data, but not specifically their causal nature.

Directionality

The direction of edges and their inherent implications are a debated topic in the Bayesian network and probabilistic graphical model communities. As mentioned several times, DAGs are often assumed to have a causal interpretation.[6, 24] Therefore, edges must be directed properly, and reversed edges are considered incorrect. The restriction of directionality and maintaining a proper DAG restricts the representation of graphs drawn by the committee. Therefore, the notion of true direction of edges is subsequently discarded. Instead, similar to the case of orienting partially directed graphs, all edges are oriented in a manner such that edges are always represented by a parent node having a lower arbitrary ordering than its child node in the adjacency matrix. Since all algorithm and committee graphs are compared in an undirected form, directionality does not impact performance.

Performance Metrics

To assess the performance of both the individual algorithms and the graphs generated by the committee several metrics were employed. The specific performance criteria were selected to compare the accuracy and completeness of structure learning. More traditional methods, such as network score comparisons, are ignored for two reasons. First, scores vastly differ between algorithms, with radical differences reported even when learned structures are very similar.[17] Second, the decision to discard directionality removes the graphs from the DAG space, rendering scoring ineffective in this case. Instead, three methods are described in this section that assess the graphs solely on the ability to learn correct edges, since relationships between variables take precedence over how well the network fits the data according to some score.

Editing Distance

The editing distance, also referred to as edit or Levenshtein distance, is a metric for measuring the differences between two strings or structures. In the case of graphs consisting of nodes and edges, the editing distance is the number of changes, in the form of addition, deletion, or reversal (for directed graphs) of edges needed to transform one structure into another. In this study, the editing distance is determined between a learned graph and the known network structure from which the data was generated.

Selectivity

Also called precision, is the ratio of correctly inferred edges to total edges present in an inferred model. The selectivity is calculated as shown below:

$$Selectivity = \frac{TP}{TP+FP}\text{; } TP = True\ positives,\ FP = False\ Positives$$

False positives are defined as incorrect edges detected in the inferred network.

Sensitivity

Also called recall, is the ratio of true edges learned to the total number of edges present in the true graph. Sensitivity is calculated as shown below:

$$Sensitivity = \frac{TP}{TP+FN}\text{; } FN = False\ Negatives$$

False negatives are defined as edges in the known network that are not detected in the inferred network.

CHAPTER 5

RESULTS

To assess the viability of the methods discussed in the previous chapter, structure learning is performed on observational datasets pooled from a number of expert-specified or synthetic networks native to the Bayes Net Toolbox and Structure Learning Package, the Causal Explorer supplementary data archive, or available from several on-line Bayesian network repositories.[28-30] Networks obtained from the repositories are provided in a java-based format, and are subsequently converted to a Matlab-compatible representation using software written by Ken Shan.[31] For each network 10 experimental datasets are taken for consistency, then used as the basis for inference. The metrics computed are the average of the performance across all learned networks. The specifics of each network are discussed in more detail below.

Small Networks

Asia

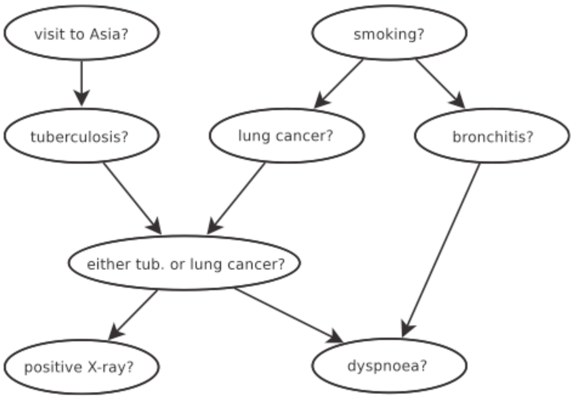


Figure 4: The Asia network[10]

The Asia network introduced by Lauritzen and Spiegelhalter is a small, fictional network depicting related variables in a medical diagnosis.[32] The network contains 8 nodes and 8 edges, with all nodes being binary. This network is not learned by the committee approach, since its size lies well below the range targeted by the committee. Instead, the small network size is used to emphasize that the performance of structure learning algorithms is lacking, even when presented the task of learning a small structure. In earlier work, Leray and Francois show only the Greedy Equivalence Search algorithm accurately reproduces the Asia network, requiring greater than 5000 observations.[17] The sub-optimal performance of structure learning algorithms on small networks, especially at a small number of observations, suggests that an alternative approach must be used to accurately and confidently learn structure in larger networks, which in turn motivates the use of a committee of algorithms.

Medium Networks

Insurance

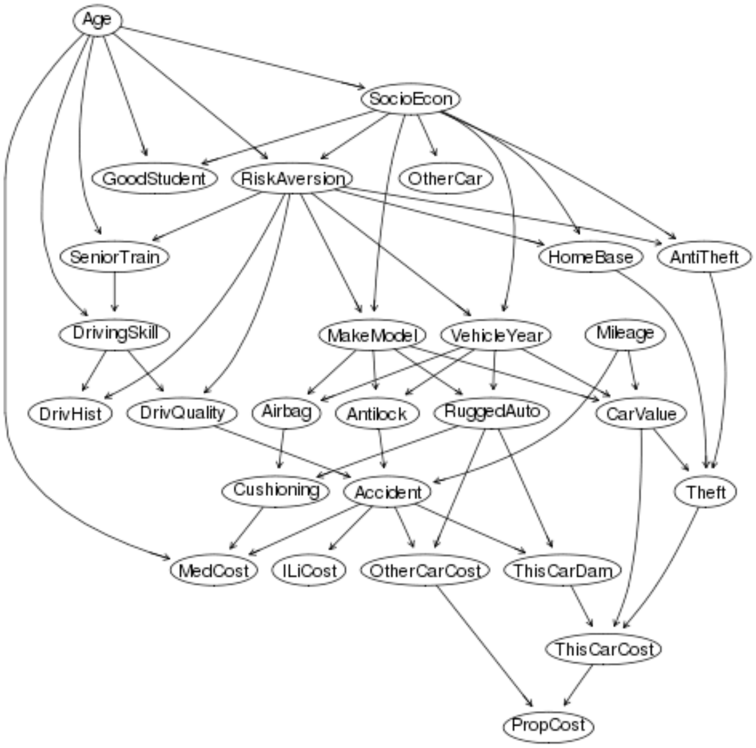


Figure 5: The insurance network

Prior to applying the committee to large-scale networks, the concept was first implemented on a network of moderate size to confirm its validity and offer insight towards necessary adjustments before proceeding to larger structures. The insurance network, consisting of 27 nodes and 52 edges, was used for this task. Nodes can take on either two or three values. The committee is tasked with learning the structure for 10 datasets each at 50, 100, and 250 observations. These sample sizes are sufficiently small to fall below the optimal range of any single algorithm, and therefore expected to elicit a meaningful contribution from the committee. Figure 6 and Figure 7 show the performance of the committee on learning the direction independent insurance network.

The results show that the intersection and majority graphs exhibit excellent selectivity, but are slower to recover edges. At 250 observations, only half of the edges from the Insurance graph are present in the intersection graph. The PC algorithm, while exhibiting the best recall of edges, has the lowest selectivity of any algorithm. Claims in the literature suggest that PC is sub-optimal for learning with small sample sizes.[9, 24] This helps to explain the poor performance illustrated throughout the study by PC, despite promising performance with excellent selectivity in earlier trials in BNT.

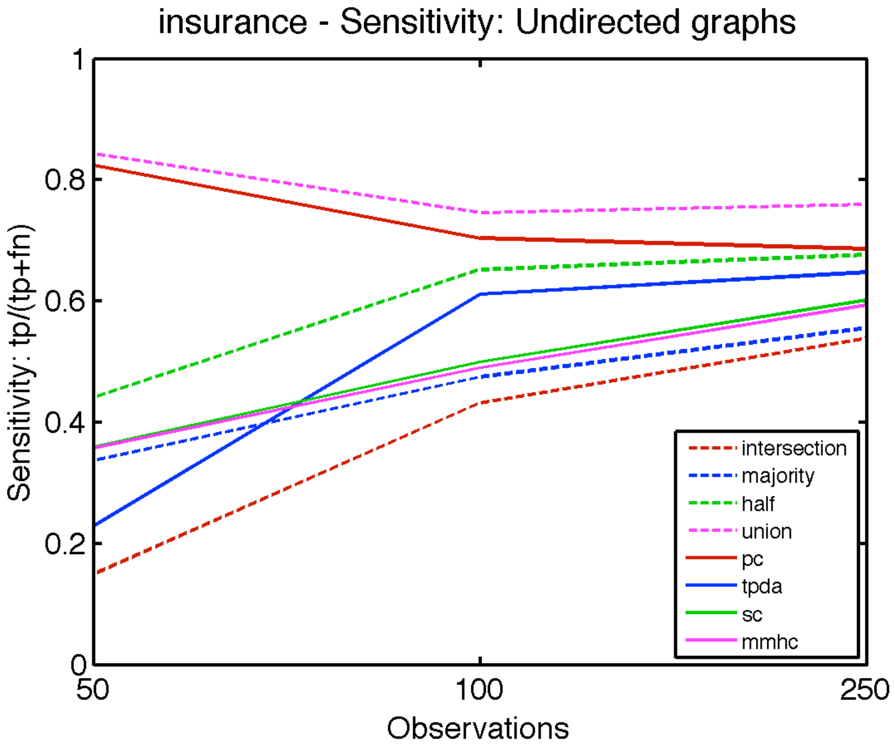


Figure 6: Sensitivity of algorithms learning Insurance network

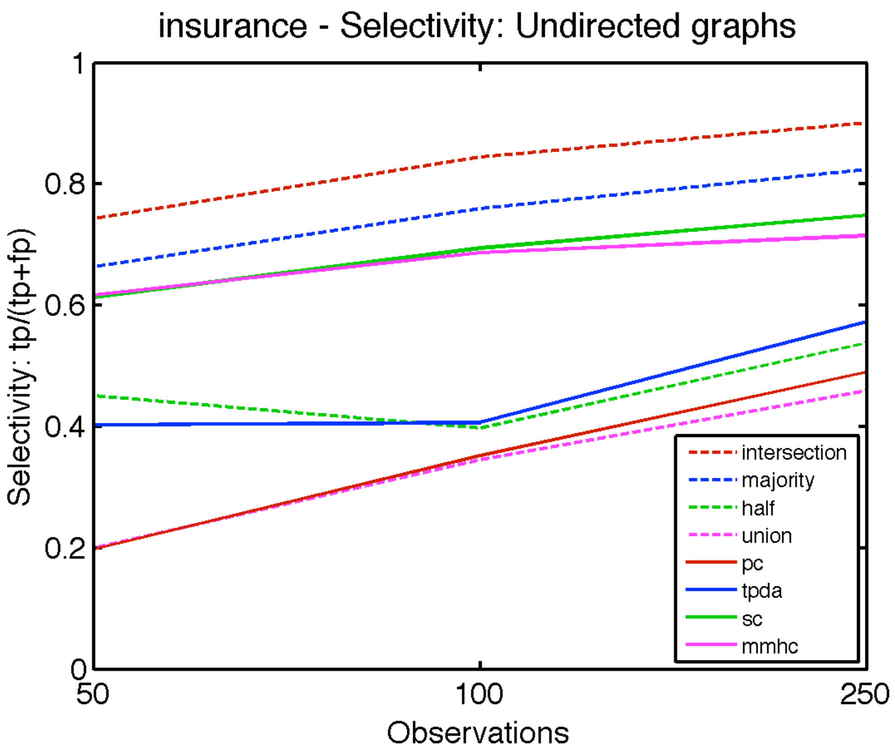


Figure 7: Selectivity of algorithms learning Insurance network

Large Networks

Full networks

Alarm

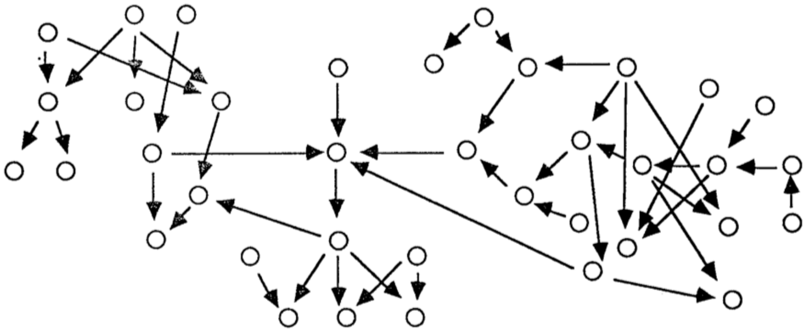


Figure 8: Original schematic of ALARM monitoring network[33]

The ALARM monitoring network is an expert medical network used to monitor intensive care patients, as described by Beinlich et al.[33] ALARM consists of 37 nodes and 46 edges, with node sizes ranging from 2-5. The network is commonly used as a benchmark to test new algorithms, so it is a good initial attempt for the committee before tackling more complex networks. The committee learned this network at four different sample sizes, with the results of all four metrics shown here.

From the editing distance plot (Figure 9), the most obvious conclusion is that the PC algorithm performs extremely poorly at lower sample size, but begins to approach the performance of the other algorithms as the sample increases. Figure 10 shows the selectivity of the intersection and majority graphs to be high, with the Sparse Candidate and MMHC also performing reasonably well. Both PC and TPDA show a rapid increase in selectivity as the sample size rises to 500. All graphs, including the intersection, show a sensitivity above 65% at 100 observations or greater. Figure 11 shows the graph of sensitivity. The performance of the committee in learning the ALARM network is promising.

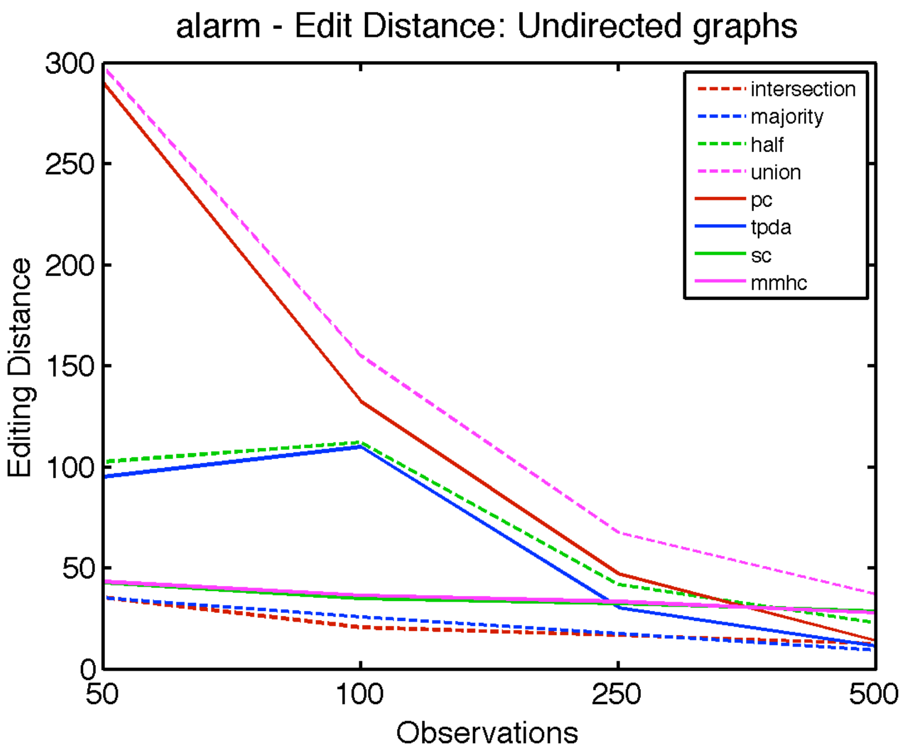


Figure 9: Edit distance of ALARM network

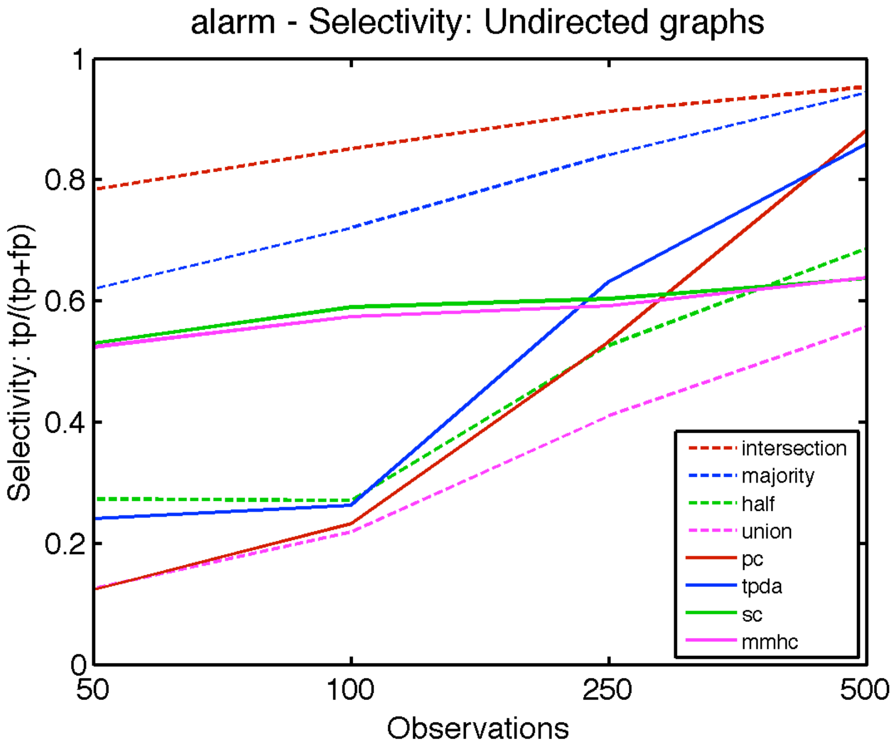


Figure 10: Selectivity of ALARM network

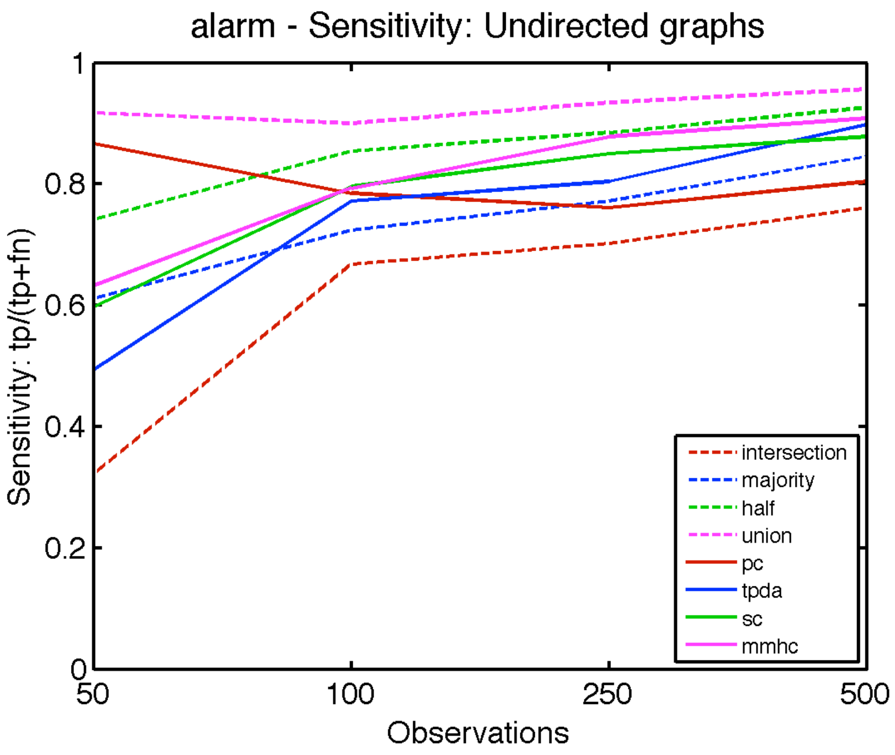


Figure 11: Sensitivity of ALARM network

Mildew

The Mildew network is another network of size similar to ALARM, with 35 nodes and 46 edges.[34] Node sizes range from 3-100 in this network. Due to the limited number of observations relative to the number of values a node can assume, the performance of the algorithms in learning this network is negatively impacted. A more detailed discussion of the impact of node size on learning network structure is presented below, in the discussion of Reduced Networks. No data is shown for 500 observations since the algorithms did not converge. No intersection graph exists for the 50 observations case, since TPDA failed to learn any edges from the data. Figure 12 confirms the hypothesis that, even when structure learning is extremely poor, the intersection graph remains the most selective. The performance remains disappointing, as the sensitivity (Figure 13) remains under 20% up to 250 observations. Even MMHC and

Sparse Candidate, who illustrate selectivity around 80% only show sensitivity at 40% in this case. This raises an interesting question of how to choose a model to accept. In this case, the preferable choice is an individual algorithm. However, in a real experiment, a lack of domain knowledge and inconsistent performance of the single-algorithms exhibited across different networks raises concerns as to the efficacy of this approach. The overall recommendation is to avoid temptation to use single-algorithm methods, since they cannot be verified to perform well across a range of tested networks.

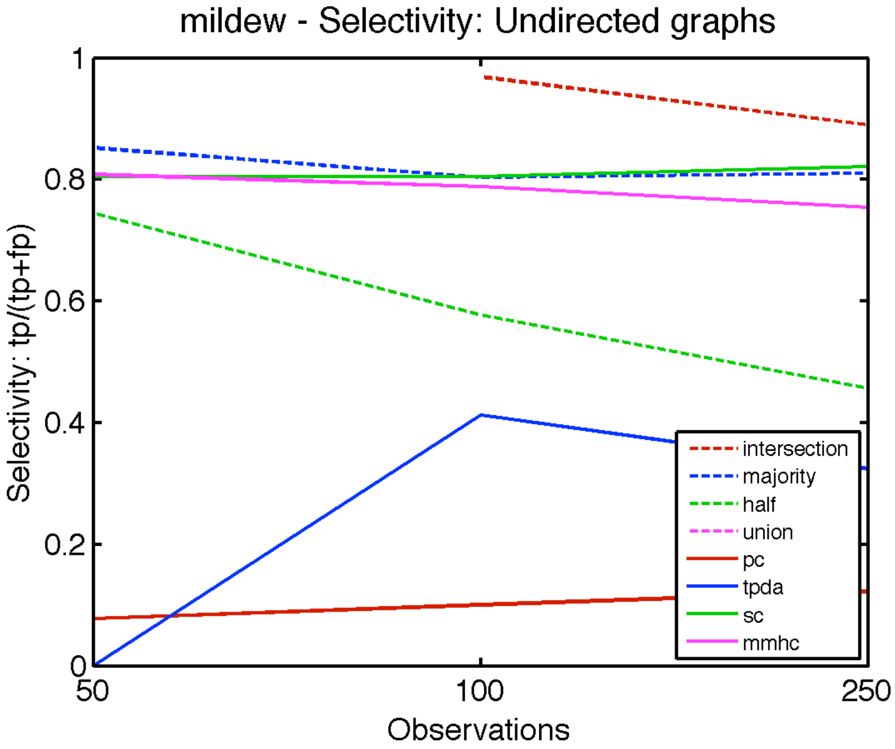


Figure 12: Selectivity for Mildew network

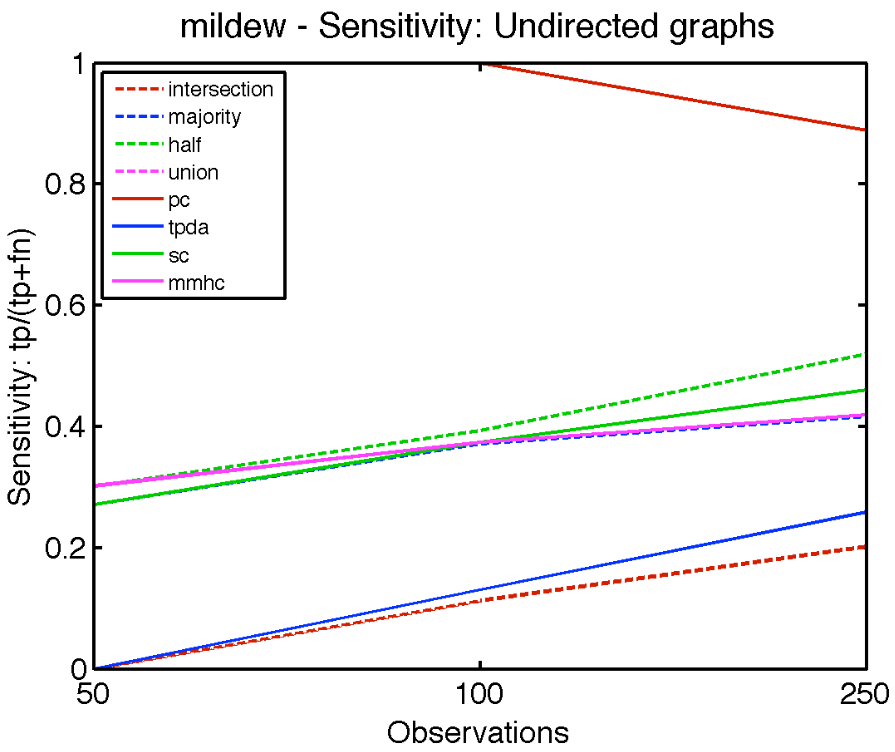


Figure 13: Sensitivity for Mildew network

Barley

Another example of comparable size is the Barley network. The network consists of 48 nodes, 84 edges.[35] In this case, performance is extremely poor. At best, the selective majority and intersection methods have sensitivity below 40% at 250 observations. The poor performance is thought to be an artifact of two phenomena: first, the Barley network contains node sizes ranging from 2-67 (discussed in the Reduced Network example), and second, the number of edges is close to twice that of nodes, meaning the network is denser than the others studied. Some algorithms have difficulty learning domains where dense hubs exist. The effect of edge density cannot be ignored, since both MMHC and Sparse Candidate are presented to be optimal at learning sparse domains.

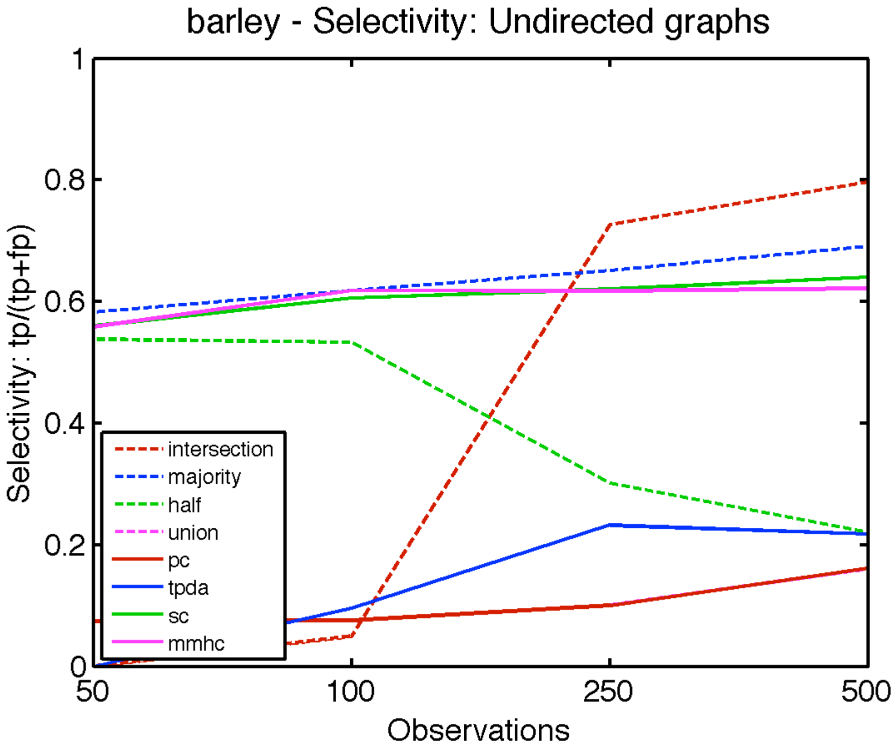


Figure 14: Selectivity for Barley network learned from Causal Explorer supplementary data

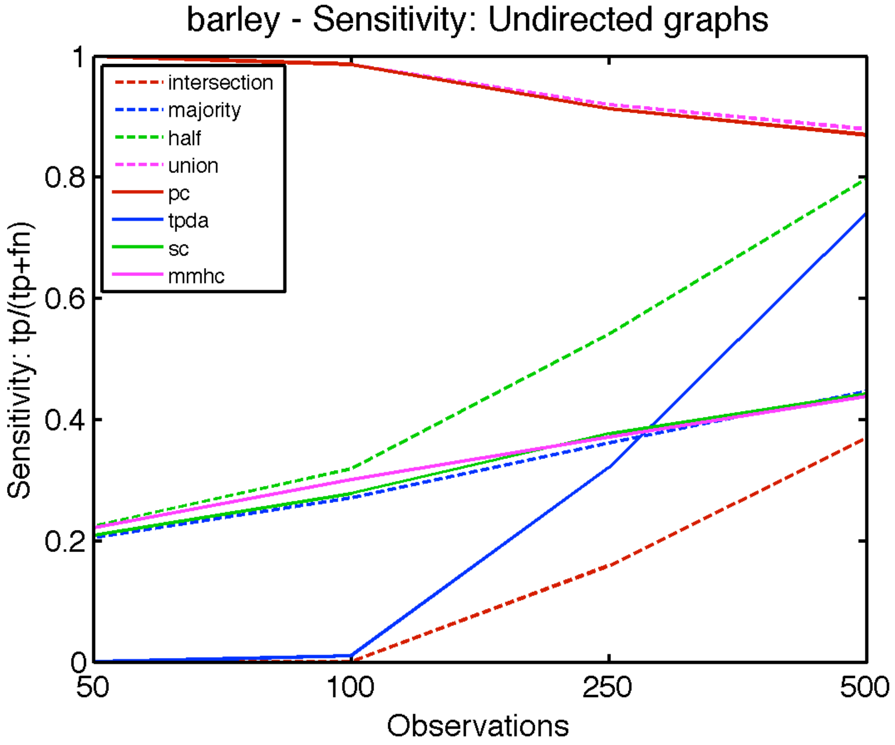


Figure 15: Sensitivity for algorithms learning Barley network learned from Causal Explorer supplementary data

Hailfinder

The Hailfinder network used to forecast severe weather is another example of a domain where large node sizes are suspected of having an effect on algorithm performance.[36] The network has 56 nodes, 66 edges, and node sizes ranging from 2-11. The selectivity and sensitivity plots are shown in Figure 16 and Figure 17, respectively. In this case, the selectivity of the intersection graph is near 80%, but its sensitivity remains between 20-40% for all trials. The majority graph exhibits both selectivity and sensitivity at or above 60% for 250 observations, suggesting that committee approaches can perform at a reasonable level. In the next section, a solution to the suspected node-size problem is presented.

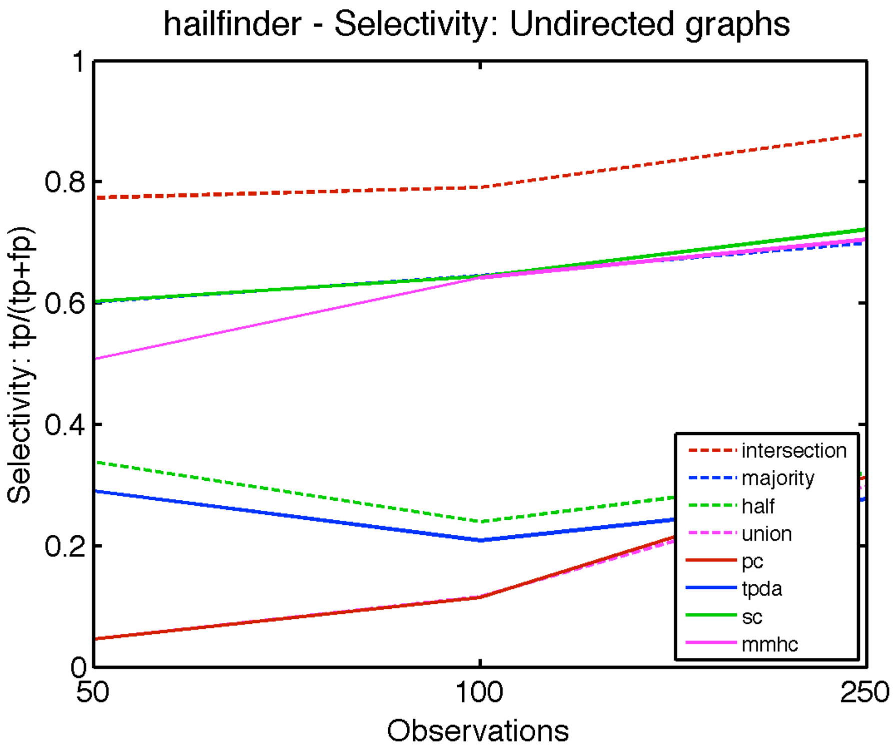


Figure 16: Selectivity of Hailfinder graph learned from sampled data

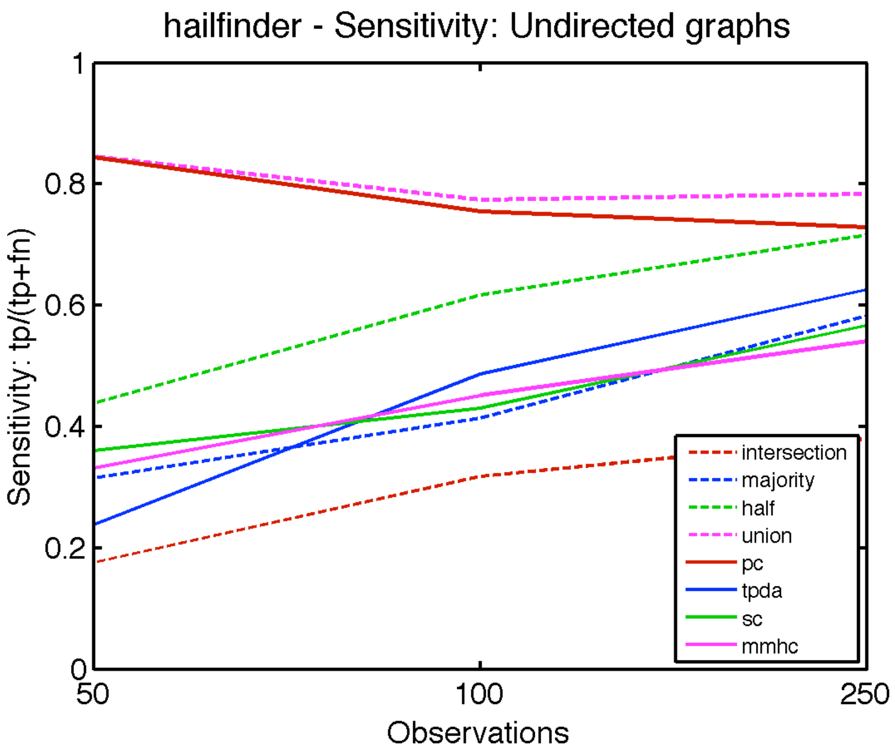


Figure 17: Sensitivity of Hailfinder graphs learned from sampled data

Reduced networks

Poor performance in learning several large-scale networks and the interest of learning sub-networks from very large domains pose an interesting problem. A first attempt to address this issue is implemented by producing “reduced networks”. Reduced networks are generated by eliminating excessive or problematic nodes, along with any associated edges in the model graph and entries in the observational data matrix. The resulting reduced data is then learned by the committee of algorithms; inferred models are assessed relative to a similarly reduced known graph, and compared to the performance metrics of the same network prior to reduction. This heuristic is naïve, as it does not consider the ramifications of removing certain nodes and edges from the network and their corresponding observations. However, if problematic nodes exist in real datasets, this approach is the only way to eliminate them. Also, large-scale networks tend to be sparse, and therefore, eliminating nodes by this method is unlikely to have a dramatic effect on learning their structure.

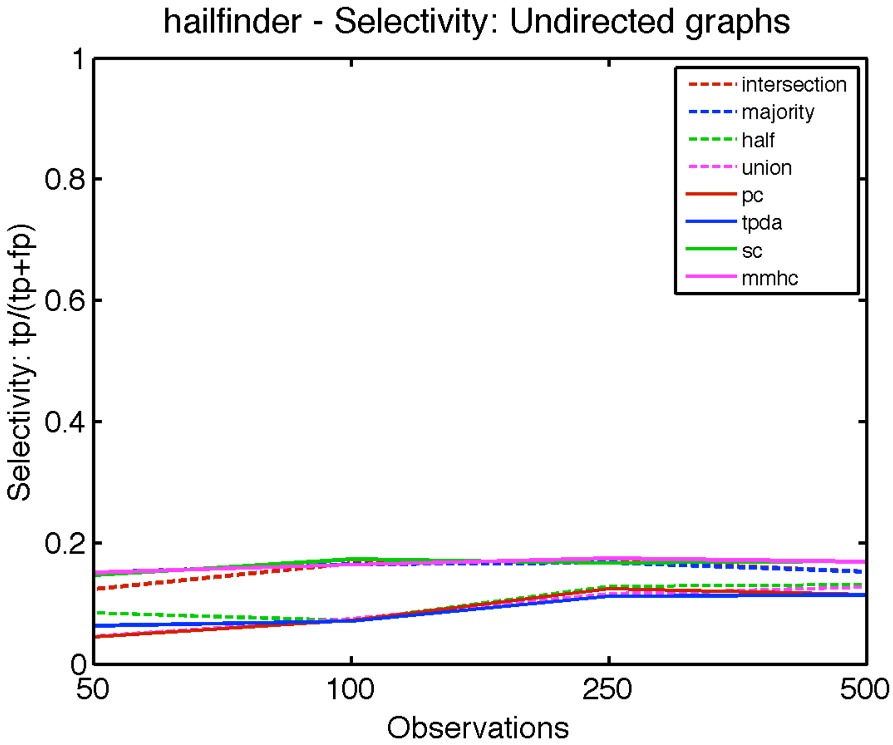


Figure 18: Selectivity of Reduced Hailfinder graphs

This study considered two types of reduced network. The first type is generated from the large Hailfinder and Barley networks by restricting the node sizes to four and ten respectively. The reduced Hailfinder network is a significantly smaller network, shrinking from 56 to 40 nodes. The reduced Barley graph only drops from 48 to 45 nodes. While a node size of 10 is not desirable for this approach, most nodes in the Barley network are of similar value, so restricting below that will drastically impact the size of the reduced network. The rationale behind eliminating nodes capable of taking on a large number of states is tied to the restricted nature of this study. The goal is to characterize algorithm performance under small observations. In a small number of observations, the network is less likely to exhibit a range of states sufficient for accurate learning. The presence of larger nodes only compounds this issue; for example, a certain node in the original Barley network can take on 67 values. A node of this type cannot be adequately observed within a small sample size so as to have edges accurately inferred, and is therefore removed in this modification.

In addition to removing problematic nodes from networks, the reduction method was also used to generate a "reduced link[37]" sub-network from massive networks. This network is a 40-node sub-network taken from the original Link network (an extremely large network consisting of 724 nodes and 1125 edges). The motivation to create these sub-networks is twofold: first, they provide more suitable-sized networks on which structure learning algorithms can be tested, and second, they allow for learning the edges among a small subset of the data. While this deviates from the original application of reduced networks, it offers a solution to learn sub-spaces when networks are too large to infer fully. The performance of the reduced link network is shown in Figure 20 and Figure 21. Again, the majority and intersection graphs are most selective, but remain under 60% sensitive.

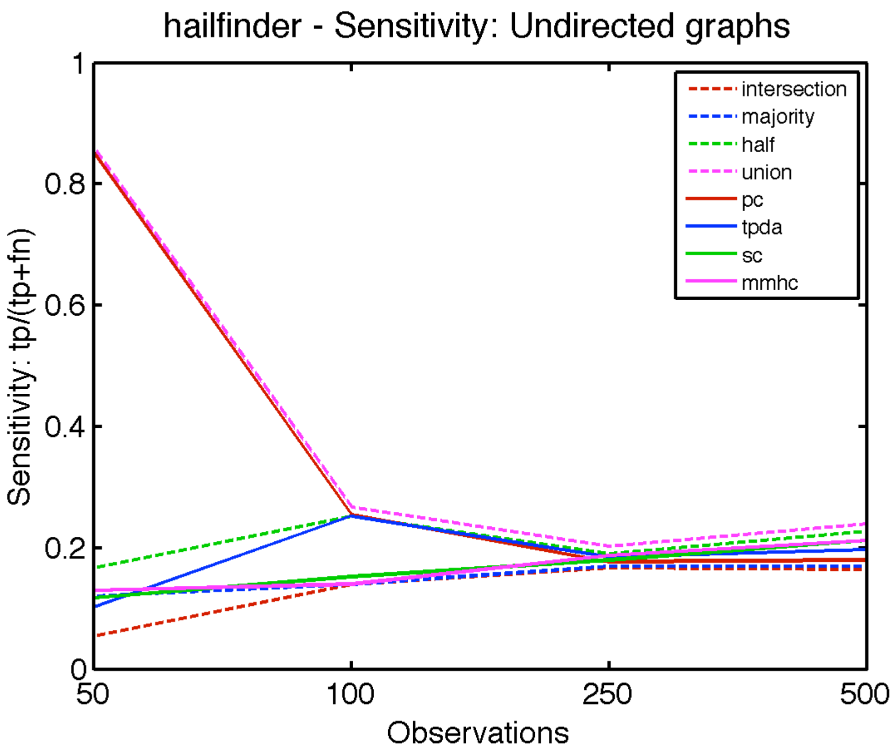


Figure 19: Sensitivity of Reduced Hailfinder graphs

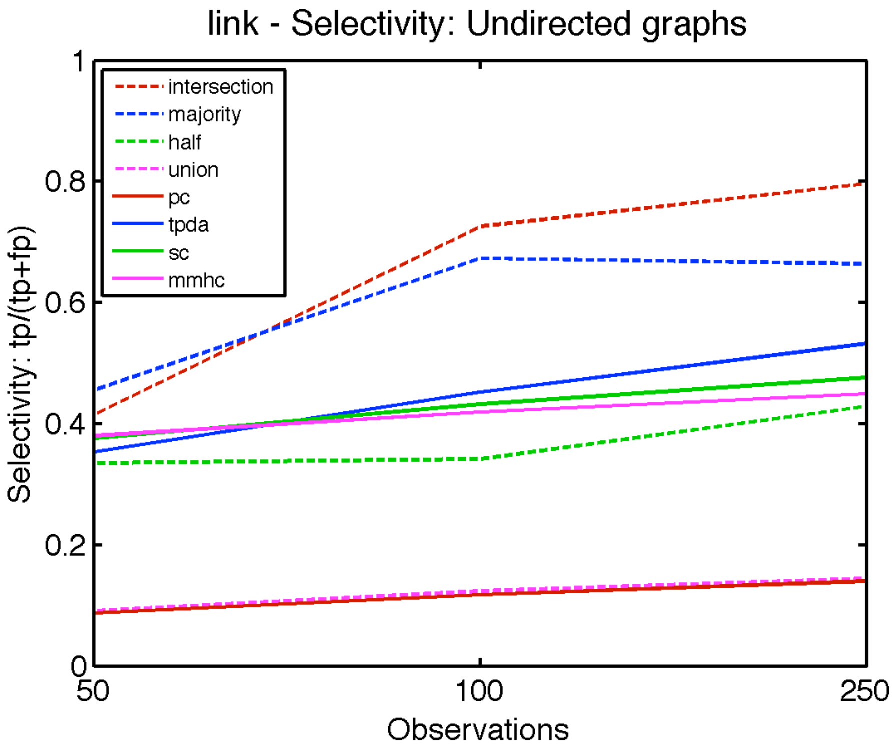


Figure 20: Selectivity of Reduced Link graphs

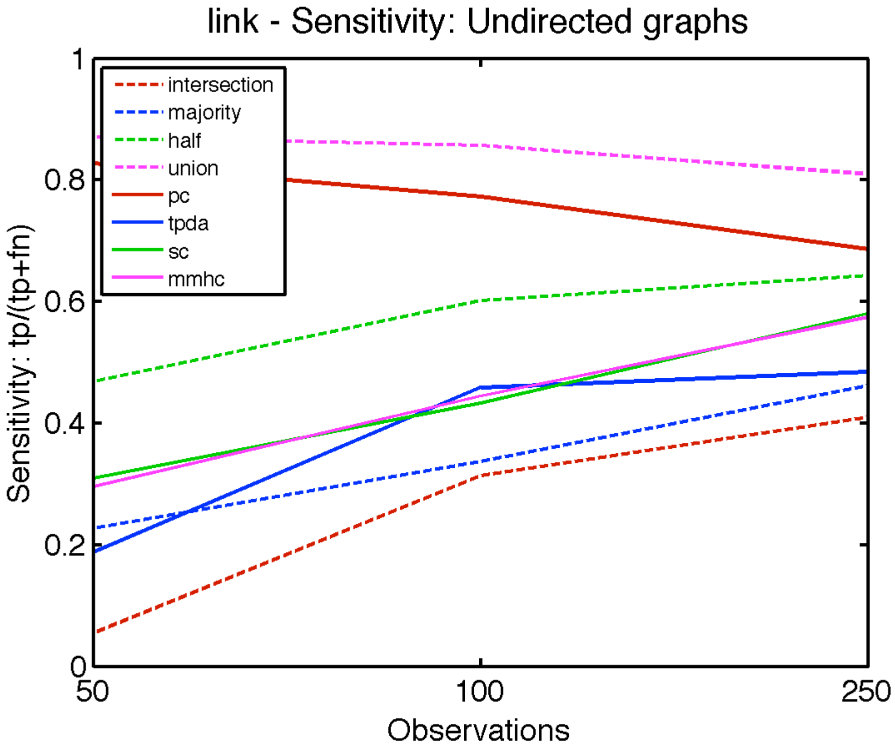


Figure 21: Sensitivity of Reduced Link graphs

Curated Networks

Drawing from the same motivation as the "reduced network" approach, another size reduction method was employed. This method, called the "curated" approach, attempts to more accurately depict network behavior when nodes and edges are eliminated. In this case, instead of simply deleting entries from the observational data table, the network Conditional Probability Distributions are changed where edges previously affecting a node are absent. For cases of simplicity, any replaced values are assumed to be of uniform probability for any state the variable can assume. Data was then re-sampled from the new network, and the committee was reassigned to learn the data. The performance on the curated network shown in Figure 22 and Figure 23 is both more selective and more sensitive than that expressed in the "reduced" example. Further, to bolster the argument that node sizes play an effect in structure learning performance, the results are compared to the graphs learned from data sampled for the original network. The sensitivity of the intersection graph is observed to jump from under 40% in the previous instance at 250 observations to slightly under 55% for the curated case, while maintaining selectivity above 80%.

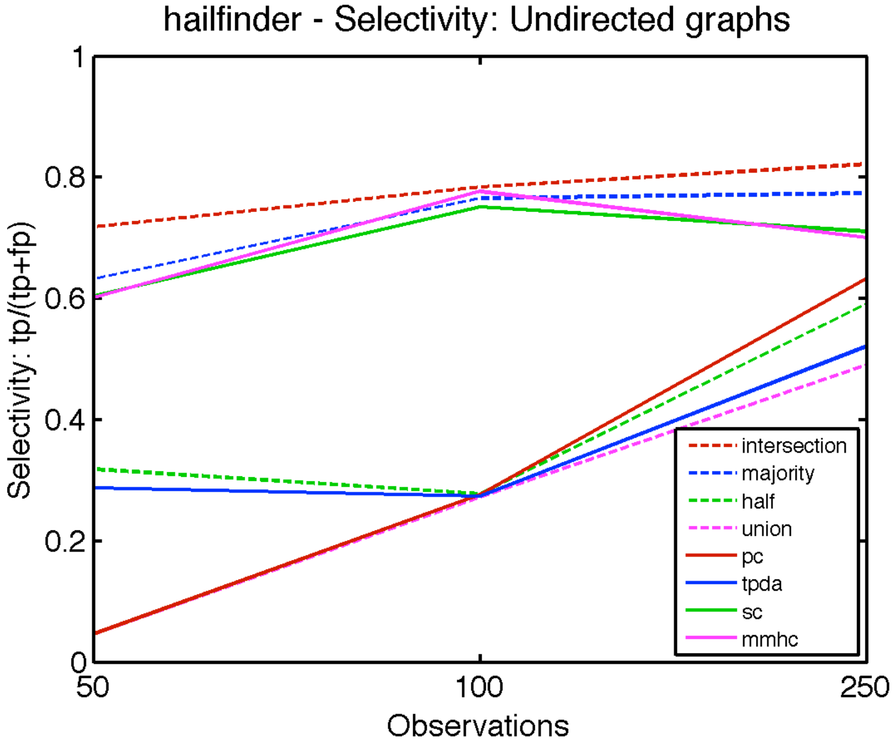


Figure 22: Selectivity of Curated Hailfinder graphs

Also presented in Figure 24 and Figure 25 are results from learning graphs of a curated and re-sampled sub-network of the Andes network (described below).[38] The performance of the intersection graph is notably exceptional, with sensitivity just under 60% at 100 observations, and selectivity above 80%.

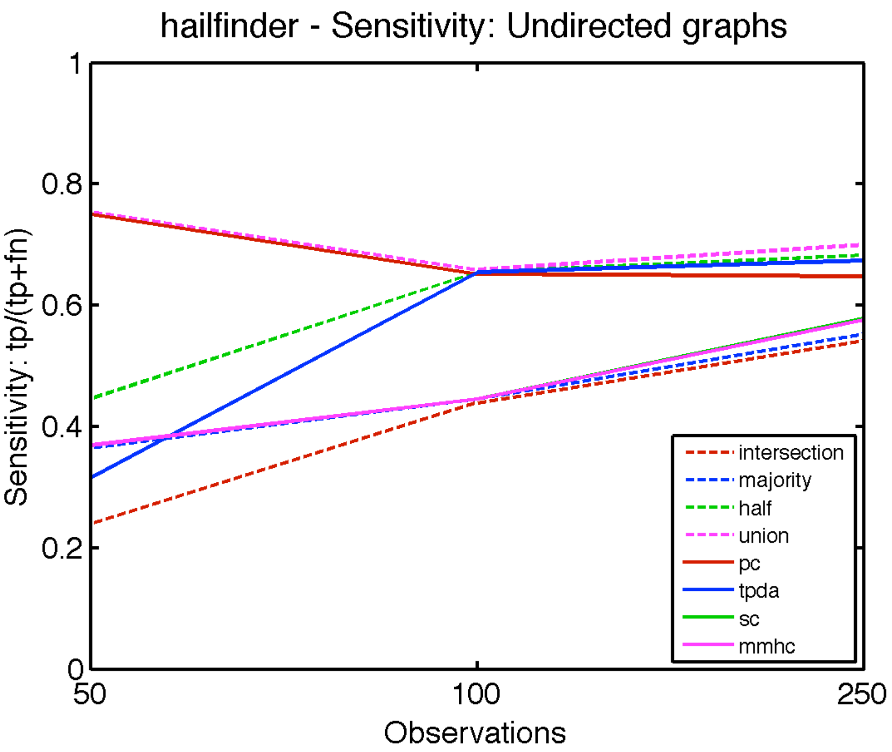


Figure 23: Sensitivity of Curated Hailfinder graphs

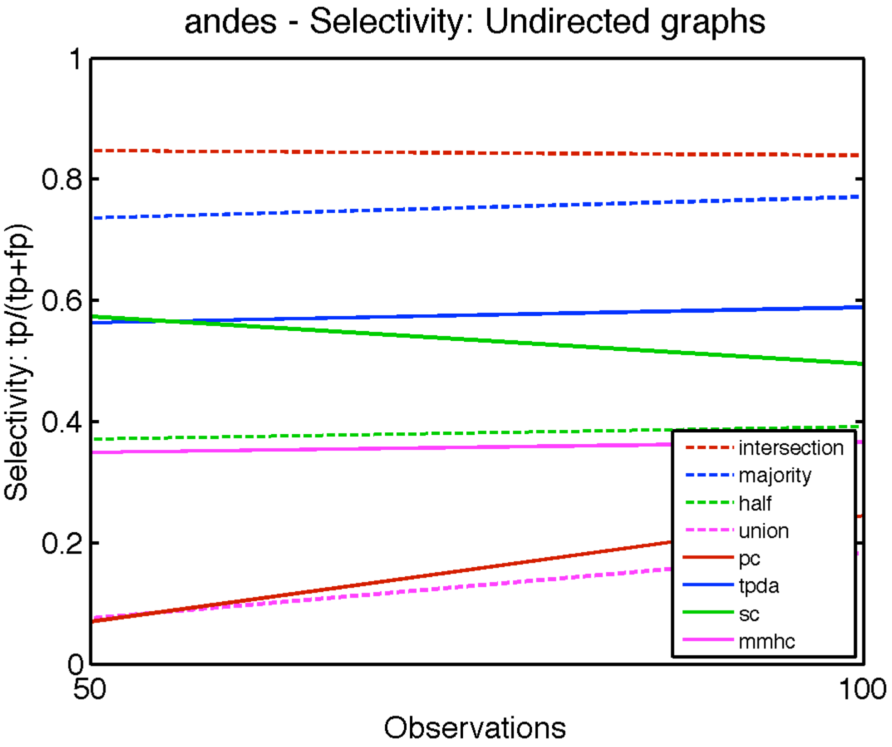


Figure 24: Selectivity of Curated Andes graph

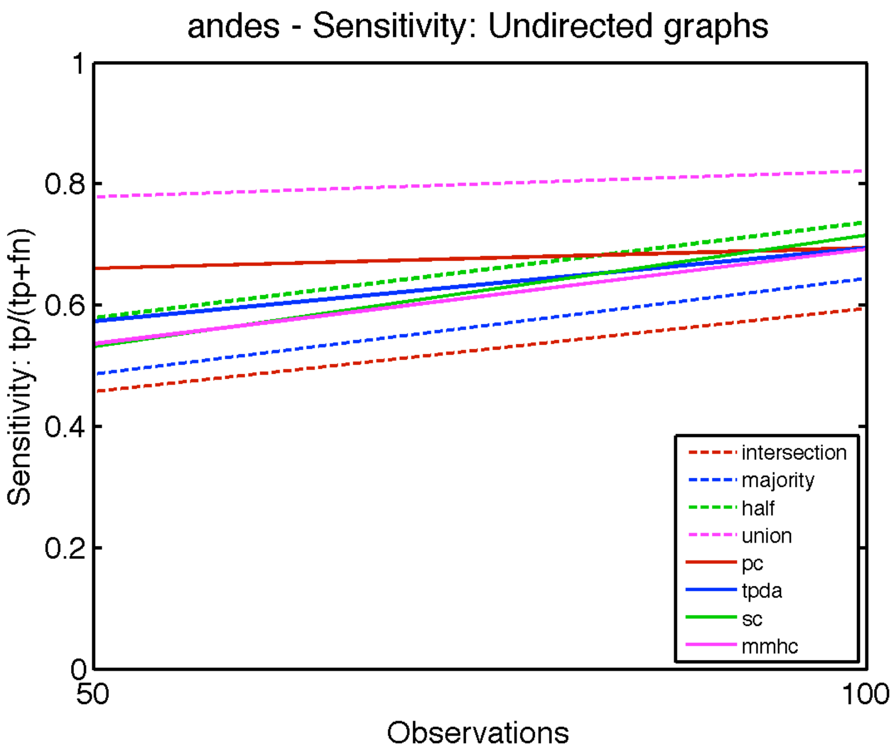


Figure 25: Sensitivity of Curated Andes graph

Very Large Networks

The ultimate goal of the committee-based approach to structure learning was to improve performance on very large networks, with hundreds of variables. The result was not realized however, due to the inability of the algorithms to all converge in these cases. While MMHC and Sparse Candidate have been developed specifically for the task of large-network discovery, the goal was to improve their accuracy by the use of the voting approach presented in this work. After two weeks of simulation, only the 50-observation instance of the Andes network, consisting of 223 nodes and 338 edges, using default parameters for all algorithms managed to converge. The results are presented in Figure 31and Figure 32. In this case, only MMHC and PC converged, rendering the committee unusable. Conclusions regarding the use of these methods and suggestions for future improvements are discussed in the coming chapters.

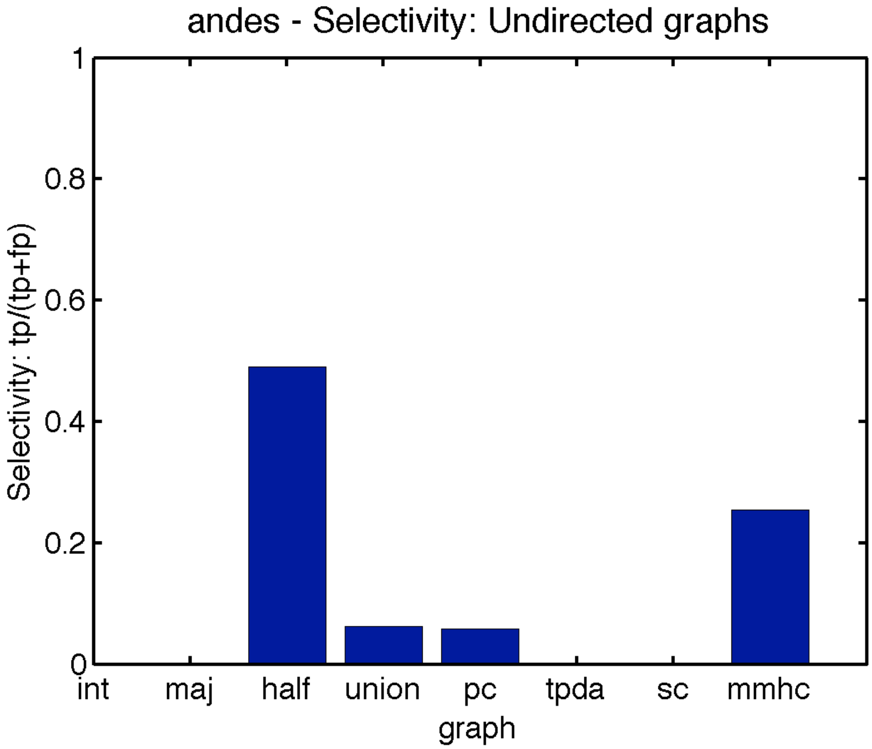


Figure 26: Selectivity of Andes graphs

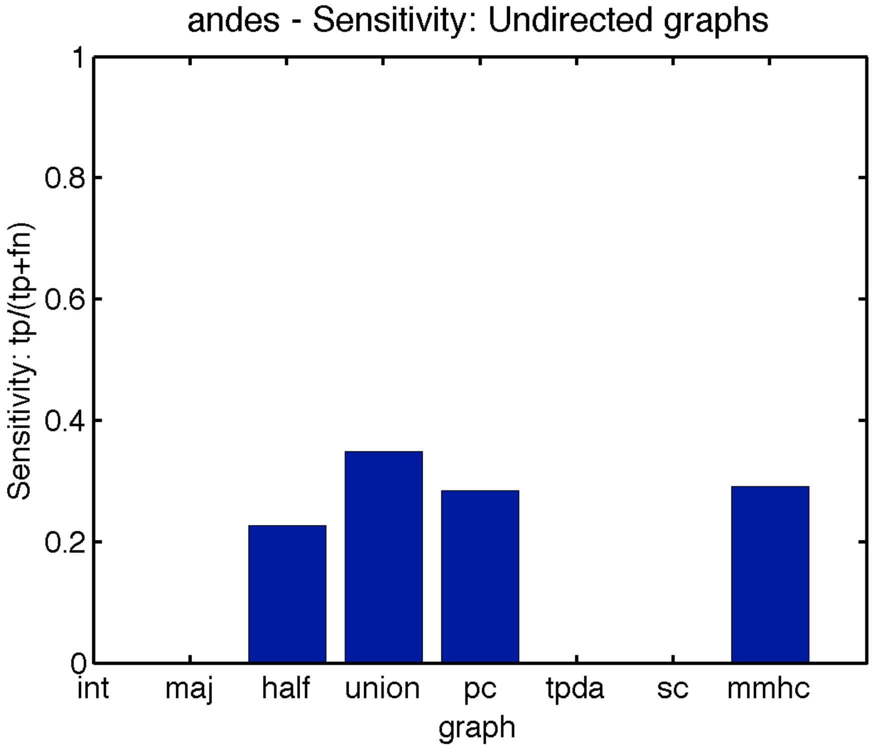


Figure 27: Sensitivity of Andes graphs

CHAPTER 6

CONCLUSIONS

Ensemble Methods in Network Inference

This work illustrates the use of ensemble approaches to infer network structure where use of a single algorithm is unreliable. The improved selectivity is undoubtedly shown for all tested networks. Sacrifices of sensitivity are non-negligible, however. Any three or four vote network learned by a committee is therefore likely to be missing more than half of its edges. The edges present in the inferred model can however be accepted with a strong degree of confidence. This voted approach improves the selectivity of the algorithms for all observation sizes studied. However, in some cases the performance significantly improves between 50 and 100 observations. Due to this inconsistency, learning network structure below 100 observations is not recommended.

With the exception of MMHC and Sparse Candidate for some networks, the selectivity of the majority and intersection graphs greatly outperforms the individual algorithms. Even when the selectivity of the algorithms approaches that of the committee graphs, the latter are still more selective. With respect to sensitivity, the committee performance is lower than that of any individual algorithm. The use of the committee is still recommended, however, as the PC and TPDA algorithms often exhibit high sensitivity with poor selectivity.

Learning Large-Scale Networks

The task of learning large-scale networks remains a difficult task. State-of-the-art algorithms designed specifically for this function, such as MMHC and Sparse Candidate, begin to exhibit long computational times, and in some cases fail to converge. While reasonable conclusions regarding the use of a voting committee can be drawn for networks of approximately 40-60 nodes, it is difficult to properly assess the contribution of a committee for networks of hundreds of variables, when there are so few algorithms capable of learning such large networks, and even these struggle to do so efficiently. Forming a reliable committee for learning such large networks is a prerequisite for further analysis.

CHAPTER 7

RECOMMENDATIONS

Learning Large Networks

At this time, committee approaches are not recommended for learning massive networks. This is not due to any flaws in the concept of ensemble learning. Rather, problems arise due to the lack of suitable algorithms to form the ensemble. Current effective algorithms, like MMHC and Sparse Candidate, are too similar in their underlying theory, and are therefore subject to the same biases and potential errors. The authors describe MMHC as a specific instantiation of Sparse Candidate. Until newer and more diverse algorithms capable of large-scale network discovery are developed, ensemble approaches should be avoided.

Following from the discussion of the curated Andes and reduced Link networks in Chapter 5, more study towards learning sub-networks is recommended. Since partitioning a network at random is likely to miss network-wide interactions, an approach randomly selecting sub-networks, learning their structure, and averaging the resulting models for a large number of trials is one potential solution. The task of learning these sub-networks is less computationally expensive than attempting to learn the structure of massive networks, especially when convergence is not guaranteed.

Improving Algorithms and the Committee

Further work in the area of Bayesian Structure learning should focus on improving algorithms for learning large networks. Leading algorithms such as MMHC and Sparse Candidate were developed for applications in studying gene regulatory networks, which tend to be mostly sparse. The algorithms have difficulty learning domains where dense hubs exist. Improvement in the area would provide a more robust tool, applicable across multiple domains. One potentially promising example is the Empirical Light Mutual Min (ELMM) algorithm, which is designed for learning denser networks than MMHC and Sparse Candidate can learn.[39] ELMM learns conditional independence graphs, instead of DAGs, which are more interesting in the context of this study, since the notion of directionality and network scoring are ignored be the committee. Another method to consider is the local committee-based method proposed by Mwebaze and Quinn, which uses a set of algorithms to vote on each edge during the inference process, rather than after each algorithm completely learns the network as done in this work.[23]

The PC algorithm notably breaks down for very large networks, and exhibits poor performance throughout the study due to the small number of observations. Both of these factors suggest PC should not be used in large-scale network discovery, or a voting committee.

Applications in Learning Complex Systems

The use of voting approaches, provided systems of interest are of reasonable size, can greatly assist in the study of complex systems interactions. Recommendations are made for the use of voting applied on systems under 100 variables in size. Despite the earlier suggestion to replace the PC algorithm, for these cases, it should be kept, as it is by far the most sensitive algorithm available for a small number of observations, and the committee easily handles the high number of false positives.

REFERENCES

1. Friedman, N., Linial, M., Nachman, I. & Pe'er, D. Using Bayesian networks to analyze expression data. *Journal of Computational Biology* **7**, 601-620 (2000).
2. Correa, E. & Goodacre, R. A genetic algorithm-Bayesian network approach for the analysis of metabolomics and spectroscopic data: application to the rapid identification of Bacillus spores and classification of Bacillus species. *Bmc Bioinformatics* **12** (2011).
3. Diez, F.J., Mira, J., Iturralde, E. & Zubillaga, S. DIAVAL, a Bayesian expert system for echocardiography. *Artificial Intelligence in Medicine* **10**, 59-73 (1997).
4. de Campos, L.M., Fernandez-Luna, J.M. & Huete, J.F. Bayesian networks and information retrieval: an introduction to the special issue. *Information Processing & Management* **40**, 727-733 (2004).
5. Pearl, J. *Probabilistic reasoning in intelligent systems*. (Morgan Kaufmann, San Mateo, CA; 1988).
6. Pearl, J. *Causality, models, reasoning, and inference*. (Cambridge University Press, 2000).
7. Spirtes, P., Glymor, C. & Scheines, R. *Causation, Prediction, and Search*, Edn. 2. (MIT Press, 2000).
8. Friedman, N., Nachman, I. & Peer, D. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. *Uncertainty in Artificial Intelligence, Proceedings*, 206-215 (1999).
9. Tsamardinos, I., Brown, L.E. & Aliferis, C.F. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning* **65**, 31-78 (2006).
10. Daly, R., Shen, Q. & Aitken, S. Learning Bayesian networks: approaches and issues. *Knowledge Engineering Review* **26**, 99-157 (2011).
11. Pearl, J. & Verma, T.S. A theory of inferred causation. *Principles of Knowledge Representation and Reasoning: Proc. Second International Conference*, 441-452 (1991).
12. Verma, T.S. & Pearl, J. Equivalence and Synthesis of Causal Models. *Uncertainty in Artificial Intelligence* **6**, 255-268 (1991).
13. Cheng, J. & Griener, R. Learning Bayesian Belief Network Classifiers: Algorithms and System. *Canadian Conference on AI*, 141-151 (2001).
14. Chickering, D.M. Learning Bayesian Networks is NP-Complete, in *Learning from Data: Artificial Intelligence and Statistics V*. (ed. D.F.a.H.J. Lenz) 121-130 (Springer-Verlag, 1996).
15. Chickering, D.W., Heckerman, D. & Meek, C. Large-Sample Learning of Bayesian Networks is NP-Hard. *Journal of Machine Learning Research* **5**, 1287-1330 (2004).
16. Murphy, K. The Bayes Net Toolbox for Matlab. *Computing Science and Statistics* **33** (2001).
17. Leray, P. & Francois, O. BNT Structure Learning Package: Documentation and Experiments. (2004).

18. Heckerman, D. A tutorial on learning with Bayesian networks. *Learning in Graphical Models* **89**, 301-354 (1998).
19. Aliferis, C.F., Tsamardinos, I., Statnikov, A.R. & Brown, L.E. Causal explorer: A causal probabilistic network learning toolkit for biomedical discovery. *Metmbs'03: Proceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences*, 371-376 (2003).
20. Breiman, L. Bagging Predictors. *Machine Learning* **24**, 123-140 (1996).
21. Maclin, R. & Opitz, D. in The Fourteenth National Conference on Artificial Intelligence (AAAI Press, Providence, Rhode Island; 1997).
22. Quinlan, J.R. Bagging, boosting, and C4.5. *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference, Vols 1 and 2*, 725-730 (1996).
23. Mwebaze, E. & Quinn, J. Fast Committee-Based Structure Learning. *JMLR Workshop and Conference Proceedings* **6**, 203-214 (2008).
24. Spirtes, P., Glymor, C. & Scheines, R. *Causality, Prediction, and Search*, Vol. 81, Edn. 1. (Springer, 1993).
25. Dor, D. & Tarsi, M. A simple algorithm to construct a consistent extension of a partially oriented graph. *Technical Report R-185* (1992).
26. Verma, T.S. A linear-time algorithm for finding a consistent expansion of a partially oriented graph. *Technical Report R-180* (1992).
27. Cheng, J., Greiner, R., Kelly, J., Bell, D. & Liu, W.R. Learning Bayesian networks from data: An information-theory based approach. *Artif. Intell.* **137**, 43-90 (2002).
28. Elidan, G. Bayesian Network Repository. http://www.cs.huji.ac.il/site/labs/compbio/Repository/ (2001).
29. Scutari, M. Bayesian Network Repository. http://www.bnlearn.com/bnrepository/ (2010).
30. Tsamardinos, I., Brown, L.E. & Aliferis, C.F. Supplemental Appendices. http://www.dsl-lab.org/supplements/mmhc_paper/mmhc_index.html (2007).
31. Shan, K. bif2bnt: BIF-BNT converter. http://www.digitas.harvard.edu/~ken/bif2bnt/ (2000).
32. Lauritzen, S.L. & Spiegelhalter, D.J. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society: : Series B (Statistical Methodology)* **50**, 157-224 (1988).
33. Beinlich, I.A., Suermondt, H.J., Chavez, R.M. & Cooper, G.F. The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. *2nd European Conference of Artificial Intelligence in Medicine*, 247-256 (1989).
34. Jensen, A.L. A probabilistic model based decision support system for mildew management in winter wheat. *Dina Research Report* **39** (1995).
35. Kiristensen, K. & Rasmussen, I.A. Production of beer from Danish malting barley grown without the use of pesticides.
36. Abramson, B., Brown, J., Edwards, W., Murphy, A. & Winkler, R.L. Hailfinder: A Bayesian system for forecasting severe weather. *International Journal of Forecasting* **12**, 57-71 (1996).

37. Jensen, C.S. & Kong, A. Blocking Gibbs Sampling for Linkage Analysis in Large Pedigrees with Many Loops. *The American Journal of Human Genetics* **65**, 885-901 (1999).
38. Conati, C., Gertner, A.S., VanLehn, K. & Druzdzel, M.J. in 6th International Conference on User Modeling 231-242 (Springer-Verlag, 1997).
39. Mahdi, R. *et al.* Empirical Bayes conditional independence graphs for regulatory network recovery. *Bioinformatics* **28**, 2029-2036 (2012).