Learning Distributed Bayesian Network Structure Using Majority-based Method

Sachin Shetty
Rowan University
Glassboro, NJ 08028, USA
Phone: 856-256-5379
Fax: 856-256-5241
Email: shetty@rowan.edu

Min Song
Old Dominion University
Norfolk, VA 23529, USA
Phone: 757-683-5194
Fax: 757-683-3220
msong@odu.edu

Houjun Yang
Qingdao University
Qingdao, China
Phone: 0532-85952834
Fax: 0532-85953151
hjyang@qdu.edu.cn

Lisa Matthews
Old Dominion University
Norfolk, VA 23529, USA
Phone: 757-683-3470
Fax: 757-683-3220
mmath014@odu.edu

Abstract

In this paper we present a majority-based method to learn Bayesian network structure from databases distributed over a peer-to-peer network. The method consists of a majority learning algorithm and a majority consensus protocol. The majority learning algorithm discovers the local Bayesian network structure based on the local database and updates the structure once new edges are learnt from neighboring nodes. The majority consensus protocol is responsible for the exchange of the local Bayesian networks between neighboring nodes. The protocol and algorithm are executed in tandem on each node. They perform their operations asynchronously and exhibit local communications. Simulation results verify that all new edges, except for edges with confidence levels close to the confidence threshold, can be discovered by exchange of messages with a small number of neighboring nodes.

Keywords – Bayesian Network, Distributed Data Mining, Peer-to-Peer Networks, Majority Voting

1. Introduction

A Bayesian network is a probabilistic graphical model to represent uncertain data. It has been known that a Bayesian network is a popular and effective model to perform data mining. In a Bayesian network, relationships between variables can be viewed pictorially through a directed acyclic graph, which shows local dependencies between variables. In a data mining context, the network structure is learnt by the implementation of an algorithm which searches for the most likely relationships between variables in a database [7]. In many applications, the database is often distributed over a peer-to-peer network. The implementation of distributed data mining in such a network is quite challenging. For instance, it is impractical to perform global communications and global synchronization due to the significant overhead. Moreover, databases are distributed so widely that it will usually not be feasible for central processing. They must be processed in place by distributed algorithms suitable to this kind of computing environment.

In this paper, we address the problem of learning a Bayesian network structure from databases distributed over a peer-to-peer network. The significant contribution of this paper is the design of *majority network learning algorithm* and the *majority consensus protocol*. The majority learning algorithm discovers the local Bayesian network structure based on the local database and updates the structure once new edges are learnt from neighboring nodes. The majority consensus protocol is responsible for the exchange of the local Bayesian networks between neighboring nodes. The algorithm and protocol work in tandem to discover the global Bayesian network structure. The key strength of the majority learning algorithm is to ascertain the confidence in the new edges discovered by the majority consensus protocol such that a node is able to learn the global Bayesian network structure as if it were given the combined database. The key strength of the majority consensus protocol is its faster convergence with fewer messages overhead. The performance of the protocol is not dependent on user configurable parameters. This independence feature makes the protocol more predictable and robust. Moreover, the protocol is implemented locally and thus requires no synchronization among nodes. This locality feature leads to faster convergence of the computation of the global Bayesian network structure with lower message overhead. By locality, we imply that each node updates its Bayesian network structure based on the edge information provided by a small set of neighbors.

The rest of the paper is organized as follows. In Section 2, we briefly introduce the related work. Section 3 formulates the problem. Section 4 presents the majority learning algorithm and majority consensus protocol. Section 5 provides the simulation results and analysis. Finally, Section 6 concludes the paper.

2. Related Work

The database used for learning a Bayesian network structure could either contain complete data or incomplete data. Learning a Bayesian network from complete data has been discussed in [4], [5], [8], and [11]. The methodologies of structure and parameter learning have been tested on local datasets only. For all the above methods to be effective, all data must be available at a central location. To the best of our knowledge, the performance of these methodologies on distributed datasets has not been experimented. Learning a Bayesian network from incomplete data has been discussed in [12], [13], [14], [15], and [17]. The learning methodologies for incomplete data are a good fit for distributed Bayesian network learning. But none of the above methodologies have been experimented in an environment where additional data is available for improving the learning performance.

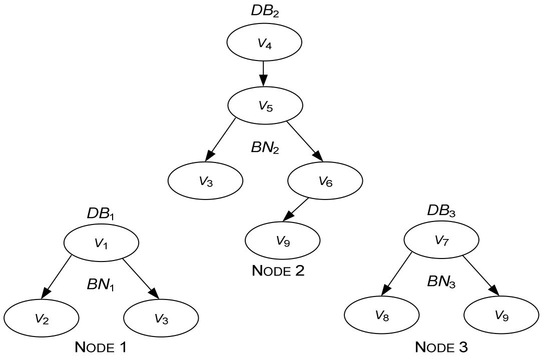
Though there have been significant research performed on learning the structure and parameters of a Bayesian network on a local database, research effort on distributed database has been minimal. To the best of our knowledge, there have been only two models presented for distributed Bayesian network learning. Kenji has worked on the homogeneous distributed learning scenario [8]. In this case, every distributed node has the same feature but different observations. A collective approach to implement a distributed Bayesian network has been proposed by Chen *et al.* [3]. But this approach is implemented in a client-server framework for heterogeneous databases. To identify cross links, every node has to submit relevant databases to the central server. The central server is responsible for learning the Bayesian network which represents the entirely combined distributed database. Each individual node, however, is only able to create its own Bayesian network based on its local database. Both models, however, propose a centralized approach that downloads all databases from distributed nodes.

The majority consensus protocol proposed in this paper is similar to the majority voting protocols [2], [22]. The majority vote problem is similar to the persistent bit problem [9], [10], for which local protocols were given. The main drawback of the aforementioned persistent bit protocols is that each of them assumes some form of synchronization. In [10], a node queries a group of other nodes and must await a reply before it proceeds. In [9] the protocol works in locked-step and assumes a global clock pulse. There are also more subtle differences which make these protocols impractical for majority vote. For instance, the former only works when the majority is very evident while the latter, because it allows any intermediate result to be corrupted, requires $O(N)$ memory at each node, where N is the network size. In contrast, our majority consensus protocol requires no synchronization at all.

The most noticeable work for distributed data mining algorithms in peer-to-peer networks is presented in [22]. In this approach a local communication based model is chosen to learn association rules in a distributed database. The messages exchanged between nodes include the data items and a confidence level for the rules. The efficiency of the local communication model is dependent on the size of the data items exchanged and the neighbor list. Not surprisingly, there is a significant message overhead due to the presence of data items in each message exchange between two peers. Also the validity of the protocol depends on user supplied frequency and confidence parameters. This makes the protocol less robust to different kinds of databases.

3. Problem Statement

The problem of learning the structure of global Bayesian network can be defined as follows. For a complete global database distributed over a peer-to-peer network, and a known variable order, the problem is to identify the structure of the Bayesian network at each node that best matches the global database. The underlying distribution of data at each node may or may not be identical. As an example, Fig. 1 illustrates that a global database is distributed across three nodes. The nine variables in the global database are distributed as illustrated. The observations for these variables are available at all three nodes. Eventually, all three nodes learn the same Bayesian network as shown in Fig. 2.



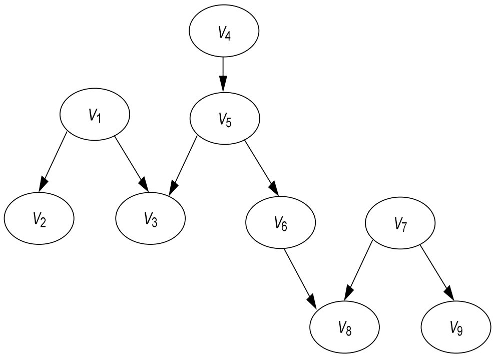


Fig. 1: A global database distributed at three nodes. Fig. 2: The global Bayesian network discovered at each node.

4. Majority-based Method

Notations

DB	The global database
V	Set of variables in *DB*, $v \in V$
E	Set of edges representing relationships among the variables in *DB*. Each edge (*edg*) has two attributes: confidence level (*conf*) and the number of records (*num*) in *DB* that are considered in determining the confidence.
BN	The global Bayesian network, $BN \equiv (V, E)$
DB_i	The database at node *i*
V_i	Set of variables in DB_i
E_i	Set of edges in DB_i
BN_i	The local Bayesian network at node *i*
N_i	Set of neighboring nodes for node *i*
r	The majority ratio to make a decision on a certain confidence level
$\prod^i$	The majority value computed by a consensus process among the nodes in N_i
$\prod^{ij}$	The majority value computed by a consensus process between nodes *i* and *j*, $j \in N_i$

The objective of our majority-based method is to ensure that all nodes in the network converge towards the correct *BN*. To this end, we develop two components, which execute on each node in tandem. The majority learning algorithm component specifies the minimal amount of information sent by each node, updates the local Bayesian network structure at each node, and terminates the learning process when a global solution has been reached. The majority consensus protocol component specifies the communication mode among the nodes, the frequency of messages exchange, and the level of co-ordination required. The rationale of our majority-based method is to combine a Bayesian network learning algorithm and a majority consensus protocol to discover all relationships between variables that exist in the combined database. The primary issues addressed are the reduction of the computation complexity and the communication overhead.

The primary steps in the algorithm and protocol are summarized as follows. The majority learning algorithm computes a local Bayesian network based on the variables associated with the data observed on each node. In the meantime, the majority consensus protocol independently identifies the neighbors for each node. A membership protocol on the lines of Newscast [6] is employed by the consensus protocol to fill up the neighbor list for each node. The consensus protocol creates a message and transmits the local Bayesian network and the confidence level to the neighbors. On receipt of the message, the majority learning algorithm updates the local Bayesian network by new edges if the resulting confidence level of the updated Bayesian network is better. The above process is repeated until the structure of *BN* is learnt at each node.

Fig. 3 illustrates the message flow between two neighboring nodes. Nodes 1 and 2 have a local Bayesian network learnt from their local databases. The majority learning algorithm at node 1 passes the Bayesian network structure to the majority consensus protocol. The protocol creates a message containing the list of edges and the confidence levels for each edge. This message is communicated to node1's neighbor node2. On the reception of message, the majority consensus protocol at node 2 keeps a record of the message received from node 1. This message is communicated to the majority learning algorithm, which updates its local Bayesian network structure accordingly. Similar process is carried out when node 2 communicates its local Bayesian network structure to node 1.

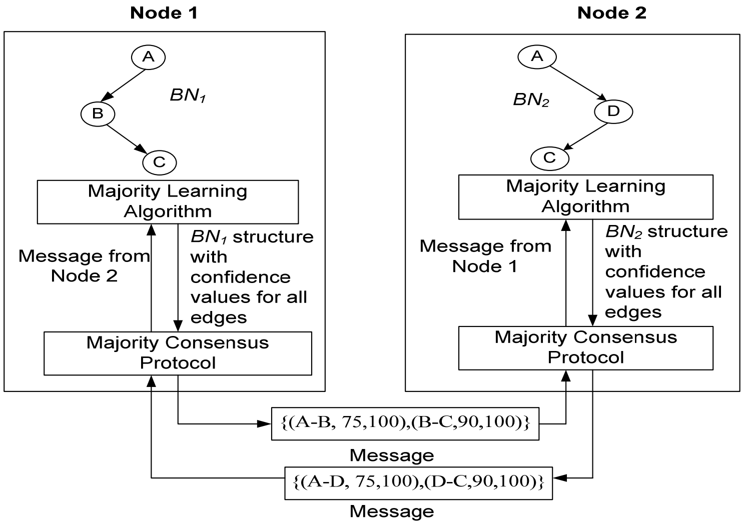

Node 1
Node 2
A
B
C
BN1
A
D
C
BN2
Majority Learning Algorithm
Message from Node 2
BN1 structure with confidence values for all edges
Majority Consensus Protocol
Majority Learning Algorithm
Message from Node 1
BN2 structure with confidence values for all edges
Majority Consensus Protocol
{(A-B, 75,100),(B-C,90,100)}
Message
{(A-D, 75,100),(D-C,90,100)}
Message

Fig. 3: Message flow between nodes 1 and 2.

4.1 Majority Learning Algorithm

The majority learning algorithm is responsible for discovering new edges based on its local data and combining edges discovered by other nodes by using a structure learning algorithm. The structure learning is a model selection process, wherein we need to select model based on the data. We adopt a searching and scoring based method K2 which defines a *confidence* that describes the fitness of each possible structure to the observed data [5]. The K2 algorithm is similar to an optimization problem: find a structure of Bayesian network that maximizes the confidence. The confidence function has a decomposability property defined as follows:

$$conf\,(BN, DB) = \sum conf\,\{(v, pa(v)), DB(v, pa(v))\} \tag{1}$$

where $pa(v)$ represents the parent of variable v, and $DB(v, pa(v))$ denotes the data involving only v and $pa(v)$. Once the confidence level is defined for the variables at each node, the next step is to identify the local Bayesian network with the highest confidence level. It is known that the search for the optimal Bayesian network is NP-hard, leading to the need for sub-optimal search methods. As the K2 algorithm is characterized by a decomposability property, the sub-optimal search methods involve making a series of edge changes (adding or deleting of edges one at a time). Every time an edge is changed, a check is performed to ascertain if the resultant local Bayesian network is a valid directed acyclic graph. For every edge change, a confidence level $conf_b(BN_b, DB)$ is calculated for BN_b before the change and $conf_a(BN_a, DB)$ for BN_a after the change. The decision on whether BN_b or BN_a is the best fit depends on which of the confidence levels has a maximum value. As the score satisfies the decomposability property, confidence levels of edges are maintained by computing the following score for each edge: $conf\,\{(v, pa(v)), DB(v, pa(v))\}$.

Once a message is received from the majority consensus protocol, the message indicates a new edge or an existing edge with updated confidence level. If the message indicates a new edge, the edge is added to the local Bayesian network. Otherwise, the confidence level of the corresponding edge is updated. Either case, the confidence level computation is then performed. The structure of the local Bayesian network is then modified based on the difference between the confidence levels before and after the change. The psuedocode for the algorithm is provided below.

Psuedocode 1 Majority Learning Algorithm

```
Newscast($N_i$)  //discover neighbor list
BNJ($E_i$)  //discover local edge list
for $e \in E_i$
   for $j \in N_i$
      $conf_{ij} = num_{ij} = conf_{ji} = num_{ji} = 0$
MessageRecv($edg_{ji}$,$conf_{ji}$, $num_{ji}$)
if $edg_{ji} \notin E_i$
   add($BN_i$,$edg_{ji}$)
Compute $conf_b(BN_b, DB)$ and $conf_a(BN_a, DB)$
if $conf_b(BN_b, DB) < conf_a(BN_a, DB)$
  Add($E_i$, $edg_{ji}$, $conf_{ji}$, $num_{ji}$)
else
   Update($BN_i$, $edg_{ji}$)
  if $conf_b(BN_b,DB) < conf_a(BN_a,DB)$
    Update($edg_{ji}$, $\prod^{ij}$, $\prod^{i}$)
    //$\prod^{ij}$ and $\prod^{i}$ are explained in consensus protocol
while($true$) //ensure neighbors have updated confidence
for $e \in E_i$
  for $j \in N_i$
    if {($\prod^{ij} < 0$) and ($\prod^{ij} < \prod^{i}$)} or
      {($\prod^{ij} > 0$) and ($\prod^{ij} > \prod^{i}$)}
    $conf_{ij} = \sum_{l \neq i, j \in N_i} conf_{li}$
    $num_{ij} = \sum_{l \neq i, j \in N_i} num_{li}$
    MessageSend($edg_{ij}$, $conf_{ij}$, $num_{jj}$)
```

The visual representation for the majority Bayesian network learning algorithm is provided in Fig. 4.

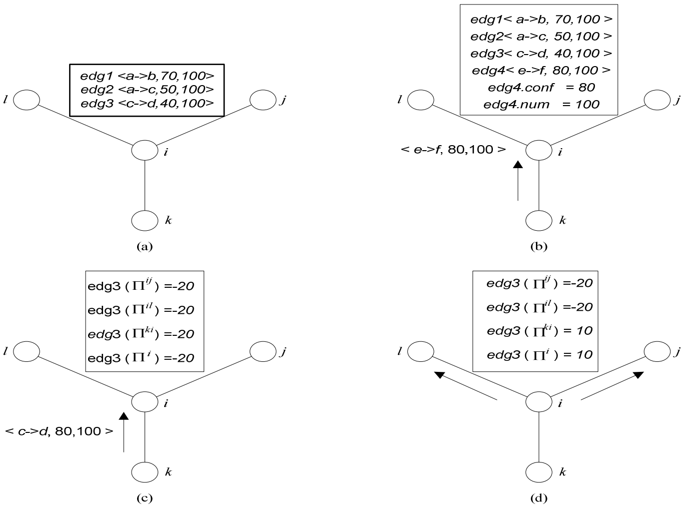


Fig. 4: A visual illustration of the majority learning algorithm from the perspective of node *i*.

The first scenario is depicted Fig. 4a wherein node *i* has three edges *a->b, a->c*, and *c->d*. On receipt of a new edge *e->f* from node *k* (Fig. 4b), the edge is added to node *i* and its attributes are computed. In the second scenario (Fig. 4c) node *i* receives an updated confidence level for edge *c->d* from node *k*. Thus, the majority values for edge *c->d* are computed at node *i* as shown in Fig. 4d. As the conditions $\prod^{i} > \prod^{ij}$ and $\prod^{i} > \prod^{il}$ are satisfied, messages are sent to nodes *j* and *l* to update their confidence levels for edge *c->d*.

4.2 Majority Consensus Protocol

The majority consensus protocol is based on an optimization problem where nodes decide whether they have all the observations necessary to learn the structure of *BN*. Following are the two main processes in the majority consensus protocol: message construction and majority selection.

a. Message Construction

The format of the message exchanged between nodes contains the identity of edge, the confidence levels of edge, and the total number of data records in the local database. The goals of the message construction process are to encode sufficient information necessary for majority selection and to maintain reasonable message length. The message sent from node *i* to node *j* is represented as a 3-field tuple, (edg_{ij}, $conf_{ij}$, num_{ij}), where *edg* represents the relation between any two variables in the database, *conf* is the confidence level that represents the accuracy of the presence of the relation, and the *num* represents the count of number of records that were considered in determining the confidence level. The confidence level and number of records are normalized to 100. Once a message is constructed, nodes communicate their messages to their neighboring nodes. Each node will record the latest messages it sent to its neighboring nodes and the latest messages it received from its neighboring nodes. These confidence values and number of records are used in the calculation of majority values for each edge.

b. Majority Selection

The objective of the majority selection process is to ensure that each node converges toward the correct majority. The correct majority refers to the correct confidence level for all edges. The consensus among nodes is measured as the proportion of nodes whose ad hoc solution agrees with the majority value. Based on message sent or received with its neighbors, node *i* calculates the majority values as follows,

$$\begin{aligned} \Pi^{i} &= \Sigma_{j \in N_i} (conf_{ij} - r \times num_{ij}) \\ \Pi^{ij} &= (conf_{ij} + conf_{ji}) - r \times (num_{ij} + num_{ji}) \end{aligned} \tag{2}$$

If no message was yet received from any neighbor, $\prod^{i}$ takes the majority value represented by the confidence levels computed by the local majority learning algorithm. The presence or absence of an edge is decided according to the majority value in $\prod^{i}$. If the sign of $\prod^{i}$ is positive, then the edge is accepted. Each time there is a change in the local confidence levels, message is received or a new neighbor connects, $\prod^{ij}$ and $\prod^{i}$ are calculated. Each node implements the majority selection process independently. Node *i* coordinates the confidence level of each edge with its neighbor *j* by maintaining the same values in $\prod^{ij}$. For every edge, *i* and *j* stop exchanging messages as long as the following conditions are true: $\prod^{i} > \prod^{ij} \geq 0$, and $\prod^{i} > \prod^{ji} \geq 0$. Similarly, when the conditions $0 > \Pi^{i} \geq \Pi^{ij}$, and $0 > \Pi^{j} \geq \Pi^{ji}$ hold, no messages are exchanged. The pseudocode for the majority selection process is shown in psuedocode 2.

Psuedocode 2 Majority Selection Process

```
Newscast($N_i$)
Getmajorityratio($r$)
for $i \in N_i$
    $\prod^{i} = 0$
    for $j \in N_i$
        $\Pi^{ij} = 0$, $conf_{ij} = 0$, $num_{ij} = 0$
    end
    if MessageRecv($edg_{ji}$,$conf_{ji}$, $num_{ji}$)
```

```
        $\Pi^{i} = \Sigma_{j \in N_i} (conf_{ij} - r \times num_{ij})$

        $\Pi^{ij} = (conf_{ij} + conf_{ji}) - r \times (num_{ij} + num_{ji})$
    for $j \in N_i$
        if {( $\Pi^{ij} < 0$) and ( $\Pi^{ij} < \Pi^{i}$ )} or
            {( $\Pi^{ij} > 0$) and ( $\Pi^{ij} > \Pi^{i}$ )}
            $conf_{ij} = \Sigma_{j,l \in N_i, j \neq l} conf_{li}$

            $num_{ij} = \Sigma_{j,l \in N_i, j \neq l} num_{li}$
            MessageSend($edg_{ii}$,$conf_{ij}$, $num_{ij}$)
            //node $i$ sends message to node $j$
    end
end
```

From the pseudocode, it is easy to derive that when the protocol dictates that no node needs to send any message, it implies that for every node *j* the $\prod^{ij}$ values for the neighbors match with the $\prod^{i}$ values. If there is a disagreement on the structure of *BN*, then due to locality there must be disagreement between two immediate neighbors. Locality is defined as the size of the neighborhood of a node. To measure locality, it is important to classify the neighbor size as maximum and average scope. The number of message interactions is bound by the size of the system. This implies that the protocol always reaches consensus in a static state. Fig. 5 depicts a graphical illustration of the majority selection process, in which *r* is set to 0.5.

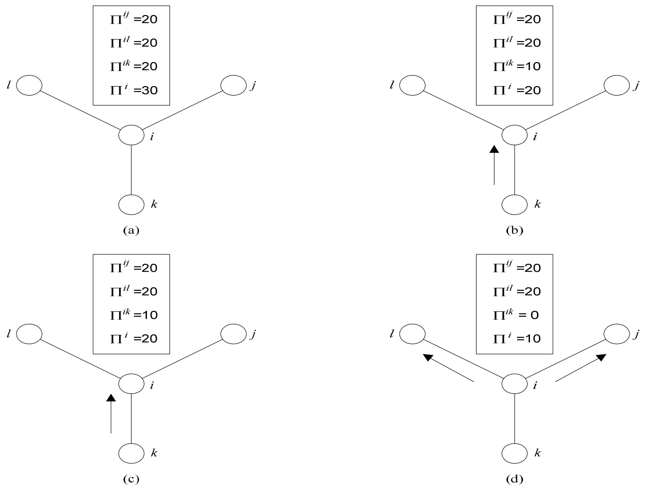


Fig. 5: A visual illustration of the Majority Selection process from the perspective of node *i*.

Fig. 5a illustrates the current state. Prior to this state, nodes *j*, *k*, and *l* have exchanged messages with node *i*, and node *i* has computed the majority values using (2). In Fig. 5b, a message is sent from node *k* to node *i*. As a result, node *i* revaluates Π^{ik} and Π^{i} . No additional messages are sent to nodes *j* and *l* as Π^{i} is equal to Π^{ij} and Π^{il} . In Fig. 5c, node *k* sends another message to node *i*.

On reception of this message (Fig. 5d), values of Π^{ik} and Π^{i} get further reduced. By now, the conditions of $\Pi^{i}<\Pi^{ij}$ and $\Pi^{i}<\Pi^{il}$ hold. Thus, node *i* sends messages to nodes *j* and *l*.

5. Simulations and Analysis

To validate and verify the majority based consensus methodology, a software package based on BNJ [1] and Peersim [16] has been developed to implement our majority learning algorithm and consensus protocol. Time is divided into rounds. A round is defined as the amount of time taken by all nodes to execute one instance of the protocol and algorithm. All of the nodes were connected in a random tree. For lack of real datasets, the ASIA model dataset [13] was used in simulations. We generated 1,000,000 observations from this model, which were distributed among the nodes in the network. In practice, we do not have control over the distribution of the data among different nodes. Local Bayesian networks were constructed using a K2 [5] structure learning algorithm.

5.1 Effect of Majority Consensus Local Communications

One of the main performance characteristics of our protocol is the locality of the majority selection protocol. Locality is quantified by measuring the scope of a node. The scope of a node is defined as the number of neighbors whose confidence levels are maintained by the node. The overall locality is measured by taking into account the maximum and average values of scope. The average locality is preferred for majority selection. The scope also provides information related to message exchange and processing overhead. Message overhead is computed based on the total number of messages exchanged between a node and its neighbors. Processing overhead is computed by calculating the number of cycles required for each node to arrive at a decision. Hence average locality also makes the communications fast and efficient. Fig. 6 shows the locality of the protocol to determine the existence of one relational edge between two variables. The confidence levels for the relational edge are based on the amount of related data at each node. As the data is randomly distributed among the 5,000 nodes, the confidence levels for the relational edge are assigned randomly.

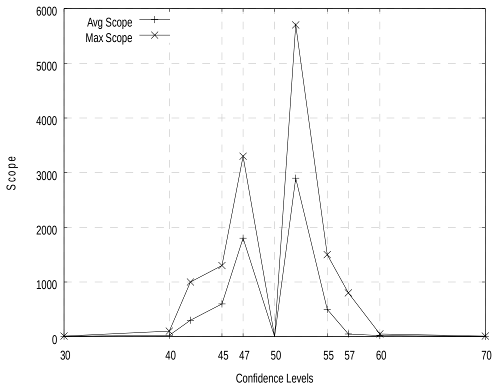


Fig. 6: Locality for network size of 5,000 nodes and with majority ratio 50%.

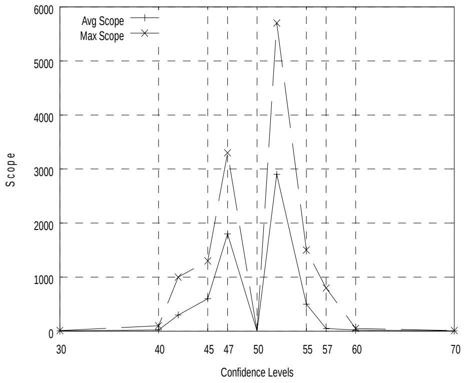


Fig. 7: Locality for network size of 10,000 nodes and with majority ratio 50%.

From Figs 6 and 7 we conclude that if data is distributed in an unbiased fashion, the locality of the protocol is assured. For a majority ratio of 50%, if the confidence level is greater than 60% or less than 40%, the number of nodes involved in the message exchange is limited to 10-12 nodes. As the confidence level approaches 50%, the average scope grows. This indicates that more nodes have to be involved in the decision-making when the confidence level approaches the majority ratio. This conclusion can be generalized as that all new edges, except the edges with confidence levels close to the confidence threshold, can be discovered by exchange of messages with a small number of neighboring nodes, whose size is independent of the size of the network.

We assess the strength of our majority consensus protocol by measuring the message overhead. The message overhead calculation is based on the difference between the average confidence levels and the majority ratio. The total message overhead in our majority vote protocol is equal to the messages exchanged between every node with its neighbors. The message exchanged is a 3-field tuple represented by <Edg_{ab}, $conf_{ij}$, len_{ij}>. If *Size* represents the length of each tuple in bytes, then the overall message overhead is given by $\sum_{i,j\in E^i} Size_{ij}$ only when the following conditions are met: $0 < ((conf_i - R)/(R)) < 0.2$ or $0 > ((conf_i - R)/(R)) > -0.2$. The message overhead is approximately equal to zero when $((conf_k - R)/(R)) > 0.2$ or $((conf_k - R)/(R)) < -0.2$. It should be noticed that the message overhead depends on the size of the neighbor list and the number of data bits exchanged between neighbors.

To prove that the message overhead incurred in majority consensus protocol is lower, we compare our protocol with a similar majority based protocol proposed by Ran Wolff *et.al.* [22]. According to Ran Wolff *et. al,* the message overhead in their majority based protocol is given by a constant function $1+\sum_{i,j\in E^i} count^{ij}$, where i and j represent the nodes in the network, $count^{ij}$ represents the message length exchanged between the nodes i and j, and E^i represents the set of edges which collide with node i. If the majority based approach proposed by Ran Wolff *et al* is applied to discover distributed Bayesian networks where the majority of the confidence levels are not close to the majority ratio, then the message overhead is given by the constant function $1+\sum_{i,j\in E^i} count^{ij}$, while in case of majority consensus protocol, the message overhead is approximately equal to zero. Thus, there is significant savings in message overhead by applying majority consensus protocol when the confidence levels are not close to the majority ratio.

5.2 Convergence of the Learning Algorithm

We measure the convergence of the majority learning algorithm by calculating the percentage of new edges discovered and the percentage of correct edges in the global Bayesian network. Fig. 7 shows the percentage of new edges discovered on an average basis during the execution of the majority learning algorithm.

In Fig. 8, the confidence levels of the edges in the majority of the nodes were away from the majority ratio. This leads to fewer messages exchanged and thereby most of the nodes agree on the majority decision quickly. All edges are discovered at the end of 12 rounds. The lesser number of rounds confirm the fact that if the confidence levels of the edges are far away from the majority ratio the nodes discover the new edges quickly. As the number of messages exchanged between the nodes decreases, the nodes can reach a decision quickly leading to fewer rounds.

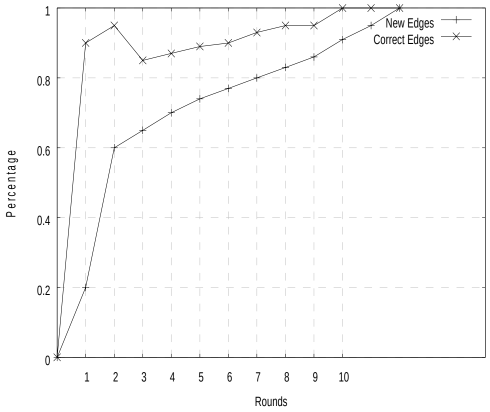


Fig. 8: Percentage of new edges and correct edges discovered for a network of 5,000 nodes.

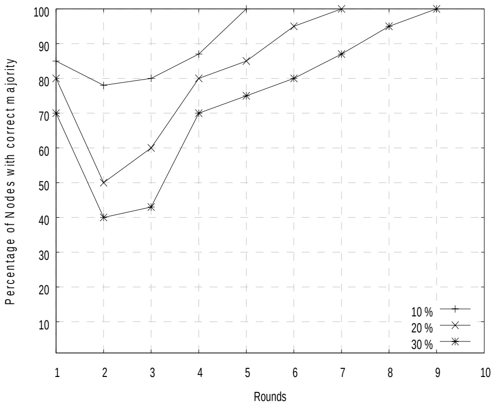


Fig. 9: Percentage of nodes that compute the right decision for confidence levels lesser than or equal to 30 % threshold.

Fig. 9 shows the percentage of the nodes that arrive at a correct global Bayesian network model when the confidence level in the edges are less than or equal to 40%. For edges with confidence levels lesser than or equal to 30%, all the nodes quickly arrive at a decision to reject the edges. The number of rounds taken to arrive at this decision is less than or equal to 10. This result confirms the fact that all nodes converge quickly when edges have confidence levels much lower than the 50% threshold.

Fig. 10 shows the percentage of the nodes that arrive at a correct global Bayesian network model when the confidence level in the edges is greater than or equal to 60%. For edges with confidence levels greater than or equal to 70%, all the nodes quickly arrive at a decision to accept the edges. The number of rounds taken to arrive at this decision is less than or equal to 10. This result confirms the fact that all nodes converge quickly when the edges have confidence levels much higher than the 50% threshold.

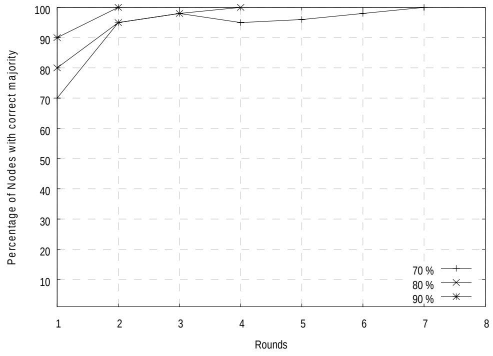


Fig. 10: Percentage of nodes that compute the right decision for confidence levels greater than or equal to 70 %.

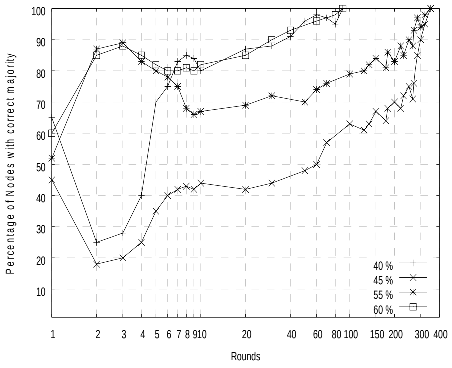


Fig. 11: Percentage of nodes that compute the right decision for confidence levels near the 50% threshold.

In Fig. 11, the confidence levels are closer to the threshold. The presence of majority of nodes with confidence levels near the 50% threshold leads to more messages exchanged thereby delaying the agreement on decision making by all the nodes. The figure indicates that for edges with confidence levels near the 50% threshold requires large portion of the network to participate to arrive at a decision. So edges with confidence levels closer to 50% take a significant amount of time for all nodes to agree upon.

To better explain Figs. 8, 9, 10, and 11, the interaction between the majority consensus protocol and majority based Bayesian network algorithm needs to be analyzed. The majority consensus protocol behaves in a wave pattern characterized by positive slopes and negative slopes. In the first round, every node assumes the confidence level to be zero. In the initial rounds, the protocol convinces more nodes of a certain confidence level leading to a wave with a positive slope. But when the majority of nodes do not agree on a certain confidence level, then the wave exhibits a negative slope. If the confidence level among the edges is between 70-90%, then the edges are learnt quicker and agreed by most nodes. These are the edges which are very significant and are expected to be discovered in the earlier rounds and agreed by all the nodes in the network fast. The same analysis applies for edges with lower confidence levels (10-30%). The edges with low confidence levels are rejected earlier by most of the nodes. They are also discovered early but are quickly rejected as greater portion of the local database is scanned. Now, the edges with confidence levels between (40-60%) require more rounds for all the nodes to agree with the majority decision. As these confidence levels are near the 50 % majority threshold, the majority based Bayesian network algorithm takes a longer time for all the nodes to agree with the majority decision.

6. Conclusions

We have developed a new methodology for discovering the structure of a global Bayesian network from data distributed in a peer-to-peer network. The crux of the methodology is the majority Bayesian network learning algorithm and the majority consensus protocol. The unique features of the protocol and algorithm are the locality of communications, less overheard, high accuracy, and quick convergence. Analysis indicates that there is a significant saving in message overhead by applying our protocol when the confidence levels are not close to the majority ratio. Simulations demonstrate that as the confidence levels are away from the majority ratio, fewer messages are exchanged and thereby the protocol and algorithm accurately discover the correct edges within less number of rounds. The efficiency of the local communication also depends on the confidence level of the edges. Simulation results verify that all new edges, except for edges with confidence levels closer to the confidence threshold, are discovered by exchange of messages with a small neighborhood of nodes, whose size is independent of the size of the network. If the edges have lower confidence level (10-30%), the communication is very efficient due to less exchange of messages. Also edges with a higher confidence level (70-90%) also result in fewer messages exchanged between neighboring nodes. As most of the edges have confidence levels which fall into the above two categories, the communication load mainly depends on the edges whose confidence level is between (40-60%). To control the communication load in our methodology, the confidence levels of the relations should be chosen based on the above constraints.

References

[1] Bayesian Network Tools in Java (BNJ) http://bnj.sourceforge.net/

[2] Burman, J., Kutten, S., Herman, T., Patt-Shamir, B., "Asynchronous and Fully Self-Stabilizing Time-Adaptive Majority Consensus," *Proceeding of 9th International Conference on Principles of Distributed Systems*, 2005.

[3] Chen, R., Sivakumar, K., and Kargupta, H., "Collective Mining of Bayesian Networks from Distributed Heterogeneous Data," *Knowledge and Information Systems Journal*, vol. 6, pp. 164-187, 2004.

[4] Cheng, J., Bell, D.A. and Liu, W., "An algorithm for Bayesian belief network construction from data," in *Proceedings of AI & STAT*, pp.83-90, 1997.

[5] Cooper, G. F. and Herskovits, E., "A Bayesian method for the induction of probabilistic networks from data," *Machine Learning* 9, pp. 309-347, 1992.

[6] Jelasity, M., Kowalczyk, W., and Van Steen M. "Newscast computing," Technical Report IR-CS-006, Department of Computer Science, Amsterdam, The Netherlands.

[7] Jensen, F., "An Introduction to Bayesian Networks," Springer, 1996.

[8] Kenji, Y., "Distributed cooperative Bayesian learning strategies," in *Proceedings of the Tenth Annual Conference on Computational Learning Theory*, ACM Press, Nashville, Tennessee, pp. 250-262, 1997.

[9] Kutten., S. and Patt-Shamir, B., "Stabilizing time-adaptive protocols," *Theoretical Computer Science*, 220(1):93.111, 1999.

[10] Kutten, S., and Peleg, D., Fault-local distributed mending. In *Proceedings of the Fourteenth Annual ACM Symposium on Principle of Distributed Computing*, pp. 20 - 27, August 1995.

[11] Lam, W. and Bacchus, F., "Learning Bayesian belief networks: An approach based on the MDL principle," *Computational Intelligence*, pp. 262-293,1994.

[12] Lauritzen, S. L., "The EM algorithm for graphical association models with missing data," *Computational Statistics and Data Analysis*, pp. 191-201, 1995.

[13] Lauritzen, S. L. and Spiegelhalter, D. J., "Local computations with probabilities on graphical structures and their application to expert systems (with discussion)," *Journal of the Royal Statistical Society*, series B 50, 157-224, 1988.

[14] Madigan, D. and Raftery, A., "Model selection and accounting for model uncertainty in graphical models using Occam's window," *Journal of the American Statistical Association*, pp. 1535-1546, 1994.

[15] Meila, M. and Jordan, M. I., "Estimating dependency structure as a hidden variable", in NIPS, 1998.

[16] PeerSim, A Peer-to-Peer Simulator http://peersim.sourceforge.net/

[17] Singh, M., "Learning Bayesian networks from incomplete data," in *Proceedings of the National conference on Artificial Intelligence*, pp. 27-31, 1997..

[18] Spiegelhalter, D. J. and Lauritzen, S. L., "Sequential updating of conditional probabilities on directed graphical structures," *Networks*, pp.570-605, 1990.

[19] Suzuki, J., "A construction of Bayesian networks from databases based on an MDL scheme," in *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, Morgam Kaufmann, pp. 266-273, 1993.

[20] Thiesson, B. , "Accelerated quantification of Bayesian networks with incomplete data," in *Proceedings of the First International Conference on knowledge Discovery and Data Mining*, AAAI Press, pp. 306-311, 1995.

[21] Thomas, A., Spiegelhalter, D. and Gilks, W., "Bugs: A program to perform Bayesian inference using Gibbs sampling," *Bayesian Statistics*, Oxford University Press, pp. 837-842, 1992.

[22] Wolff, R. and Schuster, A. "Association Rule Mining in Peer-to-Peer Systems," *IEEE Transactions on Systems, Man and Cybernetics,* Volume 34, Issue 6, Dec. 2004.