2014

Structure learning in Bayesian networks and session analysis of people search within a professional social network

Ru He
Iowa State University

Structure learning in Bayesian networks and session analysis of people search within a professional social network

by

Ru He

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Co-majors: Computer Science; Statistics

Program of Study Committee:

Jin Tian, Co-major Professor

Huaiqing Wu, Co-major Professor

Samik Basu, Co-major Professor

Andrew S. Miner

Arka Ghosh

Iowa State University

Ames, Iowa

2014

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. GENERAL INTRODUCTION

1.1 Introduction and Motivation

Statistical learning refers to a set of methodologies for modeling and understanding data. As an interdisciplinary field between computer science and statistics, nowadays statistical learning has played a critical role in many scientific areas and business practices.

In statistical learning, there is one problem that has been a focus of many researchers in the last two decades. This is the problem of learning Bayesian network structures from the data. A Bayesian network structure is a directed acyclic graph (DAG) whose nodes represent random variables and whose edges correspond to the direct probabilistic dependencies. The problem of learning Bayesian network structures is so attractive because the learned Bayesian network structures not only can be used for inference but also open a door to cause-and-effect analysis and enable people to discover the underlying mechanism of the problem domain. For instance, with the semantics of a Bayesian network structure G, the existence of an edge from node X to node Y in G can be interpreted as the fact that variable X directly influences variable Y; and the existence of a directed path from node X to node Y can be interpreted as the fact that X eventually influences Y.

Therefore, this dissertation will present three different approaches for this learning problem. The first approach is to propose a dynamic programming algorithm that can compute the exact posterior probability of any modular feature of a Bayesian network with any general structure prior in $O(3^n)$ time. The second approach is to develop a dynamic programming algorithm for obtaining the k-best Bayesian network structures and then use these k-best network structures to compute the posterior probabilities of hypotheses of interest based on Bayesian model averaging. The third approach is to develop new algorithms to efficiently sample Bayesian network structures according to the exact structure posterior and then use these sampled structures to construct estimators for the posterior of any feature. These

three approaches all use dynamic programming techniques to learn Bayesian network structures from data.

In statistical learning, recently one classification problem has become more and more important as a sharply increasing number of people join professional social networks and use their search engines for professional development purposes. This is session identification of people search within a professional social network. Session is typically defined as a series of queries submitted by a single user during one episode of interactions between the user and the web search engine for one single information need. Given all the search queries submitted from a user, session identification is to classify whether or not a given search query is the start of new session so that a session boundary can be inserted. Once session identification can be correctly performed, the corresponding analysis is able to serve as the basis to analyze web users' search behaviors. Using session analysis, the search quality of current web search engines can be measured and then the corresponding improvements can possibly be made. Moreover, the performance of web search can often be improved if all the information in the same session of current query is used as the context of the search.

Therefore, this dissertation will present the work of session identification and analysis for the domain of people search within a professional social network. Two important refinements are proposed to address the drawbacks of the content-based method, one of two main session identification methods commonly used in real applications. We describe the underlying rationale of our refinements and then empirically show that the content-based method equipped with our refinements is able to achieve an excellent identification performance in the domain. Finally, based on our refined content-based session identification method, the corresponding session analysis is performed and the profession-oriented nature of the domain is illustrated.

1.2 Dissertation Organization

The dissertation consists of four main chapters, preceded by this general introduction. Each of these four main chapter corresponds to a conference or journal paper. Chapter 2 presents an algorithm that can compute the exact posterior probability of any modular feature of an Bayesian network with a general structure prior. Chapter 3 provides an algorithm that can find the k-best Bayesian network structures and

proposes to use these k-best structures to compute the posterior probabilities of hypotheses of interest by Bayesian model averaging technique. Chapter 4 describes sampling algorithms including the first algorithm that is able to efficiently sample Bayesian network structures according to the exact structure posterior. Chapter 5 presents session identification and analysis for the domain of people search within a professional social network. Finally Chapter 6 concludes the dissertation.

1.3 Background

Before describing the four main chapters, in this section we introduce some background knowledge for the dissertation.

1.3.1 Probability

In the subsection, we introduce one important definition in probability which serves a basis for Bayesian networks. Please refer to (Casella and Berger, 2002) for other basic knowledge in probability and statistics. Please also note that we focus on the case of discrete random variables.

Definition 1 (Conditional independence) *Let* $\mathbf{X}$*,* $\mathbf{Y}$ *and* $\mathbf{Z}$ *be three sets of random variables. We say that* $\mathbf{X}$ *is conditionally independent of* $\mathbf{Y}$ *given* $\mathbf{Z}$*, denoted as* $\mathbf{X} \perp \mathbf{Y}|\mathbf{Z}$*, if* $P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}|\mathbf{Z} = \mathbf{z}) = P(\mathbf{X} = \mathbf{x}|\mathbf{Z} = \mathbf{z}) \cdot P(\mathbf{Y} = \mathbf{y}|\mathbf{Z} = \mathbf{z})$ *for all the values* $\mathbf{x}$*,* $\mathbf{y}$*,* $\mathbf{z}$ *of* $\mathbf{X}$*,* $\mathbf{Y}$*,* $\mathbf{Z}$ *such that* $P(\mathbf{Z} = \mathbf{z}) > 0$*.*

1.3.2 Graph

In this subsection, we cover some important concepts in graph that are closely related to our work. You can refer to (Cormen et al., 2001; Koller and Friedman, 2009) for the comprehensive knowledge in graph.

A directed graph G is a pair (V, E), where V is a finite set and E is a binary relation on V. The set V is called the node set of G, and its elements are called nodes. The set E is called the edge set of G, and its elements are called edges, which are ordered pairs of nodes. By convention, we use $u \rightarrow v$ to represent an edge, where $u, v \in V$. The notation $v \leftarrow u$ is equivalent to $u \rightarrow v$.

In a directed graph $G = (V, E)$, If $u \rightarrow v \in E$, we say that $u \rightarrow v$ leaves node u and enters node v. In G, the in-degree of a node is the number of edges entering it, and the out-degree of a node is the

number of edges leaving it.

In a directed graph $G = (V, E)$, a undirected path of length k from a node u to a node u' is defined as a sequence $< v_0, v_1, v_2, \cdots, v_k >$ such that $u = v_0$, $u' = v_k$, and for each $i \in \{1, 2, \cdots, k\}$, either $v_{i-1} \rightarrow v_i \in E$ or $v_{i-1} \leftarrow v_i \in E$. Different from a undirected path, a directed path of length k from a node u to a node u' is defined as a sequence $< v_0, v_1, v_2, \cdots, v_k >$ such that $u = v_0$, $u' = v_k$, and for each $i \in \{1, 2, \cdots, k\}$, $v_{i-1} \rightarrow v_i \in E$.

In a directed graph G, a directed path $< v_0, v_1, v_2, \cdots, v_k >$ forms a cycle if $v_0 = v_k$ and $k \geq 1$. If G has no cycle, it is called acyclic. We also take the first letters of "directed acyclic graph" and call such a graph as a DAG.

In a directed graph $G = (V, E)$, whenever we have $u \rightarrow v \in E$, we say that u is the parent of v in G, and that v is the child of u in G. We use $Pa(u)$ to denote the set of all the parents of u, $Ch(u)$ to denote the set of all the children of u. We say that u is an ancestor of v, and that v is a descendant of v, if there exists a directed path from u to v. We use $Anc(u)$ to denote the set of all the ancestors of u, $Des(u)$ to denote the set of all the descendants of u, and $NonDes(u)$ to denote the set of all the nodes in $V - Des(u)$, i.e., the set of all the non-descendants of u.

1.3.3 Bayesian Network

In this subsection, we cover the background knowledge for Bayesian network (BN).

Definition 2 (Bayesian network structure) *A Bayesian network structure G is a DAG (directed acyclic graph) whose nodes represent random variables $X_1, X_2, \cdots, X_n$. A BN structure G encodes the following set of conditional independence assumptions (denoted by $\boldsymbol{I}_l(G)$): for each variable (node) X_i, X_i is conditionally independent of its non-descendants given its parents, i.e., $X_i \perp NonDes(X_i)|Pa(X_i)$. $\boldsymbol{I}_l(G)$ is often called the set of local Markov independencies of G.*

Definition 3 (Independencies in P) *Let P be a distribution over a set of random variables $V = \{X_1, X_2, \cdots, X_n\}$. We define $\boldsymbol{I}(P)$ to be the set of independence assertions of the form $(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z})$ that hold in P.*

Definition 4 (Factorization) *Let G be a BN structure over a set of random variables $V = \{X_1, X_2, \cdots, X_n\}$. We say that a distribution P over the same set V factorizes according to G if P can be expressed as*

a product $P(X_1, X_2, \cdots, X_n) = \prod_{i=1}^{n} P(X_i|Pa(X_i))$. *This equation is called the chain rule for Bayesian networks. The individual factors* $P(X_i|Pa(X_i))$ *are called conditional probability distributions (CPDs).*

Definition 5 (Bayesian network) *A Bayesian network is a pair* (G, P)*, where* G *be a BN structure, and* P *factorizes according to* G *and is specified as a set of CPDs associated with* G*'s nodes.*

Definition 6 (d-separation) *Let* G *be a BN structure. A collider on a undirected path p in* G *is a node with two incoming edges that belong to p. A path p between node* X *and node* Y *given a conditioning set* $\mathbf{Z}$ *in* G *is active, if (i) every collider of p is in* $\mathbf{Z}$ *or has some descendant in* $\mathbf{Z}$*, and (ii) no other node on p is in* $\mathbf{Z}$ *(Pearl, 1988). If a path is not active, then it is blocked. Let* $\mathbf{X}$*,* $\mathbf{Y}$*,* $\mathbf{Z}$ *be three sets of nodes in* G*. We say that* $\mathbf{X}$ *and* $\mathbf{Y}$ *are d-separated given* $\mathbf{Z}$*, denoted by d-sep*$_G(\mathbf{X}; \mathbf{Y}|\mathbf{Z})$*, if there is no active path between any node* $X \in \mathbf{X}$ *and any node* $Y \in \mathbf{Y}$ *given* $\mathbf{Z}$*. We let* $\boldsymbol{I}(G)$ *denote the set of independencies that corresponds to d-separation, i.e.,* $\boldsymbol{I}(G) = \{(\mathbf{X} \perp \mathbf{Y}|\mathbf{Z}) : \text{d-sep}_G(\mathbf{X}; \mathbf{Y}|\mathbf{Z})\}$*.* $\boldsymbol{I}(G)$ *is often called the set of global Markov independencies of* G*.*

Definition 7 (I-map, D-map and P-map) *Let* G *be a BN structure over a set of random variables* V *and let* P *be a distribution over the same set* V*. We say that* G *is an I-map (Independency map) for* P *if* $\boldsymbol{I}(G) \subseteq \boldsymbol{I}(P)$*. We say that* G *is a minimal I-map for* P *if one of its edges can be deleted without destroying its I-mapness. We say that* G *is a D-map (Dependency map) for* P *if* $\boldsymbol{I}(G) \supseteq \boldsymbol{I}(P)$*. We say that* G *is a P-map (Perfect map) for* P *if* $\boldsymbol{I}(G) = \boldsymbol{I}(P)$*.*

Definition 8 (I-equivalence) *Two BN structures* G_1 *and* G_2 *over the same random variable set* V *are said to be I-equivalent (Independence equivalent) if* $\boldsymbol{I}(G_1) = \boldsymbol{I}(G_2)$*. Since the I-equivalence relation is an equivalence relation (reflexive, symmetric and transitive), the set of all DAGs over* V *is partitioned into a set of mutually exclusive and exhaustive I-equivalence classes.*

1.3.4 Learning Bayesian Network Structures

In the last two decades, there have been a large number of researches focusing on the problem of learning Bayesian network structure(s) from the data. These researches deal with a common real situation where the underlying Bayesian network is typically unknown so that it has to be learned from

the observed data. Note that learning BN structures constitutes the central and the most challenging part of the learning of whole Bayesian networks. Once a BN structure is learned, the remaining learning task for the whole Bayesian network, the learning of the corresponding parameters in the distribution, is relatively easy. It is essentially a standard parameter estimation problem which has been well studied in statistics and has closed-form solutions under appropriate distribution assumptions.

There are three general approaches for learning BN structures from data in the literature: the constraint-based approach, the score-based approach and the Bayesian model averaging approach. In the following we will introduce these general learning approaches.

The first BN structure learning approach is constraint-based approach (Margaritis and Thrun, 1999; Spirtes et al., 2001; Aliferis et al., 2003; Tsamardinos et al., 2003; Pellet and Elisseeff, 2008). The constraint-based approach views a BN structure G as a representation of a set of conditional independence/dependence relations and uses these represented relations to discover the underlying G, under the assumption of P-mapness. Based on the (conditional independence/dependence) conclusions returned from some statistical test procedure (such as Pearson X^2 test or G^2 test), a constraint-based method learn each local structural feature (the existence/nonexistence of the undirected edge feature and edge direction feature). All these learned local structural features can be combined to rebuild a global BN structure (or more precisely an I-equivalence class of BN structures) that has the best incorporation of these conditional independence/dependence conclusions.

Constraint-based methods are very intuitive since they directly follow the semantics of BN structure. Under the assumption of bounded in-degree, the number of statistical tests of many constraint-based methods are polynomial in terms of the number of variables n, i.e., their complexity are polynomial in n. However, there are one general problem inherent in the constraint-based approach. The correctness of the constraint-based approach relies on the assumption that their used statistical conditional independence tests are perfect, i.e., there is no error (neither type I error nor type II error) in each performed test. Such an assumption, of course, does not hold in the reality and any kind of statistical test will have some probability of making errors. Unfortunately, the constraint-based approach is sensitive to an error from an individual statistical test. An error of a statistical test can lead to the wrong decision in the correspondingly local structural feature, which can further result in the propagated errors in the consequent learning process.

The second BN structure learning approach is the score-based approach (Heckerman et al., 1995; Chickering, 2002a,b; Castelo and Kocka, 2003; Singh and Moore, 2005; Silander and Myllymaki, 2006). The score-based approach views a BN structure as a statistical model and addresses the learning as a model selection problem. Each score-based method defines a space of potential models (i.e., the set of all the possible BN structures) as well as a scoring function $Score(G : D)$ that evaluates how well a BN structure G fits the observed data D. (In Bayesian approach, the score of a DAG G is simply the posterior $p(G|D)$ of G given data D.) Then the main task of score-based approach is to find a best scoring BN structure by searching through the space of all the possible BN structures. Finally, this best scoring BN structure (or its I-equivalence class) is used for the future prediction.

The advantage of the score-based approach is that it views the whole BN structure at once so that it is less sensitive to the error in some local structure discovery. The main challenge for any score-based method is that the space of BN structures is super-exponential so that the exhaustive search over the whole space is not tractable for the domain including a large number of variables. In general, such a search problem is NP-hard (Chickering et al., 1994) and some heuristic search skills have to be applied for the large domain. For a domain with less than 30 variables, recently researchers (Singh and Moore, 2005; Silander and Myllymaki, 2006) develop the method to learn a truly best scoring network using dynamic programming techniques. Another problem in the usage of the score-based approach is in the situations where the amount of data is small relative to the size of the model. In these situations, there are often many high-scoring models so that using a single model will probably lead to unwarranted conclusions about the structural features and poor predictions about new observations.

The third approach of learning BN structures is the Bayesian model averaging approach (Madigan and York, 1995; Friedman and Koller, 2003; Koivisto and Sood, 2004; Koivisto, 2006; Eaton and Murphy, 2007; Grzegorczyk and Husmeier, 2008; Parviainen and Koivisto, 2011; Niinimaki et al., 2011). Instead of learning and then using only one single DAG (i.e., BN structure), this approach intends to make the prediction based on a set of all the possible DAGs. The approach assigns each DAG G a posterior $P(G|D)$ based on the observed data D. Then $P(h|D)$, the posterior probability of any hypothesis of interest h, can be computed by averaging over all possible DAGs as $P(h|D) = \sum_G P(h|G, D)P(G|D)$.

Bayesian model averaging approach directly addresses the problem in score-based approach when the size of the observed data is small relative to the size of the model. However, just as we have

described for score-based approach, since the space of BN structures is super-exponential, the main challenge in Bayesian model averaging approach is also its computational costs. When the exact full Bayesian model averaging is intractable, some approximate techniques are often resorted for Bayesian model averaging. In this dissertation, we will describe three methods for Bayesian model averaging: one performs the exact full model averaging and the other two methods use approximate techniques in the model averaging.

CHAPTER 2. COMPUTING POSTERIOR PROBABILITIES OF STRUCTURAL FEATURES IN BAYESIAN NETWORKS BY FULL BAYESIAN MODEL AVERAGING

A paper published in *Proceedings of the* 25*th Conference on Uncertainty in Artificial Intelligence*

Jin Tian and Ru He

Abstract

We study the problem of learning Bayesian network structures from data. Koivisto and Sood (2004) and Koivisto (2006) presented algorithms that can compute the exact posterior probability of a subnetwork, e.g., a single edge, in $O(n2^n)$ time and the posterior probabilities for all $n(n-1)$ potential edges in $O(n2^n)$ total time, assuming that the in-degree, i.e., the number of parents per node, is bounded by a constant. One main drawback of their algorithms is the requirement of a special structure prior that is non uniform and does not respect Markov equivalence. In this paper, we develop an algorithm that can compute the exact posterior probability of a subnetwork in $O(3^n)$ time and the posterior probabilities for all $n(n-1)$ potential edges in $O(n3^n)$ total time. Our algorithm also assumes a bounded in-degree but allows general structure priors. We demonstrate the applicability of the algorithm on several data sets with up to 20 variables.

2.1 Introduction

Bayesian networks are being widely used for probabilistic inference and causal modeling (Pearl, 2000; Spirtes et al., 2001). One major challenge is to learn the structures of Bayesian networks from data. In the Bayesian approach, we provide a prior probability distribution over the space of possible Bayesian networks and then computer the posterior distributions $P(G|D)$ of the network structure G

given data D. We can then compute the posterior probability of any hypothesis of interest by averaging over all possible networks. In many applications we are interested in structural features. For example, in causal discovery, we are interested in the causal relations among variables, represented by the edges in the network structure (Heckerman et al., 1999).

The number of possible network structures is super-exponential $O(n!2^{n(n-1)/2})$ in the number of variables n. For example, there are about 10^4 directed acyclic graphs (DAGs) on 5 nodes, and 10^{18} DAGs on 10 nodes. As a result, it is impractical to sum over all possible structures unless for very small networks (less than 8 variables). One solution is to compute approximate posterior probabilities. Madigan and York (1995) used Markov chain Monte Carlo (MCMC) algorithm in the space of network structures. Friedman and Koller (2003) developed a MCMC procedure in the space of node orderings which was shown to be more efficient than MCMC in the space of DAGs. One problem to the MCMC approach is that there is no guarantee on the quality of the approximation in finite runs.

Recently, a dynamic programming (DP) algorithm was developed that can compute the exact marginal posterior probabilities of any subnetwork (e.g., an edge) in $O(n2^n)$ time (Koivisto and Sood, 2004) and the exact posterior probabilities for all $n(n-1)$ potential edges in $O(n2^n)$ total time (Koivisto, 2006), assuming that the in-degree, i.e., the number of parents of each node, is bounded by a constant. One main drawback of the DP algorithm and the order MCMC algorithm is that they both require a special form of the structure prior $P(G)$. The resulting prior $P(G)$ is non uniform, and does not respect Markov equivalence (Friedman and Koller, 2003; Koivisto and Sood, 2004). Therefore the computed posterior probabilities could be biased. MCMC algorithms have been developed that try to fix this structure prior problem (Eaton and Murphy, 2007; Ellis and Wong, 2008).

Inspired by the DP algorithm in (Koivisto and Sood, 2004; Koivisto, 2006), we have developed an algorithm for computing the exact posterior probabilities of structural features that does not require a special prior $P(G)$ other than the standard structure modularity (see Eq. (4.6)). Assuming a bounded in-degree, our algorithm can compute the exact marginal posterior probabilities of any subnetwork in $O(3^n)$ time, and the posterior probabilities for all $n(n-1)$ potential edges in $O(n3^n)$ total time. The memory requirement of our algorithm is $O(n2^n)$, which is the same as the one of the DP algorithm in (Koivisto and Sood, 2004; Koivisto, 2006). We have demonstrated our algorithm on data sets with up to 20 variables. The main advantage of our algorithm is that it can use very general structure prior

$P(G)$ that can simply be left as uniform and can satisfy Markov equivalence requirement. We acknowledge here that our algorithm was inspired by and used many techniques in (Koivisto and Sood, 2004; Koivisto, 2006). Their algorithm is based on summing over all possible total orders (leading to the bias on prior $P(G)$ that graphs consistent with more orders are favored). Our algorithm directly sums over all possible DAG structures by exploiting sinks, nodes that have no outgoing edges, and roots, nodes that have no parents, and as a result the computations involved are more complicated. We note that the dynamic programming techniques have also been used to learn the optimal Bayesian networks in (Singh and Moore, 2005; Silander and Myllymaki, 2006).

The rest of the paper is organized as follows. In Section 4.2 we briefly review the Bayesian approach to learn Bayesian networks from data. In Section 2.3 we present our algorithm for computing the posterior probability of a single edge and in Section 2.4 we present our algorithm for computing the posterior probabilities of all potential edges simultaneously. We empirically demonstrate the capability of our algorithm in Section 4.4 and discuss its potential applications in Section 4.5.

2.2 Bayesian Learning of Bayesian Networks

A Bayesian network is a DAG G that encodes a joint probability distribution over a set $X = \{X_1, \ldots, X_n\}$ of random variables with each node of the graph representing a variable in X. For convenience we will typically work on the index set $V = \{1, \ldots, n\}$ and represent a variable X_i by its index i. We use $X_{Pa_i} \subseteq X$ to represent the set of parents of X_i in a DAG G and use $Pa_i \subseteq V$ to represent the corresponding index set.

Assume we are given a training data set $D = \{x^1, x^2, \ldots, x^N\}$, where each x^i is a particular instantiation over the set of variables X. We only consider situations where the data are complete, that is, every variable in X is assigned a value. In the Bayesian approach to learn Bayesian networks from the training data D, we compute the posterior probability of a network G as

$$P(G|D) = \frac{P(D|G)P(G)}{P(D)}. \tag{2.1}$$

We can then compute the posterior probability of any hypothesis of interest by averaging over all possible networks. In this paper, we are interested in computing the posteriors of structural features. Let f be a structural feature represented by an indicator function such that $f(G)$ is 1 if the feature is present

in G and 0 otherwise. We have

$$P(f|D) = \sum_G f(G)P(G|D). \tag{2.2}$$

Assuming global and local parameter independence, and parameter modularity, $P(D|G)$ can be decomposed into a product of local marginal likelihood often called local scores as (Cooper and Herskovits, 1992; Heckerman et al., 1995)

$$P(D|G) = \prod_{i=1}^{n} P(x_i|x_{Pa_i} : D) \equiv \prod_{i=1}^{n} score_i(Pa_i : D), \tag{2.3}$$

where, with appropriate parameter priors, $score_i(Pa_i : D)$ has a closed form solution. In this paper we will assume that these local scores can be computed efficiently from data.

For the prior $P(G)$ over all possible DAG structures we will assume structure modularity (Friedman and Koller, 2003):

$$P(G) = \prod_{i=1}^{n} Q_i(Pa_i). \tag{2.4}$$

In this paper we consider modular features:

$$f(G) = \prod_{i=1}^{n} f_i(Pa_i), \tag{2.5}$$

where $f_i(Pa_i)$ is an indicator function with values either 0 or 1. For example, an edge $j \to i$ can be represented by setting $f_i(Pa_i) = 1$ if and only if $j \in Pa_i$ and setting $f_l(Pa_l) = 1$ for all $l \neq i$. In this paper, we are interested in computing the posterior $P(f|D)$ of the feature, which can be obtained by computing the joint probability $P(f, D)$ as

$$\begin{aligned} P(f, D) &= \sum_G f(G)P(D|G)P(G) && (2.6)\\ &= \sum_G \prod_{i=1}^{n} f_i(Pa_i)Q_i(Pa_i)score_i(Pa_i : D) \\ &= \sum_G \prod_{i=1}^{n} B_i(Pa_i), && (2.7) \end{aligned}$$

where for all $Pa_i \subseteq V - \{i\}$ we define

$$B_i(Pa_i) \equiv f_i(Pa_i)Q_i(Pa_i)score_i(Pa_i : D). \tag{2.8}$$

It is clear from Eq. (2.6) that if we set all features $f_i(Pa_i)$ to be constant 1 then we have $P(f=1,D) = P(D)$. Therefore we can compute the posterior $P(f|D)$ if we know how to compute the joint $P(f,D)$. In the next section, we show how the summation in Eq. (2.7) can be done by dynamic programming in time complexity $O(3^n)$.

2.3 Computing Posteriors of Features

Every DAG must have a root node, that is, a node with no parents. Let $\mathcal{G}$ denote the set of all DAGs over V, and for any $S \subseteq V$ let $\mathcal{G}^+(S)$ be the set of DAGs over V such that all variables in $V-S$ are root nodes (setting $\mathcal{G}^+(V) = \mathcal{G}$). Since every DAG must have a root node we have $\mathcal{G} = \cup_{j\in V}\mathcal{G}^+(V-\{j\})$. We can compute the summation over all the possible DAGs in Eq. (2.7) by summing over the DAGs in $\mathcal{G}^+(V-\{j\})$ separately. However, there are overlaps between the set $\mathcal{G}^+(V-\{j_1\})$ and the set $\mathcal{G}^+(V-\{j_2\})$ for any $j_1 \neq j_2$ in V; and in fact $\cap_{j\in T}\mathcal{G}^+(V-\{j\}) = \mathcal{G}^+(V-T)$. We can correct for those overlaps using the inclusion-exclusion principle. Define the following function for all $S \subseteq V$

$$RR(S) \equiv \sum_{G\in\mathcal{G}^+(S)} \prod_{i\in S} B_i(Pa_i). \tag{2.9}$$

We have $P(f,D) = RR(V)$ since $\mathcal{G}^+(V) = \mathcal{G}$. Then by the weighted inclusion-exclusion principle, Eq. (2.7) becomes

$$\begin{aligned}
P(f,D) = RR(V) &= \sum_{G\in\mathcal{G}} \prod_{i=1}^{n} B_i(Pa_i) \\
&= \sum_{k=1}^{|V|} (-1)^{k+1} \sum_{\substack{T\subseteq V \\ |T|=k}} \sum_{G\in\mathcal{G}^+(V-T)} \prod_{i=1}^{n} B_i(Pa_i) \\
&= \sum_{k=1}^{|V|} (-1)^{k+1} \sum_{\substack{T\subseteq V \\ |T|=k}} \prod_{j\in T} B_j(\emptyset) \sum_{G\in\mathcal{G}^+(V-T)} \prod_{i\in V-T} B_i(Pa_i) \\
&= \sum_{k=1}^{|V|} (-1)^{k+1} \sum_{\substack{T\subseteq V \\ |T|=k}} \prod_{j\in T} B_j(\emptyset) RR(V-T),
\end{aligned} \tag{2.10}$$

which says that $P(f,D)$ can be computed in terms of $RR(S)$. Next we show that $RR(S)$ for all $S \subseteq V$ can be computed recursively. We define the following function for each $i \in V$ and all $S \subseteq V - \{i\}$

$$A_i(S) \equiv \sum_{Pa_i\subseteq S} B_i(Pa_i), \tag{2.11}$$

where in particular $A_i(\emptyset) = B_i(\emptyset)$. We have the following results, which roughly correspond to the backward computation in (Koivisto, 2006).

Proposition 1

$$P(f, D) = RR(V), \tag{2.12}$$

where $RR(S)$ can be computed recursively by

$$RR(S) = \sum_{k=1}^{|S|} (-1)^{k+1} \sum_{\substack{T \subseteq S \\ |T|=k}} RR(S - T) \prod_{j \in T} A_j(V - S) \tag{2.13}$$

with the base case $RR(\emptyset) = 1$.

Proof: We will say a node $j \in S$ is a source node in $G \in \mathcal{G}^+(S)$ if none of j's parents are in S, that is, $Pa_j \cap S = \emptyset$. For $T \subseteq S$ let $\mathcal{G}^+(S, T)$ denote the set of DAGs in $\mathcal{G}^+(S)$ such that all the variables in T are source nodes (setting $\mathcal{G}^+(S, \emptyset) = \mathcal{G}^+(S)$). It is clear that $\mathcal{G}^+(S) = \cup_{j \in S} \mathcal{G}^+(S, \{j\})$, and that $\cap_{j \in T} \mathcal{G}^+(S, \{j\}) = \mathcal{G}^+(S, T)$. The summation over the DAGs in $\mathcal{G}^+(S)$ in Eq. (2.9) can be computed by summing over the DAGs in $\mathcal{G}^+(S, \{j\})$ separately and correcting for the overlapped DAGs. Define

$$RF(S, T) \equiv \sum_{G \in \mathcal{G}^+(S,T)} \prod_{i \in S} B_i(Pa_i), \quad \text{for any } T \subseteq S, \tag{2.14}$$

where $RF(S, \emptyset) = RR(S)$. Then by the weighted inclusion-exclusion principle, $RR(S)$ in Eq. (2.9) can be computed as

$$RR(S) = \sum_{k=1}^{|S|} (-1)^{k+1} \sum_{T \subseteq S, |T|=k} RF(S, T). \tag{2.15}$$

$RR(S)$ and $RF(S, T)$ can be computed recursively as follows. For $T = \{j\} \subseteq S$,

$$\begin{aligned} RF(S, \{j\}) &= \sum_{G \in \mathcal{G}^+(S,\{j\})} B_j(Pa_j) \prod_{i \in S - \{j\}} B_i(Pa_i) \\ &= [\sum_{Pa_j \subseteq V - S} B_j(Pa_j)][\sum_{G \in \mathcal{G}^+(S - \{j\})} \prod_{i \in S - \{j\}} B_i(Pa_i)] \\ &\quad \text{(see Figure 2.1(a))} \\ &= A_j(V - S) RR(S - \{j\}). \end{aligned} \tag{2.16}$$

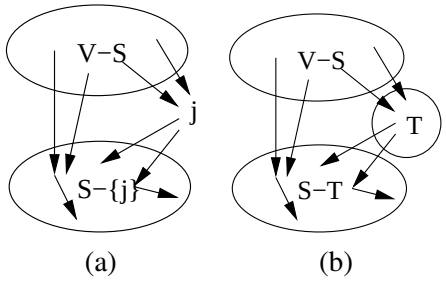


Figure 2.1: Figures Illustrating the Proof of Proposition 1

Similarly, for any $T \subseteq S$,

$$RF(S,T) = \sum_{G \in \mathcal{G}^+(S,T)} \prod_{j \in T} B_j(Pa_j) \prod_{i \in S-T} B_i(Pa_i)$$

$$= \prod_{j \in T} [\sum_{Pa_j \subseteq V-S} B_j(Pa_j)][\sum_{G \in \mathcal{G}^+(S-T)} \prod_{i \in S-T} B_i(Pa_i)]$$

(see Figure 2.1(b))

$$= \prod_{j \in T} A_j(V-S) \cdot RR(S-T). \tag{2.17}$$

Combing Eqs. (2.15) and (2.17) we obtain Eq. (2.13).

□

Based on Proposition 1, provided that the functions A_j have been computed, $P(f, D)$ can be computed in the manner of dynamic programming, starting from the base case $RR(\emptyset) = 1$, then $RR(\{j\}) = A_j(V - \{j\})$, and so on, until $RR(V)$.

Given the functions B_i, the functions A_i as defined in Eq. (2.11) can be computed using the fast Möbius transform algorithm in time $O(n2^n)$ (for a fixed i) (Kennes and Smets, 1991). In the case of a fixed in-degree bound k, $B_i(Pa_i)$ is zero when Pa_i contains more than k elements. Then $A_i(S)$ for all $S \subseteq V - \{i\}$ can be computed more efficiently using the truncated Möbius transform algorithm given in (Koivisto and Sood, 2004) in time $O(k2^n)$ (for a fixed i).

The functions RR may be computed more efficiently if we precompute the product of A_j. Define

$$AA(S,T) \equiv \prod_{j \in T} A_j(V-S) \quad \text{for } T \subseteq S \subseteq V. \tag{2.18}$$

Then using Eq. (2.18) for $AA(S, T - \{j\})$ we have

$$AA(S,T) = A_j(V - S)AA(S, T - \{j\}) \quad \text{for any } \ j \in T. \tag{2.19}$$

For a fixed S, $AA(S,T)$ for all $T \subseteq S$ can be computed in the manner of dynamic programming in $O(2^{|S|})$ time starting with $AA(S, \{j\}) = A_j(V - S)$. We then compute $RR(S)$ using

$$RR(S) = \sum_{k=1}^{|S|}(-1)^{k+1} \sum_{\substack{T \subseteq S \\ |T|=k}} RR(S - T)AA(S,T) \tag{2.20}$$

in $O(2^{|S|})$ time. The functions $RR(S)$ for all $S \subseteq V$ can be computed in $O(\sum_{k=0}^{n} \binom{n}{k} 2^k) = O(3^n)$ time.

In summary, we obtain the following results.

Theorem 1 *Given a fixed maximum in-degree k, $P(f|D)$ can be computed in $O(3^n + kn2^n)$ time.*

2.4 Computing Posterior Probabilities for All Edges

If we want to compute the posterior probabilities of all $O(n^2)$ potential edges, we can compute $RR(V)$ for each edge separately and solve the problem in $O(n^2 3^n)$ total time. However, there is a large overlapping in the computations for different edges. After computing $P(D)$ with constant features $f_i(Pa_i) = 1$ for all $i \in V$, the computation for an edge $i \to j$ only needs to change the function f_j and therefore B_j and A_j. We can take advantage of this overlap and reduce the total time for computing for all edges.

Inspired by the forward-backward algorithm in (Koivisto, 2006), we developed an algorithm that can compute all edge posterior probabilities in $O(n3^n)$ total time. The computations of $P(f, D)$ in Section 2.3 are based exploiting root nodes and roughly correspond to the backward computation in (Koivisto, 2006). Next we first describe a computation of $P(f, D)$ by exploiting sink nodes (nodes that have no outgoing edges) which roughly corresponds to the computation in (Koivisto and Sood, 2004) called forward computation in (Koivisto, 2006). Then we describe how to combine the two computations to reduce the total computation time.

2.4.1 Computing $P(f, D)$ by Exploiting Sinks

For any $S \subseteq V$, let $\mathcal{G}(S)$ denote the set of all the possible DAGs over S, and $\mathcal{G}(S, T)$ denote the set of all the possible DAGs over S such that all the variables in $T \subseteq S$ are sinks. We have $\mathcal{G}(V) = \mathcal{G}$ and $\mathcal{G}(S, \emptyset) = \mathcal{G}(S)$. For any $S \subseteq V$ define

$$H(S) \equiv \sum_{G \in \mathcal{G}(S)} \prod_{i \in S} B_i(Pa_i). \tag{2.21}$$

We have $P(f, D) = H(V)$ since $\mathcal{G}(V) = \mathcal{G}$. As in Section 2.3 we can show that $H(S)$ can be computed recursively. For any $T \subseteq S \subseteq V$, define

$$F(S, T) \equiv \sum_{G \in \mathcal{G}(S,T)} \prod_{i \in S} B_i(Pa_i), \tag{2.22}$$

where $F(S, \emptyset) = H(S)$. We have the following results.

Proposition 2

$$P(f, D) = H(V), \tag{2.23}$$

and $H(S)$ can be computed recursively by

$$H(S) = \sum_{k=1}^{|S|} (-1)^{k+1} \sum_{\substack{T \subseteq S \\ |T|=k}} H(S - T) \prod_{j \in T} A_j(S - T) \tag{2.24}$$

with the base case $H(\emptyset) = 1$.

Proof: Since every DAG has a sink we have $\mathcal{G}(S) = \cup_{j \in S} \mathcal{G}(S, \{j\})$. It is clear that $\cap_{j \in T} \mathcal{G}(S, \{j\}) = \mathcal{G}(S, T)$. The summation over the DAGs in $\mathcal{G}(S)$ in Eq. (2.21) can be computed by summing over the DAGs in $\mathcal{G}(S, \{j\})$ separately and correcting for the overlapped DAGs. By the weighted inclusion-exclusion principle, $H(S)$ in Eq. (2.21) can be computed as

$$H(S) = \sum_{k=1}^{|S|} (-1)^{k+1} \sum_{T \subseteq S, |T|=k} F(S, T). \tag{2.25}$$

$H(S)$ and $F(S,T)$ can be computed recursively as follows. For $T=\{j\}\subseteq S$, we have

$$\begin{aligned}
F(S,\{j\}) &= \sum_{G\in\mathcal{G}(S,\{j\})}\prod_{i\in S} B_i(Pa_i)\\
&= [\sum_{Pa_j\subseteq S-\{j\}} B_j(Pa_j)][\sum_{G\in\mathcal{G}(S-\{j\})}\prod_{i\in S-\{j\}} B_i(Pa_i)]\\
&\quad \text{(see Figure 2.2(a))}\\
&= A_j(S-\{j\})H(S-\{j\}). \qquad (2.26)
\end{aligned}$$

Similarly, for any $j\in T\subseteq S$,

$$\begin{aligned}
F(S,T) &= \sum_{G\in\mathcal{G}(S,T)}\prod_{i\in S} B_i(Pa_i)\\
&= [\sum_{Pa_j\subseteq S-T} B_j(Pa_j)][\sum_{G\in\mathcal{G}(S-\{j\},T-\{j\})}\prod_{i\in S-\{j\}} B_i(Pa_i)]\\
&\quad \text{(see Figure 2.2(b))}\\
&= A_j(S-T)F(S-\{j\},T-\{j\}). \qquad (2.27)
\end{aligned}$$

Let $T=\{j_1,\ldots,j_k\}$. Repeatedly applying Eq. (2.27) and using the fact $(S-T')-(T-T')=S-T$ for any $T'\subseteq T$, we obtain

$$\begin{aligned}
F(S,T) &= A_{j_1}(S-T)A_{j_2}(S-T)F(S-\{j_1,j_2\},T-\{j_1,j_2\})\\
&= \ldots\\
&= A_{j_1}(S-T)\cdots A_{j_{k-1}}(S-T)F(S-\{j_1,\ldots,j_{k-1}\},\{j_k\})\\
&= H(S-T)\prod_{j\in T} A_j(S-T), \qquad (2.28)
\end{aligned}$$

where Eq. (2.26) is applied in the last step. Finally combining Eqs. (2.28) and (2.25) leads to (2.24).

□

Based on Proposition 2, $H(S)$ can be computed in the manner of dynamic programming. Each $H(S)$ is computed in time $O(\sum_{k=1}^{|S|}\binom{|S|}{k}k)=O(|S|2^{|S|-1})$. All $H(S)$ for $S\subseteq V$ can be computed in time $O(\sum_{k=1}^{n}\binom{n}{k}k2^{k-1})=O(n3^{n-1})$. We could store all $F(S,T)$ and compute all $H(S)$ in time $O(3^n)$. But the memory requirement would become $O(3^n+n2^n)$ instead of $O(n2^n)$.

We could compute the posterior of a feature using $P(f,D)=H(V)$ but this is a factor of n slower than computing $RR(V)$. Next we show how to reduce the total time for computing the posterior probabilities of all edges by combining the contributions of $H(S)$ and $RR(S)$.

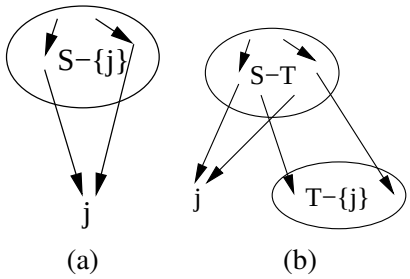


Figure 2.2: Figures Illustrating the Proof of Proposition 2

2.4.2 Computing Posteriors for All Edges

Consider the summation over all the possible DAGs in Eq. (2.7). Assume that we are interested in computing the posterior probability of an edge $i \to v$. We want to extract the contribution of B_v from the rest of B_i. The idea is as follows. For a fixed node v, we can break a DAG into the set of ancestors U of v and the set of nonancestors $V - \{v\} - U$.[1] Roughly speaking, conditioned on U, the summation over all DAGs can be decomposed into the contributions from the summation over DAGs over U, which corresponds to the computation of $H(U)$, and the contributions from the summation over DAGs over $V - \{v\} - U$ with the variables in $U \cup \{v\}$ as root nodes, which corresponds to the computation of $RR(V - \{v\} - U)$.

Define, for any $v \in V$ and $U \subseteq V - \{v\}$ the following function

$$\begin{aligned} & K_v(U) \\ & \equiv \sum_{T \subseteq V - \{v\} - U} (-1)^{|T|} RR(V - \{v\} - U - T) \prod_{j \in T} A_j(U). \end{aligned} \tag{2.29}$$

We have the following results.

Proposition 3

$$P(f, D) = \sum_{U \subseteq V - \{v\}} A_v(U) H(U) K_v(U). \tag{2.30}$$

The proof of Proposition 3 is given in the appendix.

[1] Or we can break a DAG into the set of nondescendants U of v and the set of descendants $V - \{v\} - U$. It could be shown that we can also use this way of breaking DAGs to derive Proposition 3. But this is not exploited in the paper.

Note that in Eq. (2.30) the computations of $H(U)$ and $K_v(U)$ do not rely on B_v and all the contribution from B_v to $P(f, D)$ is represented by A_v. After we have computed the functions A_v, H and K_v, $P(f, D)$ can be computed using Eq. (2.30) in time $O(2^n)$.

To compute $K_v(U)$, we can precompute the product of A_j. Define

$$\eta(U, T) \equiv \prod_{j \in T} A_j(U). \tag{2.31}$$

Then $K_v(U)$ can be computed as

$$K_v(U) = \sum_{T \subseteq V - \{v\} - U} (-1)^{|T|} RR(V - \{v\} - U - T)\eta(U, T), \tag{2.32}$$

where $\eta(U, T)$ can be computed recursively as

$$\eta(U, T) = A_j(U)\eta(U, T - \{j\}) \quad \text{for any } \ j \in T. \tag{2.33}$$

For a fixed U, all $\eta(U, T)$ for $T \subseteq V - \{v\} - U$ can be computed in $O(2^{n-1-|U|})$ time, and then $K_v(U)$ can be computed in $O(2^{n-1-|U|})$ time. For a fixed v all $K_v(U)$ for $U \subseteq V - \{v\}$ can be computed in $O(\sum_{k=0}^{n-1} \binom{n-1}{k} 2^{n-1-k}) = O(3^{n-1})$ time.

Based on Proposition 3, to compute the posterior probabilities for all possible edges, we can use the following algorithm.

1. Precomputation. Set constant feature $f(G) \equiv 1$. Compute functions B_i, A_i, RR, H, and K_i.

2. For each edge $u \to v$:

   (a) For all $S \subseteq V - \{v\}$, recompute $A_v(S)$.

   (b) Compute $P(f, D)$ using Eq. (2.30). Then $P(f|D) = P(f, D)/RR(V)$.

For a fixed maximum in-degree k, Step 1 takes time $O(n3^n)$ as discussed before, and Step 2 takes time $O(n^2(k2^n + 2^n))$. It takes $O(n3^n + kn^2 2^n)$ total time to compute the posterior probabilities for all possible edges.

The computation time of Step 2 can be further reduced by a factor of n using the techniques de-

scribed in (Koivisto, 2006). Plug in the definition of $A_v(U)$ into Eq. (2.30)

$$\begin{aligned} P(f,D) &= \sum_{U\subseteq V-\{v\}} [\sum_{Pa_v\subseteq U} B_v(Pa_v)]H(U)K_v(U) \\ &= \sum_{Pa_v\subseteq V-\{v\}} B_v(Pa_v) \sum_{Pa_v\subseteq U\subseteq V-\{v\}} H(U)K_v(U) \\ &= \sum_{Pa_v\subseteq V-\{v\}} B_v(Pa_v)\Gamma_v(Pa_v), \end{aligned} \tag{2.34}$$

where for all $Pa_v \subseteq V - \{v\}$ we define

$$\Gamma_v(Pa_v) \equiv \sum_{Pa_v\subseteq U\subseteq V-\{v\}} H(U)K_v(U). \tag{2.35}$$

For a fixed maximum in-degree k, since we set $B_v(Pa_v)$ to be zero for Pa_v containing more than k variables, we need to compute the function $\Gamma_v(Pa_v)$ only at sets Pa_v containing at most k elements, which can be computed using the k-truncated downward Möbius transform algorithm described in (Koivisto, 2006) in $O(k2^n)$ time for all Pa_v (for a fixed v). Then $P(f,D)$ for an edge $u \to v$ can be computed as

$$P(u\to v,D) = \sum_{\substack{u\in Pa_v\subseteq V-\{v\} \\ |Pa_v|\le k}} B_v(Pa_v)\Gamma_v(Pa_v), \tag{2.36}$$

which takes $O(n^{k-1})$ time.

In summary, we propose the algorithm in Figure 2.3 to compute the posterior probabilities for all possible edges. The main result of the paper is summarized in the following theorem.

Theorem 2 *For a fixed maximum in-degree k, the posterior probabilities for all $n(n-1)$ possible edges can be computed in $O(n3^n)$ total time.*

2.5 Experimental Results

The algorithm in Figure 2.3 has been implemented in our C++ software tool called POSTER, which is available at http://www.cs.iastate.edu/ rhe/POSTER/; and we run various experiments to demonstrate its capabilities. We compared our implementation with REBEL[2], a C++ language implementation of the DP algorithm in (Koivisto, 2006).

[2]REBEL is available at http://www.cs.helsinki.fi/u/mkhkoivi/REBEL/.

Algorithm Computing posteriors of all edges given maximum in-degree k

1. Precomputation. Set trivial feature $f(G) \equiv 1$
   (a) For all $i \in V$, $Pa_i \subseteq V - \{i\}$ with $|Pa_i| \leq k$, compute $B_i(Pa_i)$. Time complexity $O(n^{k+1})$.
   (b) For all $i \in V$, $S \subseteq V - \{i\}$, compute $A_i(S)$. Time complexity $O(kn2^n)$.
   (c) For all $S \subseteq V$, compute $RR(S)$. Time complexity $O(3^n)$.
   (d) For all $S \subseteq V$, compute $H(S)$. Time complexity $O(n3^n)$.
   (e) For all $i \in V$, $S \subseteq V - \{i\}$, compute $K_i(S)$. Time complexity $O(n3^n)$.
   (f) For all $i \in V$, $Pa_i \subseteq V - \{i\}$ with $|Pa_i| \leq k$, compute $\Gamma_i(Pa_i)$. Time complexity $O(kn2^n)$.
2. For each edge $u \to v$, compute $P(u \to v|D)$ using Eq. (2.36), and output $P(u \to v|D) = P(u \to v, D)/RR(V)$. Time complexity $O(n^{k+1})$.

Figure 2.3: Algorithm for Computing the Posterior Probabilities for All Possible Edges in Time Complexity $O(n3^n)$ Assuming a Fixed Maximum In-degree k

We used BDe score for $score_i(Pa_i : D)$ (with the hyperparameters $\alpha_{x_i;pa_i} = 1/(|Dm(X_i)| \cdot |Dm(Pa_i)|)$) (Heckerman et al., 1995). In the experiments our algorithm used a uniform structure prior $P(G) = 1$ and REBEL used a structure prior specified in (Koivisto, 2006). All the experiments were run under Linux on an ordinary desktop PC with a 3.0GHz Intel Pentium processor and 2.0GB of memory.

2.5.1 Speed Test

We tested our algorithm on several data sets from the UCI Machine Learning Repository (Asuncion and Newman, 2007): Iris, Tic-Tac-Toe, and Zoo. All the data sets contain discrete variables (or are discretized) and have no missing values. We also tested our algorithm on a synthetic data set coming with REBEL. For each data set, we ran our algorithm and REBEL to compute the posterior probabilities for all potential edges. The time taken under different maximum in-degree k is reported in Table 3.1, which also lists the number of variables n and the number of instances m for each data set. We also show the time T_B for computing all the local scores in Table 3.1 as this time also depends on the number of instances m in a data set.

The results demonstrate that our algorithm is capable of computing the posterior probabilities for

Table 2.1: Speed of the Algorithm for Computing Posteriors of All Edges (Time Is in Seconds)

Name	n	m	k	T_B	Ours	REBEL
Iris	5	150	4	2.2e-3	3.5e-3	3.1e-3
TicTacToe	10	958	4	4.7e-1	6.2e-1	5.1e-1
			5	9.1e-1	1.1	9.4e-1
			6	1.3	1.5	1.4
			9	1.7	1.9	1.7
Zoo	17	101	4	1.4	602.3	13.4
			5	4.4	607.0	19.2
			6	11.5	610.6	28.7
Synthetic	20	500	4	9.2	23083	128.3

all the potential edges in networks over around $n = 20$ variables. The memory requirement of the algorithm is $O(n2^n)$, the same as REBEL, which will limit the use of the algorithm to about $n = 25$ variables. It may take our current implementation a few months for $n = 25$.

2.5.2 Comparison of Computations

For the Tic-Tac-Toe data set with $n = 10$, our algorithm is capable of computing the "true" exact edge posterior probabilities by setting the maximum in-degree $k = 9$,[3] although an exhaustive enumeration of DAGs with $n = 10$ would not be feasible. We then vary the maximum in-degree k and compare the edge posterior probabilities computed by our algorithm with the true probabilities. The results are shown as scatter plots in Figure 3.1 (Note that in these graphs most of the points are located at (0,0) or closely nearby). Each point in a scatter plot corresponds to an edge with its x and y coordinates denoting the posterior computed by the two compared algorithms. We see that with the increase of k the computed probabilities gradually approach the true probabilities. With $k = 3$ the computed probabilities already converge to the true probabilities. Studying the effects of the approximation due to the maximum in-degree restriction in general need more substantial experiments and is beyond the scope of this paper.

We also compared the exact posterior probabilities computed by REBEL (setting $k = 9$) with the true probabilities. The results are shown in Figure 3.2. We see that the exact probabilities computed

[3]We will call the exact posterior probabilities computed using uniform structure prior $P(G) = 1$ the "true" probabilities.

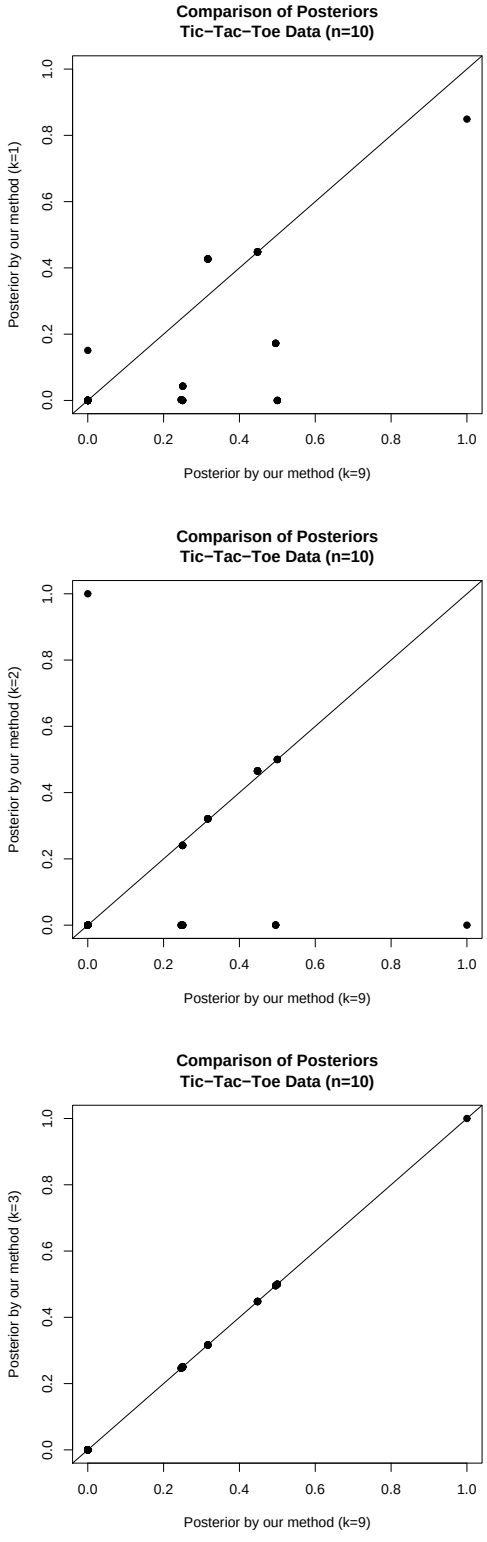


Figure 2.4: Scatter Plots that Compare Posterior Probability of Edges on the Tic-Tac-Toe Data Set as Computed by Our Algorithm with Different k against the True Posterior

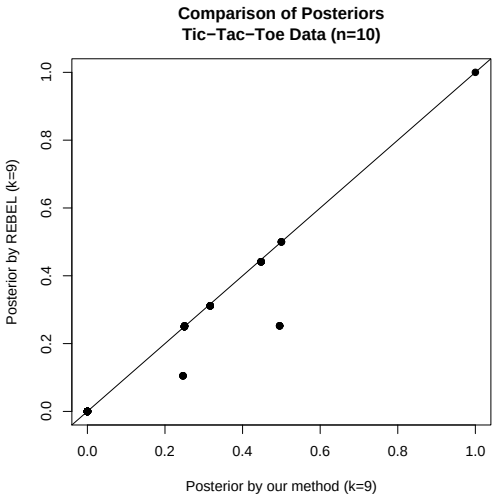


Figure 2.5: Scatter Plot that Compares Posterior Probability of Edges on the Tic-Tac-Toe Data Set as Computed by REBEL against the True Posterior

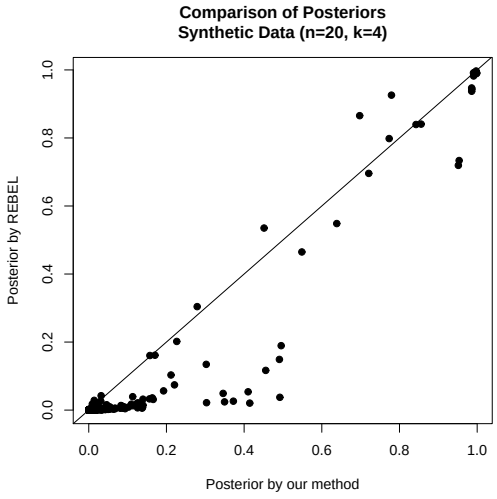


Figure 2.6: Scatter Plot that Compares Posterior Probability of Edges on the Synthetic Data Set as Computed by REBEL and Our Algorithm

by REBEL without in-degree bound sometimes still differ with the true probabilities. This is due to the highly non uniform structure prior used by REBEL.

We compared our algorithm with REBEL over a larger network, the synthetic data set with $n = 20$. The results are shown in Figure 2.6. We see that with the same maximum in-degree, the computed probabilities often differ. Again, this can be attributed to the non uniform structure prior used by REBEL.

2.6 Conclusion

We have presented an algorithm that can compute the exact posterior probability of a modular feature such as a single edge in $O(3^n)$ time and the exact posterior probabilities for all $n(n-1)$ potential edges in $O(n3^n)$ total time. We demonstrated its capability on data sets containing up to 20 variables.

The main advantage of our algorithm over the current state-of-the-art algorithms, the DP algorithm in (Koivisto, 2006) and the order MCMC in (Friedman and Koller, 2003), for computing the posterior probabilities of structural features is that those algorithms require special structure prior $P(G)$ that is highly non uniform while we allow general prior $P(G)$.

Our algorithm computes exact posterior probabilities and works in moderate size networks (about 20 variables), which make it a useful tool for studying several problems in learning Bayesian networks. One application is to assess the quality of the DP algorithm due to the influence of non uniform prior $P(G)$. Another application is to study the effects of the approximation due to the maximum in-degree restriction. We have shown some initial experimental results in Section 4.4. Other potential applications include assessing the quality of approximate algorithms (e.g., MCMC algorithms), studying the effects of data sample size on the learning results, and studying the effects of model parameters (such as parameter priors) on the learning results.

2.7 Appendix: Proof of Proposition 3

For a fixed node v, we can break a DAG uniquely into the set of ancestors S of v and the set of nonancestors $V - \{v\} - S$. For $v \notin S$ let $\mathcal{G}^v(S)$ denote the set of DAGs over $S \cup \{v\}$ such that every node in S is an ancestor of v. Then the summation over all possible DAGs in Eq. (2.7) can be

decomposed into

$$P(f,D) = \sum_{S\subseteq V-\{v\}} [\sum_{G\in\mathcal{G}^v(S)} \prod_{i\in S\cup\{v\}} B_i(Pa_i)] \cdot[\sum_{G\in\mathcal{G}^+(V-\{v\}-S)} \prod_{i\in V-\{v\}-S} B_i(Pa_i)] = \sum_{S\subseteq V-\{v\}} LL_v(S)RR(V-\{v\}-S), \tag{2.37}$$

where for any $S \subseteq V - \{v\}$ we define

$$LL_v(S) \equiv \sum_{G\in\mathcal{G}^v(S)} \prod_{i\in S\cup\{v\}} B_i(Pa_i). \tag{2.38}$$

$\mathcal{G}^v(S)$ consists of the set of DAGs over $S \cup \{v\}$ in which v is the unique sink. We have

$$\mathcal{G}(S\cup\{v\},\{v\}) = \mathcal{G}^v(S) \cup (\cup_{j\in S}\mathcal{G}(S\cup\{v\},\{v,j\})), \tag{2.39}$$

from which, by the weighted inclusion-exclusion principle, we obtain

$$\begin{aligned} LL_v(S) &= \sum_{G\in\mathcal{G}(S\cup\{v\},\{v\})} \prod_{i\in S\cup\{v\}} B_i(Pa_i) \\ &\quad - \sum_{k=1}^{|S|}(-1)^{k+1} \sum_{\substack{T\subseteq S\\|T|=k}} \sum_{G\in\mathcal{G}(S\cup\{v\},T\cup\{v\})} \prod_{i\in S\cup\{v\}} B_i(Pa_i) \\ &= F(S\cup\{v\},\{v\}) - \sum_{k=1}^{|S|}(-1)^{k+1} \sum_{\substack{T\subseteq S\\|T|=k}} F(S\cup\{v\},T\cup\{v\}) \\ &= \sum_{T\subseteq S}(-1)^{|T|}F(S\cup\{v\},T\cup\{v\}) \\ &= \sum_{T\subseteq S}(-1)^{|T|}A_v(S-T)F(S,T) \\ &= \sum_{U\subseteq S}(-1)^{|S|-|U|}A_v(U)F(S,S-U). \end{aligned} \tag{2.40}$$

Plugging Eq. (2.40) into Eq. (2.37), we obtain

$$\begin{aligned}
P(f,D) &= \sum_{S\subseteq V-\{v\}} \sum_{U\subseteq S} (-1)^{|S|-|U|} A_v(U) \\
&\qquad \cdot F(S, S-U) RR(V-\{v\}-S) \\
&= \sum_{U\subseteq V-\{v\}} \sum_{U\subseteq S\subseteq V-\{v\}} (-1)^{|S|-|U|} A_v(U) \\
&\qquad \cdot F(S, S-U) RR(V-\{v\}-S) \\
&= \sum_{U\subseteq V-\{v\}} A_v(U) H(U) \sum_{U\subseteq S\subseteq V-\{v\}} (-1)^{|S|-|U|} \\
&\qquad \cdot \prod_{j\in S-U} A_j(U) RR(V-\{v\}-S) \\
&= \sum_{U\subseteq V-\{v\}} A_v(U) H(U) K_v(U), \qquad (2.41)
\end{aligned}$$

where we have used the definition of function $K_v(U)$ in Eq. (2.29).

CHAPTER 3. BAYESIAN MODEL AVERAGING USING THE K-BEST BAYESIAN NETWORK STRUCTURES

A paper published in *Proceedings of the* 26*th Conference on Uncertainty in Artificial Intelligence*

Jin Tian, Ru He and Lavanya Ram

Abstract

We study the problem of learning Bayesian network structures from data. We develop an algorithm for finding the k-best Bayesian network structures. We propose to compute the posterior probabilities of hypotheses of interest by Bayesian model averaging over the k-best Bayesian networks. We present empirical results on structural discovery over several real and synthetic data sets and show that the method outperforms the model selection method and the state-of-the-art MCMC methods.

3.1 Introduction

Bayesian networks (BN) are being widely used in various data mining tasks for probabilistic inference and causal modeling (Pearl, 2000; Spirtes et al., 2001). One major challenge in the applications of BN is to learn the structures of BNs from data. In the Bayesian approach, we provide a prior probability distribution over the space of possible Bayesian networks and then compute the posterior distributions $P(G|D)$ of the network structure G given data D. We can then compute the posterior probability of any hypothesis of interest by averaging over all possible networks. In some applications we are interested in structural features. For example, in causal discovery, we are interested in the causal relations among variables, represented by the edges in the network structure (Heckerman et al., 1999). In other applications we are interested in predicting the posterior probabilities of new observations, for example, in classification tasks.

The number of possible network structures is super-exponential $O(n!2^{n(n-1)/2})$ in the number of variables n. For example, there are about 10^4 directed acyclic graphs (DAGs) on 5 nodes, and 10^{18} DAGs on 10 nodes. As a result, it is impractical to sum over all possible structures unless for tiny domains (less than 8 variables). The most common solution is to use model selection approach in which we use $P(D, G) = P(G|D)P(D)$ or other measures as a scoring metric and we attempt to find a single network with the best score, the MAP network. We then use that model (or its Markov equivalence class) to make future predictions. This may be a good approximation if the amount of data is large relative to the size of the model such that the posterior is sharply peaked around the MAP model. However, in domains where the amount of data is small relative to the size of the model there are often many high-scoring models with non-negligible posterior. In this situation using a single model could lead to unwarranted conclusions about the structure features and also poor predictions about new observations. For example, the edges that appear in the MAP model do not necessarily appear in other approximately equally likely models. Also the model selection may be sensitive to the data samples given in the sense that a different set of data (from the same distribution) might well lead to a different MAP model. In such cases, using Bayesian model averaging is preferred.

Recently there has been progress on computing exact posterior probabilities of structural features such as edges or subnetworks using dynamic programming techniques (Koivisto and Sood, 2004; Koivisto, 2006; Tian and He, 2009). These techniques have exponential time and memory complexity and are capable of handling data sets with up to around 20 variables. One problem with these algorithms is that they can only computer posteriors over modular features such as directed edges but can not compute non-modular features such as paths ("is there a path between nodes X and Y"). Another problem is that it is very expensive to perform data prediction tasks. They can compute the exact posterior of new observational data $P(x|D)$ but the algorithms have to be re-run for each new data case x.

When computing exact posterior probabilities of features is not feasible, one solution that has been proposed is to approximate full Bayesian model averaging by finding a set of high-scoring networks and making prediction using these models (Madigan and Raftery, 1994; Heckerman et al., 1999). This leaves open the question of how to construct the set of representative models. One possible approach is to use the bootstrap technique which has been studied in (Friedman et al., 1999). However there are

still many questions that need further study on how to use the bootstrap to approximate the Bayesian posterior.

One theoretically well-founded approach is to use Markov Chain Monte Carlo (MCMC) techniques. Madigan and York (1995) used MCMC algorithm in the space of network structures (i.e., DAGs). Friedman and Koller (2003) developed a MCMC procedure in the space of node orderings which was shown to be more efficient than MCMC in the space of DAGs and to outperform the bootstrap approach in (Friedman et al., 1999) as well. Eaton and Murphy (2007) developed a hybrid MCMC method (DP+MCMC) that first uses the dynamic programming technique in (Koivisto, 2006) to develop a global proposal distribution and then runs MCMC in the DAG space. Their experiments showed that the DP+MCMC algorithm converged faster than previous two methods (Madigan and York, 1995; Friedman and Koller, 2003) and resulted in more accurate structure learning. One common problem to the MCMC and bootstrap approach is that there is no guarantee on the quality of the approximation in finite runs.

Madigan and Raftery (1994) has proposed to discard the models whose posterior probability is much lower than the best ones (as well as complex models whose posterior probability is lower than some simpler one). In this paper, we study the approach of approximating Bayesian model averaging using a set of best Bayesian networks. It is intuitive to make predictions using a set of the best models, and we believe it is due to the computational difficulties of actually finding the best networks that this idea has not been systematically studied. In this paper, we develop an algorithm for finding the k-best network structures by generalizing the dynamic programming algorithm for finding the optimal Bayesian network structure (i.e. the MAP network structure) in (Singh and Moore, 2005; Silander and Myllymaki, 2006). We demonstrate the algorithm on several real data sets from the UCI Machine Learning Repository (Asuncion and Newman, 2007) and synthetic data sets from a gold-standard network. We then empirically study the quality of Bayesian model averaging using the k-best networks in structure discovery and show that the method outperforms the model selection method and the state-of-the-art MCMC methods.

3.2 Bayesian Learning of Bayesian Networks

A Bayesian network is a DAG G that encodes a joint probability distribution over a set $X = \{X_1, \ldots, X_n\}$ of random variables with each node of the graph representing a variable in X. For convenience we will typically work on the index set $V = \{1, \ldots, n\}$ and represent a variable X_i by its index i. We use $X_{Pa_i} \subseteq X$ to represent the set of parents of X_i in a DAG G and use $Pa_i \subseteq V$ to represent the corresponding index set.

In the problem of learning BNs from data we are given a training data set $D = \{x^1, x^2, \ldots, x^N\}$, where each x^i is a particular instantiation over the set of variables X. In this paper we only consider situations where the data are complete, that is, every variable in X is assigned a value. In the Bayesian approach to learning Bayesian networks from the training data D, we compute the posterior probability of a network G as

$$P(G|D) = \frac{P(D|G)P(G)}{P(D)}. \tag{3.1}$$

Assuming global/local parameter independence, and parameter modularity, $\ln P(D|G)$ can be decomposed into a summation of so-called (log) local scores as (Cooper and Herskovits, 1992; Heckerman et al., 1995)

$$\ln P(D|G) = \sum_{i=1}^{n} score_i(Pa_i : D) \equiv score(G : D), \tag{3.2}$$

where, with appropriate parameter priors, $score_i(Pa_i : D)$ has a closed form solution. In this paper we will focus on discrete random variables assuming that each variable X_i can take values from a finite domain. We will use the popular BDe score for $score_i(Pa_i : D)$ and we refer to (Heckerman et al., 1995) for its detailed expression. Often for convenience we will omit mentioning D explicitly and use $score_i(Pa_i)$ and $score(G)$. Then with the additional assumption of structure modularity $P(G)$ $= \prod_{i=1}^{n} Q_i(Pa_i)$, $\ln[P(D|G)P(G)]$ can be decomposed as $\sum_{i=1}^{n}[score_i(Pa_i : D) + \ln Q_i(Pa_i)]$.

In the Bayesian framework, we compute the posterior probability of any hypothesis of interest h by averaging over all possible networks.

$$P(h|D) = \sum_{G} P(h|G, D)P(G|D). \tag{3.3}$$

Since the number of possible DAGs is super-exponential in the number of variables n, it is impractical to sum over all DAGs unless for very small networks. One solution is to approximate this exhaustive

enumeration by using a selected set of networks, denoted by $\mathcal{G}$,

$$\hat{P}(h|D) = \frac{\sum_{G\in\mathcal{G}} P(h|G,D)P(G|D)}{\sum_{G\in\mathcal{G}} P(G|D)} \tag{3.4}$$

$$= \frac{\sum_{G\in\mathcal{G}} P(h|G,D)P(G,D)}{\sum_{G\in\mathcal{G}} P(G,D)}, \tag{3.5}$$

where $\hat{P}(.)$ denote approximated probabilities. In the model selection approach, we find a high-scoring model G_s and use it to make predictions:

$$\hat{P}(h|D) = P(h|G_s, D). \tag{3.6}$$

In this paper we will perform model averaging using the set $\mathcal{G}$ of k-best networks.

We can estimate the posterior probability of a network $G \in \mathcal{G}$ as

$$\hat{P}(G|D) = \frac{P(G,D)}{\sum_{G\in\mathcal{G}} P(G,D)}. \tag{3.7}$$

Since $\sum_{G\in\mathcal{G}} P(G,D)$ is not greater than $P(D)$, the estimate $\hat{P}(G|D)$ is always an upper bound of $P(G|D)$, and it is a good estimate only if $\sum_{G\in\mathcal{G}} P(G|D)$ is close to 1. We can then estimate the posterior probability of hypothesis h by

$$\hat{P}(h|D) = \sum_{G\in\mathcal{G}} P(h|G,D)\hat{P}(G|D). \tag{3.8}$$

When we are interested in computing the posteriors of structural features such as edges, paths, Markov Blankets, etc., we let f denote a structural feature represented by an indicator function such that $f(G)$ is 1 if the feature is present in G and 0 otherwise. We have $P(f|G,D) = f(G)$ and

$$\hat{P}(f|D) = \sum_{G\in\mathcal{G}} f(G)\hat{P}(G|D). \tag{3.9}$$

When we are interested in predicting the posteriors of future observations, we let D^T denote a set of new data examples sampled independently of D (i.e., D and D^T are independent and identically distributed). Then

$$\hat{P}(D^T|D) = \sum_{G\in\mathcal{G}} P(D^T|G,D)\hat{P}(G|D), \tag{3.10}$$

and

$$\ln P(D^T|G,D) = \ln[P(D^T,D|G)/P(D|G)] \tag{3.11}$$

$$= score(G : D^T, D) - score(G : D). \tag{3.12}$$

3.3 Finding the k-best Network Structures

We find the k-best structures using the dynamic programming techniques extending the algorithm for finding the optimal Bayesian network structures in (Silander and Myllymaki, 2006). Our algorithm consists of three steps:

1. Compute the local scores for all possible $n2^{n-1}$ (i, Pa_i) pairs.
2. For each variable $i \in V$, find the k-best parent sets in parent candidate set C for all $C \subseteq V \setminus \{i\}$.
3. Find the k-best networks.

Step 1 is exactly the same as in (Silander and Myllymaki, 2006) and we will use their algorithm. Assuming that we have calculated all the local scores, next we describe how to accomplish Steps 2 and 3 using dynamic programming technique.

3.3.1 Finding the k-best Parent Sets

We can find the k-best parent sets for a variable v from a candidate set C recursively. The k-best parent sets in C for v are the k-best sets among the whole candidate set C itself, and the k-best parent sets for v from each of the smaller candidate sets $\{C \setminus \{c\} | c \in C\}$. Therefore, to compute the k-best parent sets for v for every candidate set $C \subseteq V \setminus \{v\}$, we start with sets of size $|C| = 1$, then consider sets of $|C| = 2$, and so on, until the set $C = V \setminus \{v\}$.

The skeleton algorithm for finding the k-best parent sets for v from a candidate set C is given in Algorithm 1, where we use $bestParents_v[S]$ to denote the k best parent sets for variable v from candidate set S stored in a priority queue, and the operation $Merge(.,.)$ outputs a priority queue of the k best parents given the two input priority queues of k elements. Assuming that the merge operation takes time $T(k)$, finding the k-best parent sets for v from a candidate set C takes time $O(T(k) * |C|)$, and computing for all $C \subseteq V \setminus \{v\}$ takes time $O(\sum_{|C|=1}^{n-1} T(k) * |C| * \binom{n-1}{|C|}) = O(T(k)(n-1)2^{n-2})$.

3.3.2 Finding the k-best Network Structures

Having calculated the k-best parent sets for each variable v from any set C, finding the k-best network structures over a variable set W can be done recursively. We will exploit the fact that every

Algorithm 1 Finding the k-best parent sets for variable v from a candidate set C

Input:
$score_v(C)$: local scores
$bestParents_v[S]$: priority queues of the k-best parent sets for variable v from candidate set S for all $S \subseteq C$ with $|S| = |C| - 1$
Output:
$bestParents_v[C]$: a priority queue of the k-best parents of v from the candidate set C

Initialize $bestParents_v[C]$
for all $S \subseteq C$ such that $|S| = |C| - 1$ **do** $\{$
  $bestParents_v[C] \leftarrow Merge(bestParents_v[C], bestParents_v[S])$
$\}$
Insert C into $bestParents_v[C]$ if $score_v(C)$ is larger than the minimum score in $bestParents_v[C]$

DAG has a sink, a node that has no outgoing edges. First for each variable $s \in W$, we can find the k-best networks over W with s as a sink. Then the k-best networks over W are the k-best networks among $\{$the k-best networks over W with s as a sink : $s \in W\}$.

The k-best networks over W with s as a sink can be identified by looking at the k-best parent sets for s from the set $W \setminus \{s\}$ and the k-best networks over $W \setminus \{s\}$.[1] More formally, let $bestParents_s[C][i]$ denote the ith best parent set for variable s in the candidate parent set C. Let $bestNets[W][j]$ denote the jth best network over W. Define the function $value(i, j)$ as

$$\begin{aligned} value(i,j) =& score_s(bestParents_s[W \setminus \{s\}][i]) \\ &+ score(bestNets[W \setminus \{s\}][j]). \end{aligned} \tag{3.13}$$

Then the k-best networks over W with s as a sink can be identified by finding the k-best scores among

$$\{value(i,j) : i = 1, \ldots, k, \quad j = 1, \ldots, k.\}. \tag{3.14}$$

This can be done by using a standard best-first graph search algorithm over a search space $\{(i, j) : i = 1, \ldots, k, \quad j = 1, \ldots, k\}$ with root node $(1, 1)$, children of (i, j) being $(i + 1, j)$ and $(i, j + 1)$, and the value of each node given by $value(i, j)$.

The skeleton algorithm for finding the k-best network structures over a set W is given in Algorithm 2. Let the time spent on the best-first search be $T'(k)$. In the worst case all k^2 nodes may need to be

[1] This is because for any networks not constructed out of these parents sets for s and networks over $W \setminus \{s\}$ we can always produce k better networks.

Algorithm 2 Finding the k-best network structures over set W

Input:

$bestParents_i[S]$: priority queues of k-best parent sets for each variable $i \in V$ from any candidate set $S \subseteq V - \{i\}$

$bestNets[S]$: priority queues of k-best network structures over all $S \subseteq W$ with $|S| = |W| - 1$

Output:

$bestNets[W]$: a priority queue of k-best networks over W

for all $s \in W$ **do** {
  do a best-first graph search over the space $\{(i, j)\}$ **until** $value(i, j) < score(bestNets[W][k])$ {
    Construct a BN G from the network $bestNets[W \setminus \{s\}][j]$ and setting the set $bestParents_s[W \setminus \{s\}][i]$ as the parents of s
    Insert the network G into $bestNets[W]$ if G is not in the queue yet
  }
}

visited. The complexity of finding the k-best network structures is $O(\sum_{|W|=1}^{n} T'(k) * |W| * \binom{n}{|W|})$ $= O(T'(k)n2^{n-1})$.

3.4 Experiments

We used BDe score for $score_i(Pa_i : D)$ with a uniform structure prior $P(G)$ and equivalent sample size 1 (Heckerman et al., 1995). We have implemented our algorithm in C++ language and all the experiments on our method were run under Linux on an ordinary desktop PC with a 3.0GHz Intel Pentium processor and 2.0GB memory.

We tested our algorithm on several data sets from the UCI Machine Learning Repository (Asuncion and Newman, 2007): Iris, Nursery, Tic-Tac-Toe, Zoo and Letter. We also tested our algorithm on synthetic data sets of various sample sizes from a gold-standard 15-variable Bayesian network with known structure and parameters. All the data sets contain discrete variables (or are discretized) and have no missing values.

3.4.1 Performance Evaluation

For each data set, we learned the k-best networks for certain k, then we can estimate the posterior probabilities $\hat{P}(h|D)$ of any hypotheses using Eqs. (4.25) and (3.8). To get an idea on how close the estimation is to the true posteriors we used the algorithm in (Tian and He, 2009) to compute the exact

$P(D)$. We can then evaluate the quality of the posterior estimation as follows. Define the following quantity:

$$\Delta \equiv \frac{\sum_{G \in \mathcal{G}} P(G, D)}{P(D)} = \sum_{G \in \mathcal{G}} P(G|D). \tag{3.15}$$

Δ represents the cumulative true probability mass of the graphs in $\mathcal{G}$. From Eqs. (3.1) and (4.25) we obtain

$$\frac{P(G|D)}{\hat{P}(G|D)} = \Delta. \tag{3.16}$$

Note that $\Delta \leq 1$ and the larger value of Δ means the closer estimation $\hat{P}(G|D)$ to exact $P(G|D)$. In general we have the following results on the quality of estimation $\hat{P}(h|D)$.

Proposition 4

$$-(1-\Delta)\hat{P}(h|D) \leq P(h|D) - \hat{P}(h|D) \leq (1-\Delta)(1-\hat{P}(h|D)), \tag{3.17}$$

or equivalently

$$\Delta\hat{P}(h|D) \leq P(h|D) \leq \Delta\hat{P}(h|D) + 1 - \Delta. \tag{3.18}$$

The proof is given in the appendix.

In practice, in the cases we do not have a large amount of data, Δ may be much smaller than 1 and the bounds in proposition 4 could be too loose to be useful. Therefore, we introduce another measure for the quality of the posterior estimation, the relative ratio of the posterior probability of the MAP network G_{map} to the posterior of the worst network G_{min} of the k best networks (the k-th best network):

$$\lambda \equiv \frac{\hat{P}(G_{map}|D)}{\hat{P}(G_{min}|D)} = \frac{P(G_{map}|D)}{P(G_{min}|D)} \tag{3.19}$$

It has been argued in (Madigan and Raftery, 1994) that we should make predictions using a set of the best models by discarding those models that predict the data far less well even though the very many models with small posterior probabilities may contribute substantially to the sum (such that Δ is much smaller than 1). A cutoff value of $\lambda = 20$ is suggested in (Madigan and Raftery, 1994) by analogy with the 0.05 cutoff for P-values.

3.4.2 Experimental Results on the k-best Networks

We tested our algorithm on several data sets and the experimental results are reported in Table 3.1, which lists for each data set the number of variables n, the number of instances (sample size) m, the value k, the time T_l for computing local scores (the time depends on m but not k), the total running time T_t for finding the k-best networks, and the quality measure Δ and λ.

As expected the values of Δ and λ (as a measure of estimation quality) increases as k increases. We see that Δ value is often too small for Proposition 4 to be useful, however Proposition 4 can indeed provide guarantee on the quality of approximation in nontrivial cases like Nursey and Tic-Tac-Toe data sets which are still too large for exhaustive enumeration of all possible networks. Based on the λ value, the best 100 networks are not enough to get reliable estimation for Zoo and letter data sets. For the synthetic data set, λ increases as the sample size m increases. Based on the λ value, the best 100 networks should give reliable estimation for $m = 3000$ and $m = 5000$.

The exact posterior probabilities of k-best networks for each data set are shown in Figures 3.1 and 3.2. We see that there could be multiple networks having the same posterior probability. For example, the number of best networks sharing the largest posterior probability is as follows: 2 for Nursery case, 76 for Tic-Tac-Toe case, 12 for Letter case, 26 for synthetic data with $m = 200$, and 6 for synthetic data with $m = 5000$. This is mainly due to that BDe scoring criterion has the likelihood-equivalence property, i.e., it assigns the same score to the Bayesian networks within the same independence equivalence class. However, we have found that it is also possible that the networks across the different equivalence classes have the same posterior probability. Take Tic-Tac-Toe case for example, we have found that the 76 best networks actually belong to multiple equivalence classes which have different skeletons. This exceptional case shows that one can not always assert that the networks having the same posterior probability must be within the same equivalence class.

In Table 3.2 we show the difference between the best equivalence class and the second best equivalence class for each case. The second column lists the number of different undirected edges between these 2 best equivalence classes and the third column shows λ (the relative ratio of their posterior probabilities). The result shows that their difference is typically small for each case.

Table 3.1: Experimental Results of the Algorithm for Finding k-best Bayesian Networks

Statistics (time in sec.)							
Name	n	m	T_l	k	T_t	Δ	λ
Iris	5	150	0.3	900	1.3	1.000	2.08e+6
Nursery	9	12960	0.4	100	2.9	1.000	2.07e+16
Tic-Tac-Toe	10	958	0.3	1000	261	0.759	2.17e+4
Zoo	17	101	9	1	22	1.31e-10	1
				10	131	1.24e-09	1.089
				100	5945	1.013e-08	1.516
Letter	17	20000	277	1	290	0.00159	1
				10	380	0.0159	1
				100	5976	0.156	1.022
Synthetic	15	200	4	1	7	1.55e-07	1
				10	30	1.55e-06	1
				100	889	1.27e-05	1.698
Synthetic	15	1000	11	1	14	2.54e-06	1
				10	36	2.37e-05	1.203
				100	884	7.00e-05	12.49
Synthetic	15	3000	17	1	20	4.77e-07	1
				10	41	3.37e-06	1.319
				100	886	3.77e-06	1282
Synthetic	15	5000	21	1	24	2.59e-07	1
				10	45	1.80e-06	4.30
				100	874	2.94e-06	2406

Table 3.2: Difference across 2 Best Equivalence Classes

Name	No. of diff. edges	λ
Iris	2	2.30
Nursery	1	15.32
Tic-Tac-Toe	4	1.00
Zoo	1	1.03
Letter	1	1.02
Synthetic ($m = 200$)	1	1.01
Synthetic ($m = 1000$)	1	1.03
Synthetic ($m = 5000$)	1	4.30

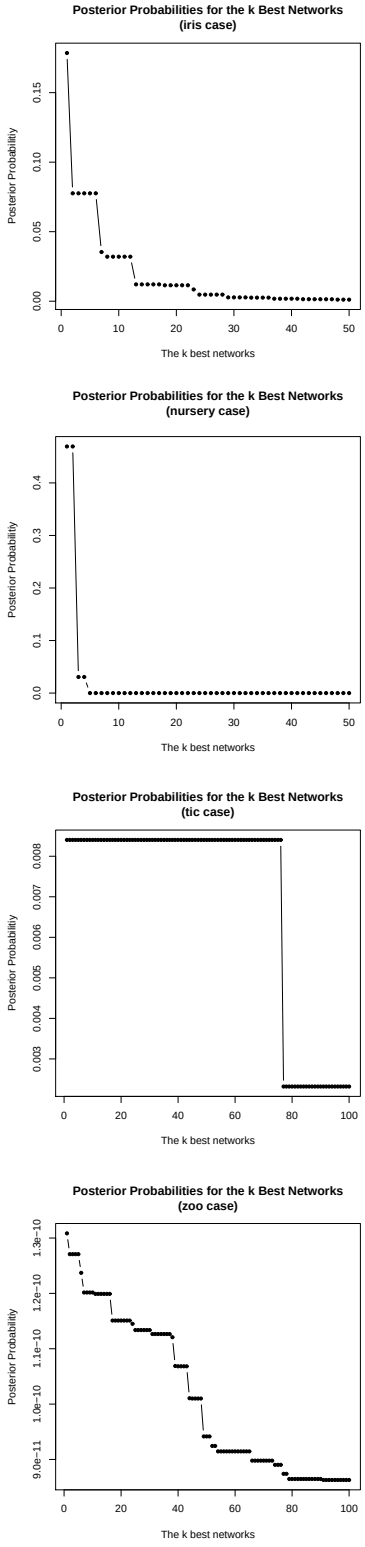


Figure 3.1: Exact Posterior Probabilities of the k-best Networks

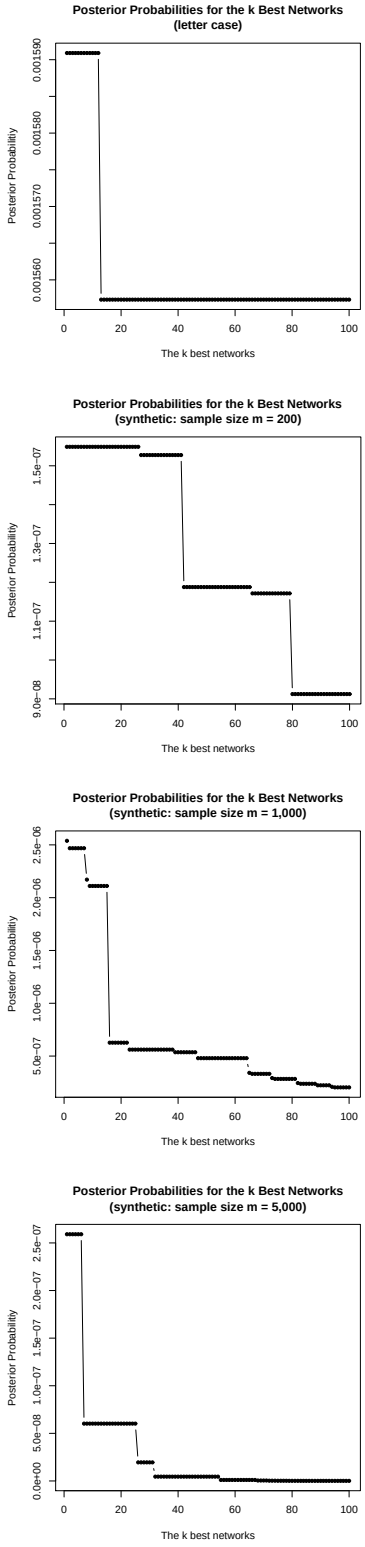


Figure 3.2: Exact Posterior Probabilities of the k-best Networks (Continued)

3.4.3 Structural Discovery

In order to evaluate the ability of the k-best networks method in structural discovery, we tested on the synthetic data set from the gold-standard 15-variable Bayesian network. We computed edge feature between each pair of variables by Eq. (3.9) under the different values of $k \in \{1, 10, 100\}$. For the comparison, we also computed the exact posterior probability of each edge by the method of full model averaging in (Tian and He, 2009). Since we have the true gold-standard network, we could compute all the corresponding ROC curves. The results are shown in Figures 3.3 and 3.4.

The figures indicate the usefulness of k-best method in structural discovery. We observe that the area under ROC (AUC) is a non-decreasing function of k. Even a small increase of k from 1 to 10 will lead to a non-negligible improvement in the corresponding ROC, even though Δ is tiny (such as $2.37e-05$) and λ is small (such as 1.203). For the data set with $m = 5,000$, the performance of Top 100 is almost the same as that of full model averaging method when λ is big $(2,406)$, regardless of the fact that Δ is still tiny $(2.94e-06)$.

The comparison with MCMC approach also demonstrates the usefulness of our method. In this paper we compared our method with the hybrid method (DP+MCMC) proposed by Eaton and Murphy (2007), which was shown to have the statistically significant improvement in structural discovery over other MCMC methods (Madigan and York, 1995; Friedman and Koller, 2003). For DP+MCMC, we set the pure global proposal (with local proposal choice $\beta = 0$) since such a setting was reported to have the best performance (with the largest mean and the smallest variance of AUC) for edge discovery in their experimental results. The tool BDAGL provided by the authors in (Eaton and Murphy, 2007) was used for the experiments on DP+MCMC. We ran totally $120,000$ iterations and discarded the first $100,000$ iterations as burn-in period. Then we set the thinning parameter as 2 to get final $10,000$ samples (networks). Finally the posterior probability of each edge was computed based on the model averaging among these $10,000$ networks and the corresponding ROC was drawn.

Because of the randomness nature of MCMC, we repeated the above process 10 times for each data set.[2] The best, worst and median of these 10 ROCs were shown in the figures, compared with the ROC

[2]The DP step (including marginal likelihood computation) took 221 seconds and MCMC iterations took the mean of 134 seconds in the case of $m = 1,000$. The most part of BDAGL was written in Matlab and we ran BDAGL under Windows XP on an ordinary laptop with 1.60GHz Intel Pentium processor and 1.5GB memory. Due to the different hardware, platforms and programming languages used, the time statistics of DP+MCMC can not be directly compared with the ones of our method.

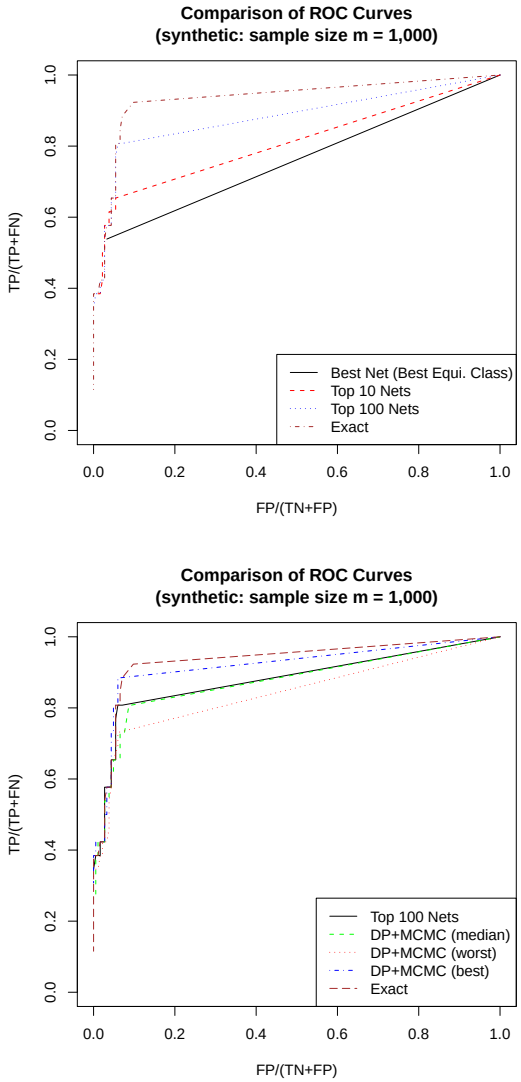


Figure 3.3: Comparison of ROC Curves ($m = 1,000$)

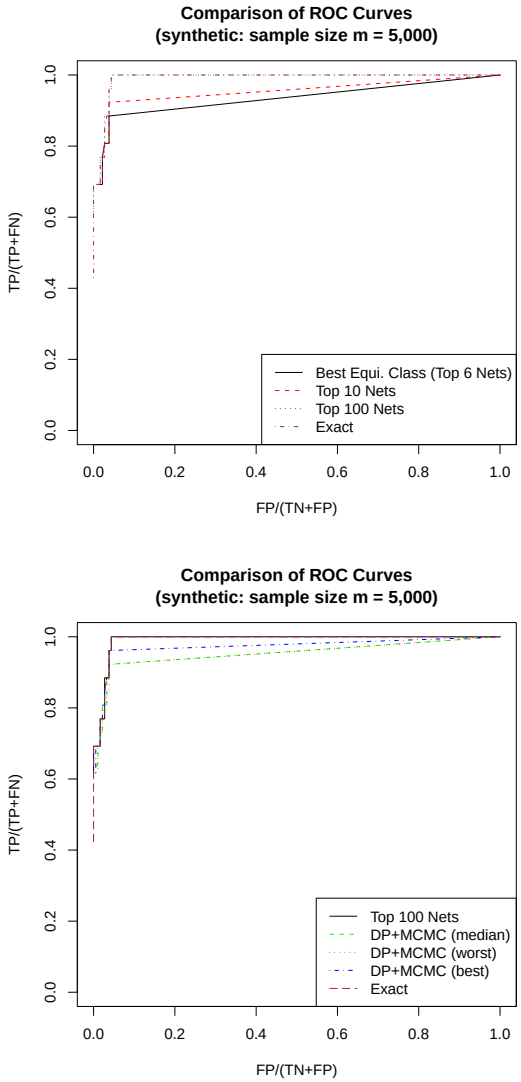


Figure 3.4: Comparison of ROC Curves ($m = 5,000$)

from Top 100 (i.e. the 100 best). In the case of $m = 1,000$, the figure shows that the performance of DP+MCMC still has a non-negligible variability and the performance of Top 100 is no worse than the performance of the median of DP+MCMC. In the case of $m = 5,000$, the variability of the performance of DP+MCMC decreases. However, even the best of DP+MCMC could not outperform Top 100. The good performance of our model averaging over only 100 networks is not surprising: the 100 networks used here are the top 100 networks so that they are relatively more important than all the other networks in the model average process. Figure 3.2 has clearly demonstrated such a relative importance of these top 100 networks.

3.5 Conclusion

We develop an algorithm for finding the k-best Bayesian network structures. We present empirical results on the structural discovery by Bayesian model averaging over the k best Bayesian networks. One nice feature of the method is that we can monotonically improve the estimation accuracy by spending more time to compute for larger k. Another interesting feature shown by our experiments is that we may evaluate the quality of the estimation based on the value of λ. The relation between the estimation quality and λ is worth more substantial study in the future.

As the experimental results show, there are many equivalent networks in the set of best networks. It is desirable if we can directly find the k best equivalence classes. However, the proposed algorithm does not search in the equivalence class space. The algorithm will find the top k individual networks regardless of the existence of equivalent networks. It seems that the dynamic programming idea cannot be naturally generalized to the equivalence class space (at least we were not able to achieve this). How to directly find the k best equivalence classes is a research direction that is worth to pursue.

3.6 Appendix: Proof of Proposition 4

$$
\begin{aligned}
& P(h|D) - \hat{P}(h|D) \\
&= \sum_{G} P(h|G,D)P(G|D) - \sum_{G\in\mathcal{G}} P(h|G,D)\hat{P}(G|D) \\
&= \sum_{G\in\mathcal{G}} P(h|G,D)[P(G|D) - \hat{P}(G|D)] + \sum_{G\notin\mathcal{G}} P(h|G,D)P(G|D) \\
&= \sum_{G\in\mathcal{G}} P(h|G,D)\hat{P}(G|D)(\Delta - 1) + \sum_{G\notin\mathcal{G}} P(h|G,D)P(G|D) \\
&= -(1-\Delta)\hat{P}(h|D) + \sum_{G\notin\mathcal{G}} P(h|G,D)P(G|D) \qquad (3.20)
\end{aligned}
$$

Since the second term in Eq. (3.20) is no less than zero we have proved

$$
P(h|D) - \hat{P}(h|D) \geq -(1-\Delta)\hat{P}(h|D). \qquad (3.21)
$$

Since $P(h|G,D) \leq 1$, from Eq. (3.20) we have

$$
\begin{aligned}
& P(h|D) - \hat{P}(h|D) \\
&\leq -(1-\Delta)\hat{P}(h|D) + \sum_{G\notin\mathcal{G}} P(G|D) \\
&= -(1-\Delta)\hat{P}(h|D) + 1 - \Delta \\
&= (1-\Delta)(1-\hat{P}(h|D)). \qquad (3.22)
\end{aligned}
$$

CHAPTER 4. STRUCTURE LEARNING IN BAYESIAN NETWORKS OF MODERATE SIZE BY EFFICIENT SAMPLING

A paper to be submitted to *The Journal of Machine Learning Research*

Ru He, Jin Tian and Huaiqing Wu

Abstract

We study the Bayesian model averaging approach to learning Bayesian network structures which are directed acyclic graphs (DAGs) from data. We develop new algorithms including the first algorithm that is able to efficiently sample DAGs according to the exact structure posterior. The DAG samples can then be used to construct estimators for the posterior of any feature. We theoretically prove good properties of our estimators and empirically show that our estimators considerably outperform the ones from previous state-of-the-art methods.

4.1 Introduction

Bayesian networks are graphical representations of multivariate joint probability distributions and have been widely used in various data mining tasks for probabilistic inference and causal modeling (Pearl, 2000; Spirtes et al., 2001).

The core of a Bayesian network (BN) representation is its Bayesian network structure. A Bayesian network structure is a DAG (directed acyclic graph) whose nodes represent the random variables $X_1, X_2, \cdots, X_n$ in the problem domain and whose edges correspond to the direct probabilistic dependencies. Semantically, a Bayesian network structure G encodes a set of conditional independence assumptions: for each variable (node) X_i, X_i is conditionally independent of its non-descendants given its parents. With the above semantics, a Bayesian network structure provides a compact representation for joint

distributions and supports efficient algorithms for answering probabilistic queries. Furthermore, with its semantics, a Bayesian network structure can often provide a deep insight into the problem domain and open the door to the cause-and-effect analysis.

In the last two decades, there have been a large number of researches focusing on the problem of learning Bayesian network structure(s) from the data. These researches deal with a common real situation where the underlying Bayesian network is typically unknown so that it has to be learned from the observed data. One motivation for the structure learning is to use the learned structure for inference or decision making. For example, we can use the learned model to predict or classify a new instance of data. Another structure-learning motivation, which is more closely related to the semantics of Bayesian network structures, is for discovering the structure of the problem domain. For example, in the context of biological expression data, the discovery of the causal and dependence relation among different genes is often of primary interests. With the semantics of a Bayesian network structure G, the existence of an edge from node X to node Y in G can be interpreted as the fact that variable X directly influences variable Y; and the existence of a directed path from node X to node Y can be interpreted as the fact that X eventually influences Y. Furthermore, under certain assumptions (Heckerman et al., 1999; Spirtes et al., 2001), the existence of a directed path from node X to node Y indicates that X causes Y. Thus, with the learned Bayesian network structure, we can answer interesting questions such as whether gene X controls gene Y which in turn controls gene Z by examining whether there is a directed path from node X via node Y to node Z in the learned structure. Just as mentioned by Friedman and Koller (2003), the extraction of these kinds of interested structural features is often the primary goal in the discovery task.

There are several general approaches to learning BN structures. One approach is to treat it as a model selection problem. This approach defines a scoring criterion that measures how well a network structure (DAG) fits the data and finds the DAG (or a set of equivalent DAGs) with the optimal score (Silander and Myllymaki, 2006). (In Bayesian approach, the score of a DAG G is simply the posterior $p(G|D)$ of G given data D.) However, when the data size is small relative to the number of variables, the posterior $p(G|D)$ often gives significant support to a number of DAGs, and using a single maximum-a-posteriori (MAP) model could lead to unwarranted conclusions (Friedman and Koller, 2003). It is therefore desirable to use the Bayesian model averaging approach by which the posterior probability of

any feature of interest is computed by averaging over all the possible DAGs (Heckerman et al., 1999).

Bayesian model averaging is, however, computationally challenging because the number of possible network structures is between $2^{n(n-1)/2}$ and $n!2^{n(n-1)/2}$, super-exponential in the number of variables n. Tractable algorithms have been developed for special cases of averaging over trees (Meila and Jaakkola, 2006) and averaging over DAGs given a node ordering (Dash and Cooper, 2004). Since 2004, dynamic programming (DP) algorithms have been developed for computing exact posterior probabilities of structural features such as edges or subnetworks (Koivisto and Sood, 2004; Koivisto, 2006; Tian and He, 2009). These algorithms have exponential time and space complexity and are capable of handling Bayesian networks of moderate size with up to around 25 variables (mainly due to their space cost $O(n2^n)$). A big limitation of these algorithms is that they can only compute posteriors of modular features such as an edge but can not compute non-modular features such as a path ("is there a path from node X to node Y"), a combined path ("is there a path from node X via node Y to node Z" or "is there a path from node X to node Y and no path from node X to node Z"), or a limited-length path ("is there a path of length at most 3 from node X to node Y"). Recently, Parviainen and Koivisto (2011) have developed a DP algorithm that can compute the exact posterior probability of a path feature under a certain assumption. (The assumption, which called order-modular assumption, will be discussed in details soon.) This DP algorithm has (even higher) exponential time and space complexity and can only handle a Bayesian network with fewer than 20 variables (mainly due to its space cost $O(3^n)$). Since this DP algorithm can only deal with a path feature, all the other non-modular features (such as a combined path) which various users would be interested in for their corresponding problems still can not be computed by any DP algorithm proposed so far. Note that generally the posterior $p(f|D)$ of a combined feature $f = (f_1, f_2, \ldots, f_J)$ can not be obtained only from the posterior of each individual feature $p(f_j|D)$ ($j \in \{1, 2, \ldots, J\}$) because the independence among these features does not hold generally. Actually, by comparing $p(f_2|f_1, D)$ with $p(f_2|D)$, a user can know the effect of the feature f_1 upon the feature f_2; but to obtain $p(f_2|f_1, D)$ ($= p(f_1, f_2|D)/p(f_1|D)$), the user typically also needs to obtain $p(f_1, f_2|D)$ first. Another limitation of all these DP algorithms is that it is very expensive for them to perform data prediction tasks. They can compute the exact posterior of a new observational data case $p(x|D)$ but the algorithms have to be re-run for each new data case x.

One solution to computing the posterior of an arbitrary non-modular feature is drawing DAG sam-

ples $\{G_1, \ldots, G_T\}$ from the posterior $p(G|D)$, which can then be used to approximate the full Bayesian model averaging to estimate the posterior of an arbitrary feature f as $p(f|D) \approx \frac{1}{T}\sum_{i=1}^{T} f(G_i)$, or the posterior predictive distribution as $p(x|D) \approx \frac{1}{T}\sum_{i=1}^{T} p(x|G_i)$. A number of algorithms have been developed for drawing sample DAGs using the bootstrap technique (Friedman et al., 1999) or the Markov Chain Monte Carlo (MCMC) techniques (Madigan and York, 1995; Friedman and Koller, 2003; Eaton and Murphy, 2007; Grzegorczyk and Husmeier, 2008; Niinimaki et al., 2011). Madigan and York (1995) developed the Structure MCMC algorithm that uses the Metropolis-Hastings algorithm in the space of DAGs. Friedman and Koller (2003) developed the Order MCMC procedure that operates in the space of orders. The Order MCMC was shown to be able to considerably improve over the Structure MCMC the mixing and convergence of the Markov chain and to outperform the bootstrap approach of Friedman et al. (1999) as well. Eaton and Murphy (2007) developed the Hybrid MCMC method (i.e., DP+MCMC method) that first runs the DP algorithm (Koivisto, 2006) to develop a global proposal distribution and then runs the MCMC phase in the DAG space. Their experiments showed that the Hybrid MCMC converged faster than both the Structure MCMC and the Order MCMC so that the Hybrid MCMC resulted in more accurate structure learning performance. An improved MCMC algorithm (often denoted as REV-MCMC) traversing in the DAG space with the addition of a new edge reversal move was developed by Grzegorczyk and Husmeier (2008) and was shown to be superior to the Structure MCMC and nearly as efficient as the Order MCMC in the mixing and convergence. Recently, Niinimaki et al. (2011) have proposed the Partial Order MCMC method which operates in the space of partial orders. The Partial Order MCMC includes the Order MCMC as its special case (by setting the parameter bucket size b to be 1) and has been shown to be superior to the Order MCMC in terms of the mixing and the structural learning performance when a more appropriate bucket size $b > 1$ is set. One common drawback of these MCMC algorithms is that there is no guarantee on the quality of the approximation in finite runs. The approach to approximating full Bayesian model averaging using the K-best Bayesian network structures was studied by Tian et al. (2010) and was shown to be competitive with the Hybrid MCMC.

Several of these state-of-the-art algorithms work in the order space, including the exact algorithms (Koivisto and Sood, 2004; Koivisto, 2006; Parviainen and Koivisto, 2011) and the approximate algorithms: the Order MCMC (Friedman and Koller, 2003) and the Partial Order MCMC (Niinimaki et al.,

2011). They all assume a special form of the structure prior, termed as order-modular prior (Friedman and Koller, 2003; Koivisto and Sood, 2004), for computational convenience. (Please refer to the beginning of Section 4.2.1 for the definition of the order-modular prior.) However, the assumption of order-modular prior has the consequence that the corresponding prior $p(G)$ cannot represent some desirable priors such as a uniform prior over the DAG space; and the computed posterior probabilities are biased since a DAG that has a larger number of topological orders will be assigned a larger prior probability. Whether a computed posterior with the bias from the order-modular prior is inferior to its counterpart without such a bias depends on the application scenario and is beyond the scope of this paper. For the detailed discussion about this issue, please see the related papers (Friedman and Koller, 2003; Grzegorczyk and Husmeier, 2008; Parviainen and Koivisto, 2011). One method that helps the Order MCMC (Friedman and Koller, 2003) to correct this bias was proposed by Ellis and Wong (2008).

In this paper, first we develop a new algorithm that can use the results of the DP algorithm (Koivisto and Sood, 2004) to efficiently sample orders according to the exact order posterior under the assumption of order-modular prior. Next, since a DAG consistent with a given order can be sampled as described by Friedman and Koller (2003) (with the assumed maximum node-indegree), our order sampling algorithm leads to the first algorithm (called DDS) that can be proved to be able to sample DAGs according to the exact DAG posterior with the same order-modular prior assumption. With our time-saving strategy for sampling DAGs, our DDS algorithm is shown to be both considerably more accurate and considerably more efficient than the Order MCMC and the Partial Order MCMC when n is moderate so that our DDS algorithm is applicable. Moreover, the estimator based on our DDS algorithm has several desirable properties; for example, unlike these MCMC algorithms, the quality of our estimator can be guaranteed by controlling the number of DAGs sampled by our DDS algorithm. The main contribution of our DDS algorithm is to solve the limitation of the exact DP algorithms (Koivisto and Sood, 2004; Koivisto, 2006; Parviainen and Koivisto, 2011) (whose usage is restricted to modular features or path features for moderate n) to estimate the posteriors of various non-modular features arbitrarily specified by users. Additionally our DDS algorithm can also be used to efficiently perform data prediction tasks in estimating $p(x|D)$ for a large number of data cases. Finally, we develop an algorithm (called IW-DDS) to correct the bias (due to the order-modular prior) in the DDS algorithm by addressing the drawbacks of the strategy of Ellis and Wong (2008). We theoretically prove that the estimator based on

our IW-DDS has several good properties; then we empirically show that our estimator is superior to the estimators based on the Hybrid MCMC method (Eaton and Murphy, 2007) and the K-best method (Tian et al., 2010), two state-of-the-art algorithms that can estimate the posterior of any feature without the order-modular prior assumption. Analogously, our IW-DDS algorithm mainly solves the limitation of the exact DP algorithm (Tian and He, 2009) to estimate the posteriors of arbitrary non-modular features and can additionally be used to efficiently perform data prediction tasks when an application situation prefers to avoid the bias from order-modular prior.

The rest of the paper is organized as follows. In Section 4.2 we briefly review the Bayesian approach to learning Bayesian networks from data, the related DP algorithms (Koivisto and Sood, 2004; Koivisto, 2006) and the Order MCMC algorithm (Friedman and Koller, 2003). In Section 4.3 we present our order sampling algorithm, DDS algorithm, and IW-DDS algorithm; and prove good properties of the estimators based on our algorithms. We empirically demonstrate the advantages of our algorithms in Section 4.4. Section 4.5 concludes the paper. Finally, the appendix provides the proofs of all the conclusions including the propositions, theorems and corollary referenced in the paper.

4.2 Bayesian Learning of Bayesian Networks

A Bayesian network is a DAG G that encodes a joint probability distribution over a set $X = \{X_1, \ldots, X_n\}$ of random variables with each node of the DAG representing a variable in X. For convenience we typically work on the index set $V = \{1, \ldots, n\}$ and represent a variable X_i by its index i. We use $X_{Pa_i} \subseteq X$ to represent the parent set of X_i in a DAG G and use $Pa_i \subseteq V$ to represent the corresponding index set. (A pair (i, Pa_i) is often called a family.) Thus, a DAG G can be represented as a vector $(Pa_1, \ldots, Pa_n)$.

Assume that we are given a training data set $D = \{x^1, x^2, \ldots, x^m\}$, where each x^i is a particular instantiation over the set of variables X. We only consider situations where the data are complete, that is, every variable in X is assigned a value. In the Bayesian approach to learning Bayesian networks from the training data D, we compute the posterior probability of a DAG G as

$$p(G|D) = \frac{p(D|G)p(G)}{p(D)} = \frac{p(D|G)p(G)}{\sum_G p(D|G)p(G)}.$$

Assuming global and local parameter independence, and parameter modularity, $p(D|G)$ can be

decomposed into a product of local marginal likelihoods (often called local scores) as (Cooper and Herskovits, 1992; Heckerman et al., 1995)

$$p(D|G) = \prod_{i=1}^{n} p(x_i|x_{Pa_i} : D) := \prod_{i=1}^{n} score_i(Pa_i : D), \tag{4.1}$$

where, with appropriate parameter priors, $score_i(Pa_i : D)$ (the local score for a family (i, Pa_i)) has a closed form solution. In this paper we will assume that these local scores can be computed efficiently from data. The standard assumption for structure prior $p(G)$ is structure-modular prior (Friedman and Koller, 2003):

$$p(G) = \prod_{i=1}^{n} p_i(Pa_i), \tag{4.2}$$

where p_i is some nonnegative function over the subsets of $V - \{i\}$.

Combing Eq. (4.1) and Eq. (4.2), we have

$$p_{\not\prec}(G, D) = p(D|G)p(G) = \prod_{i=1}^{n} score_i(Pa_i : D)p_i(Pa_i). \tag{4.3}$$

Note that the subscript $\not\prec$ is intentionally added by us to mean that the corresponding probability is the one obtained without order-modular prior assumption. This is different from the probability computed with order-modular prior assumption, which will be marked by the subscript $\prec$ for the distinction.

We can compute the posterior probability of any hypothesis of interest by averaging over all the possible DAGs. For example, we are often interested in computing the posteriors of structural features. Let f be a structural feature represented by an indicator function such that $f(G)$ is 1 if the feature is present in G and 0 otherwise. By the full Bayesian model averaging, we have the posterior of f as

$$p(f|D) = \sum_{G} f(G)p(G|D). \tag{4.4}$$

Note that $p_{\not\prec}(f|D)$ will be obtained if $p(G|D)$ in Eq. (4.4) is $p_{\not\prec}(G|D)$; $p_{\prec}(f|D)$ will be obtained if $p(G|D)$ in Eq. (4.4) is $p_{\prec}(G|D)$. This difference is the key to understanding the bias issue which will be described in details later.

Since summing over all the possible DAGs is generally infeasible, one approach to computing the posterior of f is to draw DAG samples $\{G_1, \ldots, G_T\}$ from the posterior $p_{\not\prec}(G|D)$ or $p_{\prec}(G|D)$, which can then be used to estimate the posterior $p_{\not\prec}(f|D)$ or $p_{\prec}(f|D)$ as

$$\hat{p}(f|D) = \frac{1}{T} \sum_{i=1}^{T} f(G_i). \tag{4.5}$$

4.2.1 The DP Algorithms

The DP algorithms (Koivisto and Sood, 2004; Koivisto, 2006) work in the order space rather than the DAG space. We define an order $\prec$ of variables as a total order (a linear order) on V represented as a vector $(U_1, \ldots, U_n)$, where U_i is the set of predecessors of i in the order $\prec$. To be more clear we may use $U_i^{\prec}$. We say that a DAG $G = (Pa_1, \ldots, Pa_n)$ is consistent with an order $(U_1, \ldots, U_n)$, denoted by $G \subseteq \prec$, if $Pa_i \subseteq U_i$ for each i. If S is a subset of V, we let $\mathcal{L}(S)$ denote the set of linear orders on S. In the following we will largely follow the notation from Koivisto (2006).

The algorithms working in the order space assume order-modular prior defined as follows: if G is consistent with $\prec$, then

$$p(\prec, G) = \prod_{i=1}^{n} q_i(U_i)\rho_i(Pa_i), \tag{4.6}$$

where each q_i and ρ_i is some function from the subsets of $V - \{i\}$ to the nonnegative real numbers. (If G is not consistent with $\prec$, then $p(\prec, G) = 0$.)

A modular feature is defined as:

$$f(G) = \prod_{i=1}^{n} f_i(Pa_i),$$

where $f_i(Pa_i)$ is an indicator function returning a $0/1$ value. For example, an edge $j \rightarrow i$ can be represented by setting $f_i(Pa_i) = 1$ if and only if $j \in Pa_i$ and setting $f_l(Pa_l) = 1$ for all $l \neq i$.

With the order-modular prior, we are interested in the posterior $p_{\prec}(f|D) = p_{\prec}(f, D)/p_{\prec}(D)$. $p_{\prec}(f|D)$ can be obtained if the joint probability $p_{\prec}(f, D)$ can be computed (since $p_{\prec}(D) = p_{\prec}(f \equiv 1, D)$ where $f \equiv 1$, meaning that f always equals 1, can be easily achieved by setting each $f_i(Pa_i)$ to be the constant 1). Koivisto and Sood (2004) show that

$$p(f, \prec, D) = \prod_{i=1}^{n} \alpha_i(U_i^{\prec}), \tag{4.7}$$

and

$$p_{\prec}(f, D) = \sum_{\prec} \prod_{i=1}^{n} \alpha_i(U_i^{\prec}), \tag{4.8}$$

where the function α_i is defined for each $i \in V$ and each $S \subseteq V - \{i\}$ as

$$\alpha_i(S) = q_i(S) \sum_{Pa_i \subseteq S} \beta_i(Pa_i),$$

in which the function β_i is defined for each $i \in V$ and each $Pa_i \subseteq V - \{i\}$ as

$$\beta_i(Pa_i) = f_i(Pa_i)\rho_i(Pa_i)score_i(Pa_i : D).$$

Thus, the DP algorithm (Koivisto and Sood, 2004) consists of the following three steps. The first step computes $\beta_i(Pa_i)$ for each $i \in V$ and each $Pa_i \subseteq V - \{i\}$. The time complexity of this step is $O(n^{k+1}C(m))$ under the assumption of the maximum in-degree k, where n is the number of variables, and $C(m)$ is the cost of computing a single local marginal likelihood $score_i(Pa_i : D)$ for m data instances. The second step computes $\alpha_i(S)$ for each $i \in V$ and each $S \subseteq V - \{i\}$. With the assumed maximum in-degree k, this step takes $O(kn2^n)$ time by using the truncated Möbius transform technique (Koivisto and Sood, 2004) which is extended from the standard fast Möbius transform algorithm (Kennes and Smets, 1991). The third step computes $p_{\prec}(f, D)$ by defining the following function (called forward contribution) for each $S \subseteq V$:

$$L(S) = \sum_{\prec \in \mathcal{L}(S)} \prod_{i \in S} \alpha_i(U_i^{\prec}), \tag{4.9}$$

where $U_i^{\prec}$ is the set of variables in S ahead of i in the order $\prec \in \mathcal{L}(S)$. It can be shown that for every $S \subseteq V$ the corresponding $L(S)$ can be computed recursively using the DP technique according to the following equation (Koivisto and Sood, 2004; Koivisto, 2006):

$$L(S) = \sum_{i \in S} \alpha_i(S - \{i\})L(S - \{i\}), \tag{4.10}$$

starting with $L(\emptyset) = 1$ and ending with $L(V)$. Combining Eq. (4.8) and Eq. (4.9), we have

$$p_{\prec}(f, D) = L(V). \tag{4.11}$$

The third step takes $O(n2^n)$ time when $L(V)$ is computed using the above DP technique. Therefore, in summary, the whole DP algorithm (Koivisto and Sood, 2004) can compute the posterior of any modular feature (such as an edge feature) in $O(n^{k+1}C(m) + kn2^n)$ time and $O(n2^n)$ space.

As the extended work of Koivisto and Sood (2004), Koivisto (2006) includes the DP algorithm (Koivisto and Sood, 2004) as its first three steps and appends two additional steps so that all the $n(n-1)$ edges can be computed in $O(n^{k+1}C(m) + kn2^n)$ time and $O(n2^n)$ space. The foundation of the two additional steps is the introduction of the following function (called backward contribution) for each

$T \subseteq V$:

$$R(T) = \sum_{\prec' \in \mathcal{L}(T)} \prod_{i \in T} \alpha_i((V - T) \cup U_i^{\prec'}). \tag{4.12}$$

Like $L(S)$, $R(T)$ can also be computed recursively using some DP technique. Please refer to the paper of Koivisto (2006) for further details of the two additional steps.

While the DP algorithms (Koivisto and Sood, 2004; Koivisto, 2006) make significant contributions to the structure learning of Bayesian networks, their limitation is also obvious: they can only compute the posteriors of modular features. In the next section of this paper, we will show how to use the results of the DP algorithm (Koivisto and Sood, 2004) to efficiently draw DAG samples, which can then be used to compute the posteriors of arbitrary features.

4.2.2 Order MCMC

The idea of the Order MCMC is to use the Metropolis-Hastings algorithm to draw order samples $\{\prec_1, \ldots, \prec_{N_o}\}$ that have $p(\prec | D)$ as the invariant distribution, where N_o is the number of sampled orders. For this purpose we need to be able to compute $p(\prec, D)$, which can be obtained from Eq. (4.7) by setting $f \equiv 1$. Let $\beta_i'(Pa_i)$ denote $\beta_i(Pa_i)$ resulted from setting each $f_i(Pa_i)$ to be the constant 1. Similarly, we define $\alpha_i'(S)$ and $L'(S)$ as the special cases of $\alpha_i(S)$ and $L(S)$ respectively by setting each $f_i(Pa_i)$ to be the constant 1. Then from Eq. (4.7) and (4.11) we have

$$p(\prec, D) = \prod_{i=1}^{n} \alpha_i'(U_i^{\prec}), \tag{4.13}$$

and

$$p_{\prec}(D) = L'(V). \tag{4.14}$$

The Order MCMC can estimate the posterior of a modular feature as

$$\hat{p}_{\prec}(f|D) = \frac{1}{N_o} \sum_{i=1}^{N_o} p(f| \prec_i, D). \tag{4.15}$$

For example, from Propositions 3.1 and 3.2 stated by Friedman and Koller (2003) as well as the definitions of β_i' and α_i', the posterior of a particular choice of parent set $Pa_i \subseteq U_i^{\prec}$ for node i given an order is

$$p((i, Pa_i)| \prec, D) = \frac{\beta_i'(Pa_i)}{\alpha_i'(U_i^{\prec})/q_i(U_i^{\prec})}, \tag{4.16}$$

and the posterior of the edge feature $j \to i$ given an order is

$$p(j \to i| \prec, D) = 1 - \frac{\alpha'_i(U_i^{\prec} - \{j\})/q_i(U_i^{\prec} - \{j\})}{\alpha'_i(U_i^{\prec})/q_i(U_i^{\prec})}. \tag{4.17}$$

In order to compute arbitrary non-modular features, we further draw DAG samples after drawing N_o order samples. Given an order, a DAG can be sampled by drawing parents for each node according to Eq. (4.16). Given DAG samples $\{G_1, \ldots, G_T\}$, we can then estimate any feature posterior $p_{\prec}(f|D)$ using $\hat{p}_{\prec}(f|D)$ shown in Eq. (4.5).

4.3 Order Sampling Algorithm and DAG Sampling Algorithms

In this section we present our order sampling algorithm, DDS algorithm and IW-DDS algorithm. We also prove good properties of the estimators based on our algorithms.

4.3.1 Order Sampling Algorithm

In this subsection, we show that using the results including $\alpha'_i(S)$ (for each $i \in V$ and each $S \subseteq V - \{i\}$) and $L'(S)$ (for each $S \subseteq V$) computed from the DP algorithm (Koivisto and Sood, 2004), we can draw order samples efficiently by drawing each element in the order one by one. Let an order $\prec$ be represented as $(\sigma_1, \ldots, \sigma_n)$ where σ_i is the ith element in the order.

Proposition 5 *The conditional probability that the kth ($1 \leq k \leq n$) element in the order is σ_k given that the $n - k$ elements after it along the order are $\sigma_{k+1}, \ldots, \sigma_n$ respectively is as follows:*

$$p(\sigma_k|\sigma_{k+1}, \ldots, \sigma_n, D) = \frac{L'(U_{\sigma_k}^{\prec})\alpha'_{\sigma_k}(U_{\sigma_k}^{\prec})}{L'(U_{\sigma_{k+1}}^{\prec})}, \tag{4.18}$$

where $\sigma_k \in V - \{\sigma_{k+1}, \ldots, \sigma_n\}$, and $U_{\sigma_i}^{\prec} = V - \{\sigma_i, \sigma_{i+1}, \ldots, \sigma_n\}$ so that $U_{\sigma_i}^{\prec}$ denotes the set of predecessors of σ_i in the order $\prec$.

Specifically for $k = n$, we essentially have

$$p(\sigma_n = i|D) = \frac{L'(V - \{i\})\alpha'_i(V - \{i\})}{L'(V)}, \tag{4.19}$$

where $i \in V$.

Note that all the proofs in this paper are provided in the appendix.

It is clear that for each $k \in \{1, \ldots, n\}$, $\sum_{i \in U^{\prec}_{\sigma_{k+1}}} p(\sigma_k = i | \sigma_{k+1}, \ldots, \sigma_n, D) = 1$ because of Eq. (4.10) and $U^{\prec}_{\sigma_k} = U^{\prec}_{\sigma_{k+1}} - \{\sigma_k\}$. Thus, $p(\sigma_k | \sigma_{k+1}, \ldots, \sigma_n, D)$ is a probability mass function (pmf) with k possible σ_k values from $U^{\prec}_{\sigma_{k+1}}$.

Based on Proposition 5, we propose the following order sampling algorithm to sample an order $\prec$:

- Sample σ_n, the last element of the order $\prec$, according to Eq. (4.19).
- For each k from $n-1$ down to 1: given the sampled $(\sigma_{k+1}, \ldots, \sigma_n)$, sample σ_k, the kth element of the order $\prec$, according to Eq. (4.18).

Sampling an order using the above algorithm takes only $O(n^2)$ time since sampling each element σ_k ($k \in \{1, \ldots, n\}$) in the order takes $O(n)$ time.

The following proposition guarantees the correctness of our order sampling algorithm.

Proposition 6 *An order $\prec$ sampled according to our order sampling algorithm has its pmf equal to the exact posterior $p(\prec | D)$ under the order-modular prior, because*

$$\prod_{k=1}^{n} p(\sigma_k | \sigma_{k+1}, \ldots, \sigma_n, D) = p(\prec | D). \tag{4.20}$$

The key to our order sampling algorithm is that we realize that the results including $\alpha'_i(S)$ and $L'(S)$ computed from the DP algorithm of Koivisto and Sood (2004) are already sufficient to guide the order sampling process. In an abstract point of view, the results computed from the DP algorithm of Koivisto and Sood (2004) correspond to the answers from the #P-oracle stated in Theorem 3.3 of Jerrum et al. (1986). Theorem 3.3 of Jerrum et al. (1986) states that with the aid of a #P-oracle that can always answer the exact information about the number of accepting configurations from the currently given configuration, a probabilistic Turing Machine can serve as a uniform generator so that every accepting configuration will be reached with an equal positive probability. In our situation we intend to sample each order according to the exact order posterior distribution $p(\prec | D)$ instead of the uniform probability $1/(n!)$. Thus, we use the information including $\alpha'_i(S)$ and $L'(S)$, which actually contains all the information necessary to guide the order sampling process.

4.3.2 DDS Algorithm

After drawing an order sample, then we can easily sample a DAG by drawing parents for each node according to Eq. (4.16) as described by Friedman and Koller (2003) (with the assumed maximum indegree). This naturally leads to our algorithm, termed Direct DAG Sampling (DDS), as follows:

- Step 1: Run the DP algorithm of Koivisto and Sood (2004) with each $f_i(Pa_i)$ set to be the constant 1.
- Step 2 (Order Sampling Step): Sample N_o orders such that each order $\prec$ is independently sampled according to our order sampling algorithm.
- Step 3 (DAG Sampling Step): For each sampled order $\prec$, one DAG is independently sampled by drawing a parent set for each node of the DAG according to Eq. (4.16).

The correctness of our DDS algorithm is guaranteed by the following theorem.

Theorem 3 *The N_o DAGs sampled according to the DDS algorithm are independent and identically distributed (iid) with the pmf equal to the exact posterior $p_{\prec}(G|D)$ under the order-modular prior.*

The time complexity of the DDS algorithm is as follows. Step 1 takes $O(n^{k+1}C(m) + kn2^n)$ time (Koivisto and Sood, 2004), which has been discussed in Section 4.2.1. In Step 2, sampling each order takes $O(n^2)$ time. In Step 3, sampling each DAG takes $O(n^{k+1})$ time. Thus, the overall time complexity of our DDS algorithm is $O(n^{k+1}C(m) + kn2^n + n^2N_o + n^{k+1}N_o)$. Since typically we assume $k \geq 1$, the order sampling process (Step 2) does not affect the overall time complexity of the DDS algorithm because of its efficiency.

The time complexity of our DDS algorithm depends on the assumption of the maximum in-degree k. Such an assumption is fairly innocuous, as discussed on page 101 in the article of Friedman and Koller (2003), because DAGs with very large families tend to have low scores. (The maximum-in-degree assumption is also justified in the context of biological expression data on page 270 in the article of Grzegorczyk and Husmeier 2008.) Accordingly, this assumption has been widely used in the literature (Friedman and Koller, 2003; Koivisto and Sood, 2004; Ellis and Wong, 2008; Grzegorczyk and Husmeier, 2008; Niinimaki et al., 2011; Parviainen and Koivisto, 2011) and the assumed maximum in-degree k is set to be no greater than 6 for their algorithms in all their experiments.

Note that the DAG sampling step of the DDS algorithm takes $O(n^{k+1}N_o)$ time. This will actually dominate the overall running time of the DDS algorithm (even if k is assumed to be 3 or 4), when n is moderate ($n \leq 25$) and the sample size N_o reaches a number around several thousands. Therefore, for the efficiency of our DDS algorithm, we also provide the time-saving strategy for the DAG sampling step, which will be described in details in Remark 1.

Given DAG samples, $\hat{p}_{\prec}(f|D)$, the estimator for the exact posterior of any arbitrary feature f, can be constructed by Eq. (4.5). If $C_{n,f}$ denotes the time cost of determining the structural feature f in a DAG of n nodes, then constructing $\hat{p}_{\prec}(f|D)$ takes $O(C_{n,f}N_o)$ time. (For example, $C_{n,f_e} = O(1)$ for an edge feature f_e; and $C_{n,f_p} = O(n^2)$ for a path feature f_p.) If we only need order samples, the algorithm consisting of Steps 1 and 2 will be called Direct Order Sampling (DOS). Given order samples, for some modular feature f such as a parent-set feature or an edge feature, $p(f|\ \prec_i, D)$ can be computed by Eq. (4.16) or (4.17), and then $p_{\prec}(f|D)$ can be estimated by Eq. (4.15). (Since computing a parent-set feature or an edge feature by Eq. (4.16) or (4.17) takes $O(1)$ time , estimating $p_{\prec}(f|D)$ by Eq. (4.15) only takes $O(N_o)$ time.)

As for the space costs of our DDS algorithm, please note that both a total order and a DAG can be represented in $O(n)$ space (since a total order can be represented as a vector $(U_1, \ldots, U_n)$ and a DAG can be represented as a vector $(Pa_1, \ldots, Pa_n)$.) Since Step 1 of our DDS algorithm requires $O(n2^n)$ space, the overall memory requirement of our DDS algorithm is $O(n2^n + nN_o)$. Since typically tens of thousands of DAG samples are sufficient for estimating $p_{\prec}(f|D)$, the additional $O(nN_o)$ space cost will not become the space issue at all. However, if a very large N_o (such as being above the magnitude of 1×10^6) is needed for the estimation due to some specific purpose of a user, N_o can be replaced with a smaller value of N_{bl} in the DDS algorithm, where N_{bl} is the size of a sample block, and then Steps 2 and 3 can be repeated $\lceil N_o/N_{bl} \rceil$ times. This will not change the properties of the estimator coming from the DDS algorithm but can reduce the overall memory requirement of our DDS to $O(n2^n + nN_{bl})$. Note for the performance of our time-saving strategy for the DAG sampling step (described in Remark 1), a large N_{bl} is actually preferred. Thus, N_{bl} can take a value that is large but still does not lead to the memory issue for a computer. (For instance, the value of N_{bl} can be set to be around millions for a computer with 2.0 to 8.0 GB memory.) In addition, note that the estimator $\hat{p}_{\prec}(f|D)$ can be constructed by Eq. (4.5) on the fly when each DAG gets sampled so that the memory of storing all the N_o sampled

DAGs can be saved.

Due to Theorem 3, the estimator $\hat{p}_{\prec}(f|D)$ based on our DDS algorithm has the following desirable properties.

Corollary 1 *For any structural feature f, with respect to the exact posterior $p_{\prec}(f|D)$, the estimator $\hat{p}_{\prec}(f|D)$ based on the N_o DAG samples from the DDS algorithm using Eq. (4.5) has the following properties:*

(i) $\hat{p}_{\prec}(f|D)$ is an unbiased estimator for $p_{\prec}(f|D)$.

(ii) $\hat{p}_{\prec}(f|D)$ converges almost surely to $p_{\prec}(f|D)$.

(iii) If $0 < p_{\prec}(f|D) < 1$, then the random variable

$$\frac{\sqrt{N_o}(\hat{p}_{\prec}(f|D) - p_{\prec}(f|D))}{\sqrt{\hat{p}_{\prec}(f|D)(1 - \hat{p}_{\prec}(f|D))}}$$

has a limiting standard normal distribution.

(iv) For any $\epsilon > 0$, any $0 < \delta < 1$, if $N_o \geq \ln(2/\delta)/(2\epsilon^2)$, then $p(|\hat{p}_{\prec}(f|D) - p_{\prec}(f|D)| < \epsilon) \geq 1 - \delta$.

In particular, Corollary 1 (iv), which is essentially from Hoeffding bound (Hoeffding, 1963; Koller and Friedman, 2009): $p(|\hat{p}_{\prec}(f|D) - p_{\prec}(f|D)| \geq \epsilon) \leq 2e^{-2N_o\epsilon^2}$, states that in order to ensure that the probability that the error of the estimator $\hat{p}_{\prec}(f|D)$ from the DDS algorithm is bounded by ϵ is at least $1 - \delta$, we just need to require the sample size $N_o \geq \ln(2/\delta)/(2\epsilon^2)$. This property, which the MCMC algorithms (Friedman and Koller, 2003; Niinimaki et al., 2011) do not have, can be used to obtain quality guarantee for the estimator from our DDS algorithm.

Remark 1 *About our time-saving strategy for the DAG sampling step of the DDS.*

The running time of the DAG sampling step (Step 3) of the DDS algorithm is $O(n^{k+1}N_o)$, which will actually dominate the overall running time of the DDS algorithm when n is moderate and the sample size N_o reaches a number around several thousands. Thus, in the following we will introduce our strategy which is able to effectively reduce the running time of the DAG sampling step so that the efficiency of the overall DDS algorithm can be achieved.

In the DAG sampling step, each sampled order $\prec_i = (\sigma_1, \ldots, \sigma_n)^{\prec_i}$ ($1 \leq i \leq N_o$) can be represented as $\{(\sigma_1, U_{\sigma_1})^{\prec_i}, \ldots, (\sigma_n, U_{\sigma_n})^{\prec_i}\}$, where U_{σ_j} denotes the set of predecessors of σ_j in the order.

For each sampled order $\prec_i$, for each $(\sigma_j, U_{\sigma_j})^{\prec_i}$ $(1 \leq j \leq n)$, we need to sample one Pa_{σ_j} of σ_j (one parent set of σ_j) from a list $\{Pa_{\sigma_j z}\}_z^{\sigma_j}$ including every parent set $Pa_{\sigma_j z} \subseteq U_{\sigma_j}^{\prec_i}$. Let Z_j be the length of such a list. Since $Z_j = \sum_{i=0}^{min\{k,j-1\}} \binom{j-1}{i} = O(n^k)$, sampling one Pa_{σ_j} for σ_j takes $O(n^k)$ time and sampling one DAG takes $O(n^{k+1})$ time. Note that Z_j is actually an increasing function of j but in the following we use the notation Z instead of Z_j for notational convenience when the context is clear.

However, for $N_o > 1$, the overall running time of the DAG sampling step can be reduced as follows. Let $\theta_z^{(\sigma_j, U_{\sigma_j})^{\prec_i}}$ be $P((\sigma_j, Pa_{\sigma_j z})|\ \prec_i, D) = P((\sigma_j, Pa_{\sigma_j z})|(\sigma_j, U_{\sigma_j})^{\prec_i}, D) = \beta'_{\sigma_j}(Pa_{\sigma_j z}) / [\alpha'_{\sigma_j}(U_{\sigma_j}^{\prec_i}) / q_{\sigma_j}(U_{\sigma_j}^{\prec_i})]$, for $z \in \{1, \ldots, Z\}$. First, using the common strategy of sampling from a discrete distribution (Koller and Friedman, 2009), for $(\sigma_j, U_{\sigma_j})^{\prec_i}$ we can create $S_I^{(\sigma_j, U_{\sigma_j})^{\prec_i}}$, a sequence of Z probability intervals with the form of $< [0, \theta_1^{(\sigma_j, U_{\sigma_j})^{\prec_i}}), [\theta_1^{(\sigma_j, U_{\sigma_j})^{\prec_i}}, \theta_1^{(\sigma_j, U_{\sigma_j})^{\prec_i}} + \theta_2^{(\sigma_j, U_{\sigma_j})^{\prec_i}}), \ldots, [\sum_{z=1}^{Z-2} \theta_z^{(\sigma_j, U_{\sigma_j})^{\prec_i}}, \sum_{z=1}^{Z-1} \theta_z^{(\sigma_j, U_{\sigma_j})^{\prec_i}}), [\sum_{z=1}^{Z-1} \theta_z^{(\sigma_j, U_{\sigma_j})^{\prec_i}}, 1) >$, where the lth interval is $[\sum_{z=1}^{l-1} \theta_z^{(\sigma_j, U_{\sigma_j})^{\prec_i}}, \sum_{z=1}^{l} \theta_z^{(\sigma_j, U_{\sigma_j})^{\prec_i}})$. Note that $S_I^{(\sigma_j, U_{\sigma_j})^{\prec_i}}$ can be created in time $O(Z)$ and sampling one Pa_{σ_j} for σ_j from a list $\{Pa_{\sigma_j z}\}_z^{\sigma_j}$ can then be achieved using binary search in time $O(logZ)$ based on $S_I^{(\sigma_j, U_{\sigma_j})^{\prec_i}}$. Then the following observation is the key reason for reducing the running time of the DAG sampling step. For two sampled orders $\prec_i$ and $\prec_{i'}$ $(1 \leq i, i' \leq N_o)$, even if $\prec_i \neq \prec_{i'}$, it is possible that $(\sigma_j, U_{\sigma_j})^{\prec_i} = (\sigma_j, U_{\sigma_j})^{\prec_{i'}}$ for some $j \in \{1, \ldots, n\}$. This is because for each j, (σ_j, U_{σ_j}) essentially has a multinomial distribution with N_o trials and a set of $n\binom{n-1}{j-1}$ cell probabilities $\{P((\sigma_j, U_{\sigma_j})|D)\}$. Actually, for any j, the following relation holds for each cell probability:

$$P((\sigma_j, U_{\sigma_j})|D) \propto \alpha'_{\sigma_j}(U_{\sigma_j}) L'(U_{\sigma_j}) R'(V - U_{\sigma_j} - \{\sigma_j\}), \tag{4.21}$$

where $R'(.)$ is the special case of $R(.)$ by setting $f \equiv 1$ and $R(.)$ is defined in Eq. (4.12). Its proof is very similar to the derivation shown by Koivisto (2006) and is provided in the appendix. Note that $(\sigma_j, U_{\sigma_j})^{\prec_i} = (\sigma_j, U_{\sigma_j})^{\prec_{i'}}$ implies $S_I^{(\sigma_j, U_{\sigma_j})^{\prec_i}} = S_I^{(\sigma_j, U_{\sigma_j})^{\prec_{i'}}}$. Thus, by storing the created $S_I^{(\sigma_j, U_{\sigma_j})^{\prec_i}}$ in the memory, once $(\sigma_j, U_{\sigma_j})^{\prec_i} = (\sigma_j, U_{\sigma_j})^{\prec_{i'}}$ for $i' > i$, creating $S_I^{(\sigma_j, U_{\sigma_j})^{\prec_{i'}}}$ can be avoided and sampling one Pa_{σ_j} for σ_j takes only $O(logZ)$ time.

On one hand, our strategy will definitely save the running time for these j's such that $n\binom{n-1}{j-1}$ (the number of all the possible values of (σ_j, U_{σ_j})) is smaller than N_o if every created $S_I^{(\sigma_j, U_{\sigma_j})}$ is stored. This is because the running time of sampling Pa_{σ_j} of σ_j is only $O(logZ)$ in at least $N_o - n\binom{n-1}{j-1}$ samples out of the overall N_o samples. (In the worst case, $S_I^{(\sigma_j, U_{\sigma_j})}$ will be created for each possible

(σ_j, U_{σ_j}).) *For example, when* $j = n$, *the number of all the possible values of* (σ_j, U_{σ_j}) *is only* n *and* Z_j *(the length of the list* $\{Pa_{\sigma_j z}\}_z^{\sigma_j}$*) achieves its maximum among all the* j*'s so that sampling one* Pa_{σ_n} *for* σ_n *takes* $O(logZ_n)$ *time in at least* $N_o - n$ *samples. Accordingly, when* $j = n$, *the worst-case running time of sampling the* N_o $(\sigma_n, Pa_{\sigma_n})$ *families is* $O(n(Z_n + logZ_n) + (N_o - n)logZ_n) = O(nZ_n + N_o logZ_n)$. *On the other hand, our strategy usually can also save the running time even for these* j*'s such that the number of all the possible values of* (σ_j, U_{σ_j}) *is larger than* N_o. *This is because the probability mass usually is not uniformly distributed among the set of all the possible values of* (σ_j, U_{σ_j}). *Once the majority of probability mass* p_Σ *is concentrated on* r_j (σ_j, U_{σ_j}) *values, where* r_j *is a number smaller than* N_o, *the probability that only these* r_j (σ_j, U_{σ_j}) *values appear in all the* N_o *order samples is* $(p_\Sigma)^{N_o}$. *Accordingly, with the probability* $(p_\Sigma)^{N_o}$, *the running time of sampling* Pa_{σ_j} *for* σ_j *is* $O(logZ_j)$ *in at least* $N_o - r_j$ *samples. As a result, the expected running time of sampling the* N_o $(\sigma_j, Pa_{\sigma_j})$ *families is below* $O([r_j Z_j + N_o logZ_j](p_\Sigma)^{N_o} + N_o(Z_j + logZ_j)(1 - (p_\Sigma)^{N_o}))$; *the expected running time of sampling the* N_o *DAGs is below* $O(\sum_{j=1}^{n}\{[r_j Z_j + N_o logZ_j](p_\Sigma)^{N_o} + N_o(Z_j + logZ_j)(1 - (p_\Sigma)^{N_o})\})$. *Typically, when* m *is not small, local score* $score_i(Pa_i : D)$ *will not be uniform at all. Correspondingly, it is likely that the multinomial probability mass function* $P((\sigma_j, U_{\sigma_j})|D)$ *will concentrate dominant probability mass on a small number of* (σ_j, U_{σ_j}) *candidates that these* $(\sigma_j, Pa_{\sigma_j})$*'s having large local scores are consistent with. As a result, our time-saving strategy will usually become more effective when* m *is not small.*

Note that we also include the policy of recycling the created $S_I^{(\sigma_j, U_{\sigma_j})}$*s for our strategy because it is possible that all the memory in a computer will be exhausted in order to store all the created* $S_I^{(\sigma_j, U_{\sigma_j})}$*s, especially when* n *is not small but* m *is small. (The space complexity of storing all the* $S_I^{(\sigma_j, U_{\sigma_j})}$*s is* $O(\sum_{j=1}^{n} n\binom{n-1}{j-1}Z_j) = O(n^{k+1}2^{n-1})$.*) For this paper, we use a simple recycling method as follows. Some upper limit for the total number of the probability intervals (representing* $[\sum_{z=1}^{l-1} \theta_z^{(\sigma_j, U_{\sigma_j})^{\prec_i}}$, $\sum_{z=1}^{l} \theta_z^{(\sigma_j, U_{\sigma_j})^{\prec_i}})$ *) is pre-specified based on the memory of the used computer. Each time such an upper limit is reached during the DAG sampling step of the DDS, which indicates a large amount of memory has been used to store* $S_I^{(\sigma_j, U_{\sigma_j})}$*s, we recycle the currently stored* $S_I^{(\sigma_j, U_{\sigma_j})}$*s according to their usage frequencies which serve as the estimates of* $P((\sigma_j, U_{\sigma_j})|D)$*s. The memory for each infrequently used* $S_I^{(\sigma_j, U_{\sigma_j})}$ *will be reclaimed to ensure that at least a pre-specified number of probability intervals will be recycled from the memory. In addition, in order to have a better use of each created* $S_I^{(\sigma_j, U_{\sigma_j})}$

before it possibly gets reclaimed, we sort the N_o sampled orders according to the posterior $p(\prec|D)$ just before executing the DAG sampling step of the DDS. The underlying rationale is that if $p(\prec_i|D)$ is relatively close to $p(\prec_{i'}|D)$, which indicates $p(\prec_i, D)$ is relatively close to $p(\prec_{i'}, D)$ (since $p_{\prec}(D)$ is a constant), due to Eq. (4.13), it is likely that $\prec_i$ and $\prec_{i'}$ share some (σ_j, U_{σ_j}) component(s). (The extreme situation is that if $p(\prec_i|D)$ equals $p(\prec_{i'}|D)$, it is very likely that $\prec_i$ equals $\prec_{i'}$ so that $\prec_i$ and $\prec_{i'}$ share every (σ_j, U_{σ_j}).) Thus, $\prec_i$ and $\prec_{i'}$ having similar posteriors tend to be close to each other after the sorting so that it is likely that the common $S_I^{(\sigma_j, U_{\sigma_j})}$ will be used before the reclamation. Furthermore, as N_o increases, the probability that two orders (out of the N_o sampled orders) share some (σ_j, U_{σ_j}) component(s) increases. Accordingly, after the sorting, the probability of reusing $S_I^{(\sigma_j, U_{\sigma_j})}$ before its reclamation will also increase. As a result, the benefit of our time-saving strategy will typically increase when N_o increases.

The experimental results show that our time-saving strategy for the DAG sampling step of the DDS is very effective. Please see the discussion in Section 4.4 about $\hat{\mu}(T_{DAG})$ and $\hat{\sigma}(T_{DAG})$, the sample mean and the sample standard deviation of the running time of the DAG sampling step of the DDS, which are reported in Table 4.2 and Table 4.4.

4.3.3 IW-DDS Algorithm

In this subsection we present our DAG sampling algorithm under the general structure-modular prior (Eq. (4.2)) by effectively correcting the bias due to the use of the order-modular prior.

As mentioned in Section 4.1, $p_{\prec}(f|D)$ has the bias due to the assumption of the order-modular prior. This is essentially because $p_{\prec}(G|D)$, which equals $\sum_{\prec s.t. G \subseteq \prec} p(\prec, G|D)$, does not equal $p_{\not\prec}(G|D)$, which is based on the standard structure-modular prior assumption instead of the order-modular prior assumption. In fact, with the common setting that $q_i(U_i)$ always equals 1 ($q_i(U_i) \equiv 1$), if $\rho_i(Pa_i)$ in Eq. (4.6) is set to be always equal to $p_i(Pa_i)$ in Eq. (4.2) ($\rho_i(Pa_i) \equiv p_i(Pa_i)$), the following relation holds:

$$p_{\prec}(G|D) = \frac{p_{\not\prec}(D)}{p_{\prec}(D)} \cdot |\prec_G| \cdot p_{\not\prec}(G|D) \tag{4.22}$$

where $|\prec_G|$ is the number of orders that G is consistent with. (The proof of Eq. (4.22) is given in the

appendix.) Accordingly,

$$p_{\not\prec}(f|D) = \sum_G f(G)p_{\not\prec}(G|D) = \sum_G f(G)\frac{p_{\prec}(D)}{p_{\not\prec}(D)} \cdot \frac{1}{\mid \prec_G \mid} p_{\prec}(G|D).$$

Since $p_{\prec}(D)$ and $p_{\not\prec}(D)$ can be computed by the DP algorithm of Koivisto and Sood (2004) and the DP algorithm of Tian and He (2009) respectively, if $\mid \prec_{G_i} \mid$ is known for each sampled G_i ($i \in \{1, 2, \ldots, N_o\}$), we can use importance sampling to obtain a good estimator

$$\tilde{p}_{\not\prec}(f|D) = \frac{1}{N_o}\sum_{i=1}^{N_o} f(G_i)\frac{p_{\prec}(D)}{p_{\not\prec}(D)} \cdot \frac{1}{\mid \prec_{G_i} \mid}, \tag{4.23}$$

where each G_i is sampled from our DDS algorithm. Unfortunately, $\mid \prec_{G_i} \mid$ is #P hard to compute for each G_i (Brightwell and Winkler, 1991); and the state-of-the-art DP algorithm proposed by Niinimaki and Koivisto (2013) for computing $\mid \prec_{G_i} \mid$ takes $O(n2^n)$ time. Therefore, due to the expensive computation cost, in the following we will propose an algorithm which can construct an estimator other than the estimator shown in Eq. (4.23).

Because $p_{\prec}(f|D)$ has the bias with respect to $p_{\not\prec}(f|D)$, a good estimator $\hat{p}_{\prec}(f|D)$ for $p_{\prec}(f|D)$ typically is not appropriate to be directly used to estimate $p_{\not\prec}(f|D)$. Noticing this problem, Ellis and Wong (2008) propose to correct this bias for the Order MCMC method as follows: first run the Order MCMC to draw order samples; then for each unique order $\prec$ out of the sampled orders, keep drawing DAGs consistent with $\prec$ until the sum of joint probabilities for the unique sampled DAGs, $\sum_i p(\prec, G_i, D)$, is no less than a pre-specified large proportion (such as 95%) of $p(\prec, D) = \sum_{G \subseteq \prec} p(\prec, G, D)$; finally the resulting union of all the DAG samples is treated as an importance-weighted sample for the structural discovery.

Unfortunately, the strategy of Ellis and Wong (2008) has some drawbacks in terms of both computation costs and the properties of the sampled DAGs. (Please refer to Remark 2 for detailed discussion.) To address the drawbacks, we propose our bias-corrected algorithm, termed IW-DDS (Importance-weighted DDS), as follows:

- Step 1 (DDS Step): Run the DDS algorithm with the setting that $q_i(U_i) \equiv 1$ and $\rho_i(Pa_i) \equiv p_i(Pa_i)$.
- Step 2 (Bias Correction Step): Make the union set $\mathcal{G}$ of all the sampled DAGs by eliminating the duplicate DAGs.

Given $\mathcal{G}$, $\hat{p}_{\not\propto}(f|D)$, the estimator for the exact posterior of any feature f, can then be constructed as

$$\hat{p}_{\not\propto}(f|D) = \sum_{G \in \mathcal{G}} f(G)\hat{p}_{\not\propto}(G|D), \tag{4.24}$$

where

$$\hat{p}_{\not\propto}(G|D) = \frac{p_{\not\propto}(G, D)}{\sum_{G \in \mathcal{G}} p_{\not\propto}(G, D)}, \tag{4.25}$$

and $p_{\not\propto}(G, D)$ is given in Eq. (4.3).

In order to easily obtain the estimator $\hat{p}_{\not\propto}(G|D)$ using Eq. (4.24) and (4.25), the joint probability $p_{\not\propto}(G_i, D)$ needs to be conveniently obtained for each sampled G_i. One way is to record only $p(D|G_i)$ with each sampled G_i if a uniform prior $p(G)$ is used, and to record both $p(D|G_i)$ and $p(G_i)$ with each sampled G_i if a non-uniform prior $p(G)$ is used. By this way, obtaining $p_{\not\propto}(G_i, D)$ for each sampled G_i can be achieved by Eq. (4.3) in $O(1)$ time.

Since checking the equality of two DAGs takes $O(n)$ time by using their vector representations, for the bias correction step of the IW-DDS algorithm, with the usage of a hash table, its expected time cost is $O(nN_o)$ and its space cost is $O(nN_o)$. Therefore, the expected time cost of our IW-DDS algorithm is $O(n^{k+1}C(m) + kn2^n + n^2N_o + n^{k+1}N_o)$, and the required memory space of our IW-DDS algorithm is $O(n2^n + nN_o)$.

Let $C_{n,f}$ denote the time cost of determining the structural feature f in a DAG of n nodes. Just as the analysis for the estimator from the DDS, constructing the estimator $\hat{p}_{\not\propto}(f|D)$ from the IW-DDS takes $O(C_{n,f}N_o)$ time when the joint probability $p_{\not\propto}(G_i, D)$ can be obtained for each sampled G_i in $O(1)$ time.

While Ellis and Wong (2008) show the effectiveness of their methods in correcting the bias merely by the experiments, we first prove good properties of $\hat{p}_{\not\propto}(f|D)$ based on our IW-DDS as follows.

Theorem 4 *For any structural feature f, with respect to the exact posterior $p_{\not\propto}(f|D)$, the estimator $\hat{p}_{\not\propto}(f|D)$ based on the DAG samples from the IW-DDS algorithm using Eq. (4.24) has the following properties:*

(i) $\hat{p}_{\not\propto}(f|D)$ is an asymptotically unbiased estimator for $p_{\not\propto}(f|D)$.

(ii) $\hat{p}_{\not\propto}(f|D)$ converges almost surely to $p_{\not\propto}(f|D)$.

(iii) The convergence rate of $\hat{p}_{\not\propto}(f|D)$ is $o(a^{N_o})$ for any $0 < a < 1$.

(iv) Define the quantity $\Delta = \sum_{G \in \mathcal{G}} p_{\not\perp}(G|D)$*, then*

$$\Delta \cdot \hat{p}_{\not\perp}(f|D) \leq p_{\not\perp}(f|D) \leq \Delta \cdot \hat{p}_{\not\perp}(f|D) + 1 - \Delta. \tag{4.26}$$

Note that the introduced quantity $\Delta = \sum_{G \in \mathcal{G}} p_{\not\perp}(G, D)/p_{\not\perp}(D)$ and $\Delta \in [0, 1]$ essentially represents the cumulative posterior probability mass of the DAGs in $\mathcal{G}$. Eq. (4.26) provides the sound interval which $p_{\not\perp}(f|D)$ must reside in. (The "sound interval" is stronger than the concept of "confidence interval" because there is no probability that $p_{\not\perp}(f|D)$ is outside the sound interval.) Also note that Δ is a nondecreasing function of N_o (because if we increase the original N_o to a larger N'_o, the resulting $\mathcal{G}'$ is always the superset of the original $\mathcal{G}$). Thus, for the cases where m (the number of data instances) is not very small, it is possible for Δ to approach 1 by a tractable number N_o of DAG samples so that a desired small-width interval for $p_{\not\perp}(f|D)$ can be obtained. (Please refer to Section 4.4 for the corresponding experimental results.)

Similar to the DDS algorithm, if a very large N_o (such as being above the magnitude of 1×10^6) is needed to obtain $\hat{p}_{\not\perp}(G|D) = (\sum_{G \in \mathcal{G}} f(G) p_{\not\perp}(G, D))/(\sum_{G \in \mathcal{G}} p_{\not\perp}(G, D))$ or the corresponding sound interval, the following memory-saving strategy can be used to reduce the memory requirement of the IW-DDS algorithm. N_o can be replaced with a smaller value of N_{bl} in the IW-DDS algorithm, where N_{bl} is the size of a sample block, and then Steps 2 and 3 of the DDS as well as the bias correction step can be repeated $\lceil N_o/N_{bl} \rceil$ times. (Similar to the DDS algorithm, N_{bl} can take a value that is large but still does not lead to the memory issue for a computer.) After each bias correction step, up to N_{bl} DAGs get sorted according to $p(D|G)$ and then are stored as a file on the hard disk. Finally, after $\lceil N_o/N_{bl} \rceil$ loops are done, each time some score threshold p_{thr} can always be found (based on $p(D|G)$ of the corresponding quantile of the sorted DAGs) such that $O(N_{bl})$ DAGs whose $p(D|G)$ is no greater than p_{thr} are newly retrieved from these $\lceil N_o/N_{bl} \rceil$ files and reloaded into the memory each time. Thus, the elimination of the duplicate DAGs only needs to be performed among these DAGs in the memory so that both the denominator and the numerator of $\hat{p}_{\not\perp}(G|D)$ can be updated accordingly. When all the DAGs have been retrieved from these $\lceil N_o/N_{bl} \rceil$ files, $\hat{p}_{\not\perp}(G|D)$ is obtained and its denominator $\sum_{G \in \mathcal{G}} p_{\not\perp}(G, D)$ can also be used to obtain Δ. Using this strategy, the expected time cost of the IW-DDS becomes $O(n^{k+1}C(m) + kn2^n + n^2 N_o + n^{k+1} N_o + N_{bl} log(N_{bl}) \lceil N_o/N_{bl} \rceil + C_{w,r}(n) N_o + n N_o)$ $= O(n^{k+1}C(m) + kn2^n + n^2 N_o + n^{k+1} N_o + log(N_{bl}) N_o + C_{w,r}(n) N_o)$, where $C_{w,r}(n)$ is the time

cost that a DAG of n nodes is written to the hard disk and then is reloaded from the hard disk to the memory. The corresponding memory requirement is $O(n2^n + nN_{bl})$, with the addition of the $O(nN_o)$ space in the hard disk required to record $O(N_o)$ DAGs among the $\lceil N_o/N_{bl} \rceil$ files.

Remark 2 *About our solution to the drawbacks of the strategy proposed by Ellis and Wong (2008).*

Our bias-correction strategy used in the IW-DDS solves the computation problem existing in the idea of Ellis and Wong (2008) and ensures the good properties of our estimator $\hat{p}_{\not\prec}(f|D)$ stated in Theorem 4.

Since in the Order MCMC, sampling an order is much more computationally expensive than sampling a DAG given an order, the strategy of Ellis and Wong (2008) emphasizes making the full use of each sampled order $\prec$, that is, keeping drawing DAGs consistent with each sampled $\prec$ until the sum of joint probabilities for the unique sampled DAGs, $\sum_i p(\prec, G_i, D)$, is no less than a large proportion (such as 95%) of $p(\prec, D)$. Unfortunately, such a strategy has a computational problem when the number of variables n is not small and the number of data instances m is small. Because there are super-exponential number ($2^{O(knlog(n))}$) of DAGs consistent with each order (Friedman and Koller, 2003), it is possible that a non-negligible portion of probability mass $p(\prec, D)$ will be distributed almost uniformly to a majority of these consistent DAGs when m is small. Consequently, $N_G^{\prec}$, the required number of DAGs sampled per each sampled order $\prec$, will be extremely large, leading to large computation costs. For sampling DAGs consistent with each sampled order $\prec$, its expected time cost is $O(n^{k+1} + nlog(n)N_G^{\prec})$ and its memory requirement is $O(nN_G^{\prec})$. If the memory requirement exceeds the memory of the running computer, the hard disk has to be used to temporarily store the sampled DAGs in some way, such as the way that our memory-saving strategy uses for the IW-DDS. (We notice that Ellis and Wong (2008) limit their experiments to the data sets with at most 14 variables.) If we take the data set "Child" (Tsamardinos et al., 2006) with $n = 20$ and $m = 100$ for example, for an order $\prec$ randomly sampled by our order sampling algorithm, our experiment shows that 1×10^7 DAGs (which contain $932,137$ unique DAGs) need to be sampled to let the ratio $\sum_i p(\prec, G_i, D) \ / p(\prec, D)$ reach 94.071%; 1.5×10^7 DAGs (which contain $1,204,262$ unique DAGs) need to be sampled to let the ratio $\sum_i p(\prec, G_i, D) \ / p(\prec, D)$ reach 94.952%. To address this problem, based on the efficiency of our order sampling algorithm, our strategy samples only one DAG from each sampled order in the DDS step so that the large computation

costs per each sampled order are avoided for any data set. Meanwhile, unlike the strategy of Ellis and Wong (2008), our strategy does not delete the duplicate order samples. Therefore, if an order $\prec$ gets sampled j (≥ 1) times in the order sampling step, essentially j DAGs will be sampled for such a unique order in the DAG sampling step. Thus, j, the number of occurrences, implicitly serves as an importance indicator for $\prec$ among the orders.

Furthermore, the strategy of Ellis and Wong (2008) can not guarantee that the sampled DAGs are independent, even if large computation costs are spent in sampling a huge number of DAGs per each sampled order. This is essentially because multiple DAGs sampled from a fixed order according to the strategy of Ellis and Wong (2008) are not independent. For example, given that a DAG G with an edge $X \rightarrow Y$ gets sampled from an order $\prec$, which implies that node X precedes node Y in the given order $\prec$, then the conditional probability that any DAG G' with a reverse edge $Y \rightarrow X$ gets sampled under the fixed order $\prec$ becomes zero, so that G and G' are not independent. In general, once the number of sampled orders is fixed, even if the number of sampled DAGs per each sampled order keeps increasing, every DAG that is consistent with none of the sampled orders will still have no chance of getting sampled. In contrast, the sampling strategy in our IW-DDS is able to guarantee the property that all the DAGs sampled from the DDS step are independent, which has been stated in Theorem 3. Such a property actually is the key to ensuring the good properties of our estimator $\hat{p}_{\nprec}(f|D)$ stated in Theorem 4.

The competing state-of-the-art algorithms also applicable to Bayesian networks of moderate size are the Hybrid MCMC method (Eaton and Murphy, 2007) and the K-best method. (Tian et al., 2010). The first competing method, the Hybrid MCMC, includes the DP algorithm of Koivisto (2006) (with time complexity $O(n^{k+1}C(m)+ kn2^n)$ and space complexity $O(n2^n)$) as its first phase and then uses the computed posteriors of all the edges to make the global proposal for its second phase (MCMC phase). When its MCMC phase eventually converges, the Hybrid MCMC will correct the bias coming from the order-prior assumption and provide DAG samples according to the DAG posterior so that the estimator $\hat{p}_{\nprec}(f|D)$ can be constituted using Eq. (4.5) for any feature f. The Hybrid MCMC has been empirically shown to converge faster than both the Structure MCMC and the Order MCMC so that the more accurate structure learning performance can be obtained (Eaton and Murphy, 2007). Note

that since the REV-MCMC method (Grzegorczyk and Husmeier, 2008) is shown to be only nearly as efficient as the Order MCMC in the mixing and convergence, the Hybrid MCMC even converges faster than the REV-MCMC method as long as n is moderate so that the Hybrid MCMC is applicable. (But the REV-MCMC method has its own value in learning Bayesian networks of a large n since all the methods using some DP technique (including the Hybrid MCMC, the K-best method and our IW-DDS method) are infeasible for a large n due to the space cost.) One big limitation of the Hybrid MCMC is that it can not obtain the interval for $p_{\not\prec}(f|D)$ as specified by Theorem 4 (iv). Additionally, the convergence rate of the estimator from the Hybrid MCMC is not theoretically provided by its authors.

The second competing method, the K-best method, applies some DP technique to obtain a collection $\mathcal{G}$ of DAGs with the K best scores and then uses these DAGs to constitute the estimator $\hat{p}_{\not\prec}(f|D)$ by Eq. (4.24) and (4.25). One advantage of the K-best method is that its estimator also has the property specified as Theorem 4 (iv) so that it can provide the sound interval for $p_{\not\prec}(f|D)$ just as our IW-DDS. However, since the K-best method has time complexity $O(n^{k+1}C(m)+ T'(K)n2^{n-1})$ and space complexity $O(Kn2^n)$, (where $T'(K)$ is the time spent on the best-first search,) the increase in K will dramatically increase the computation costs of the K-best method. Thus, to obtain the interval of the width similar to the one from our IW-DDS, much more time and space costs are required for the K-best method. This computation problem becomes extremely severe for $n \geq 19$ since K can only take some small value (such as no more than 40) in order to prevent the K-best method from exhausting the memory of current desktop computers. According, Δ obtained from the K-best method is usually smaller than the one from our IW-DDS (so that the interval from the K-best method is usually wider) even if the setting of K reaches the memory limit of a computer. (Please refer to Section 4.4.2 for detailed discussion.)

4.4 Experimental Results

We have implemented our algorithm in C++ language and run several experiments to demonstrate its capabilities. Our tested data sets include several real data sets from the UCI Machine Learning Repository (Asuncion and Newman, 2007): "Wine," "Housing," "Zoo," "Letter," and "German". Our tested data sets also include three synthetic data sets: the first one is a synthetic data set "Syn15"

generated from a gold-standard 15-node Bayesian network built by us; the second one is a synthetic data set "Insur19" generated from a 19-node subnetwork of "Insurance" Bayesian network (Binder et al., 1997) such that the 19-node subnetwork is causally sufficient (Spirtes et al., 2001); and the third one is a synthetic data set "Child" from a 20-node "Child" Bayesian network used by Tsamardinos et al. (2006). All the data sets contain only discrete variables (or are discretized) and have no missing values. For the four data sets ("Syn15," "Letter," "Insur19" and "Child"), since a large number of data instances are available, we also vary m (the number of instances) to see the corresponding learning performance. All the experiments in this section were run under Linux on one ordinary desktop PC with a 3.0GHz Intel Pentium processor and 2.0GB of memory if no extra specification is provided. In addition, the maximum in-degree k is assumed to be 5 for all the experiments.

4.4.1 Experimental Results for the DDS

In this subsection, we compare our DDS algorithm with the Partial Order MCMC method (Niinimaki et al., 2011), the state-of-the-art learning method under the order-modular prior.

The Partial Order MCMC (PO-MCMC) method is implemented in BEANDisco, a C++ language tool provided by Niinimaki et al. (2011). (BEANDisco is available at http://www.cs.helsinki.fi/u/tzniinim /BEANDisco/.) The current version of BEANDisco can only estimate the posterior of an edge feature, but as Niinimaki et al. (2011) have stated, the PO-MCMC readily enables estimating the posterior of any structural feature by further sampling DAGs consistent with an order.

Since n (the size of the problems) is moderate, we are able to use REBEL, a C++ language implementation of the DP algorithm of Koivisto (2006), to get the exact posterior of every edge under the assumption of the order-modular prior. (REBEL is available at http://www.cs.helsinki.fi/u/mkhkoivi /REBEL/.) Thus we can use the criterion of the sum of the absolute differences (SAD) (Eaton and Murphy, 2007) to measure the feature learning performance, where SAD is $\sum_{ij} |p_{\prec}(i \to j|D) - \hat{p}_{\prec}(i \to j|D)|$, $p_{\prec}(i \to j|D)$ is the exact posterior of edge $i \to j$, and $\hat{p}_{\prec}(i \to j|D)$ is the corresponding estimator. A smaller SAD will indicate a better performance in structure discovery. Note that the criterion SAD is closely related to another criterion MAD (the mean of the absolute differences) since MAD = SAD$/(n(n-1))$. Thus, for each data case the conclusion based on the comparison of SAD values is the same as the one based on the comparison of MAD values since $n(n-1)$ is just a constant for each

Table 4.1: Comparison of the PO-MCMC, the DOS & the DDS in Terms of SAD

Name	n	m	PO-MCMC		DOS		DDS	
			$\hat{\mu}$(SAD)	$\hat{\sigma}$(SAD)	$\hat{\mu}$(SAD)	$\hat{\sigma}$(SAD)	$\hat{\mu}$(SAD)	$\hat{\sigma}$(SAD)
Wine	13	178	0.1616	0.0403	0.0505	0.0138	0.0839	0.0137
Housing	14	506	0.3205	0.1303	0.0691	0.0146	0.1150	0.0117
Zoo	17	101	0.6079	0.1809	0.1020	0.0130	0.2756	0.0137
German	21	1,000	2.8802	2.0191	0.0994	0.0295	0.1298	0.0338
Syn15	15	100	0.9024	0.2258	0.1246	0.0142	0.2622	0.0190
		200	0.6449	0.1569	0.0975	0.0163	0.2228	0.0172
		500	0.3424	0.1214	0.0651	0.0153	0.1116	0.0126
		1,000	0.1558	0.0496	0.0515	0.0128	0.0724	0.0118
		2,000	0.0465	0.0209	0.0359	0.0075	0.0473	0.0071
		5,000	0.0217	0.0144	0.0196	0.0064	0.0247	0.0086
Letter	17	100	0.9530	0.1285	0.1559	0.0210	0.2948	0.0229
		200	0.3854	0.0825	0.0827	0.0153	0.1758	0.0142
		500	0.4369	0.1529	0.0755	0.0157	0.1326	0.0107
		1,000	0.3007	0.1254	0.0537	0.0140	0.0828	0.0171
		2,000	1.3740	0.9177	0.0895	0.0238	0.1386	0.0288
		5,000	0.0669	0.0139	0.0218	0.0059	0.0292	0.0088
Insur19	19	100	0.6150	0.1995	0.0947	0.0179	0.1575	0.0213
		200	0.4428	0.1319	0.0663	0.0111	0.1024	0.0162
		500	0.2757	0.1127	0.0436	0.0140	0.0594	0.0099
		1000	0.4539	0.3031	0.0301	0.0088	0.0422	0.0134
		2000	0.0100	0.0073	0.0073	0.0042	0.0079	0.0029
		5000	0.0110	0.0100	0.0094	0.0041	0.0116	0.0043
Child	20	100	0.4997	0.1153	0.0745	0.0147	0.1772	0.0146
		200	0.1896	0.0528	0.0425	0.0081	0.0982	0.0101
		500	0.2385	0.0702	0.0434	0.0068	0.0816	0.0123
		1,000	0.1079	0.0525	0.0307	0.0093	0.0406	0.0080
		2,000	0.0864	0.0521	0.0231	0.0090	0.0275	0.0083
		5,000	0.0938	0.0539	0.0235	0.0078	0.0246	0.0066

data case.

For fair comparison, for our algorithms, we used the K2 score (Heckerman et al., 1995) and we set $q_i(U_i) = 1$ and $\rho_i(Pa_i) = 1/\binom{n-1}{|Pa_i|}$ for each i, U_i, Pa_i, where $|Pa_i|$ is the size of the set Pa_i, since such a setting is used in both BEANDisco and REBEL.

For the setting of the PO-MCMC, we set the bucket size b to be 10, which is based on the suggestion for the optimal setting from Niinimaki et al. (2011). We ran the first $10,000$ iterations for "burn-in," and then took 200 partial order samples at intervals of 50 iterations. Thus, there were $20,000$ iterations in total. (The time cost of each iteration in the PO-MCMC is $O(n^{k+1} + n^2 2^b n/b)$.) In the PO-MCMC, for each sampled partial order P_i, $p(f|D, P_i)$ is obtained by $p(D, f, P_i)/p(D, P_i) = p(D, f, P_i)/p(D, f \equiv 1, P_i)$, where $p(D, f, P_i) = \sum_{\prec \supseteq P_i} \sum_{G \subseteq \prec} f(G)\, p(\prec, G) p(D|G)$. The notation $\sum_{\prec \supseteq P_i}$ means that all the total order $\prec$'s that are linear extensions of the sampled partial order P_i will

Table 4.2: Comparison of the PO-MCMC, the DOS & the DDS in Terms of Time (Time Is in Seconds)

Name	n	m	PO-MCMC	DOS	DDS	DDS					
			$\hat{\mu}(T_t)$	$\hat{\mu}(T_t)$	$\hat{\mu}(T_t)$	$\hat{\mu}(T_{DP})$	$\hat{\sigma}(T_{DP})$	$\hat{\mu}(T_{ord})$	$\hat{\sigma}(T_{ord})$	$\hat{\mu}(T_{DAG})$	$\hat{\sigma}(T_{DAG})$
Wine	13	178	374.17	2.56	2.53	1.63	0.0121	0.35	0.0039	0.53	0.0024
Housing	14	506	510.65	4.92	4.54	3.88	0.0143	0.40	0.0033	0.23	0.0020
Zoo	17	101	1,331.80	13.75	21.90	12.05	0.0837	0.62	0.0039	9.20	0.1156
German	21	1,000	4,887.33	333.43	356.17	330.64	1.3304	0.97	0.0087	24.52	0.4441
Syn15	15	100	677.29	4.82	7.13	3.49	0.0824	0.50	0.0021	3.12	0.0337
		200	677.47	5.91	8.91	4.59	0.0473	0.47	0.0044	3.83	0.0258
		500	686.51	8.56	8.44	7.41	0.1911	0.48	0.0100	0.53	0.0050
		1,000	716.31	13.01	12.52	11.78	0.0927	0.48	0.0115	0.24	0.0022
		2,000	731.50	21.70	21.24	20.58	0.5310	0.49	0.0086	0.15	0.0016
		5,000	731.05	48.01	47.26	46.63	0.4110	0.47	0.0031	0.14	0.0008
Letter	17	100	1,322.43	16.35	21.91	14.64	0.2160	0.64	0.0036	6.60	0.0497
		200	1,315.01	19.74	20.46	18.14	0.0809	0.55	0.0026	1.74	0.0234
		500	1,338.33	27.97	28.21	26.35	0.0489	0.51	0.0066	1.32	0.0130
		1,000	1,343.88	39.45	39.03	38.11	0.3011	0.47	0.0051	0.43	0.0089
		2,000	1,358.29	61.75	61.44	60.52	0.5109	0.47	0.0078	0.42	0.0063
		5,000	1,610.37	126.53	126.49	125.67	0.7967	0.52	0.0058	0.27	0.0053
Insur19	19	100	2,616.56	53.39	86.06	51.06	0.2520	0.86	0.0091	34.11	0.9630
		200	2,633.44	62.12	77.44	59.95	0.3025	0.84	0.0083	16.61	0.2278
		500	2,680.85	80.70	85.03	77.89	0.7618	0.79	0.0082	6.32	0.0535
		1000	2,734.10	106.37	109.39	104.63	1.0958	0.89	0.0122	3.85	0.0296
		2000	2,915.60	155.05	158.31	154.26	3.7703	0.90	0.0154	3.13	0.0155
		5000	3,445.84	297.31	300.51	297.41	4.7737	0.96	0.0090	2.11	0.0091
Child	20	100	3,710.49	102.42	181.38	99.71	0.3497	1.07	0.0109	80.56	1.1980
		200	3,717.10	112.68	168.48	110.05	0.1793	1.04	0.0078	57.36	0.5335
		500	3,757.76	136.98	193.11	134.32	0.4652	1.07	0.0111	57.69	0.7276
		1,000	3,799.47	174.18	186.15	171.54	1.9832	1.08	0.0104	13.50	0.9244
		2,000	4,018.03	241.48	254.26	238.37	2.4952	1.15	0.0214	14.71	0.4833
		5,000	4,531.20	443.54	455.30	441.64	4.8785	1.16	0.0068	12.47	0.4113

be included to obtain $p(D, f, P_i)$. For example, for a data set with $n = 20$, since our bucket size $b = 10$, there are $20!/(10!10!) = 184,756$ total orders that will be included for each sampled partial order P_i. The inclusion of the information of a very large number of total orders consistent with each sampled partial order gives great learning power to the PO-MCMC method; and such an inclusion can be efficiently computed by the algorithm of Parviainen and Koivisto (2010) with the assumptions of the order-modular prior and the maximum in-degree k. Finally, for the PO-MCMC, the estimated posterior of each edge is computed using $\hat{p}_{\prec}(f|D) = (1/T)\sum_{i=1}^{T} p(f|D, P_i)$.

Because the to-be-learned feature is the edge feature, we can also use our DOS algorithm for the comparison. For both the DOS algorithm and the DDS algorithm, we set $N_o = 20,000$, that is, $20,000$ (total) orders were sampled.

Theoretically, we expect that the learning performance of the DOS should be better than the performance of the DDS because the additional approximation coming from the DAG sampling step is avoided by the DOS. By listing the performance of the DOS, we mainly intend to examine how much the performance of the DDS decreases due to the additional approximation from sampling one DAG per order. However, since the DDS but not the DOS can learn a non-modular feature, the comparison between the PO-MCMC method and the DDS method is our main task.

Table 4.1 shows experimental results in terms of SAD for each data case with n variables and m instances, while Table 4.2 lists the results about the total running time T_t corresponding to Table 4.1. For each of the three methods, we performed 15 independent runs for each data case. The sample mean and the sample standard deviation of the 15 SAD values of each method, denoted by $\hat{\mu}$(SAD) and $\hat{\sigma}$(SAD) respectively, are listed along each method in Table 4.1. Correspondingly, the sample mean of the total running time T_t of each method, denoted by $\hat{\mu}(T_t)$, is shown in Table 4.2. (Precisely speaking, the reported total running time T_t of the DDS method includes not only the time of running the three steps of the DDS but also the small $O(N_o)$ time cost of computing $\hat{p}_{\prec}(f|D)$ for each edge f using Eq. (4.5) at the end. Similarly, the reported total running time T_t of the DOS method also includes the small $O(N_o)$ time cost of computing $\hat{p}_{\prec}(f|D)$ for each edge f using Eq. (4.17) and Eq. (4.15) at the end.) In addition, the sample mean and the sample standard deviation of the running time of three steps of the DDS (including the DP step, the order sampling step and the DAG sampling step), denoted by $\hat{\mu}(T_{DP})$, $\hat{\sigma}(T_{DP})$, $\hat{\mu}(T_{ord})$, $\hat{\sigma}(T_{ord})$, $\hat{\mu}(T_{DAG})$ and $\hat{\sigma}(T_{DAG})$ respectively, are listed in the last six

columns in Table 4.2. Note that we still show $\hat{\mu}(T_{DP})$, the mean running time of the DP step of the DDS in the 15 independent runs, though the DP step is not a random algorithm at all. The running time of the DP step is not exactly the same for each run due to the randomness from uncontrolled factors such as the internal status of the computer. Using $\hat{\mu}(T_{DP})$, we can clearly see the percentage of the total running time that the DP step typically takes by comparing $\hat{\mu}(T_{DP})$ and $\hat{\mu}(T_t)$.

Tables 4.1 and 4.2 clearly illustrate the performance advantage of our DDS method over the PO-MCMC method. The overall time costs of our DDS based on $20,000$ DAG samples are much smaller than the corresponding costs of the PO-MCMC method based on $20,000$ MCMC iterations in the partial order space. Using much shorter time, $\hat{\mu}$(SAD) using our DDS method is much smaller than $\hat{\mu}$(SAD) using the PO-MCMC method for 24 out of all the 28 data cases. The four exceptional cases are Syn15 with $m = 2,000$, Syn15 with $m = 5,000$, Insur19 with $m = 2,000$, and Insur19 with $m = 5,000$. (For Insur19 with $m = 2,000$, $\hat{\mu}$(SAD) using our DDS method is still smaller than the one using the PO-MCMC method but their difference is not large relatively to $\hat{\sigma}$(SAD) from the PO-MCMC method.) Furthermore, since both $\hat{\mu}$(SAD) and $\hat{\sigma}$(SAD) are given, by the two-sample t test (Casella and Berger, 2002) with unequal variances, we can conclude with very strong evidence (at the significance level 5×10^{-4}) that the real mean of SAD using our DDS method is smaller than the real mean of SAD using the PO-MCMC method for each of the 24 data cases. For each of the four exceptions, by the same t test we accept (with the p-value > 0.2) the null hypothesis that there is no significant difference in the real means of SAD. Thus, the advantage of our DDS algorithm over the PO-MCMC method in learning Bayesian networks of a moderate n can be clearly seen, though the value of the PO-MCMC method still remains for a larger n where our DDS algorithm is infeasible.

In terms of the total running time of the DDS algorithm, Table 4.2 shows that the running time of the DP step always accounts for the largest portion. The running time of the DAG sampling step is less than 81 seconds to get $20,000$ DAG samples for all the 28 cases. Though both the order sampling step and the DAG sampling step involve randomness, the variability of their running time is actually small. This can be seen from the ratio of $\hat{\sigma}(T_{ord})$ to $\hat{\mu}(T_{ord})$ (which is always less than 2.40% for all the 28 cases) and the ratio of $\hat{\sigma}(T_{DAG})$ to $\hat{\mu}(T_{DAG})$ (which is always less than 6.85% for all the 28 cases). The ratio of $\hat{\mu}(T_{DAG})$ to $\hat{\mu}(T_{ord})$ ranges from 0.29 to 75.29 across the 28 cases, which is much smaller than the upper bound of the ratio of $O(n^{k+1}N_o)$ to $O(n^2 N_o)$. This indicates that our time-saving

strategy introduced in Remark 1 can effectively reduce the running time of the DAG sampling step. In addition, the running time of the DAG sampling step often decreases further when m increases, which can be clearly seen from all the four data sets (Syn15, Letter, Insur19 and Child) with different values of m. Take the data set Letter for example, when m increases from 100 to $1,000$, the corresponding $\hat{\mu}(T_{DAG})$ decreases from 6.60 to 0.43 second, a 93.48% of decrease. In summary, the effectiveness of our time-saving strategy introduced in Remark 1 has been clearly shown in Table 4.2.

4.4.2 Experimental Results for the IW-DDS

In this subsection, we compare our IW-DDS algorithm with the Hybrid MCMC (i.e., DP+MCMC) method (Eaton and Murphy, 2007) and the K-best method (Tian et al., 2010), two state-of-the-art methods that can estimate the posteriors of any features without the order-modular prior assumption. The implementation of the Hybrid MCMC method (called BDAGL) and the implementation of the K-best method (called K-best) are both made available online by their corresponding authors. (BDAGL is available at http://www.cs.ubc.ca/∼murphyk/Software/BDAGL/. Most part of BDAGL is written in Matlab. The original BDAGL code for the DP+MCMC method has very expensive time cost and space cost because it pre-computes the local scores of all the $n2^{n-1}$ possible families and stores them in an array. Therefore, we have updated the original Matlab code in BDAGL so that only the local scores of all the families whose sizes are no more than the assumed maximum in-degree are pre-computed and stored in a hash table.) (K-best is available at http://www.cs.iastate.edu∼/rhe/KBest_POSTER/.)

Again, since n is moderate, we are able to use POSTER, a C++ language implementation of the DP algorithm of Tian and He (2009) to get $p_{\not\curvearrowright}(i \to j|D)$, the exact posterior of each edge $i \to j$ under the assumption of the structure-modular prior instead of the order-modular prior. (POSTER is available at http://www.cs.iastate.edu/∼rhe/POSTER/.) Thus we can also use the SAD criterion ($\sum_{ij} |p_{\not\curvearrowright}(i \to j|D) - \hat{p}_{\not\curvearrowright}(i \to j|D)|$) to measure the performance of these three methods in the structural learning.

For fair comparison, for our algorithm we used the BDeu score (Heckerman et al., 1995) with equivalent sample size 1 and set $p_i(Pa_i)(\equiv \rho_i(Pa_i)) \equiv 1$, since these settings are also used in POSTER and the implementation of the K-best method.

As for the DP+MCMC method, we note that most part of its implementation in BDAGL tool is written in Matlab, which is different from the implementations of the K-best method and our IW-DDS

Table 4.3: Comparison of the DP+MCMC, the K-best & the IW-DDS in Terms of SAD

Name	n	m	DP	DP'	DP+MCMC		K-best	IW-DDS		K-best	IW-DDS	
			SAD	SAD	$\hat{\mu}$(SAD)	$\hat{\sigma}$(SAD)	SAD	$\hat{\mu}$(SAD)	$\hat{\sigma}$(SAD)	Δ	$\hat{\mu}(\Delta)$	$\hat{\sigma}(\Delta)$
Wine	13	178	1.4786	1.4676	0.4605	0.2968	0.2011	0.1041	0.0075	9.023E-01	9.670E-01	1.700E-03
Housing	14	506	5.6478	7.3294	8.0000	3.3408	9.1179	4.8276	0.0624	2.880E-02	1.096E-01	1.200E-03
Zoo	17	101	8.2142	20.1560	32.4189	10.0953	35.1215	19.7025	4.0767	3.815E-08	1.652E-11	1.211E-11
German	21	1,000	3.7261	5.4604	5.0207	3.2223	5.5902	0.9891	0.0449	7.800E-02	7.422E-01	7.200E-03
Syn15	15	100	4.9321	11.7195	12.8705	7.6384	11.8685	10.1341	0.1495	1.604E-04	1.183E-04	4.664E-06
		200	3.2557	7.5427	4.5090	2.5875	7.5232	4.9079	0.0583	2.700E-03	7.300E-03	1.265E-04
		500	6.9798	8.5252	5.5466	1.9175	4.4379	4.2965	0.1619	1.526E-01	2.388E-01	4.900E-03
		1,000	1.3000	1.1990	0.3974	0.3299	0.0848	0.0498	0.0048	9.699E-01	9.843E-01	8.909E-04
		2,000	1.7192	2.4594	1.8263	1.7095	0.3701	0.1081	0.0147	8.521E-01	9.703E-01	3.300E-03
		5,000	1.9473	2.1511	0.0304	0.0094	8.89E-04	0.0022	0.0002	9.998E-01	9.994E-01	9.821E-05
Letter	17	100	9.2140	19.8831	27.1507	4.0940	24.4313	15.8780	0.2764	2.908E-04	1.621E-04	5.336E-06
		200	7.2855	11.9685	15.1587	3.5615	9.4512	6.7936	0.1191	7.800E-03	1.220E-02	3.341E-04
		500	6.0961	9.8180	3.4637	4.6789	1.7237	0.6347	0.0119	6.948E-01	8.808E-01	3.000E-03
		1,000	0.6394	0.9274	0.1761	0.0166	0.0837	0.0766	0.0039	9.834E-01	9.864E-01	8.678E-04
		2,000	2.3913	3.6437	3.5085	3.1132	2.0976	0.1338	0.0213	6.859E-01	9.756E-01	2.700E-03
		5,000	0.8407	1.5749	0.1182	0.0442	0.0160	0.0072	0.0005	9.948E-01	9.972E-01	2.077E-04
Insur19	19	100	5.3356	9.8493	9.4318	3.9576	11.7779	6.5062	0.0891	6.000E-03	2.010E-02	4.767E-04
		200	5.9844	5.8133	5.5465	2.7295	2.2572	1.4630	0.0557	4.495E-01	7.049E-01	1.120E-02
		500	1.8274	2.0374	0.3605	0.2287	0.4970	0.1328	0.0105	7.464E-01	9.379E-01	3.000E-03
		1000	1.7386	3.0339	0.2186	0.0762	0.7498	0.0623	0.0111	5.866E-01	9.785E-01	2.600E-03
		2000	1.2737	1.2602	0.1217	0.0380	0.0174	0.0062	0.0012	9.900E-01	9.976E-01	4.864E-04
		5000	1.9511	2.0935	0.1765	0.0594	0.0103	0.0092	0.0010	9.970E-01	9.973E-01	2.086E-04
Child	20	100	6.9783	14.2007	11.8987	3.1086	11.6189	7.0304	0.0909	7.848E-04	2.700E-03	1.066E-04
		200	3.2826	5.9934	4.7066	4.3749	5.0729	2.8510	0.0192	4.800E-03	1.506E-01	1.200E-03
		500	2.5580	3.3813	2.4716	1.3489	1.5304	0.5416	0.0305	3.582E-01	8.222E-01	5.100E-03
		1,000	2.4708	2.9516	2.6061	2.2909	0.7066	0.1499	0.0198	7.013E-01	9.545E-01	3.400E-03
		2,000	2.3330	2.3932	1.4286	1.2290	1.5279	0.0662	0.0161	6.509E-01	9.828E-01	2.800E-03
		5,000	2.0365	2.0479	1.2533	1.7313	0.8783	0.0150	0.0013	8.209E-01	9.940E-01	4.696E-04

method, which are written in C++. In order to make relatively fair comparison in terms of the running time, we used REBEL tool, a C++ implementation of the DP algorithm of Koivisto (2006), to perform the computation in the DP phase; but for fair comparison we changed its scoring criterion into the BDeu score with equivalent sample size 1 and set $q_i(U_i) \equiv 1$. We tried two different $\rho_i(Pa_i)$ prior settings for our data cases: $\rho_i(Pa_i) \equiv 1$ and $\rho_i(Pa_i) = 1/\binom{n-1}{|Pa_i|}$. Consistent with the experimental results shown by Eaton and Murphy (2007), our experiments also show that the former prior setting introduces smaller biases in most of the data cases. (Please refer to the two columns DP and DP' in Table 4.3 and later discussion.) Thus, in order to get better performance of the DP+MCMC method, we used the former prior setting $\rho_i(Pa_i) \equiv 1$ in the DP phase of the DP+MCMC. To perform the computation in the MCMC phase, we had to use the Matlab implementation and we ran it under Windows 7 on an ordinary laptop with 2.40 Intel Core i5 CPU and 4.0 GB memory. The MCMC used the pure global

Table 4.4: Comparison of the DP+MCMC, the K-best & the IW-DDS in Terms of Time (Time Is in Seconds)

Name	n	m	DP+MCMC	K-best	IW-DDS	IW-DDS					
			$\hat{\mu}(T_t)$	T_t	$\hat{\mu}(T_t)$	$\hat{\mu}(T_{DP})$	$\hat{\sigma}(T_{DP})$	$\hat{\mu}(T_{ord})$	$\hat{\sigma}(T_{ord})$	$\hat{\mu}(T_{DAG})$	$\hat{\sigma}(T_{DAG})$
Wine	13	178	2.44 + 1,127.80	153.06	3.52	2.15	0.0199	0.63	0.0061	0.74	0.0057
Housing	14	506	4.83 + 1,421.00	335.23	5.67	4.21	0.0813	0.63	0.0042	0.52	0.0043
Zoo	17	101	22.33 + 2,107.50	5,543.67	26.16	12.49	0.1853	1.11	0.0048	12.05	0.1898
German	21	1,000	600.19 + 1,849.90	10,993.15	540.61	392.25	2.8656	1.68	0.0207	146.65	1.4290
Syn15	15	100	5.21 + 1,284.00	901.83	9.41	3.56	0.0667	0.81	0.0048	4.80	0.0273
		200	6.28 + 1,286.20	913.09	10.29	4.76	0.2090	0.80	0.0111	4.53	0.0256
		500	9.64 + 1,336.80	911.28	9.59	7.50	0.0821	0.85	0.0051	1.08	0.0128
		1,000	13.69 + 1,364.60	922.62	13.35	12.15	0.7578	0.85	0.0156	0.33	0.0038
		2,000	22.64 + 1,372.10	918.88	21.98	20.81	0.5200	0.85	0.0061	0.30	0.0019
		5,000	54.79 + 1,356.70	944.82	52.07	47.80	1.0887	3.99	0.0235	0.28	0.0023
Letter	17	100	25.53 + 1,572.60	7,650.88	20.27	15.83	0.0601	1.15	0.0062	2.83	0.0292
		200	30.01 + 1,576.80	7,978.11	25.87	20.22	0.1769	1.09	0.0069	4.21	0.0304
		500	39.58 + 1,598.90	8,269.46	31.88	29.84	0.1575	0.95	0.0055	1.01	0.0123
		1,000	52.85 + 1,575.60	8,392.43	44.48	42.93	0.4835	0.98	0.0093	0.56	0.0100
		2,000	77.48 + 1,591.00	7,631.15	69.14	66.34	0.5184	2.01	0.0241	0.78	0.0068
		5,000	157.69 + 1,636.40	8,146.71	136.95	134.94	1.6127	1.36	0.0097	0.65	0.0067
Insur19	19	100	101.47 + 1,828.00	6,757.27	112.28	55.85	0.1540	1.41	0.0117	54.71	0.5438
		200	113.79 + 1,896.10	6,795.62	82.52	68.12	0.4187	1.45	0.0120	12.90	0.1329
		500	137.18 + 1,864.40	6,906.01	98.96	91.44	0.4547	1.43	0.0128	6.07	0.0358
		1,000	168.55 + 1,862.30	6,978.76	133.87	123.42	0.8007	1.44	0.0157	9.00	0.0601
		2,000	226.36 + 1,781.70	7,289.46	185.02	177.66	1.3165	3.30	0.0483	4.06	0.0356
		5,000	380.78 + 1,814.80	7,073.62	336.03	329.78	4.5841	1.89	0.0220	4.34	0.0713
Child	20	100	203.44 + 1,785.10	15,097.72	223.62	106.01	0.3976	1.67	0.0176	115.60	1.3183
		200	215.70 + 1,760.80	14,234.72	225.76	119.82	1.8604	1.69	0.0107	104.09	0.9659
		500	248.17 + 1,818.70	14,028.80	214.91	149.73	0.8344	1.67	0.0183	63.48	0.5783
		1,000	292.40 + 1,817.20	15,516.68	232.11	193.18	1.1041	1.72	0.0135	37.20	0.3176
		2,000	371.12 + 1,841.40	16,121.77	306.42	268.78	2.3209	1.82	0.0165	35.81	0.3680
		5,000	589.99 + 1,846.40	15,384.47	524.63	483.90	6.7403	1.93	0.0160	38.80	0.5411

proposal (with local proposal choice β = 0) since such a setting was reported by Eaton and Murphy (2007) to have the best performance for edge discovery when up to about $190,000$ MCMC iterations were performed in their experimental results. We ran totally 190,000 MCMC iterations each time and discarded the first 100,000 iterations as the burn-in period. Then we set the thinning parameter to be 3 to get the final 30,000 DAG samples. As a result, the time statistics of the DP phase (the number before + sign) but not the MCMC phase (the number after + sign) can be directly compared with the ones of the other two methods. For each data case, we performed 20 independent MCMC runs based on the DP outcome from REBEL to get the results.

For our method, we set $N_o = 30,000$. We performed 20 independent runs for each data case to get the results. For the K-best method, note that its SAD is fixed since there is no randomness in the

computed results. So we only ran it once to get the result. We set K to be 100 for Wine, Housing, Zoo, Syn15, and Letter, that is, we got the 100 best DAGs from Wine, Housing, Zoo, each of the six cases of Syn15, and each of the six cases of Letter. We set K to be only 40 for Insur19 because our experiments showed that for Insur19 the K-best program ran out of memory with the setting of $K > 40$. For the same out-of-memory issue, We set K to be only 20 for Child and only 9 for German. The fact that K can only take a very small value for $n \geq 19$ confirms our claim about the computation problem of the K-best method in terms of its space cost.

Table 4.3 shows experimental results in terms of SAD for each data case while Table 4.4 shows experimental results in terms of the total running time T_t corresponding to Table 4.3. The column named DP in Table 4.3 shows the SAD ($\sum_{ij} |p_{\not\prec}(i \to j|D) - p_{\prec}(i \to j|D)|$), where each edge posterior $p_{\not\prec}(i \to j|D)$ is computed by the exact DP method of Tian and He (2009), and each edge posterior $p_{\prec}(i \to j|D)$ is computed by the exact DP method of Koivisto (2006) with the prior setting $\rho_i(Pa_i) \equiv 1$. The column named DP' in Table 4.3 shows the SAD ($\sum_{ij} |p_{\not\prec}(i \to j|D) - p_{\prec}(i \to j|D)|$), where each edge posterior $p_{\not\prec}(i \to j|D)$ is still computed by the exact DP method of Tian and He (2009), but each edge posterior $p_{\prec}(i \to j|D)$ is computed by the exact DP method of Koivisto (2006) with the prior setting $\rho_i(Pa_i) = 1/\binom{n-1}{|Pa_i|}$. The SAD values reported in these two columns indicate the bias due to the assumption of the order-modular prior. Since the SAD in the DP column is smaller than the one in the DP' column for 24 out of 28 cases, we use the prior setting $\rho_i(Pa_i) \equiv 1$ for the DP phase so that the DP+MCMC method can have better performance. Next to the DP and DP' columns, the SAD values of the three methods are listed in Table 4.3. Both the DP+MCMC method and the IW-DDS method are random so that both $\hat{\mu}$(SAD) and $\hat{\sigma}$(SAD) are shown for these two methods. The outcome of the K-best method is not random so that only its SAD is shown. Finally Table 4.3 also shows Δ for both the K-best method and the IW-DDS method.

Tables 4.3 and 4.4 clearly demonstrate the advantage of our method over the other two methods. By using much shorter computation time, $\hat{\mu}$(SAD) using our method is less than the corresponding one using the DP+MCMC method for 27 out of the 28 data cases. The only exceptional case is Syn15 with $m = 200$. Furthermore, based on the two-sample t test with unequal variances, we can conclude at the significance level 0.05 that the real mean of SAD using our method is less than the corresponding one using the DP+MCMC method for each of the 26 cases. The two exceptional cases are Syn15 with

$m = 100$ and Syn15 with $m = 200$. Meanwhile, $\hat{\sigma}$(SAD) using our method is always much smaller than the one using the DP+MCMC for each of the 28 cases, which indicates higher stability of the performance of our method. Similarly, using much shorter computation time, $\hat{\mu}$(SAD) of our method is less than the one using the K-best method for 27 out of 28 cases. The only exception is Syn15 with m = 5,000. Furthermore, based on the one-sample t test (Casella and Berger, 2002), we can conclude at the significance level 5×10^{-4} that the real mean of SAD using our method is less than the corresponding one using the K-best method for each of these 27 cases.

There are several other interesting things shown in Tables 4.3 and 4.4. In terms of the SAD, for very small m, $\hat{\mu}$(SAD) using the DP+MCMC method is even larger than the SAD from the DP phase (Koivisto, 2006) itself. For example, for the data set Zoo, the SAD from the DP phase is 8.2142 but $\hat{\mu}$(SAD) obtained after the MCMC phase of the DP+MCMC method is 32.4189. Similar situations also occur in Syn15, Letter, Insur19, and Child when $m = 100$. This indicates that for very small m, the MCMC phase of the DP+MCMC method is unable to reduce the bias from the DP method (Koivisto, 2006) for all these cases based on $190,000$ MCMC iterations. With regard to the running time, please note that $\hat{\mu}(T_{DP})$ of our IW-DDS is always less than the running time of the DP phase of the DP+MCMC method. This is because the DP step of our method uses the DP algorithm of Koivisto and Sood (2004), that is, the first three steps of the DP algorithm of Koivisto (2006); while the DP phase of the DP+MCMC method uses all the five steps of the DP algorithm of Koivisto (2006). In other words, compared with the DP algorithm of Koivisto and Sood (2004), the DP algorithm of Koivisto (2006) includes a larger constant factor hidden in the $O(n^{k+1}C(m) + kn2^n)$ notation though these two DP algorithms have the same time complexity. This difference will make the total running time of our IW-DDS even less than the running time of the DP phase of the DP+MCMC method when the remaining steps of the IW-DDS run faster than the last two steps of the DP algorithm of Koivisto (2006). For example, for the data set Child with $m = 5,000$, $\hat{\mu}(T_t)$ of the IW-DDS is 524.63 seconds while the corresponding running time of the DP phase of the DP+MCMC method is 589.99 seconds. Actually, Table 4.4 shows that there are 20 out of the 28 cases where $\hat{\mu}(T_t)$ of the IW-DDS is less than the running time of the DP phase of the DP+MCMC method. In addition, just as shown in Subsection 4.4.1, the effectiveness of our time-saving strategy can also be clearly seen from Table 4.4. For example, the ratio of $\hat{\mu}(T_{DAG})$ to $\hat{\mu}(T_{ord})$ ranges from 0.07 to 87.29 across the 28 cases, which is much smaller than the

upper bound of the ratio of $O(n^{k+1}N_o)$ to $O(n^2N_o)$.

Table 4.3 also shows the resulting Δ from the K-best method and our IW-DDS method for all the data cases. (Note that Δ is computed by the formula $\Delta = \sum_{G \in \mathcal{G}} p_{\not\perp}(G, D)/p_{\not\perp}(D)$, where $p_{\not\perp}(D)$ is computed using POSTER tool.) In the table, $\hat{\mu}(\Delta)$ from our IW-DDS is greater than Δ from the K-best method for 24 out of 28 data cases. The four exceptional cases are Zoo, Syn15 with $m = 100$, Syn15 with $m = 5,000$, and Letter with $m = 100$. Interestingly, for three out of the four exceptional cases (as well as the other 24 cases), $\hat{\mu}$(SAD) from our IW-DDS is significant smaller than SAD from the K-best method. One possible reason is that the K best DAGs tend to have the same or similar local structures (family (i, Pa_i)'s) that have relatively large local scores while a large number of DAGs sampled from our IW-DDS include various local structures for each node i. When Δ is far from 1, the inclusion of various local structures seems to be more effective in improving the structural learning performance.

Table 4.3 also shows that when m is not very small (such as no smaller than 500 or $1,000$), Δ from our IW-DDS with $N_o = 30,000$ can reach a large percentage (such as greater than 90%). As a result, we can obtain the sound interval of $p_{\not\perp}(f|D)$ with the width less than 0.1 for any feature f.

To further demonstrate that our IW-DDS method can obtain a large Δ efficiently when m is not very small, we increased N_o from $100,000$ to $600,000$ with each increment $100,000$ to see its performance for the data sets Letter with $m = 500$, Child with $m = 500$, and German. Again, we performed 20 independent runs for each data case to get the results. Figures 4.1, 4.3 and 4.5 show the increase in Δ with respect to the increase in N_o for these three data cases. Correspondingly, Figures 4.2, 4.4 and 4.6 indicate the increase in $\hat{\mu}(T_t)$, $\hat{\mu}(T_{ord})$ and $\hat{\mu}(T_{DAG})$ with respect to the increase in N_o for these three data cases, where the running time is in seconds. Combining these figures, it is clear that our IW-DDS can efficiently achieve a large Δ. Take the data set German for example, with the time cost $\hat{\mu}(T_t)$ = $1,493.02$ seconds, our IW-DDS can collect $N_o = 600,000$ DAG samples so that the corresponding mean of Δ can reach 91.74%. Therefore, for any feature f in the data set German, our IW-DDS can provide the sound interval of $p_{\not\perp}(f|D)$ with the width less than 0.083. Note that the K-best method can only provide the sound interval of $p_{\not\perp}(f|D)$ with the huge width 0.922 because its corresponding Δ can only reach 0.078 before running out of the memory. Also note that the ratio of $\hat{\mu}(T_{DAG})$ to $\hat{\mu}(T_{ord})$ decreases from 56.45 to 30.95 when N_o increases from $100,000$ to $600,000$. (The increase rate of $\hat{\mu}(T_{ord})$ is a constant with respect to N_o; but the increase rate of $\hat{\mu}(T_{DAG})$ actually decreases as

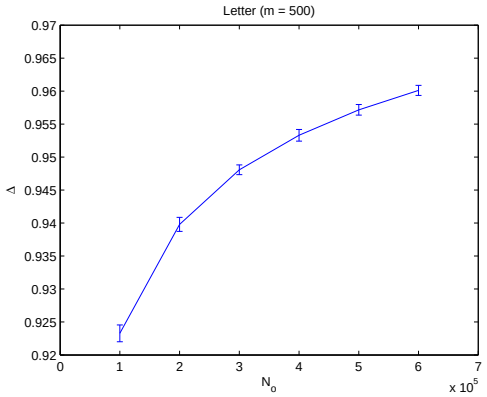


Figure 4.1: Plot of Δ versus N_o for Letter ($m = 500$)

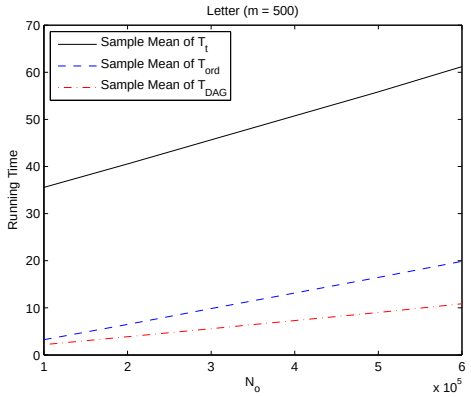


Figure 4.2: Plot of the Running Time versus N_o for Letter ($m = 500$)

N_o increases.) This witnesses the statement in Remark 1 that the benefit from our time-saving strategy will typically increase when N_o increases. In addition, the ratio of $\hat{\mu}(T_{DAG})$ to $\hat{\mu}(T_{ord})$ is always much smaller than the upper bound of the ratio of $O(n^{k+1}N_o)$ to $O(n^2 N_o)$, which indicates the effectiveness of our time-saving strategy for the DAG sampling step.

As for the data set Insur19 with small $m = 200$, Figures 4.7 and 4.8 show that our IW-DDS method can use the memory-saving strategy introduced in Section 4.3.3 to efficiently collect large N_o DAGs samples and obtain a large Δ. We increased N_o from 2×10^6 to 1.2×10^7 with each increment 2×10^6 and showed the corresponding change of Δ and the running time in Figures 4.7 and 4.8. (Again, 20 independent runs were performed for each data case to get the results.) By temporarily storing the sampled DAGs in the hard disk, our IW-DDS can sample $N_o = 1.2^7$ DAGs so that the resulting mean of Δ can achieve 95.39% with the time cost $\hat{\mu}(T_t) = 1,933.44$ seconds. Note that the ratio of $\hat{\mu}(T_{DAG})$ to $\hat{\mu}(T_{ord})$ is around the constant 2.07 across the different values of N_o since we fixed $N_{bl} = 2 \times 10^6$ as the size of the sample block.

4.4.3 Learning Performance of Non-modular Features

In Sections 4.4.1 and 4.4.2 we did not provide experimental results on the learning performance of non-modular features. We did not do so in Section 4.4.2 because there is no known method to compute the true/exact posterior probability of any non-modular feature $p_{\not\prec}(f|D)$ except by the brute force

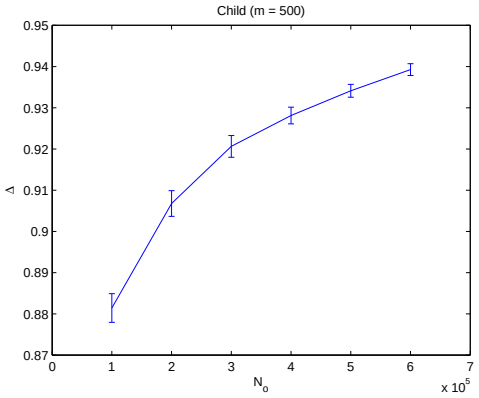


Figure 4.3: Plot of Δ versus N_o for Child ($m = 500$)

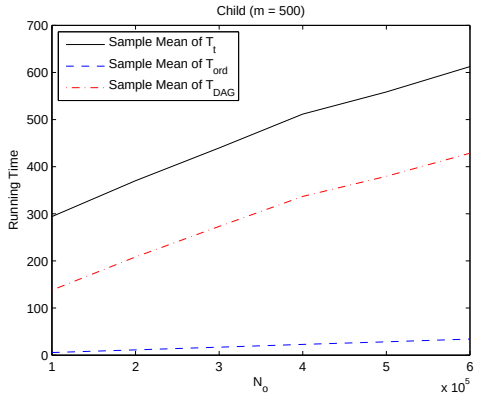


Figure 4.4: Plot of the Running Time versus N_o for Child ($m = 500$)

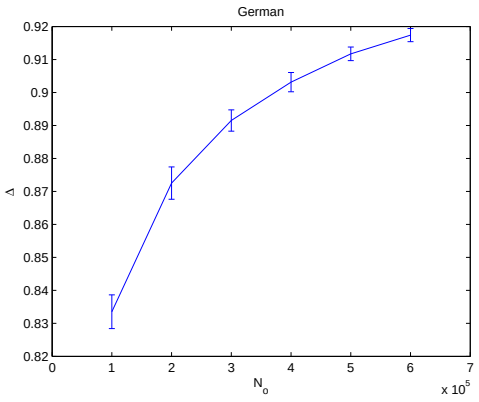


Figure 4.5: Plot of Δ versus N_o for German

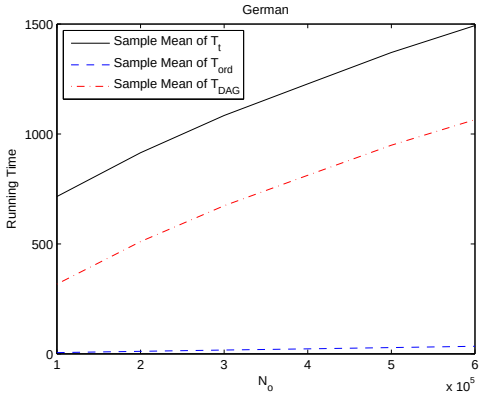


Figure 4.6: Plot of the Running Time versus N_o for German

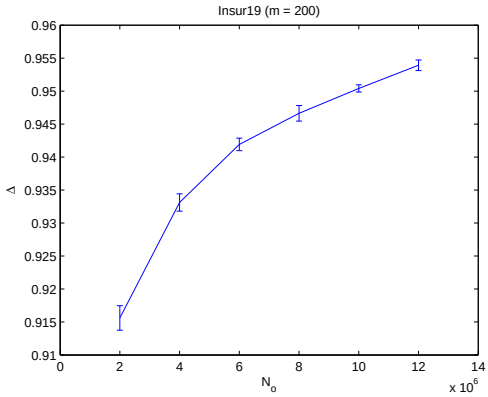


Figure 4.7: Plot of Δ versus N_o for Insur19 ($m = 200$)

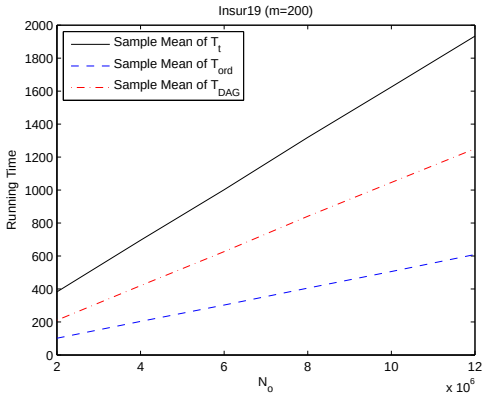


Figure 4.8: Plot of the Running Time versus N_o for Insur19 ($m = 200$)

enumeration over all the (super-exponential number of) DAGs so that the quality of the corresponding $\hat{p}_{\not\prec}(f|D)$ learned from any method cannot be precisely measured. We did not do so in Section 4.4.1 because the current PO-MCMC tool (BEANDisco) only supports the estimation of the posterior of an edge feature so that the comparison of our method and the PO-MCMC can only be made for the edge feature. (Thus, we did not make the comparison for the path feature (which is one particular non-modular feature), though the DP algorithm of Parviainen and Koivisto (2011) can compute the exact posterior of a path feature $p_{\prec}(f|D)$.) Our idea is that by showing that our algorithms have significantly better performance in computing one feature (a directed edge feature), which should be due to the better quality of our DAG samples with respect to the corresponding $p_{\not\prec}(G|D)$ or $p_{\prec}(G|D)$, we expect that they will also be superior in computing other features using the same set of DAG samples.

To verify our expectation, we performed the experiments on the real data set "Iris" (with $n = 5$) from the UCI Machine Learning Repository (Asuncion and Newman, 2007) and the well-studied data set "Coronary" (Coronary Heart Disease) (with $n = 6$) (Edwards, 2000). Since n is small, by enumerating all the DAGs we were able to compute $p_{\not\prec}(f_i|D)$, the true posterior probability for several interesting non-modular features: f_1, a directed path feature; f_2, a limited-length directed path feature that has its path length no more than 2; f_3, a combined directed path feature that represents a directed path from node i via node j to node k. Then we compared the SAD performance of the (directed) edge feature from the DP+MCMC, the K-best and the IW-DDS with the corresponding SAD performance

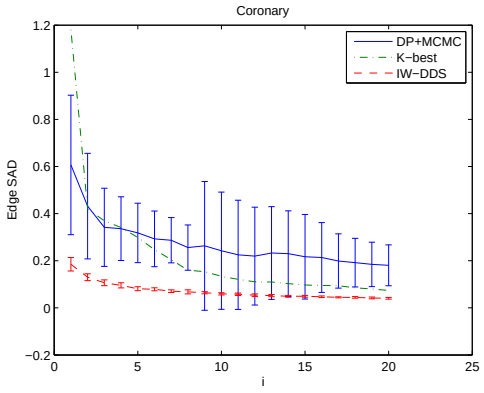


Figure 4.9: SAD of the Learned Edge Features for Coronary

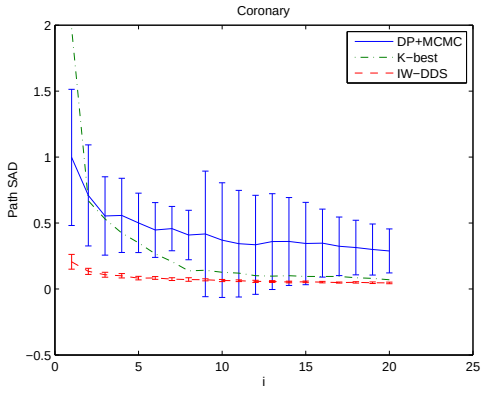


Figure 4.10: SAD of the Learned Path Features for Coronary

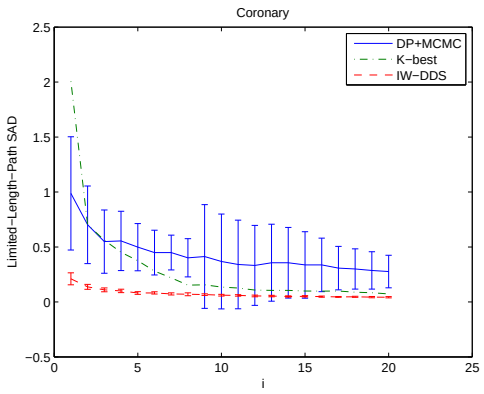


Figure 4.11: SAD of the Learned Limited-length-path Features for Coronary

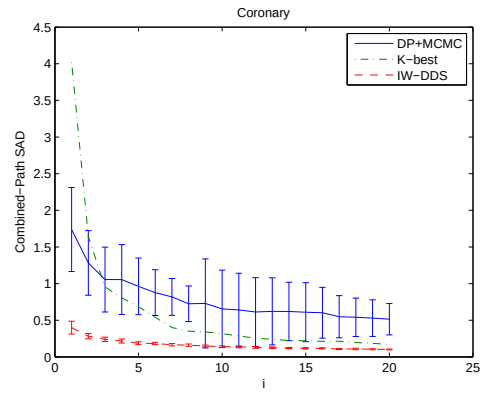


Figure 4.12: SAD of the Learned Combined-path Features for Coronary

of feature f_i ($i \in \{1, 2, 3\}$) from these three methods. The experimental results on both data sets show that if the SAD of the IW-DDS is significantly smaller than the SAD of the competing method (the DP+MCMC or the K-best) for the edge feature, then the SAD of the IW-DDS will also be smaller (usually significantly smaller) than the SAD of the competing method for each feature f_i. Thus, our expectation is supported by the experiments. The detailed experimental results are as follows.

The following is our experimental design for the data set Coronary with $n = 6$ and $m = 1841$. For the IW-DDS, we tried the sample size $N_o = 1,000 \cdot i$, where $i \in \{1, 2, \ldots, 20\}$. For each i, we independently ran the IW-DDS 20 times to get the sample mean and sample standard deviation of SAD

for the (directed) edge feature and three non-modular features (f_1, f_2, and f_3). For the DP+MCMC, we ran $10,000 \cdot i$ MCMC iterations, where $i \in \{1, 2, \ldots, 20\}$. For each i, we discarded the first $5,000 \cdot i$ MCMC iterations for "burn-in" and set the thinning parameter to be 5 so that $1,000 \cdot i$ DAGs got sampled. Again, for each i, we independently ran the MCMC 20 times to get the sample mean and sample standard deviation of SAD for the edge feature, f_1, f_2, and f_3. For the K-best method, we ran the K-best with $K = 10 \cdot i$, where $i \in \{1, 2, \ldots, 20\}$. For each i, we ran the K-best just once to get SAD for the edge feature, f_1, f_2, and f_3 since there is no randomness in the outcome of the K-best algorithm.

Figure 4.9 shows the SAD performance of the three methods with each i for the edge feature, where an error bar represents one sample standard deviation across 20 runs for the DP+MCMC or the IW-DDS at each i. Correspondingly, Figure 4.10 shows the SAD performance ($\sum_{xy} |p_{\not\leftarrow}(x \rightsquigarrow y|D) - \hat{p}_{\not\leftarrow}(x \rightsquigarrow y|D)|$) of the three methods with each i for the path feature $x \rightsquigarrow y$; Figure 4.11 shows the SAD performance ($\sum_{xy} |p(x \rightsquigarrow y|D) - \hat{p}(x \rightsquigarrow y|D)|$) of the three methods with each i for the path feature $x \rightsquigarrow y$ whose length is no more than 2; Figure 4.12 shows the SAD performance ($\sum_{xyz} |p_{\not\leftarrow}(x \rightsquigarrow y \rightsquigarrow z|D) - \hat{p}_{\not\leftarrow}(x \rightsquigarrow y \rightsquigarrow z|D)|$) of the three methods with each i for the combined path feature $x \rightsquigarrow y \rightsquigarrow z$. Combining Figure 4.9 and each of Figures 4.10, 4.11, and 4.12, one can clearly see that if the SAD of the IW-DDS is significantly smaller than the SAD of the competing method (the DP+MCMC or the K-best) for the edge feature, then the SAD of the IW-DDS will also be significantly smaller than the SAD of the competing method for each of the three investigated non-modular features.

We also performed the same experiments for the data set Iris with $n = 5$ and $m = 150$. The results are shown in Figures 4.13, 4.14, 4.15, and 4.16. A conclusion similar to the one from Coronary can be drawn to support our expectation: if the SAD of the IW-DDS is significantly smaller than the SAD of the competing method for the edge feature, then the SAD of the IW-DDS will also be smaller than the SAD of the competing method for each of the three investigated non-modular features and such a SAD decrease due to the IW-DDS is usually significant.

4.4.4 Performance Guarantee for the DDS Algorithm

To testify the quality guarantee for the estimator based on the DDS algorithm (Corollary 1 (iv)), we performed experiments based on two data cases (Syn15 with $m = 100$ and Letter with $m = 100$),

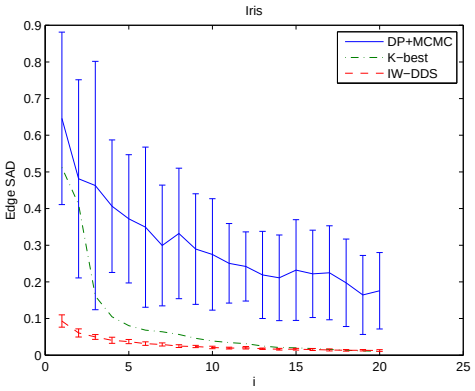


Figure 4.13: SAD of the Learned Edge Features for Iris

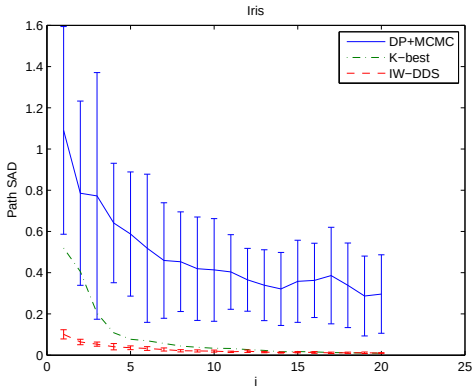


Figure 4.14: SAD of the Learned Path Features for Iris

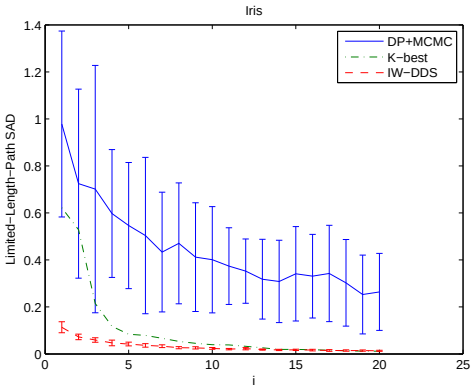


Figure 4.15: SAD of the Learned Limited-length-path Features for Iris

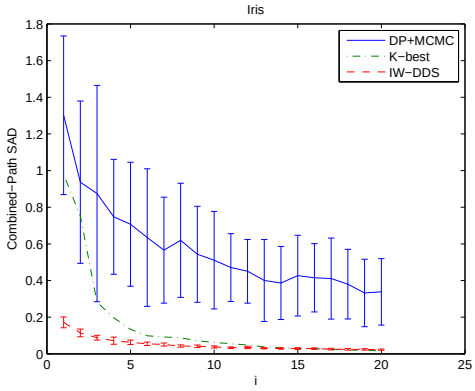


Figure 4.16: SAD of the Learned Combined-path Features for Iris

which have relatively large $\hat{\mu}$(SAD) or $\hat{\mu}$(MAD) ($= \hat{\mu}$(SAD) $/(n(n-1))$) shown in Table 4.1. Based on the hypothesis testing, we can conclude with very strong evidence that performance guarantee for our estimator holds for both data cases. The details of the experiments are as follows.

For the first set of experiments, we choose the data set Letter with $m = 100$, which has the largest $\hat{\mu}$(SAD) shown in Table 4.1. We first consider the setting of the parameters specified as $\epsilon = 0.02$ and $\delta = 0.05$, which serves as our performance requirement. By setting the DAG sample size $N_o = \lceil(\ln(2/\delta))/(2\epsilon^2)\rceil = 4612$, we intend to show that the estimator $\hat{p}_{\prec}(f|D)$ coming from our DDS has the performance guarantee such that the Hoeffding inequality $p(|\hat{p}_{\prec}(f|D) - p_{\prec}(f|D)| \geq \epsilon) \leq \delta$ holds. Each directed edge feature f is investigated here since the posterior of each edge $p_{\prec}(f|D)$ can be easily obtained by using the DP algorithm of Koivisto (2006). Meanwhile, the corresponding $\hat{p}_{\prec}(f|D)$ based on N_o $(= 4612)$ DAG samples can be obtained by our DDS algorithm. For each edge f, we call the event of $|\hat{p}_{\prec}(f|D) - p_{\prec}(f|D)| \geq \epsilon$ as the event of violation (of the pre-specified estimation error bound) in the learning of f. Define the indication variable W for the event of violation in the learning of f. Thus, W is a Bernoulli random variable with the success probability $p_{vio} = p(|\hat{p}_{\prec}(f|D) - p_{\prec}(f|D)| \geq \epsilon)$. We independently repeat the DDS algorithm (with the same N_o) $R = 400$ times and use the average of W as the estimator $\hat{p}_{vio}$ for each edge. Note that the mean of $\hat{p}_{vio}$ is p_{vio} and the variance of $\hat{p}_{vio}$ is $p_{vio}(1 - p_{vio})/R$ because $R\hat{p}_{vio}$ has a binomial distribution with the trial number R and success probability p_{vio}. Since we expect that p_{vio} will be small so that $p_{vio}(1 - p_{vio})$ will be large relative to p_{vio}, we choose large $R = 400$ to make the variance of $\hat{p}_{vio}$ relatively small with respect to the mean of $\hat{p}_{vio}$. Figure 4.17 shows the histogram of $\hat{p}_{vio}$ for each of all the $n(n-1) = 272$ directed edges. For each of the 272 edges, it can be clearly seen that the corresponding $\hat{p}_{vio}$ is much less than $\delta = 0.05$ marked by the vertical bar. For 257 out of the 272 edges, the corresponding $\hat{p}_{vio}$'s are exactly equal to 0. Even for the largest $\hat{p}_{vio} = 0.015$, corresponding to 6 successes among 400 trials, we can use the one-sided hypothesis testing to reject the null hypothesis that $p_{vio} = 0.05$ and to conclude that $p_{vio} < 0.05$ with the p-value less than 2×10^{-4}. Therefore, the Hoeffding inequality holds for the learning of each edge in this parameter setting.

Next, we consider another setting of the parameters with $\epsilon = 0.01$ and $\delta = 0.02$, which has more demanding performance requirement. By setting the DAG sample size $N_o = \lceil(\ln(2/\delta))/(2\epsilon^2)\rceil = 23026$, we want to show that the estimator $\hat{p}_{\prec}(f|D)$ coming from our DDS has the performance guarantee sat-

isfying the Hoeffding inequality. With the same logic, We independently repeat the DDS algorithm (with the same N_o) $R = 1250$ times and use the average of W as the estimator $\hat{p}_{vio}$ for each edge. (Here we choose even larger R since we expect that p_{vio} will be smaller in this parameter setting.) Figure 4.18 shows the histogram of $\hat{p}_{vio}$ for each of all the 272 directed edges. For each edge, it can be clearly seen that the corresponding $\hat{p}_{vio}$ is much less than $\delta = 0.02$. Even for the largest $\hat{p}_{vio} = 0.004$, corresponding to 5 successes among 1250 trials, we can use the one-sided hypothesis testing to reject the null hypothesis that $p_{vio} = 0.02$ and to conclude that $p_{vio} < 0.02$ with the p-value less than 2×10^{-6}. Therefore, the Hoeffding inequality also holds in this parameter setting.

For the second set of experiments, we choose the data case Syn15 with $m = 100$, which has the largest $\hat{\mu}$(MAD) ($= \hat{\mu}$(SAD) $/(n(n-1))$) shown in Table 4.1. The same experiments are also performed for this data case. For the parameter setting with $\epsilon = 0.02$ and $\delta = 0.05$, the corresponding result is shown in Figure 4.19. For each of the 210 edges, the corresponding $\hat{p}_{vio}$ is clearly much less than $\delta = 0.05$. Even for the largest $\hat{p}_{vio} = 0.01$, corresponding to 4 successes among 400 trials, we can use the one-sided hypothesis testing to conclude that $p_{vio} < 0.05$ with the p-value less than 2×10^{-5}. For the parameter setting with $\epsilon = 0.01$ and $\delta = 0.02$, the corresponding result is shown in Figure 4.20. For each edge, the corresponding $\hat{p}_{vio}$ can be clearly seen to be much less than $\delta = 0.02$. Even for the largest $\hat{p}_{vio} = 0.004$, corresponding to 5 successes among 1250 trials, we can use the one-sided hypothesis testing to conclude that $p_{vio} < 0.02$ with the p-value less than 2×10^{-6}. Thus, the Hoeffding inequality also holds in the set of experiments for this data case.

Finally, for the data case Syn15 with $m = 100$, we fix $\epsilon = 0.02$ but increase N_o from $2,000$ to $12,000$ by an increment of $2,000$ each time. For each N_o, we plot the Hoeffding bound $2e^{-2N_o\epsilon^2}$ for the probability of violation $p_{vio} = p(|\hat{p}_{\prec}(f|D) - p_{\prec}(f|D)| \geq \epsilon)$ in Figure 4.21. (Note that the Hoeffding bound decreases at an exponential rate as N_o increases.) Then for each N_o, we also plot both the maximum and the mean of all the $n(n-1)$ $\hat{p}_{vio}$'s in Figure 4.21. Again $\hat{p}_{vio}$ for each edge is the average of W by independently running the DDS algorithm (with the same N_o) R times. We set $R = max\{400, N_o/10\}$ and use the larger R for the larger N_o since we expect that $\hat{p}_{vio}$ will be smaller for the larger N_o. From Figure 4.21, we can clearly see that $\hat{p}^*_{vio}$, the maximum of all the $n(n-1)$ $\hat{p}_{vio}$'s, is always far below the Hoeffding bound for each N_o. Furthermore, for each N_o we can use the one-sided hypothesis testing to reject the null hypothesis that $p_{vio} \geq 2e^{-2N_o\epsilon^2}$ and to conclude that

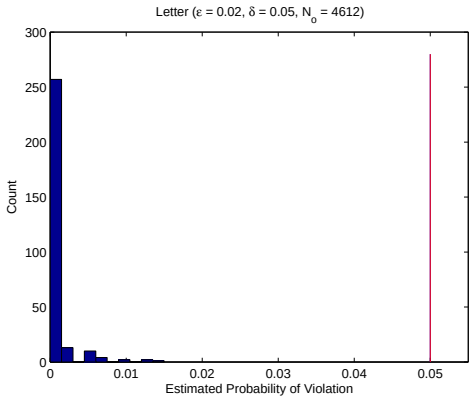


Figure 4.17: Histogram of Estimated Probabilities of Violation in Edge Learning for Letter ($m = 100$) with $\epsilon = 0.02$, $\delta = 0.05$ and $N_o = 4612$

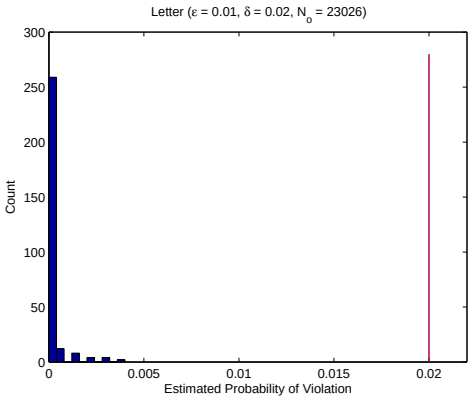


Figure 4.18: Histogram of Estimated Probabilities of Violation in Edge Learning for Letter ($m = 100$) with $\epsilon = 0.01$, $\delta = 0.02$ and $N_o = 23026$

$p_{vio} < 2e^{-2N_o\epsilon^2}$ with the p-value less than 3×10^{-3}. Therefore, the Hoeffding inequality holds in the learning for each N_o.

4.5 Conclusion

We develop new algorithms for efficiently sampling Bayesian network structures (DAGs) of moderate size. The sampled DAGs can then be used to build estimators for the posteriors of any features. Our algorithms address the limitations of the exact algorithms (Koivisto and Sood, 2004; Parviainen and Koivisto, 2011; Tian and He, 2009) so that the posteriors of arbitrary non-modular features can be estimated when the size of the Bayesian network is moderate. We theoretically prove good properties of our estimators. Finally we empirically show that the performance of our estimators is considerably better than the performance of the estimators from previous state-of-the-art methods with or without assuming the order-modular prior.

4.6 Appendix: Proofs of Propositions, Theorems and Corollary

This appendix provides the proofs of the propositions, theorems and corollary referenced in the paper.

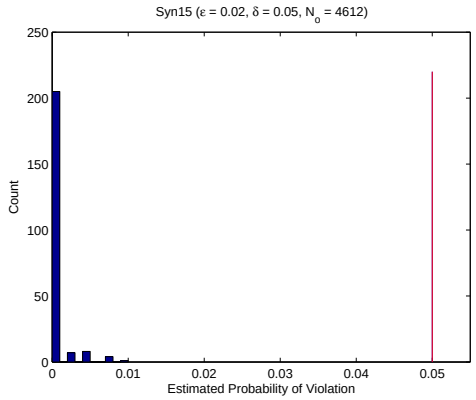


Figure 4.19: Histogram of Estimated Probabilities of Violation in Edge Learning for Syn15 ($m = 100$) with $\epsilon = 0.02$, $\delta = 0.05$ and $N_o = 4612$

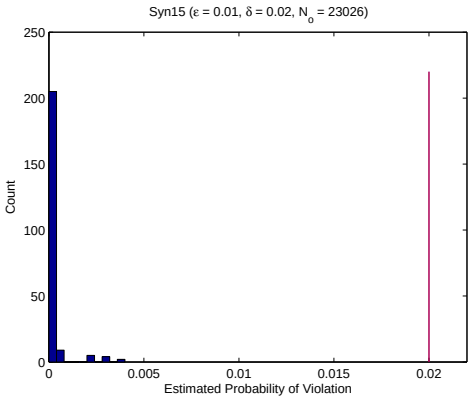


Figure 4.20: Histogram of Estimated Probabilities of Violation in Edge Learning for Syn15 ($m = 100$) with $\epsilon = 0.01$, $\delta = 0.02$ and $N_o = 23026$

Proof of Proposition 5

We first prove a lemma for Proposition 1.

Let an order $\prec$ be represented as $(\sigma_1, \ldots, \sigma_n)$ where σ_i is the ith element in the order.

Lemma 1 *The probability that the last $n - k + 1$ elements along the order are $\sigma_k, \sigma_{k+1}, \ldots, \sigma_n$ respectively is given by*

$$p(\sigma_k, \sigma_{k+1}, \ldots, \sigma_n, D) = L'(U^{\prec}_{\sigma_k}) \prod_{i=k}^{n} \alpha'_{\sigma_i}(U^{\prec}_{\sigma_i}),$$

where $U^{\prec}_{\sigma_i} = V - \{\sigma_i, \sigma_{i+1}, \ldots, \sigma_n\}$.

Proof:

$$\begin{aligned}
& p(\sigma_k, \sigma_{k+1}, \ldots, \sigma_n, D) \\
&= \sum_{\prec' \in \mathcal{L}(U^{\prec}_{\sigma_k})} p(\prec', \sigma_k, \sigma_{k+1}, \ldots, \sigma_n, D) \\
&= \prod_{i=k}^{n} \alpha'_{\sigma_i}(U^{\prec}_{\sigma_i}) \sum_{\prec' \in \mathcal{L}(U^{\prec}_{\sigma_k})} \prod_{i \in U^{\prec}_{\sigma_k}} \alpha'_i(U^{\prec}_i) \quad \text{(from (4.13))} \\
&= L'(U^{\prec}_{\sigma_k}) \prod_{i=k}^{n} \alpha'_{\sigma_i}(U^{\prec}_{\sigma_i}). \quad \text{(from (4.9))}
\end{aligned}$$

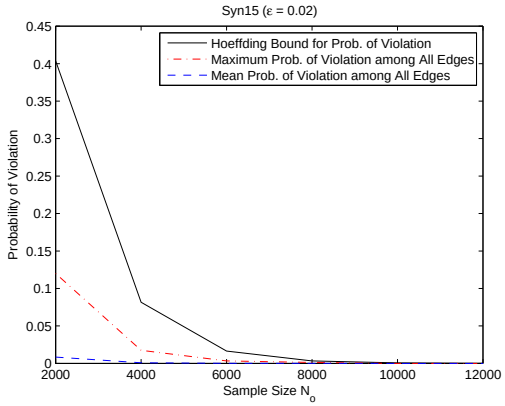


Figure 4.21: Plot of Probability of Violation versus N_o for Syn15 ($m = 100$) with $\epsilon = 0.02$

□

Proposition 5 can be directly proved from the conclusion of Lemma 1 according to the definition of conditional probability.

Proof of Proposition 6

Proof:

From Proposition 5 and $L'(U^{\prec}_{\sigma_1} = \emptyset) = 1$ we have

$$\begin{aligned}
&\prod_{k=1}^{n} p(\sigma_k|\sigma_{k+1}, \ldots, \sigma_n, D) \\
&= \frac{\prod_{i=1}^{n} \alpha'_{\sigma_i}(U^{\prec}_{\sigma_i})}{L'(V)} \\
&= \frac{p(\prec, D)}{p_{\prec}(D)} \quad \text{(from (4.13) \& (4.14))} \\
&= p(\prec |D),
\end{aligned}$$

which proves Eq. (4.20). □

Proof of Theorem 3

Proof:

First, we show that each DAG G sampled according to our DDS algorithm has its pmf $p_s(G)$ equal to the exact posterior $p_{\prec}(G|D)$ by assuming the order-modular prior.

On one hand, from the following derivation, we can get the exact form for $p_{\prec}(G|D)$:

$$\begin{aligned} & p_{\prec}(f|D) \\ &= \sum_{\prec} p(\prec|D)p(f|\prec, D) \\ &= \sum_{\prec} p(\prec|D) \sum_{G\subseteq\prec} f(G)p(G|\prec, D) \\ &= \sum_{\prec} \sum_{G\subseteq\prec} f(G)p(\prec, G|D) \\ &= \sum_{G} f(G) \sum_{\prec s.t. G\subseteq\prec} p(\prec, G|D). \end{aligned} \tag{4.27}$$

Thus, for each possible DAG G_i, by setting $f(G)$ to be the indication function $I[G = G_i]$ and then relating Eq. (4.27) with Eq. (4.4), we know that $p_{\prec}(G_i|D) = \sum_{\prec s.t. G_i\subseteq\prec} p(\prec, G_i|D)$ for each G_i.

On the other hand, the event that a DAG G gets sampled according to our DDS algorithm occurs if and only if one of the orders that G is consistent with gets sampled in Step 2 of the DDS algorithm and then G gets sampled from the sampled order in Step 3 of the DDS algorithm. Therefore, based on Total Probability Formula, $p_s(G) = \sum_{\prec s.t. G\subseteq\prec} p(\prec|D)p(G|\prec, D) = \sum_{\prec s.t. G\subseteq\prec} p(\prec, G|D) = p_{\prec}(G|D)$.

Second, since N_o orders are sampled independently and each DAG per sampled order is sampled independently, $p_s(G_1, G_2, \ldots, G_{N_o}|D) = \prod_{i=1}^{N_o}[\sum_{\prec s.t. G\subseteq\prec} p(\prec|D)p(G_i|\prec, D)] = \prod_{i=1}^{N_o} p_{\prec}(G_i|D)$.

Therefore, Theorem 3 is proved.

□

Proof of Corollary 1

Proof:

For each G_i in the DAG set $\{G_1, G_2, \ldots, G_{N_o}\}$ sampled from the DDS algorithm, $f(G_i) \in \{0, 1\}$. Since $G_1, G_2, \ldots, G_{N_o}$ are iid with pmf $p_{\prec}(G|D)$ (by Theorem 3), $f(G_1), f(G_2), \ldots, f(G_{N_o})$ are iid with Bernoulli pmf.

For each G_i, the following is true for $E(f(G_i))$, the expectation of $f(G_i)$.

$$\begin{aligned} & E(f(G_i)) \\ & = \sum_G f(G) p_\prec(G|D) \\ & = p_\prec(f|D). \end{aligned}$$

Thus, $f(G_1), f(G_2), \ldots, f(G_{N_o})$ are iid with Bernoulli($p_\prec(f|D)$). In other words, $f(G_1), f(G_2), \ldots, f(G_{N_o})$ are independent; and for each G_i, $P(f(G_i) = 1) = p_\prec(f|D)$ and $P(f(G_i) = 0) = 1 - p_\prec(f|D)$.

(i) Proof that $\hat{p}_\prec(f|D)$ is an unbiased estimator for $p_\prec(f|D)$, that is, $E(\hat{p}_\prec(f|D)) = p_\prec(f|D)$.

$$\begin{aligned} & E\left(\hat{p}_\prec(f|D)\right) \\ & = E\left(\frac{1}{N_o}\sum_{i=1}^{N_o} f(G_i)\right) \\ & = \frac{1}{N_o}\sum_{i=1}^{N_o} E\left(f(G_i)\right) \\ & = \frac{1}{N_o} N_o p_\prec(f|D) \\ & = p_\prec(f|D). \end{aligned}$$

(ii) Proof that $\hat{p}_\prec(f|D)$ converges almost surely to $p_\prec(f|D)$.

Since $f(G_1), f(G_2), \ldots, f(G_{N_o})$ are iid with $E(f(G_i)) = p_\prec(f|D) < \infty$, and since

$$\hat{p}_\prec(f|D) = \frac{1}{N_o}\sum_{i=1}^{N_o} f(G_i),$$

based on the Strong Law of Large Numbers (Theorem 5.5.9) (Casella and Berger, 2002), $\hat{p}_\prec(f|D)$ converges almost surely to $p_\prec(f|D)$ (as $N_o \to \infty$).

Note that, by Theorem 2.5.1 (Athreya and Lahiri, 2006), the property that $\hat{p}_\prec(f|D)$ converges almost surely to $p_\prec(f|D)$ implies that $\hat{p}_\prec(f|D)$ converges in probability to $p_\prec(f|D)$, that is, $\hat{p}_\prec(f|D)$ is a consistent estimator for $p_\prec(f|D)$.

(iii) Proof that if $0 < p_\prec(f|D) < 1$, then the random variable

$$\frac{\sqrt{N_o}(\hat{p}_\prec(f|D) - p_\prec(f|D))}{\sqrt{\hat{p}_\prec(f|D)(1 - \hat{p}_\prec(f|D))}}$$

has a limiting standard normal distribution, denoted by

$$\frac{\sqrt{N_o}(\hat{p}_{\prec}(f|D) - p_{\prec}(f|D))}{\sqrt{\hat{p}_{\prec}(f|D)(1 - \hat{p}_{\prec}(f|D))}} \longrightarrow^d \mathcal{N}(0,1).$$

Since $f(G_1)$, $f(G_2), \ldots, f(G_{N_o})$ are iid with Bernoulli($p_{\prec}(f|D)$), for each G_i, the following is true for $Var(f(G_i))$, the variance of $f(G_i)$.

$$\begin{aligned} & Var(f(G_i)) \\ & = E(f(G_i))(1 - E(f(G_i))) \\ & = p_{\prec}(f|D)(1 - p_{\prec}(f|D)) \\ & < \infty. \end{aligned}$$

Since $0 < p_{\prec}(f|D) < 1$, $Var(f(G_i))$ is also strictly greater than 0.

Again, we have already known that $E(f(G_i)) = p_{\prec}(f|D) < \infty$.

Thus, by the Central Limit Theorem (Theorem 5.5.15) (Casella and Berger, 2002),

$$\frac{\sqrt{N_o}(\hat{p}_{\prec}(f|D) - E(f(G)))}{\sqrt{Var(f(G))}} \longrightarrow^d \mathcal{N}(0,1),$$

that is,

$$\frac{\sqrt{N_o}(\hat{p}_{\prec}(f|D) - p_{\prec}(f|D))}{\sqrt{p_{\prec}(f|D)(1 - p_{\prec}(f|D))}} \longrightarrow^d \mathcal{N}(0,1).$$

Since $\hat{p}_{\prec}(f|D)$ converges in probability to $p_{\prec}(f|D)$, denoted by $\hat{p}_{\prec}(f|D) \longrightarrow^p p_{\prec}(f|D)$, by the Continuous Mapping Theorem (Theorem 5.5.4) (Casella and Berger, 2002),

$$\sqrt{\hat{p}_{\prec}(f|D)(1 - \hat{p}_{\prec}(f|D))} \longrightarrow^p \sqrt{p_{\prec}(f|D)(1 - p_{\prec}(f|D))}.$$

Finally, by Slutsky's Theorem (Theorem 5.5.17) (Casella and Berger, 2002),

$$\frac{\sqrt{N_o}(\hat{p}_{\prec}(f|D) - p_{\prec}(f|D))}{\sqrt{\hat{p}_{\prec}(f|D)(1 - \hat{p}_{\prec}(f|D))}} \longrightarrow^d \mathcal{N}(0,1).$$

(iv) Proof that for any $\epsilon > 0$, any $0 < \delta < 1$, if $N_o \geq \ln(2/\delta)/(2\epsilon^2)$, then $p(|\hat{p}_{\prec}(f|D) - p_{\prec}(f|D)| < \epsilon) \geq 1 - \delta$.

Since $f(G_1), f(G_2), \ldots, f(G_{N_o})$ are iid with Bernoulli($p_\prec(f|D)$), the Hoeffding bound (Hoeffding, 1963; Koller and Friedman, 2009) holds:

$$p(|\hat{p}_\prec(f|D) - p_\prec(f|D)| \geq \epsilon) \leq 2e^{-2N_o\epsilon^2}.$$

This is equivalent to

$$p(|\hat{p}_\prec(f|D) - p_\prec(f|D)| < \epsilon) \geq 1 - 2e^{-2N_o\epsilon^2},$$

and the conclusion is implied straightforward.

□

Proof of Equation (4.21)

Proof: For any $j \in \{1, \ldots, n\}$:

$$\begin{aligned}
&P((\sigma_j, U_{\sigma_j})|D) \\
&\propto P((\sigma_j, U_{\sigma_j}), D) \\
&= \sum_{\substack{(\sigma_1,\ldots,\sigma_{j-1}) \\ \in\mathcal{L}(U_{\sigma_j})}} \sum_{\substack{(\sigma_{j+1},\ldots,\sigma_n) \\ \in\mathcal{L}(V-U_{\sigma_j}-\{\sigma_j\})}} P(\sigma_1, \ldots, \sigma_{j-1}, \sigma_j, \sigma_{j+1}, \ldots, \sigma_n, D) \\
&= \sum_{\substack{(\sigma_1,\ldots,\sigma_{j-1}) \\ \in\mathcal{L}(U_{\sigma_j})}} \sum_{\substack{(\sigma_{j+1},\ldots,\sigma_n) \\ \in\mathcal{L}(V-U_{\sigma_j}-\{\sigma_j\})}} \prod_{i=1}^{n} \alpha'_{\sigma_i}(U_{\sigma_i}) \\
&= \alpha'_{\sigma_j}(U_{\sigma_j}) L'(U_{\sigma_j}) R'(V - U_{\sigma_j} - \{\sigma_j\}).
\end{aligned}$$

□

Proof of Equation (4.22)

Proof:

$$
\begin{aligned}
& p_{\prec}(G|D) \\
&= \sum_{\prec s.t. G \subseteq \prec} p(\prec, G|D) \\
&= \frac{1}{p_{\prec}(D)} \sum_{\prec s.t. G \subseteq \prec} p(\prec, G, D) \\
&= \frac{1}{p_{\prec}(D)} \sum_{\prec s.t. G \subseteq \prec} p(\prec, G) p(D|G) \\
&= \frac{1}{p_{\prec}(D)} \sum_{\prec s.t. G \subseteq \prec} (\prod_{i=1}^{n} q_i(U_i)\rho_i(Pa_i)) p(D|G) \\
&= \frac{1}{p_{\prec}(D)} \sum_{\prec s.t. G \subseteq \prec} (\prod_{i=1}^{n} p_i(Pa_i)) p(D|G) \\
&= \frac{1}{p_{\prec}(D)} \sum_{\prec s.t. G \subseteq \prec} p(G) p(D|G) \\
&= \frac{1}{p_{\prec}(D)} \cdot | \prec_G | \cdot p_{\not\prec}(G, D) \\
&= \frac{p_{\not\prec}(D)}{p_{\prec}(D)} \cdot | \prec_G | \cdot p_{\not\prec}(G|D).
\end{aligned}
$$

Since both $p_{\not\prec}(D) > 0$ and $p_{\prec}(D) > 0$, $p_{\prec}(G|D) \propto$ $| \prec_G | \cdot p_{\not\prec}(G|D)$, which also has been shown by Ellis and Wong (2008).

□

Proof of Theorem 4

Proof:

Let Ω denote the set of all the DAGs. Define $\mathcal{U}^+ = \{G \in \Omega : p_{\not\prec}(G, D) > 0\}$. $\mathcal{U}^+$ will equal Ω if the user chooses a prior such that $p(G) > 0$ for every DAG G (such as a uniform DAG prior $p(G) \equiv 1$). However, if the user has some additional domain knowledge so that he/she sets some prior to exclude some DAGs a priori, $\mathcal{U}^+$ will be a proper subset of Ω. Note that $p_{\not\prec}(f|D) = p_{\not\prec}(f, D)/p_{\not\prec}(D)$, where $p_{\not\prec}(f, D) = \sum_G f(G) p_{\not\prec}(G, D) = \sum_{G \in \mathcal{U}^+} f(G) p_{\not\prec}(G, D)$, and $p_{\not\prec}(D) = p_{\not\prec}(f \equiv 1, D) = \sum_G p_{\not\prec}(G, D) = \sum_{G \in \mathcal{U}^+} p_{\not\prec}(G, D)$.

Let $I[]$ denote the indicator function. Rewrite $\hat{p}_{\not\prec}(f|D)$ in Eq. (4.24) as $\hat{p}_{\not\prec}(f,D)/\hat{p}_{\not\prec}(D)$, where $\hat{p}_{\not\prec}(f,D) = \sum_{G\in\mathcal{G}} f(G)p_{\not\prec}(G,D) = \sum_G f(G)\, p_{\not\prec}(G,D)I[G\in\mathcal{G}] = \sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)I[G\in\mathcal{G}]$, and $\hat{p}_{\not\prec}(D) = \hat{p}_{\not\prec}(f\equiv 1,D) = \sum_{G\in\mathcal{G}} p_{\not\prec}(G,D) = \sum_G p_{\not\prec}(G,D)I[G\in\mathcal{G}] = \sum_{G\in\mathcal{U}^+} p_{\not\prec}(G,D)I[G\in\mathcal{G}]$.

Note that by Theorem 3, $P(G\in\mathcal{G}) = 1-(1-p_{\prec}(G|D))^{N_o}$, where $p_{\prec}(G|D)$ is the exact posterior of G under the order-modular prior assumption. Also note that by Eq. (4.22), $\forall G : (p_{\not\prec}(G,D)>0) \Rightarrow (p_{\prec}(G|D)>0)$.

(i) Proof that $\hat{p}_{\not\prec}(f|D)$ is an asymptotically unbiased estimator for $p_{\not\prec}(f|D)$, that is,

$$\lim_{N_o\to\infty} E(\hat{p}_{\not\prec}(f|D)) = p_{\not\prec}(f|D). \tag{4.28}$$

For notational convenience, let γ denote $\hat{p}_{\not\prec}(f,D)$, and let τ denote $\hat{p}_{\not\prec}(D)$. Define $g(\gamma,\tau)=\gamma/\tau$ so that $g(\gamma,\tau)$ is essentially $\hat{p}_{\not\prec}(f|D)$.

Note that

$$\begin{aligned}
&E(\gamma)\\
&= E(\hat{p}_{\not\prec}(f,D))\\
&= E\left(\sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)I[G\in\mathcal{G}]\right)\\
&= \sum_{G\in\mathcal{U}^+} E(f(G)p_{\not\prec}(G,D)I[G\in\mathcal{G}])\\
&= \sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)P(G\in\mathcal{G})\\
&= \sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)(1-(1-p_{\prec}(G|D))^{N_o}).
\end{aligned}$$

Thus,

$$\begin{aligned}
&\lim_{N_o\to\infty} E(\gamma) \\
&= \lim_{N_o\to\infty} \left(\sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D)(1-(1-p_{\prec}(G|D))^{N_o}) \right) \\
&= \sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D) \lim_{N_o\to\infty} (1-(1-p_{\prec}(G|D))^{N_o}) \\
&= \sum_{G\in\mathcal{U}^+} f(G)p_{\not\prec}(G,D) \\
&= p_{\not\prec}(f,D).
\end{aligned}$$

Similarly, by setting $f \equiv 1$, we have

$$\begin{aligned}
&E(\tau) \\
&= E(\hat{p}_{\not\prec}(D)) \\
&= \sum_{G\in\mathcal{U}^+} p_{\not\prec}(G,D)(1-(1-p_{\prec}(G|D))^{N_o}),
\end{aligned}$$

and

$$\lim_{N_o\to\infty} E(\tau) = p_{\not\prec}(D).$$

Next, by Taylor's Theorem (with the Lagrange form of the remainder),

$$\begin{aligned}
&g(\gamma,\tau) \\
&= g(E(\gamma),E(\tau)) \\
&\quad + \left[\frac{\partial g(\gamma,\tau)}{\partial\gamma}\right]_{\gamma=E(\gamma),\tau=E(\tau)} (\gamma^* - E(\gamma)) \\
&\quad + \left[\frac{\partial g(\gamma,\tau)}{\partial\tau}\right]_{\gamma=E(\gamma),\tau=E(\tau)} (\tau^* - E(\tau)),
\end{aligned}$$

where $\gamma^* = E(\gamma) + \theta(\gamma - E(\gamma))$, $\tau^* = E(\tau) + \theta(\tau - E(\tau))$, and θ is a random variable such that $0 < \theta < 1$.

Examine the components separately:

$$\begin{aligned}
&g(E(\gamma),E(\tau)) \\
&= E(\gamma)/E(\tau).
\end{aligned}$$

$$\left[\frac{\partial g(\gamma,\tau)}{\partial \gamma}\right]_{\gamma=E(\gamma),\tau=E(\tau)}$$
$$= \left[\tau^{-1}\right]_{\gamma=E(\gamma),\tau=E(\tau)}$$
$$= (E(\tau))^{-1}.$$

$$\left[\frac{\partial g(\gamma,\tau)}{\partial \tau}\right]_{\gamma=E(\gamma),\tau=E(\tau)}$$
$$= \left[-\gamma(\tau)^{-2}\right]_{\gamma=E(\gamma),\tau=E(\tau)}$$
$$= -E(\gamma)(E(\tau))^{-2}.$$

Since neither $E(\gamma)$ nor $E(\tau)$ is random,

$$\begin{aligned}
&E\left(g(\gamma,\tau)\right)\\
&= E(\gamma)/E(\tau)\\
&\quad + (E(\tau))^{-1}E(\gamma^* - E(\gamma))\\
&\quad - E(\gamma)(E(\tau))^{-2}E(\tau^* - E(\tau))\\
&= E(\gamma)/E(\tau)\\
&\quad + (E(\tau))^{-1}E(\theta(\gamma - E(\gamma)))\\
&\quad - E(\gamma)(E(\tau))^{-2}E(\theta(\tau - E(\tau))).
\end{aligned}$$

Consider the limit of each component separately:

$$\begin{aligned}
&\lim_{N_o\to\infty}(E(\gamma)/E(\tau))\\
&= \left(\lim_{N_o\to\infty}E(\gamma)\right) / \left(\lim_{N_o\to\infty}E(\tau)\right)\\
&= p_{\not\perp}(f,D)/p_{\not\perp}(D)\\
&= p_{\not\perp}(f|D).
\end{aligned}$$

$$\begin{aligned}
&\lim_{N_o\to\infty}(E(\tau))^{-1}\\
&=\left(\lim_{N_o\to\infty}E(\tau)\right)^{-1}\\
&=(p_{\not\propto}(D))^{-1}.
\end{aligned}$$

$$\begin{aligned}
&\lim_{N_o\to\infty}-E(\gamma)(E(\tau))^{-2}\\
&=-\left(\lim_{N_o\to\infty}E(\gamma)\right)\left(\lim_{N_o\to\infty}E(\tau)\right)^{-2}\\
&=-p_{\not\propto}(f,D)(p_{\not\propto}(D))^{-2}.
\end{aligned}$$

Note that both $(p_{\not\propto}(D))^{-1}$ and $-p_{\not\propto}(f,D)(p_{\not\propto}(D))^{-2}$ are constant real numbers.

Finally, we intend to prove the following two equalities:

$$\lim_{N_o\to\infty}E(\theta(\gamma-E(\gamma)))=0, \tag{4.29}$$

$$\lim_{N_o\to\infty}E(\theta(\tau-E(\tau)))=0. \tag{4.30}$$

Once this is done,

$$\begin{aligned}
&\lim_{N_o\to\infty}E\left(g(\gamma,\tau)\right)\\
&=\lim_{N_o\to\infty}(E(\gamma)/E(\tau))\\
&\quad+\lim_{N_o\to\infty}(E(\tau))^{-1}\cdot 0\\
&\quad+\lim_{N_o\to\infty}\left(-E(\gamma)(E(\tau))^{-2}\right)\cdot 0\\
&=p_{\not\propto}(f|D).
\end{aligned}$$

The whole proof for Eq. (4.28) will then be done.

The proof for Eq. (4.29) is as follows.

Based on the definition of limit, proving Eq. (4.29) is equivalent to proving

$$\lim_{N_o\to\infty}|E(\theta(\gamma-E(\gamma)))|=0. \tag{4.31}$$

Since

$$
\begin{aligned}
&|E(\theta(\gamma - E(\gamma)))| \\
&\leq E\left(|\theta(\gamma - E(\gamma))|\right) \\
&= E\left(|\theta| \cdot |\gamma - E(\gamma)|\right) \\
&\leq E\left(|\gamma - E(\gamma)|\right) \quad \text{(due to } 0 < \theta < 1\text{)},
\end{aligned}
$$

and $|E(\theta(\gamma - E(\gamma)))| \geq 0$, to prove Eq. (4.31), it is sufficient to prove

$$
\lim_{N_o \to \infty} E\left(|\gamma - E(\gamma)|\right) = 0. \tag{4.32}
$$

$$
\begin{aligned}
&E\left(|\gamma - E(\gamma)|\right) \\
&= E\left(\left|\sum_{G \in \mathcal{U}^+} f(G) p_{\not\prec}(G, D) I[G \in \mathcal{G}]\right.\right. \\
&\qquad \left.\left. - \sum_{G \in \mathcal{U}^+} f(G) p_{\not\prec}(G, D) P(G \in \mathcal{G})\right|\right) \\
&= E\left(\left|\sum_{G \in \mathcal{U}^+} f(G) p_{\not\prec}(G, D)(I[G \in \mathcal{G}] - P(G \in \mathcal{G}))\right|\right) \\
&\leq E\left(\sum_{G \in \mathcal{U}^+} f(G) p_{\not\prec}(G, D)\,|I[G \in \mathcal{G}] - P(G \in \mathcal{G})|\right) \\
&= \sum_{G \in \mathcal{U}^+} f(G) p_{\not\prec}(G, D) E\left(|I[G \in \mathcal{G}] - P(G \in \mathcal{G})|\right) \\
&= \sum_{G \in \mathcal{U}^+} f(G) p_{\not\prec}(G, D)[(1 - P(G \in \mathcal{G}))P(G \in \mathcal{G}) \\
&\qquad + (P(G \in \mathcal{G}) - 0)(1 - P(G \in \mathcal{G}))] \\
&= \sum_{G \in \mathcal{U}^+} f(G) p_{\not\prec}(G, D)[2P(G \in \mathcal{G})(1 - P(G \in \mathcal{G}))] \\
&= \sum_{G \in \mathcal{U}^+} f(G) p_{\not\prec}(G, D)[2(1 - (1 - p_{\prec}(G|D))^{N_o}) \\
&\qquad \times (1 - p_{\prec}(G|D))^{N_o}].
\end{aligned}
$$

Since $\forall G \in \mathcal{U}^+ : p_{\prec}(G|D) > 0$,

$$\begin{aligned}
&\lim_{N_o \to \infty} \sum_{G \in \mathcal{U}^+} f(G) p_{\not\prec}(G, D)[2(1 - (1 - p_{\prec}(G|D))^{N_o}) \\
&\qquad \times (1 - p_{\prec}(G|D))^{N_o}] \\
&= \sum_{G \in \mathcal{U}^+} f(G) p_{\not\prec}(G, D) \lim_{N_o \to \infty} [2(1 - (1 - p_{\prec}(G|D))^{N_o}) \\
&\qquad \times (1 - p_{\prec}(G|D))^{N_o}] \\
&= \sum_{G \in \mathcal{U}^+} f(G) p_{\not\prec}(G, D) \cdot 0 \\
&= 0,
\end{aligned}$$

Eq. (4.32) is proved, and the proof for Eq. (4.29) is done.

Setting $f \equiv 1$ in Eq. (4.29) leads to Eq. (4.30).

Thus, the whole proof for Eq. (4.28) is done.

(ii) Proof that $\hat{p}_{\not\prec}(f|D)$ converges almost surely to $p_{\not\prec}(f|D)$, denoted by $\hat{p}_{\not\prec}(f|D) \longrightarrow^{a.s.} p_{\not\prec}(f|D)$.

Remember that Ω denotes the set of all the DAGs, that is, $\Omega = \{G_1, G_2, \ldots, G_{W^*}\}$, where W^* is $|\Omega|$, the number of all the DAGs. Note that W^* is a finite positive integer though it is super-exponential in the number of variables n.

Let $\mathcal{F}$ be $\mathcal{P}(\Omega)$, the power set of Ω. Thus, $\mathcal{F}$ is a $\sigma-$algebra on Ω (Athreya and Lahiri, 2006). Define for any $A \in \mathcal{F}$, $\mu(A) = \sum_{G_j \in A} p_{\prec}(G_j|D)$. It is well-known that μ is a probability measure on $\mathcal{F}$ so that $(\Omega, \mathcal{F}, \mu)$ is a probability space (Athreya and Lahiri, 2006).

For each $i \geq 1$, let $\Omega_i = \Omega$, $\mathcal{F}_i = \mathcal{F}$ and $\mu_i = \mu$. Let $\Omega^\infty = \{(G^{(1)}, G^{(2)}, \ldots) : G^{(i)} \in \Omega_i, i \geq 1\}$. Let a cylinder set $\underset{\sim}{A} = A_1 \times A_2 \times \ldots \times A_k \times \Omega_{k+1} \times \Omega_{k+2} \times \ldots$, where there exists $1 \leq k < \infty$ such that $A_i \in \mathcal{F}_i$ for $1 \leq i \leq k$ and $A_i = \Omega_i$ for $i > k$. Let $\mathcal{F}^\infty = \sigma < \{\underset{\sim}{A} : \underset{\sim}{A} \text{ is a cylinder set } \} >$, that is, $\mathcal{F}^\infty$ is a $\sigma-$algebra generated by the set of all the $\underset{\sim}{A}$'s. $\mathcal{F}^\infty$ is a a $\sigma-$algebra on Ω^∞ and is called a product $\sigma-$algebra.

Define, for each $(A_1, A_2, \ldots, A_k, \Omega_{k+1}, \Omega_{k+2}, \ldots) \in \mathcal{F}^\infty$, $\mu^\infty(A_1, A_2, \ldots, A_k, \Omega_{k+1}, \Omega_{k+2}, \ldots)$ $= \mu_1(A_1) \times \mu_2(A_2) \times \ldots \times \mu_k(A_k)$. By Kolmogorov's consistency theorem, $(\Omega^\infty, \mathcal{F}^\infty, \mu^\infty)$ is a probability space (Athreya and Lahiri, 2006).

Let $\Omega^{\infty,0} = \{(G^{(1)}, G^{(2)}, \ldots) \in \Omega^{\infty} : \exists G \in \mathcal{U}^{+} : \forall i \geq 1 : G^{(i)} \neq G\}$. Let $\Omega^{\infty,1} = \Omega^{\infty} - \Omega^{\infty,0}$. Thus, $\Omega^{\infty,1} = \{(G^{(1)}, G^{(2)}, \ldots) \in \Omega^{\infty} : \forall G \in \mathcal{U}^{+} : \exists i \geq 1 : G^{(i)} = G\}$.

Define, for each $\omega^{\infty} \in \Omega^{\infty}$,

$$\hat{p}_{\not\perp}^{N_o}(\omega^{\infty}) = \frac{\sum_{G \in \mathcal{U}^{+}} f(G) p_{\not\perp}(G, D) I[G \in \mathcal{G}^{N_o}(\omega^{\infty})]}{\sum_{G \in \mathcal{U}^{+}} p_{\not\perp}(G, D) I[G \in \mathcal{G}^{N_o}(\omega^{\infty})]},$$

where $\mathcal{G}^{N_o}(\omega^{\infty})$ is the DAG set including the first N_o coordinates of ω^{∞}. By the definition, we know that $\hat{p}_{\not\perp}^{N_o}(\omega^{\infty}) = \hat{p}_{\not\perp}(f|D)$.

For each $\omega^{\infty} \in \Omega^{\infty,1}$, for each $G \in \mathcal{U}^{+}$, let $N(G, \omega^{\infty})$ be the smallest integer such that $G^{(N(G,\omega^{\infty}))} = G$. Let $N(\omega^{\infty}) = max_{G \in \mathcal{U}^{+}} N(G, \omega^{\infty})$. Then for each $N_o \geq N(\omega^{\infty})$, for each $G \in \mathcal{U}^{+}$, $I[G \in \mathcal{G}^{N_o}(\omega^{\infty})] = 1$.

Accordingly, for each $\omega^{\infty} \in \Omega^{\infty,1}$, there exists $N(\omega^{\infty})$ such that for each $N_o \geq N(\omega^{\infty})$:

$$\begin{aligned} &\hat{p}_{\not\perp}^{N_o}(\omega^{\infty}) \\ &= \frac{\sum_{G \in \mathcal{U}^{+}} f(G) p_{\not\perp}(G, D)}{\sum_{G \in \mathcal{U}^{+}} p_{\not\perp}(G, D)} \\ &= p_{\not\perp}(f|D). \end{aligned}$$

This implies that $\lim_{N_o \to \infty} \hat{p}_{\not\perp}^{N_o}(\omega^{\infty}) = p_{\not\perp}(f|D)$ for each $\omega^{\infty} \in \Omega^{\infty,1}$.

Finally, we intend to prove the following equality:

$$\mu^{\infty}(\Omega^{\infty,1}) = 1. \tag{4.33}$$

Once this is done, the whole proof for $\hat{p}_{\not\perp}(f|D) \longrightarrow^{a.s.} p_{\not\perp}(f|D)$ is done.

Proving Eq. (4.33) is equivalent to proving

$$\mu^{\infty}(\Omega^{\infty,0}) = 0. \tag{4.34}$$

For any $G \in \Omega$, let $\Omega^{\infty,0,G} = \{(G^{(1)}, G^{(2)}, \ldots) \in \Omega^{\infty} : \forall i \geq 1 : G^{(i)} \neq G\}$. Thus, $\Omega^{\infty,0} = \bigcup_{G \in \mathcal{U}^{+}} \Omega^{\infty,0,G}$. Accordingly, $\mu^{\infty}(\Omega^{\infty,0}) \leq \sum_{G \in \mathcal{U}^{+}} \mu^{\infty}(\Omega^{\infty,0,G})$.

For each $j \geq 1$, let $\Omega^{\infty,0,G,j} = \{(G^{(1)}, G^{(2)}, \ldots) \in \Omega^{\infty} : \forall i \in \{1, \ldots, j\} : G^{(i)} \neq G\}$. Note $\Omega^{\infty,0,G,1} \supseteq \Omega^{\infty,0,G,2} \supseteq \ldots \supseteq \Omega^{\infty,0,G,j-1} \supseteq \Omega^{\infty,0,G,j}$ for each $j \geq 1$. Thus, $\Omega^{\infty,0,G} = \bigcap_{j=1}^{\infty} \Omega^{\infty,0,G,j}$.

Finally, for any $G \in \mathcal{U}^+$,

$$\begin{aligned}
&\mu^\infty(\Omega^{\infty,0,G}) \\
&= \mu^\infty(\bigcap_{j=1}^{\infty} \Omega^{\infty,0,G,j}) \\
&= \lim_{j\to\infty} \mu^\infty(\Omega^{\infty,0,G,j}) \\
&= \lim_{j\to\infty} \mu_1(\Omega - \{G\}) \times \mu_2(\Omega - \{G\}) \times \ldots \times \mu_j(\Omega - \{G\}) \\
&= \lim_{j\to\infty} [1 - \mu(\{G\})]^j \\
&= \lim_{j\to\infty} [1 - p_\prec(G|D)]^j \\
&= 0.
\end{aligned}$$

Thus, $\sum_{G\in\mathcal{U}^+} \mu^\infty(\Omega^{\infty,0,G}) = 0$, so that Eq. (4.34) is proved. The whole proof is done.

Note that, by Theorem 2.5.1 (Athreya and Lahiri, 2006), the property that $\hat{p}_{\not\prec}(f|D)$ converges almost surely to $p_{\not\prec}(f|D)$ implies that $\hat{p}_{\not\prec}(f|D)$ converges in probability to $p_{\not\prec}(f|D)$, that is, $\hat{p}_{\not\prec}(f|D)$ is a consistent estimator for $p_{\not\prec}(f|D)$.

(iii) Proof that the convergence rate of $\hat{p}_{\not\prec}(f|D)$ is $o(a^{N_o})$ for any $0 < a < 1$.

In the proof for (ii), we have shown that for each $\omega^\infty \in \Omega^{\infty,1}$, there exists $N(\omega^\infty)$ such that for each $N_o \geq N(\omega^\infty)$, $\hat{p}_{\not\prec}^{N_o}(\omega^\infty) = p_{\not\prec}(f|D)$. This means that for any $0 < a < 1$, $(a^{N_o})^{-1}[\hat{p}_{\not\prec}^{N_o}(\omega^\infty) - p_{\not\prec}(f|D)] = 0$. Thus, $\lim_{N_o\to\infty}(a^{N_o})^{-1}[\hat{p}_{\not\prec}^{N_o}(\omega^\infty) - p_{\not\prec}(f|D)] = 0$ so that the proof is done.

(iv) Proof that if the quantity $\Delta = \sum_{G\in\mathcal{G}} p_{\not\prec}(G|D)$, then $\Delta\cdot\hat{p}_{\not\prec}(f|D) \leq p_{\not\prec}(f|D) \leq \Delta\cdot\hat{p}_{\not\prec}(f|D) + 1 - \Delta$.

On one hand,

$$\begin{aligned}
&\Delta \cdot \hat{p}_{\not\prec}(f|D) \\
&= \frac{\sum_{G\in\mathcal{G}} p_{\not\prec}(G,D)}{p_{\not\prec}(D)} \cdot \frac{\sum_{G\in\mathcal{G}} f(G) p_{\not\prec}(G,D)}{\sum_{G\in\mathcal{G}} p_{\not\prec}(G,D)} \\
&= \frac{\sum_{G\in\mathcal{G}} f(G) p_{\not\prec}(G,D)}{p_{\not\prec}(D)} \\
&= \frac{\sum_{G} f(G) p_{\not\prec}(G,D)}{p_{\not\prec}(D)} - \frac{\sum_{G\notin\mathcal{G}} f(G) p_{\not\prec}(G,D)}{p_{\not\prec}(D)} \\
&\leq \frac{\sum_{G} f(G) p_{\not\prec}(G,D)}{p_{\not\prec}(D)} \\
&= p_{\not\prec}(f|D).
\end{aligned}$$

On the other hand,

$$\begin{aligned}
&\Delta \cdot \hat{p}_{\not\prec}(f|D) + 1 - \Delta \\
&= \frac{\sum_{G\in\mathcal{G}} f(G) p_{\not\prec}(G,D)}{p_{\not\prec}(D)} + \frac{\sum_{G\notin\mathcal{G}} p_{\not\prec}(G,D)}{p_{\not\prec}(D)} \\
&\geq \frac{\sum_{G\in\mathcal{G}} f(G) p_{\not\prec}(G,D)}{p_{\not\prec}(D)} + \frac{\sum_{G\notin\mathcal{G}} f(G) p_{\not\prec}(G,D)}{p_{\not\prec}(D)} \\
&= \frac{\sum_{G} f(G) p_{\not\prec}(G,D)}{p_{\not\prec}(D)} \\
&= p_{\not\prec}(f|D).
\end{aligned}$$

Combining the two proved inequalities, the whole proof is done.

□

CHAPTER 5. SESSION ANALYSIS OF PEOPLE SEARCH WITHIN A PROFESSIONAL SOCIAL NETWORK

A paper published in *The Journal of the American Society for Information Science and Technology*

Ru He, Jiong Wang, Jin Tian, Cheng-Tao Chu, Bradley Mauney and Igor Perisic

Abstract

We perform session analysis for our domain of people search within a professional social network. We find that the content-based method is appropriate to serve as a basis for the session identification in our domain. However, there remain some problems reported in previous research which degrade the identification performance (such as accuracy) of the content-based method. Therefore, in this article, we propose two important refinements to address these problems. We describe the underlying rationale of our refinements and then empirically show that the content-based method equipped with our refinements is able to achieve an excellent identification performance in our domain (such as 99.820% accuracy and 99.707% F-measure in our experiments). Next, because the time-based method has extremely low computation costs, which makes it suitable for many real-world applications, we investigate the feasibility of the time-based method in our domain by evaluating its identification performance based on our refined content-based method. Our experiments demonstrate that the performance of the time-based method is potentially acceptable to many real applications in our domain. Finally, we analyze several features of the identified sessions in our domain and compare them with the corresponding ones in general web search. The results illustrate the profession-oriented characteristics of our domain.

5.1 Introduction

Session identification and analysis have been important research areas in web search in the last two decades (Silverstein et al., 1999; Lau and Horvitz, 1999; He et al., 2002; Ozmutlu et al., 2004; Jansen et al., 2007). Session is typically defined as a series of queries submitted by a single user during one episode of interactions between the user and the web search engine for one single information need. Once session identification can be correctly performed, the corresponding analysis is able to serve as the basis to analyze web users' search behaviors. Using session analysis, the search quality of current web search engines can be measured and then the corresponding improvement can possibly be made. Meanwhile, the performance of web search can often be improved if all the information in the same session of current query is used as the context of the search.

All the session analysis is based on the accurate and effective session identification. Currently there are two main session identification approaches that are computation-inexpensive and easily adopted in real application. The first one is the time-based method (Silverstein et al., 1999; Jansen et al., 2007). It is the most commonly used session identification method and has extremely low computation costs. The time-based method examines the time gap between two consecutive search queries (Q_i and Q_{i+1}) from a user and then compares it with a pre-specified time-out threshold (e.g. 10 minutes). Q_{i+1} is identified as the start of a new session if and only if the time gap exceeds the threshold. The main drawback of such a method is that users of search engine have various search behaviors so that it is possible that a user changes his search for a different information need within the specified time threshold or continues his search for the same information need after a long idle time. In these cases, the time-based method can not accurately identify sessions.

The second approach is the content-based method (He et al., 2002; Jansen et al., 2007), which identifies Q_{i+1} as the start of a new session if and only if there is no common term between two consecutive search queries Q_i and Q_{i+1} from a user. The content-based method is based on the users' search refinement patterns and several researchers (He et al., 2002; Jansen et al., 2007) argue that such a method is directly related to a user's information need and should be superior to its time-based counterpart. However, there are also problems degrading the identification performance (such as accuracy) of the content-based method, which have been reported in the previous researches (He et al., 2002; Jansen

et al., 2007). For example, in the comparison between the time-based method and the content-based method made by Jansen et al. (2007), they find that the accuracy rate of the content-based method (95.55%) is actually lower than that of the time-based method (98.95%). T hey explain the reasons for the problems, for example, main errors associated with the content-based method are caused by typos. However, they do not provide any solution to address these problems.

In recent years, as a sharply increasing number of people join professional social networks and use their search engines for professional development purposes, people search within a professional social network has become an emerging sub-domain worth special study in the web search area. Therefore, in this article, we will examine these two main session identification approaches in our specific domain and perform the corresponding session identification and analysis, which have not been substantially studied in the previous researches.

We first argue that because of the profession-oriented feature of our domain, the content-based method is able to serve as a basis for our session identification. At the same time, to address the problems of the content-based method, we provide two important refinements to improve identification performance. The first refinement is to filter out common stop words in our domain and the second refinement is to add the similarity check between terms. We describe the underlying reasons why our refinements effectively address the problems and then develop a new efficient similarity check algorithm suitable for our second refinement purpose. Our experimental results show that the content-based method with our two refinements is able to achieve an excellent identification performance (such as no worse than 99.820% accuracy) in our domain. We also show that the improvement from our two refinements is statistically significant and will potentially have an impact on practical applications in our domain.

Next, we investigate the feasibility of the time-based session identification method in our domain because the computation costs of the time-based method are extremely low and it will become a good candidate in the real applications as long as its identification performance is close to its content-based counterpart. Based on our refined content-based method, we evaluate the identification performance of the time-based method by varying its time-out thresholds. We empirically find the optimal threshold for the time-based method in our domain and show that its corresponding performance is potentially acceptable to many real applications.

Finally, we examine several features of the sessions in our domain which are identified by our refined content-based method. These features such as session length and session duration are analyzed and then compared with the corresponding ones in general web search. These results give more insights into the characteristics of our domain of people search inside a professional network, which have not been shown in the previous researches.

5.2 Related Work

As noted in Section 5.1, the time-based method is the most commonly used session identification method. One example using the time-based session identification method is the analysis performed by Silverstein et al. (1999), where they analyze search query log that includes about 1 billion entries from AltaVista search engine from August 2, 1998 to September 13, 1998. Their work includes session analysis as well as the analysis of individual queries, query duplicates and correlation of log entries. In their study, they use cookies to identify the users and then use the time-based method with 5-minute time-out threshold to define sessions within the search process of each single user. Based on their session identification method, they show that most of sessions are very short, for example, 77.6% of sessions include only one query.

Lau and Horvitz (1999) propose the methodology of query refinement classes when they manually tag 4,690 search queries submitted to Excite search engine on September 16, 1997. They define a set of exhaustive and mutually exclusive query refinement classes including new, generalization, specialization, reformulation etc. so that each query can be manually tagged into exactly one refinement class. Especially, each search query in the query refinement class of "new" corresponds to a query seeking for a new information need, that is, the start of a new session. They also mention the feasibility of automatic assignment of refinement class but they do not develop a concrete algorithm. Based on their manual assignment of query refinement classes and information goals, they perform the corresponding statistical analysis about users' search activities and manually build Bayesian network models to describe users' search processes.

He et al. (2002) propose a new approach to identify session boundaries inside search queries of each user based on Reuters log data including 9,534 search records from March 30, 1999 to April 7,

1999. In their paper, the content-based session identification method is developed by automatically using the methodology of query refinement classes. The main idea of the content-based method is simple: If there is no common term between two consecutive search queries Q_i and Q_{i+1} from a user, then the search pattern is new, that is, Q_{i+1} is the start of a new session. Otherwise, these two queries are in the same session and the more specific search pattern such as generalization, specialization, reformulation, or repetition can further be determined based on the relation between the terms in Q_i and the terms Q_{i+1}. The content-based approach is computation-inexpensive and their experimental results show that the content-based approach is significantly better than the time-based approach in the performance of session identification. However, they find that the content-based approach is more accurate in determining generalization, specialization, and repetition but less accurate in determining new (i.e. session boundary) and reformulation. Since session boundary identification is typically of primary importance, on the basis of Dempster-Shafer theory, He et al. further combine the time-based approach and the content-based approach using statistical evidences of both time intervals and search patterns. Their experiments show that the new combined approach is a small improvement on the pure content-based approach: for instance, recall increases from 96.63% into 98.15% and precision only decreases from 59.85% to 59.55%.

Several computation-expensive techniques have also been proposed by researchers. Ozmutlu et al. (2004) propose using neural networks to perform session identification. A sample of about 10,000 queries from Excite data log are first manually examined so that binary tags about session change or session continuation are marked. At the same time each query is categorized in terms of its time interval and search pattern. The first half of the data are used to train a neural network and the second half of the data are used to test the identification performance of the trained neural network. Their experiments show that 76% of session changes and 92% of session continuations are identified correctly. By extending this work, Ozmutlu et al. (2008) then report that it is possible for neural networks to perform effective session identification even if the training data and test data are from different search engine data logs.

Sriram et al. (2004) suggest a statistical method to perform session identification. First, they suggest measuring the difference between two terms based on Jensen-Shannon (JS) Divergence, which compares the probability distribution of two terms over a given set of web pages. Then the difference

scores between two queries Q_i and Q_j can be calculated by summing JS Divergence of each term in these two queries, i.e., $\sum_i \sum_j JS(t_i, t_j)$ where t_i and t_j are terms from Q_i and Q_j respectively. Finally Q_j will be identified as the start of a new session if and only if the difference scores exceed some specified threshold.

Jansen et al. (2007) examine three major methods of identifying web search sessions based on 2,465,145 interactions from 534,507 users of Dogpile.com on May 6, 2005. These three examined session identification methods are: (1) IP and cookie; (2) IP, cookie, and 30-minute time-out threshold; and (3) IP, cookie, and content change (based on query refinement patterns). Note that Method (1) essentially groups all the daily queries from each user into one session; Method (2) is the time-based method; and Method (3) is the content-based method. All these three session identification methods are computation-inexpensive and can be easily adopted in the real application. They show that the application of each method will lead to the similar conclusion that sessions are usually short: for example, the mean session length is less than three queries and the mean session duration is less than 30 minutes. Meanwhile, they argue that Method (3) is the best session identification method because it can address the contextual aspects which the other two methods can not address. Unfortunately, their experiments, based on their 2,000 manually identified queries, show that the accuracy rate of Method (2) is 98.95% whereas the accuracy rate of Method (3) is 95.55%. They point out the reasons that the accuracy rate of Method (3) is lower than the one of Method (2): most of identification errors of Method (3) are from typos misspelled by users. (See our detailed discussion in Section 5.3.3.) However, they do not provide any method to reduce these errors to improve the identification accuracy of Method (3).

In addition, session identification problem also occurs in the domain of web usage mining, which uses data mining techniques to discover users' web navigation patterns from web log data. There are also two main session identification approaches in web usage mining. One is the time-oriented approach (Cooley et al., 1999; Spiliopoulou and Faulstich, 1999; Frias-martinez and Karamcheti, 2003), which uses either the entire duration threshold or the time-out gap threshold to identify the session boundaries inside a group of web pages. The other is the navigation-oriented approach (Cooley et al., 1999, 2000), which uses the hyper-link relations among web pages to group related web pages into the same session. Huang et al. (2004) propose a new session definition method using statistical language modeling. By assuming a set of web pages (or objects) for the same information need are frequently visited one after

another, they claim that the corresponding sequence of web pages (or objects) has a high estimated probability and low empirical entropy. Therefore, when a new web page (or object) is observed in the sequence that is not related to the original information need, the resulting sequence will have a detectable increase in its empirical entropy, serving as a natural signal for adding a session boundary. Recently, Bayir et al. (2009) combine the time-oriented approach and the navigation-oriented approach and then define a session as a set of paths (in the web graph) where each path corresponds to a user's navigation among web pages.

In the last several years, because of the dramatically increasing number of people around the world who get engaged in professional social network sites and use their search engines, people search within a professional social network has become an emerging sub-domain in web search. Within a professional social network, each member provides his professional information which is stored as his profile document in the professional social networking site. When a member performs people search in the corresponding social networking site, based on the content of his search query as well as the searcher's own personal information, the search engine will return the searcher the professional information of the most relevant people.

A number of studies (Travers and Milgram, 1969; Kleinberg, 2000; Haynes and Perisic, 2009) related to people search within a social network have already been made, from several aspects other than session analysis. Travers and Milgram (1969) empirically show the small-world phenomenon which is often called "six degrees of separation", that is, two arbitrary persons can be connected in about six steps on average in a worldwide social network. Kleinberg (2000) demonstrates that social networks include the information fundamental for finding a short path between an arbitrary pair of persons in social networks. Recently, Haynes and Perisic (2009) empirically show that the social network structure has great value in improving the search result relevance, even after adjusting other factors including result rank, geodesic distance (shortest path length) from the searcher, and the search type, based on LinkedIn data including 3,835,364 search queries in a week of March 2008 as well as the information about LinkedIn members.

In contrast with all the previous researches introduced above, in this article, we contribute to the existing literature by performing session identification and analysis particularly for the domain of people search within a professional social network. First, we show that equipped with two important and

efficient refinements which we propose, the content-based session identification method is able to effectively reduce identification errors, especially those typo-related errors reported by Jansen et al. (2007), and achieve an excellent identification performance in our domain. Next, based on our refined content-based method, we investigate the applicability of the time-based method in our domain by evaluating its identification performance with various time-out thresholds. Finally, we analyze several features of identified sessions to illustrate further the profession-oriented nature of our domain, which has not been well studied in previous research.

5.3 Our Content-based Session Identification Approach

We use LinkedIn daily people search log files for the week from June 14, 2010 (Monday) to June 20, 2010 (Sunday) as the data source for our session analysis on people search within a professional social network. LinkedIn is a popular professional social networking site and provides people search service inside its social network which includes more than 70 million members (registered users). Each LinkedIn daily people search log file records all the people search queries submitted from its members on the given day. Because every member within LinkedIn social network has been assigned a unique member id in the system, the search records from each user can be easily grouped based on his member id, which is more accurate than the user identification methods based on IP address and/or cookies used by many previous researchers for general web search.

In this article we will use the following format to present our analysis. We first describe in details our analysis for the data for June 14, 2010. The extended analysis over the whole week from June 14, 2010 to June 20, 2010 will then be demonstrated either in the main part of the paper or in the Appendix, depending on its importance and relevance. The data for June 14, 2010 we have used include 5,552,891 search queries from 806,799 users, where these 806,799 users are randomly sampled from all the users who submitted at least one query on June 14, 2010. The data of each of other six days include the queries from 270,000 randomly sampled users.

5.3.1 Preliminary Analysis Based on User Identification

We first perform some preliminary analysis based on the method that groups daily people search queries purely according to user identification. Such a method corresponds to session identification method (1) performed by Jansen et al. (2007), which is even directly used as a session identification method in many Web-searching studies (Spink and Jansen, 2004). Our preliminary analysis will serve as the basis for the further session analysis in this article.

To ease the analysis, we define a random variable X to be the number of people search queries submitted by a single user on the investigated day. The (empirical) distribution of X based on the data for June 14, 2010 is given in Table 5.1 and its corresponding histogram is shown in Fig. 5.1.

The distribution of X indicates that a LinkedIn user typically submits a small number of people search queries per day. For example, 33.085% of users submitted only one query and 60.273% of users submitted no greater than three queries during the whole day of June 14, 2010. At the same time, the distribution of X has a very heavy right tail. For example, there were still 1.222% of users submitting 31 - 40 queries on June 14, 2010. Note that 0.013% of users submitted more than 500 queries on that day and the maximum value of X even reached 2,699. However, based on our manual examination as well as the commonsense reasoning, we believe that they were most likely to be the queries submitted by software agents instead of human beings.

In addition, we make a brief comparison between X in our domain and the corresponding one obtained from method (1) of Jansen et al. (2007) as follows. The reported adjusted mean of X from method (1) of Jansen et al. (2007) is 2.85, which is based on the daily data excluding all the records from the users who submitted more than 99 search queries a day. In our domain, the corresponding adjusted mean of X is 5.94, which does not decrease much from 6.88, the original mean of X. This comparison shows that a user in our domain submits relatively more search queries per day on average.

Because any time-based or content-based method needs to further identify sessions inside all the search queries from each user, the analysis of X will have an important impact on the further session analysis. For example, the distribution of X will affect the corresponding distribution of session length. We can safely conclude that at least 33.085% of session length will be 1 on June 14, 2010 because one single daily query from one single user will trivially become one session with session length 1 by any

Table 5.1: Distribution of X on June 14, 2010

X	Count	Percentage	X	Count	Percentage
1	266,927	33.085%	11 - 20	65,407	8.107%
2	135,241	16.763%	21 - 30	21,670	2.686%
3	84,107	10.425%	31 - 40	9,857	1.222%
4	57,988	7.187%	41 - 50	5,728	0.710%
5	42,598	5.280%	51 - 100	9,916	1.229%
6	31,378	3.889%	101 - 500	4,134	0.512%
7 - 10	71,740	8.892%	501 - 2699	108	0.013%

Table 5.2: Descriptive Statistics of X over a Week

Date	Min	1st Qu.	Median	Mean	3rd Qu.	Max	SD
06/14/2010	1	1	3	6.88	6	2,699	18.63
06/15/2010	1	1	3	7.03	6	4,430	19.19
06/16/2010	1	1	3	6.85	6	2,082	17.67
06/17/2010	1	1	3	7.19	6	3,431	19.47
06/18/2010	1	1	3	7.63	7	1,679	18.71
06/19/2010	1	1	3	6.64	6	1,553	18.40
06/20/2010	1	1	3	6.53	6	3,102	18.75

time-based or content-based method.

Meanwhile, the question whether the distribution of X remains the same across multiple days is often an interesting topic. Therefore, we also list some descriptive statistics of X over the week from June 14, 2010 to June 20, 2010 in Table 5.2. The listed descriptive statistics of X include minimum, first quartile, median, mean, third quartile, maximum and standard deviation. With the Kolmogorov-Smirnov (KS) two-sample test (Conover, 1999), we can conclude that the distribution of X on each day from June 15, 2010 to June 20, 2010 is the same as the distribution of X on June 14, 2010. Please refer to Appendix C for the details.

5.3.2 Consideration of Content-based Session Identification Method

Remember that session definition emphasizes one single information need from one user. Because the identification of one user is relatively easy in our domain, the identification of one information need

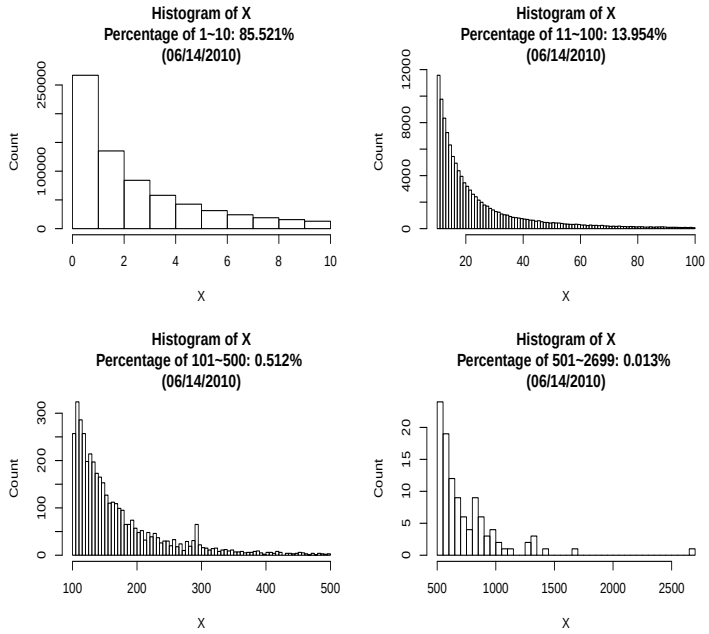


Figure 5.1: Histogram of X on June 14, 2010

becomes the critical part in our session definition. To get a general idea about how to identify one information need of each user from the recorded data, we first manually go through a small random sample from the search log files over the week from June 14, 2010 to June 20, 2010. We find that because of the professional-social-network nature of LinkedIn, information needs in LinkedIn people search are always profession-oriented and can neatly fall into one of the following four categories: (1) search for person(s) based on one particular name; (2) search for persons who are qualified for one particular job title; (3) search for persons in one particular company; (4) search for persons in one particular school.

Meanwhile, in LinkedIn people search log file each record corresponds to one submitted people search query. There are the following eight fields in each record closely related to our session analysis: (1) member id; (2) tracking time; (3) keywords; (4) query first name; (5) query last name; (6) query title; (7) query company; (8) query school.

Field (1) records the unique id of the user who submits the query and field (2) records the query submission time (UTC Time). In this paper, we assume that all the queries on the investigated day are

ordered by the pair of these two fields, (member id, tracking time), so that on the given day all the queries from the same user are grouped together and sorted according to the ascending order of their submission time. This assumption, which can be easily achieved by the modern database techniques, will simplify the demonstration in the paper.

Each field from (3) to (8) records the words the user enters in the text-box denoted with the corresponding name on the web page. Note that field (3) appears in both basic search and advanced search while each field from (4) to (8) appears only in advanced search. LinkedIn users can choose to use either basic search or advanced search according to their wishes. If a user uses basic search, the contents in field (3) typically include the information about one or more fields from (4) to (8).

It can be clearly seen that the fields from (3) to (8) directly correspond to the four categories of information needs we have pointed out. Accordingly, we find that the content-based method is appropriate to serve as the basis for the session identification in our domain. In other words, because the terms in a query in our domain are directly related to the name of one particular person, one particular job title, the name of one particular company or the name of one particular school, we find that "in most cases" the fact that there is no common term between two consecutive queries Q_i and Q_{i+1} of a user implies that a new information need (i.e., a new session) starts at Q_{i+1}.

Meanwhile, we also observe that "in most cases" does not mean "in all the cases." There are some cases in which there is some common (case-insensitive) term between two consecutive queries Q_i and Q_{i+1} of a user but Q_{i+1} should be identified as the start of a new session. There are also some cases that even if there is no common term between Q_i and Q_{i+1}, Q_i and Q_{i+1} still should be grouped into the same session. These mentioned cases are the main difficulties which degrade the identification performance of the original content-based method. To address these cases, in the next subsection, we will propose two important refinements to improve the original content-based method. These two refinements are general enough to improve any content-based session identification method for web search. Furthermore, because of the characteristics of our domain, both refinements can be realized with very low computation costs to help the content-based method to achieve an excellent performance.

5.3.3 Two Important Refinements to Content-based Method

5.3.3.1 Refinement 1: Filter Out Stop Words.

The first refinement is to filter out some trivial words (terms) from two consecutive search queries Q_i and Q_{i+1} before starting looking for any common words between them. The purpose of this refinement is to avoid missing some true session boundaries. There are some words, such as articles and prepositions, whose concurrence in two consecutive queries does not necessarily imply the continuation of the same session. In this paper, we call these words as "stop words," a common term used in the search industry.

Accordingly, we create a short list of the most common stop words whose occurrence should be omitted when determining a session boundary in our domain. On one hand, the list includes some common stop words in web search: "the," "a," "an," "and," "or," and "of." On the other hand, it also includes the following stop words that are specific to our domain. In regard to the information need of school query, the terms "school," "university," and "college" themselves convey no concrete information about the name of the searched school so that we add them into our stop-word list. For example, if Q_i is "University of Maryland" and Q_{i+1} is "University of Washington," then Q_{i+1} obviously should be the start of a new session even though there are two common words ("university" and "of") between Q_i and Q_{i+1}. Similarly, in terms of the information need of company query, the terms "company," "LLC" (Limited Liability Company), and "inc." (incorporation) are in the stop-word list. Note that the length of the resulting stop-word list is just a small constant. However, our experiments in Section 5.3.5 will show that our short stop-word list can non-negligibly improve the identification performance of the content-based method.

5.3.3.2 Refinement 2: Check Similar Terms.

The second refinement is to perform a similarity check between the terms in Q_i and the terms in Q_{i+1}, which plays a critical role in improving the identification performance (such as accuracy) of the content-based method. This refinement is based on our observation that in two consecutive queries Q_i and Q_{i+1} of a user, some term T in Q_i can possibly be modified mildly into the corresponding term T' in Q_{i+1} but T and T' really mean the same thing from the user's perspective. This is usually because of

the following two common situations. The first situation is that T is the term including some typo that a user accidently inputs in Q_i and T' is the term that the user enters in Q_{i+1} to correct the typo. The second situation is that a user does not know the exact spelling of the query term so that he submits his guessed T in Q_i. When he does not get the desired search result, he modifies T into newly guessed T' in Q_{i+1} to see whether the better search result can be returned to him. The second situation is particularly common in our domain of people search within a professional social network. For example, a recruiter user may only know the pronunciation of the name of his interested person so that he has to try several times to guess the spelling of the name.

In both situations, Q_i and Q_{i+1} are really for the same information need while the string T in Q_i and the string T' in Q_{i+1} are only similar but not exactly the same in their surface forms. If we only use the exact (case-insensitive) string-match method, it will lead to the identification error that a session boundary is incorrectly inserted between two consecutive queries when every term T in Q_i is modified mildly into the corresponding term T' in Q_{i+1}. This kind of errors have been reported by Jansen et al. as the main error source degrading the performance of the content-based method, though they do not provide a corresponding solution (Jansen et al., 2007). Therefore, in this paper we will address this kind of errors by adding the similarity check for two terms between Q_i and Q_{i+1}.

We notice that many conventional similarity check algorithms are expensive in time and space costs. One type of algorithms compute the Levenshtein distance (edit distance) (Levenshtein, 1966) between two terms for the similarity check. The Levenshtein distance between two strings is defined as the minimum number of edits needed to transform one string into the other, where three allowed operations are insertion, deletion, and substitution of a single symbol. Unfortunately, this type of algorithms use dynamic programming technique and have time complexity $O(m \cdot n)$ and space complexity $O(m)$ (Wagner and Fischer, 1974), where m and n are the length of two strings. The high time and space complexity make this type of algorithms undesirable for the similarity check for a large number of people search queries submitted to the website every minute.

Another type of algorithms are N-gram algorithms (Manning and Schuetze, 1999), where N is a pre-specified positive integer and a N-gram is a N-length subsequence from a term. By converting each term into a set of all the possible N-grams of this term, N-gram algorithms examine the similarity of two terms on the basis of the similarity measure of two sets of N-grams. Different similarity measures

of two sets of N-grams can be used according to different scenarios. Since N-gram algorithms have much cheaper time complexity $O(m+n)$ and space complexity $O(m)$, we will consider them as the candidates for our similarity check. Please refer to Appendix A for the details of the N-gram algorithms that we use for our domain.

Meanwhile, based on our examination of the LinkedIn search log files, we also propose a new efficient similarity check algorithm suitable for our domain, which has both $O(1)$ time complexity and $O(1)$ space complexity. Our algorithm is based on the following important observation: when a user in our domain modifies a term T into T' for the same information need, he usually modifies a very small portion of T and leaves the rest unchanged. Therefore, to check the similarity of two terms in our domain, it is usually enough to perform the constant-length string-match check from the beginning (head) and/or from the end (tail) of these two terms. This constitutes the essential idea of our algorithm "isSimilarByHeadOrTail()", whose details are shown in Algorithm 1.

Remark 3 *Explanation of Algorithm 1.*

In Algorithm 1, the input parameters term1 and term2 are the two terms for which we intend to perform the similarity check, and the input parameter thresholdLen is the constant-length threshold (upper bound) with which we will perform the string-match check. The output result of Algorithm 1 is a Boolean value which indicates whether term1 and term2 are similar or not. In Line 1 and 2, term1 and term2 are transformed into str1 and str2 which include only lower-case characters since the similarity check is case-insensitive. From Line 3 to Line 6, four variables involving the lengths are initialized in a straightforward fashion.

From Line 7 to Line 20, the string-match check from the left end (beginning) of two terms is performed. This check addresses the situation that a user's modification(s) is/are near the end of T so that the part from the beginning of T remains unchanged. leftLen is the variable recording the number of same symbols from the left end of two terms. Once leftLen reaches the threshold, our algorithm will immediately return true in Line 19, that is, the two terms are similar. However, there are also situations that both terms themselves are shorter than the threshold so that leftLen is impossible to reach the threshold. In these situations, if leftLen is large enough with respect to the lengths of two terms, our algorithm will still return true to claim the similarity. These situations are checked in these "or"

conditions in Line 18. From Line 21 to Line 34, the string-match check from the right end of two terms is performed, which is exactly symmetric to the check from Line 7 to Line 20. Finally, the string-match check based on both sides is performed from Line 35 to Line 38. This addresses the situation that a user's modification(s) is/are near the middle of T.

5.3.4 Evaluation Strategy for Session Identification Methods

To investigate the performance of the content-based method and the effects of two refinements we propose, in this subsection we describe our evaluation strategy which will be used to measure the performance of any session identification method including the content-based method or the time-based method. Our evaluation strategy is consistent with a commonly used evaluation strategy described by Yang (1999).

We will use a binary string to represent the session identification for a series of queries submitted on the investigated day. We let one bit represent one query. Bit 0 indicates that the current query is the start of a new session whereas bit 1 indicates that the current query belongs to the same session that the previous query is in. In this way the number of 0's in the binary string will equal the number of identified sessions.

For a series of queries submitted on a given day, we call the binary string representing the correct session identification the gold-standard binary string. At the same time, a binary string representing the session identification returned from a session identification method is the corresponding reported binary string. We use the Hamming distance between the gold-standard binary string and the reported binary string, that is, the number of different bits in these two strings, to represent the number of errors made by the corresponding session identification method. We define the accuracy rate (also just called accuracy) of the session identification method as 1 minus the quotient of the number of errors and the length of the binary string.

We will also use some common performance measurements of classifiers for our evaluation purpose. We regard bit 0 as positive symbol and bit 1 as negative symbol. Therefore, the number of true positive (TP) is the number of 0's that are both in the reported binary string and in the gold-standard binary string; the number of false positive (FP) is the number of 0's that are in the reported binary string but

not in the gold-standard binary string; the number of true negative (TN) is the number of 1's that are both in the reported binary string and in the gold-standard binary string; and the number of false negative (FN) is the number of 1's that are in the reported binary string but not in the gold-standard binary string. Then true positive rate (also called recall) (TPR = TP / (TP + FN)), false positive rate (FPR = FP / (TN + FP)), precision (TP / (TP + FP)) and F-measure (2 $\times$ TPR $\times$ precision / (TPR + precision)) are all defined in the usual way. Also note that the previously defined accuracy rate can be expressed as $1 -$ (FP + FN) / (TP + TN + FP + FN) = (TP + TN) / (TP + TN + FP + FN).

5.3.5 Evaluation of Our Content-based Method

To evaluate the content-based session identification method and the effects of our two refinements, we use the following sampling strategy to get the subset of the data set on June 14, 2010. We define 50 buckets such that each bucket i ($i \in \{1, 2, \cdots, 49, 50\}$) is the group of users who submitted i people search queries during the day. Then we randomly sample 10 users within each bucket so that we have in total 500 users. Finally, all the search records of these 500 users are retrieved so that totally $12,750$ $(10 \times \sum_{i=1}^{50} i)$ records (queries) are used as the sample for our evaluation purpose. (Again, we assume that all these 12,750 records are ordered by the pair (member id, tracking time).)

The reason that we use the above sampling strategy (random sample of users inside each bucket) instead of complete random sample of users directly from the daily search log file is that the latter strategy will lead to a sample where the number of search records from a user per day is usually small. As we have shown in Table 5.1 and 5.2, 33.085% of users submitted only one query on June 14, 2010 and the mean of X (i.e., the mean number of search queries from a user on that day) is 6.88. If the mean of X is small, the performance of a session identification method will be measured trivially high. For example, all the records including only one query from a user per day will be correctly but trivially identified as one session by any time-based or content-based method. In general, the first query of every user will always be correctly but trivially identified as the start of a new session. As a consequence, the percentage of correctly but trivially identified records equals the reciprocal of the mean of X. Therefore, we intentionally use the new sampling strategy described above to increase the mean of X into 25.5 $((1+50)/2)$ and decrease the percentage of correctly but trivially identified records into 3.92% $(500/12750 = 1/25.5)$ in the resulting sample.

Given the sample of $12,750$ records (queries), we first perform the manual session identification to get the corresponding gold-standard binary string of length $12,750$. (Please refer to Remark 4 at the end of Section 5.3.5 for our manual session identification process.) Based on the gold-standard binary string, the performance of various content-based session identification methods is then recorded in Table 5.3.

The first row of the table gives the performance of the original content-based session identification method. Correspondingly, the details of the original content-based method we have used are described in Algorithm 2. Given two consecutive queries Q_1 and Q_2, Algorithm 2 will output the corresponding Boolean value to indicate whether Q_1 and Q_2 are in the same session. The pseudo code from Line 4 to Line 11 in Algorithm 2 is the main part of the original content-based method, describing how to decide whether two nonempty consecutive queries from the same user are in the same session or not. (The pseudo code from Line 12 to Line 15 describes our solution for the infrequent cases that at least one query is empty based on LinkedIn domain context. Please refer to Remark 5 at the end of Section 5.3.5 for the details.) From the first row of Table 5.3, we can see both accuracy and F-measure of the original content-based method are good (98.565% and 97.706%), which indicates that it can be used as a good starting point for session identification in our domain.

Extended from Algorithm 2, Algorithm 3 presents the details of the content-based method with our two refinements. The pseudo code from Line 7 to Line 8 is for Refinement 1 and the pseudo code from Line 11 to Line 12 is for Refinement 2. The second row in Table 5.3 gives the performance of the content-based method with only Refinement 1. It shows that Refinement 1 can effectively reduce FN so that the performance has a non-negligible increase.

Starting from the third row of Table 5.3, we record the performance of our content-based method with both Refinement 1 and Refinement 2. We try different algorithms and/or parameters for the similarity-check function isSimilar() (in Line 11 of Algorithm 3) to see their effects on the performance.

In the cases that we use purely N-gram algorithm as the similarity-check function, the setting of N = 4, that is, 4-gram, leads to the best performance result. In such a setting, the number of errors decreases to 40; and the accuracy rate and F-measure increase to 99.686% and 99.493% respectively. (Please refer to Appendix A for the details of N-gram algorithms we use.) In the cases that we use purely HeadOrTail algorithm, that is, isSimilarByHeadOrTail() shown in Algorithm 1, as the similarity-check function, the

setting of thresholdLen = 4 gives the best performance result (with accuracy 99.820% and F-measure 99.707%).

Finally, we try the similarity-check algorithm that combines N-gram algorithm and HeadOrTail algorithm, with its performance recorded in the last row of Table 5.3. Our combined similarity-check algorithm first uses HeadOrTail algorithm with thresholdLen = 4. If HeadOrTail algorithm returns true, then the similarity check is done. Otherwise, our combined algorithm uses 4-gram algorithm to do the second-round similarity check. This second-round check is designed to safeguard against the case that a user simultaneously updates the term T both near its beginning and near its end so that HeadOrTail algorithm fails to report the similarity. Table 5.3 shows that our combined similarity-check algorithm leads to the same best performance (accuracy 99.820% and F-measure 99.707%) as the purely HeadOrTail algorithm with thresholdLen = 4.

For the remaining parts of the paper, based on its reported optimal performance, our refined content-based identification method will be the content-based method that includes two refinements and utilizes our combined similarity-check algorithm (with thresholdLen = 4 and N = 4) for Refinement 2. Remember that our sampling strategy intentionally increases the mean of X and reduces the number of trivially correct session identification. Therefore, we can expect that in the real situation, with more trivially correct session identification, our refined content-based method will have even better performance than reported here. As a result, our refined content-based method really can achieve an excellent identification performance in our domain.

Note that the improvement from our refined content-based method is significant in terms of theory. Actually, we can conclude (with an extremely small p-value) that the accuracy of our refined content-based method is statistically significantly better than the accuracy of the original content-based method based on either the sign test (Conover, 1999) or the McNemar test (Conover, 1999). Such a conclusion is mainly based on the fact that the total number of our investigated queries is large $(12,750)$, though the conclusion may seem surprising at first glance because the reported accuracy of our refined method (99.820%) is close to the corresponding one of the original method (98.565%). Please refer to Appendix E for the details of the statistical tests.

Also note that the improvement from our refined content-based method will potentially have an impact on the applications in our domain. In the future all the context information within the same

Table 5.3: Performance of Various Content-based Session Identification Methods (Based on the Sample of 12,750 Queries)

Method	TP	FP	TN	FN	No. of Errors	Accuracy	F-measure
Original	3,898	150	8,669	33	183	98.565%	97.706%
Filter-Out	3,930	151	8,668	1	152	98.808%	98.103%
Filter-Out & 3-gram	3,881	23	8,796	50	73	99.427%	99.068%
Filter-Out & 4-gram	3,924	33	8,786	7	40	99.686%	99.493%
Filter-Out & 5-gram	3,926	46	8,773	5	51	99.600%	99.355%
Filter-Out & HeadOrTail (thresholdLen = 3)	3,831	8	8,811	100	108	99.153%	98.610%
Filter-Out & HeadOrTail (thresholdLen = 4)	3,920	12	8,807	11	23	99.820%	99.707%
Filter-Out & HeadOrTail (thresholdLen = 5)	3,925	24	8,795	6	30	99.765%	99.619%
Filter-Out & HeadOrTail + 4-gram (thresholdLen = 4)	3,918	10	8,809	13	23	99.820%	99.707%

session is intended to be used by our LinkedIn search engine to improve the performance of people search within the professional social network so that the users can find the appropriate professionals more easily and quickly. Although all the information from a correctly identified session can help our search engine to improve its search performance, the information from an incorrectly identified session can also mislead our search engine and degrade its performance. Because we intend to minimize occurrences of the latter situation, especially for those important users who have bought their business accounts and perform extensive people searches in our domain, it is desirable for our search engine to have a computation-inexpensive session identification method that has the best possible identification performance. In fact, because of a huge number of users and their submitted daily search queries in our professional social network, even a one percent increase in accuracy could have non-negligible effects. Take the investigated data of June 14, 2010, which include $806,799$ users and $5,552,891$ search queries for example, by excluding the 500 queries that have been correctly but trivially identified from the sample of $12,750$ manually identified records, the nontrivial accuracy of the original contented-based method on the sample of $12,750$ records is actually 98.51% $((3,898+8,669-500)/(12,750-500))$ and the corresponding nontrivial accuracy of our refined contented-based method is actually 99.81% $((3,918+8,809-500)/(12,750-500))$. We can do the following rough calculation to see the effects

after the nontrivial accuracy increases from 98.51% to 99.81% for the investigated data of June 14, 2010. With reference to the queries, the number of correctly identified queries is expected to increase by about $61,699$ ($(5,552,891 - 806,799) \times (0.9981 - 0.9851)$), which is still a large number. In terms of the users, among the $135,241$ users who submitted two queries on that day (which is shown in Table 5.1), the probability that a user has all his queries correctly identified will roughly increase from $0.9851^{(2-1)}$ to $0.9981^{(2-1)}$ so that about $1,758$ ($135,241 \times (0.9981 - 0.9851)$) more users are expected to have all their queries correctly identified. Similarly, among the $84,107$ users who submitted three queries on that day, by assuming that the identification results of multiple queries from a user are independent and identically distributed (iid), the probability that a user has all his queries correctly identified will roughly increase from $0.9851^{(3-1)}$ to $0.9981^{(3-1)}$ so that about $2,168$ ($84,107 \times (0.9981^2 - 0.9851^2)$) more users are expected to have all their queries correctly identified. Applying the same logic to all the $525,714$ users who submitted $2 \sim 50$ queries on that day, roughly $38,078$ $(1,758 + 2,168 + \cdots + 182)$ more users on that day are expected to have all their queries correctly identified so that they will get the better service from our search engine in the future.

Remark 4 *About Our Manual Session Identification Process.*

The manual session identification for our sample of 12,750 queries is performed in the following way. First, to ease our manual check, we build a visualization tool which generates one html page for each selected user. Each html page lists all the queries from the corresponding user according to the ascending order of their submission time. Inside each html page different colors are used to highlight the changes of terms of two consecutive queries. Then each session is manually identified based on the combination of syntax and semantics.

(1) We manually check whether a common term between two consecutive queries Q_i and Q_{i+1} is a significant word which helps to indicate the same entity of person, company, school, or job title based on its relationship to the semantics of the queries. If it is not a significant word, we will omit such a common term because its existence can not imply the continuation of a session. If it is, then we assign Q_i and Q_{i+1} into the same session.

(2) We manually perform the similarity check for significant terms. On the basis of the syntax as well as the context of two consecutive queries, we usually can relatively easily recognize whether two

terms are similar or not.

(3) The most challenging cases to our manual check are when there is semantic connection between T in Q_i and some term T' in Q_{i+1} but T and T' are not similar at all in their surface forms. One real example is as follows: Q_i is "internal audit" and Q_{i+1} is "kpmg." Based on the surface forms a session boundary should be inserted between Q_i and Q_{i+1}, but in semantics "kpmg" is a company providing the audit service. For these cases, we typically use auxiliary relevant information stored in our system (such as the background information of the corresponding user) to decide whether to insert a session boundary. For the above example, because the user is actually a recruiter in financial services, we can reasonably assume that the user has one information purpose to search persons suitable for a particular job title "internal audit" and another different information purpose to search persons inside the company "kpmg" so that a session boundary is needed. Fortunately, these challenging situations occur rarely so that our additional effort of manually checking auxiliary relevant information is not tremendous.

Remark 5 *About Our Identification Solution for Empty Queries.*

In those cases in which two consecutive queries involve at least one empty query, different session identification solutions have been used by different researchers based on the contexts of studied applications. In this paper, our solution is based on the context of LinkedIn people search domain.

The service of empty query search is provided by LinkedIn people search in the following way. If a user submits an empty query, he will get the search result including the professional information of all the people directly or indirectly connected in the user's social network. According to this context, we use the following strategy to address the situation when at least one query is empty in the two consecutive queries Q_i and Q_{i+1}. If exactly one query (either Q_i or Q_{i+1} but not both) is empty (such a situation accounts for about 1.74% of two consecutive queries on the day of June 14, 2010), the user is considered to check all the connections in his network so that Q_{i+1} is regarded as the start of a new session. If both Q_i and Q_{i+1} are empty (such a situation constitutes about 4.53% of two consecutive queries on the day of June 14, 2010), the user is considered to continue checking all the connections in his network so that Q_i and Q_{i+1} are regarded as being in the same session. This strategy is used in our manual check as well as both the original content-based method (the pseudo code from Line 12 to Line

15 *in Algorithm 2) and our refined content-based method (the pseudo code from Line* 16 *to Line* 19 *in Algorithm 3).*

5.4 Evaluation of Time-based Session Identification Method

Because our refined content-based session identification method is highly accurate, its identification result can be regarded as the approximate gold-standard session identification for our LinkedIn daily people search log file, which includes millions of records and makes manual checking infeasible. We are then able to evaluate the performance of the time-based session identification method with different time-out thresholds for our daily people search log file. (Remark 6 at the end of this section will explicitly show that the evaluation result based on the true gold-standard session identification is very similar to the one based on our refined content-based identification method.) Because the time-based method is extremely cheap in time and space, it will become an effective session identification method in our everyday application if its performance is not much different from its content-based counterpart. Therefore, we intend to find out the optimal time-out threshold and examine its corresponding identification performance in our domain.

To make our experimental results more meaningful and useful, we preprocess our daily data from June 14, 2010 to June 20, 2010 in the following way. We exclude all the records from the users who submitted more than 500 people search queries a day because these users are very likely to be software agents. Note that here we use the large and conservative threshold of 500, which is much larger than the threshold of 99/100 used in previous papers (Jansen and Spink, 2005; Jansen et al., 2005, 2007). The reason of using the large threshold of 500 is that in our domain there are some users (especially recruiters or urgent job-hunters) who have bought their business accounts and submit a lot of search queries every day. With the conservative threshold of 500, we ensure that we will not discard queries from these important and active users, though many software agents' queries will also not be discarded at the same time.

The experimental results based on the preprocessed data on June 14, 2010 are shown in Fig. 5.2, where the preprocessed data on that day includes 5,472,206 search queries from 806,691 users. (Again, we assume that all the records are ordered by (member id, tracking time).) We vary the time-out

thresholds from 0.25 to 30 minutes. We increase thresholds with 0.25-minute increments in the interval from 0.25 to 5 minutes and then increase thresholds with 0.5-minute increments in the interval from 5.5 to 30 minutes. Note that the thresholds we have investigated include those commonly used thresholds in other papers (Silverstein et al., 1999; Jansen et al., 2007).

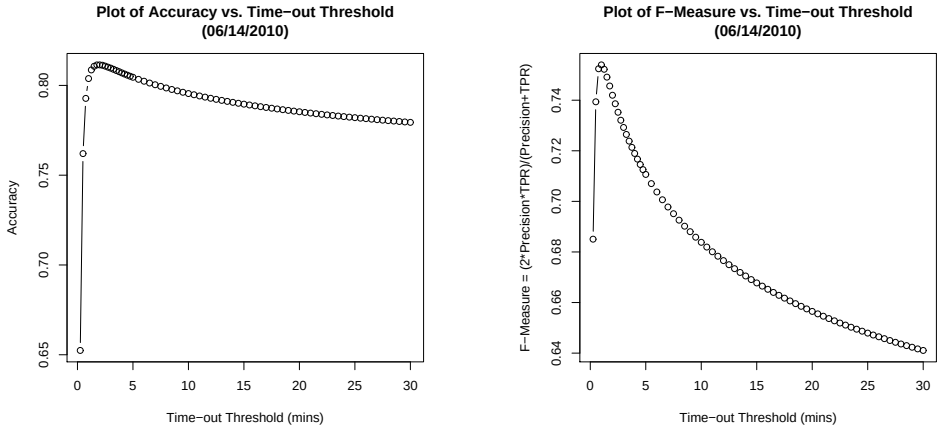
Fig. 5.2 (a) shows that the accuracy rate increases quickly in the threshold interval from 0.25 minute to 2 minutes and then decreases slowly afterwards. The optimal accuracy rate 81.14% of time-based method is achieved (with TP = 1,483,601, FP = 273,032, TN = 2,956,688, FN = 758,885) when the threshold is set at 2 minutes. The four thresholds between 1.75 minutes and 2.5 minutes are the four best thresholds whose corresponding accuracy rates are all above 81.07%. Fig. 5.2 (b) demonstrates that the F-measure increases sharply in the threshold interval from 0.25 minute to 1 minute and then decreases steeply afterwards. The optimal F-measure 75.40% is reached (with TP = 1,644,972, FP = 476,073, TN = 2,753,647, FN = 597,514) when the threshold is set at 1 minute. The four thresholds between 0.75 minute and 1.5 minutes are the four best thresholds whose corresponding F-measures are all above 74.91%.

Our experiments show that the optimal time-out thresholds for both accuracy rate and F-measure are very small in our domain. In addition, it can be seen that the optimal time-out threshold for F-measure is even smaller than the one for accuracy rate. The question about which performance measure is better to use depends on the application scenario and is beyond the scope of this paper. Once the appropriate measure is chosen, the corresponding best time-out threshold can be chosen to optimize its performance. Therefore, if in some real application the overall optimal performance such as 81.14% accuracy rate or 75.40% F-measure is acceptable, then the time-based method can be used with its extreme cheap costs in time and space.

The experiments based on the preprocessed daily data from June 15, 2010 to June 20, 2010 show the similar patterns. Please refer to Appendix B for the details.

Remark 6 *Experimental Results based on the True Gold-standard Identification*

We also perform similar evaluation experiments on the sample of 12,750 records, for which we have manually made the identification as the gold-standard in Section 5.3.5. The optimal accuracy rate 80.11% *of the time-based method can be reached when the time-out threshold is set at 2.25 or*



(a) Plot of Accuracy vs. Time-out

(b) Plot of F-measure vs. Time-out

Figure 5.2: Plot of Performance vs. Time-out on June 14, 2010

3.5 minutes. All the seven thresholds between 2.25 and 3.75 minutes are good thresholds since their corresponding accuracy rates are all above 80.03%*. The optimal F-measure* 65.32% *can be achieved when the threshold is set at 0.75 minute. The four thresholds between 0.5 and 1.25 minutes are the four best thresholds whose corresponding F-measures are all above* 64.15%*.*

Note that if we use the approximate gold-standard identification from our refined content-based method on the sample of 12,750 records, the evaluation results will be very similar to the ones based on the manual gold-standard identification. Based on the approximate gold-standard identification, the optimal accuracy rate is 79.89% *when the time-out threshold is set at 2.25 minutes or 3.5 minutes; the optimal F-measure is* 64.90% *when the time-out threshold is set at 0.75 minute. This indicates that our refined content-based method, with its excellent identification performance, can be used to evaluate the time-based method and to find the corresponding optimal time-out threshold.*

Please also note that these optimal values for the sample of 12,750 records are not exactly the same as the ones for preprocessed data including 5,472,206 records on June 14, 2010. The underlying reason is that in the manually identified sample we intentionally increase the mean of X (i.e., the mean number of search queries from a user on that day) to lower the percentage of correctly but trivially identified records, as we have described in Section 5.3.5. As a result, the increased mean of X changes the corresponding optimal thresholds, and the lowered percentage of correctly but trivially identified

Table 5.4: Descriptive Statistics of Session Length

Date	Min	1st Qu.	Median	Mean	3rd Qu.	Max	SD
06/14/2010	1	1	1	2.44	2	413	4.82
06/15/2010	1	1	1	2.41	2	497	4.79
06/16/2010	1	1	1	2.41	2	482	4.74
06/17/2010	1	1	1	2.44	2	469	4.90
06/18/2010	1	1	2	2.67	3	459	4.82
06/19/2010	1	1	1	2.50	2	399	4.58
06/20/2010	1	1	1	2.51	2	390	4.61

records decreases the corresponding performance of the time-based method.

5.5 Statistics of Sessions

In this section, on the basis of our refined content-based session identification method, we analyze the following four features of the sessions in LinkedIn people search: (1) the session length (the number of queries in one session); (2) the session duration (the period from the submission of the first query to the submission of the last query in one session); (3) the time gap between two sessions, denoted by TG^B; and (4) the time gap between two queries inside the same session, denoted by TG^I. All these four features are random variables whose distributions demonstrate the characteristics of people search in our domain. We get the (empirical) distributions of these four features based on the same preprocessed daily data from June 14, 2010 to June 20, 2010, which have been used in Section 5.4.

5.5.1 Session Length & Duration

The distribution of session length is shown in Table 5.4, 5.5 and Fig. 5.3. They clearly indicate that the session length is usually very small. For example, on each day from June 14, 2010 to June 20, 2010, the mean of session length is no greater than 2.67; on the day of June 14, 2010, 60.554% of session length is 1, 86.016% of session length is no greater than 3, and 96.956% of session length is no greater than 10. Compared with the distribution of X shown in Table 5.1, the distribution of session length shown in Table 5.5 has more probability mass on the smaller values. This implies that users usually submit a very small number of queries for one information need. Note that the fact that the maximum

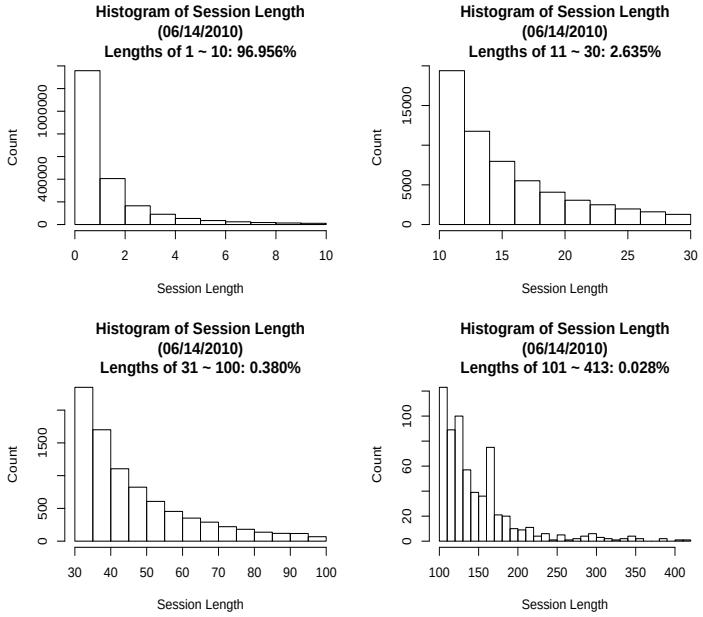


Figure 5.3: Histogram of Session Length on June 14, 2010

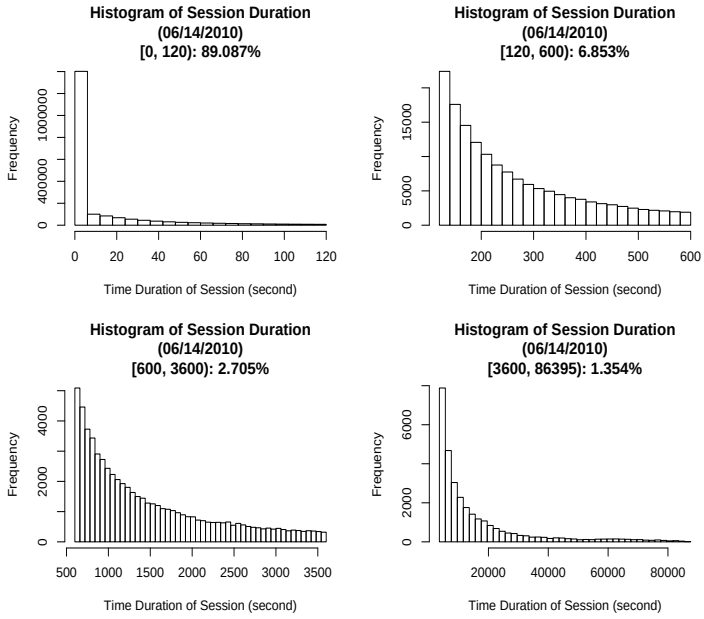


Figure 5.4: Histogram of Session Duration on June 14, 2010

Table 5.5: Distribution of Session Length on June 14, 2010

Session Length	Count	Percentage	Session Length	Count	Percentage
1	1,357,916	60.554%	8 - 10	41,901	1.869%
2	405,631	18.088%	11 - 20	48,694	2.171%
3	165,342	7.373%	21 - 30	10,401	0.464%
4	90,575	4.039%	31 - 40	4,048	0.181%
5	53,554	2.388%	41 - 50	1,929	0.086%
6	35,117	1.566%	51 - 100	2,546	0.114%
7	24,195	1.079%	101 - 413	637	0.028%

Table 5.6: Descriptive Statistics of Session Duration (in Seconds)

Date	Min	1st Qu.	Median	Mean	3rd Qu.	Max	SD
06/14/2010	0	0	0	279.6	26.8	86,394.6	2,617.4
06/15/2010	0	0	0	271.9	25.8	86,231.0	2,556.7
06/16/2010	0	0	0	262.7	26.1	86,012.8	2,398.4
06/17/2010	0	0	0	260.9	25.8	86,193.0	2,455.2
06/18/2010	0	0	0.003	217.8	25.7	86,308.1	2,034.8
06/19/2010	0	0	0	191.0	23.2	85,244.9	2,177.3
06/20/2010	0	0	0	210.7	22.9	86,316.3	2,343.8

of session length can reach hundreds (such as 413) is very likely caused by software agents.

In addition, based on the Kolmogorov-Smirnov two-sample test (Conover, 1999), we can also conclude that the distribution of session length on each day from June 15, 2010 to June 20, 2010 is the same as the one on June 14, 2010. (Please refer to Appendix C for the details.)

Table 5.6 and Fig. 5.4 show the distribution of session duration. Note that according to the definition of session duration, session duration will be 0 second if the length of the session is 1. Table 5.6 and Fig. 5.4 clearly indicate that session duration is usually small. For example, on each day from June 14, 2010 to June 20, 2010, the mean session duration is less than 280 seconds and the 3rd quartile is less than 27 seconds; on June 14, 2010, 89.087% of session duration is less than 2 minutes. This implies that users spend an extremely short time on one information need in our domain. We notice that the maximum of session duration can be close to 24 hours (86,400 seconds), which mainly comes from the situation that some users around the world would continue to search for the same information needs on the next day

Table 5.7: Comparison between Our Domain and General Web Search Reported by Jansen et al. (2007)

Percentage of Session Length			Percentage of Session Duration		
Length	Our Domain	General Web Search	Duration	Our Domain	General Web Search
1	60.55%	71.64%	$<$ 1 min	83.46%	82.32%
2	18.09%	15.85%	1 to $<$ 5 min	10.36%	8.94%
3	7.37%	6.06%	5 to $<$ 10 min	2.12%	2.90%
4	4.04%	2.81%	10 to $<$ 15 min	0.87%	1.27%
5	2.39%	1.47%	15 to $<$ 30 min	1.10%	1.42%
6	1.57%	0.80%	30 to $<$ 60 min	0.73%	1.31%
7	1.08%	0.46%	60 to $<$ 120 min	0.53%	0.78%
8	0.78%	0.29%	120 to $<$ 180 min	0.23%	0.34%
9	0.60%	0.18%	180 to $<$ 240 min	0.14%	0.20%
$\geq$10	3.52%	0.42%	$\geq$ 240 min	0.45%	0.50%

in their time zones.

The comparison between the session length or duration in our domain and the one in general web search also demonstrates the characteristics of our domain. Here we use the session length and duration generated from the content-based method (Method (3)) of Jansen et al. (2007) as the one in general web search for the purpose of the comparison. The session length and duration in our domain are generated from our preprocessed daily data on June 14, 2010. The detailed comparison results are shown in Table 5.7.

In terms of session length, Table 5.7 clearly shows that the session length in our domain has relatively larger probability (percentage) than the one in general web search of having a value greater than 1. For example, session length 6 has the probability 1.57% in our domain, almost doubling the probability 0.80% in general web search. When session length increases to 9, the probability becomes 0.60% in our domain, three times more than the probability 0.18% in the general search. We believe that such a difference is mainly from the profession-oriented characteristics of our domain. Our domain has more serious searchers such as recruiters or job-hunters who are more likely to keep searching for every information need until they get satisfactory search results.

Concerning session duration, Table 5.7 indicates that the session duration in our domain has relatively smaller probability than the one in general web search of having a value no less than five minutes. Again, we believe that such a difference comes from the characteristics of our domain. A user in our

Table 5.8: Descriptive Statistics of TG^B (Time Gap between Sessions) (in Seconds)

Date	Min	1st Qu.	Median	Mean	3rd Qu.	Max	SD
06/14/2010	0.0	26.8	99.3	2,635.4	842.2	86,284.3	8,216.5
06/15/2010	0.0	25.7	90.7	2,553.3	759.9	86,119.1	8,070.1
06/16/2010	0.0	25.9	91.9	2,444.6	755.3	86,302.7	7,640.5
06/17/2010	0.0	25.8	90.3	2,479.6	730.9	86,277.8	7,911.0
06/18/2010	0.0	24.9	79.8	2,016.0	574.7	86,213.0	6,491.3
06/19/2010	0.0	21.5	54.0	1,761.2	244.3	85,665.8	7,102.3
06/20/2010	0.0	21.9	58.7	2,001.3	301.1	86,055.0	7,728.5

Table 5.9: Descriptive Statistics of TG^I (Time Gap between Queries inside the Same Session) (in Seconds)

Date	Min	1st Qu.	Median	Mean	3rd Qu.	Max	SD
06/14/2010	0.0	8.8	16.3	194.2	33.8	85,951.9	2,068.4
06/15/2010	0.0	8.8	16.3	192.4	33.8	85,684.5	2,040.1
06/16/2010	0.0	8.8	16.4	186.6	34.0	85,935.9	1,911.7
06/17/2010	0.0	8.6	16.1	181.4	33.4	86,193.0	1,934.8
06/18/2010	0.0	4.6	12.9	130.2	27.8	84,932.4	1,476.8
06/19/2010	0.0	5.2	13.2	127.5	27.3	84,514.2	1,687.9
06/20/2010	0.0	5.1	13.2	139.4	27.4	83,618.5	1,818.6

domain has a relatively simple and profession-related purpose of searching right person(s) within a professional social network, rather than those diversified purposes such as seeking for some entertainment-related material which a user would spend more time in reading. Therefore, a user in our domain tends to finish searching for a single information need faster than a user in general search.

5.5.2 Time Gap TG^B & TG^I

The distribution of TG^B (time gap between two sessions) is shown in Table 5.8 and Fig. 5.5 whereas the distribution of TG^I (time gap between two queries inside the same session) is shown in Table 5.9 and Fig. 5.6. The difference of these two distributions is significant and TG^B tends to be much larger than TG^I. Such a difference can be easily seen by comparing Fig. 5.5 with Fig. 5.6. For instance, on June 14, 2010, 91.546% of TG^I is in the interval less than 2 minutes while only 52.855%

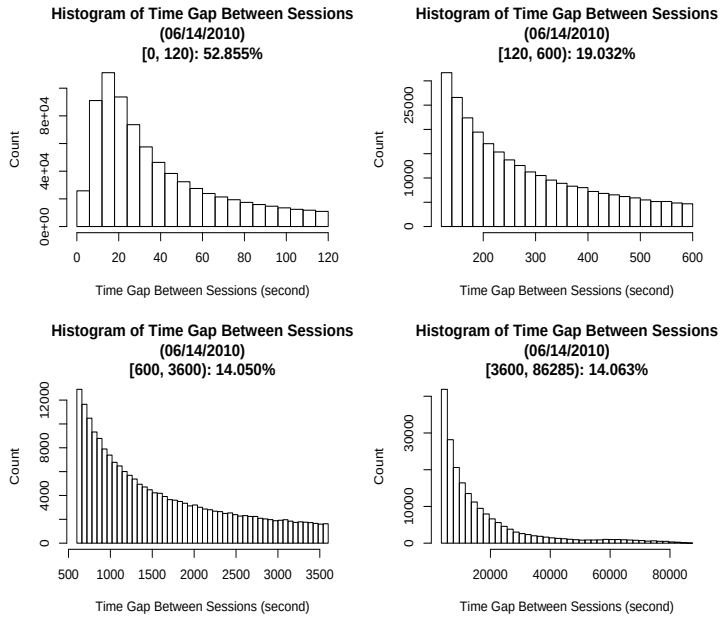


Figure 5.5: Histogram of TG^B on June 14, 2010

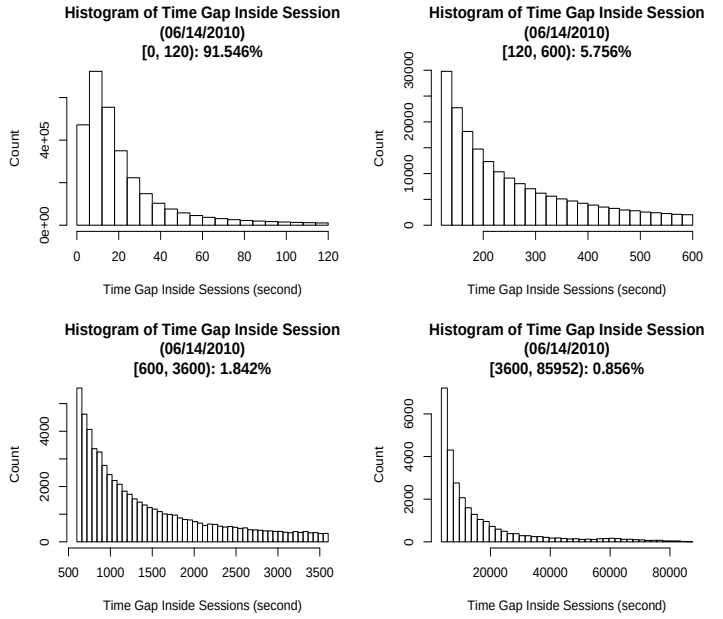


Figure 5.6: Histogram of TG^I on June 14, 2010

Table 5.10: Conditional Probability that a Time Gap TG is TG^B Given that TG Is in the Specified Time Interval T_i (T_i Is in Seconds)

i	Interval T_i	$M_i^B + M_i^I$	Cond. Pr. of TG^B	i	Interval T_i	$M_i^B + M_i^I$	Cond. Pr. of TG^B
1	[0, 15)	1,675,173	0.1037	16	[270, 300)	28,302	0.6135
2	[15, 30)	1,043,020	0.2124	17	[300, 360)	45,880	0.6313
3	[30, 45)	415,879	0.2977	18	[360, 420)	36,365	0.6473
4	[45, 60)	217,089	0.3608	19	[420, 480)	29,233	0.6676
5	[60, 75)	136,550	0.4038	20	[480, 540)	24,283	0.6806
6	[75, 90)	96,888	0.4424	21	[540, 600)	21,085	0.6965
7	[90, 105)	72,990	0.4720	22	[600, 900)	74,007	0.7181
8	[105, 120)	57,984	0.4987	23	[900, 1200)	45,880	0.7531
9	[120, 135)	47,439	0.5113	24	[1200, 1500)	32,480	0.7759
10	[135, 150)	40,029	0.5337	25	[1500, 1800)	24,822	0.7887
11	[150, 165)	34,313	0.5460	26	[1800, 3600)	84,030	0.8243
12	[165, 180)	29,522	0.5517	27	[3600, 7200)	76,006	0.8576
13	[180, 210)	49,316	0.5726	28	[7200, 14400)	67,776	0.8869
14	[210, 240)	39,937	0.5914	29	[14400, 28800)	51,594	0.8997
15	[240, 270)	33,455	0.6029	30	[28800, 86400)	34,188	0.8833

of TG^B is in the same interval. Such a difference will also be seen by comparing some statistic (such as median/mean) of TG^B with the one of TG^I. For example, on each day from June 14, 2010 to June 20, 2010, the median of TG^B (shown in Table 5.8) is much larger than the median of TG^I (shown in Table 5.9).

The difference between TG^B and TG^I will become even clearer if we examine the conditional probability that a time gap TG is TG^B instead of TG^I given that the value of TG is known. For demonstration purposes, in Table 5.10 we partition 24-hour period of June 14, 2010 into 30 time intervals and then show the corresponding (estimated) conditional probability of being TG^B in each time interval T_i, that is, the (estimated) conditional probability that a time gap TG is TG^B given that the value of TG is in the interval T_i. The conditional probability of being TG^B in each T_i ($i \in \{1, 2, \cdots, 30\}$) can be computed by the following formula:

$$\frac{M_i^B}{M_i^B + M_i^I},$$

where $M_i^B = |\{TG^B : TG^B \in T_i\}|$, denoting the number of TG^B's that are in T_i; $M_i^I = |\{TG^I : TG^I \in T_i\}|$, denoting the number of TG^I's that are in T_i. In other words, the conditional probability of being TG^B in each T_i is the ratio of the number of TG^B's that are in T_i to the total number of time

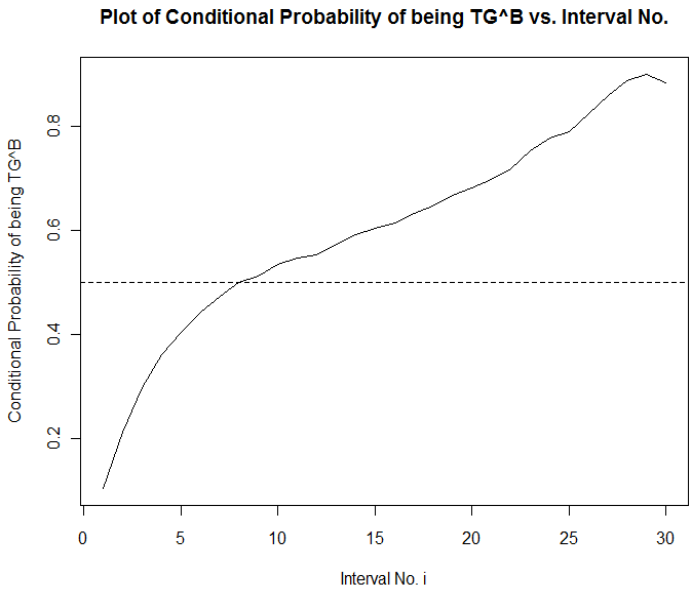


Figure 5.7: Conditional Probability of Being TG^B on June 14, 2010

gaps that are in T_i.

From Table 5.10, we can see that the conditional probability of being TG^B in each T_i increases as i increases, except for the last $i = 30$. The corresponding plot of the conditional probability of being TG^B versus the interval number i is also shown in Fig. 5.7. It implies that the larger a time gap TG is, the more likely the time gap is a gap between two sessions. This indicates the underlying rationale of the time-based method: in most cases we are able to successfully distinguish whether or not a given time gap is between two different sessions merely based on its value. Note that the conditional probability that a time gap TG is TG^B starts being larger than 0.5 when the value of TG is no less than the threshold of 120 seconds, that is, $TG \in \cup_{i=9}^{30} T_i$. Based on decision theory framework (Bickel and Doksum, 2001), since the zero-one loss is used for accuracy rate, this 120-second threshold essentially equals the time-out threshold for the optimal accuracy rate computed in Section 5.4.

When the threshold is set at 120 seconds, from Table 5.10, the accuracy rate can be computed as $(\sum_{i=9}^{30} M_i^B + \sum_{i=1}^{8} M_i^I)/\sum_{i=1}^{30}(M_i^B + M_i^I) = 77.88\%$, which is actually consistent with our evaluation result in Section 5.4. Remember that for 5,472,206 queries in preprocessed data on June 14, 2010, there are 806,691 trivial TP's (true positives) coming from 806,691 users. If we exclude

these trivial TP's, the optimal accuracy rate computed in Section 5.4 will exactly decrease into 77.88% (with TP = $1,483,601 - 806,691 = 676,910$, FP = 273,032, TN = 2,956,688, FN = 758,885) with 2-minute threshold. Note that our optimal nontrivial accuracy rate for the time-based method, 77.88%, is much worse than the counterpart obtained from Jansen et al. (2007). In their study, the accuracy rate is 98.85% when the time-out threshold is set at 30 minutes for 2,000 manually identified queries. Since their reported mean of X (the number of queries per user) is 2.85, there are about $2,000/2.85$ trivial TP's. If these trivial TP's are excluded, the nontrivial accuracy rate in their study will become $(2,000 \times 98.95\% - 2,000/2.85)/(2,000 - 2,000/2.85) = 98.38\%$. Therefore, the optimal nontrivial accuracy rate for the time-based method in our domain, which can only reach 77.88% due to the feature of our domain, is much worse than the nontrivial accuracy rate for the time-based method in the domain studied by Jansen et al. (2007).

The similar relation between the conditional probability of being TG^B and the interval number i is found on each day from June 15, 2010 to June 20, 2010. Please refer to Appendix D for the details.

5.6 Summary & Future Work

We perform session analysis on the domain of people search within LinkedIn professional social network. We propose two effective and efficient refinements to the original content-based session identification method and show the excellent identification performance of the refined method in our domain. Using our refined content-based method as the gold-standard, we then evaluate the performance of the time-based session identification method and find the optimal time-out threshold in our domain. Finally, we analyze several features of identified sessions to provide more insights into the nature of our domain.

In the future, we will first investigate possible solutions for identifying the semantic connection between two consecutive queries from a user, whose occurrences constitute the most challenging cases of the session identification in our domain. (Please see Remark 4 in Section 5.3.5.) For example, a user submitting the query "software developer" followed by the query "java expert" has more likely the same information need in these two queries, while neither the original content-based method nor our refined content-based method can address this case. A solution to this kind of cases can improve further the identification performance of our refined content-based method. Second, we will develop

approaches to improve the performance of our search engine based on the contextual information gained from identified sessions in our domain. This study will have a direct impact on real-world applications in our industry. Third, we intend to combine a wide variety of information (such as users' background information) stored in our system to investigate the underlying factors that affect the users' search behavior in our domain. The identified factors may enable us to provide customized search service for various users.

5.7 Appendix

Appendix A

The N-gram algorithms that we use in Section 5.3.5 are described in Algorithm 4. Given two terms term1 and term2 and one pre-specified positive integer N, Algorithm 4 will output a Boolean value to indicate whether term1 and term2 are similar.

In our algorithm the function f(minStrLen, N) is used to set requiredNumOfSimilarNgrams, the required number of common N-grams in A1 and A2, which serves as the similarity measure between A1 and A2. The following four forms of f (including f_1, f_2, f_3 and f_4) are tested in our experiments. $f_1 \equiv 1$ sets requiredNumOfSimilarNgrams to be the constant threshold 1, which says that A1 and A2 are regarded to be similar as long as there exists at least 1 common N-gram in A1 and A2. Let I[$\cdot$] be an indicator function. $f_2 = 1 + \text{I}[\text{minStrLen} - \text{N} \geq 2]$ applies the constant increase 1 to requiredNumOfSimilarNgrams if the condition minStrLen $-$ N ≥ 2 is true. $f_3 = 1 + \lfloor(\text{minStrLen} - \text{N})/2\rfloor$ and $f_4 = 1 + \lfloor(\text{minStrLen} - \text{N}) \cdot (4/5)\rfloor$ apply linear increase with the corresponding increase rates $1/2$ and $4/5$ to requiredNumOfSimilarNgrams respectively. The performance of our N-gram algorithms with different N's and f's based on the sample of $12,750$ queries is listed in Table 5.11. Note that for each N $\in \{3, 4, 5\}$, we select the performance of N-gram algorithm with its best f choice and list it from the third row to the fifth row in Table 5.3 respectively. For the combined similarity-check algorithm in the last row of Table 5.3, we use 4-gram with f_2 choice for the second-round similarity check since such a setting has the best performance in Table 5.11.

Table 5.11: Performance of N-gram Algorithms with Different N's and f's (Based on the Sample of $12,750$ Queries)

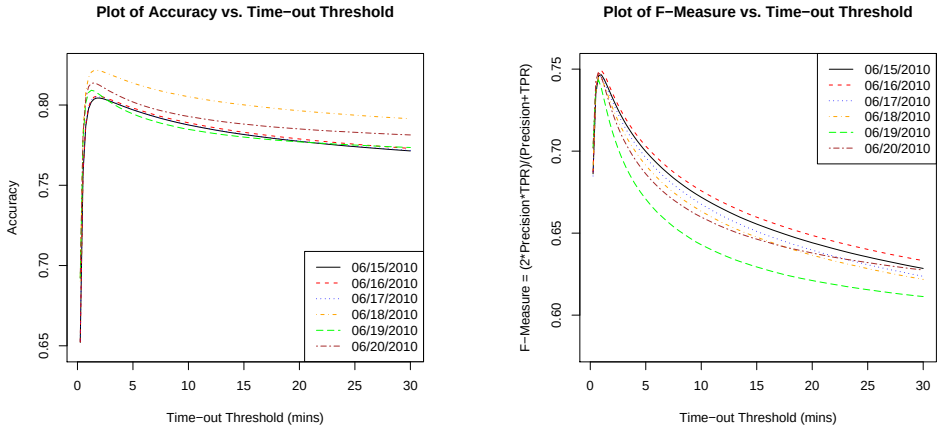
Method	TP	FP	TN	FN	No. of Errors	Accuracy	F-measure
Filter-Out & 3-gram with f_1	3,671	9	8,810	260	269	97.890%	96.466%
Filter-Out & 3-gram with f_2	3,863	15	8,804	68	83	99.349%	98.937%
Filter-Out & 3-gram with f_3	3,881	23	8,796	50	73	99.427%	99.068%
Filter-Out & 3-gram with f_4	3,891	43	8,776	40	83	99.349%	98.945%
Filter-Out & 4-gram with f_1	3,903	22	8,797	28	50	99.608%	99.364%
Filter-Out & 4-gram with f_2	3,924	33	8,786	7	40	99.686%	99.493%
Filter-Out & 4-gram with f_3	3,927	42	8,777	4	46	99.639%	99.418%
Filter-Out & 4-gram with f_4	3,928	62	8,757	3	65	99.490%	99.179%
Filter-Out & 5-gram with f_1	3,926	46	8,773	5	51	99.600%	99.355%
Filter-Out & 5-gram with f_2	3,929	58	8,761	2	60	99.529%	99.242%
Filter-Out & 5-gram with f_3	3,930	64	8,755	1	65	99.490%	99.180%
Filter-Out & 5-gram with f_4	3,930	77	8,742	1	78	99.388%	99.017%

Appendix B

The experimental results for the time-based method based on the preprocessed daily data from June 15, 2010 to June 20, 2010 are shown in Fig. 5.8. It can be seen that the shape of everyday curve is somewhat similar. For the detailed comparison, Table 5.12 lists, for each examined day, Accuracy^{opt} (the optimal accuracy), Threshold$^{opt}_{Acc}$ (the time-out threshold for the optimal accuracy), F-Measure^{opt} (the optimal F-measure), and Threshold$^{opt}_{F-M}$ (the time-out threshold for the optimal F-measure). Table 5.12 shows that the optimal threshold (Threshold$^{opt}_{Acc}$ /Threshold$^{opt}_{F-M}$) is generally quite similar on each day of the examined week but it does decrease slightly at the weekend.

Appendix C

In Section 5.3.1 we use the Kolmogorov-Smirnov (KS) two-sample test (Conover, 1999) to check whether the distribution of X on each day from June 15, 2010 to June 20, 2010 is the same as the distribution of X on June 14, 2010. The KS two-sample test is a commonly used statistical test to check whether two examined distributions are the same. The test has its null hypothesis H_0 that two examined distributions are the same. Then the corresponding KS statistic can be computed as the measure of the difference of two distributions. If KS statistic is larger than the threshold set from a specified level



(a) Plot of Accuracy vs. Time-out

(b) Plot of F-measure vs. Time-out

Figure 5.8: Plot of Performance vs. Time-out from June 15, 2010 to June 20, 2010

α such as 0.05, we can reject H_0 that two distributions are the same with the corresponding level. Otherwise, we will conclude H_0 that two distributions are the same. Our test results are shown in Table 5.13. Since the KS statistic in each row of Table 5.13 is much smaller than the corresponding threshold, we can conclude that the distribution of X on each day from June 15, 2010 to June 20, 2010 is the same as the one on June 14, 2010.

Similarly, we use the Kolmogorov-Smirnov two-sample test to check whether the distribution of session length on each day from June 15, 2010 to June 20, 2010 is the same as the one on June 14, 2010 in Section 5.5. The corresponding test results are shown in Table 5.14. On one hand, we still can conclude that the distribution of session length on each day from June 15, 2010 to June 20, 2010 is the same as the one on June 14, 2010 since the KS statistic in each row of Table 5.14 is smaller than the corresponding threshold. On the other hand, the increased KS statistics at the weekend (from June 18, 2010 to June 20, 2010) shows that there is an increase in the difference between the distribution of session length on June 14, 2010 and the corresponding one at the weekend.

Appendix D

Using the similar logic in Section 5.5.2, Fig. 5.9 shows the curve of the conditional probability of being TG^B versus the interval number i on each day from June 15, 2010 to June 20, 2010. Generally

Table 5.12: Optimal Performances and Their Corresponding Time-out Thresholds (in Minutes) from June 14, 2010 to June 20, 2010

Date	Accuracy^{opt}	Threshold$^{opt}_{Acc}$	F-Measure^{opt}	Threshold$^{opt}_{F-M}$
06/14/2010	81.14%	2	75.40%	1
06/15/2010	80.43%	1.75	74.66%	1
06/16/2010	80.55%	1.75	74.93%	1
06/17/2010	80.51%	1.75	74.55%	1
06/18/2010	82.17%	1.75	74.51%	0.75
06/19/2010	80.92%	1.25	74.33%	0.75
06/20/2010	81.37%	1.5	74.84%	0.75

Table 5.13: Comparison of X's Distribution over a Week (Based on Kolmogorov-Smirnov Two-sample Test)

vs. 06/14/2010	KS statistic	Threshold (α = 0.1)	Threshold (α = 0.05)	H_0
06/15/2010	0.0101	0.0749	0.0835	Conclude
06/16/2010	0.0062	0.0758	0.0845	Conclude
06/17/2010	0.0168	0.0745	0.0830	Conclude
06/18/2010	0.0474	0.0754	0.0841	Conclude
06/19/2010	0.0108	0.0833	0.0928	Conclude
06/20/2010	0.0088	0.0817	0.0910	Conclude

speaking, each of these six curves shows the similar trend as the curve on June 14, 2010. Again, on the basis of decision theory framework, for each day from June 15, 2010 to June 20, 2010, the threshold over which the conditional probability of being TG^B starts being larger than 0.5 is the same as the corresponding time-out threshold for the optimal accuracy rate listed in Table 5.12.

Appendix E

In this appendix we describe how to draw the conclusion that the accuracy rate of our refined content-based method (Method 2) is statistically significantly better than the accuracy rate of the original content-based method (Method 1), which has been claimed in the subsection Evaluation of Our Content-based Method.

First, we use the following notation for clarity. Let p_{CC} denote the probability that a query will be

Table 5.14: Comparison of the Distribution of Session Length over a Week (Based on Kolmogorov-Smirnov Two-sample Test)

vs. 06/14/2010	KS statistic	Threshold (α = 0.1)	Threshold (α = 0.05)	H_0
06/15/2010	0.0061	0.1090	0.1215	Conclude
06/16/2010	0.0055	0.1112	0.1239	Conclude
06/17/2010	0.0009	0.1089	0.1214	Conclude
06/18/2010	0.1100	0.1118	0.1246	Conclude
06/19/2010	0.0732	0.1195	0.1332	Conclude
06/20/2010	0.0723	0.1185	0.1321	Conclude

correctly identified by both Method 1 and Method 2. Let p_{II} denote the probability that a query will be incorrectly identified by both Method 1 and Method 2. Let p_{CI} denote the probability that a query will be correctly identified by Method 1 but will be incorrectly identified by Method 2. Let p_{IC} denote the probability that a query will be incorrectly identified by Method 1 but will be correctly identified by Method 2.

Accordingly, given totally N queries, let N_{CC} denote the number of queries that are correctly identified by both Method 1 and Method 2; let N_{II} denote the number of queries that are incorrectly identified by both Method 1 and Method 2; let N_{CI} denote the number of queries that are correctly identified by Method 1 but are incorrectly identified by Method 2; let N_{IC} denote the number of queries that are incorrectly identified by Method 1 but are correctly identified by Method 2.

Note that the random vector $(N_{CC}, N_{II}, N_{CI}, N_{IC})$ has the Multinomial distribution with the total number N and probability vector $(p_{CC}, p_{II}, p_{CI}, p_{IC})$, denoted by $((N_{CC}, N_{II}, N_{CI}, N_{IC}) \sim Multinomial(N; p_{CC}, p_{II}, p_{CI}, p_{IC})$, where $N = N_{CC} + N_{II} + N_{CI} + N_{IC}$ and $p_{CC} + p_{II} + p_{CI} + p_{IC} = 1$.

Here we set up the null hypothesis H_0 that $p_{CC} + p_{IC} = p_{CC} + p_{CI}$, that is, $p_{IC} = p_{CI}$, which says that the accuracy of Method 2 is the same as the accuracy of Method 1. Correspondingly, the alternative hypothesis H_A is the claim that $p_{CC} + p_{IC} > p_{CC} + p_{CI}$, that is, $p_{IC} > p_{CI}$, which says that the accuracy of Method 2 is better than the accuracy of Method 1.

Conditioning on the number of queries for which the Method 1 and Method 2 disagree, $N^* = N_{CI} + N_{IC}$, the above hypothesis test is performed as the sign test (Conover, 1999) based on the

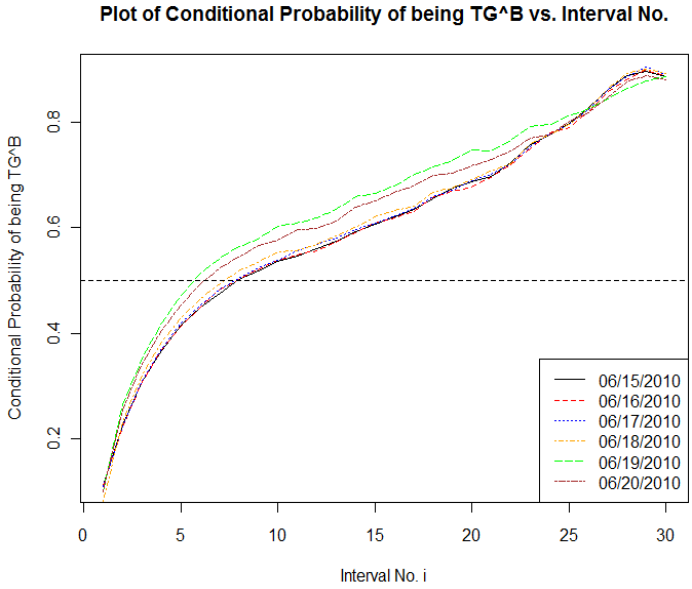


Figure 5.9: Conditional Probability of Being TG^B from June 15, 2010 to June 20, 2010

binomial distribution with sample size N^* and success probability 0.5. In addition, if N^* is large enough, typically N^* is required to be no less than 25, the McNemar test (Conover, 1999) based on the normal approximation can also be used.

For our problem, $N_{CC} = 12555, N_{II} = 11, N_{CI} = 12, and N_{IC} = 172$. Therefore, if we perform the sign test, the sign test statistic $T_{sign} = N_{IC} = 172$. Under H_0, T_{sign} has the binomial distribution with sample size 184 ($N^* = N_{CI} + N_{IC} = 184$) and success probability 0.5, denoted by $T_{sign} \sim Binomial(184, 0.5)$. The corresponding p-value of the one-sided sign test is less than $1e-30$. If we perform the McNemar test, the McNemar test statistic $T_{Mc} = (N_{CI} - N_{IC})^2/(N_{CI} + N_{IC}) = 139.1304$. Under H_0, T_{Mc} has the central chi-square distribution with one degree of freedom, denoted by $T_{Mc} \sim \chi^2(1)$. The corresponding p-value of the one-sided McNemar test is also less than $1e-30$. Since the p-values from both the sign test and the McNemar test are tiny, using either test we can reject H_0 with very strong evidence and conclude that the accuracy of Method 2 is better than the accuracy of Method 1.

The above conclusion may seem surprising at first glance since in Table 5.3 we see that the accuracy of Method 2 (99.820%) is close to the accuracy of Method 1 (98.565%). However, since N, the total

number of queries we have examined, is very large $(12,750)$, the value of N^* is already sufficiently large though it only constitutes a small portion of N. As a result, the tiny p-values are obtained from both tests so that we can safely conclude the alternative hypothesis H_A.

Algorithm 1 isSimilarByHeadOrTail(term1, term2, thresholdLen)

```
str1 ← toLowerCase(term1)
str2 ← toLowerCase(term2)
str1Len ← length(str1)
str2Len ← length(str2)
minStrLen ← min(str1Len, str2Len)
maxStrLen ← max(str1Len, str2Len)
leftLen ← 0
for i ← 0 to minStrLen − 1 do
    if str1[i] = str2[i] then
        leftLen ← leftLen + 1
        if leftLen ≥ thresholdLen then
            break
        end if
    else
        break
    end if
end for
if (leftLen ≥ thresholdLen)
    or (leftLen ≥ 2 and minStrLen = 3 and maxStrLen = 3)
    or (leftLen ≥ 3 and minStrLen ≤ 4 and maxStrLen ≤ 4)
    or . . .
    or (leftLen ≥ thresholdLen − 1 and minStrLen ≤ thresholdLen and maxStrLen ≤ thresholdLen)
 then
    return true
end if
rightLen ← 0
for i ← 0 to minStrLen − 1 do
    if str1[str1Len − 1 − i] = str2[str2Len − 1 − i] then
        rightLen ← rightLen + 1
        if rightLen ≥ thresholdLen then
            break
        end if
    else
        break
    end if
end for
if (rightLen ≥ thresholdLen)
    or (rightLen ≥ 2 and minStrLen = 3 and maxStrLen = 3)
    or (rightLen ≥ 3 and minStrLen ≤ 4 and maxStrLen ≤ 4)
    or . . .
    or (rightLen ≥ thresholdLen − 1 and minStrLen ≤ thresholdLen and maxStrLen ≤ thresholdLen)
 then
    return true
end if
if (leftLen + rightLen ≥ thresholdLen + 1)
    or (leftLen + rightLen ≥ 2 and minStrLen = 3 and maxStrLen = 3)
    or (leftLen + rightLen ≥ 3 and minStrLen ≤ 4 and maxStrLen ≤ 4)
    or . . .
    or (leftLen + rightLen ≥ thresholdLen and minStrLen ≤ thresholdLen + 1 and maxStrLen ≤ thresholdLen +
    1)
 then
    return true
end if
return false
```

Algorithm 2 isSameSession(Q_1, Q_2) (Original Content-based Method)

Require: All the query records are ordered by (member id, tracking time)
Require: Q_1 and Q_2 are two consecutive queries in the ordered records
if Q_1 and Q_2 have different member id's **then**
  return false
end if
if Q_1 and Q_2 are both nonempty **then**
  $TS_1 \leftarrow$ the set of all the terms in Q_1
  $TS_2 \leftarrow$ the set of all the terms in Q_2
  if there exist some term T_{1i} in TS_1 and some T_{2j} in TS_2 such that isEqualIgnoreCase(T_{1i}, T_{2j}) = true **then**
    return true
  else
    return false
  end if
else if Q_1 and Q_2 are both empty **then**
  return true
else
  return false
end if

Algorithm 3 isSameSession(Q_1, Q_2) (Content-based Method with Two Refinements)

Require: All the query records are ordered by (member id, tracking time)
Require: Q_1 and Q_2 are two consecutive queries in the ordered records
if Q_1 and Q_2 have different member id's **then**
  return false
end if
if Q_1 and Q_2 are both nonempty **then**
  $TS_1 \leftarrow$ the set of all the terms in Q_1
  $TS_2 \leftarrow$ the set of all the terms in Q_2
  Filter out all the terms in TS_1 that are also in the stop-word List
  Filter out all the terms in TS_2 that are also in the stop-word List
  if there exist some term T_{1i} in TS_1 and some T_{2j} in TS_2 such that isEqualIgnoreCase(T_{1i}, T_{2j}) = true **then**
    return true
  else if there exist some term T_{1i} in TS_1 and some T_{2j} in TS_2 such that isSimilar(T_{1i}, T_{2j}) = true **then**
    return true
  else
    return false
  end if
else if Q_1 and Q_2 are both empty **then**
  return true
else
  return false
end if

Algorithm 4 isSimilarByNgram(term1, term2, N)

str1Len $\leftarrow$ length(term1)
str2Len $\leftarrow$ length(term2)
minStrLen $\leftarrow$ min(str1Len, str2Len)
if minStrLen $<$ N **then**
  return false
end if
str1 $\leftarrow$ toLowerCase(term1)
str2 $\leftarrow$ toLowerCase(term2)
construct A1, the set of all N-grams (substrings with length N) of str1
construct A2, the set of all N-grams of str2
compute numOfSimilarNgrams, the number of common elements in both A1 and A2
set requiredNumOfSimilarNgrams to be f(minStrLen, N)
if numOfSimilarNgrams $\geq$ requiredNumOfSimilarNgrams **then**
  return true
else
  return false
end if

CHAPTER 6. GENERAL CONCLUSIONS

In this dissertation, we have described three methods for learning Bayesian network structures as well as our session identification and analysis for people search within a professional social network.

In Chapter 2, we develop a dynamic programming (DP) algorithm which uses the full Bayesian model averaging to compute the exact posterior probability of any subnetwork. Such an algorithm avoids the bias problem inherent in previous work (Koivisto and Sood, 2004; Koivisto, 2006) and is able to compute the exact posterior probability of a subnetwork in $O(3^n)$ time and the posterior probabilities for all $n(n-1)$ potential edges in $O(n3^n)$ total time. We demonstrate the applicability of the algorithm on several data sets with up to 20 variables.

In Chapter 3, we develop a DP algorithm for finding the k-best Bayesian network structures. Then we propose to compute the approximate posterior probability of any hypothesis of interest by Bayesian model averaging over these k-best Bayesian networks. We present empirical results on structural discovery over several real and synthetic data sets and show that our method outperforms the score-based method and the state-of-the-art MCMC methods.

In Chapter 4, we develop new algorithms for efficiently sampling Bayesian network structures (DAGs) of moderate size. The sampled DAGs can then be used to build estimators for the posteriors of any features. Our algorithms address the limitations of the exact algorithms previously proposed so that the posteriors of arbitrary non-modular features can be estimated when the size of the Bayesian network is moderate. We theoretically prove good properties of our estimators and empirically show that our estimators considerably outperform the ones from previous state-of-the-art methods.

In Chapter 5, we perform session analysis on the domain of people search within LinkedIn professional social network. We propose two effective and efficient refinements to the original content-based session identification method and show the excellent identification performance of the refined method in the domain. Using our refined content-based method as the gold-standard, we then evaluate the per-

formance of the time-based session identification method and find the optimal time-out threshold in the domain. Finally, we analyze several features of identified sessions to provide more insights into the nature of the domain.

BIBLIOGRAPHY

Aliferis, C., Tsamardinos, I., and Statnikov, A. (2003). HITON: A novel Markov blanket algorithm for optimal variable selection. In AMIA 2003 Annual Symposium Proceedings, pages 21–25.

Asuncion, A. and Newman, D. J. (2007). UCI machine learning repository.

Athreya, K. B. and Lahiri, S. N. (2006). Measure Theory and Probability Theory. Springer.

Bayir, M. A., Toroslu, I. H., Cosar, A., and Fidan, G. (2009). Smart miner: a new framework for mining large scale web usage data. In WWW '09: Proceedings of the 18th international conference on World Wide Web.

Bickel, P. and Doksum, K. (2001). Mathematical statistics: basic ideas and selected topics.

Binder, J., Koller, D., Russell, S., and Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables. Machine Learning, 29(2-3):213–244.

Brightwell, G. and Winkler, P. (1991). Counting linear extensions is #P-complete. In Proceedings of the twenty-third annual ACM symposium on Theory of computing, pages 175–181.

Casella, G. and Berger, R. L. (2002). Statistical Inference. Duxbury.

Castelo, R. and Kocka, T. (2003). On inclusion-driven learning of Bayesian networks. Journal of Machine Learning Research, 4:527–574.

Chickering, D. (2002a). Learning equivalence classes of Bayesian network structures. Journal of Machine Learning Research, 2:445–498.

Chickering, D. (2002b). Optimal structure identification with greedy search. Journal of Machine Learning Research, 3:507–554.

Chickering, D., Geiger, D., and Heckerman, D. (1994). Learning Bayesian networks is NP-hard. Technical report.

Conover, W. J. (1999). Practical Nonparametric Statistics. John Wiley & Sons, 3 edition.

Cooley, R., Mobasher, B., and Srivastava, J. (1999). Data preparation for mining world wide web browsing patterns. Knowledge and Information Systems, 1(1):5–32.

Cooley, R., Tan, P., and Srivastava, J. (2000). Discovery of interesting usage patterns from web data. In Advances in Web Usage Analysis and User Profiling, volume 1836, pages 163–182. Springer.

Cooper, G. F. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. Machine Learning, 9:309–347.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). Introduction to Algorithms (2nd Edition). The MIT Press.

Dash, D. and Cooper, G. F. (2004). Model averaging for prediction with discrete Bayesian networks. Journal of Machine Learning Research, 5:1177–1203.

Eaton, D. and Murphy, K. (2007). Bayesian structure learning using dynamic programming and MCMC. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pages 101–108.

Edwards, D. (2000). Introduction to Graphical Modelling. Springer, 2nd edition.

Ellis, B. and Wong, W. H. (2008). Learning causal Bayesian network structures from experimental data. Journal of the American Statistical Association, 103:778–789.

Frias-martinez, E. and Karamcheti, V. (2003). A customizable behavior model for temporal prediction of web user sequences. In WEBKDD 2002, volume 2703, pages 66–85.

Friedman, N., Goldszmidt, M., and Wyner, A. (1999). Data analysis with Bayesian networks: A bootstrap approach. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pages 196–205.

Friedman, N. and Koller, D. (2003). Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. Machine Learning, 50(1-2):95–125.

Grzegorczyk, M. and Husmeier, D. (2008). Improving the structure MCMC sampler for Bayesian networks by introducing a new edge reversal move. Machine Learning, 71:265–305.

Haynes, J. and Perisic, I. (2009). Mapping search relevance to social networks. In Proceedings of the 3rd Workshop on Social Network Mining and Analysis, SNA-KDD '09, pages 2:1–2:7. ACM.

He, D., Goker, A., and Harper, D. J. (2002). Combining evidence for automatic web session identification. In Information Processing and Management, pages 727–742.

Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning, 20:197–243.

Heckerman, D., Meek, C., and Cooper, G. (1999). A Bayesian approach to causal discovery. In Glymour, C. and Cooper, G., editors, Computation, Causation, and Discovery, pages 141–165.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association, 58(301):pp. 13–30.

Huang, X., Peng, F., An, A., and Schuurmans, D. (2004). Dynamic web log session identification with statistical language models. Journal of the American Society for Information Science and Technology, 55:1290–1303.

Jansen, B. J. and Spink, A. (2005). An analysis of web searching by European AlltheWeb.com users. Information Processing and Management, 41:361–381.

Jansen, B. J., Spink, A., Blakely, C., and Koshman, S. (2007). Defining a session on web search engines. Journal of the American Society for Information Science and Technology, 58(6):862–871.

Jansen, B. J., Spink, A., and Pedersen, J. (2005). A temporal comparison of AltaVista web searching. Journal of the American Society for Information Science and Technology, 56(6):559–570.

Jerrum, M. R., Valiant, L. G., and Vazirani, V. V. (1986). Random generation of combinatorial structures from a uniform distribution. Theoretical Computer Science, 43:169 – 188.

Kennes, R. and Smets, P. (1991). Computational aspects of the Möbius transformation. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pages 401–416.

Kleinberg, J. (2000). The small-world phenomenon: an algorithm perspective. In Proceedings of the thirty-second annual ACM symposium on Theory of computing, pages 163–170. ACM.

Koivisto, M. (2006). Advances in exact Bayesian structure discovery in Bayesian networks. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pages 241–248.

Koivisto, M. and Sood, K. (2004). Exact Bayesian structure discovery in Bayesian networks. Journal of Machine Learning Research, 5:549–573.

Koller, D. and Friedman, N. (2009). Probabilistic Graphical Models: Principles and Techniques. The MIT Press.

Lau, T. and Horvitz, E. (1999). Patterns of search: analyzing and modeling web query refinement. In Proceedings of the seventh international conference on user modeling.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady, 10(8):707–710. Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.

Madigan, D. and Raftery, A. (1994). Model selection and accounting for model uncertainty in graphical models using occam's window. Journal of Americal Statistical Association, 89:1535–1546.

Madigan, D. and York, J. (1995). Bayesian graphical models for discrete data. International Statistical Review, 63:215–232.

Manning, C. D. and Schuetze, H. (1999). Foundations of Statistical Natural Language Processing. The MIT Press, 1 edition.

Margaritis, D. and Thrun, S. (1999). Bayesian network induction via local neighborhoods. In Advances in Neural Information Processing Systems 12, pages 505–511. MIT Press.

Meila, M. and Jaakkola, T. (2006). Tractable Bayesian learning of tree belief networks. Statistics and Computing, 16:77–92.

Niinimaki, T. and Koivisto, M. (2013). Annealed importance sampling for structure learning in Bayesian networks. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pages 1579–1585.

Niinimaki, T., Parviainen, P., and Koivisto, M. (2011). Partial order MCMC for structure discovery in Bayesian networks. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pages 557–564.

Ozmutlu, H. C., Cavdur, F., and Ozmutlu, S. (2008). Cross-validation of neural network applications for automatic new topic identification. Journal of the American Society for Information Science and Technology, 59(3):339–362.

Ozmutlu, H. C., Cavdur, F., Ozmutlu, S., and Spink, A. (2004). Neural network applications for automatic new topic identification on excite web search engine data logs. Proceedings of the American Society for Information Science and Technology, 41(1):310–316.

Parviainen, P. and Koivisto, M. (2010). Bayesian structure discovery in Bayesian networks with less space. In Proceedings of AI and Statistics (AISTATS), pages 589–596.

Parviainen, P. and Koivisto, M. (2011). Ancestor relations in the presence of unobserved variables. In Proceedings of the European conference on machine learning and knowledge discovery in databases (ECML PKDD), pages 581–596.

Pearl, J. (1988). Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, San Mateo, CA.

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

Pellet, J.-P. and Elisseeff, A. (2008). Using Markov blankets for causal structure learning. Journal of Machine Learning Research, 9:1295–1342.

Silander, T. and Myllymaki, P. (2006). A simple approach for finding the globally optimal Bayesian network structure. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pages 445–452.

Silverstein, C., Henzinger, M., Marais, H., and Moricz, M. (1999). Analysis of a very large web search engine query log. SIGIR Forum, 33(1):6–12.

Singh, A. P. and Moore, A. W. (2005). Finding optimal Bayesian networks by dynamic programming. Technical report, Carnegie Mellon University, School of Computer Science.

Spiliopoulou, M. and Faulstich, L. (1999). Wum: A tool for web utilization analysis. In The World Wide Web and Databases, volume 1590, pages 184–203.

Spink, A. and Jansen, B. J. (2004). Web search: Public searching of the web.

Spirtes, P., Glymour, C., and Scheines, R. (2001). Causation, Prediction, and Search (2nd Edition). MIT Press, Cambridge, MA.

Sriram, S., Shen, X., and Zhai, C. (2004). A session-based search engine (poster). In Proceedings of SIGIR 2004.

Tian, J. and He, R. (2009). Computing posterior probabilities of structural features in Bayesian networks. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pages 538–547.

Tian, J., He, R., and Ram, L. (2010). Bayesian model averaging using the k-best Bayesian network structures. In Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI), pages 589–597.

Travers, J. and Milgram, S. (1969). An experimental study of the small world problem. Sociometry, 32:425–443.

Tsamardinos, I., Aliferis, C., and Statnikov, A. (2003). Time and sample efficient discovery of Markov blankets and direct causal relations. In Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (KDD), pages 673–678.

Tsamardinos, I., Brown, L., and Aliferis, C. (2006). The max-min hill-climbing Bayesian network structure learning algorithm. Machine Learning, 65:31–78.

Wagner, R. A. and Fischer, M. J. (1974). The string-to-string correction problem. Journal of ACM, 21(1):168–173.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. Journal of Information Retrieval, 1:69–90.