Learning Bayes Net Structures for Social Network Analysis

Phuay Wee Eugene Ang, Yuan Tian, Tianrou Wang and Rong Zhou
Department of Computer Science
Columbia University
New York, NY 10027
{pa2466,yt2545,tw2565,rz2361}@columbia.edu

Abstract

Learning social network structure is one of the important perspectives in social network analysis. Bayesian network as the most commonly used graphical model has been applied into different scenarios in social network analysis. However, due to large size of data sets, training a full Bayesian network can be challenging. Goldenberg and Moore proposed an algorithm named *Bayes Net Structure search algorithm* (SBNS) to tackle with the large data sets. Based on the SBNS algorithm, we proposed likelihood SBNS (lSBNS) algorithm using joint probability as scoring measurement in edge-screening step. lSBNS not only holds the properties of the original algorithm, but guarantees a faster searching process with relatively sparser final Bayesian network.

1 Introduction

Structural learning for graphical models is one of the key components in the field of social network analysis. Bayesian networks, one of the common network structures, have been applied and help people make inferences and decisions. In a Bayesian network for social network analysis, each individual is considered as a single node $v \in V$ and directed edges $e \in E$ indicate relationships between nodes. By learning a Bayesian network from the records of past actions or relationships between different elements, it is possible to gain information about how individuals are aggregated [1], and how likely some individuals impact a group of people in some specific settings.

The challenges of using graphical models to analyze social network come from a large amount of sparse data sets that a single network generates overtime. This usually makes the learning process for probabilistic models time-consuming, especially for structural learning. Graphical models that express social networks usually have a large number of nodes but most individual nodes are not connecting to every other nodes. Meanwhile, a lot of nodes may cluster together and form complete subgraphs. Therefore, the adjacency matrices that represent edges in the social network will be very large and sparse.

Over the years, researchers have proposed different algorithms for structural learning in Bayesian networks, and tried to obtain some inferences from the structure. Most existing algorithms rely on "search and score", which constructs the networks by measuring scoring functions and gradually adding edges into the network that can increase the score. Some methods are based on constraints that are used for removing edges from graphs based on dependency tests [2]. Although variants of structural learning algorithms have been developed, not every method will be suitable for large sparse datasets. Goldenberg and Moore [3, 4] proposed *Screen-based Bayes Net Structure search* (SBNS) algorithm, which was then successfully applied into several co-authorship data sets to construct corresponding Bayesian networks. They adopted the concept of frequent set, which comes from the association rule learning, but fits the sparseness of social network data very well. The whole

COMS4772 (Spring 2017), Columbia University, New York, NY, USA.

algorithm is mainly based on "search and score" technique, which provokes us the idea of combining dependency computation into the algorithm for Bayesian network construction. Inspired by the SBNS algorithm and other related works, we replaced the scoring function in edge-screening test with the joint probability of each directed acyclic graph (DAG) to select candidate edges for the global-based construction of Bayesian network.

The goal of this project is to apply likelihood SBNS (lSBNS) algorithm, the modified version of SBNS, into social network analysis. We used Wikipedia adminship election data as our illustrative data analysis to show how Bayesian network is constructed using lSBNS. Results show that lSBNS method's construction for Bayesian network is more efficient, and at the same time it tends to provide sparser network compared to the original SBNS algorithm.

The rest of this report is organized as follows: Section 2 discusses basic concepts and terminologies of Bayesian network; Section 3 talks about the SBNS algorithm [3, 4] and our modified version of SBNS algorithm; Section 4 is the dataset information and experiment design; Section 5 shows the results and discussion; Section 6 is future work and conclusion.

2 Background Study

2.1 Bayesian Network

A Bayesian network is a probabilistic graphical model which uses directed acyclic graph (DAG) and a set of parameters to infer outcomes of events. The DAG is composed of nodes and edges. The nodes represent the random variables, $X = X_1, \ldots X_i, \ldots X_n$ from the domain, which is the universal set of events. Their conditional dependencies are reflected in the edges, for example, X_i -> X_j represents the direct dependencies between the two variables. The structure of the network captures qualitative relationships between nodes. In particular, two nodes should be connected directly if one affects or causes the other, with the edge indicating the direction of the effect.

For example, in Figure 1 below, there are three nodes, A, B, C which represents three different events. Suppose event A and event B could cause event C happening and event A has a direct effect on event B, then these three nodes could form a Bayesian network with node A pointing to B and C, and node B pointing to C.

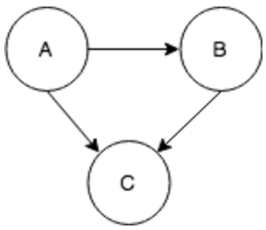


Figure 1: The Bayesian network example.

The joint probability function for the above example is equation 1. Generally, the joint probability function is formed as equation 2. The $Pa(X_i)$ represents the set of parents of variable X_i in the DAG. Acyclicity of the DAG guarantees the product in equation 2 is a coherent probability distribution. More information on Bayesian networks can be found in [6, 7].

$$P(A, B, C) = P(A) \times P(B|A) \times P(C|A, B) \tag{1}$$

$$P(X_1, ...X_n) = \prod_i P(X_i|Pa(X_i)) \tag{2}$$

2.2 Scoring Functions for Bayesian Network

Learning Bayesian networks consists of finding the network that best fits, for a certain scoring function, the data. Cooper [8] showed that the inference of a general Bayesian network is a NP-hard problem, and later, Dagum and Luby [9] showed that even finding an approximate solution is NP-hard.

In this case, learning Bayesian network has been restricted to tree structures and it is realized by scoring metrics optimization. Various scoring functions of Bayesian networks for recovering true underlying structures have different capabilities. Several score metrics have been developed, such as uniform prior score metric (UPSM), conditional uniform prior score metric (CUPSM), Dirichlet prior score metric (DPSM), Bayesian Dirichlet equivalent uniform (BDeu), and minimum description length (MDL) [12]. In our project, we used BDeu, which is less sensitive to low marginal counts [3, 4]. The BDe score was originally proposed by Buntine [10], and later BDeu scoring function was introduced by adding the uniform prior over structures to capture additional information of Bayesian network parameters. Meanwhile, the robustness to low support sizes of BDeu will be more suitable for various situations compared with other scoring functions.

2.3 Bayesian Learning Problem

In most cases, we should learn our Bayesian network from the data which is called Bayesian Learning problem. They can be stated informally as follows: given the training data, learn the structure or the graph topology and the parameters of Bayesian network. Learning structure is considered a little harder than learning parameters [11]. In our paper, we focused on learning the structure of Bayesian network. This problem comes from the assumption where we do not know the structure and the dataset is fully observed. Moreover, whether we could construct our structure depends on the learning goal [18]. Our goal in this project is to learn the dependencies of variables from the learned network. In addition, Bayesian network reveals much finer structure which we could distinguish between dependencies and independencies.

There are several methods for structural learning of Bayesian network. One method is that we could first determine the basic skeleton of the structure by prior knowledge and relationships. Then we could decide the directions of each edge via conditional independencies. More algorithms which used this method are illustrated in [13, 14, 15].

Different from the methods above, the algorithm we used in this project is to optimize our network structure based on searching and scoring function. We were first inspired by the SBNS algorithm applied to the co-authorship problem. This algorithm first constructs a graph which consists of nodes representing authors. Then it uses the relationship of co-author to learn the edges. For example, an edge from author A to author B means if there is a paper whose author is author A, it has high possibility that this paper is also written by author B. In our project, we are more interested in the relationship between the nominees in elections. Specifically, if a nominee is supported, is there a possibility that another nominee is also supported? For this case, a Bayesian network structure based on the election data will be helpful to answer the above question. In addition, we extended our algorithm in edge-screening step of SBNS by selecting edges from local Bayesian network with high likelihood instead of high BDeu score. In the process of global Bayesian network construction, incrementing the number of edges is determined by whether the edge can improve the scores of our current network. In the end, we could subjectively examine our results by the number of edges included in our global Bayesian network. Too few edges means missing out on dependencies and too many edges means spurious dependencies [18]. At last, we could get some inferences from the final structure of our network to predict the nominee's election results. More detailed algorithms will be illustrated in section 3.

3 Algorithm

3.1 SBNS Algorithm

The SBNS describes a procedure for computing the scores of DAGs to obtain a large collection of edges from which the global Bayesian Network will be constructed. The SBNS algorithm uses frequent set concept to speed up the structural search. This makes the algorithm not only "cache m-way sufficient statistics", but emphasizes the local structure at the same time [4].

The frequent set problem is introduced in the "market-basket" model and is often viewed as the discovery of association rules [16]. Items appear in many of the same baskets are said to be "frequent". More specifically, if we have a set of items I, the support for I is the number of baskets which contain all the items in I and I is frequent if its support is larger than a certain threshold s.

In the SBNS algorithm, firstly, frequent sets from size of 2 to k (user defined maximum frequent set size) are screened. For each frequent set, several DAGs would be considered, and the BDeu metric is used to measure how well the DAG explains the dependencies of the nodes based on the data. Since only DAGs that contain a node with $m-1$ parents (m-way interaction) may not be considered in the lower order interactions, only these kinds of DAGs with best BDeu score could pass the screening test, and the corresponding edges in those DAGs will be added to the edge dump [4], which contains all the candidate edges being considered in the global Bayesian network construction.

In the global Bayesian network construction phase, edges in the edge dump will be sorted based on the number of m-way interactions they are involved. We start with a Bayesian network with no edges and iterate over every possible edge in the sorted edge dump. Only if an edge improves the current global score and does not form a cycle will it be added to the structure of the Bayesian network. Figure 2 illustrates how the algorithm works.

algorithm	SBNS
input	**K** - max Frequent Set size **s** - support
output	BN - Bayes Net

Also: Ed - Edgedump - a collection of directed edges represented as (source,dest,count)
DS - DAG storage

1. **for** k = 2 .. K
2. obtain counts for all Frequent Sets of size k
3. **foreach** Frequent Set
4. find best scoring DAG
5. **if** DAG contains a node that has $k-1$ parents
6. store DAG in DS
7. **end foreach**
8. **end for**
9. **foreach** DAG in DS
10. store all edges $\{source, dest, count{+}{+}\}$in Ed
11. order Ed in decreasing order of edge counts
12. **foreach** edge $e \in Ed$
13. **if** e doesn't form a cycle in BN
14. **and** e improves BDeu
15. add e to BN
16. **end foreach**
17. return BN

Figure 2: The SBNS algorithm

3.2 Adjustment and Innovation of the Algorithm

The modification of the algorithm mainly comes from the edge-screening step, where we replaced the BDeu scoring with joint probability of a local Bayesian network for selecting the best scoring DAG. Using joint probability could not only make the algorithm efficient, but provide a easier and more intuitive way to add the edges as well. Given the model, we want to choose the most optimal one to represent the relationship of the variables based on the data set.

For clarity of computing, we illustrate an example with three nodes below. Given three random variables, we could compute the likelihood as follows. As we only consider the graph with a node that has $m-1$ parents, we give high priority to compute the likelihood of those graphs. If the graph has three nodes, then we should consider DAGs that contain a node with two parents. We first consider cases shown in Figure 3 (a) and Figure 3 (b). We need to compare their likelihood to the fully independent scenario shown in Figure 3 (c). If the fully independent case has higher likelihood than every other DAG satisfying the m-way interactions, then there is no need to consider other cases. If one of the 2-parent cases wins, we need to continue the comparison with all the other DAGs with three nodes to guarantee that it is the highest scoring DAG. The 2-parent case with highest likelihood among all the DAGs will be taken into consideration into the next step. Take Figure 3 (a) equation as an example, P(A) and P(C) are just marginal probability and they are easily calculated from the

datasets. P(B| A, C) could be written as P(A, B, C) / P(A, C) by probability theory. Furthermore, P(A, B, C) could be calculated as

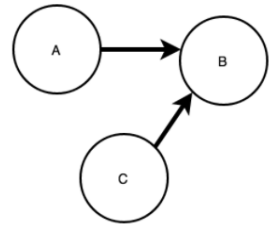
$$\frac{freq(A, B, C)}{(totalA + totalC)} \tag{3}$$

and P(A, C) could be calculated as

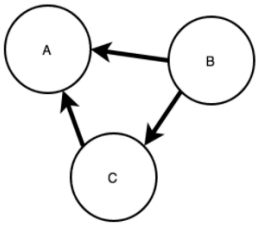
$$\frac{freq(A, C)}{(totalA + totalC)} \tag{4}$$

The freq(A, B, C) means the frequent set of A, B and C. Total A means the number of people who participate in the elections with nominee A. At last, the equation could be simplified as

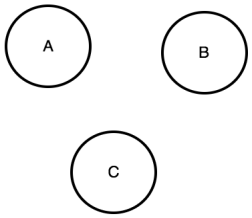
$$\frac{freq(A, B, C)}{freq(A, C)} \tag{5}$$



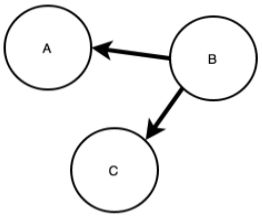
(a) $P = P(A) \times P(C) \times P(B|A, C)$



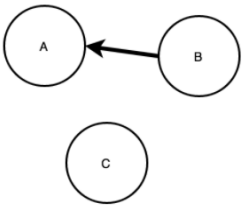
(b) $P = P(B) \times P(B|C) \times P(A|B, C)$



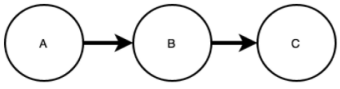
(c) $P = P(A) \times P(B) \times P(C)$



(d) $P = P(B) \times P(A|B) \times P(C|B)$



(e) $P = P(B) \times P(A|B) \times P(C)$



(f) $P = P(A) \times P(B|A) \times P(C|B)$

Figure 3: DAG with 3 nodes

Other occasions could also be computed in a similar way. We just need to compare the results of different cases. If the node with $m - 1$ parents is the winner, we could add the graph?s edges to edge dump. Otherwise, we just skip those cases and continue with the next frequent set.

In general, compared with the SBNS algorithm, the edges of our algorithm are weighted. For instance, a paper with two or more authors could only be published once. But in the election setting, a voter could support or oppose the same nominee several times in different elections. Thus in this case, we consider the duplicates which means they are counted several times. In the original algorithm, however, those cases would only be counted once which might cause the loss of information.

4 Real Data Analysis and Experiment Design

4.1 Dataset

The paper utilized the Wikipedia adminship election datasets and tried to extract value of the datasets from different perspectives. Administrator in Wikipedia community are users with access to additional technical features that aid in maintenance of the website. To become an admin, the Wikipedia community member can either submit a request for adminship, or may be nominated by

other members. Subsequently, other editors could cast a support, neutral, or oppose vote during the election period. The datasets record all administrator elections and vote history data since January 3, 2008. Some statistics of the datasets are summarized in Table 1.

Table 1: Statistics of the Wikipedia adminship election dataset

Dataset	Value
Votes	114040
Support Votes	83962
Oppose Votes	23118
Voters	6210
Nominees	2391

4.2 Experiment Design

The data has been preprocessed via data cleaning and data reduction techniques. After preprocessing, only the voter, nominee, and vote (1:support, 0:neutral, -1:oppose) are kept. Then the data could be transformed into a $m \times n$ matrix which contains a collection of M records of N variables $X_1, \ldots X_N$. X_{ij} is the jth attribute of the ith observation, where $1 \leq i \leq M$ and $1 \leq j \leq N$. Table 2 below shows an example of preprocessed matrix of Wikipedia adminship election data.

Table 2: Example of preprocessed matrix of Wikipedia adminship election data

	Nominee A	Nominee B	Nominee C	Nominee D
Voter 1	-1	0	0	1
Voter 2	0	1	0	0
Voter 3	0	0	-1	1

In our project, offline experiments are applied as the main method for evaluating the performance of the Bayesian network structure learned based on the algorithms. Four sets of experiments are designed. The first attempts to find out the reasonable support choices. The second expects to observe whether there will be a great improvement in the BDeu score of the Bayesian network when increasing the maximum set size. And the third tries to gain additional information from the Oppose Bayesian Nets that cannot be inferred only from the Support Bayes Nets. Lastly, we compare the lSBNS to the original SBNS algorithm.

4.2.1 Comparisons of Different Support Thresholds

Our algorithm is based on the presumption that frequent sets contain most of the essential information about the whole dataset. An itemset is defined frequent if its greater or equal than a user defined threshold, support s. In our design, only those nominees are occurred simultaneously greater than or equal to s times will be considered. If s is set to be higher, We may reduce the total number of nominees significantly and thus there may be a loss in accuracy. Otherwise, if the value of s is set too small, However, it will introduce a lot of interactions that have two-way marginal counts. Model fitting in contingency tables in general is sensitive to very low marginal counts [17]. To test how support would affect the BDeu score, frequent sets with different supports are considered and the corresponding Bayesian networks are built. As time is limited, only the support of frequent set of size 2 are tuned. The database has been thoroughly studied. It turned out that there were about one million pairs of nominees who have been simultaneously supported by at least one voter. And only about 5000 pairs of nominee won the votes from thirty or more voters. So we set the frequent set of size 2 support separately to be 3,10,20 and 30.

4.2.2 Comparisons of Different Maximum Frequent Set

Maximum frequent set (MFS) is also one of the important parameters for the performance of our algorithm. There may be a loss in accuracy when high order interactions are not taken into account. However, It is also not necessary to examine every frequent set. The algorithm may perform well,

even when we prune some of the candidate itemsets, because there is a natural upper bound on MFS due to the sparsity of the datasets.

In our experiments, frequent sets of size two, that is, only pairs of items were considered. Then 3-node interactions were also involved. Since more edges were considered, and the input matrix for calculating the BDeu score was much denser, the algorithm running with MFS of 3 was quite time-consuming. Thus, we didn't continue further investigating cases with MFS larger than 3. The two experiments results were compared against one another over a predefined support.

4.2.3 Comparison of Support Model and the Oppose Model

In each election a voter can cast a support, oppose or neutral vote towards a nominee. Previously, states of X_j in the input matrix were binary variables, and in the model a support vote was treated as a positive sample while the rest were negatives. The frequent set in this model can be easily computed as: $freq(S) = |\{i : \forall j \in S, x_{ij} = 1\}|$ In our third experiment, we redefined the semantics of each edge in the oppose model - defining opposition as a positive sample while the rest were negatives. The frequent set then can be calculated using the following formula $freq(S) = |\{i : \forall j \in S, x_{ij} = -1\}|$ We compared the results of this new oppose model against our original support model experiment expecting to gain more insights. More findings will be discussed in section 5.3.

4.2.4 Comparison of SBNS Algorithm and lSBNS Algorithm

The lSBNS algorithm introduced in this report is designed to speed up the structural learning of Bayesian network. To study its performance, we conduct a fourth set of experiments to compare the final BDeu score, computing time, and sparsity using lSBNS to the original SBNS algorithm. Due to long running time of SBNS algorithm, we did not run on frequent sets of size 3. Both algorithms in the experiments only used 1,044,353 frequent sets of size 2, indicating the same MFS, and the same threshold for support is set to 3 to satisfy the uniformity condition. In lSBNS the total number of edges added into the edge dump was 5,585. Using SBNS, this value can reach 438,644. We were expecting that lSBNS is on average faster than SBNS, since it might take less time to find all 3,548 edges added in the global Bayesian network. Clearly, if SBNS finds at least one more edge than lSBNS we could conclude that the Bayesian network constructed using lSBNS is sparser. So we do not construct the global Bayesian network using SBNS due to time limitations. To find out whether both algorithms increase the BDeu score as more edges were added in practice, and which algorithm performs better in respect of accuracy, the top 5,585 edges with highest BDeu score in the SBNS's edge dump were selected to construct the global Bayesian network.

5 Results and Discussion

Based on the learned Bayesian network, we can gain inferences about relationships between different nominees. For example, if there is a directed edge from nominee A to nominee B, we can then infer that if nominee A is supported, then it is very likely that nominee B will be elected as well.

By only considering frequent sets of size 2, the final Bayesian network has already captured sufficient amount of useful information. For example, in the following subgraph of the final Bayesian network shown below in Figure 4, nominee with id "4417" has linked with multiple nodes. By tracing back to the original election files, we found out that those edges are easily interpretable. Table 3 shows the number of common voters of different nodes linked with node "4417".

To further investigate the nodes with only one common voter with "4417", we checked the original voting data. It turns out that nodes that have only one common voter with "4417" fall into one of the following categories: 1. The candidate is very popular that most of the voters vote for him/her; 2. The candidate is disliked by most voters, and only few voters gave him/her support in election; 3. There are active voters that blindly voted yes to all the candidates.

Experiment results are shown in the following subsections.

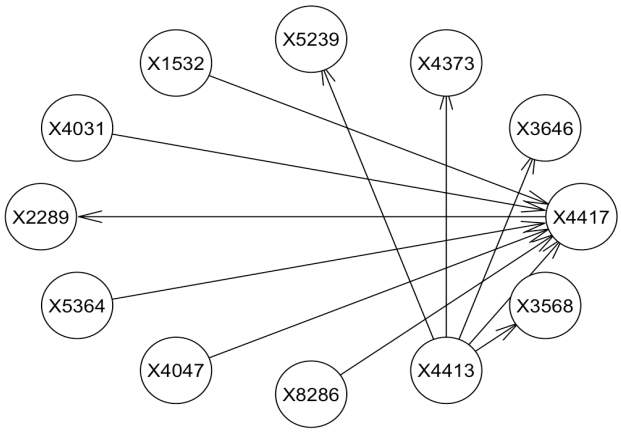


Figure 4: Subgraph of the learned Bayesian network

Table 3: Number of common voters for node "4417"

Node ID	Number of common voters
4031	7
2289	25
5364	3
4047	6
1532	1
4413	1
8286	1

5.1 Comparisons of Different Support Thresholds

After studying the data, we found that there were about one million pairs of nominees who were simultaneously supported by at least one common voter. As we increased the support level, the number of pairs of nominees dropped, ending at about 5000 pairs for support level of 30.

As time was limited, we decided to only tune the support thresholds of frequent sets of size 2. Experimenting with support thresholds of 1,3,10,20 and 30 respectively, we decided to evaluate their differences comparing the different edge dumps generated. This makes sense because edge dumps are intermediate data generated before the final network is built. If the network structure learning algorithm is given the same set of edges in the edge dump, then the resulting network would be the same. Therefore, by comparing similarity of edge dumps we can learn a lot about the resulting structure without having to fully compute the solutions. We were initially surprised to learn that despite using rather widely varying support thresholds, the same edge dump is produced each time, implying that all of them would have gone on to learn the same network. This puzzled us as we had expected the size of the edge dump to decrease as we increased the support threshold level.

After some investigation, it turned out that our decision to consider only frequent sets of size 2 was fundamentally flawed. Edges are added to the edge dump based on an evaluation of a small local DAG. For frequent sets of size 2, there is only one possible DAG of size 2 containing the edge X_j->X_j. For frequent sets of size 3, due to a larger combinatorial space, there are many more possible DAGs of size 3 containing the same edge. Therefore, evaluation of frequent sets of size 3 and larger allow more variation to manifest in the relationship between support threshold and edge dump size.

Despite the setback in our initial experiment, we were still able to salvage the situation. In our implementation of the original SBNS algorithm, we generated edge dumps for frequent sets of size 3 over a similar range of support threshold levels. We analyzed these edge dumps resulting from each support level, comparing their sizes. The results are shown in Table 4. Although the scoring

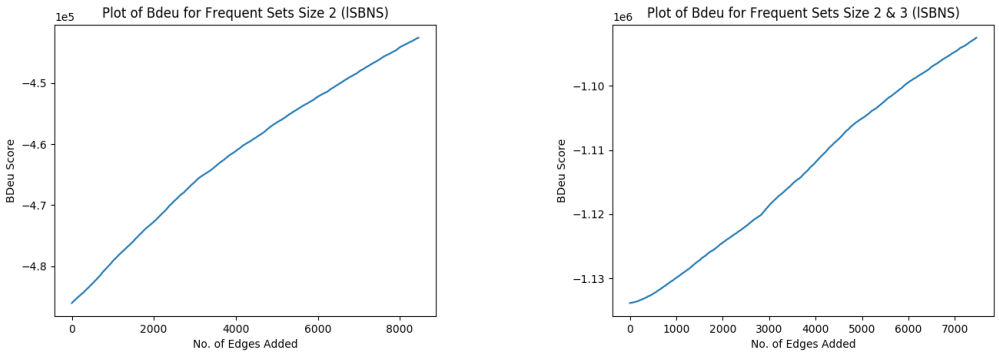
mechanism is different in SBNS, the fundamental point that support thresholds are inversely related to size of edge dump was demonstrated and the experiment produced an intuitive result.

Table 4: The results of different support sizes

	Initial BDeu score	Final BDeu score	Number of edges	Avg. BDeu score	Increase rate(%)
Support3	-402,763	-396,730	49,121	-8.0766	1.50
Support10	-402,763	-396,748	45,885	-8.6466	1.49
Support20	-402,122	-396,107	45,289	-8.7462	1.49
Support30	-401,426	-395,370	45,226	-8.7421	1.50

5.2 Comparisons of Different Maximum Frequent Set

Table 4 shows the comparison of MFS of 2 and MFS of 3. As MFS of 3 is a much larger case that requires longer running time, we set a threshold for number of edges added in the final Bayes net to be 7467. The final BDeu scores when the 7467th edge added into both cases are shown below. To compare the performance, average BDeu score defined by the final BDeu score divided by number of edges in edge dump could tell the difference. The score of MFS of 3 was higher than it of MFS of 2 which indicated that we got more information.



(a) $MFS = 2$ (b) $MFS = 3$

Figure 5: Increase of BDeu as more edges added

Another interesting observation was that the number of unique nodes in both cases were close. Ideally, they should have a large difference as in the process of finding the best DAG, MFS of 3 can possibly add some more unique nodes. However, the observation was different from our expectation. Based on the experiment and the data set, we have the following hypothesis: using joint probability as the measurement for finding the optimal DAG will not have much effect on adding new nodes in the final Bayes net, but adding weights in the edge dump to distinguish the existing edges.

Table 5: The results of MFS of 2 and MFS of 3

	MFS of 2	MFS of 3
Initial BDeu score	-486,031	-1,133,808
Edges added	7,467	7,467
Final BDeu score	-446,296	-1,092,541
Number of unique nodes	1,988	1,989
Number of edges in edge dump	29,157	72,693
Average BDeu score	-15.1796	-15.0295

5.3 Comparison of Support Model and the Oppose Model

Because of the prohibitive time cost of running a single experiment to completion, as an interim solution we ran our lSBNS algorithm on both support and oppose models for comparable durations and analyzed the results both models yielded. We found that the oppose model resulted in more edges learnt by our algorithm over the same time period - 5,700 edges as opposed to 2,000 edges in the support model. This ran counter to our intuition as the voting data generally contained more instances of people voting support (or abstaining) rather than opposing the election of a nominee. Since the data contains relatively few examples of oppose votes, we expected that the corresponding Bayes net would be sparser.

We have two hypotheses as to why this might be the case. Firstly, a smaller number of opposers means a sparser dataset, meaning it is highly likely that the support levels of the frequent sets would be lower and much more uniform (nearly all of them would be only 1). Therefore, given that these frequent sets would generate edge dumps containing edges of uniform importance. Thus, any edge from this set becomes very likely to improve BDeu scores when considered as a candidate and therefore be added into the Bayes net.

Another hypothesis stems from an interesting discovery we made when looking at the data closer. We discovered a handful of voters who fulfilled two conditions: (1) they participated in a relatively large number of elections, and (2) they voted negative for most nominees in these elections (they cast nearly no support votes at all). We think that in doing so this group of people established a large number of connections between nominees in the final oppose Bayes net. There is an interesting interpretation that the group would like to put forward - this small group of negative voters may be trolling and thereby polluting the election data. We find it very interesting to discover the presence of trolls through the analysis of our results and think it may be worthwhile in future work to apply similar analysis techniques for "trolling detection/identification". Given this election dataset is not very large, it is likely that the votes of the trolls affected the learnt Bayes net structure more than it would have affected a larger and more noise-resilient dataset. Table 6 shows the result of trolling detection.

Table 6: The result of trolling detection

Intersection edge	Trolling voter id	Support number	Opposing number	Neutral number
(2997, 2707)	841	22	41	2
(5506, 5509)	5020	45	99	33

Our last step in this experiment involved finding the overlap between connections learnt in the support model and those of the oppose model. We found that out of 2,000 edges in the support Bayes net and 5,700 in the oppose, the intersection size is surprisingly small - only 17 edges were found in the intersection between both sets. This is interesting because we expected that in practice certain nominees that had many supporters and opposers alike would cause both models to have significant overlaps. Contrapositively, the absence of overlaps indicates the absence of such controversial election figures. Perhaps it is the case that, for the most part, the political climate of the Wikipedia world is relatively tame, with most people playing nice and upvoting each other during elections. This to some extent informs us about how critically each voting member actually considers nominees before casting votes.

5.4 Comparison of SBNS Algorithm and lSBNS Algorithm

The results could be categorized into 3 parts: the final BDeu score, the computing time, and the sparsity. Table 7 shows the trend of BDeu scores for the original SBNS and the lSBNS algorithm, with support=3 and MFS=2. The initial BDeu score of lSBNS was much higher than that of SBNS, as it involved more nodes in the input data, so we mainly focused on the average and the increase rate of BDeu score in the Bayesian network construction step. For the average rate, we calculated the number of the unique nodes in SBNS edge dump and lSBNS edge dump. Next, the average BDeu score was given by final BDeu score divided by number of unique nodes. We found that the average BDeu score of lSBNS algorithm was higher than SBNS algorithm. In addition, we found that the lSBNS algorithm has higher increase rate, which means it is more efficient in fitting the data with

more proper structures. Although BDeu score could not be the only criteria to evaluate our algorithm, it to some extent could prove a better performance compared with SBNS algorithm.

As for the time aspect, since SBNS algorithm generates more candidate edges in edge dump, it will increase the running time for building the final Bayesian network structure. Therefore, lSBNS has shorter running time in the construction of Bayesian network. Also, we compared the time for generating the edge dumps in the two algorithms. For lSBNS algorithm, it took a few seconds to complete, but for SBNS algorithm, it took about a few hours. Thus, we make a hypothesis that the replacement of using joint probability with BDeu score will increase the efficiency of the whole algorithm.

For the other hand, we evaluated the sparsity of the Bayesian network by comparing the number of the edges added in the final Bayesian network. First of all, the size of the edge dump for SBNS algorithm is 438,644, while the size of the edge dump for lSBNS algorithm is 5,585. This provided us an indirect clue that the final Bayesian network learned by SBNS will have more edges included. This is further proved by the implementation of the algorithm. lSBNS added 3548 edges, but the SBNS algorithm added its 3600th edge when about 1/40 of the edge dump was scanned. Thus, our algorithm has sparser structure, which preserves most important information of the network at the same time. If only the key component of the graph is needed, then lSBNS algorithm will be a better choice.

Table 7: The comparison of SBNS and lSBNS algorithm

	Initial BDeu score	Final BDeu score	Number of unique nodes	Avg. BDeu score	Increase rate(%)
SBNS	-506255	-473377	1709	-276.99	6.5
lSBNS	-120743	-92804	1519	-61.10	23.1

6 Future Work and Conclusion

In conclusion, lSBNS algorithm performs well in the election dataset, especially in catching strong relationships between nominees. lSBNS algorithm demonstrates the advantages in BDeu score, time and sparsity those three aspects. If a user is willing to capture the core structure of the graph in short time, lSBNS algorithm is undoubtedly a better choice.

For future work we will enlarge the edge dump including higher order interactions to obtain achieve better Bdeu scores. Also, due to the limitation of the machines, we were not able to conduct experiments in the full election analysis in some settings, and we would like to use machines with better performance to conduct more extensive experiments.

During our experiments comparing supporting votes with opposing votes, we found a small handful of voters whose voting behaviour is less than constructive because they voted nearly everyone they saw negatively. An interesting side project to do would be to re-purpose the methods that we have developed for trolling filtering and detection.

Another area of improvement we identified was in the use of the BNLearn package to perform scoring of candidate networks. While BNLearn provided us much convenience during the implementation of the original SBNS algorithm and for our very own lSBNS modification, we noticed that in some instances the library did not work as expected. For instance, during the network building stage the best scoring candidate edge is greedily added to the network, but is checked for acyclicity before proceeding to finalize the addition of the edge because the result should be acyclic. However, we observed that despite performing checks for cycles using BNLearn utilities, some of our resultant networks still contain cycles. We propose in future work to implement our own BDeu scoring mechanism, and to also leverage on robust graph libraries.

In addition, we focused more on the relationship between nominees in this experiment. Furthermore, voter-election relation can be done using the same dataset. We could also evaluate our algorithm in those three aspects and test the universality and reasonability of lSBNS algorithm.

References

[1] Acemoglu, D., Dahleh, M. A., Lobel, I., & Ozdaglar, A. (2011). *Bayesian learning in social networks.* The Review of Economic Studies, 78(4), 1201-1236.

[2] Schulte, O., Luo, W., & Greiner, R. (2007, June). *Mind change optimal learning of bayes net structure.* In International Conference on Computational Learning Theory (pp. 187-202). Springer Berlin Heidelberg.

[3] Goldenberg, A., & Moore, A. W. (2005, August). *Bayes net graphs to understand co-authorship networks?.* In Proceedings of the 3rd international workshop on Link discovery (pp. 1-8). ACM.

[4] Goldenberg, A., & Moore, A. (2004, July). *Tractable learning of large Bayes net structures from sparse data.* In Proceedings of the twenty-first international conference on Machine learning (p. 44). ACM.

[5] Heckerman, D., Geiger, D., & Chickering, D. M. (1995). *Learning Bayesian networks: The combination of knowledge and statistical data.* Machine learning, 20(3), 197-243.

[6] G. Cooper and E. Herskovits. *A Bayesian method for constructing Bayesian belief network from databases.* In UAI 7, pages 86-94, 1991.

[7] D. Heckerman, D. Geiger, and D. Chickering.*Learning Bayesian Networks: The combination of knowledge and statistical data.* JMLR, 20:197-243, 1995.

[8] G. F. Cooper. *The computational complexity of probabilistic inference using Bayesian belief networks. Artif. Intell.*, 42(2-3):393-405, 1990.

[9] P. Dagum and M. Luby. *Approximating probabilistic inference in Bayesian belief networks is NP-hard.* Artif. Intell., 60(1):141-153, 1993.

[10] W. Buntine. *Theory refinement on Bayesian networks.* In UAI 7, pages 52-60, 1991.

[11] Ben-Gal, I. (2007).*Bayesian networks. Encyclopedia of statistics in quality and reliability.*

[12] Yang, S., & Chang, K. C. (2002). *Comparison of score metrics for Bayesian network learning.* IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 32(3), 419-428.

[13] Spirtes, P., & Glymour, C. (1991). *An algorithm for fast recovery of sparse causal graphs.* Social science computer review, 9(1), 62-72.

[14] Spirtes, P., Glymour, C. N., & Scheines, R. (2000). *Causation, prediction, and search.* MIT press.

[15] udea Pearl, T. V. J. (1991). *Equivalence and synthesis of causal models.* In Proceedings of Sixth Conference on Uncertainty in Artijicial Intelligence (pp. 220-227).

[16] Ma, B. L. W. H. Y., & Liu, B. (1998, August).*Integrating classification and association rule mining.* In Proceedings of the 4th.

[17] Bishop, Y., Fienberg, S., & Holland, P. (1977). *Discrete multivariate analysis: Theory and practice.* MIT Press.

[18] Sargur Srihari. *Structure Learning in Bayesian Networks.* Machine learning, New York. 1, May, 2014, Lecture.