JMLR: Workshop and Conference Proceedings 10: 22-34 The Fourth Workshop on Feature Selection in Data Mining

A Statistical Implicative Analysis Based Algorithm and MMPC Algorithm for Detecting Multiple Dependencies

Elham Salehi SALEHIE@UWINDSOR.CA
Jayashree Nyayachavadi SHREE@UWINDSOR.CA
Robin Gras RGRAS@UWINDSOR.CA
department of Computer Science
University of Windsor
Windsor, Ontario, N9B 3P4

Editor: Huan Liu, Hiroshi Motoda, Rudy Setiono, and Zheng Zhao

Abstract

Discovering the dependencies among the variables of a domain from examples is an important problem in optimization. Many methods have been proposed for this purpose, but few large-scale evaluations were conducted. Most of these methods are based on measurements of conditional probability. The statistical implicative analysis offers another perspective of dependencies. It is important to compare the results obtained using this approach with one of the best methods currently available for this task: the MMPC heuristic. As the SIA is not used directly to address this problem, we designed an extension of it for our purpose. We conducted a large number of experiments by varying parameters such as the number of dependencies, the number of variables involved or the type of their distribution to compare the two approaches. The results show strong complementarities of the two methods.

Keywords: Statistical Implicative Analysis, multiple dependencies, Bayesian network.

1. Introduction

There are many situations in which finding the dependencies among the variables of a domain is needed. Therefore having a model describing these dependencies provides significant information. For example, which variable(s) affect(s) the other variable(s) may be very useful for the problem of selection of variables; decomposition of a problem to independent sub-problems; predicting the value of a variable depending on other variables to solve the classification problem; finding an instantiation of a set of variables for maximizing the value of some function, etc (A. Goldebberg, 2004; Y. Zeng, 2008).

The classical model used for the detection of dependencies is the Bayesian network. This network is a factorization of the probability distribution of a set of examples. It is well known that the construction of a Bayesian network from examples is a NP-hard problem, thus different heuristic algorithms have been designed to to solve this problem (Neapolitan, 2003; E. Saheli, 2009) . Most of these heuristics are greedy and/or try to reduce the size of the exponential search space by a filtering strategy. The filtering is based on some measures that aim to discover sets of variables that have high potentiality to be mutually dependent or independent.

These measures rely on an evaluation of the degree of conditional independency. However other measures exist which are not based on conditional probability measurements that have the ability to discover dependencies. Using another measure that is not based on conditional dependencies can provide another perspective about the structure of dependencies of variables of a domain. Statistical Implicative Analysis (SIA) has already shown a great capability in extracting quasi-implications also called as association rules (R. Gras, 2008). We present a measure for multiple dependencies based on SIA and then use this measure in a greedy algorithm for solving the problem of multiple dependencies detection. We have compared our new algorithm for finding dependencies with one of the most successful conditional dependencies based heuristic introduced so far, MMPC (I. Tsamardinos, 2006). We have designed a set of experiments to evaluate the capacity of each of them to discover two kinds of knowledge: the fact that one variable conditionally depends on another one and the sets of variables that are involved in a conditional dependencies relation. Both of this information can be used to decompose the NP-hard problem of finding the structure of a Bayesian network into independent sub-problems and therefore can reduce considerably the size of corresponding search space.

This paper organized as follows: In the next section we describe the MMPC heuristic. In section 3 we present our SIA based measure and algorithm for finding multiple dependencies and the experimental results of the algorithms are presented in Section 4. Finally we conclude in section 5 with a brief discussion.

2. The MMHC Heuristic

Discovering multiple dependencies from a set of examples is a difficult problem. It is clear that this problem cannot be solved exactly when the number of variables approaches few dozens . However, for some problems, the number of variables can be several hundred or several thousand. Therefore, it is particularly important to have some methods to obtain an approximate solution with good quality. A local search approach is usually used in these problems. In this case the model of dependencies is built incrementally by adding or removing one or more dependencies at each step. The dependencies are chosen to be added or removed using a score that assesses the quality of the new model according to the set of examples (E. Saheli, 2009). In this approach the search space is exponential in terms of maximum number of variables on which a variable may depend. Therefore, there is a need to develop methods to increase the chances of building a good quality model without exploring the whole search space exhaustively. One possible approach is to use a less computationally expensive method to determine a promising subset of the search space on which we can subsequently apply a more systematic and costly method.

The final model is usually a Bayesian network in which the dependencies represent conditional independencies among variables. It is possible to build this model using information from other measures besides conditional probability. Indeed, the measurements in the first phase are used as a filter to eliminate the independent variables or bring the variables with shared dependencies together in several sub-groups. The second phase uses this filtered information to build a Bayesian network. The goal of our study is to compare the ability of two approaches for the detection of dependencies for the first phase. In this section a measure based on conditional probability is described and in the section 4 this measure will be compared with a SIA based measure.

2.1 Definition and Notation

A Bayesian network is a tool to represent the joint distribution of a set of random variables. Dependency properties of this distribution are coded as a direct acyclic graph (DAG). The nodes of this graph are random variables and the arcs correspond to direct influences between the variables.

We consider a problem consisting of n variables $v_1, v_2, \ldots, v_n$. Each variable v_i can take any values in set$M_i = m_{i,1}, m_{i,2}, \ldots, m_{i,k}$. For the detection of dependencies a set of N examples is available. Each example is an instantiation of each of the n variables in one of k possible ways.

Par_i, the set of all variables on which variablev_i depends, is the parent set of v_i .Any $v_j \in Par_i$ is a parent of v_i and v_i is a child of v_j. A table of conditional probability distribution (CPD), also known as the local parameters, is associated for each node of the graph. This table represents the probability distribution $P(v_i|Par_i)$.

2.2 MMPC Approach

Although learning Bayesian networks might seem a very well-researched area and even some exact algorithms have been introduced for networks with less than 30 variables (M. Koivisto, 2004), applying them to many domains such as biological or social networks, faces the problem of high dimensionality. In recent years several algorithms have been devised to solve this problem by restricting the space of possible network structures using various heuristics (N. Friedman, 1999; I. Tsamardinos, 2006). One of these algorithms, which has a polynomial complexity is "Sparse Candidate" algorithm (N. Friedman, 1999). The principle of this method is to restrict the parent set of each variable assuming that if two variables are almost independent in the set of examples, it is very unlikely that they are connected in the Bayesian network. Thus, the algorithm builds a small fixed-size candidate parent set for each variable. A major problem of this algorithm is to define the size of the possible parent sets and another one is that the algorithm assumes a uniform sparseness in the network. More recently, another algorithm called Max-Min Hill Climber (MMHC) has been proposed to solve these two problems and obtain better results on a wider range of network structures (I. Tsamardinos, 2006).This algorithm, uses a constrained based method to discover possible parents-children relationships and then uses them to build a Bayesian network. The first step of this algorithm, the one we use in this section to detect dependencies, is called Max-Min Parent Children (MMPC). The MMPC algorithm uses a data structure called parent-children set, for each variable v_i that contains all variables that are a parent or a child of v_i in any Bayesian network faithfully representing the distribution of the set of examples. The definition of faithfulness can be found in (Neapolitan, 2003; I. Tsamardinos, 2006). MMPC uses G^2 statistical test (P. Spirtes, 2000) on the set of examples to determine the conditional independency between pairs of variables given a set of other variables. The MMPC algorithm consists of two phases. In the first phase, an empty set of candidate parents-children (CPC) is associated with v_i. Then it tries to add more nodes one by one to this set using MMPC heuristic. This heuristic selects the variable v_j that maximizes the minimum association with v_i relative to current CPC and add this variable to it. The minimum association of v_j and v_i relative to a set of variables CPC is defined as below.

$$MinAssoc(v_i; v_j|CPC) = argminAssoc(v_i; v_j|S) \text{ for all subset } S \text{ of CPC.}$$

Assoc $(v_i, v_j|S)$ is an estimate of the strength of the association between v_i and v_j knowing the CPC and is equal to zero if v_i and v_j are conditionally independent given the CPC. The function Assoc uses the p-value returned by the G^2 test of independence: the smaller the p-value the higher the association. The first phase of MMPC stops when all remaining variables are considered independent of v_i given the subset of CPC. This approach is greedy, because a variable added in one step of this first phase may be unnecessary after other variables were added to the CPC. The second phase of MMPC tries to fix this problem by removing those variables in CPC which are independent of v_i given a subset of the CPC. Since this algorithm looks for candidate parents-children set for each node, if node T is in CPC of node X, node X should also be in CPC of node T.

What is not clear about these methods are their capabilities to discover any kind of structures and how different conditional probabilities and structures of real networks influence on the quality of results. We present the result we have obtained using the MMPC algorithm on examples generated from various Bayesian networks in Section 4.

3. SIA Based Approach

Statistical Implicative Analysis (SIA) (R. Gras, 2008) is a data analysis method that offers a framework for extracting quasi-implications also called as association rules. In a dataset D of N instances, each instance being a set of n Boolean variables, the implicative intensity measures to what extent variable b is true if variable a is true. The quality measure used in SIA is based on the unlikelihood of counter-examples where b is false and a is true. We are interested in the capabilities of SIA for finding multiple dependencies especially in situations that are difficult for conventional methods that are based on other measurements. For example, a situation in which two variables are independent but often take the same value in a large number of examples. We want to study the efficiency of SIA to refute the hypothesis of dependence by taking into account the counter examples. In order to use the SIA in general, some modifications are necessary. Indeed, we do not restrict ourselves to the binary variables and generalize the method for variables with higher cardinalities. We also want to be able to detect a situation where a combination of variables implies another variable, using an overall measure. In other word we want to measure one or more combinations of variables as the parents of a child variable. For example for variables A, B and $C \in \{0, 1, 2\}$, we want to define a measure which is able to detect a dependency from B and C to A because when $B = 0 \wedge C = 2, A = 1$ is abnormally frequent and when $B = 0 \wedge C = 0, A = 0$ is abnormally frequent. Current version of the SIA cannot be used for this purpose.

3.1 Definition and Notation

We use the following definitions and notations besides those presented in section 2.1. All the definitions presented here and the proofs for the rational of the measures and their properties can be found in (R. Gras, 2008). Let $Card(m_{i,j})$ be the number of times the variable v_i takes the value $m_{i,j}$ in N examples. $Card(\overline{m}_{i,j})$ is the number of times the variable v_i takes a value different from $m_{i,j}$ and $Card(m_{i_1,j_1}, m_{i_2,j_2})$ the number of times the variable v_{i_1} takes the value m_{i_1,j_1} and variable v_{i_2} takes value m_{i_2,j_2} in N examples.

Let π_i be an instantiation of the parents of v_i chosen from Π_i, the list of all combinations of instantiation of v_i parents. For example, in the previous example with the variables A, B and C, $\Pi_A = (0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)$. If $k = |M_i|$ then for each $v_j \in Par_i$

$$|\Pi_i| = k^{|Par_i|}.$$

Let $Card(\pi_i)$ be the number of times all parents of v_i, take value π_i in the N examples. Then the measure q extended from SIA is

$$q(\pi_i, m_{i,j}) = \frac{Card(\pi_i \wedge \overline{m}_{i,j}) - \frac{Card(\pi_i)\times Card(\overline{m}_{i,j})}{N}}{\sqrt{\frac{Card(\pi_i)\times Card(\overline{m}_{i,j})}{N}}}.$$

And the inclusion index $i(\pi_i, m_{i,j})$ for measuring the imbalances is extended from SIA is

$$i(\pi_i, m_{i,j}) = (\hat{I}^{\alpha}_{m_{i,j}/\pi_i} . \hat{I}^{\alpha}_{\overline{m}_{i,j}/\overline{\pi}_i})^{1/2\alpha}.$$

If we define function f as below

$$f(a,b) = \frac{Card(a \wedge \bar{b})}{card(a)}$$

Then

$$\hat{I}^{\alpha}_{m_{i,j}/\pi_i} = 1 + ((1 - f(\pi_i, m_{i,j}))\log_2((1 - f(\pi_i, m_{i,j})) + f(\pi_i, m_{i,j})\log_2((f(\pi_i, m_{i,j})).$$

If $Card(\pi_i \wedge \overline{m}_{i,j} \in [0, \frac{Card(\pi_i}{2}[$; otherwise, $\hat{I}^{\alpha}_{m_{i,j}/\pi_i} = 0$; and

$$\hat{I}^{\alpha}_{\overline{\pi}_i/\overline{m}_{i,j}} = 1 + ((1 - f(\overline{m}_{i,j}, \overline{\pi}_i))\log_2((1 - f(\overline{m}_{i,j}, \overline{\pi}_i)) + f(\overline{m}_{i,j}, \overline{\pi}_i)\log_2((f(\overline{m}_{i,j}, \overline{\pi}_i)).$$

In above equations $\alpha = 1$.The score we try to maximize is

$$s(\pi_i, m_{i,j}) = -i(\pi_i, m_{i,j}) \times q(\pi_i, m_{i,j}).$$

3.2 Extension of SIA

Unfortunately, the current SIA measure considers only one instantiation of the parent set at a time. If we want to consider all possible instantiations of the parent set we will obtain as many different dependency measures as there are different possible combination of instantiation. However, for each variable v_i, we need a single measure that represents its degree of dependency with its parent set. Therefore we must consider all the combination of variables for Π_i and use the measures $s(\pi_i, m_{i,j})$, to see how they imply all the possible values of v_i. Consequently we build a table T_i containing the set Π_{si} of measures s for all the combination of Π_i and M_i with size

$$k \times |\pi_i| = k^{|par(v_i)|}.$$

We tried various methods to combine the information of this table to a single measure. The simplest way is to consider just the maximum of Π_{si}. Other possibilities are to take the average of Π_{si} or the average of the $x\%$ of highest scores. We conducted many test with these approaches and none of them has yielded satisfactory results. In the first series of measures we considered the scores of one instantiation of π_i , but different values of M_i

B	**C**	**A= 0**	**A =1**	**A =2**	**Sup**	**E**
0	0	0	1.3	0.6	1.3	0.272
0	1	0	0	0	0	0
0	2	2.1	0	0.2	2.1	0.129
1	0	0	0	0	0	0
1	1	0.4	0.2	0.5	0.5	0.45
1	2	1.1	0	0	1.1	0
2	0	0	0	0	0	0
2	1	0	0	0	0	0
2	2	0	0	0	0	0

Table 1: An example of table T_i with A,B and $C \in$ {0, 1, 2}and A={(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)}.

independently. What we want to detect is that a value of π_i imply one specific instantiation of v_i and we want that it is true for several different instantiations of π_i. Therefore a measure is needed to detect that s is high for a couple $(\pi_i, m_{i,j})$ with $m_{i,j} \in M_i$ and low for all the others $m_{i,\bar{j}} \in M_i$ and that it is true for several π_i. We have therefore defined a score which combine, for a given π_i, the maximum value Sup_{π_i} of s for all $m_{i,j} \in M_i$ and the entropy E_{π_i} of s for all the values $m_{i,j} \in M_i$.

$$Sup_{\pi_i} = \max(s(\pi_i, m_{i,j})) \text{ where } 1 \leq j \leq k,$$

$$E_{\pi_i} = -\sum_{j=1}^{k} \frac{p(s(\pi_i, m_{i,j})) \log(p(s(\pi_i, m_{i,j}))}{\log(k)}$$

where

$$p(s(\pi_i, m_{i,j})) = \frac{s(\pi_i, m_{i,j})}{\sum_{\bar{j}=1}^{k} s(\pi_i, m_{i,\bar{j}})}.$$

For calculating a measure associated with a table T_i, we consider a set H of those π_i corresponding to the highest $x\%$ of Sup_{π_i} values in the table. Then the score of the table is

$$S_{i,Par_i} = \frac{\sum_{\pi_i \in H} Sup_{\pi_i}}{\sum_{\pi_i \in H} E_{\pi_i}}.$$

This is the measure we want to maximize. Table 1 presents T_A for the example with variables A, B, C. If you select the highest 20% Sup, only lines 1 and 3 will be selected and S_A will be equal to 8.48. In the following section we give an algorithm that uses this measure to determine the major dependencies of a problem.

3.3 SIA Based Algorithm

In previous section we defined a measure S_i for each variable v_i knowing its parent set. To determine the dependencies of a problem we should consider different possible configurations of parent sets for all variables and choose the configuration that leads to a maximum total score. Since the number of possible configurations is exponential in the number of variables, we need a heuristic approach. We chose an greedy approach for this heuristic. In the beginning of the algorithm we set the parent set of each variable to empty. Then at each step a new variable is chosen to be added to any of the parent sets using measure S. We stop adding variables when a fixed number of edges, maxEdge, has been added. The calculation of the table T_i is also exponential in the number of parents of variables so we restrict the maximum number of parents for each variable to four.The next variable to be added to a parent set is chosen by comparing the highest score of four different tables. The algorithm is presented in Table 2. This algorithm avoids calculating the score for all combinations of 2, 3 and 4 variables in a parent set. Only combinations that include x parents can be selected to calculate the score with $x + 1$ parents. The variable structMax includes: the score of the variable regarding its parent set, the child variable and the candidate parent variable to be added to the parent set. After initialization, table max_1 contains a list of the scores in descending order of all the combinations including one parent and one child. So there is n^2 scores in it. Tables max_2, max_3 and max_4 are initially empty. They are used to store the scores of child-parents combination when there are 2, 3 and 4 parents in the parent sets respectively. Thus at each stage of the algorithm, the variable to be added to the parent set of another variable will be determined by selecting the highest score of 4 tables. If Max_i is the selected table, the parent set of the variable associated with the maximum score for this table goes from i-1 to i variables. The score is then removed from the table and a new max score is calculated and inserted in the table max_{i+1}.The four tables are kept sorted in descending order so the maximum value of each table is always in position 0.

4. Experimental Study

In this section we study the capabilities of the MMPC heuristics and our SIA algorithm in finding the conditional dependencies and dependent variables involved in conditional dependencies.

4.1 Experimental Design

In our experiments, we use artificial data produced by sampling from randomly generated Bayesian networks. Each network has A arcs and n = 100 variables divided into two sets: a set of D variables for which there are direct dependency relations with at least one of the n-D-1 other variables; a set of variables I with no dependency relationship with any of the other n-1 variables. The CPD of each variable is randomly generated taking into account the possible dependency relations. Each variable can take 3 different values.

We represent the distribution of independent variables as a triplet such (p$_1$,p_2, p_3). For example (80, 10, 10) means that each random variable has a probability of 0.8 for one of its three possible values, and a probability of 0.1 for the other two. The value with a probability of 0.8 is chosen randomly among the three random variables. For distributions called 'random', each variable has a different distribution (p$_1$,p_2, p_3).

```
for all $V_i$
   $Par_i$ ={ $\emptyset$ }
$structMax = \{0, 0, 0\}$
$max_1 = \emptyset$
for sall $V_i$ {
   for all $v_j \neq v_i${
      if ($S_{i,Par_i+v_i} > structMax.score$) {
         $sturctMax.score = S_{i,Par_i+v_i}$
         $structMax.child = i$
         $structMax.parent = j$
      }
   }
   $max_1 = max_1 + structMax$
}
DescendingSort ($max_i$)
$max_2 = \emptyset, max_3 = \emptyset, max_4 = \emptyset$
$nbEdge = 0$
while ($nbEdge < maxEdge$) {
   $k = getIndexOfTableWithMaxScore(max_1, max_2, max_3, max_4)$
   $enf = maxk[0].child$
   $par_{child} = par_{child} + maxk[0].parent$
   if ($k < 4$) {
      $structMax = \{0, 0, 0\}$
      for all $v_j \notin par_{child}$ {
      if ($S_{i,Par_i+v_i} > structMax.score$) {
      $sturctMax.score = S_{i,Par_i+v_i}$}
      $structMax.child = i$
      $structMax.parent = j$
      }
   }
   $max_{k+1} = max_{k+1} + structMax$
   DescendingSort ($max_{k+1}$)
   $maxk[0]$ = {0, 0 ,0 }
   DescendingSort ($max_k$)
   $nbEdge = nbEdge + 1$
}
```

Table 2: SIA based Algorithm.

4.2 Evaluation of MMPC Heuristic

In this section, we study the ability of the MMPC algorithm to discover good parent-child sets of variables from data generated from Bayesian networks.

In our study, we vary the characteristics of the networks to analyze the consequences of this variation on the effectiveness of the MMPC algorithm. These changes include the distribution of independent variables I, the number of dependent variables D and the number of dependencies among the variables D (i.e. the number of arcs A in the network). The results are presented in Tables 3 to 4. Each row of these tables represents an average of results for 10 different sets of examples generated from 10 different networks but with the same characteristics. In each experiment, we calculate the mean and standard deviation of the number of true positive (TP), False Positives (FP), False Negative (FN) and the computational time. TP is the number of parent-children relationships correctly predicted by the algorithm. Thus, the number of TP at most can be twice the number of arcs of the network because if there is an arc between node X and node T it means each of them

Distribution of I	Average of TP	SD of TP	Average of FP	SD of FP	Average of FN	SD of FN	Run times(S)	precision= TP/(TP+FN)
(80, 10, 10)	30	7.29	116	11.33	49	7.28	6.5	37.5%
(50, 25, 25)	30	8.12	113	11.9	49	8.16	6.4	37.5%
(40, 30, 30)	29	7.28	117	11.63	51	7.28	6.7	36.25%
Random	29	6.76	118	10.54	50	6.78	6.2	36.25%

Table 3: Effectiveness of the MMPC algorithm according to the distribution of independent variables.

should be in the CPC(Candidate Parent-Children set) of the other node. In the same way, the number of FNs, i.e. the existing arcs in the network that have not been predicted by the algorithm, can be at most twice the number of arcs. The sum TP + FN is equal to twice the number of arcs of the network. The number of FP is the number of dependencies predicted by the algorithm and which do not exist in the network.

4.2.1 Finding the Dependencies

In this section, first we investigate the effects of the distribution of independent variables on the effectiveness of the MMPC algorithm. Bayesian networks used for this purpose include I = 75 independent variables and D = 25 dependent variables. The distribution used to generate the independent variables varies from almost uniform to completely random. The results are presented in Table 3. The number of arcs for all these networks is A = 40. One can see from these results that the distribution of independent variables has virtually no effect on the efficiency of the MMPC algorithm. The algorithm, under these conditions, was able to discover about 37% of dependencies. It may be noted that the number of FP is high, which means that the algorithm tends to predict many more dependencies than that really exists. In order to investigate the effect of the proportion of independent variables, we keep the ratio A / D almost the same while changing the numbers D and I (n remains equal to 100). As it can be seen from the results presented in the first three rows of Table 4, when the network contains only the dependent variables (D = 100), the MMPC algorithm performs much better and is able to find almost 80% of dependencies. However, where the number of dependent variables is equal to 25, only about 35% of the dependencies are discovered. The number of FP is also very low when all variables are dependent. It seems this method has difficulty in determining the independent variables. However, it can be noted that the run time increases considerably in the case where all variables are dependent. This can be problematic when the number of variables in the problem is much higher than 100.

If we vary the number of arcs in the networks with n dependent variables(D = n, I=0), like in previous section,the TP is high. However, the percentage slightly decreases when the complexity of the networks increases. But it seems that the complexity is less important than the proportion of dependent and independent variables. Although, it should be noticed that the complexity of the network influences the computation time.

4.2.2 Problem of Selection of Variables

We mentioned in the introduction the possibility of methods that detect dependencies for the selection of variables involved in dependency relations. The idea is to decompose the original problem by locating the independent variables (those with empty candidate parent-children sets) for which the optimization can be performed independently. As the search space is reduced, the chance of finding a good quality solution is increased. The problem here

D	I	A	Average of TP	SD of TP	Average of FP	SD of FP	Average of FN	SD of FN	Run times (s)	precision= TP/(TP+FN)
25	75	40	29	6.76	118	10.54	50	6.78	0.31	36.2
50	50	80	53.4	8.81	99.4	11.35	106.6	8.81	0.34	33.4
100	0	150	243.8	8.17	16.6	6.81	56.2	8.17	21.1	81.3
25	0	30	52.4	3.55	2.4	1.96	7.6	3.55	0.31	87.3
25	0	40	65.6	4.17	2.2	1.89	14.4	4.17	1.83	82
25	0	60	91.8	4.51	3.2	3.37	28.2	4.51	6.63	76.5
100	0	120	200.6	5.51	25.2	7.28	39.4	5.52	9.98	83.6
100	0	150	243.8	8.17	16.6	6.81	56.2	8.17	21.1	81.3
100	0	200	312.4	9.67	10.8	3.37	87.6	9.67	28.3	78.1

Table 4: The average efficiency of MMPC algorithm regarding the proportion of independent variables and complexity of the Bayesian network.

Distribution	D	A	TP	TN
random	25	60	24.2	11.4
random	25	40	23.2	12.4
random	25	30	24	10.6
(80, 10, 10)	25	40	23.8	12.6
(50, 25, 25)	25	40	23.8	13.6
(40, 30, 30)	25	40	23.8	13.8

Table 5: Results obtained by the MMPC algorithm for the problem of selection of variables.

is slightly easier than the one studied in section 4.2.1 because the goal here is to determine the list of variables involved in dependency relationships without finding the dependencies precisely. We therefore conducted a series of experiments to measure the capacity of the MMPC on this problem.

We used networks with different independent variable distributions generated using the method described in section 4.1. We also vary the complexity of the networks by changing the number of arcs. The results are presented in Table 5. Although the MMPC approach could discover more than 90% of the dependent variables in Table 5 (23 out of 25), it discovered just about 17% of independent variables (TN in Table 5). This means that this method tends to significantly overestimate the number of dependencies. The results are little affected by changing distributions of independent variables and the complexity of the network (results not shown). It seems that this method cannot be used for the problem of selection of variables because almost all variables are selected.

4.3 Evaluation of SIA Based Algorithm

We repeated the same experiences as those in Section 4.2 to evaluate our SIA based detection algorithm in order to achieve the most possible honest comparison. It should be noted though that this disadvantaged SIA. Indeed, the data were generated from the models, Bayesian networks, which are based on conditional probability measurement. The SIA approach uses an alternative measure that does not have the same properties. In particular, a very significant difference is that the Bayesian network model is not transitive while the SIA is. But a totally fair comparison is not possible and, taking into account these differences in our analysis, this comparison seemed to be the best way to proceed.

Dist. of I & maxEdge	Avg of TP	Avg of FP	Av of FN	P	Run Time
(80, 10, 10), 35	0.9	34.1	39.1	2.25	37
(50, 25 ,25), 35	6.7	28.3	33.3	16.7	61.7
(40, 30, 30), 35	7.8	27.2	32.2	19.5	69.3
Random, 35 ,	0.6	34.4	39.4	1.5	44.6
(80, 10, 10), 50	1	49	39	2.25	46.1
(50, 25 ,25), 50	8.4	41.6	31.6	21	76.4
(40, 30, 30), 50	11	39	29	27.5	89.8
random , 50	1.2	48.8	38.8	3	57.4
(80, 10, 10), 150	1.2	148.8	38.8	3	63.6
(50, 25 ,25), 150	12.3	137.7	27.7	30.7	179
(40, 30, 30), 150	15.3	134.7	24.7	38.2	184
random , 150	4.9	145.1	36.1	12.2	194

Table 6: Results based on distribution of I, x=10%.

Dist. of I & maxEdge	Avg of TP	Avg of FP	Av of FN	P	Run Time
(80, 10, 10), 35	0.2	34.8	39.8	0.5	33.8
(50, 25 ,25), 35	7.7	27.3	32.3	19.25	59.7
(40, 30, 30), 35	5.1	29.9	34.9	12.7	66.4
random ,35	0.4	34.6	39.6	1	36.9
(80, 10, 10) , 50	0.2	49.8	39.8	0.5	42.3
(50, 25 ,25) , 50	6.2	43.8	33.8	15.5	70.2
(40, 30, 30) , 50	7.1	42.9	32.9	17.7	80.9
random , 50	0.5	49.5	39.5	1.25	41.3
(80, 10, 10), 150	0.3	149.7	39.7	0.75	55.5
(50, 25 ,25) , 150	6.6	143.3	33.4	16.5	164.2
(40, 30, 30), 150	8	142	32	20	177.9
random , 150	4.2	145.8	36.8	10.5	140.9

Table 7: Results based on distributions of I, x=50%.

4.3.1 Finding the Dependencies

We use the same data as in section 4.2. Our algorithm uses several parameters: the percentage of best Sup, x for each table T_i and the maximum number of variables to be added to all parent sets, maxEdge. For each of these parameters we used different values. Those we found most relevant and we presented here are 10% and 50% for x and 35, 50 and 150 edges for maxEdge parameter. We have evaluated three different configurations corresponding to a real situation in which we do not know the number of dependencies of the problem in advance. Actually we search slightly less, slightly more and much more dependencies that really exist by setting maxEdge to 35, 50 and 150 respectively. The results presented in Tables 6 and 7 indicate that our algorithm discovered few dependencies. The measure appears more sensitive to the distribution used to generate the independent variables. The results obtained with the value x = 10% is slightly better. The calculation time is also higher than the max-min algorithm, but our program has not yet been optimized for computational efficiency.

4.3.2 The Problem of Selection of Variables

We used the same data sets to test the ability of our algorithm to solve the problem of selection of variables involved in dependencies relation. The results are presented in Tables 8 and 9 and show a strong potential of our algorithm for this problem. The results are much better than those obtained with the max-min algorithm. Although the number of TP is slightly lower, the number of FP is considerably lower. What is most important is the fact that the level of prediction is much better that one would expect by chance. As the ratio of dependent variables to the number of independent variables is 1/3 in the model used to generate the data, a random prediction would give the same ratio of TP / FP (ie, in this case TP / (75-TN)). In Tables 8-9 in column TP / (0.33xFP), we present the gain compared to a random selection of variables. In the cases with distributions of independent variables (40, 30, 30) and (50, 25, 25) the gain is very significant, up to 16.1. For comparison, the results of the max-min algorithm show more stability, but a gain that never exceeds 1.18. Our algorithm seems to have more difficulty when the independent variables have extreme distributions, 'random' or (80, 10, 10). With x = 10% and when we search less dependencies that it really exists (35 Edges), the gain is always at least 1. Although this is a first version, our algorithm seems to have a very high potential to detect the dependent variables and thus to solve the problem of selection of variables. We also tested our algorithm on the data

Dist. of I, maxEdge	Avg of TP	Avg of TN	TP/(0.33xFP)
(80, 10, 10), 35	6.7	55.5	1.03
(50, 25 ,25), 35	15.5	71.8	14.7
(40, 30, 30), 35	15.4	72.1	16.1
random, 35	3.4	64.7	1
(80, 10, 10), 50	6.8	46.4	0.73
(50, 25 ,25), 50	18.3	68.7	8.79
(40, 30, 30), 50	18.3	69.7	10.5
random, 50	6.1	61.3	1.36
(80, 10, 10) 150	9.2	13	0.45
(50, 25 ,25) 150	22.8	31.8	1.6
(40, 30, 30) 150	21.8	46.7	2.33
random 150	17.3	32.6	1.24

Table 8: Results for selection of variables, x=10%.

Dist. of I, maxEdge	Avg of TP	Avg of TN	TP/(0.33xFP)
(80, 10, 10), 35	7.6	60.4	1.58
(50, 25 ,25), 35	16.8	66.2	5.79
(40, 30, 30), 35	14.8	67	1.85
random, 35	4	57	0.67
(80, 10, 10), 50	8	53.5	1.13
(50, 25 ,25), 50	18.3	59.9	3.67
(40, 30, 30), 50	18.1	63.7	4.85
random, 50	5.5	50	0.67
(80, 10, 10), 150	11.8	6.7	0.52
(50, 25 ,25), 150	22.1	22.8	1.28
(40, 30, 30), 150	22	41.2	1.97
random, 150	22	16.5	1.14

Table 9: Results for selection of variable, x= 50%.

presented in section 2.3.2 in which D = 50, I = 50 and A = 80 (results not presented here). The results show that with configuration x = 10%, the gains are between 1.28 and 1.82.

5. Conclusion

We conducted a study on the capabilities of two methods based on different measures for discovering the dependencies of a problem: 1) the max-min algorithm, which is based on the test of conditional dependency G^2; 2) an algorithm that we developed based on an extension of the SIA measure. We applied these algorithms to several datasets by varying the parameters of the problem such as the distribution of independent variables, the number of dependent variables and the number of dependencies. We also considered two different problems: to determine the dependencies relations and to identify the variables involved in the dependency relationships. Of course finding a solution for the first problem can also solve the second. However, it is generally not possible to directly and fully resolve this problem. Being able to see at first just what is the subset of variables involved in the set of dependencies reduces the complexity of the first problem and thus help to reach a better solution.

Our results showed a good efficiency of the max-min algorithm for discovering the dependencies when all the variables of the problem are involved. The algorithm appears to be little affected by the change in the complexity of the model and the distributions of the independent variables. However, it has some significant limitations to detect dependencies when part of the variables is independent. The algorithm max-min does not appear to be effective for the second problem: the selection of variables. Our SIA based algorithm, does not seem capable of directly detecting the dependencies whatever the configuration was. But it seems very effective to determine the dependent variables. However, it is less efficient in situations where the independent variables have extreme distributions like (80, 10, 10) or 'random'. The two approaches seem complementary and promising. It would be very interesting to develop a method combining these two approaches. In a first phase our algorithm, using the extended version of the SIA, would select a subset of variables for which there is a strong presumption of dependency. Then, in a second phase, the max-min approach is applied to this sub-set to determine more precisely where these dependencies are. All these information would then be used to build a Bayesian network. It would be also interesting to compare the methods based on the importance of the dependencies using some connection strength (Ebert-Uphoff, 2007) measure instead of just counting the number of discovered dependencies. It would be also interesting to compare the modified SIA with multi-dimensional form of classical measures to detect correlation between variable distributions.

References

A. Moore A. Goldebberg. Tractable learning of large bayes net structures from sparse data. In 21th *International Conference on Machine Learning*, pages 44–51, 2004.

R. Gras E. Saheli. An empirical comparison of the efficiency of several local search heuristics algorithms for bayesian network structure learning. In *Learning and Intelligent Optimization IEEE international conference*, 2009.

I. Ebert-Uphoff. Measuring connection strenghts and link strenghts in discrete bayesian networks. Technical Report GT-IIC-07-01, Georgia Tech, College of Computing, January 2007.

C.F. Aliferis I. Tsamardinos, L. E. Brown. The mmpc hill-climbing bayesian network structure learning algorithm. *Machine Learning Journal*, 65(1):31–78, 2006.

K Sood M. Koivisto. Exact bayesian structure discovery in bayesian networks. *Journal of Machine Learning Research*, 5:549–573, 2004.

D. Peer N. Friedman, I. Nachman. Learning bayes network structure from massive datasets: The "sparse candidate" algorithm. In *15th Conference on Uncertainty in Artificial Intelligence*, pages 206–215, 1999.

R. Neapolitan. *Learning Bayesian networks.* Prentice Hall, 2003.

R. Scheines P. Spirtes, C. Glymour. *Causation, prediction, and search.* The MIT Press, second Edition, 2000.

P Kuntz et al R. Gras. *An overview of the Statistical Implicative. In: Statistical Implicative Analysis.* Springer-Verlag, 2008.

C.A. Hernandez Y. Zeng. Decomposition algorithm for learning bayesian network structures from data. In *PAKDD 2008, Lecture Notes in Artificial Intelligence*, pages 441–453, 2008.