Tabu Search Enhanced Markov Blanket Classifier for High Dimensional Data Sets

Xue Bai

January 2005
CMU-CALD-05-101

Advised by:
Peter Spirtes

Center for Automated Learning and Discovery
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

Data sets with many discrete variables and relatively few cases arise in health care, ecommerce, information security, text mining, and many other domains. Learning effective and efficient prediction models from such data sets is a challenging task. In this paper, we propose a Tabu Search enhanced Markov Blanket (TS/MB) procedure to learn a graphical Markov Blanket classifier from data. The TS/MB procedure is based on the use of restricted neighborhoods in a general Bayesian Network constrained by the Markov condition, called Markov Blanket Neighborhoods. Computational results from real world data sets drawn from several domains indicate that the TS/MB procedure is able to find a parsimonious model with substantially fewer predictor variables than in the full data set, and provides comparable prediction performance when compared against several machine learning methods.

Keywords: Markov Blanket, Bayesian Networks, Tabu Search

1. Introduction

The deployment of comprehensive information systems and online databases has made extremely large collections of real-time data readily available. In many domains such as genetics, clinical diagnoses, direct marketing, finance, and on-line business, data sets arise with thousands of variables and a small ratio of cases to variables. Such data present dimensional difficulties for classification of a target variable (Berry and Linoff, 1997), and identification of critical predictor variables. Furthermore, they pose even greater challenges in the determination of actual influence, i.e., causal relationships between the target variable and predictor variables. The problem of identifying essential variables is critical to the success of decision support systems and knowledge discovery tools due to the impact of the number of variables on the speed of computation, the quality of decisions, operational costs, and understandability and user acceptance of the decision model. For example, in medical diagnosis (Cooper et al., 1992), the elimination of redundant tests may reduce the risks to patients and lower health care cost; in interactive marketing (Jesus, 2001), conducting consumer profile analysis using non-essential variables tends to increase the risks of a firm's strategic decisions, raise customers' privacy concerns, hurt consumers' trust and the firm's reputation, and reduce profitability; in the text mining field, being able to extract sentiments from unstructured text would be helpful in business intelligence applications and recommender systems where user input and feedback could be quickly summarized (Pang et al., 2002).

In this study, we address this problem of efficiently identifying a small subset of predictor variables from among a large number, using *Markov Blanket* (MB) and *Tabu Search* (TS) approaches. The Markov Blanket of a variable Y, (MB(Y)), by definition, is the set of variables such that Y is conditionally independent of all the other variables given $MB(Y)$. A *Markov Blanket Directed Acyclic Graph* (MB DAG) is a Directed Acyclic Graph over that subset of variables. When the parameters of the MB DAG are estimated, the result is a *Bayesian Network*, defined in the next section. Recent research by the machine learning community (Tsamardinos et al., 2002, Madden, 2003, Tsamardinos et al., 2003, Margaritis, 2003) has sought to identify the Markov Blanket of a target variable by filtering variables using statistical decisions for conditional independence and using the MB predictors as the input features of a classifier.

However, learning MB DAG classifiers from data is an open problem (Chickering, 2002). There are several challenges: the problem of learning the graphical structure with the highest score (for a variety of scores) is NP hard (Chickering et al., 2003); for methods that use conditional independencies to guide graph search, identifying conditional independencies in the presence of limited data is quite unreliable; and the presence of multiple local optima in the

space of possible structures makes the search process difficult. In this paper, we propose a Tabu Search enhanced Markov Blanket procedure that finds a MB DAG for a target variable. This two-stage algorithm generates an MB DAG in the first stage as a starting solution; in the second stage, the Tabu Search metaheuristic strategy is applied to improve the effectiveness of the MB DAG as a classifier, with conventional Bayesian updating and a heuristic method of estimating parameters.

Classification using the Markov Blanket of a target variable in a Bayesian Network has important properties: it specifies a statistically efficient prediction of the probability distribution of a variable from the smallest subset of variables that contains all of the information about the target variable; it provides accuracy while avoiding overfitting due to redundant variables; and it provides a classifier of the target variable from a reduced set of predictors. The TS/MB procedure proposed in this paper allows us to move through the search space of Markov Blanket structures quickly and escape from local optima, thus learning a more robust structure.

This paper is organized as follows: Section 2 provides background and literature review. Section 3 presents our proposed method and examines several relevant issues such as move selection, neighborhood structure, and evaluation metric for our specific problem. Section 4 presents the layout of the experimental design. Section 5 details the conduct of the experiments and the analysis of the computational results in a health care case study. Section 6, 7 and 8 present the applications in three different domains: text mining, health care and Internet marketing. Section 9 summarizes our general findings. Section 10 studies the theoretical properties of our algorithm.

2. Representation and Background Knowledge

Definition and Notation

Bayesian Networks and Markov Blankets. A *Bayesian Network* is a graphical representation of the joint probability distribution of a set of random variables. A Bayesian Network for a set of variables $X = \{X_1, ..., X_n\}$ consists of: (i) a directed acyclic graph (DAG) S that encodes a set of conditional independence assertions among variables in X; (ii) a set $P = \{p_1, ..., p_n\}$ of local conditional probability distributions associated with each node and its parents. A DAG is an ordered pair of a set of vertices, X, and a set of directed edges, where each directed edge is an ordered pair of distint vertices in X. If there is a directed edge $X \rightarrow Y$, in S, then X is a parent of Y and Y is a child of X.

DEFINITION 1 *A joint probability distribution p satisfies the Markov condition for DAG S if every node X_i in S is independent of its non-descendants and non-parents in S, conditional on its parents*[1].

The Markov Condition implies that the joint distribution p can be factorized as a product of conditional probabilities, by specifying the distribution of each node conditional on its parents (Pearl, 2000). In particular, for a given DAG S, the joint probability distribution for X can be written as

$$p(X) = \prod_{i=1}^{n} p_i(X_i|pa_i) \ , \tag{1}$$

where pa_i denotes the set of parents of X_i in S; this is called a Markov factorization of p according to S.

DEFINITION 2 *Given the set of variables X and target variable Y, a Markov Blanket (MB) for Y is a smallest subset Q of variables in X such that Y is independent of $X \backslash Q$, conditional on the variables in Q.*

DEFINITION 3 *p is faithful to the DAG S with the vertex set X if and only if there are no conditional independence relations in p other than those entailed by satisfying the Markov condition for S.*

The set of distributions represented by S is the set of distributions that satisfy the Markov condition for S. If p is faithful to the graph S, then given a Bayesian Network (S, p), there is a unique Markov Blanket for Y consisting of pa_Y, the set of parents of Y; ch_Y, the set of children of Y; and $pa\,ch_Y$, the set of parents of children of Y.

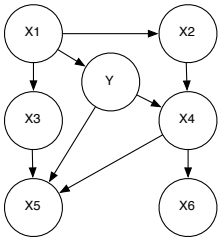


Figure 1. The Bayesian Network (S, P)

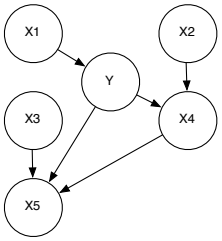


Figure 2. A Markov Blanket DAG for Y

For example, consider the two DAGs in Figure 1 and 2, above. The factorization of p entailed by the Bayesian Network (S, P) is

$$\begin{aligned} p(Y, X_1, ..., X_6) \quad = \quad & p(Y|X_1) \cdot p(X_4|X_2, Y) \cdot p(X_5|X_3, X_4, Y) \cdot \\ & \cdot p(X_2|X_1) \cdot p(X_3|X_1) \cdot p(X_6|X_4) \cdot p(X_1) \ , \end{aligned} \tag{2}$$

The factorization of the conditional probability $p(Y|X_1, ..., X_6)$ entailed by the Markov Blanket for Y corresponds to the product of those (local) factors in equation (2) that contain the term Y.

$$p(Y|X_1, ..., X_6) = C' \cdot p(Y|X_1) \cdot p(X_4|X_2, Y) \cdot p(X_5|X_3, X_4, Y) \ , \tag{3}$$

where C' is a normalizing constant independent of the value of Y.

DEFINITION 4 *MB DAGs that entail the same set of conditional independence relations are Markov equivalent; and the set of all MB DAGs that are Markov equivalent belong to the same Markov equivalence class.*

Tabu Search Heuristic. A *Heuristic* is an algorithm or procedure that provides a shortcut to solving complex decision problems. Heuristics are used when you have limited time and/or information to make a decision. For example, some optimization problems, such as the travelling salesman problem, may take far too long to compute an optimal solution. A good heuristic is fast and able find a solution that is no more than a few percentage points worse than the optimal solution. Heuristics lead to a good decisions most of the time, but not always.

There have been several Meta-heuristic application recently in Machine Learning, Evolutionary Algorithms, and Fuzzy Logic problems. For instance, "What parameter settings do I use to get good results when applying heuristic method X to problem Y?" "How do I adjust the parameters of heuristic X so I get better results on problem Y?" "Which is 'better', heuristic X or heuristic Y?"

DEFINITION 5 *A Meta-heuristic(Greenberg, 2004) is*
(1) A high-level algorithmic framework or approach that can be specialized to solve optimization problems, or
(2) A high-level strategy that guides other heuristics in a search for feasible solutions

Tabu Search is a *meta-heuristic* strategy that is able to guide traditional local search methods to escape the trap of local optimality with the assistance of adaptive memory (Glover, 1997). Its strategic use of memory and responsive exploration is based on selected concepts that cut across the fields of artificial intelligence and operations research.

Given an optimization problem $Minimize\, c(x) : x \in X$, where $c(x)$ is the objective function, which may be linear or nonlinear, and x is any solution in the solution space X, many procedures for solving optimization problems can often be characterized by reference to sequences of *moves* that lead from one solution ($x \in X$) to another.

DEFINITION 6 *A Move s is a mapping defined on a subset of $X(s)$ of X,*

$$s : X(s) \rightarrow X, s \in S, \tag{4}$$

where X is the solution space. In some settings, X may be a superset of the solutions which are feasible. S is a set of moves. S can be written as $S(x)$; $S(x)$ can be viewed as a neighborhood function of x.

A simple version of Tabu Search may be expressed as follows:

Simple Tabu search (Glover, 1989)

1 Select an initial solution $x \in X$, and let $x^* := x$ and $x_0 := x$.
Set iteration counter $k = 0$ and tabu list $TL = \emptyset$.

2 If $S - TL = \emptyset$, then go to step 4;
Otherwise $k := k + 1$ and select $s_k \in S - TL$ such that
$s_k(x_{k-1}) = OPTIMUM(s(x_{k-1}) : s_k \in S - TL)$.

3 Let $x_k = s_k(x_{k-1})$. If $c(x_k) < c(x^*)$ where x^* donates the best solution currently found, let $x^* = x_k$.

4 If a chosen number of iterations has elapsed either in total number or since x^* was last improved or $S - TL = \emptyset$ upon reaching this step from step 2, stop.
Otherwise, update TL (explained below) and return to step 2.

The mathematical definitions of the *tabu list* and *tabu tenure* are as follows:

DEFINITION 7 *Tabu list (TL) is given by*

$$TL = \{s^{-1} : s = s_i, i > k - t, \} \tag{5}$$

where k is the iteration index and s^{-1} is the inverse of the move s; i.e., $s^{-1}(s(x)) = x$. In words, TL is the set of those moves that would undo one of those moves in the t most recent iterations. t is called the tabu tenure.

The use of TL provides the "constrained search" element of the approach, and hence the solution generated depends critically on the composition of TL and the way it is updated. Tabu search makes no reference to the condition of local optimality, except implicitly where a local optimum improves on the best solution. A *best* move, rather than an improving move is chosen at each step. This strategy is embedded in the $OPTIMUM$ function.

Tabu search is viewed as "intelligent" search because it makes use of adaptive memory. The adaptive memory feature of TS allows the implementation of procedures that are capable of searching the solution space economically and effectively. Memoryless heuristics, such as semi-greedy heuristics and the prominent genetic algorithm(GA) and simulated annealing(SA) approaches, rely heavily on semi-random processes that implement a form of sampling. *Recency-based memory structure* and *frequency-based memory structure* are

two most commonly used structures. They are used in *short-term Tabu search memory* and *long-term Tabu search memory* respectively.

DEFINITION 8 *Recency-based memory, a memory structure used in short-term Tabu search memory, keeps track of solution attributes that have changed during the recent past.*

Attributes that have changed in the recent solutions are labelled as *tabu-active* in Recency-based memory. Solutions that contain tabu active elements, or particular combination of the elements will not be revisited for a short term. The tabu list is one specification of recency-based memory structure. While Tabu classification strictly refers to the solutions that are forbidden to be visited, it also often refers to the moves that lead to such solutions as being tabu.

DEFINITION 9 *Frequency-based memory, a memory structure used in long-term Tabu search memory, records the counts of each move in the search history. In non-improving phases, frequency based memory also penalizes choices of moves that drive toward configurations often visited.*

Long-term memory helps to guide the search to new regions once the search by short-term memory get stuck, and determines the new starting solution for the search. Two strategies that make use of short-term memory and long-term memory are *intensification* and *diversification* strategies. Intensification strategies are based on modifying choice rules to encourage move combinations and solution features historically found good. They may also initiate a return to attractive regions to search them more thoroughly. The *candidate list* is an important consideration during the intensification stage or even in the general Tabu search process. When the neighborhood is large, narrowing the examination of elements of $S(x)$ can achieve an effective tradeoff between the quality of x and the effort expended to find it. The diversification stage on the other hand encourages the search process to examine unvisited regions and to generate solutions that differ in various significant ways from those seen before.

A version of Tabu Search that uses both short-term memory and long-term memory may be formulated as follows:

Tabu search with diversification

1 Select an initial solution $x \in X$, and let $x^* := x$, and $x_0 := x$.
Set the iteration counter $k = 0$, the non-improving move counter $l = 0$, and tabu list $TL = \emptyset$.

2 If $S - TL = \emptyset$, then go to step 4;
Otherwise $k := k + 1$ and select $s_k \in S - TL$ such that $s_k(x) = OPTIMUM(s(x_{k-1}) : s_k \in S - TL)$.

3 Let $x_k = s_k(x_{k-1})$.
If $c(x_k) < c(x^*)$ where x^* donates the best solution currently found, let $x^* = x_k$; $l := 0$
Otherwise, $l := l + 1$

4 If a chosen number of iterations has elapsed either in total number, or l reaches the maximum value non-improving count, or $S - TL = \emptyset$, upon reaching this step from step 2, stop.
Otherwise, update TL and return to step 2.

Having introduced the definition and representation schema, in the next subsection, we briefly survey the literatures relevant to our study.

Background Literature

There are a few recent studies using Markov Blankets in the machine learning literature (Koller and Sahami, 1996, Margaritis and Thrun, 1999, Tsamardinos et al., 2003). But few of these generate the Markov Blanket from data, and those are usually for small number of variables. Those studies use the notion of Markov Blanket as a set of variables; none of them have generated the DAG of Markov Blankets, nor have they use it for Bayesian inference in classification problems. Theoretically correct Bayesian algorithms in the large sample limit (Chickering, 2002) for finding DAGs are now known, but have not been applied to the problem of finding Markov Blankets for data sets with large numbers of variables.

Koller and Sahami (Koller and Sahami, 1996) first used a heuristic procedure to find the Markov Blanket variables for a target variable in data sets with large numbers of variables. The heuristic is based on two assumptions that are not always true in real world data: (1) The target influences the predictors; and (2) that the variables most strongly associated with the target are in its Markov Blanket. No classifier is studied. In Kohler and Sahami's experiments with large variable sets, one hundred or more predictor variables remain.

Margaritis and Thrun proposed the GS algorithm in 1999 (Margaritis and Thrun, 1999). GS uses a measure of association with the target variable and conditional independence tests to find a reduced set of variables estimated to be the Markov Blanket, and use the set of Markov Blanket nodes to learning a Bayesian network structure. The scoring criterion they used is global in the sense that the evaluation function chooses the structure that fits best the overall structure.

Aliferis et al. proposed IAMBnPC in 2003 (Tsamardinos et al., 2003). IAMBnC uses a dynamical variant of the feature selection filter, followed by the PC algorithm (Spirtes et al., 2000). No graphical Markov Blanket DAG is

generated from the modified PC algorithm output. A variant IAMBnC, interIAMBnPC (Cheng et al., 2002), interleaves the PC algorithm with the filter. Another feature selection procedure that uses Markov Blanket notion, HITON, (Aliferis et al., 2003), supplements the dynamic variable filter of IAMBnPC with a "wrapper" using any of several non-Bayesian classifiers, and then classifies the target with the non-Bayesian classifier. A graphical Markov Blanket is not produced. The results are compared on five empirical data sets from biomedicine domains, each with a very large ratio of variables to cases.

Tabu search has been applied successfully to a wide variety of continuous and combinatorial optimization problems (Johnson and McGeoch, 1997, Toth and Vigo, 2003), and is capable of reducing the complexity of the search process and accelerating the rate of convergence. In its simplest form, Tabu Search starts with a feasible solution and chooses the *best move* according to an evaluation function while taking steps to ensure that the method does not re-visit a solution previously generated. This is accomplished by introducing *tabu restrictions* on possible moves to discourage the reversal and in some cases repetition of selected moves. The tabu list that contains these forbidden move attributes is known as the short term memory function. It operates by modifying the search trajectory to exclude moves leading to new solutions that contain attributes (or attribute mixes) belonging to solutions previously visited within a time horizon governed by the short term memory. Intermediate and long-term memory functions may also be incorporated to intensify and diversify the search.

We were motivated to use Tabu Search because its adaptive memory capability - both short term and long term - appears particularly suited to the Bayesian Networks and Markov Blanket approaches. Our choice of TS was also motivated by the extensively documented situations where its adaptive memory capability has proved successful, both directly and embedded within other "hybrid" methods such as those involving genetic algorithms, evolutionary computation methods (http://www.tabusearch.net/, Glover, 1997) and scatter search (Rego and Alidaee, 2004). Certainly it would be worthwhile to investigate additional metaheuristic procedures, and also to investigate more advanced forms of tabu search. Yet our results with the version of TS we propose have proved comparable (using many fewer variables) on some examples than those previously obtained by researchers in this area, and on this basis we conclude that our work provides a useful contribution even though we may subsequently find ways to improve upon it - as in fact we hope to do.

Metaheuristic search methods such as genetic algorithms (Holland, 1975), Artificial Neural Networks (Freeman and Skapura, 1991), simulated annealing (Metropolis et al., 1953, Johnson et al., 1989), Tabu search (Glover, 1997), and others have been applied to machine learning and data mining methods such as decision trees and Bayesian Networks (Sreerama et al., 1994, Har-

wood and Scheines, 2002) with significant success in finding good solutions and accelerating convergence in the learning process. Recent applications include a Genetic Algorithm based approach to building accurate decision trees in the marketing domain (Fu et al., 2004), Neural Networks applied to Hybrid Intelligent Systems for Stock Market Analysis (Abraham et al., 2001) and hill-climbing heuristics to model a reinforcement learning algorithm for learning to control partially-observable Markov decision processes (Moll et al., 2000).

3. Tabu Search Enhanced Markov Blanket Algorithm

Our algorithm first generates an *initial Markov Blanket* DAG for the target variable. However, the initial MB may be highly suboptimal due to the application of repeated conditional independence tests(Spirtes et al., 2000) and propagation of errors in causal orientation (Bai et al., 2003). Therefore, Tabu Search is applied to improve the initial MB DAG. The algorithm stops after a fixed number of iterations or a fixed number of non-improved iterations. These steps are explained in the following two subsections. The detailed algorithm is presented as below.

TS/MB Algorithm

InitialMBsearch (Data D, Target T, Depth d, Significance α):

/*

$sepSet(v_i, v_j)$: a mapping of a set of nodes s.t. $(v_i \perp v_j \mid sepSet(v_i, v_j))$;
$adj(v_i)$: the set of adjacent nodes to node v_i in G;
A: an edge list not in the output $MBDAG$, but might be added;
$vertex(G)$: the set of vertexes in the graph G;
$edges(G)$: the set of edges in the graph G;
*/

Procedure **checkedges**[*1] (Vertex V, Graph G, Depth-of-search d, Array of Sets $sepSet$, set of edges $Forbidden$):
  For each $v_i \in vertex(G) \setminus V$;
    If $V - v_i \notin Forbidden$
    add $V - v_i$ to $edges(G)$;
    For each $depth = 0, ..., d$
      If ($adj(V)$ has at least $depth + 1$ vertexes in it)
        For each $v_i \in adj(V)$

If (v_i is independent of V conditional on S))
where $S \subset \{adj(V) \setminus v_i\} \& (|S| = depth)$
remove edge $V - v_i$ from $edges(G)$;
add edge $V - v_i$ to $Forbidden$;
$sepSet(v_i, V) = S$;

End procedure

(*1 Note that the graph G is modified by the procedure **checkedges**, and so it contains different adjecencies at different points in the algorithm. The graphs we consider in the algorithm have edges with two kinds of points: non-arrow head, " - ", or arrow head, ">". A "*" is a meta-symbol that represents either arrow head or arrow tail. Thus $A * - B$ represents either $A \leftarrow B$ or $A - B$. When an edge $A * - B$ is replaced by $A* \rightarrow B$, the end point with $*$ is left the same. Thus the instruction to replace $A * - B$ with $A* \rightarrow B$ says if the edge is $A - B$, replace it with $A \rightarrow B$; and if the edge is $A \leftarrow B$, replace it with $A \leftrightarrow B$.)

Initialize $vertex(G)$ to all variables in data set D;
Initialize $edges(G)$ to $\emptyset$*2;
Initialize $sepSet(v_i, v_j)$ to $Null$;
Initialize $Forbidden$ to $\emptyset$;

(*2 Note that if the final $sepSet(v_i, v_j)$ is $Null$, v_i and v_j are not $d - separated$ conditional on any subset of vertices, this is different than being $d - separated$ conditional on the empty set.)

/* Finding adjacency; T is the target variable. */
checkedges ($T, G, d, sepSet, Forbidden$);
For each V in $adj(T)$
checkedges ($V, G, d, sepSet, Forbidden$);
For each $V \in adj(adj(T)) \setminus adj(T)$
checkedges ($V, G, d, sepSet, Forbidden$);

/* Orient G using Orientation rules. */
For each triple of vertices (X, Y, Z)
If $X * - Y - *Z$ and X is not adjacent to Z , **and If** $Y \notin (sepSet(X, Z))$, then
orient as $X* \rightarrow Y \leftarrow *Z$;

Repeat
If $X \rightarrow Y - Z$ **and** X, Z are not adjacent **then** orient as $Y \rightarrow Z$;

If $(X \rightarrow Z \rightarrow Y)$, **and** $X * - Y$, **then** orient $X* \rightarrow Y$;

If $W \rightarrow Y \leftarrow Z$, **and** Z is not adjacent to W, **and** W, Y, Z are all adjacent to V, $W - V - Z$ do not collide at V, **then** orient $V * - Y$ as $V* \rightarrow Y$;

Until no more edges can be oriented.

/* Transform into a $MBDAG$: */

For any v_i in G s. t. $v_i \leftrightarrow T$ or $v_i - T$;

orient this edge as $T \rightarrow v_i$; put edge $T \leftarrow v_i$ into the edge list A;

For any w, v_i in G s. t. $w - v_i \rightarrow T \& w \notin \{T\} \cup adj(T)$ remove the edge $w - v_i$; put $w - v_i$ into the edge list A;

For any node v_i, $v_i \notin \{T\} \cup adj(T) \cup adj(adj(T))$

remove v_i and all the corresponding edges; record the corresponding edges in A;

Remove any undirected or bi-directed edges from G; put them into the edge list A

Remove any remaining edges among parents and among $adj(T)$ from G; put them into the edge list A

For any node v_i, $v_i \notin \{T\} \cup adj(T) \cup adj(adj(T))$

remove v_i and all the corresponding edges; record the corresponding edges in A;

/* At this point G is an MB DAG: $MBDag$ */

Return ($MBDag, A$).

TabuSearch (Data D, Target Y):

initialize $bestSolution := currentSolution := MBDag$;

$bestScore := currentScore := thescoreof MBDag$;

$tabuTenure = 7$ (in our experiments); $tabuList := \emptyset$

repeat until ($bestScore$ does not improve for k consecutive iterations)

form $candidateMoves$ for $currentSolution$

find $bestMove$ among $candidateMoves$ according to function *score*

update $currentSolution$ by applying $bestMove$

add $bestMove$ to $tabuList$

/* tabu moves will not be re-visited in the next k iterations.*/

if ($bestScore <$ **score** ($bestMove$))
update $bestSolution$ and $bestScore$ by applying $bestMove$
return $bestSolution$(a $MBDag$)

InitialMBsearch (D, T, d, α)
TabuSearch (D, Y)

End pseudo-code of TS/MB algorithm.

1^{st} Stage: Learning Initial Dependencies - *InitialMBsearch*

Suppose that there is a DAG G without hidden variables such that the population distribution p satisfies the Markov and Faithfulness conditions for G. In that case, there is a set of DAGs, $Equiv(G)$ that entail exactly the same set of conditional independence relations as G (in some cases $Equiv(G)$ contains only G for discrete distribution.) Any member of $Equiv(G)$ is as good a predictor of the target variable T as is G. The goal of the *InitialMBsearch* is to generate an initial MB DAG for T from the data (Ramsey et al., 2004) that is in $Equiv(G)$. The *InitialMBsearch* has four different phases.

1 Find the vertices adjacent to T, and the vertices adjacent to the vertices adjacent to T. After the first phase, pairs of vertices that are adjacent are connected by an undirected edge.

2 Apply the edge orientation rules that guarantee that: for each undirected edge $X - Y$, if it is oriented as $X \rightarrow Y$ in every DAG conditional independence equivalent to G, the edge is oriented as $X \rightarrow Y$.

3 Orient the remaining undirected edges to form a DAG.

4 Prune the vertices and edges that are not part of the MB DAG.

If there is a DAG G without hidden variables, such that the population distribution p satisfies the Markov and Faithfulness conditions for G, and as long as the results of conditional independence tests are correct, the correctness of the adjacencies in the DAG returned by *Phase 1* is justified by the following Theorem:

Theorem: If a distribution p satisfies the Markov condition for a DAG G, and is faithful to G, then 1) if X and Y are independent conditional on any subset S then there is no edge between X and Y in G; 2) if there is no edge between X and Y in G then X and Y are independent conditional on the parents of Y in G, or independent conditional on the parents of X in G.

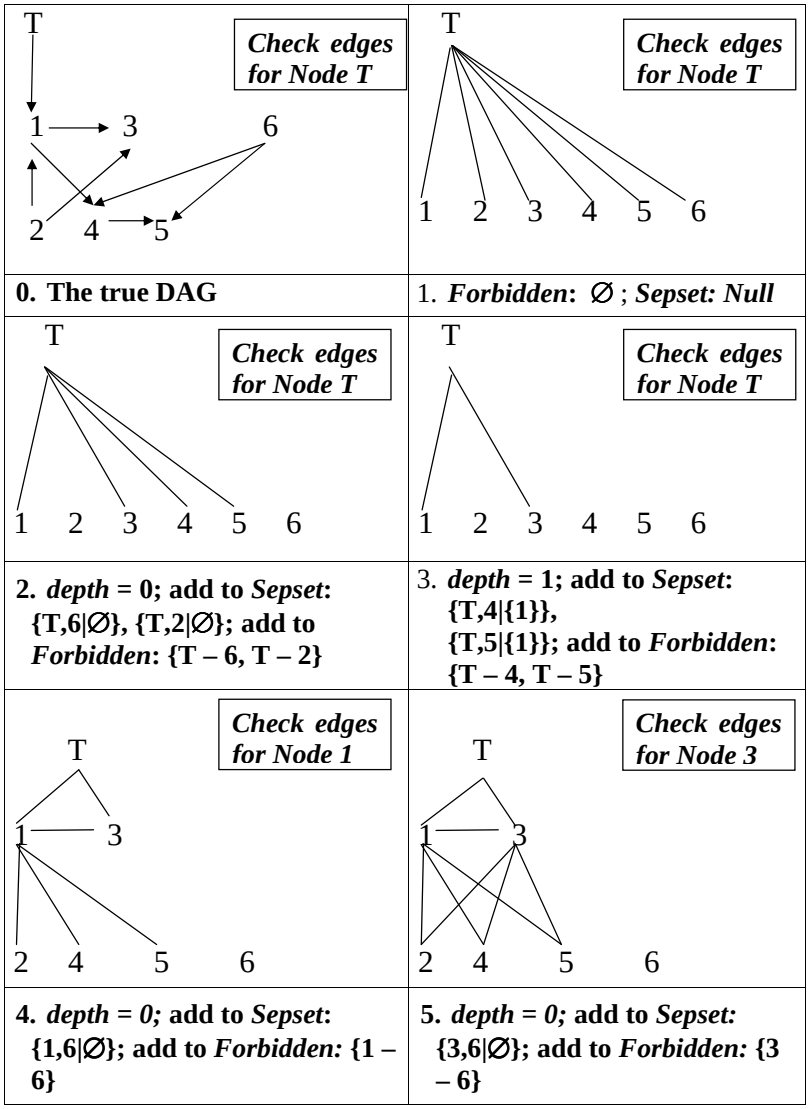


Figure 3. Steps of the initial solution creation (Steps 0-5)

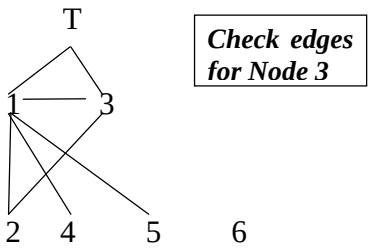
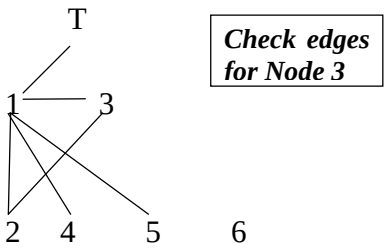
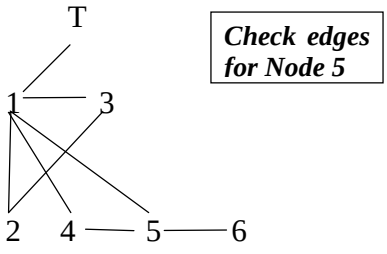
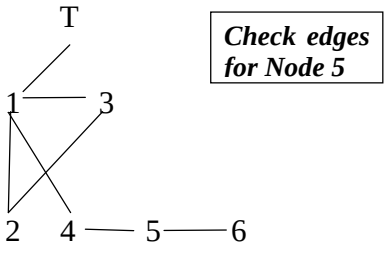
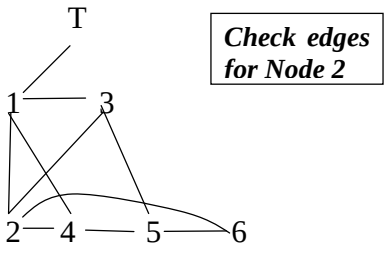
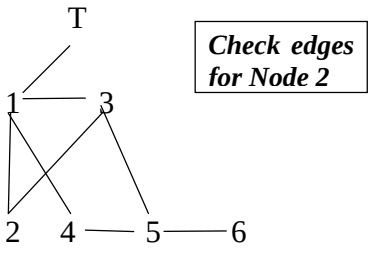
	
6. *depth* = 1; add to *Sepset*: {3,5|{1}}, {3,4|{1}}; add to Forbidden: {3 – 5}, {3 – 4}	**7. *depth* = 2; add to *Sepset*: {3,T|{1,2}}; add to *Forbidden* {3 – T}**
	
8. *depth* = 1; add to *Sepset*: {2,5|{1}}; add to *Forbidden*: {2 – 5}	**9. *depth* = 2; add to *Sepset*: {1,5|{4, 6}}; add to *Forbidden*: {1 – 5}**
	
10. the same operation for Node 2; add to *Sepset* {2,6|{∅}}	**11. *depth* = 1; add to *Sepset*: {2,4|{1}}; add to *Forbidden*: {2 –4}, {2 –6}**

Figure 4. Steps of the initial solution creation (Steps 6-11)

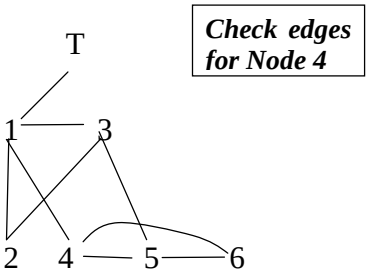
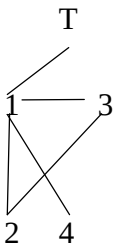
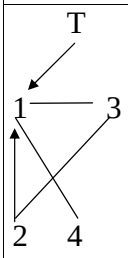
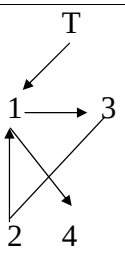
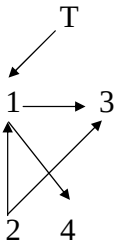
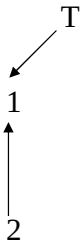
	
12. The *same* operation for Node 4. Note that there is no need to check edges to Node 6 because all the candidate edges for Node 6 have been checked before.	**13. Prune redundant edges**
	
14. Orient colliders	**15. Orient away from colliders**
	
16. Orient away from cycles	**17. The resulting Markov Blanket**

Figure 5. Steps of the initial solution creation (Steps 12-16)

Phase 1 consists basically of repeated calls to the procedure $Checkedges$. The procedure $Checkedges(V)$ returns a graph U (with undirected edges); the set of vertices adjacent to V in U may be a superset of the vertices adjacent to V in G. $Checkedges(V)$ starts by making V adjacent to each other vertex (except for those that have already been determined are not adjacent to V.) It performs a sequence of conditional independence tests, and removes the edge between V and X if it finds some subset S of the other vertices adjacent to V such that V and X are independent conditional on S. This edge removal is justified by condition 1) of the Theorem.

Even if V and X are adjacent in U, V and X might not be adjacent in G. This is because $Checkedges(V)$ checks whether V and X are independent conditional on the parents of V (because the parents of V are a subset of the vertices adjacent to V). However, it might be the case that V and X are independent conditional on the parents of X rather than the parents of V, and $Checkedges(V)$ did not check that possibility (if, for example, there is a parent of X that is not adjacent to V). However, a subsequent call of $Checkedges(X)$ will check whether V and X are independent conditional on the parents of X (which are a subset of the vertices adjacent to X.)

So *phase 1* first calls $Checkedges(T)$ to find a superset of the variables adjacent to the target T in G. For each vertex X adjacent to T, *phase 1* calls $Checkedges(X)$. At this point, the graph U has the correct adjacencies to T, and a superset of the vertices adjacent to the vertices adjacent to T in G. For each variable Y adjacent to a variable X adjacent to T, $Checkedges(Y)$ is called. After this stage, the set of vertices adjacent to T is correct in U, and the set of vertices adjacent to vertices adjacent to T is correct in U.

In *phase 2*, the algorithm orients some of the undirected edges. In some cases, every DAG in $Equiv(G)$ orients the edges in the same way, i.e. . The algorithm finds all of those orientations that are shared by every member of $Equiv(G)$. This part of the algorithm is the same as the PC algorithm (Spirtes et al., 2000).

In *phase 3*, the algorithm chooses an orientation for those undirected edges that are not oriented the same way in every member of $Equiv(G)$, and any bidirected edges. (If the assumptions under which the correctness of the algorithm has been proved are true, there will be no bidirected edges.) This part of the algorithm is a simplified heuristic - it does not guarantee that the orientations are the same as some member of $Equiv(G)$. This is because extensive experimentation has shown that the PC algorithm is much more reliable on adjacencies than it is on orientations, and the orientations after the *InitialMB-search* are simply used as one starting point for the subsequent Tabu Search.

After *phase 3*, it may be that some vertices that are adjacent to vertices adjacent to T are nevertheless not parents of children of T, and hence are not

part of the MB DAG. These vertices and the edges they are part of are then pruned from the DAG.

2^{nd} Stage: Tabu Search Optimization

Tabu Search (TS) is applied to improve the initial MB DAG. Our algorithm searches for solutions in the space of Markov Blankets DAGs; moves that result in cyclic graphs are not valid moves.

Briefly, four kinds of moves are allowed in the TS procedure: edge addition, edge deletion, edge reversal, and edge reversal with node pruning. At each stage, and for each allowed move, the corresponding MB DAG is computed, its conditional probability factored, its predictions scored, and the best move is then selected and applied. The best solution and best score at each step are tracked. The tabu list keeps a record of m previous moves, so that moves in the tabu list will not be repeated till their corresponding tabu tenure expires.

Markov Blanket Neighborhoods and Choice of Moves. Our algorithm uses the set of Markov Blanket DAGs as the set of possible states in the search space. The set of operators are the feasible moves, which transforms the current MB DAG to another MB DAG. Thus the neighborhood for any state is the set of *new* Markov Blankets DAGs that can be constructed via one feasible move. We call this a Markov Blanket Neighborhood. However, in order to make our search more efficient, I do not consider the MB DAGs in the neighborhood whose only differences from a current MB DAG are the edges among parents, because the conditional independence relations entailed by these edges have no effect on the discrete probability distribution of the target variable conditional on the other variables in the DAG. So we do not have moves such as adding or deleting an edge between parents of the target from a current MB DAG in our candidate move list for a current solution or state.

We allow the following kinds of moves, as illustrated in Figure 6:

- **edge addition** For example: from (b) to (c), by adding edge $X_2 \rightarrow Y$, X_2 becomes a parent of Y; the conditional probability of Y in the Markov factorization Changes from $p(Y|X_1)$ to $p(Y|X_1, X_2)$.

- **edge deletion** For example: from (a) to (b), by deleting edge $X_4 \rightarrow X_5$, X_4 is no longer a parent of X_5; the conditional probability of X_5 in the Markov factorization Changes from $p(X_5|X_3, X_4, Y)$ to $p(X_5|X_3, Y)$.

- **edge reversal** For example: from (c) to (d), by switching the direction of edge $X_1 \rightarrow Y$, X_1 changes from a parent of Y to a child of Y; the Markov factorization of the graph has a new item $p(Y|X_2) \cdot p(X_1|Y)$, and it replaces $p(Y|X_1, X_2)$.

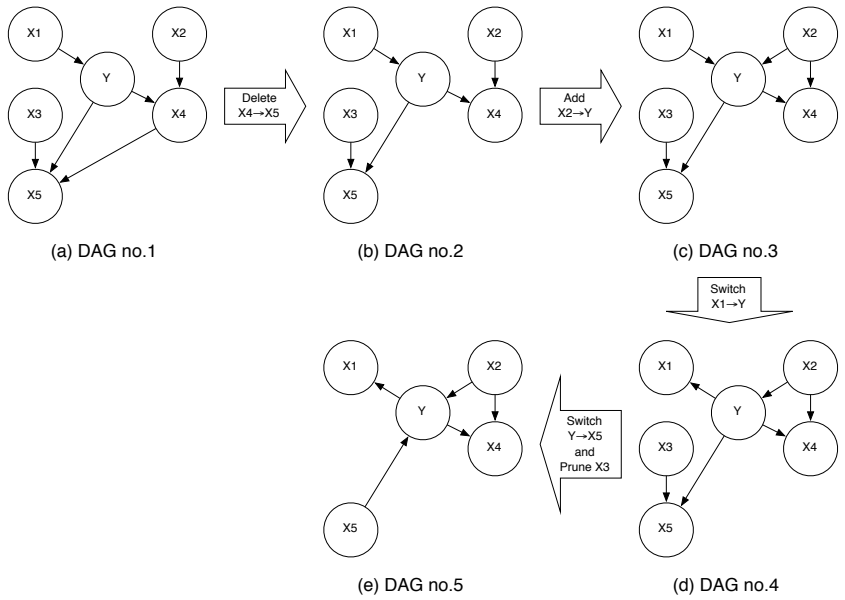


Figure 6. An example of moves in Tabu search enhanced Markov Blanket procedure

- **edge reversal with node pruning** For example: from (d) to (e), by switching the direction of edge $Y \rightarrow X_5$, X_5 becomes a parent of Y; X_3 becomes a parent of a parent(X_5) of Y. X_3 is longer in the Markov Blanket, so it is pruned away. The Markov factorization of the graph has a new item $p(Y|X_2, X_5)$, and it replaces $p(Y|X_2) \cdot p(X_5|X_3, Y)$.

Table 1 lists the corresponding factorizations of each move in Figure 6.

Table 1. The Markov Factorizations for Moves in Figure 6

DAGs	Score Computation
DAG no.1	$p(Y\|X_1, ..., X_5) = C_1 \cdot p(Y\|X_1) \cdot p(X_4\|X_2, Y) \cdot p(X_5\|X_3, X_4, Y)$
DAG no.2	$p(Y\|X_1, ..., X_5) = C_2 \cdot p(Y\|X_1) \cdot p(X_4\|X_2, Y) \cdot p(X_5\|X_3, Y)$
DAG no.3	$p(Y\|X_1, ..., X_5) = C_3 \cdot p(Y\|X_1, X_2) \cdot p(X_4\|X_2, Y) \cdot p(X_5\|X_3, Y)$
DAG no.4	$p(Y\|X_1, ..., X_5) = C_4 \cdot p(Y\|X_2) \cdot p(X_1\|Y) \cdot p(X_4\|X_2, Y) \cdot p(X_5\|X_3, Y)$
DAG no.5	$p(Y\|X_1, ..., X_5) = C_5 \cdot p(Y\|X_2, X_5) \cdot p(X_1\|Y) \cdot p(X_4\|X_2, Y)$

Moves that yield cyclic graphs are not valid moves. At each stage, for each allowed move, the resulting Markov Blanket is constructed, the condi-

tional probability for the target node is factored and computed, its prediction is scored, and the current Markov Blanket is modified with the best move.

Tabu List and Tabu Tenure. In our implementation, the tabu list keeps a record of *m* previous moves, or more precisely, of the attributes of the current solution that are changed by these moves. By reference to this record, new solutions are classified tabu if they contain attributes of previous solutions encountered within the *m* move horizon. A move is tabu if it leads to a tabu solution. [2] The value of *m*, called the tabu tenure, can vary depending on the type of strategic design applied. We use a simple design that permits *m* to vary for different types of moves but, once assigned, remains constant throughout the search. We also employ aspiration criteria to permit moves that are *good enough* to be selected in spite of being classified tabu.

The magnitude of Tabu tenure can vary depending on the complexity of the MB DAGs in different problems. When the dependency structure of the Markov Blanket is very dense, the number of neighborhood states that need to be considered can grow exponentially, and a larger Tabu tenure is preferred. Implementations of simple versions of TS based on Tabu tenures between 7 and 12 have been found to work well in several settings where tabu restrictions rule out a non-trivial portion of the otherwise available moves (Glover, 1997). Our setting appears to be one of this character, and in our experiments, we use a static Tabu tenure of 7 because the structure of the MB DAGs is not complex. Considering the computational cost of each type of move, it is reasonable to assign larger value of Tabu tenure to moves that are computationally more expensive. Such moves include edge reversals that involve pruning nodes that are no longer present in the resulting MB DAG, or edge reversals that result in significant changes in the parent-child relations in the resulting MB DAG. It is possible to optimize these parameters in future research and to replace the use of a static tenure by a dynamic tenure.

Intensification and Diversification. We alternate between intensification and diversification in the TS/MB procedure. The intensification process focuses on convergence to the best Markov Blanket DAG in a local Markov Blanket neighborhood. The diversification process attempts to explore MB structures that are far from the current neighborhood. The distance of two Markov Blankets can be roughly understood as the difference in the MB structures and the resulting Markov factorizations. In our experiments, we seed different starting solutions by altering the significance level of the independence tests and by altering the edge orientations in the generated pattern at the end of the InitialMBsearch procedure. By doing this, we generate starting solutions with a variety of independence structures and complexity.

The result of local search is a local optimum. One typical local search is Best First Search (BFS). Best First Search always chooses the next node or state to be that with the best score according to the evaluation function. It is exhaustive because it tries all possible paths. Tabu search can navigate through local optima by accepting non-improving moves and pushes the solution toward the global optimum with the guidance of dynamic memory. It may or may not be exhaustive so it works on both finite and infinite search space. The following is an illustration of *Tabusearch* procedure. It is also a step-by-step example illustrating how Tabu search gets out of local optima and pushes the solution toward the global optimum. It shows why Tabu search works and works better than local search such as Best First Search (BFS) in this domain.

Suppose the output of *InitialMBsearch* (a) and the true graph (b) are as in Figure 7. Then the DAG transformation procedure transforms the output into a DAG, as shown in Figure 7(c), which is the initial MB DAG for Tabu search procedure.

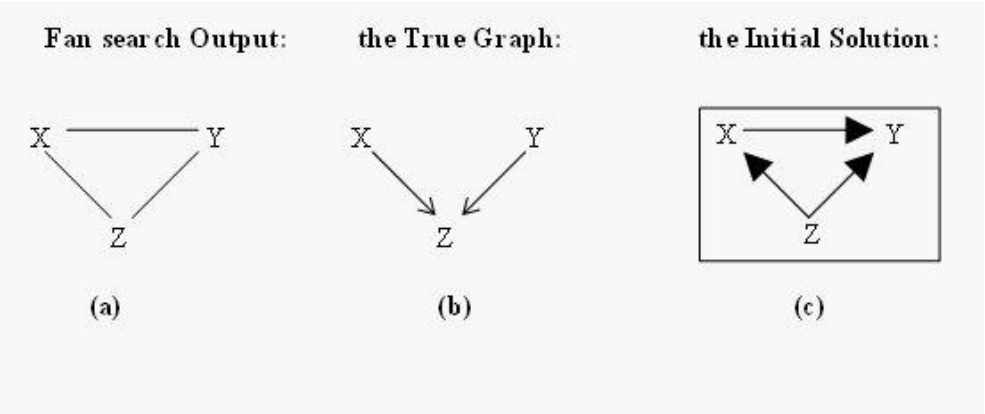


Figure 7. The *InitialMBsearch* output and the true graph

As shown in Figure 8, the current solution and best solution are initialized to this MB DAG at the beginning of Tabu search procedure.

By comparing the initial solution with the true graph:

- *Observation 1:* Any single valid edge reversal would not improve the independence structure because the resulting structure is Markov equivalent to the previous one;
- *Observation 2:* Any single edge deletion will result in the decrease of the score because it throws away the relevant information entailed by the initial solution, and the resulting structure will have a different Markov factorizations than the true graph;

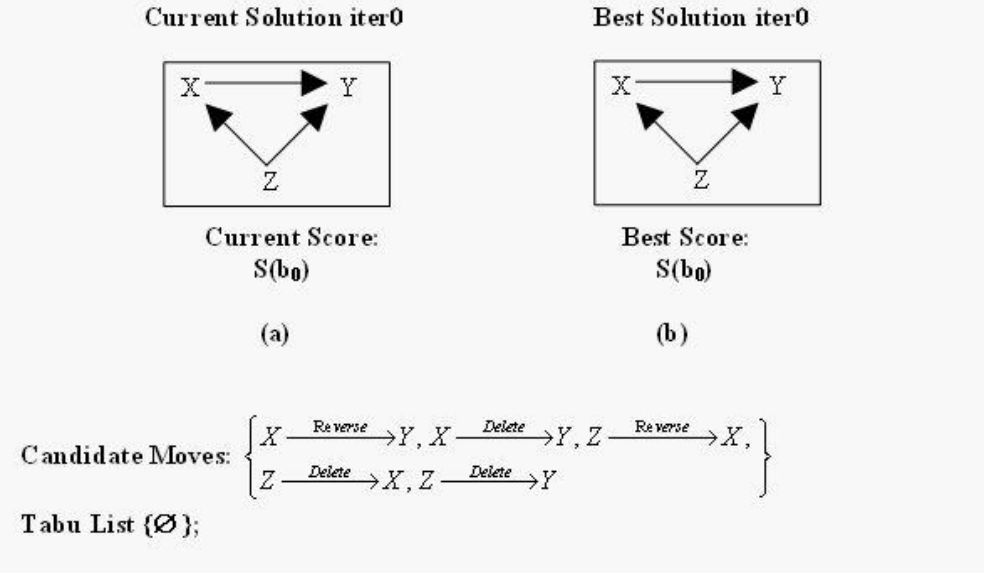


Figure 8. Tabu search initiation

- *Observation 3:* Any edge addition or edge reversal with node pruning does not apply here.

Best First Search (BFS) will stop right here, which we call a *local optimum*. Tabu search, unlike BFS, continues the search by keeping an adaptive memory of the best solution, current solution, current neighborhood, and current Tabu list. The Tabu procedure accepts non-improving solutions, picks up the best available move, and moves on. In this example, suppose by comparing the Markov factorization of the output and the true graph, the best available move is most likely to be $X \xrightarrow{Delete} Y$ (The best move is actually chosen by calculating the score of each candidate move based on the dependence structure and magnitude of the parameters of the variable). In our procedure, we set the tabu tenure as 7, constraining the re-entrance of a tabu move till after the seventh iteration.

Suppose in the 1st iteration, TS chooses to delete $X \rightarrow Y$. After the first iteration, the state information is as in Figure 9. The best solution remains the same, as does the best score. The current solution moves to its best neighbor, where the score is lower. Candidate moves get updated and the move $X \xrightarrow{Delete} Y$ is entered into the Tabu list.

In the 2nd iteration, TS chooses to reverse edge $Z \rightarrow X$. After the second iteration, the state information is as in Figure 10. The best solution still remains the same, as does the best score. The current solution moves to its best

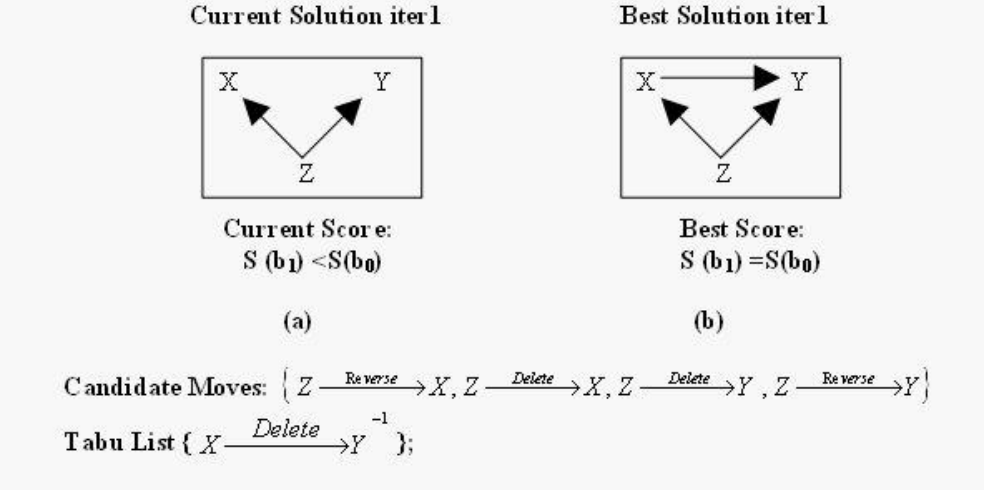


Figure 9. Tabu search iteration 1

neighbor, whereas the score remains the same. Candidate moves get updated and the move $Z \xrightarrow{Reverse} X$ is entered into the Tabu list.

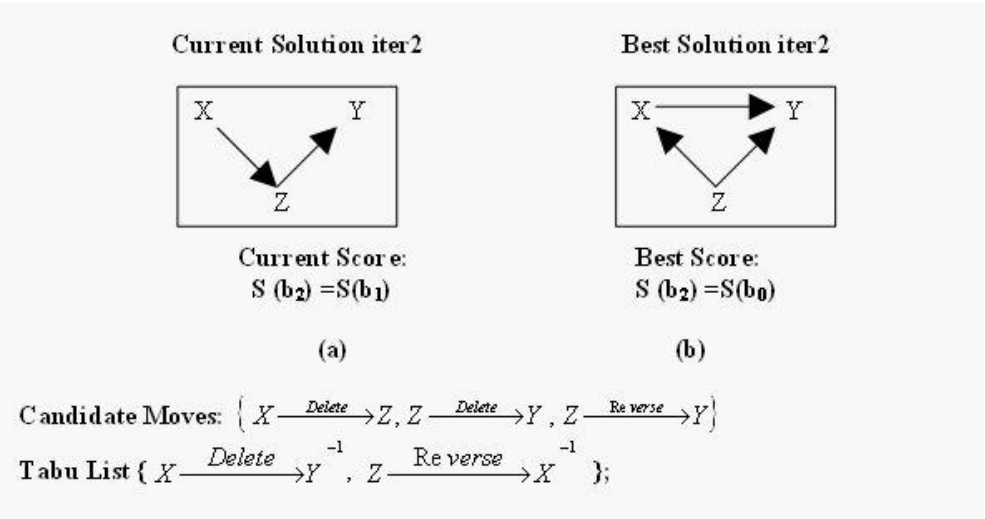


Figure 10. Tabu search iteration 2

In the 3rd iteration, TS chooses to reverse edge $Z \rightarrow Y$. After the third iteration, the state information is as in Figure 11. The procedure finds the true graph. Both the best solution and the best score get updated. The current solution moved to its best neighbor, which is the true graph. The score is

improved. Candidate moves get updated and the move $Z \xrightarrow{Reverse} X$ entered the Tabu list.

The procedure stops after a fixed number of iterations. The true graph is found to be the best solution.

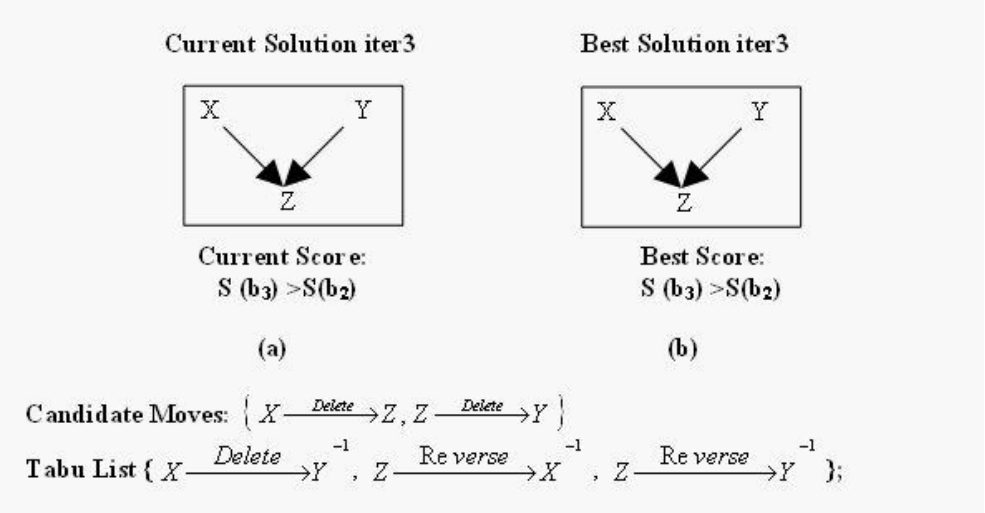


Figure 11. Tabu search iteration 3

Classification

At the end of the Tabu search, the TS/MB procedure returns an improved MB DAG. We use this MB DAG to do classification for the target variable. The classification is done in the following several steps. Suppose we have an MB DAG $MB(Y)$ (Figure 11):

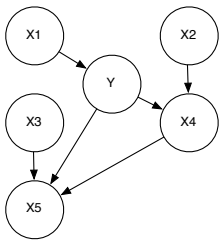


Figure 12. The Tabu search enhanced Markov Blanket for $Y : MB(Y)$

First, we factorize the conditional probability of Y given all the other variables as product of conditional probabilities of each variable given its parents:

$$p(Y|X_1,...,X_6) = C' \cdot p(Y|X_1) \cdot p(X_4|X_2,Y) \cdot p(X_5|X_3,X_4,Y) \ , \quad (6)$$

where C' is a normalizing constant independent of Y.

In order to estimate $p(Y = y_i|X_1 = x_1, X_2 = x_2, ...)$, we first estimate each factor in equation 6, and multiply the estimates. The estimate of a factor $p(A = a|B = b)$ is the number of data points in the training set in which $\mathbf{A} = \mathbf{a}$ & $\mathbf{B} = \mathbf{b}$, divided by the number of data points in the training set for which $\mathbf{B} = \mathbf{b}$. We assign the category with the highest conditional probability in the training set to this testing data point. For example, suppose we have a testing point $(x_1, x_2, x_3, x_4, x_5, x_6)$. By the definition of a Markov blanket, X_6 is not in the Markov blanket, i. e., Y (Y has k categories: $y_1, ..., y_k$) is independent of X_6 conditional on X_1, X_2, X_3, X_4 and X_5. The classification procedure is illustrated in Table 2:

Table 2. Calculation of $p(y|x_1)$

number of $(y_1, x_1) =$	number of $(y_2, x_1) =$	...	number of $(y_k, x_1) =$
n_1	n_2	...	n_k
$p(y_1\|x_1) =$	$p(y_2\|x_1) =$	...	$p(y_k\|x_1) =$
$n_1/(n_1 + n_2 + ... + n_k)$	$n_2/(n_1 + n_2 + ... + n_k)$	...	$n_k/(n_1 + n_2 + ... + n_k)$

By doing this calculation on every factor in the factorization and repeating this for each category of Y, we get the estimates for all the categories of Y as illustrated in Table 3:

Table 3. Classification of y given $(x_1, x_2, x_3, x_4, x_5, x_6)$

Estimate of $p(y\|\hat{x_1}, ..., x_6)$	$\hat{y}$
$p(y_1\|x_1, ..., x_6) = C' \cdot p(y_1\|x_1) \cdot p(x_4\|x_2, y_1) \cdot p(x_5\|x_3, x_4, y_1) = p_1$	
$p(y_2\|x_1, ..., x_6) = C' \cdot p(y_2\|x_1) \cdot p(x_4\|x_2, y_2) \cdot p(x_5\|x_3, x_4, y_2) = p_2$	
...	
$p(y_k\|x_1, ..., x_6) = C' \cdot p(y_k\|x_1) \cdot p(x_4\|x_2, y_k) \cdot p(x_5\|x_3, x_4, y_k) = p_k$	$\arg_{y_i} \max_{i \in 1,...k}\{p_i\}$

Suppose that one of the factors in the factorized distribution is $p(X|X_a = x_a, X_b = x_b, , X_i = x_i)$, and there are some training set data points with values $(x_a, x_b, , x_i)$. In that case, the preceding procedure gives a maximum

likelihood estimate of $p(X|X_a, , X_i)$. However, there may be cases in which no training data point has values $(x_a, x_b, , x_i)$ in which case there is no unique maximum likelihood estimate. Then we use a heuristic similar to nearest neighbor to estimate $p(X|X_a = x_a, , X_i = x_i)$. However, because we still use the factorization of the distribution to classify the target variable, the results in general are not the same as simply employing nearest neighbor to classify the target. Suppose for example that we the algorithm is estimating $p(y = 1|x_1, x_2, x_3, x_4, x_5, x_6)$, one of the factors in the factorization is $p(x_5|x_3, x_4, y = 1)$, and no data point in the training set contains $(x_3, x_4, y = 1)$. In that case, the algorithm determines if there any data points in the training set that contains (x_3, x_4), $(x_3, y = 1)$, or $(x_4, y = 1)$. If there are members of the training set that contain $(x_4, y = 1)$, and other members that contain $(x_3, y = 1)$, then the estimate of $p(x_5|x_3, x_4, y = 1)$ is set to the average of the relative frequency of x_5 among the training set data points with $(x_4, y = 1)$ and the relative frequency of x_5 among the training set data points with $(x_3, y = 1)$. If there are no data points in the training set that match any subset of $(x_3, x_4, y = 1)$ that leaves out a single variable value, then the algorithm checks whether there are any training set data points that leave out two variable values (ie. (x_3), (x_4), or $(y = 1)$), and so on until some matching training set data points are found.

This heuristic for estimating the conditional probability of the target variable has the net effect of penalizing adding more parents to the target variable less than simply guessing when there is no training set data matching the parent set of one of the factors in the factorization. This may partially explain why in actual practice the parent sets of the target variable in the graph that was output were generally fairly large.

In practice, the size of the suppression sets very rarely goes beyond one. The explanation is that although the training set does not contain all possible combinations of the predictor variables, some combinations of predictor values are unlikely to occur in both training and test sets. For example in the movie review case which I will describe later, $p(rating = 1|Worst = 1, Boring = 1, Lame = 1, Perfect = 1)$ cannot be estimated but it is very unlikely that this combination of values would occur in the test set. So the actual chance of finding the matched case from the training set is larger than that assuming all the configurations of the selected features are equally probable. There could be other heuristic methods for handling classifying data with no matched training cases.

By orienting an edge out of the target node and making the adjacent node child, the number of parameters to be estimated is reduced by a factor of 2, if the predictor is binary. The more children in the graph, the fewer parameters are there to estimate, and the more efficient the algorithms are. However, in the experimental results we got, the learned MB Dags have more parents than

children in the output and sometimes have no parents of the children. One possible explanation to this is that in order to be able to identify those edges, either the edge dependence has to be very strong or there has to be a large amount of data. The data we are working on is high in dimension and low in the number of the sample points. Thus it is very difficult to identify these edges.

Sample size affects the running time of our algorithm. It may take a long time to run on data sets with large number of samples. This is because the number of evidences needed to check is increased very time an independence test is performed. These effects occurred in our experiments.

4. Experimental Design

Our algorithm is evaluated on three real world data sets from different domains, such as health care, internet marketing and text mining. The detailed results and analyses are presented as case studies in Section 5, 6 and 7. The results are compared against several state-of-the-art classifiers. The classifiers compared are the naive Bayes classifier, the support vector machine (SVM), the voted Perceptron, the maximum entropy method, k-nearest neighbor, and logistic regression.

Regarding the classifiers, support vector machines are learning machines that can perform binary classification (pattern recognition) and real valued function approximation (regression estimation) tasks. Support Vector Machines non-linearly map their n-dimensional input space into a high dimensional feature space. In this high dimensional feature space a linear classifier is constructed. Naive Bayes classification is a simple probabilistic classification method. The underlying probability model assumes features independent conditional on the target, which is often an unrealistic assumption. The perceptron algorithm is a type of artificial neural network classification algorithm. The voted perceptron method is one or more layers of the Threshold Logic Units where the inputs are fed directly to the outputs via a series of weights. Maximum Entropy [3] functions by maximizing an entropy function. k-nearest neighbor is an instance based learning method which selects "k nearest" examples in the training set according to a distance function, and predict the most common class among its K-nearest neighbors as the label of the dependant variable [4]. Complete definitions of these classifiers can be found in the book *Machine Learning* by Mitchell (Mitchell, 1997).

Since several parameters are central to the performance of the algorithm, an appropriate experimental design has an important role in the evaluation study.

Design of Experimental Parameters

The parameters in our experiments are: data-splits, scoring criteria, starting solution structure, the depth of conditional independence search (*d*), and significance level (α).

Data-splits. We split the data in two ways: 90 percent for training/10 percent for testing and 80 percent for training/20 percent for testing. In our experiments, each configuration is cross validated. For example, in a 5-fold cross validation scheme (80 percent for training/20 percent for testing), the data set is divided into 5 subsets. In each run, one of the 5 subsets is used as the test set and the other 4 subsets are assembled to form a training set. Then the average error across all 5 trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set 4 times. The mean and the variance of the estimated error are reduced as the number of folds increases. In that sense, 5-fold cross validation yields more conservative estimates than 10-fold cross validation (90 percent for training/10 percent for testing). We use a nested, stratified cross-validation scheme (Weiss and Kulikowski, 1991).

We found the *dominant configuration* of the parameters on the training data and estimated the performance on the testing data. For example, in a 5-fold cross-validation schema, in order to find this configuration, within each fold i, we further split the training data in two (TR_{i1} and TR_{i2}), trained the MB classifier on TR_{i1} for each parameter configuration, and tested the performance on TR_{i2}. The configuration that led to the best MB, in terms of accuracy on TR_{i2} across all folds $i = 1, ..., 5$, was chosen as the best configuration. The outer layer of cross-validation estimates the performance of the best Markov Blanket classifier on the testing data.

Scoring criteria. The applied scoring criteria are AUC and prediction accuracy. Prediction accuracy is widely used in the Machine Learning community for comparison of classification effectiveness. However, accuracy estimation is not the most appropriate metric when cost of misclassification or class distributions are not specified precisely (Provost et al., 1998). For example, wrong prediction in the diagnosis and treatment of a seriously ill patient has different consequences than incorrect prediction of the patronage of an online consumer. The quality metric AUC, the area under the Receiver Operator Characteristic (ROC) curve, takes into account the costs of the different kinds of misclassification (Hanley and McNeil, 1983). An ROC curve is a plot of true-positive rate and false positive rate in binary classification problems as a function of the variation in the classification threshold. This metric has gained popularity among statisticians for evaluating diagnosis tests, and has also been adopted by the machine learning community for general binary classification evalua-

tion. ROC curves are similar to the precision/recall curves used in information retrieval (Lewis, 1991) as well as lift curves used in marketing communities (Piatetsky-Shapiro and Steingold, 2000). AUC ranges from 0 to 100 percent. The higher the AUC is, the better the quality of the classifier. We test both AUC and prediction accuracy as scoring criteria to score each move. For example, in experiments where AUC is used as the scoring criterion, the procedure calculates AUC for every neighborhood move and identifies the move with the highest AUC as the best move, and similarly with prediction accuracy.

Starting solution structure. *InitialMBsearch* is only able to learn the true graph in the condition of large sample limit with unlimited depth of search and no latent variables. In practice, this is never the case. The orientation rules of *InitialMBsearch* is not reliable with finite data and and high dimensions. Heuristic search starting from there comes into play naturally. Tabu Search picks one MB DAG as the starting solution. Variations in the orientation of the edges result in different starting solutions. We have tested two specific alternatives. In Structure I, all the undirected edges adjacent to the target variable Y are oriented into *Y*, i.e., all the nodes that are connected to *Y* by undirected edges are treated as the parents of Y. In Structure II, all but one of the undirected edges adjacent to the target variable *Y* are oriented out of *Y*, i.e., all but one of the nodes that are connected to *Y* by undirected edges are treated as the children of Y; that one node is treated as a parent of Y. By orienting the edges differently, we create two different types of Markov Blanket structures for the starting solution, as shown in Figures 13 and 14.

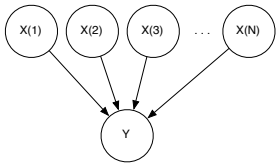


Figure 13. Starting solution I

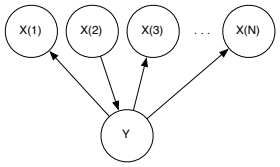


Figure 14. Starting solution II

Depth of the conditional independence search (*d*). The depth of conditional independence search (*d*) and the significance level (α) are usually exogenous variables that the user has to provide. The depth of search specifies the maximum size of the set conditioned on in the conditional independence tests.

Significance level (α). The alpha level is the threshold for the statistical independence tests. The smaller alpha is, the stricter the tests are and the stronger he dependence between two nodes has to be in order to retain the edge. We do

not adjust for multiple tests, and treat it as a search parameter. Tabu search enhancement in the second stage will search iteratively to improve the predictive structure of the learned Markov blanket

There are thus a total of 96 configurations of parameter combinations (Table 4).

Table 4. Experimental Parameter Configurations

Parameters	Data-splits (train/test)	Scoring Criteria	Starting Solution	Depth of Search	Alpha
Configurations	90%/10%	AUC, Accuracy	Type I, Type II	1,2,3	0.001,0.005
	80%/20%				0.01,0.05

Evaluation Criteria

To evaluate the classification results of the generated models, we use both the AUC and prediction accuracy on the testing set. These results are shown in Table 6 and Table 7. To compare our algorithm with the commonly used classifiers, we use AUC and prediction accuracy on the testing set, as well as the size of reduction in the set of variables. The size reduction was evaluated based on the fraction of variables in the resulting models. All metrics (variable size reduction, AUC, and accuracy) were averaged over cross-validation splits. The comparative results are shown in the case study sections(Section 6, 7 and 8).

5. A Demonstration of Experiments and Analysis

In this section, we demonstrate the whole experimental process and detailed analysis on Prostate cancer (PCA) data set (Adam et al., 2002).

PCA Data

PCA data set has been widely used in medical informatics research (Tsamardinos et al., 2003). The task is to diagnose prostate cancer from analysis of mass spectrometry signal peaks obtained from human sera. 326 serum samples from 167 PCA patients, 77 patients with benign prostate hyperplasia (BPH), and 82 age-matched unaffected healthy men are used for learning and prediction. Peak detection was performed using Ciphergen SELDI software versions 3.0 and 3.0.5. Powerful peaks in discriminating normal versus PCA, normal versus BPH, and BPH versus PCA were selected as features for classification. After

the clustering and peak alignment process, 779 peaks were identified. Table 5 summarizes the characteristics of PCA data set.

Table 5. Characteristics of the PCA Data Set

Task	Variables	#Samples	Variable Types	Target Variable Type
Prostate Cancer Diagnosis	779	326	discretized [5]	binary (cancer/normal)

Experimental Results and Analysis

Table 6 and Table 7 present the classification accuracy and AUC for different combinations of the experimental parameters (Comparisions with other methods are shown in Section 7). The results are averaged over cross-validation runs. The number in parentheses is the standard error. We did not list the parameter "depth of search" here because the variation of the computational result is very small across depth configurations in all the experiments. The results presented in Table 6 and 7 are the outcomes with depth 3.

Table 6. Classification Accuracy on the Testing Set (%)

Data-splits	Scoring	Starting	Alpha(α)			
(train/test)	Criteria	Solution	(α=0.001)	(α=0.005)	(α= 0.01)	(α=0.05)
293/33	AUC	I	83.8 (1.3)	87.1 (1.2)	83.8 (1.5)	90.3 (1.3)
		II	83.8 (1.4)	87.1 (1.0)	74.1 (3.5)	90.3 (1.0)
	Accuracy	I	83.8 (2.2)	87.1 (1.0)	83.8 (2.0)	90.3 (1.7)
		II	83.8 (2.3)	87.1 (1.1)	74.1 (4.9)	83.8 (3.5)
259/67	AUC	I	90.3 (1.2)	88.7 (1.2)	83.8 (2.2)	93.5 (1.0)
		II	61.6 (28.3)	88.7 (1.2)	83.8 (2.2)	91.9 (1.2)
	Accuracy	I	90.3 (1.2)	88.7 (1.3)	90.3 (1.0)	93.5 (1.0)
		II	90.3 (1.2)	88.7 (1.3)	90.3 (1.0)	91.9 (1.9)

As shown in Table 6 and Table 7, the optimal parameter configurations for best prediction accuracy (259/67, AUC or Accuracy, Structure I, 0.05) is different from the optimal parameter configurations for best AUC (293/33, AUC or Accuracy, Structure I, 0.05). In real world applications, users can choose the evaluation criterion they deem appropriate for the application. As shown in Figures 15 and 16, the resulting best fitting MB DAGs are also different.

In our experiments, the scoring criterion used in Tabu Search does not impact the classification performance, both in terms of prediction accuracy and AUC. Setting the alpha value at 0.05 generates the best result for both accu-

Table 7. Classification AUC on the Testing Set (%)

Data-splits	Scoring	Starting	Alpha(α)			
(train/test)	Criteria	Solution	(α=0.001)	(α=0.005)	(α= 0.01)	(α=0.05)
293/33	AUC	I	97.1 (1.0)	95.0 (2.1)	94.5 (1.6)	98.3 (0.8)
		II	97.1 (1.0)	94.1 (1.1)	94. (1.0)	96.2 (1.2)
	Accuracy	I	97.1 (1.1)	95.0 (1.9)	94.5 (1.3)	98.3 (0.8)
		II	97.1 (0.9)	95.0 (2.1)	94.1 (1.9)	83.7 (1.8)
259/67	AUC	I	96.9 (1.2)	96.0 (0.9)	94.0 (0.8)	96.3 (1.1)
		II	69.3 (24.0)	96.0 (1.0)	94.0 (1.1)	96.5 (1.9)
	Accuracy	I	96.9 (1.2)	96.0 (0.9)	95.8 (0.9)	95.0 (0.8)
		II	96.9 (1.2)	96.0 (1.3)	97.6 (1.2)	96.5 (1.2)

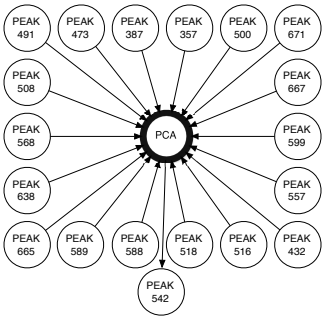


Figure 15. The best fitting MB DAG by AUC

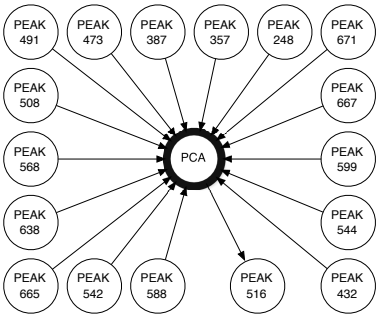


Figure 16. The best fitting MB DAG by accuracy

racy and AUC. The reason may be that a larger alpha value imposes fewer constraints on accepting dependence in the independence tests. This allows the algorithm to generate more complex MB structures and test more edges. Furthermore, the Structure I initial solution seems to be a better configuration under both evaluation criteria. All the directed edges are robust over almost all cross validation runs, with very small variation. However, more extensive and systematic experiments on larger data sets are necessary to explore these relationships further.

6. Application in Text Mining Field: Consumer Sentiments Case Study

In this section, we study a case from text mining domain: sentiment extraction from text documents. Traditionally, researchers have used surveys to collect a limited amount of data in a structured form for their analyses. In recent years, the advent of the Internet, and the widespread use of advanced in-

formation technologies in general, have resulted in a surge of information that is freely available on-line in an *unstructured format*. For example, many discussion groups and review sites exist where people post their opinions about a product. The automatic understanding of *sentiments* expressed within the texts of such posts could lead to a number of new applications in the fields of marketing and information retrieval.

Researchers have been investigating the problem of automatic text categorization for the past two decades. Satisfactory solutions have been found for the cases of topic categorization and of authorship attribution; briefly, topics are captured by sets of keywords (Mitchell, 1997), whereas authors are identified by their choices about the use of non-contextual, high-frequency words (Mosteller and Wallace, 1964; Mosteller and Wallace, 1984; Airoldi et al., 2004). Pang et al. (Pang et al., 2002) showed that such solutions, or extensions of them, yield cross-validated accuracies and areas under the curve (AUC) in the low 80%s when ported to sentiment extraction.

Movie Reviews Data

We tested our method on the data set used in Pang et al. (Pang et al., 2002). This data set contains approximately 29,000 posts to the rec.arts.movies.reviews newsgroup archived at the Internet Movie Database (IMDb). The original posts are available in the form of HTML pages. Some pre-processing was performed to produce the version of the data we used (Pang et al., 2002). Specifically, only reviews where authors' ratings were expressed explicitly (either by stars or by numerical values) were selected. Then explicit ratings were removed and converted into one of three categories: positive, negative, or neutral. Finally, 700 positive reviews and 700 negative reviews, which the authors of the corpus judged to be more extreme, were selected for our study. Various versions of the data are available on-line (http://www.cs.cornell.edu/people/pabo/movie-review-data/, 2002).

Feature Definition

In our study, we used words as features, where *words* are strings of letters enclosed by non-letters to the left and to the right. Note that our definition excludes punctuation sign even though exclamation signs and question marks may be helpful for our task. Intuitively the task of sentiment extraction is a hybrid task between authorship attribution and topic categorization; we look for frequent words, possibly not related to the context, that help express lexical patterns, as well as low frequency words which may be specific to a few review styles, but very indicative of an opinion. We considered all the words that appeared in more than 8 documents as our input features, whereas words with lower counts were discarded since they appear too rarely to be helpful in the

classification of many reviews. We were left with a total number of 7,716 words, as input features. In our experiments, we represented each document as a vector, $X := [X_1, ..., X_{7716}]$, of the size of the initial vocabulary, where each X_i is a binary random variable that takes the value of 1 if the i^{th} word in the vocabulary is present in the document and the value of 0 otherwise. Table 8 summarizes the characteristics of Movie Review data set.

Table 8. Characteristics of the Movie Review Data Set

Task	Variables	#Samples	Variable Types	Target Variable Type
Reviewer opinion classification	7,716	1,400	binary	binary (positive/negative)

Results and Analysis

We compare the our results first with the MB classifier, and then with the four widely used classifiers in text mining field: a naive Bayes classifier based on the multivariate Bernoulli distribution, discussed in Nigam et al. (Nigam et al., 2000), a support vector machine (SVM) classifier, discussed by Joachims (Joachims, 2001), an implementation of the voted Perceptron, discussed in Freund and Schapire (Freund and Schapire, 1999), and a maximum entropy conditional random field learner, introduced by Lafferty et al. (Lafferty et al., 2001).

Table 9 compares the TS/MB with the performances of the other classifiers using the *whole feature set* as input. As shown in the first two rows of Table 9, although the MB procedure itself can identify a discriminating subset of predictors, the MB procedure coupled with TS improves both AUC and accuracy. In this data set, MB procedure selects 18 out of 7716 features with simpler Markov Blanket graphical structure. TS/MB procedure picks up 3 more features and constructs more complex Markov Blanket graphical structures. This suggests that TS/MB is able to fine tune the initial Markov Blanket and find better independence structures with produce better predictions, than single-stage MB procedure. The comparative classification results of TS/MB against the other four methods are shown in table 9. The TS/MB procedure selects 21 relevant words out of 7,716 words in the vocabulary. The feature reduction ratio is 99.71%; the cross-validated AUC based on the 21 words and their dependencies is 77.26%, which is comparable with the other four methods; the corresponding cross-validated accuracy is 69.12%, which is slight lower than the average of the other four methods. This is understandable, because the TS/MB procedure is designed to find a parsimonious set of vocabulary

along with their conditional dependance relationships. Its contribution lies in providing insights to the sentiment understanding while achieving comparable prediction results. It has important implications in both the computational linguistic field and the text mining field.

Table 9. Sentiment Extraction Case Average Performance - Comparison I

Method	AUC (%)	Accuracy (%)	#Original Features	# Selected Features	Size Reduction
MB	71.24	65.00	7,716	18	99.76%
TS/MB	77.26	69.12	7,716	21	99.71%
Naive Bayes	82.61	66.22	7,716	7,716	0%
SVM + TFIDF	81.32	84.07	7,716	7,716	0%
Voted perceptron	77.09	70.00	7,716	7,716	0%
Max. entropy	75.79	79.43	7,716	7,716	0%

We notice that the TS/MB classifier is able to automatically identify a very discriminating subset of features (or words) that are relevant to the target variable (Y, the label of the review). Specifically, the selected features are those that form the Markov Blanket for Y. One might be interested to see how well these benchmark methods will do using the same number of features. In order to do this comparison, we selected 21 features with the highest information gain score [6], and used these as the input for the other four classifiers.

Table 10 compares the performance of the TS/MB with others classifiers using the *same number of features* selected using information gain criterion. We notice that feature selection using information gain criterion does not tell us how many features have to be selected, but rather allows us to rank the features from most to least discriminating instead. The TS/MB produce comparable AUC and accuracy to most of the other methods.

Table 10. Sentiment Extraction Case Average Performance - Comparison II

Method	AUC (%)	Accuracy (%)	#Original Features	# Selected Features	Size Reduction
MB	71.24	65.00	7,716	18	99.76%
TS/MB	77.26	69.12	7,716	21	99.71%
Naive Bayes	76.34	68.44	7,716	21	99.71%
SVM + TFIDF	68.90	69.43	7,716	21	99.71%
Voted perceptron	78.82	70.71	7,716	21	99.71%
Max. entropy	64.09	70.93	7,716	21	99.71%

In Table 11 we compare the performance of the TS/MB with others classifiers using the *same exact features*. We find that all the four competing classifiers performed better on the set of features in the Markov blanket. This suggests that the Markov Blanket approach is able to identify better sets of distinctive features in an automatic fashion. Whereas other feature selection methods, such as the information gain criterion, are not able to automatically identify the optimal number of features to select, or, given the number, the best features to choose.

Table 11. Sentiment Extraction Case Average Performance - Comparison III

Method	AUC (%)	Accuracy (%)	#Original Features	# Selected Features	Size Reduction
MB	71.24	65.00	7,716	18	99.76%
TS/MB	77.26	69.12	7,716	21	99.71%
Naive Bayes	80.81	70.36	7,716	21	99.71%
SVM + TFIDF	69.47	69.08	7,716	21	99.71%
Voted perceptron	79.61	70.93	7,716	21	99.71%
Max. entropy	66.03	71.14	7,716	21	99.71%

One possible explanation for the improvement in the accuracy and AUC may be the fact that the MB classifier encodes and takes advantage of dependencies among words conditional on the target variable.

Finally, in Figure 17 and Figure 18 below we show the initial MB DAG and the Tabu search enhanced MB DAG learned by the TS/MB classifier. Most of the directed edges are robust over five cross validation runs; the variation is fairly small.

Discussion - Sentiment Extraction Case

The TS/MB classifier that we have proposed is a fully automated system able to select a parsimonious vocabulary, customized for the classification task in terms of size and relevant features. Many techniques have been tried in order to automatically capture the way people express their opinions, including models for the contextual effects of negations, the use of feature frequency counts instead of their presence or absence, the use of different probability distributions for different positions of the words in the text, the use of sequences of words or N-grams, the combination of words and part of speech tags, noun-phrase chunks, and so on. However, the empirical results in terms of prediction accuracy and AUC always remain in the lower 70%s.

We performed three sets of experiments to compare the methods along various dimensions, in Tables 9, 10 and 11. The comparison of results of Table

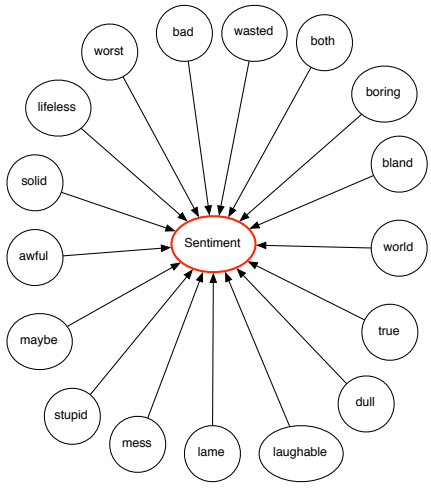


Figure 17. The initial MB DAG for the movie review data

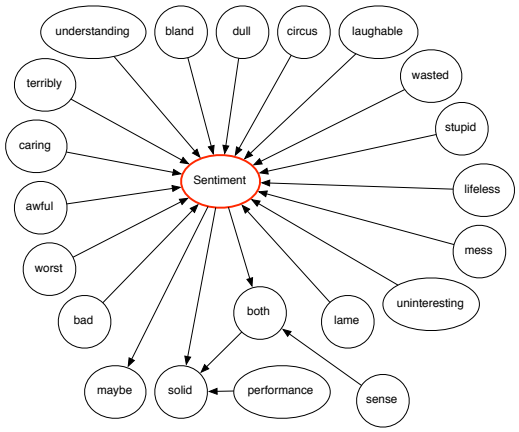


Figure 18. the best fitting MB DAG for the movie review data

10 and Table 11 suggests that information gain may not be the best criterion to select discriminating variables, but the statistical tests that measure association among features are better tools to perform the selection. The running time for TS/MB classifier varies according to the number of the variables and the number of the samples in cases where both the number of the variables and the sample size are small, TS/MB runs about the same time magnitude as the other classifiers. However, for the movie review data set , the number of variables is large (7716 variables), and the number of data samples is fairly big (1400

data samples), TS/MB took around tens of thousand of seconds, whereas the benchmark classifiers took at most thousands of seconds.

7. Application in Health Care Field: PCA Case Study

Early detection and diagnosis of prostate cancer has been a challenging task in Cancer Research community. Although efforts by health care researchers have resulted in better identification of individuals with cancer (Djavan et al., 1999, Howe et al., 2001, Stamey et al., 2002), overall early detection or determination of aggressive cancers is needed. In Section 5, we dicsussed the testing of our experimental design on PCA data set. In this section, we compare the results we got from the best experimental parameter configuration against several state-of-the-art classifiers.

We compare the performance of the TS/MB classifier first with single-stage MB classifier, the Markov Blanket classification without TS enhancing procedure, and then with those of six widely used classifiers mentioned in the experimental design section (Section 4).

As shown in the first two rows of Table 12, although the MB procedure by itself can identify a discriminating subset of predictors, the MB procedure coupled with TS further improves both AUC and accuracy. In Tables 12, 13 and 14 we present the average best-fitting classification results when compared against several state-of-the-art classifiers in three different ways. Comparison I presents the results when using the full set of variables as input. Comparison II uses the same number of variables as identified by TS/MB as input for all the other classifiers, selected by information gain criterion. Comparison III uses the exact same variables as identified by TS/MB as input variables.

Table 12. PCA Case Average Performance - Comparison I

Method	AUC (%)	Accuracy (%)	#Original Variables	#Predictor Variables	Size Reduction
MB	96.5	87.1	779	21	97.30%
TS/ MB	98.3	90.3	779	19	97.56%
Naive Bayes	97.5	89.3	779	779	0
SVM	97.1	98.5	779	779	0
Voted Perceptron	73.9	58.0	779	779	0
Maximum Entropy	97.4	98.8	779	779	0
K-NN	96.3	88.6	779	779	0
Logistic Regression	failed	failed	-	-	-

For Comparison I, we use the full data set as the input for each method. As shown in Table 12, the TS/MB classifier produces a substantially smaller variable set. In terms of AUC, the TS/MB classifier consistently yields the

best results; on accuracy, even with the smaller number of variables employed, it produces results comparable to the average performance of the state-of-the-art methods. Moreover, our algorithm identifies the 19 most discriminatory peaks out of 779 peaks that were identified by SELDI software program and the follow up clustering and peak alignment processes.

Table 13 presents the results for Comparison II. The TS/MB classifier dominates other methods both in terms of AUC and accuracy. This is possible because the TS/MB classifier is able to select the predictive features in the absence of evidence. In Table 14, we compare the results by using exact the same variables as were selected by the TS/MB classifier as the input variables for the other methods. Similar to Comparison II, the TS/MB substantially outperforms all the competitors.

Table 13. PCA Case Average Performance - Comparison II

Method	AUC (%)	Accuracy (%)	#Original Variables	#Predictor Variables	Size Reduction
MB	96.5	87.1	779	21	97.30%
TS/MB	98.3	90.3	779	19	97.56%
Naive Bayes	67.5	63.2	779	19	97.56%
SVM	63.3	62.0	779	19	97.56%
Voted Perceptron	65.2	59.2	779	19	97.56%
Maximum Entropy	64.7	64.1	779	19	97.56%
K-NN	65.6	58.6	779	19	97.56%
Logistic Regression	73.6	56.4	779	19	97.56%

Table 14. PCA Case Average Performance - Comparison III

Method	AUC (%)	Accuracy (%)	#Original Variables	#Predictor Variables	Size Reduction
MB	96.5	87.1	779	21	97.30%
TS/MB	98.3	90.3	779	19	97.56%
Naive Bayes	77.4	69.4	779	19	97.56%
SVM	72.6	69.9	779	19	97.56%
Voted Perceptron	75.4	67.8	779	19	97.56%
Maximum Entropy	74.9	70.9	779	19	97.56%
K-NN	72.1	65.3	779	19	97.56%
Logistic Regression	78.1	70.0	779	19	97.56%

For PCA data, the number of variables is not too big (779 variables), and the number of data samples are not too big (326 data samples), TS/MB on average took about several hundred of seconds while the other classifiers took from no time to about 100 seconds. These results are obtained from the discrete version

of the PCA data. We don't exclude the possibility that the benchmark classifiers such as SVM may have better performance on the continuous version of the data. A version of this case study can be found in the proceedings of the 9^{th} INFORMS Computing Society (Bai and Padman, 2005)

8. Application in Online Marketing Field: Consumer Response Case Study

In this case study, we use the data from DealTime (Brynjolfsson and Smith, 2002) to predict consumer response to

The Internet provides a wealth of information about online competing firms. Consumers use shopbots such as *www.dealtime.com* to search and compare offers online, and purchase the offer that maximized their latent utilities. Firms can monitor the changing prices, announcements of new products or services or of promotions of competitors through the customized intelligent searches such as *www.whizbang.com* (Montgomery and Kannan, 2002). Search engine companies, such as DealTime and Whizbang, can mine data to optimize their operational decisions about which stores to search, what advertisements to present, and which offers to present to a certain user. Online data offers many opportunities for firms to reduce their costs and increase sales, service quality and consumer utility. However, online data source are easy to collect but usually large and difficult to process. Thus being able to identify a set of essential features is important for business decision making.

Data

The dataset is from DealTime.com. DealTime is an international company with local shopbot sites in the U.S., U. K., and Germany. By using DealTime, customers can make worldwide price comparisons for books and other products from around 70 different retailers operating in 10 different countries. Customers visiting the site first identify the book they are interested in by searching on the title, the author, the publisher, or the ISBN. DealTime then queries distinct book retailers for information on this book. The prices and the delivery times are queried in real-time and thus represent the most up-to-date data from the retailers. Customers then evaluate different options in the price comparison table and click through a particular offer to complete the purchase at a retailer's site.

The data used in this paper includes three categories: offer data, session data and choice data. The offer data contains an individual price quote from a retailer, which in our analysis refers to item price, total price, discount and discount rate, tax information, price rank, shipping service type, shipping cost, delivery time, delivery availability and the position in an offer table. It also contains the information about the retailer such as retailer ID, country ID of

the retailer. Session data contains information of an individual search occasion for a book, which refers to the session identification number, customer IP address, country ID of the customer, the time and date the session inquiry is made, and the ISBN number for a book. Choice data includes click-through. A customer clicks through the link to a particular retailer if he or she is interested in its offer. By using the binary variable "click-through" as the proxy for customer response, we are able to track the traffic driven to a web site through the shopbot. Although the customer's visit to a store doesn't guarantee the actual purchase, Brynjolfsson and Smith (Brynjolfsson and Smith, 2002) show that the conversion rate of click-throughs and the actual purchase is positive and stable; it does not change significantly across stores.

We discretized the data by adding dummies to each category of categorical and non-ordinal variables in the data set. For continues ordinal variables, we first discretize the value, and put the value into different ranges, and assign dummy variables to each range. For discrete ordinal variables, we rank the value and assign dummy variables to each rank. The processed data has 2,936 observations from 48 retailers in 7 countries from January, 2002 to July, 2002. It includes 271 distinct customer searches for 79 distinct ISBN's, an average of 37 offers per search. Table 15 summarizes the characteristics of DealTime data set.

Table 15. Characteristics of the DealTime Data Set

Task	Variables	#Samples	Variable Types	Target Variable Type
Online Customer Response	235	2,936	discretized	binary (click/not click)

Results and Analysis

We conduct comparative studies the same manner as we did in the PCA case (Section 6). We compare the performance of our algorithm first with single-stage MB classifier, and then with the other six classifiers.

In Tables 16, 17 and 18 we show the average best-fitting classification results when compared against the other classifiers. Comparison I presents the results when using the full set of variables as input. Comparison II uses the same number of variables as identified by TS/MB as input for all the other classifiers, selected by information gain. Comparison III uses the exact same variables as identified by TS/MB as input variables.

For Comparison I, we use the full data set as the input for each method. As shown in Table 16, the TS/MB classifier identifies 7 out 234 predictor vari-

Table 16. Customer Response Case Average Performance - Comparison I

Method	AUC (%)	Accuracy (%)	#Original Variables	#Predictor Variables	Size Reduction
MB	78.8	97.0	234	12	94.87%
TS/ MB	82.8	97.6	234	7	97.01%
Naive Bayes	84.6	90.5	234	234	0
SVM	25.3	97.6	234	234	0
Voted Perceptron	73.1	97.7	234	234	0
Maximum Entropy	27.8	96.5	234	234	0
K-NN	46.0	97.6	234	234	0
Logistic Regression	82.8	97.7	234	234	0

ables. In terms of AUC, TS/MB classifier does substantially better than all the other classifiers; on accuracy, it produces results comparable to the average performance of the other methods.

Table 17. Customer Response Case Average Performance - Comparison II

Method	AUC (%)	Accuracy (%)	#Original Variables	#Predictor Variables	Size Reduction
MB	78.8	97.0	234	12	94.87%
TS/ MB	82.8	97.6	234	7	97.01%
Naive Bayes	80.5	92.6	234	7	97.01%
SVM	24.9	97.6	234	7	97.01%
Voted Perceptron	68.6	97.7	234	7	97.01%
Maximum Entropy	24.9	97.7	234	7	97.01%
K-NN	failed	failed	-	-	-
Logistic Regression	72.2	97.2	234	7	97.01%

Table 17 and 18 present the results for comparisons with the same number of variables and the same exact variables, respectively as input for other methods. Similar to Comparison I, TS/MB performs better on AUC and no worse on average accuracy.

Figure 19 below we show the best MB DAG learned by the TS/MB classifier for DealTime data.

For DealTime data, the number of the samples (2936 samples) are relatively large comparing to the number of the variables (235 variables). TS/MB on average took about tens of thousand of seconds to run, whereas the benchmark classifiers took from 500 to 5,000 seconds.

One characteristic of the DealTime data is that it is unbalanced data. About 96 percent of the labels of the target variable is 0. This makes the baseline accuracy [7] as high as 96 percent, which is the reason why all the accuracies in the

Table 18. Customer Response Case Average Performance - Comparison III

Method	AUC (%)	Accuracy (%)	#Original Variables	#Predictor Variables	Size Reduction
MB	78.8	97.0	234	12	94.87%
TS/ MB	82.8	97.6	234	7	97.01%
Naive Bayes	82.2	93.0	234	7	97.01%
SVM	25.4	97.9	234	7	97.01%
Voted Perceptron	70.5	97.9	234	7	97.01%
Maximum Entropy	25.4	97.8	234	7	97.01%
K-NN	failed	failed	-	-	-
Logistic Regression	74.8	97.2	234	7	97.01%

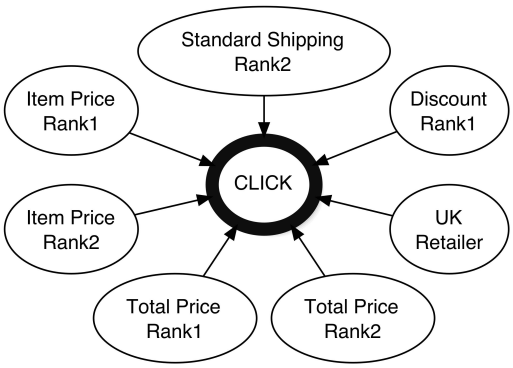


Figure 19. The best fitting MB DAG for the DealTime data set

tables look high. However, the AUC score varies a lot. This is partially because of the skewness of the data as well as the way the data is processed. When we assigned dummy variables to the ordinal variables in the data, we created dependant correlations among dummy variables. These correlations may make the data not linearly separable in high dimensional space. For classifiers that work well only on linearly separable data, they will not be able to distinguish the positive label from negative. In skewed data, the minority label have the bigger misclassification cost in AUC score. For DealTime data, misclassifying a label "click" as "not click" has bigger penalty in AUC score than the other way round. This explains why some classifiers have poor AUC score.

9. Summary

On average, the TS/MB classifier reduces the set of predictor variables by at least an order of magnitude from the full set of variables, in some cases to a sufficiently small set for entry into hand calculators or paper and pen-

cil decision procedures in clinical decision settings. At the same time, when compared to the state-of-the-art classification methods, the TS/MB classifier procedures good classification results in all the real world applications tested in this study, where the cost of misclassification has significant implications. These experiments, as well as more results we have obtained on data sets from text mining domain, suggest that for problems where the ratio of samples to the number of the variables is small, the TS/MB classifier is comparable in terms of the prediction performance, effectiveness in identifying critical predictors, and robustness.

It is possible that different Markov Blanket graphical structures consistent with the TS/MB classifier output would give slightly different classification results. The reasons can be the following: Any undirected and bi-directed edges are deleted after the edge orientation step, and these deletions might be suboptimal decisions. After the edge pruning step, the graph is often a partially directly graph (also called pattern), not a real DAG. The algorithm transforms this partially directly graph into a set of DAGs, and arbitrarily picks one of them as the starting solution of Tabu search procedure. Tabu search iteratively investigates alternative orientations and further edge additions to minimize the sub-optimality.

There are limitations of our algorithm. Although our algorithm is able to identify substantially fewer variables out of a large number of predictor variables, when the number of variables in the full data set goes beyond hundreds of thousands, our algorithm will not be able to perform and store the conditional probabilities needed for all the conditional independence tests. However, this problem can be solved by pairing with a segmentation method. This is feasible because the *InitialMBsearch* part is an "anytime" algorithm in the sense that it can be stopped at any stage, or run on any subset or superset of variables from a DAG, and the results are asymptotically correct in the following senses: edges not in the output are not in the DAG. Another alternative is to develop and run the parallel version of our algorithm. Theoretically, the computational complexity of our algorithm is bounded by the density or connectivity level of the underlying true graph, the empirical complexity may be worse. This is partially because the way the algorithm is implemented is not optimized, and partially because the characteristic of our algorithm itself. Since the algorithm is developed and run in the graphical space, generating, storing, reading and updating a graph alone with its parameters and inferences can be time consuming. On the other hand, for data sets with large number of samples, our algorithms takes a lot of time to run, because the number of evidences to check is increased very time an independence test is performed. These effects occurred in our experiments. For PCA data where the number of variables is under one thousand with a few hundred of data samples, TS/MB took about several hundred of seconds while the benchmark classifiers took from no time

to about 100 seconds. In movie review data or the DealTime data, where either the the number of variables is large (several thousands in movie review data) or the number of samples is large (several thousands in DealTime data), TS/MB took around tens of thousand of seconds, whereas the benchmark classifiers took from 500 to 5,000 seconds.

This research can be extended to address the interesting problem of simultaneously building classifiers for all variables in a large variable data set, or the problem of discovering a causal model for all variables in such data. Future research could be the exploration of heuristic approaches to global causal discovery problems.

10. Theoretical Properties

In this section we provide some preliminary theoretical properties of our algorithm. We first describe the intuitions of search space design and choice of search heuristic; then we prove the asymptotic correctness of *InitialMBsearch* in limit, and give the complexity analysis of *InitialMBsearch*; then we prove that Tabu search enhanced MB search improve the performance on finite sample cases; Finally we summarize the favorable aspects that Tabu search heuristic has for our learning and prediction problem settings.

Search for Markov Blankets Vs. DAGs

When searching in space of DAGs, the number of DAGs is super-exponential in the number of observed variables: the lower bound is $2^{\binom{n}{2}}$. For a data set with 5 variables, the number of possible DAGs is at least $2^5 = 1,024$; for a data set with 10 variables, the number of possible DAGs is at least $2^{45} \approx 10^{15}$. Theoretically, finding the optimal DAG has been shown NP-hard for the posterior probability score(Chickering, 2002). We suspect that it is the case as well for other scoring functions, but it has not been proved.

In a practical implementation, one faces two problems: computational complexity and limited data. Construction of a full Bayesian network for the purposes of classification may be computationally inefficient, as the whole structure may not be relevant to classification. Specifically, classification is unaffected by parts of the structure that lie outside the classification node's Markov blanket (Pearl, 2000). It is simpler to search for Markov Blanket DAGs than for the full DAGs because they contain fewer variables.

We also search directly for prediction score (accuracy and AUC). To our knowledge, previous research on learning Markov Blanket DAGs score the overall fit of a structure. The score is calculated on a weighted average schema over all the nodes in the network, which is called a kind of global scoring criterion. A "best-overall" structure may not be the "best" structure for certain

node. In classification or prediction structure learning problems, we only care about the local fit for the target node, which is measured as the conditional probability of the target node. The MB DAGs that yield the highest conditional probability of the target node are the structures that yield the highest prediction performance. In this sense, our scoring criterion is geared for the prediction problem, and is superior to global scoring criteria.

Heuristic search Vs. exhaustive search

When searching the space of Markov Blankets, the number of Markov Blankets is still exponential in the number of observed variables with a lower bound 2^n, where c is the number of children of the target variable. Chickering (Chickering, 2002)proposed and proved a two stage greedy search algorithm using single edge deletion and addition operations in the large sample limit is guaranteed to find the inclusion optimal structure giving the assumption that the probability distribution satisfies the Markov condition and faithfulness to the DAG. However, he also pointed out that his algorithm worked well only under the conditions where the graph is sparse and the dimensionality of the problem is small. He acknowledged that, in practice, it is still an open question whether the local maximum reached by his algorithm applied to real world data correspond to a model that is close in score to the global maximum, where there is no guarantee that the generative distribution has a perfect map in a DAG; and that there is enough data to support the asymptotic properties of the Bayesian scoring criterion. In summary, theoretically finding the optimal score is known to be correct in limit. In practical cases, there has not been a known way to find the optimal score because it takes too long or impossible to perform exhaustive search. Thus one needs to resort to heuristic search strategies.

InitialMBsearch **finds the true Markov Blanket in limit. (The Correctness of the** ***InitialMBsearch*****)**

Assumptions:

- (i) Markov and Faithfulness Assumptions: the probability distribution p satisfies the Markov condition (Definition 1) and Faithfulness (Definition 3) for some DAG G,
- (ii) no hidden variables,
- (iii) unlimited depth of search

Lemma 1: If $p(V)$ is Markov and faithful (Definition 3) to the graph G with the vertex set V, then

- (i) for each pair of vertices X, Y in G, X and Y are not adjacent if and only if X and Y are independent conditional on either the parents of X in G, or the parents of Y in G; and

- (ii) for each pair of vertices X, Y in G, X and Y are not adjacent if and only if X and Y are independent conditional on some subset of vertices not containing X or Y

Theorem: In the large sample limit, if the probability distribution is Markov and faithful to a DAG G that contains no hidden variables, the output of the *InitialMBsearch* is a pattern that represents the Markov equivalence class of G.

Proof Sketch:

Since an edge between T and v is removed from the graph if and only if T and v are independent conditional on some subset of vertices Z not containing T or v, by Lemma 1 (ii) the search only removes edges that are not in the true graph. After the first call to *checkedges*, since the search has checked whether T and any other vertex v are independent conditional on every subset of vertices adjacent to T, and the parents of T are adjacent to T, the search has checked whether v and T are independent conditional on the parents of T. However, some edges not in G might not have been removed yet, because the algorithm might not yet have checked whether T and v are independent conditional on the parents of v.

After the second set of calls to *checkedges*, if T and v are still adjacent, the algorithm checks whether T and v are independent conditional on any subset of vertices adjacent to v. Since the parents of v are a subset of the vertices adjacent to v, the search has checked whether T and v are independent conditional on the parents of v. So after this stage, the search has checked whether T and v are independent conditional on the parents of T, and on the parents of v. By Lemma 1(i), $adj(T)$ is the set of parents and children of T. Similarly, the *checkedges* method is applied to one endpoint of each edge v—w such that v is in $adj(T)$ and w is in $adj(v)$. Hence for each v in $adj(T)$, and w in $adj(v)$ the algorithm has checked whether v and w are independent conditional on the parents of v.

checkedges is then applied to the other endpoint of each such edge, if it hasn't already been so applied, without re-adding any edges that have been removed. So at this point, for each v in $adj(T)$, and w in $adj(v)$ the algorithm has checked whether v and w are independent conditional on the parents of v, and on the parents of w. Hence the adjacencies for T are correct, and the adjacencies for each vertex v adjacent to T are correct. Hence for T (or any vertex v adjacent to T), the set of vertices adjacent to T (or v) is the union of the parents and children of T (or v).

Then, the orientation rules are applied. The correctness of orientation rules have been proved in (Spirtes et al., 2000, page 410). So, any orientation in the output is the same as in the true graph G.

After orienting the edges, any node that is outside the range of MB DAG is trimmed away as well as the corresponding edges.

Some edges may be un-oriented or bi- oriented in finite samples. In the large sample limit where the data is sufficient, and the distribution satisfies the Markov conditions and is faithful to the true DAG, and there are no unmeasured common causes of a pair of variables in the true causal graph G, there will be no bi- oriented edges. The *InitialMBsearch* is able to identify the pattern representing the true MB DAG.

Complexity analysis of the Initial MB search

In the large sample limit, the complexity for a MB is bounded by the largest degree (i. e.) in MB. Let k be the maximal degree of any vertex and let n be the number of vertices. The number of conditional independence tests in the *InitialMBsearch* in the worst case is bounded by $\frac{k^2 \cdot (n-1)^{k-1}}{(k-1)!}$ (Spirtes et al., 2000), which is the same as PC algorithm. The worst case is everyone adjacent to T In real world applications, this does not happen very often. Thus empirically, Tabu Search enhanced Markov Blanket algorithm is much faster than PC because it does not consider adjacencies or orientations outside the MB DAG.

Evaluation of Tabu Search

Criteria for evaluating search strategies.

- Correctness in the large sample limit
- Search Cost
  - Time Efficiency
  - Space (Memory) Efficiency
- Path Cost - Optimality: The optimal solution to a problem is the cheapest, quickest, or otherwise most efficient route through the state space

The correctness of our algorithm in the large sample limit has been addressed in the above discussion. In terms of search cost or efficiency, with a variable search like Tabu search, it is impossible to determine what, if any, improvements to efficiency will be made The algorithm will not necessarily find the most efficient route through the state space. However, Tabu search has several favorable properties because of its adaptive memory capability - both short term and long term - and it appears particularly suited to the Bayesian Networks and Markov Blanket approaches:

- **Tabu Search does not stop at the first local optima**, whereas local heuristic search such as Best First Search (BFS) does. In this sense, TS searches a bigger space in general.

- **Tabu Search efficiently drives the search away from previous visited neighborhood** by controlling the size of the Tabu list, the longer the list is, the further away the solutions are. Because if we interpret reversal as a two step look-ahead move, an operation of reversal of $X_1 \rightarrow X_2$, is equivalent to the set without ordering $deleteX_1 \rightarrow X_2, addX_2 \rightarrow X_1$, based on this argument, there is a unique set of moves (addition, deletion) between two *MBs* : $MB_1 \rightarrow MB_2$, i. e., $MB_2 = f(MB_1) \rightarrow MB_1 = f^{-1}(MB_2)$. In this sense, by just keeping m previous moves instead of m previous solutions (MBs), TS prevents revisiting the m most recent solutions. The smart use of dynamic memory is one of the efficiency and effectiveness of our procedure. BFS does not remember any history and it is more likely that it get stuck at the very first several steps, in Bayesian Network and Markov Blanket search space.

In summary, *InitialMBsearch* identifies the asymptotically correct structure in the large sample limit as long as there are no latent variables, and Tabu search searches for the highest score structure on finite samples.

Acknowledgments

We thank Professor Fred Glover of the University of Colorado at Boulder, Joseph Ramsey at Carnegie Mellon University, and Frank Wimberly at Institute for Human and Machine Cognition for valuable suggestions and help. We also thank Professor William Cohen of Carnegie Mellon University for providing the MinorThird software package as well as his helpful comments, and Roy Lipski, founder of Infonic, for providing us with the annotated corpora. Finally, we thank the email Learning Group at School of Computer Science at Carnegie Mellon University for generously providing us with software.

Notes

1. If there is a directed path from node A to node B in a DAG, then node B is a descendant of node A; node A is a parent of node B

2. Because there are some moves which are identical of combinations of other moves, it is possible aht some states may be visited more than once, although they cannot be revisited indefinitely many times.

3. Maximum Entropy methods are also know as autoregressive model.

4. In our experiments, k is equal to 3.

5. The data was discretized by Aliferis, Tsamardinos and Statnikov from Discovery Systems Laboratory of Vanderbilt University Medical Center.

6. Information gain is a measure of the effectiveness of an attribute in classifying the training data. It is simply the expected reduction in entropy caused by partitioning the examples according to this attribute. Entropy is a measure of the impurity in a collection of training points. Details can be found in *Machine Learning* (Mitchell, 1997)

7. Baseline accuracy is calculated by classifying every data point as the dominant label.

References

Abraham, A., Nath, B., and Mahanti, P.K. (2001). Hybrid intelligent systems for stock market analysis. *Computational Science*, pages 337–345.

Adam, B.L., Qu, Y., Davis, J.W., Ward, M.D., Clements, M.A., Cazares, L.H., Semmes, O.J., Schellhammer, P.F., Yasui, Y., Feng, Z., and Wright, G.L.Jr. (2002). Serum protein fingerprinting coupled with a pattern-matching algorithm distinguishes prostate cancer from benign prostate hyperplasia and healthy men. *Cancer Research*, 62:3609–3614.

Airoldi, E., Anderson, A., Fienberg, S., and Skinner, K. (2004). Who wrote Ronald Reagan radio addresses? Manuscript.

Aliferis, C., Tsamardinos, I., and Statnikov, A. (2003). Hiton, a novel markov blanket algorithm for optimal variable selection.

Bai, X., Glymour, C., Padman, R., Spirtis, P., and Ramsey, J. (2003). Pcx classifier for large data sets with few cases. Technical Report CMU-CALD-04-102, School of Computer Science, Carnegie Mellon University.

Bai, X. and Padman, R. (2005). Tabu search enhanced markov blanket classifier for high dimensional data sets. In *Proceedings of* 9^{th} *INFORMS Computing Society*, pages 338–354. Kluwer Academic Publisher.

Berry, M.J.A. and Linoff, G.S. (1997). *Data Mining Techniques: For Marketing, Sales, and Customer Support*. John Wiley and Sons.

Brynjolfsson, E. and Smith, M. D. (2002). The great equalizer? the role of shopbots in electronic markets. Working Paper, MIT Sloan School of Management, Cambridge, MA.

Cheng, J., Hatzis, C., Hayashi, H., Krogel, M.A., Morishita, ., Page, D., and Sese, J. (2002). KDD cup 2001 report. *SIGKDD Explorations*, 3(2):47–64.

Chickering, D.M. (2002). Learning equivalence classes of bayesian-network structures. *Journal of Machine Learning Research*, 3:507–554.

Chickering, D.M., Meek, C., and Heckerman, D. (2003). Large-sample learning of bayesian networks is np-hard. In *Proceedings of Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 124–133. Morgan Kaufmann.

Cooper, G.F., Aliferis, C.F., Aronis, J., Buchanan, B.G., Caruana, R., Fine, M.J., Glymour, C., Gordon, G., Hanusa, B.H., Janosky, J.E., Meek, C., Mitchell, T., Richardson, T., and Spirtes, P. (1992). An evaluation of machine-learning methods for predicting pneumonia mortality. *Artificial Intelligence in Medicine*, 9:107–139.

Freeman, J. and Skapura, D. (1991). *Neural Networks*. Addison-Wesley.

Freund, Y. and Schapire, R.E. (1999). Large margin classification using the perceptron algorithm. *Machine Learning*, 37:277–296.

Fu, Z., Golden, B.L., Lele, S., Raghavan, S., and Wasil, E.A. (2004). A genetic algorithm-based approach for building accurate decision trees. *INFORMS Journal on Computing*, 15:3–22.

Glover, F. (1989). Tabu search - part i. *ORSA Journal on Computing*, 1, No. 3:190–206.

Glover, F. (1997). *Tabu Search*. Kluwer Academic Publishers.

Greenberg, H. J. (2004). Mathematical programming online glossary.

Hanley, J.A. and McNeil, B.J. (1983). A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148:839–843.

Harwood, S. and Scheines, R. (2002). Genetic algorithm search over causal models. Technical Report CMU-PHIL-131, Department of Philosophy, Carnegie Mellon University.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

http://www.cs.cornell.edu/people/pabo/movie-review-data/ (2002). Data movie review.

http://www.tabusearch.net/. Tabu search weblink.

Jesus, M. (2001). *Web Mining for Profit: E-Business Optimization*. Butterworth-Heinemann.

Joachims, T. (2001). A statistical learning model of text classification with support vector machines. In *Proceedings of the Conference on Research and Development in Information Retrieval*, pages 128–136. ACM.

Johnson, D.S., Aragon, C.R., McGeoch, L.A., and Schevon, C. (1989). Optimization by simulated annealing: an experimental evaluation. *Part I, graph partitioning, Operations Research*, 37:6:865–892.

Johnson, D.S. and McGeoch, L.A. (1997). The traveling salesman problem: A case study in local optimization. In E.H.L., Aarts and J.K., Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley and Sons.

Koller, D. and Sahami, M. (1996). Towards optimal feature selection. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 284–292. Morgan Kaufmann.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289. In: Proceedings of the Eighteenth International Conference on Machine Learning.

Lewis, D. D. (1991). Evaluating Text Categorization. In *Proceedings of Speech and Natural Language Workshop*, pages 312–318. Morgan Kaufmann.

Madden, M. G. (2003). Evaluation of the performance of the markov blanket bayesian classifier algorithm.

Margaritis, D. (2003). Learning bayesian network model structure.

Margaritis, D. and Thrun, S. (1999). Bayesian network induction via local neighborhoods. In *Advances in Neural Information Processing System*.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal. Chemical Physics*, 21-6:1087–1092.

Mitchell, T. (1997). *Machine Learning.* McGraw-Hill.

Moll, R., Perkins, T.J., and Barto, A.G. (2000). Machine learning for subproblem selection. In *Proceedings 17th International Conf. on Machine Learning*, pages 615–622. Morgan Kaufmann, San Francisco, CA.

Montgomery, A. and Kannan, S. (2002). Learning about customers without asking. GSIA Working Paper,Carnegie Mellon University.

Mosteller, F. and Wallace, D.L. (1964). *Inference and Disputed Authorship: The Federalist*. Addison-Wesley.

Mosteller, F. and Wallace, D.L. (1984). *Applied Bayesian and Classical Inference: The Case of The Federalist Papers*. Springer-Verlag.

Nigam, K., McCallum, A., Thrun, S., and Mitchell, T. (2000). Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39:103–134.

Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 79–86.

Pearl, J. (2000). *Causality: Models, Reasoning, and Inference*. Cambridge University Press.

Piatetsky-Shapiro, G. and Steingold, S. (2000). Measuring lift quality in database marketing. *SIGKDD Explorations,*, 2:76Ű80.

Provost, F., Fawcett, T., and Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453.

Rego, C. and Alidaee, B. (2004). *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*. Kluwer Academic Publishers.

Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, Prediction, and Search*. MIT Press.

Sreerama, K. M., Kasif, S., and Salzberg, S. (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32.

Toth, P. and Vigo, D. (2003). The granular tabu search and its application to the vehicle routing problem. *INFORMS Journal on Computing*, 15:4:334–346.

Tsamardinos, I., Aliferis, C., and Statnikov, A. (2002). Large-scale feature selection using markov blanket induction for the prediction of protein-drug binding.

Tsamardinos, I., Aliferis, C., and Statnikov, A. (2003). Algorithms for large-scale local causal discovery in the presence of small sample or large causal neighborhoods. Technical Report DSL-02-08, Vanderbilt University.

Weiss, S.M. and Kulikowski, C.A. (1991). *Computer Systems That Learn*. Morgan Kaufmann.