Approximation Algorithms for Probabilistic Reasoning: Sampling and Iterative Inference

Bozhena Bidyuk
School of Information and Computer Science
University Of California Irvine
Irvine, CA 92697-3425

bbidyuk@ics.uci.edu
http://www.ics.uci.edu/ bbidyuk

May 12, 2004

This report was presented as part of the Ph.D. candidacy of Bozhena Bidyuk

Abstract

The complexity of the exact inference increases exponentially with size and complexity of the network. As a result, the exact inference methods become impractical for large networks and we seek to approximate the results. A variety of approximation methods exist. This research focuses on two approximation methods for finding posterior marginals $P(x_i|e)$ in Bayesian networks: iterative belief updating (defined by Pearl [Pearl 1988]) and sampling.

The belief updating is an exact inference method for singly-connected networks. It can be applied to loopy networks to obtain approximate answers. The algorithm is based on message passing: in some order, each node computes and sends messages to its neighbors incorporating the latest messages it recieved. In a singly-connected network, we can order nodes so that it will be sufficient for each node to pass one messages in each direction. In a loopy network, the nodes compute several iterations of messages to achieve convergence (or to demonstrate the lack of convergence). Thus, belief updating in loopy networks is often referred to as *Iterative Belief Propagation* or IBP. Although IBP generally computes only approximate answers, it is known to perform extremely well in several special classes of networks such as coding networks and noisy-or networks. At the same time, we know that in some instances IBP does not converge or generates approximate answers far from correct. Currently, we do not have any methodology that would allow us in general case to predict the convergence of IBP or provide some practical error bounds on the approximate marginals it computes. In this research work, we examine the influence of the ϵ-*cutset* criteria on the convergence and quality of approximate marginals computed by IBP. We conjecture the ϵ-cutset (defined as a cycle-cutset with extreme posterior marginals) has effect similar to an observed cycle-cutset which breaks the loops and leaves the network singly-connected. We prove that the conjecture is true for Bayesian networks without evidence and show that the error in the approximate marginals computed by IBP converges to 0 as ϵ tends to 0. We provide empirical support for instances of Bayesian networks with evidence.

The idea behind the sampling methods for Bayesian networks is to generate a set of samples (where a sample in a vector space $X = \{X_1, ..., X_N\}$ is just an assignment of values to the elements of vector X) and then estimate the posterior marginals of interest from samples. In general, the quality of the approximate answers depends primarily on the number of samples generated and the approximate values converge to the exact values as number of samples increases. However, the sampling variance increases with the size of the sampling space. In this research work, we focus on the the variance reduction techniques on the example of the Gibbs sampler for Bayesian networks. It is obvious that we can achieve the reduction in variance by sampling only a subset of variables. However, the implication is that we have to carry out a lot more analytical computations which may render the whole approach impractical. We demonstrate that we can reduce sampling space efficiently if we take into consideration the underlying network structure. The time/space complexity of the exact inference in Bayesian networks is exponential in the induced width of the graph. In our sampling scheme, called w-*cutset sampling*, we sample a subset of variables (called a cutset) that is carefully chosen to reduce the complexity of the graph bounded by the induced width w. We analyze the problem of finding an optimal w-cutset of a graph (NP-hard in general case) and provide a heuristic algorithm for finding w-cutset in practice. We show empirically that w-cutset sampling typically finds better approximate answers than standard Gibbs sampler for a range of w values although its performance eventually deteriorates as w increases.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we define essential terminology and provide background information on Bayesian networks.

1.1 Notation

For clarity, we define here basic notation used throughout this document:

- use an upper case letter without subscripts, such as X or V, to denote a set of variables;
- use an upper case letter with a subscript, such as X_i, to denote a single variable;
- use "-i" subscript to denote the set of variables X minus variable X_i: $X_{-i} = X \backslash X_i$;
- use D_i or $D(X_i)$ depending to denote domain of the variable X_i;
- use a lower case letter without a subscript to denote an instantiation of a group of variables (e.g. x indicates that each variable in set X is assigned a value);
- use a lower case letter with a subscripts, to denote an instantiated variable (e.g. x_i denotes a value in the domain of X_i and means $X_i = x_i$);
- use a superscript in a subscripted lower case letter to denote a specific value (defined by superscript) in the domain of the corresponding variable: $D(X_i) = x_i^1, x_i^2, ...$

1.2 Graph Concepts

A *directed graph* is a pair $D = \{V, E\}$, where $V = \{X_1, ..., X_n\}$ is a set of nodes, or variables, and $E = \{(X_i, X_j) | X_i, X_j \in V\}$ is the set of edges. Given $(X_i, X_j) \in E$, X_i is called a *parent* of X_j, and X_j is called a *child* of X_i. The set of X_i's parents is denoted $pa(X_i)$, or pa_i, while the set of X_i's children is denoted $ch(X_i)$, or ch_i. The family of X_i includes X_i and its parents.

The *moral graph* of a directed graph D is the undirected graph obtained by connecting the parents of all the nodes in D and removing the arrows.

A *cycle-cutset* of an undirected graph a subset of nodes in the graph that, when removed, results in a graph without cycles.

The underlying graph G of a directed graph D is the undirected graph formed by ignoring the directions of the edges in D.

A node X in a directed graph D is called a *root* if no edges are directed into X. A node X in a directed graph D is called a *leaf* if all of its adjacent edges are directed into X.

A *cycle* in undirected graph G is a path whose two end-points coincide. A *cycle-cutset* of undirected graph G is a set of vertices that contains at least one node in each cycle in G.

A *loop* in a directed graph D is a subgraph of D whose underlying graph is a cycle. A vertex v is a *sink* with respect to loop $\mathcal{L}$ if the two edges adjacent to v in $\mathcal{L}$ are directed into v. A vertex that is not a sink with respect to a loop $\mathcal{L}$ is called an *allowed* vertex with respect to $\mathcal{L}$. A *cycle-cutset* (aka *loop-cutset*) of a directed graph G is a set of vertices that contains at least one allowed vertex with respect to each loop in D.

A directed graph is acyclic if it has no loops. A graph is *singly connected* (also called a *polytree*), if its underlying undirected graph has no cycles. Otherwise, it is called *multiply connected*.

1.3 Belief Networks

DEFINITION **1.3.1** *Let $X = \{X_1, ..., X_n\}$ be a set of random variables over multi-valued domains $D(X_1), ..., D(X_n)$. A* belief network (BN) *is a pair (G, P) where G is a directed acyclic graph on X and $P = \{P(X_i|pa_i)|i = 1, ..., n\}$ is the set of conditional probability tables (CPTs) associated with each*

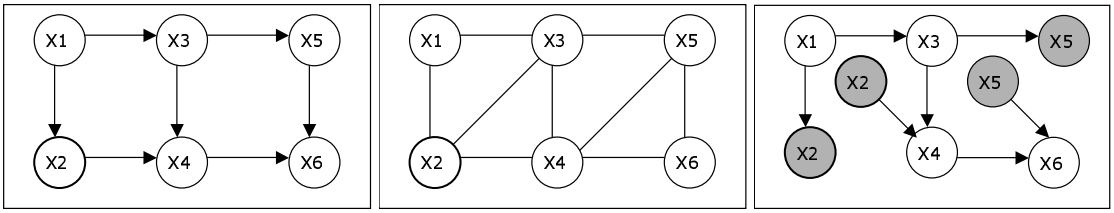


Figure 1.1: Bayesian network (left), its moral graph(center), and conditioned poly-tree (right) (conditioned on $C = \{X_2, X_5\}$).

X_i. An assignment ($X_1 = x_1, ..., X_n = x_n$) can be abbreviated as $x = \{x_1, ..., x_n\}$. The BN represents a joint probability distribution having the product form:

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i | x_{pa(X_i)})$$

An evidence e is an instantiated subset of variables E.

Bayesian networks facilitate a compact representation of the joint probability distribution over a set of variables X. Instead of enumerating all possible instantiations of X, we only enumerate the instantiations of parents of each node X_i in corresponding conditional probability table (CPT). Thus, the storage requirement is reduced from $d|X|$ to $|X| * d^{max_i|pa(X_i)|}$ where $d = max_i|D(X_i)|$ is the maximum domain size of a variable in $\mathcal{B}$.

The structure of the directed acyclic graph reflects the dependencies between the variables. The parents of a variable X_i together with its children and parents of its children form a *Markov blanket* $M(X_i)$ of node X_i: given the Markov blanket of a variable, its probability distribution is independent from the rest of the variables in the network: $P(x_i|x_{-i}) = P(x_i|M(X_i))$. A complete theory of Bayesian networks can be found in [Pearl 1988]

Figure 1.2 shows a belief network(left) and its moral graph(center). When a node X_i in a Bayesian network is observed (conditioned), we can transform the Bayesian network into an equivalent network as follows: remove all directed edges between the nodes and its children; for each child $ch_j(X_i)$, create a copy X_ij of node X_i add a directed edge from X_{ij} to $ch_j(X_i)$. If instantiated nodes in a Bayesian network form a cycle-cutset, then the Bayesian network can be transformed into an equivalent network that is a singly-connected. The graph on the right in Figure 1.2 shows a poly-tree network that is equivalent to the original network conditioned on cycle-cutset $C = \{X_2, X_5\}$.

The structure of a reasoning problem (including Bayesian networks) can be depicted via several graph representations.

DEFINITION **1.3.2 (Primal-, dual-,hyper-graph of a problem)** *The* primal *graph G=<X,E>of a reasoning problem <X,F> has the variables X as its nodes and an arc connects two nodes if they appear in the scope of the same function $f \in F$. A* dual *graph of a reasoning problem has the scopes of the functions as the nodes and an arc connects two nodes if the corresponding scopes share a variable. The arcs are labelled by the shared variables. The* hyper-graph *of a reasoning problem has the variables X as nodes and the scopes as edges. There is a one-to-one correspondence between the hyper-graph and the dual graphs of a problem.*

DEFINITION **1.3.3 (tree-width)** *The* tree-width *of a tree-decomposition $< T, \chi, \psi >$ is $\max_{v \in V}|\chi(v)|$ [Arnborg 1985]. Given two adjacent vertices u and v of a tree-decomposition, the* separator *of u and v is defined as $sep(u, v) = \chi(u) \cap \chi(v)$.*

DEFINITION **1.3.4 (induced-width)** *The* width of a node *in an ordered undirected graph is the number of the node's neighbors that precede it in the ordering. The* width of an ordering d*, denoted $w(d)$, is the width over all nodes. The* induced width *of an ordered graph, $w^*(d)$, is the width of the ordered graph obtained by processing the nodes from last to first. When node X is processed, all its preceding neighbors are connected. The resulting graph is called* induced graph *or* triangulated graph.

It is well known that the tree-width of a graph is identical to its induced-width. Also, the task of finding the tree-width of a graph is NP-complete [Arnborg 1985]. In the past 2 decades, substantial research focused on designing exact and approximate algorithms for finding the tree-width [Shoiket & Geiger 1997; Becker & Geiger 1996].

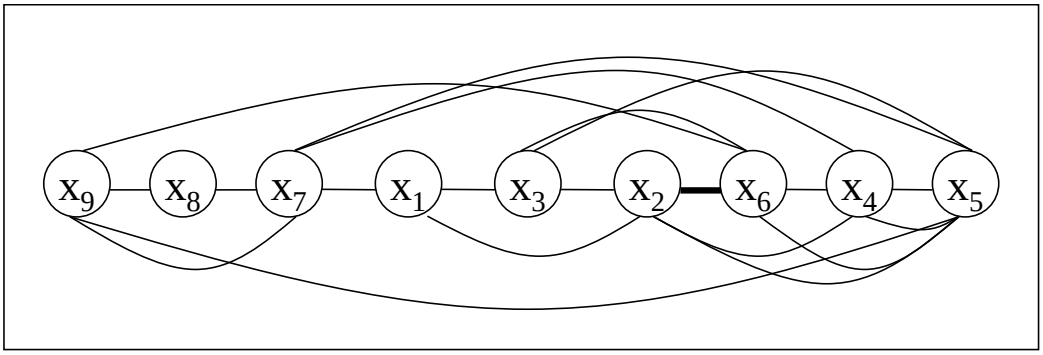


Figure 1.2: Induced graph with added edge $\{X_2, X_6\}$: d=$\{X_5, X_4, X_6, X_2, X_3, X_1, X_7, X_8, X_9\}$, w*=4.

For the sample network in Figure 1.2, along ordering $\{X_5, X_4, X_6, X_2, X_3, X_1, X_7, X_8, X_9\}$, the induced width $w^* = 4$ after we add edge $\{X_2, X_6\}$. The induced ordered graph is presented in Figure 1.2.

We next describe the notion of a tree-decomposition [Dechter & Pearl 1987; Lauritzen & Spiegelhalter 1988; Georg Gottlob & Scarello 1999]. It is applicable to any graphical model since it is defi ned relative to its hyper-graph or dual graph.

DEFINITION **1.3.5 (tree-decomp., cluster-tree, tree-width)** *Let $R =< X, D, F >$ be a reasoning problem with its hyper-graph $\mathcal{H} = (X, F)$). (We abuse notation when we identify a function with its scope). A* tree-decomposition *for R (resp., its hyper-graph $\mathcal{H}$) is a triple $< T, \chi, \psi >$, where T=$<V, E>$ is a tree, and χ and ψ are labelling functions which associate with each vertex $v \in V$ two sets, $\chi(v) \subseteq X$ and $\psi(v) \subseteq F$ such that*

1. *For each function $f_i \in F$, there is* exactly *one vertex $v \in V$ such that $f_i \in \psi(v)$, and $scope(f_i) \subseteq \chi(v)$.*
2. *For each variable $X_i \in X$, the set $\{v \in V | X_i \in \chi(v)\}$ induces a connected subtree of T. This is also called the running intersection property.*

We will often refer to a node and its functions as a cluster *and use the term* tree-decomposition *and* cluster tree *interchangeably. The maximum size of node χ_i minus 1 is the width of the tree decomposition. The* tree width *of a graph G denoted $tw(G)$ is the minimum width over all possible tree decompositions of G $[Arnborg1985]$.We sometime denote the optimal tree-width of a graph by $tw*$. We use the notation of tree-width and induced-width interchangeably.*

There is a known relationship between tree-decompositions and chordal graphs. A graph is *chordal* if any cycle of length 4 or more has a chord. Every tree-decomposition of a graph corresponds to a chordal graph that augments the input graph where the tree clusters are the cliques in the chordal graph. And vice-versa, any chordal graph is a tree-decomposition where the clusters are the maximal cliques. Therefore, much of the discussion that follows, relating to tree-decompositions of graphs, can be rephrased using the notion of chordal graph embeddings.

1.4 Inference in Bayesian networks

Given a Bayesian network $\mathcal{B}$ over a set of variables X and observations e ($E \subset X$), the *inference* task over $\mathcal{B}$ is to infer the values of unobserved variables. The typical inference tasks are:

1. *Belief Updating*: given evidence e and a query variable $X_i \in X$, compute a posterior probability distribution $P(x_i|e)$;
2. *Belief Revision (MPE)*: fi nd **m**ost **p**robable **e**xplanation for evidence e, namely, fi nd a maximum probability assignment to the unobserved variables $Y = X \backslash E$:

$$y \leftarrow \arg\max_y P(y|e)$$

3. *MAP*: fi nd **m**aximum **a p***osteriori* hypothesis, namely fi nd a maximum probability assignment to a subset of unobserved variables $y \in x \backslash e$ given evidence e:

$$y \leftarrow \arg\max_y P(y|e)$$

The exact inference in Bayesian networks is NP-hard ([Cooper 1990; Shimony 1994]). Furthermore, finding an approximate solution for any of the tasks above with a fixed error bound is also NP-hard ([Dagum & Luby 1993; Abdelbar & Hedetniemi 1998]). More recent results have demonstrated that between the tree tasks, belief updating is harder than belief revision, and MAP is harder than belief updating:

$$MPE < BeliefUpdating < MAP$$

To be exact, MPE remains NP-complete, Belief Updating was shown to be $\#P$-complete ([Roth 1996]) as the task is similar to that of counting the number of solutions, and MAP was shown to be NP^{PP}-complete [Park 2002].

MPE and belief updating are can be solved in time linear in the size of the network when Bayesian network is singly-connected (has no loops) using belief propagation algorithms proposed by Pearl ([Pearl 1988]). Park ([Park 2002; Park & Darwiche 2003]) showed that MAP task remains hard even when MPE and belief updating become easy (for example, in poly-tree networks).

This research work is focused on answering the Belief Updating queries, namely finding the posterior marginals of the form $P(x_i|e)$.

1.5 Classes of the Bayesian Networks

Here, we will define a few special classes of Bayesian networks as we will refer to them multiple times in our empirical studies.

1.5.1 Noisy-Or Networks

The Noisy-Or model was introduced by Judea Pearl [Pearl 1988]. All nodes in a simple Noisy-Or relationship are assumed bi-valued. A Noisy-Or node in a Bayesian network is a generalization of a logical *or*. The parents of node X_i model potential causes of the event X_i. The Noisy-Or relationship is embedded in the CPT of node X_i. As in the case of logical *or*, the probability of event X_i is 0 if all conditions that cause event are false:

$$P(X_i = TRUE|\forall j, pa_j(x_i) = FALSE) = 0$$

However, unlike logical *or*, we assume that each cause $pa_j(X_i)$ of X_i has associated inhibitory influence which is active with probability q_i. Thus, given the parents with TRUTH assignment, the probability of X_i being FALSE equals the probability that all of those parents were inhibited. Since all inhibitors are presumed independent, their joint probability equals the product of individual probabilities:

$$P(X_i = FALSE|pa(x_i)) = \prod_{j, pa_j = TRUE} q_i$$

As a result:

$$P(X_i = TRUE|pa(x_i)) = 1 - \prod_{j, pa_j = TRUE} q_i$$

We can further extend the Noisy-Or relationship by introducing the leak probability θ of the even X_i being TRUE even if all of its causes (parents) evaluate to FALSE:

$$P(X_i = TRUE|\forall j, pa_j(X_i) = FALSE) = \theta$$

1.5.2 Coding Networks

Coding networks are associated with probabilistic decoding problems. A coding network contains 4 classes of nodes:
- code bits $X = \{X_1, ..., X_n\}$;
- parity check bits $Y = \{Y_1, ..., Y_m\}$;
- transmitted code bits $T = \{T_1, ..., T_n\}$;
- transmitted parity check bits $Z = \{Z_1, ..., Z_m\}$.

All nodes have domains of size 2 of form $D = \{0, 1\}$. The code bits are root nodes (without parents) with uniform priors: $P(x_i) = 0.5$ Each transmitted code bit T_i has one parent node, the code bit X_i. Each transmitted parity check bit Z_i has one parent node, the parity check bit Y_i. Sample coding network is shown in Figure 1.5.2. The conditional probability table of parity check nodes is deterministic and equals the parity check function:

$$P(Y_i|X_{j_1}, ..., X_{j_i}) = X_{j_1} \oplus ... \oplus X_{j_i}$$

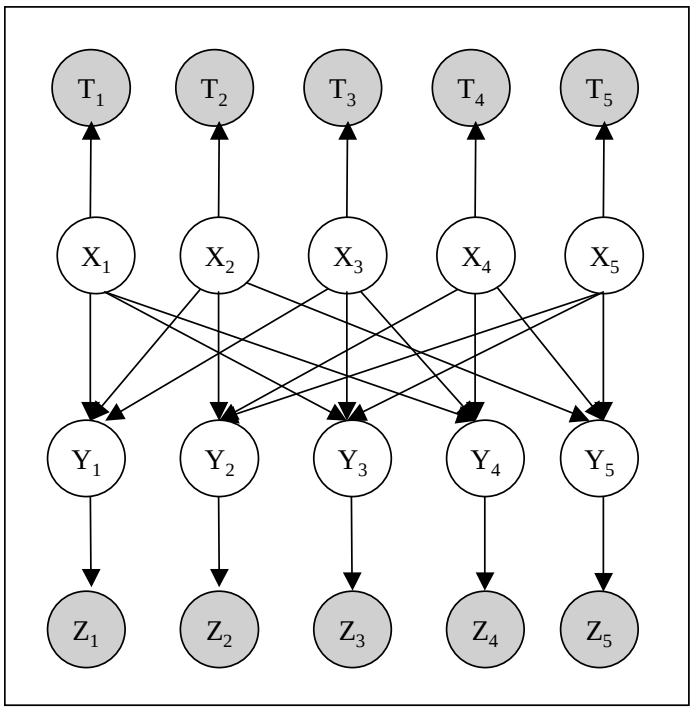


Figure 1.3: A sample coding network with code bits $X = \{X_1, ..., X_5\}$, parity check bits $Y = \{Y_1, ..., Y_5\}$, transmitted code bits $T = \{T_1, ..., T_5\}$, and transmitted parity check bits $Z = \{Z_1, ..., Z_5\}$.

The conditional probabilities of the transmitted bits are parametrized by the noise parameter σ. The distribution $P(t_i|x_i)$ is Gaussian with mean $\mu = x_i$ and noise σ. Similarly, the distribution $P(z_i|y_i)$ is Gaussian with mean $\mu = y_i$ and noise σ. The decoding task can be defined as either finding the most probable assignment to the code bits and parity check bits (MPE task) given the transmitted bits (evidence $E = \{t_1, ..., t_n, z_1, ..., z_m\}$) or belief updating task of computing posterior marginals $P(x_i|e)$ of each code bit node and assigning each code bit the most probable value. The rigorous probabilistic decoding study is presented in [I. Rish & Dechter 1998].

Chapter 2

Cutset sampling for Bayesian networks

In this chapter, we discuss the sampling-based approximation methods for Bayesian networks with the focus on Gibbs sampling (a instance of Markov Chain Monte Carlo sampling). The key principles behind all sampling methods for Bayesian networks are that first, we generate a set of samples and then estimate posterior marginals $P(x_i|e)$ from samples. However, the sampling variance can increase dramatically with the size of the sampling space. We can reduce variance by sampling only a subset of variables and carrying out exact computation as much as possible. The caveat is that exact computation is usually expensive and increases sample computation time so that within the same time period we can produce only a fraction of samples of smaller size compared to sampling all variables rendering this variance reduction scheme impractical in many applications. In Bayesian networks, we can control the cost of exact inference via the induced width w of the graph. We introduce the concept of a w-cutset of a graph as a subset of nodes such that, when observed, the induced width of the graph is w. If sampling nodes form a w-cutset of a graph, then the exact inference time is bounded by $O(N \cdot d^w)$ where d is maximum domain size and, consequently, sampling time is bounded by $O(M \cdot d \cdot N \cdot d^w)$ where M is the size of the w-cutset. We demonstrate the efficiency of this scheme empirically for several classes of benchmarks.

2.1 Motivation for Cutset Sampling

Sampling methods for Bayesian networks are commonly used approximation techniques, applied successfully where exact inference is not possible due to prohibitive time and memory demands. In this paper, we focus on Gibbs sampling, a member of the Markov Chain Monte Carlo sampling methods group for Bayesian networks [Geman & Geman 1984; Gilks, Richardson, & Spiegelhalter 1996; MacKay 1996]. Given a Bayesian network over the variables $X = \{X_1, ..., X_n\}$, and evidence e, Gibbs sampling [Geman & Geman 1984; Gilks, Richardson, & Spiegelhalter 1996; MacKay 1996] generates a set of samples $\{x^t\}$ from $P(X|e)$ where each sample $x^t = \{x_1^t, ..., x_n^t\}$ is an instantiation of all the variables in the network. It is well-known that a function $f(X)$ can be estimated using the generated samples by:

$$E[f(X)|e] \cong \frac{1}{T} \sum_t f(x^t) \tag{2.1}$$

where T is the number of samples. Namely, given enough samples, the estimate converges to the exact value. The central query of interest over Bayesian networks is computing the posterior marginals $P(x_i|e)$ for each value x_i of variable X_i. A significant limitation of all existing sampling schemes, including Gibbs sampler, is the increase in the statistical variance for high-dimensional spaces. In addition, standard sampling methods fail to converge to the target distribution when the network is not ergodic.

There exist a number of variance reduction methods that are well-known from statistics. The most efficient approach is to reduce the sampling space and carry out analytical computation whenever possible. The difficulty lies in performing analytical computation efficiently, without slowing down the sampling algorithm too much. In this chapter, we demonstrate that we can efficiently sample only a subset of the variables (a cutset) and apply exact inference for the rest of the variables.

Alternatively, we find motivation for sampling a subset of variables when dealing with the state-space explosion associated with the cycle-cutset conditioning method. We know that when cycle-cutset nodes of a graph are instantiated, the graph can be transformed into an equivalent singly-connected graph where exact inference is easy. This fact provides basis for the cycle-cutset conditioning method of Judea Pearl ([Pearl 1988]) where we compute exact posterior marginals $P(x_i|c, e)$ over all possible instantiations of the cycle-

cutset nodes C and then sum:

$$P(x_i|e) = \alpha \sum_c P(x_i|c,e)P(c,e)$$

where α is a normalization constant. Clearly, the time complexity of cycle-cutset conditioning grows exponentially with the size of the cycle-cutset. To avoid state-space explosion, it was proposed ([Horvitz, Suermondt, & Cooper 1989]) that we can obtain good bounds on posterior marginals processing only a subset of all possible tuples if we can pick the tuples with high probability mass $P(c,e)$. Of course, the challenge is to identify high probability mass tuples without actually computing their probabilities. It appears that sampling from $P(c,e)$ offers a natural solution to the problem as samples tends to concentrate in the areas of importance. The sampling-based estimate of the posterior marginals is similar to summing over a part of the cycle-cutset space $C_s \subset D(C)$ that was explored while sampling:

$$P(x_i|e) = \sum_{c \in C_s} P(x_i|c,e)\hat{P}(c,e)$$

where $\hat{P}(c,e)$ is a sampling estimator for $P(c,e)$. We will discuss this notion further in section 2.6.

In sections 2.2 and 2.3, we provide background information on the sampling methods for Bayesian networks and describe the standard Gibbs sampler. In section 2.4, we discuss the variance reduction techniques. In section 2.6, we describe our cutset sampling approach followed by empirical evaluation in section 2.9.

2.2 Monter Carlo Sampling

Given that task of evaluating the expectation E(f(x)), the basic idea behind sampling techniques is to draw independent and identically distributed (i.i.d.) samples $x^{(1)}$, $x^{(2)}$, ..., $x^{(m)}$ and then evaluate as follows:

$$\hat{f_m} = E(f(x)) = \frac{1}{m}\sum_t f(x^{(t)}) \tag{2.2}$$

Following the *law of large numbers*, it can be shown that:

$$\lim_{m \to infty} \hat{f_m} = f$$

The convergence rate is derived from central limit theorem (CLT):

$$\sqrt{m}(\hat{f_m} - f) \to N(0, \sigma^2)$$

where $\sigma^2 = varf(x)$. As a result, the "error term" in the approximate result evaluates to $O(m^{-1/2})$ and does not depend on the dimensionality of x. This makes the sampling methods attractive to solve large problems, including complex Bayesian networks. The limiting factors are:

- as dimensionality of X increases, the variance σ^2, which measures how uniform is the function f(x) over X , can increase dramatically;
- we maybe unable to produce independent uniform samples;
- we do not know the sampling distribution $\pi(x)$.

We will address the variance reduction techniques in section 2.4. Here, we will summarize the basic sampling schemes for Bayesian networks.

The target sampling distribution $\pi(x)$ in a Bayesian network is the posterior joint distribution $P(x|e)$ and it is typically also the unknown we want to evaluate. in absence of evidence, we actually can sample from target distribution $P(x)$ using approach known as Logic sampling ([Henrion 1988]): sample node values in topological order; first, sample the root node values are from their priors; sample other node values X_i in topological order from $P(X_i|pa(X_i))$. Since the parents of node X_i are sampled first, their values are fixed by the time we need to sample node X_i. Thus, we only need to look up $P(X_i|pa(X_i))$ in the CPT of node X_i. The samples produced are independent and posterior marginals estimation is straight-forward.

When evidence is present, we can apply Logic sampling ignoring evidence, but we have to discard (reject) samples where values assigned to evidence node conflict with actual evidence values resulting in the Rejection sampling ([Gilks & Wild 1992]). In this scenario though, many samples are wasted if the probability of evidence is small: $P(e) \to 0$. Importance sampling ([Shachter & Peot 1989]) overcomes this problem; it allows to draw samples from trial distribution g(x), different from target distribution, and then weighing each sample against the target distribution $\pi(x)$:

$$E[f(x)] = \frac{1}{m}\sum_{t=1}^{m} f(x^t)\frac{\pi(x^t)}{g(x^t)}$$

If we define $w^t = \frac{\pi(x^t)}{g(x^t)}$, then we obtain:

$$E[f(x)] = \frac{1}{m}\sum_{t=1}^{m} f(x^t)w^t \tag{2.3}$$

The equation 2.3 defines an unbiased estimator for f(x). In many instances, a biased estimator maybe preferred:

$$E[f(x)] = \frac{\sum_{t=1}^{m} f(x^t)w^t}{\sum_{t=1}^{m} w^t} \tag{2.4}$$

The biased estimator maybe preferred because it allows us to compute the weights w^t only up to normalization constant. Additionally, although inducing a small bias, 2.4 often has a smaller mean squared error than the unbiased one ([Liu 2001]). In Bayesian networks, an easy to sample from trial distribution is $P(X)$, same as the distribution used in Logic sampling and Rejection sampling. This special case of importance sampling is known as Likelihood weighting ([Fung & Chang 1989; Shachter & Peot 1989]) However, an importance sampling performs much better when $g(x)$ and $\pi(x)$ are close. Thus, in presence of low-probability evidence, $P(X)$ is a bad trial distribution. Variations of importance sampling were proposed that update the trial distribution during sampling process to bring it closer to $\pi(x)$ ([Cheng & Druzdzel 2000; Yuan & j. Druzdzel 2003]). Importance sampling analysis and its applications can be found in [A. Doucet 2001; Liu 2001].

An alternative approach is to generate dependent samples based on the idea of Markov Chain Monte Carlo sampling. In this case, each sample represents the state of the system and generating a sample s^{t+1} after sample s^t is equivalent to system state transition. The key is to define state transition function such that the stationary distribution of the Markov Chain is the target distribution $\pi(x)$. The Markov Chain-based Monte Carlo sampling was first proposed by [Metropolis *et al.* 1953] and is known as Metropolis sampling. The original Metropolis sampling algorithm required a symmetric transition function. Hastings generalized the algorithm ([Hastings 1970]) so that the only critical requirement on the transition function is that the function remains positive. The generalized algorithm is known as Metropolis-Hastings sampling.

Gibbs sampling is a special MCMC scheme introduced by Geman and Geman ([Geman & Geman 1984]). Metropolis and Metropolis-Hastings schemes do not specify how to choose state transition function. Gibbs sampler specifically uses conditional distribution $P(x_i|x_{-i})$ to define state transition rules. We will describe Gibbs sampler in detail in the next section.

The power of MCMC sampling is that it allows to sample from almost any target distribution $\pi(x)$. A Markov chain that is *aperiodic* and *irreducible* and has an invariant distribution $\pi(x)$ is guaranteed to become stationary at its invariant distribution. The convergence is guaranteed regardless of the initial state.

DEFINITION **2.2.1 (Aperiodic)** *A Markov chain is said to be* aperiodic *if the maximum common divider of the number of steps it takes for the chain to come back to the starting point (any) is equal to one. [[Liu 2001]]*

DEFINITION **2.2.2 (Irreducible)** *Markov chain is* irreducible *if the chain has nonzero probability (density) to move from one position in the state space to any other position in a finite number of steps. [[Liu 2001]]*

DEFINITION **2.2.3 (Ergodic)** *An aperiodic, irreducible Markov chain is called* ergodic.

The aperiodicity means that the chain does not have regular loops where every k_i steps we return to state S_i. The irreducibility guarantees that we can get to any state S_j from any state S_i with non-zero probability and thus, will be able to visit (as number of samples $T \rightarrow infty$) all statistically important regions of state space. The conditions are almost always satisfied for the Metropolis algorithm ([Tierney 1994]).

Since Markov chain is only guaranteed to converge to the stationary distribution after some K samples (the "burn in" period), the first K samples are often discarded. While there is no exact theory on how the burn-in period must be, it is typical to choose $K = 1000$.

The main drawback of MCMC sampling is that samples, dependent by definition, can be highly correlated. As a result, the estimates obtained from those samples tend to have higher variance then those obtained from independent samples ([Liu 2001]). Thus, a lot of MCMC sampling research is focused on either generating less correlated samples or reducing variance directly. The latter is the focus of this chapter.

2.3 Gibbs sampling

In this section, we will define Gibbs sampling in the context of Bayesian networks. Gibbs sampling generates samples from $\hat{P}(X|e)$ which converges to $P(X|e)$ as the number of samples increases [Pearl 1988; MacKay 1996] as long as the network is ergodic. Given a Bayesian network $\mathcal{B}$ with variables $X = \{X_1, X_2, ..., X_n\}$, a sample x^t is an instantiation of variables in X:

$$x^{(t)} = \{x_1^{(t)}, x_2^{(t)}, ..., x_n^{(t)}\}$$

where t denotes a sample number and x_i^t is the value of X_i in sample t. Gibbs sampling generates a set of samples x^t where t denotes a sample and $x_i^{(t)}$ is the value of X_i in sample t. The first sample can be initialized at random. The values of evidence nodes remain fixed. The next sample $x_i^{(t+1)}$ is generated from previous sample $x_i^{(t)}$ following one of the two schemes:

Random Scan Gibbs Sampler. Given a sample $x_i^{(t)}$ at iteration t, pick a variable X_i at random and samples a new value from conditional distribution leaving other variables unchanged:

$$x_i \leftarrow P(x_i|x_{-i}^{(t)})$$

Systematic Scan Gibbs Sampler. Given a sample x^t, we sample the value of each variable x_i in some order :

$$\begin{aligned}
x_1 &\leftarrow P(x_1|x_2^{(t)}, x_3^{(t)}, ..., x_n^{(t)}) \\
x_2 &\leftarrow P(x_2|x_1^{(t+1)}, x_3^{(t)}, ..., x_n^{(t)}) \\
&... \\
x_i &\leftarrow P(x_i|x_1^{(t+1)}, ..., x_{i-1}^{(t+1)}, x_{i+1}^{(t)}, ..., x_n^{(t)}) \\
&... \\
x_n &\leftarrow P(x_n|x_1^{(t+1)}, x_2^{(t+1)}, ..., x_{n-1}^{(t+1)})
\end{aligned}$$

Systematic Scan Gibbs sampler is also referred to as an Ordered Gibbs sampler. The algorithm is formally defined in Figure 2.3. It was shown in [Roberts & Sahu 1997] that random scan Gibbs sampler can outperform the systematic scan Gibbs sampler. However, ultimately, the convergence rate of Gibbs sampler strongly depends on the correlation between two consecutive samples whether the sampling variables are ordered or picked at random. We discuss this subject further in the section 2.4.

In Bayesian networks, the conditional probability distribution $P(x_i|x_1^{(t+1)}, ..., x_{i-1}^{(t+1)}, x_{i+1}^{(t)}, ..., x_n^{(t)})$ is dependent only on the assignment to the Markov blanket of variable X_i. Thus, we can rewrite the sampling of value x_i more concisely as follows:

$$x_i \leftarrow P(x_i|markov^{(t)}(x_i))$$

Ordered Gibbs Sampler
Input: A belief network ($\mathcal{B}$), variables $X = (X_1, ..., X_n)$, and evidence tuple $e = \{(X_i = e_i)|X_i \in E, E \subseteq X\}$.
Output: A set of samples $\{x^t\}$,
1. **Initialize:** Assign random value x_i^0 to each variable $X_i \in X \backslash E$ from $D(X_i)$. This is initial assignment x^0. Assign evidence variables their observed values.
2. **Generate samples:**
For t = 1 to T (generate a new sample x^t):
For i = 1 to n, (compute a new value x_i^t for variable X_i):
Compute $P(x_i|markov^t(x_i))$ and sample:

$$x_i^t \leftarrow P(x_i|markov^t(x_i))$$

Set $X_i = x_i^t$.
End (for i)
Add a new sample to list of samples.
End (for t)

Figure 2.1: *A Gibbs sampling* Algorithm

Once samples $\{x^{(1)}, ..., x^{(T)}\}$ are generated, we can estimate posterior marginal beliefs $P(x_i|e)$ for each variable X_i. Two estimators are available. One is called *histogram estimator*:

$$\hat{P}(x_i|e) = \frac{1}{T}\sum_{t=1}^{T} P(X_i = x_i|x^{(t)}) \tag{2.5}$$

Since $P(X_i = x_i|x^{(t)}) = 1$ if $x_i^{(t)} = x_i$ and $P(X_i = x_i|x^t) = 0$ otherwise, the histogram estimator is effectively counting the samples where $X_i = x_i$ and can re-written as follows:

$$\hat{P}(x_i|e) = E[P(x_i|e)] = \frac{1}{T}\sum_{t=1}^{T}\delta_{x_i}(x^{(t)}) \tag{2.6}$$

where $\delta_{x_i}(x^t) = 1$ if $x_i^t = x_i$ and equals 0 otherwise. Alternatively, a *mixture estimator* can be used:

$$\hat{P}(x_i|e) = \frac{1}{T}\sum_{t=1}^{T}E[P(x_i)|x_{-i}^{(t)}] \tag{2.7}$$

Gelfand and Smith have pointed out (in [Gelfand & Smith 1990]) that mixture estimator should be preferred because it guarantees lower variance in the estimator. The mixture estimator is also called the "Rao-Blackwellised" estimator because it is based on estimating conditional expectation. Rao-Blackwellisation always improves the effi ciency of Monte Carlo estimates. The variance reduction in the conditional expectation was fi rst proven in Rao-Blackwell theorem ([Groot 1986]). In Bayesian networks, the mixture estimator is simply an average of conditional probability distributions:

$$\hat{P}(x_i|e) = \frac{1}{T}\sum_{t=1}^{T}P(x_i|x_{-1}^{(t)})) \tag{2.8}$$

Again, the probability distribution $P(x_i|x_{-1}^{(t)}))$ only depends on the Markov blanket of node X_i yielding the fi nal form of mixture estimator:

$$\hat{P}(x_i|e) = \frac{1}{T}\sum_{t=1}^{T}P(x_i|markov^{(t)}(x_i)) \tag{2.9}$$

Given a Markov blanket of X_i, the closed form for the sampling probability distribution is given explicitly by ([Pearl 1988]) :

$$P(x_i|markov^t(x_i)) = \alpha P(x_i|x_{pa(X_i)}^t)\prod_{\{j|X_j\in ch_j\}}P(x_j^t|x_{pa_j}^t) \tag{2.10}$$

Thus, generating a complete new sample requires $O(n \cdot r)$ multiplication steps where r is the maximum family size and n is the number of variables. Subsequently, computing the posterior marginals is linear in the number of samples.

2.4 Variance Reduction Schemes

The convergence of the Gibbs sampling strongly depends on the degree of correlation between samples ([Liu 1991; Schervish & Carlin 1992; J. S. Liu & Kong 1995]). A few techniques were proposed to reduce dependence between samples such as randomly selecting a sampling variable in Random Scan Gibbs sampler or subsampling the Gibbs sampler by including only every k^{th} sample into the estimate (see analysis in [Geyer 1992; MacEachern & Berliner 1994]. However, the subsampling of Gibbs sampler actually increases the statistical variance of the estimate. Furthermore, neither random variable selection nor subsampling address the main problem that the convergence rate is controlled by the *maximal correlation* between the states of two consecutive Gibbs iterations (see [Liu 2001], chapter 12, for details). In other words, the strong between two variables in a sample will drive the convergence rate of the Gibbs sampler.

The two main approaches that directly reduce variance are *blocking* (grouping variables together and sampling simultaneously) and *collapsing* (integrating out some of the random variables and sampling a subset). Given Bayesian network with three random variables: X, Y, and Z, we can schematically describe those three sampling schemes as follows:

1. Standard Gibbs: samples values from $P(x|y,z)$, $P(y|x,z)$, $P(z|x,y)$
2. Collapsing: sample X from $P(x|y)$, sample Y from $P(y|x)$ integrating out random variable X.
3. Blocking Gibbs: samples X from $P(x|y,z)$ and (Y,Z) from $P(y,z|x)$

As shown in [J. Liu & Kong 1994], the blocking Gibbs sampling scheme, where several variables are grouped together and sampled simultaneously, is expected to converge faster than standard Gibbs sampler. Variations to this scheme have been investigated in [Jensen, Kong, & Kjaerulff 1995; Kjærulff 1995]. The

collapsed Gibbs sampler ([Liu 1994])actually allows to integrate some of the random variables out, thus reducing sampling space, and it is expected to converge the fastest [J. Liu & Kong 1994]. Thus, of the two basic data augmentation scheme, namely collapsing and blocking, collapsing is generally preferred. The analysis of the collapsed Gibbs sampler can be found in [Escobar 1994; MacEachern 1994; Liu 1996].

The caveat in the utilization of the collapsed Gibbs sampler is that computation of the probabilities $P(x|y)$ and $P(y|x)$ must be efficient. In case of Bayesian networks, the task of integrating variables out equates to performing exact inference on the network where evidence nodes and sampling nodes are observed and its time complexity is exponential in the induced width of the network. Taken *a priori* that performance of the sampler will be severely impacted when many variables are integrated out, collapsing has been applied only in a few special cases of Bayesian networks ([Doucet *et al.* 2000; Liu 1994]). In this research work, we demonstrate that indeed we can reduce sample size and still maintain efficiency of the Gibbs sampler if we choose sampling variables carefully, taking into account the underlying graph structure.

2.5 Related Work

The idea of grouping of the variables to improve efficiency of Gibbs sampler was first proposed in [J. Liu & Kong 1994]. Grouping strongly correlated variables together can significantly improve convergence rates of the Gibbs sampler and reduce the sampling variance in the estimates. In [Jensen, Kong, & Kjaerulff 1995], groups of variables were sampled simultaneously (using exact inference to compute conditional probability distribution) conditional on the sufficiently large number of variables to reduce the complexity of the exact inference problem. The empirical results in [Jensen, Kong, & Kjaerulff 1995] demonstrate a significant improvement in the convergence of the Gibbs sampler over time. The major differences between our cutset sampling approach (define in the next section) and one proposed in [Jensen, Kong, & Kjaerulff 1995] are that, first, in the proposed blocking Gibbs sampling, a sample consists of all the variables in the network (as usual) while cutset sampling never assigns values to those variables that are integrated out; second, in [Jensen, Kong, & Kjaerulff 1995], exact inference is used to perform joint sampling step for a group of variables while cutset sampling uses exact inference to integrate variables out.

A different combination of sampling and exact inference for join-trees was described in [Koller, Lerner, & Angelov 1998] and [Kjærulff 1995]. Both papers proposed to use sampling to estimate the probability distribution in each cluster from which they compute messages sent to the neighboring clusters. In [Kjærulff 1995], the Gibbs sampling was used only large clusters to estimate the joint probability distribution $P(V_i), V_i \subset X$ of variables in cluster i. The estimation of the $P(V_i)$ is recorded instead of the true joint probability distribution to conserve memory: the motivation is that the low probability tuples will not be generated during sampling and can be assumed to have probability 0 while only the joint probabilities of tuples with high-probability mass will be generated an recorded. In small clusters, the exact joint distribution $P(V_i)$ is computed and recorded. However, the paper does not analyze the introduced errors or compare the performance of this scheme with standard Gibbs sampler or the exact algorithm.

In [Koller, Lerner, & Angelov 1998], the sampling is used to represent the intermediate local probability distributions used in computing messages (sent from cluster i to cluster j) and the posterior joint distributions in a cluster-tree of a hybrid network that contains both discrete and continous variables. This approach subsumes the sampling scheme described in [Kjærulff 1995] and includes rigorous considerations with respect to minimizing the error in the estimated posterior distributions. However, the method has obvious difficulties with propagation of evidence. The empirical evaluation (limited to two hybrid network instances) only compares the quality of the estimates to those of likelihood weighting, an instance of importance sampling that does not perform well in presence of low-probability evidence. Thus, it is not clear how much we gain (if we gain anything at all) from implementing proposed scheme compared to discretizing continous variable values or using Gibbs sampling.

Sampling of a subset variable In particular, it has been applied to the Particle Filtering (using importance sampling) method for Dynamic Bayesian networks [Doucet *et al.* 2000] in cases where some of the variables can be integrated out easily either because they are conditionally independent given the sampled variables (plus evidence) or because their probability distribution permits tractable exact inference (for example, using Kalman filter).

Previously, sampling from a subset of variables was successfully applied to particle sampling for Dynamic Bayesian networks (DBNs) [Doucet *et al.* 2000]. Indeed, the authors demonstrated that sampling from a subspace combined with exact inference yields a better approximation. Yet, the application of described approach is limited to summing out those nodes that are independent from the values assigned in a previous time slice conditional on the assignment to the sampling variables. Our cutset sampling scheme, defined in section 2.6 offers a generic approach to collapsing a Gibbs sampler in any Bayesian network.

2.6 Cutset sampling

This section presents the cutset sampling method. As noted earlier, the basic scheme partitions variables X into two subsets C and $X \backslash C$. If we can effi ciently compute sampling distribution $P(c_i|c_{-1})$ for all $C_i \in C$ as well as conditional distribution $P(z_i|c^t, e)$ for all $X_i \in X \backslash C$, then we can effi ciently sample the subset of variable C and sum out the rest of the variables.

Let us assume for a moment that we can compute all necessary probability distributions effi ciently. The benefi ts over plan Gibbs sampler are two-fold. First, we achieve reduction in the sampling variance due to collapsing the sampling space. Second, we maybe able to extend Gibbs sampler to the non-ergodic networks that have an ergodic subspace. Namely, although standard Gibbs sampler will not converge if some of the transition probabilities $P(x_i|x_{-1})$ are deterministic, we maybe able to select a subset of variables $C \subset X$ such that all transition probabilities are positive: $\forall C_i \in C, P(c_i|c_{-i}) > 0)$ and thus, guarantee convergence. We demonstrate this on the example of the coding networks in the empirical section 2.9.

Given a set of samples over a subset of variables C, we can estimate posterior marginals of any variable in the network using mixture estimator defi ned in section 2.3. For the sampling variables, the estimator will have a form:

$$\hat{P}P(c_i|e) = \frac{1}{T}\sum_{t=1}^{T} P(c_i|c_{-i}^{(t)}, e) \tag{2.11}$$

For variables in $X \backslash C$ that do not participate in sampling, we estimate the posterior marginals as follows:

$$\hat{P}(x_i|e) = \frac{1}{T}\sum_{t=1}^{T} P(x_i|c^{(t)}, e) \tag{2.12}$$

Thus, for each variable in the network, we only need to compute $P(c_i|c_{-i}^{(t)}, e)$ or $P(x_i|c^{(t)}, e)$ and maintain a running sum to obtain the posterior marginal estimation. The total amount of additional memory required is O(N).

2.6.1 Cutset sampling algorithm.

Thus, assume we are given a Bayesian network $\mathcal{B}$ with variables X. Let C=$\{C_1, C_2, ..., C_m\}$ be a subset of variables. We want to generate samples from space C where each sample $c^{(t)}$ is an instantiation of variables in C.

In order to generate samples, following the Gibbs sampling scheme, we need to sample values of variables in C from conditional distributions:

$$c_i \leftarrow P(c_i|c_{-i}^{(t)}, e)$$

Thus, we need to obtain $P(c_i|c_{-i}^{(t)}, e)$ for each sampling variable $C_i \in T$. We also need to obtain $P(x_i|c^{(t)}, e)$ for each of the remaining variables in order to estimate posterior marginals.

The relevant conditional distributions can be computed by exact inference algorithms whose complexity is tied to the network structure. We use JTC as a generic name for a class of variable-elimination or join tree-clustering algorithms that compute that compute the exact joint probabilities $JTC(e)$ as well as the exact posterior marginals $JTC(x_i|e)$ of node X_i given evidence e. [Lauritzen & Spiegelhalter 1988; Dechter 1999; Jensen, Lauritzen, & Olesen 1990]. It is known that the complexity of JTC is time and space exponential in the induced-width w of the moral graph of the network with evidence variables removed.

We can use JTC algorithm to obtain exact probabilities $P(c_i|c_{-i}^{(t)}, e)$ and $P(x_i|c^{(t)}, e)$. The sampling probability $P(c_i|c_{-i}^{(t)}, e)$ can be obtained by marginalising joint distributions obtained by JTC algorithm. For each sampling node C_i we compute $JTC(c_i, c_{-1}^{(t)}, e)$ for each value $c_i in D(C_i)$. Then, we marginalize:

$$P(c_i|c_{-i}^{(t)}, e) = \alpha JTC(c_i, c_{-1}^{(t)}, e)$$

The nodes in C as well as evidence nodes E remain observed throughout the exact inference computation. Thus, the complexity of the exact inference is tied to the induced width of the graph conditioned on C and E.

Evidence nodes can reduce complexity of the graph by breaking cycles (we demonstrate this in chapter 4). Generally, the more nodes we instantiate, the easier exact inference becomes until instantiated nodes form a cycle-cutset and graph becomes a polytree. Choosing sampling nodes C carefully so as to minimize the induced width w of the graph when nodes in C are observed, we can control the complexity of exact inference required. Thus, we arrive at the concept of w-cutset.

DEFINITION **2.6.1 (w-cutset)** *Given an undirected graph $G = (X, E)$, if C is a subset of X such that, when observed, the induced width of the resulting graph is $\leq w$, then C is called a w-cutset of G and the adjusted induced width of G relative to C is w.*

Using the notion of w-cutset, we can balance sampling and exact inference to optimize the performance as necessary. At one end of the spectrum, we will have plain Gibbs sampling which is fast, requires linear amount of additional space, but has higher variance. At the other end, we have exact inference that requires time and space exponential in the induced width of the graph.

The difficult problem remaining is that of finding an optimal w-cutset. Roughly, we want to find minimum size w-cutset. In section 2.8, we describe several simple heuristic schemes for finding w-cutset. We provide a more rigorous analysis and empirical study of this problem in chapter 3. For now, we assume that w-cutset is given as an input to the problem.

w-Cutset Sampling
Input: A belief network $(\mathcal{B})$, w-cutset $C = \{C_1, ..., C_m\}$, evidence e.
Output: A set of samples c^t, $t = 1...T_c$.
1. **Initialize:** Assign random value c_i^0 to each $C_i \in C$ and assign e.
2. **Generate samples:**
For t = 1 to T, generate a new sample c^t:
For i = 1 to m, compute new value c_i^t for variable C_i as follows:
Using algorithm **join-tree clustering**, compute $JTC(C_i, c_{-i}^{(t)}, e)$ and sample:

$$c_i^{t+1} \rightarrow P(c_i|c_{-i}^{(t)}, e) = \alpha JTC(c_i, c_{-i}^{(t)}, e) \quad (2.13)$$

End For i
End For t

Figure 2.2: w-*Cutset sampling* Algorithm

The w-cutset sampling algorithm using the systematic scan Gibbs sampler is formally defined in Figure 2.2. Clearly, it can be adapted to use with the random scan Gibbs sampler.

We provide a proof of the convergence of this general scheme in Section 2.7. Namely, computing $P(x_i|e)$ by cutset sampling is (1) guaranteed to converge to the exact quantities. In general, cutset sampling requires fewer samples to converge than full sampling [Casella & Robert 1996; Groot 1986; J. Liu & Kong 1994].

Example. Consider again a belief network shown in Figure 1.2. When sampling from set $C = \{X_2, X_5\}$ (although there is a better cutset $C = \{X_3\}$), we will have to compute for each sample t the probabilities $P(x_2|x_5^{t-1})$ and $P(x_5|x_2^t)$. These probabilities can be computed using belief propagation over the singly connected network (Figure 1.2, right) or bucket elimination in linear time. For each new value of variables X_2 and X_5, we profane the updated messages through the (singly-connected) network. The desired joint $P(x_2', x_5', e)$ can be computed at any variable and then normalized to yield the conditional distribution.

2.6.2 Complexity

Clearly, computing a new sample c^t in cutset-sampling is more complex (step 1) than Gibbs sampling. However, we can control that complexity with the *induced width* parameter w. In particular, it is still very efficient when the cutset C is a cycle-cutset of the Bayesian network ($w = 1$). In this case, JTC reduces to belief propagation algorithm [Pearl 1988; Peot & Shachter 1992] that can compute the joint probability $P(c_i, c_{-i}^{(t)}, e)$ in linear time and then normalize it relative to C_i. When C is a w-cutset, the complexity of JTC (equation 2.13) is exponential in w and will dominate the complexity of generating the next sample. Therefore:

THEOREM **2.6.1 (Complexity of sample generation)** *The complexity of generating a sample by cutset sampling with cutset C is $O(m \cdot d \cdot n \cdot d^w)$ where C is a w-custet of size m, d bounds the variables domain size, and n is the number of nodes.*

COROLLARY **2.6.1** *If C is a cycle-cutset, the complexity of generating a sample by cycle-cutset sampling is linear in the size of the network.*

Computing $P(X_i|e)$ using equation (2.12) requires computing $P(X_i|c^t, e)$ for each variable using JTC algorithm is also exponential in w, the adjusted induced width relative to cutset:

THEOREM **2.6.2** *Given a w-cutset C, the complexity of computing the posterior of all the variables using cutset sampling over T samples is $O(T \cdot n \cdot d^w)$.*

COROLLARY **2.6.2** *If C is a cycle-cutset, the complexity of computing the posterior of all the variables by cycle-cutset sampling is linear in the size of the network.*

2.7 Convergence of cutset-sampling

In this section we demonstrate the convergence of the $\hat{P}(c_i|e)$ and $\hat{P}(x_i|e)$ as defined in equations (2.11) and (2.12) to the correct probabilities $P(c_i|e)$ and $P(x_i|e)$ respectively based on cutset conditioning formula and using the basic assumption that samples are generated from $P(c, e)$. While this proof does not address the issues concerning the convergence of Monte Carlo chain sampling in general, it shows the connection between Gibbs sampling and cutset conditioning.

THEOREM **2.7.1 (cutset convergence)** *Given a network $\mathcal{B}$ over X and a subset of evidence variables E, and given a cutset C, assuming $\hat{P}(c_i|e)$ and $\hat{P}(x_i|e)$ were computed by equations (2.11) and (2.12) over the cutset sample, then $\hat{P}(c_i|e) \rightarrow P(c_i|e)$ and $\hat{P}(x_i|e) \rightarrow P(x_i|e)$ as the number of samples T_c increases.*

Proof. Let $|C| = m$. Let $|X| = n$. The computation of $\hat{P}(c_i|e)$ is done exactly in the same way as in Gibbs sampling. There are several different ways to prove convergence of Gibbs sampling and we will not repeat them here. Therefore, based on the correct convergence of Gibbs sampling we can conclude that $\hat{P}(c_i|e) \rightarrow P(c_i|e)$ as the number of samples increases.

Consider now a variable X_i not in C and not in E. We could write the posterior distribution of variable X_i as follows:

$$P(x_i|e) = \sum_c P(x_i|c, e)P(c|e)$$

Assume that we have generated a collection of samples $c^1, c^2, ..., c^T$ from the correct distribution $P(C|e)$. Let $q(c)$ be the number of times c occurs in the samples. Then, for each tuple $C = c$:

$$P(c|e) = \frac{q(c)}{T} \tag{2.14}$$

After we substitute the right hand side of the equation 2.14 in the expression for $P(x_i|e)$:

$$P(x_i|e) = \sum_c P(x_i|c, e)\frac{q(c)}{T} \tag{2.15}$$

Factoring out $\frac{1}{T}$ we get:

$$P(x_i|e) = \frac{1}{T}\sum_c P(x_i|c, e)q(c). \tag{2.16}$$

Clearly, $\sum_c q(C) = T$. Therefore, we can sum over T instead of summing over instantiations of C, yielding:

$$\sum_c P(x_i|c, e)q(c) = \sum_{t=1}^{T} P(x_i|c, e) \tag{2.17}$$

After replacing the sum over C in (2.16) with the sum over T, we get:

$$P(x_i|e) = \frac{1}{T}\sum_{t=1}^{T} P(x_i|c, e) \tag{2.18}$$

Therefore we obtained expression (2.18), assuming that $\frac{m(c)}{T}$ converges to the exact $P(C|e)$. Since $\hat{P}(c_i|e)$ converges to $P(c_i|e)$ in cutset-sampling, as we have already shown, then we can conclude that $\hat{P}(x_i|e) \rightarrow P(x_i|e)$.

■

2.8 Selecting w-cutset

Selecting an optimal w-cutset for a given w is critical to obtaining good approximations. An optimal w-cutset for cutset sampling can be defined as the w-cutset that allows to obtain the best approximation within the given time period. The quality of the approximation is affected by several factors such as the size of the cutset, correlations between sampling variables, and number of samples. Smaller cutset and less correlation between variables lead to reduction in sampling variance and improve convergence of Gibbs sampler so that we can obtain better estimates using fewer samples. As mentioned earlier, the complexity of exact inference is linear on the cutset size $|C| = M$ and exponential in the induced width w of the network with w-cutset nodes

instantiated. Although the smaller cutset results in a smaller linear factor M, the increase in the exponential factor w may render exact inference prohibitively expensive and, given a fixed amount of time, we maybe able to generate very few samples using a small cutset compared to a larger one. Since, convergence also depends on the number of samples generated, reducing cutset size and increasing w beyond some threshold value w_{max} may become impractical unless reduction in M is significant.

Ideally, we want to combine all those factors into a single optimization function, but, in practice, it is hard to measure and predict their joint effects. Since we cannot quantify the effect of the collapsing Gibbs sampling space or predict the maximum correlation factor, we will limit our objective function to optimizing the relationship between the size of the cutset and the complexity of exact inference. Optimizing only those two factors is still a difficult problem. Given two w-cutsets, C_1 and C_2 with the same induced width bound w such that $|C_1| < |C_2|$, instantiation of a larger cutset C_2 may result in a fewer clusters of large size in the tree-decomposition used by JTC and allow to generate samples faster.

We have tried several greedy schemes for selecting an optimal w-cutset. Each scheme starts with a set C that contains all nodes in X in topological order except evidence E: $C = X \setminus E$ and then removes nodes from C in some order until no other node can be removed without violating the maximum induced width bound w_{max}.

Greedy Algorithm (GA) In the topological order, process nodes in C. Select the next node C_i from C and evaluate the induced width of the bucket tree w_i with nodes $C \setminus C_i$ and E observed. If $w_i \leq w_{max}$, remove node C_i from sampling set: $C = C \setminus C_i$.

Monotonous Greedy Algorithm (GA) First, obtain 1-cutset by removing from C (in some order) all such nodes that the adjusted induced width w of the min-fill ordering of nodes $X \setminus C, E$ is bounded by $w = 1$. The 1-cutset becomes a starting sampling set for selection of 2-cutset. We repeat this process selecting a $(w+1)$-cutset from w-cutset until maximum adjusted induced width w_{max} is reached. Following this scheme, nodes with smaller degrees will be removed from the sampling set first unless they are a part of a large family.

Heuristic Greedy Algorithm (HG) In this scheme, we order nodes in C by the size of the Markov blanket (consisting of node's children, parents, and children's parents) which reflects how many conditional probabilities tables sizes would be reduced if a given node was instantiated. Then, the Greedy Algorithm was applied.

	Alg	w*=2		w*=3		w*=4		w*=5		w*=6	
		|C|	Time	|C|	Time	|C|	Time	|C|	Time	|C|	Time
cpcs54	GA	24	3.73	16	6.42	14	11.2	10	20.8	8	38
2000	MG	25	3.51	18	5.55	15	15.9	12	24.3	11	30.2
samples	HG	23	3.68	15	4.67	11	10.3	10	15.3	8	19.1
cpcs179	GA	16	2.08	12	5.3	9	6.04	6	51	5	128
200	MG	17	2.47	12	5.22	9	11.2	7	33.1	4	213
samples	HG	16	2.15	11	2.91	8	16.6	6	64	5	162
cpcs360b	GA	28	4.94	22	4.12	19	4.62	17	6.42	17	7.74
100	MG	28	4.23	21	4.61	19	4.56	16	5.72	15	9.11
samples	HG	27	4.34	21	3.73	18	3.82	16	5.02	16	7.01
cpcs422b	GA	81	6.54	69	5.89	60	6.59	55	4.83	57	10.8
20	MG	83	5.61	72	5.99	64	4.44	58	6.75	53	7.52
samples	HG	78	5.55	66	5.5	57	6.16	50	7.42	46	8.13

Figure 2.3: Size of the w-cutset and sampling time for a fixed number of samples obtained by different algorithms: GA=greedy algorithm, MG=monotonous greedy algorithm, HG=heuristic greedy algrorithm.

The size of the minimal w-cutset found by each algorithm and corresponding time to generate a fixed number of samples (specified in the table after the benchmark's name) are shown in Figure 2.3 (tested on 1GHz CPU, 128 Mb RAM PC). Overall, the heuristic greedy search usually finds the smallest w-cutset and shows the best time. Surprisingly, a simple greedy search also performs well. Most of the time, it finds w-cutset of the same size as heuristic greedy search or second smallest. The monotonous greedy search consistently

yields in performance to the heuristic greedy search (with a few deviations) and performs comparably with simple greedy search.

The described schemes for selecting w-cutset are presented primarily to demonstrate the influence of the different factors on the performance of cutset sampling. Those algorithms are not optimal and further investigation is necessary to identify good w-cutset selection methods.

2.9 Experiments

In this section, we present empirical studies of the w-cutset sampling algorithm for Bayesian networks. In section 2.9.1, we compare the performance of the plain Gibbs sampler that samples from $X \backslash E$ and cycle-cutset sampling that is a special instance of w-cutset sampling where $w = 1$. In section 2.10, we compare the performance of Gibbs sampler and w-cutset for a range of values of w. Fixing the number of samples generated, we show that convergence of cutset sampling dramatically increases as predicted theoretically. When the sampling time is bounded, the performance of w-cutset sampling varies with the time required for exact inference. However, experiments clearly that there exists a range of w-values where w-cutset sampling outperforms Gibbs sampler. In fact, in some instances, despite reducing the sampling set size, we are able to generate samples faster than Gibbs sampler to efficient implementation of JTC algorithm.

2.9.1 Cycle-Cutset Sampling

We compared cycle-cutset sampling with full Gibbs sampling on several CPCS networks, random networks, Hailfinder network, and coding networks. Generally, we are interested in how much accuracy we can achieve in a given period of time. Therefore, we provide here figures showing accuracy of the Gibbs and cycle-cutset sampling as a function of time. For comparison, we also show the accuracy of Iterative Belief Propagation algorithm (IBP) after 25 iterations. IBP is an iterative message-passing algorithm that performs exact inference in Bayesian networks without loops ([Pearl 1988]). It can also be applied to Bayesian networks with loops to compute approximate posterior marginals. The advantage of IBP as an approximate algorithm is that it is very efficient. It requires linear space and usually converges very fast. IBP was shown to perform well in practice ([I. Rish & Dechter 1998; K. P. Murphy & Jordan 1999]) and is considered the best algorithm for inference in coding networks where finding the most probable variable values equals the decoding process.

For each Bayesian network $\mathcal{B}$ with variables $X = \{X_1, ..., X_n\}$, we computed exact posterior marginals $P(x_i|e)$ using bucket-tree elimination and computed the mean square error (MSE) in the approximate posterior marginals $\hat{P}(x_i|e)$ for each approximation scheme:

$$MSE = \frac{1}{\sum_i |D(x_i)|} \sum_i \sum_{D(x_i)} (P(x_i|e) - \hat{P}(x_i|e))^2$$

and averaged MSE over the number of instances tried. In all networks, except for coding networks, evidence nodes were selected at random. The cutset was always selected so that evidence and sampling nodes together constitute a cycle-cutset of the network using the mga algorithm ([Becker, Bar-Yehuda, & Geiger 1999]).

CPCS networks. We considered four CPCS networks derived from the Computer-based Patient Case Simulation system. The largest network, cpcs422b, consisted of 422 nodes with induced width $w^* = 23$. With evidence, its cycle-cutset size was 42. The results are shown in Figure 2.4. Each chart title specifies network name, number of nodes in the network N, the size of evidence set $|E|$, size of cycle-cutset (sampling set) $|C|$, and induced width w^* of the network instance. For all four CPCS networks, we observed that the cutset sampling is far better than Gibbs sampling. In case of cpcs179 (Figure 2.4, middle), cpcs360b (Figure 2.4, bottom), and cpcs422b (Figure 2.5) cutset sampling achieves even greater accuracy than IBP. Gibbs sampling does not converge on cpcs179 due to non-ergodic properties of the network. The cutset sampling overcomes this limitation because the cycle-cutset selected is an ergodic subspace.

Random networks. We generated a set of random networks with bi-valued nodes. Each network contained total of 200 nodes. The first 100 nodes, $\{X_1, ..., X_{100}\}$, were designated as root nodes. Each non-root node X_i was assigned 3 parents selected randomly from the list of predecessors $\{X_1, ..., X_{i-1}\}$. The conditional probability table values $P(X_i = 0|pa(X_i))$ were chosen randomly from a uniform distribution. We collected data for 10 instances (Figure 2.6, top). Cutset sampling always converged faster than Gibbs sampling.

2-Layer networks. We generated a set of random 2-layer networks with bi-valued nodes. Each network contained 50 root nodes (first layer) and a total of 200 nodes. Each non-root node (second layer) was assigned a maximum of 3 parents selected at random from the root nodes. The conditional probability table values $P(X_i = 0|pa(X_i))$ were chosen randomly from uniform distribution. We collected data for 10 instances (Figure 2.6, middle). On those types of networks, Iterative Belief Propagation often does not perform well.

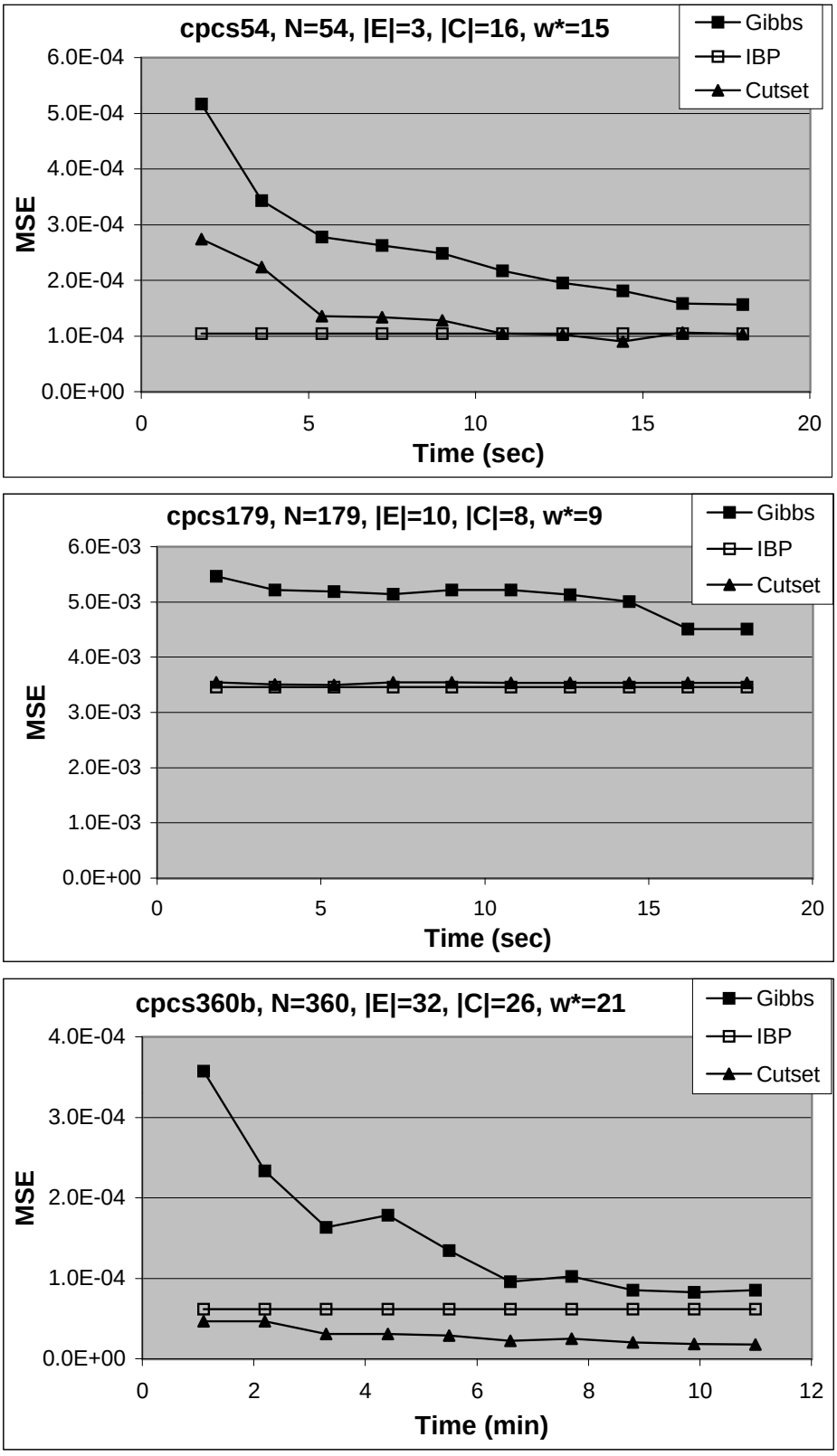


Figure 2.4: Comparing cycle-cutset sampling, Gibbs sampling and IBP on CPCS networks averaged over 3 instances each. MSE is a function of time.

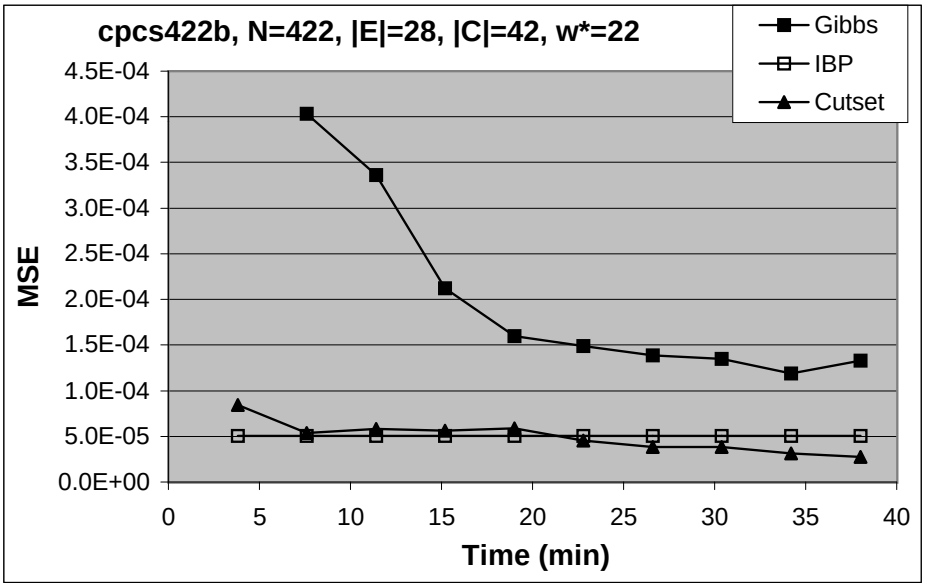


Figure 2.5: Comparing cycle-cutset sampling, Gibbs sampling and IBP on cpcs422b network averaged over 2 instances. MSE is a function of time.

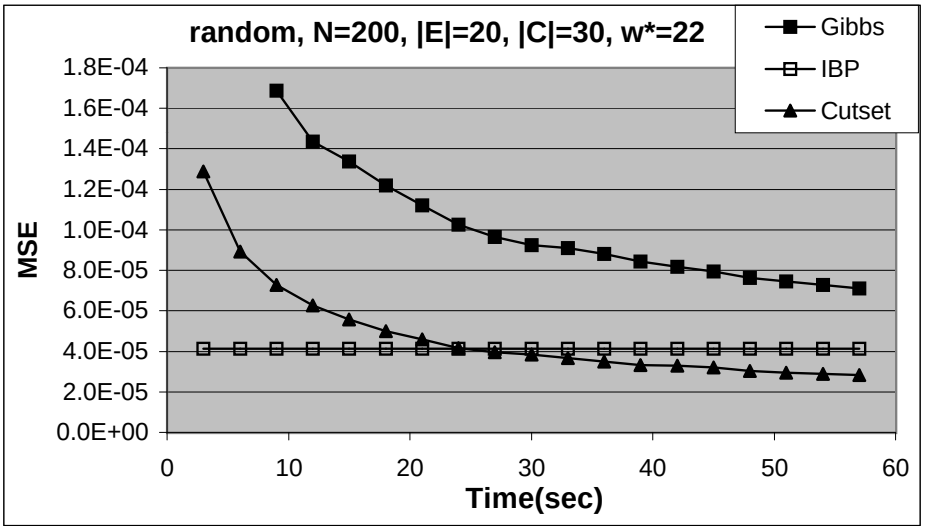
And, as our experiments show, cutset sampling outperforms both Gibbs sampling and IBP (although it takes longer time to converge than IBP).

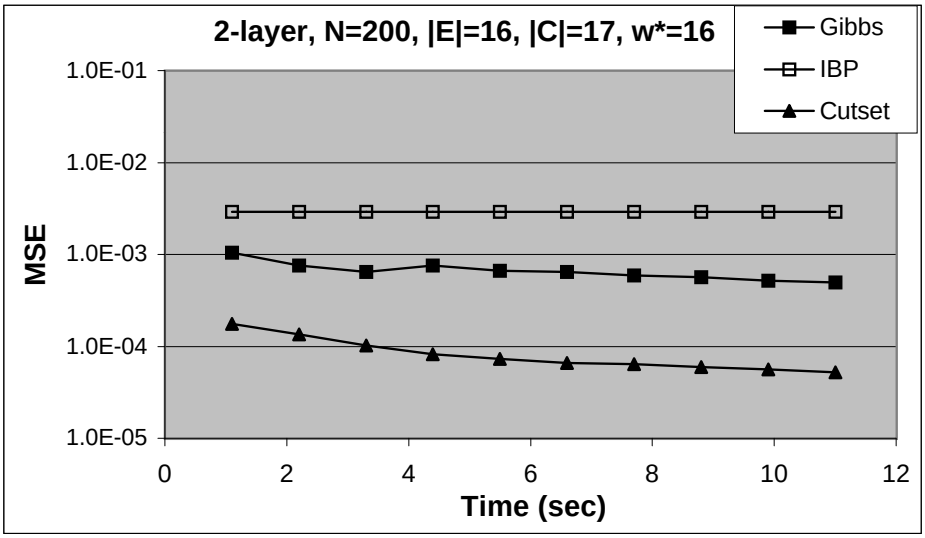
Coding Networks. We experimented with coding networks with 100 nodes (25 coding bits, 25 parity check bits). The results are shown in Figure 2.6, bottom. Those networks had cycle-cutset size between 12 and 14 and induced width between 13 and 16. The parity check matrix was randomized; each parity check bit had three parents. We computed MSE over all coding bits and averaged over 10 networks. Coding networks are not ergodic due to the deterministic parity check function. As a result, Gibbs sampling does not converge. At the same time, the subspace of code bits only is ergodic and cutset sampling, that samples a subset of coding bits, converges and generates results comparable to those of IBP. In practice, IBP is certainly preferable for coding networks since the size of the cycle-cutset grows linearly with the number of code bits.

Hailfinder network. Hailfinder is a non-ergodic network with many deterministic relationships. It has 56 nodes and cycle-cutset of size 5. Indeed, this network is easy to solve exactly. We used this network to compare the behavior cutset sampling and Gibbs sampling in non-ergodic networks. As expected, Gibbs sampling fails while cycle cutset sampling computes more accurate marginals and its accuracy continues to improve with time (Figure 2.7).

In summary, the empirical results demonstrate the cycle-cutset is cost-effective time-wise and is superior to Gibbs sampling. We measured the ratio $R = \frac{M_g}{M_c}$ of the number of samples generated by Gibbs M_g to the number of samples generated by cycle-cutset sampling M_c in the same time period. For cpcs54, cpcs179, cpcs360b, and cpcs422b the ratios were correspondingly 1.4, 1.7, 0.6, and 0.5. We also obtained R=2.0 for random networks and R=0.3 for random 2-layer networks. While cutset sampling algorithm often takes more time to generate a sample, it produced substantially better results overall due to its variance reduction. In some cases, cutset sampling could actually compute samples faster than Gibbs sampler. in which case the improvement in the accuracy was due to both large sample set and variance reduction. Cutset sampling also achieves better accuracy than IBP on some CPCS and random networks although takes more time to achieve same or better accuracy. In 2-layer networks and coding networks, cycle-cutset sampling achieves the IBP level of accuracy very quickly and is able to substantially improve with time.

The direction of our future work is to investigate methods for finding a sampling set with good convergence properties. Some of the factors that strongly affect convergence of MCMC methods are the sampling set size, the complexity of sample generation, and the correlations between variables. Reducing sampling set size generally leads to a reduction in the sampling variance due to Rao-Blackwellisation, but it also results in the increased complexity of exact inference when generating a new sample. Another factor is strong correlations between sampled variables (deterministic probabilities, present in non-ergodic networks, are an extreme example of strong correlation). If two variables are strongly dependent, it is preferred to either integrate one of them out or group them together and sample jointly (as in blocking Gibbs sampler) (see





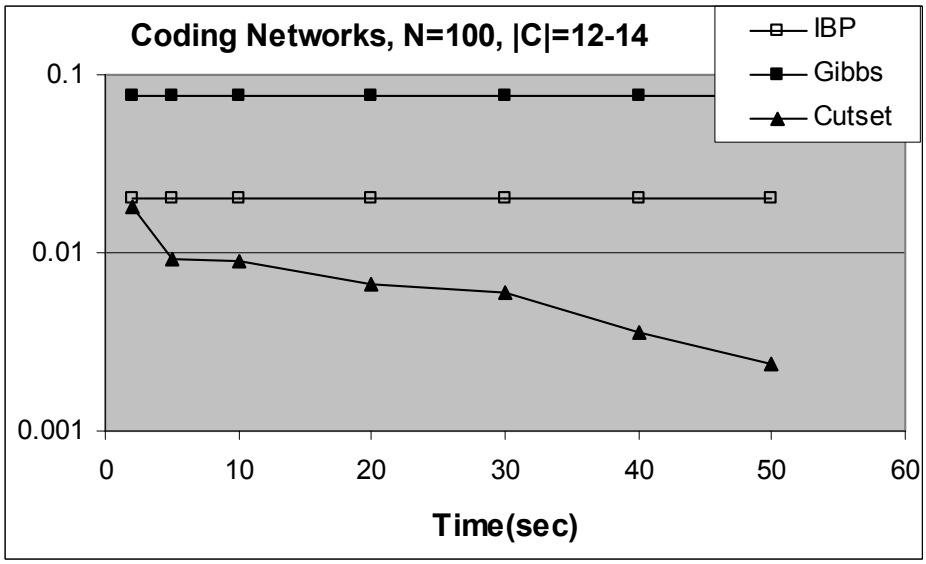


Figure 2.6: Comparing cycle-cutset sampling, Gibbs sampling and IBP on random networks (top), 2-layer random networks (middle), and coding networks, σ=0.4 (bottom), averaged over 10 instances each. MSE as a function of time.

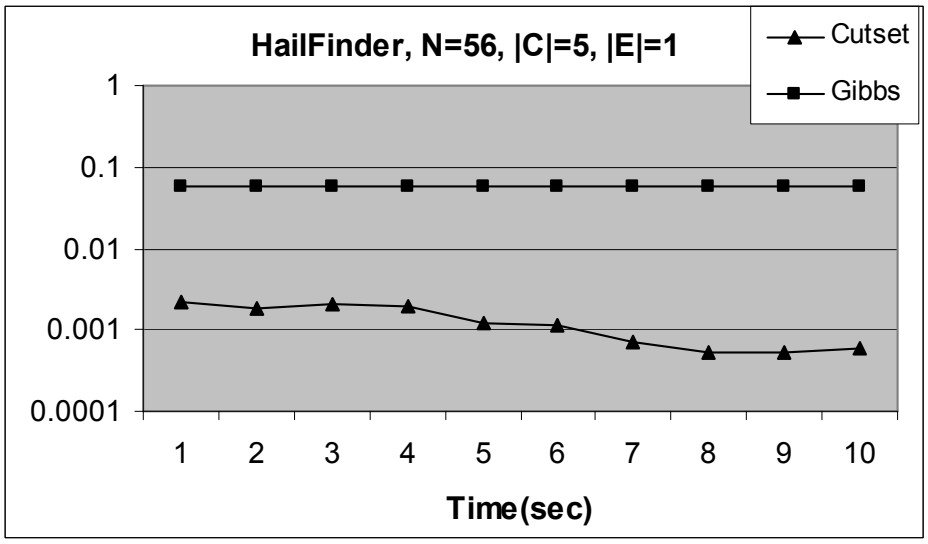


Figure 2.7: Comparing cycle-cutset sampling and Gibbs sampling on Hailfinder network, 1 instance. MSE as a function of time.

[J. Liu & Kong 1994]). Taking above into consideration, a good sampling set could be defined as a minimal w-cutset with a small w and with all strongly-correlated variables removed.

2.10 w-cutset Sampling

2.10.1 Computing an Error Bound

Gibbs sampling provides a simple sampling scheme for Bayesian networks that is guaranteed to converge to the correct posterior distribution in ergodic networks. The drawback of Gibbs sampling compared to many other sampling methods is that it is hard to estimate how many samples are needed to achieve a certain degree of convergence. It is possible to derive bounds on the absolute error based on sample variance for any sampling method if it generates independent samples, for example forward sampling and importance sampling. In Gibbs and other Monte Carlo methods, samples are dependent, and we cannot apply the confidence interval estimate directly.

We can create independent samples restarting the chain after every T samples. Let $\hat{P}_m(x_i|e)$ be an estimate derived from a single chain $m \in [1, ..., M]$ of length T (meaning, containing T samples) as defined in equations (2.11)-(2.12). The estimates $\hat{P}_m(x|e)$ are independent random variables. Every time we restart the chain, we randomly assign new values to each sampling variable and this assignment is independent from the results generated in previous chains. If we generate a total of M such chains, the posterior marginals $\hat{P}(x|e)$ will be an average of the M results obtained from each chain:

$$\hat{P}(x|e) = \frac{1}{M} \sum_{m=1}^{M} \hat{P}_m(x|e) \tag{2.19}$$

Then, we can use the well-known sample variance estimate for random variables:

$$S^2 = \frac{1}{M-1} \sum_{m=1}^{M} (\hat{P}_m(x|e) - \hat{P}(x|e))^2$$

An equivalent representation for sampling variance is:

$$S^2 = \frac{\sum_{m=1}^{M} \hat{P}_m^2(x|e) - M\hat{P}^2(x|e)}{M-1} \tag{2.20}$$

where S^2 is easy to compute incrementally storing only the running sums of $\hat{P}_m(x|e)$ and $\hat{P}_m^2(x|e)$. By the Central Limit Theorem, ergodic mean converges to Normal distribution $N(\mu, \sigma)$. Therefore, we can compute

confi dence interval in the $100(1-\alpha)$ percentile used for random variables with normal distribution for small sampling set sizes ([Hogg & Tanis 2001]). Namely:

$$P\left[P(x|e) \in [\hat{P}(x|e) \pm t_{\frac{\alpha}{2},(M-1)}\frac{S}{\sqrt{M}}\right] = 1-\alpha \tag{2.21}$$

where $t_{\frac{\alpha}{2},(M-1)}$ is a table value from t distribution with $(M-1)$ degrees of freedom. In general, this method may yield confi dence interval that is too large to be useful. In the experimental section, we provide results showing 90% confi dence interval computed for Gibbs sampler and cutset sampling over 20 Markov chains and analyze the feasibility of using confi dence interval as a metrics in evaluating performance of Gibbs and cutset sampling.

2.10.2 Methodology

The primary goal of our empirical study was to investigate performance of w-cutset as a function of w. We compared the performance of Gibbs sampling, w-cutset sampling for different values of w and a special case of w-cutset sampling, cycle-cutset sampling. In all empirical studies, cycle-cutset of the network was found using the mga algorithm ([Becker, Bar-Yehuda, & Geiger 1999]). The w-cutset was selected using monotonous greedy algorithm (MG) defi ned in Section 2.8. It is not the most effi cient scheme, but its performance is comparable with other w-cutset selection schemes and it guarantees that each $(w+1)$-cutset is a proper subset of w-cutset. Therefore, given the same number of samples, the $(w+1)$-cutset is predictably better following the Rao-Blackwellisation theory. This property allows us to eliminate some of the uncertainty associated with selecting different sampling sets and focus the empirical study on the trade-offs between cutset size reduction and the associated increase in the complexity of the exact inference as we gradually increase the induced width bound w.

Our benchmarks are several CPCS networks, grid networks, 2-layer networks, random networks, and coding networks. All the sampling algorithms were given a fi xed time bound. Small networks, cpcs54 ($w^* = 15$) and cpcs179 ($w^* = 8$), where exact inference is easy, we allocated 20 seconds. Larger networks were allocated up to 20% of the exact inference time not to exceed 6 minutes.

Each sampling algorithms generated $M = 20$ independent Markov chains of size $T_m = \frac{1}{20}T$ where T is the maximum number of samples that an algorithm could generate within the given time bound.The resulting chain length for each sampling algorithm for different benchmarks as well as sampling set sizes are given in Figure 2.8. Each chain m produces approximation $\hat{P}_m(x_i|e)$ for the posterior marginals over T_m samples as shown in eq.(2.19). We obtained a fi nal estimate by averaging over $\hat{P}_m(x_i|e)$ values.

For comparison, we also show the performance of Iterative Belief Propagation (IBP) algorithm on each benchmark after 25 iterations. IBP is an iterative message-passing algorithm that performs exact inference in Bayesian networks without loops ([Pearl 1988]). Applied to Bayesian networks with loops, it computes approximate posterior marginals. The advantage of IBP as an approximate algorithm is that it requires linear space and usually converges fast.

The quality of the approximate posterior marginals is measured by Mean Square Error (MSE) between the exact posterior marginals $P(x_i|e)$ and the approximate posterior marginals $\hat{P}(x_i|e)$:

$$MSE = \frac{1}{\sum_i |D(x_i)|}\sum_i \sum_{D(x_i)} (P(x_i|e) - \hat{P}(x_i|e))^2$$

averaged over the number of instances tried. The exact posterior marginals were obtained via bucket-tree elimination algorithm ([Dechter 1999]). All experiments were performed on 2 GHz CPU 256 Mb RAM PC.

2.10.3 Benchmarks

CPCS networks. CPCS networks are derived from the Computer-based Patient Case Simulation system. The nodes in CPCS networks correspond to diseases and fi ndings and conditional probabilities describe their correlations. For all CPCS networks, we present two charts: one chart demonstrates the convergence over time of Gibbs sampling and w-cutset sampling for several selected w values; the second chart plots the change in the quality of approximation (MSE) as a function of w for two time lines - half of the total sampling time and the total sampling time.

cpcs54 network consists of $N = 54$ nodes and has a relatively large cycle-cutset of size $|LC| = 15$ ($> 25\%$ of the nodes). Its induced width is 15. The performance of Gibbs sampling and cutset sampling is shown in Figure 2.9. The chart title contains the following notation: N - number of nodes in the network; $|E|$ - average number of evidence nodes; $|LC|$ - size of cycle-cutset; $w*$ - adjusted induced width of the network. The results are averaged over 10 instances with different evidence, $1-4$ observed nodes. The fi rst

	#samples								
	Gibbs	LC	w*=2	w*=3	w*=4	w*=5	w*=6	w*=7	w*=8
cpcs54	1000	700	700	450	300	200	100	-	-
cpcs179	130	60	300	80	40	20	-	-	-
cpcs360b	500	900	1000	1000	900	700	400	300	-
cpcs422b	10	14	200	200	180	170	140	75	60
grid	2000	500	300	260	150	105	60	35	20
random	2000	1000	1400	700	450	300	140	75	-
2layer	200	700	900	320	150	75	40	-	-
coding	650	450	800	600	250	150	100	-	-

	Sampling Set Size								
	Gibbs	LC	w*=2	w*=3	w*=4	w*=5	w*=6	w*=7	w*=8
cpcs54	51	16	25	18	15	12	11	-	-
cpcs179	151	8	17	12	9	7	4	-	-
cpcs360b	328	26	28	21	19	16	15	14	-
cpcs422b	392	42	78	67	59	53	48	43	37
grid	410	169	163	119	95	75	60	50	13
random	190	30	61	26	25	24	18	17	-
2layer	185	17	22	15	13	12	11	-	-
coding	100	26	38	23	18	17	-	-	-

Figure 2.8: Markov chain sample count and sampling set size as a function of w.

graph, Figure 2.9, shows the means square error in the posterior marginals as a function of time for Gibbs sampling, cycle-cutset sampling, and w-cutset sampling for w=2 and w=3. The second chart shows accuracy as a function of w. The first point corresponds to Gibbs sampling ($\sim w* = 1$), other points correspond to cutset sampling with different bound w in range from 2 to 6. The cycle-cutset result is embedded with the w-cutset values at w=4. In cpcs54, the best results are obtained by 3-cutset sampling; the cycle-cutset sampling is second best.

cpcs179 network consists of N=179 nodes. It has a small cycle-cutset of size $|LC| = 8$ but with a relatively large corresponding adjusted induced width w*=8. The cutset sampling algorithm performance is very similar for all different cutsets (cycle-cutset, 2-,3-,4-,5-cutset) as seen in the accuracy vs. time chart at the top of Figure 2.10 and at the accuracy vs. w chart at the bottom. The cutset sampling algorithm performed significantly better compared to Gibbs sampler as the network has some non-ergodic properties.

cpcs360b is a larger CPCS network with 360 nodes, adjusted induced width of 21, and cycle-cutset $|LC| = 26$. Exact inference on cpcs360b averaged ~ 30 min. As we can see from Figure 2.11), the cycle-cutset sampling and the 2-, 3-, and 4-cutset sampling perform comparatively the same. The top chart in Figure 2.11 provides a little more insight demonstrating that in the first half of the sampling time period, 3-cutset and 4-cutset perform better. Later, the 2-cutset performs better. This is typical of MCMC methods where the convergence is guaranteed as number of samples approaches infinity but the improvement in behavior may be non-monotonous. All cutset sampling implementations substantially outperform Gibbs sampling taking advantage of both sampling space reduction and greater efficiency in generating samples. All cutset sampling implementations outperform IBP given enough time.

cpcs422b is the largest of the CPCS networks with 422 nodes, cycle-cutset size $|LC| = 47$, and induced width $w^* = 22$. It also contains several large CPTs so that even when cycle-cutset is instantiated, bucket-tree records the functions of size 14. We observed that w-cutset (but not cycle-cutset!) sampling handled large function sizes efficiently and computed samples an order of magnitude faster than Gibbs sampling or cycle-cutset sampling. In fact, within the 6 minutes time limit given to the sampling algorithm, Gibbs sampling and cycle-cutset sampling could only generate on the order of $10 - 15$ samples each which is statistically insignificant and thus, were left out of the charts. The results for w-cutset sampling are shown in Figure 2.12.

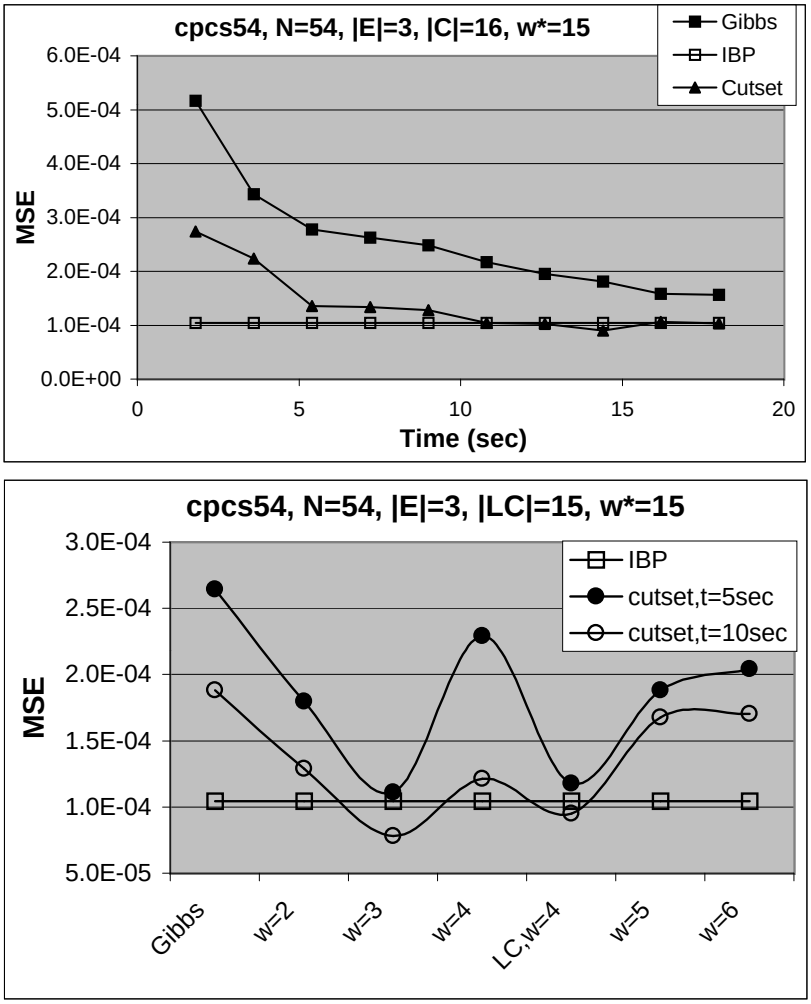


Figure 2.9: cpcs54, 10 instances, time bound=12 seconds.

Note that in cpcs422b, the w-cutset was able to take advantage of the network structure to the extent that allowed to increase w effi ciently. The bottom chart in Figure 2.12 shows that w-cutset performed well in range of $w = 2 - 8$.

Random networks. We generated a set of random networks with N=200 binary nodes and r=50 root nodes with uniform priors. Each non-root node X_i was assigned P=3 parents (selected randomly). The conditional probabilities were also chosen randomly, from uniform distribution. Evidence nodes E were selected at random from leaf nodes (nodes without children). The w-cutset sampling signifi cantly improves over Gibbs sampling and IBP reaching optimal perform for $w = 2 - 4$. In this range, its performance is similar to that of cycle-cutset sampling.

2-Layer networks. We generated a set of random 2-layer networks with binary nodes, with r=50 root nodes (with uniform priors), and a total of N=200 nodes (with binary domains). Each non-root node was assigned 3 parents selected at random from root nodes. The CPT values were chosen randomly from uniform distribution. We collected data for 10 instances (Figure 2.13(a)). As we can see the performance of both Gibbs sampling and IBP is very poor (most likely due to extreme conditional probabilities present in the networks). The w-cutset sampling offers substantial gain in accuracy compared to those algorithms reaching the optimal performance at $w = 2, 3$ that is very near the performance of cycle-cutset sampling.

Grid networks. The grid networks with 450 nodes (15x30) were the only class of the networks where full Gibbs sampling was able to generate samples faster than cutset sampling and produce comparable estimates. The fastest instance of cutset sampling, in this case cycle-cutset sampling, was 4 times slower compared to Gibbs sampling (see Figure 2.8). This indicates that networks with regular network structure (that cutset

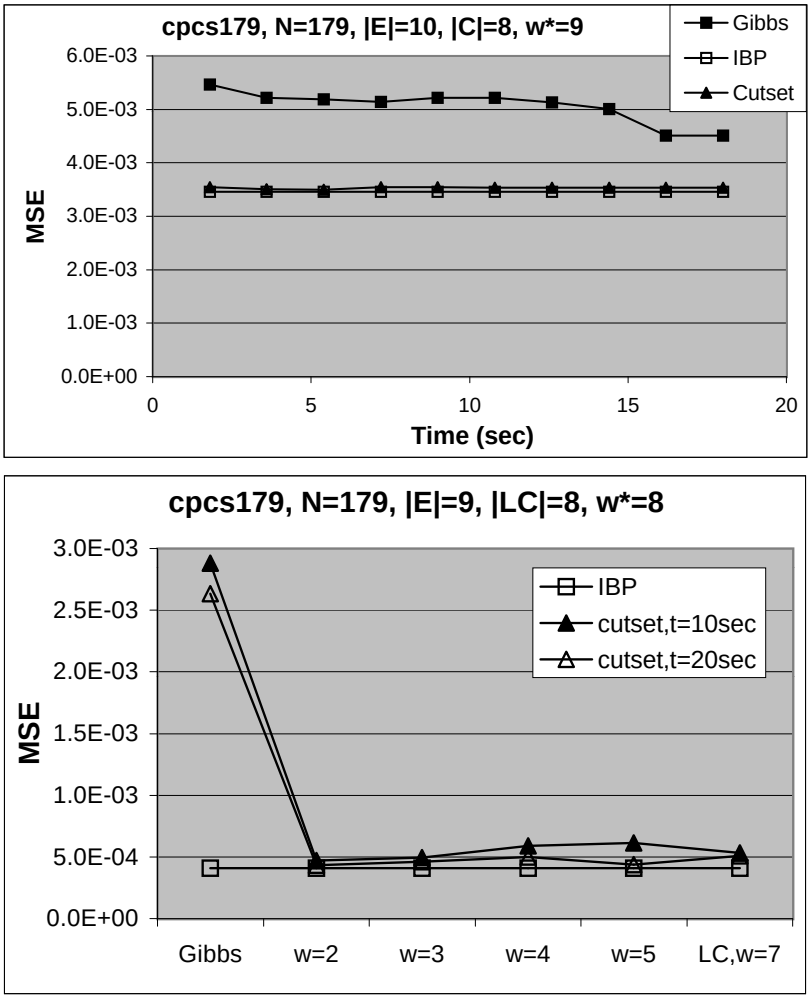


Figure 2.10: cpcs179, 10 instances, time bound=20 seconds.

sampling cannot exploit to its advantage) and small CPTs (in a two-dimensional grid network, each node has at most 3 parents) represent a class of networks where Gibbs sampling is a strong player while cutset sampling requires further optimizations. With respect to the accuracy of the estimates, Gibbs sampler, cycle-cutset sampling, and 3-cutset sampling were the best and achieved similar quality results.

Coding Networks. We experimented with coding networks with 200 nodes (50 coding bits, 50 parity check bits). Each parity check bit was assigned three parents from coding bits. The Gallager code parity check matrices were generated using the source code by David MacKay (see http://www.inference.phy.cam.ac.uk/mackay/CodesFiles.html). The results are shown in Figure 2.14. In this class of networks, the induced speed varied from 18 to 22 making exact inference quite feasible. However, we additionally tested and observed that even a small increase in the network size to 60 code bits, a total of 240 nodes, the induced width exceeds 24 and and exact inference requires considerably longer time while sampling time scales up linearly. We collected results for 10 networks (10 different parity check matrices) with 10 different evidence instantiations (total of 100 instances). In decoding, the Bit Error Rate (BER) is a standard error measure. However, we computed MSE over all unobserved nodes to evaluate the quality of approximate results more precisely. Coding networks are not ergodic due to the deterministic parity check function. As a result, Gibbs sampling did not converge, generated sporadic results, and was left off the charts. At the same time, the subspace of code bits only is ergodic and cutset sampling on a subset of coding bits converges and generates results comparable to those of IBP. Given enough time, cutset sampling can even outperform IBP. The charts in Figure 2.14 show that cycle-cutset has actually proven to be the best cutset for the coding networks closely followed by 2-cutset sampling.

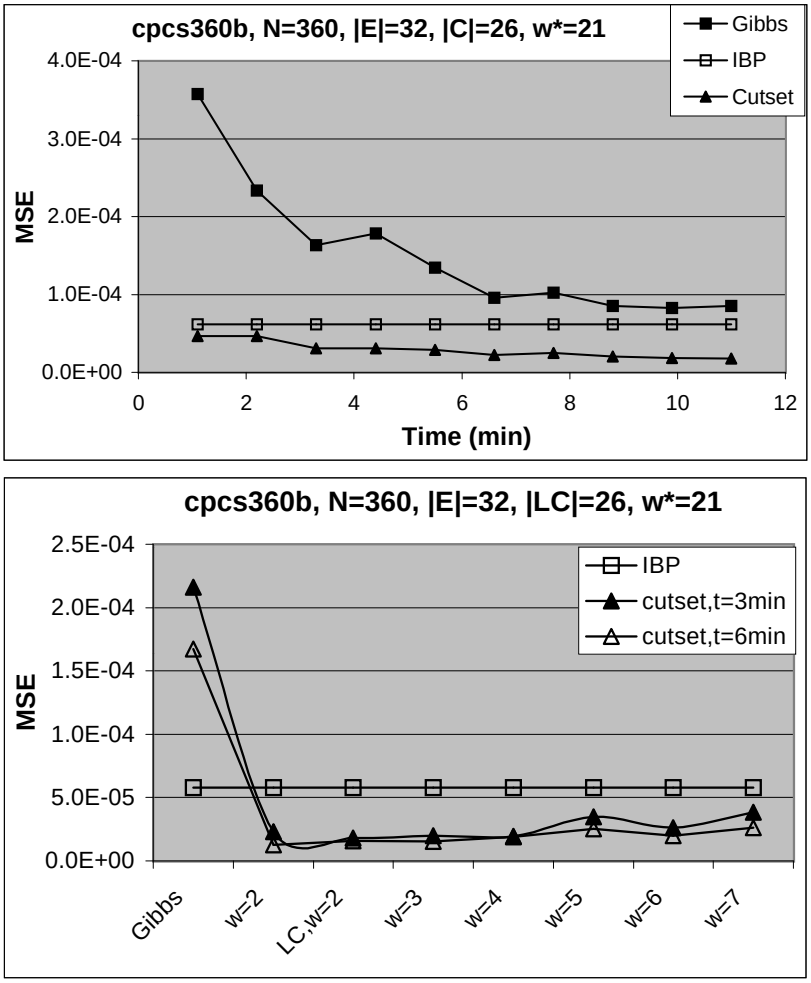


Figure 2.11: cpcs360b, 10 instances, time bound=6 minutes.

2.10.4 Estimating Absolute Error

We have computed a confidence interval for estimated posterior marginal $P(x_i|e)$ based on the on the sampling variance of $P_m(x_i|e)$, the estimates produced by independent Markov chains, as described in Section 2.10.1. We computed sampling variance S^2 from eq.(2.20) and 90% confidence interval $\Delta_{0.9}(x_i)$ from eq.(2.21) and averaged over all nodes:

$$\Delta_{0.9} = \frac{1}{N \sum_i |D(X_i)|} \sum_i \sum_{x_i \in D(X_i)} \Delta_{0.9}(x_i)$$

As noted earlier, estimated confidence interval can be too large to be practical. Thus, we compared $\Delta_{0.9}$ with exact average absolute error Δ:

$$\Delta = \frac{1}{N \sum_i |D(X_i)|} \sum_i \sum_{x_i \in D(X_i)} |\hat{P}(x_i|e) - P(x_i|e))$$

The objective of this study was to observe whether computed confidence interval $\Delta_{0.9}$ (estimated absolute error) reflects true absolute error Δ, that is $\Delta < \Delta_{0.9}$? and how big the estimate is compared to the true error since large confidence interval is not very informative.

Figure 2.15 shows the average confidence interval and average absolute error for a set of benchmarks described earlier. As we can see, for all of the networks except cpcs179, we observe that $\Delta < \Delta_{0.9}$. In cpcs179, the estimated bound is sometimes a little smaller than average error which maybe attributed to non-ergodic properties of cpcs179 (even though the variance is small, the algorithm did not explore the sampling space properly) and stochastic nature of the estimate. Also, we observe that in most cases the estimated

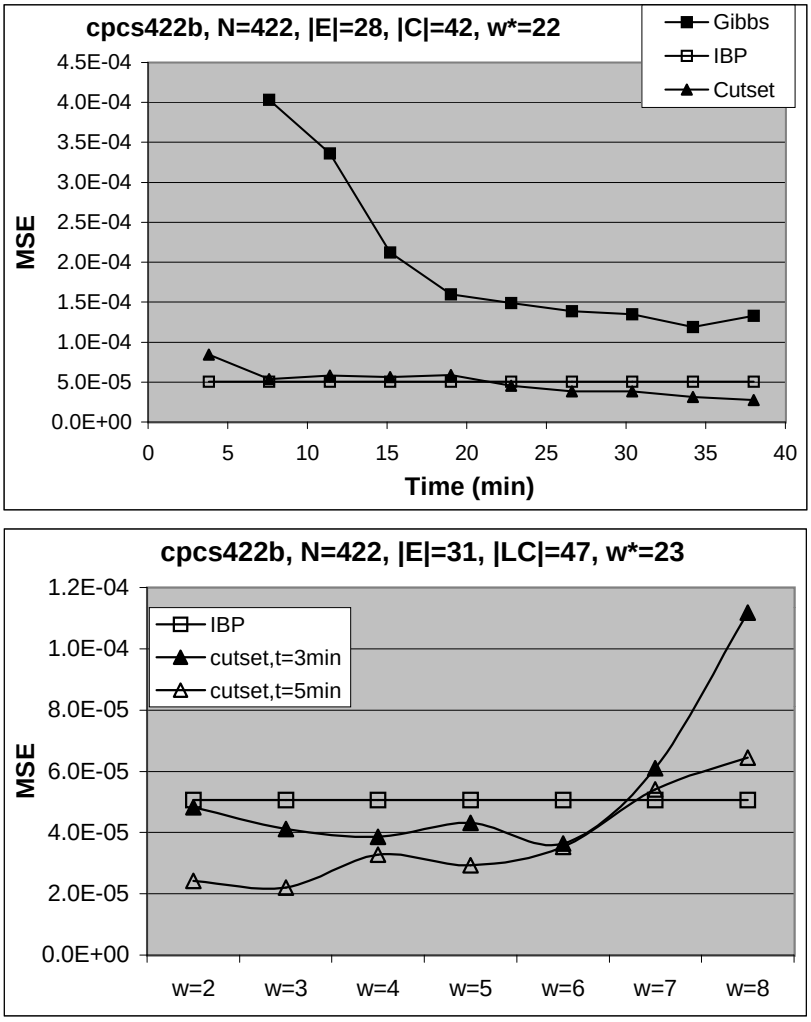


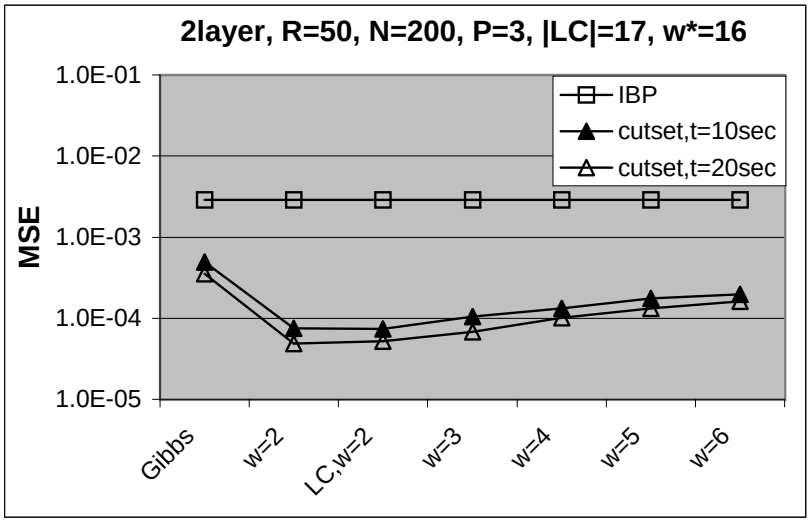
Figure 2.12: cpcs422b, 3 instances, time bound=5 minutes.

confi dence interval is no more than 2-3 times the size of average error. Thus, we can conclude that confi dence interval estimate could be used as a criteria reflecting the quality of the posterior marginal estimate by the sampling algorithm where the comparison against exact posterior marginals is not possible.

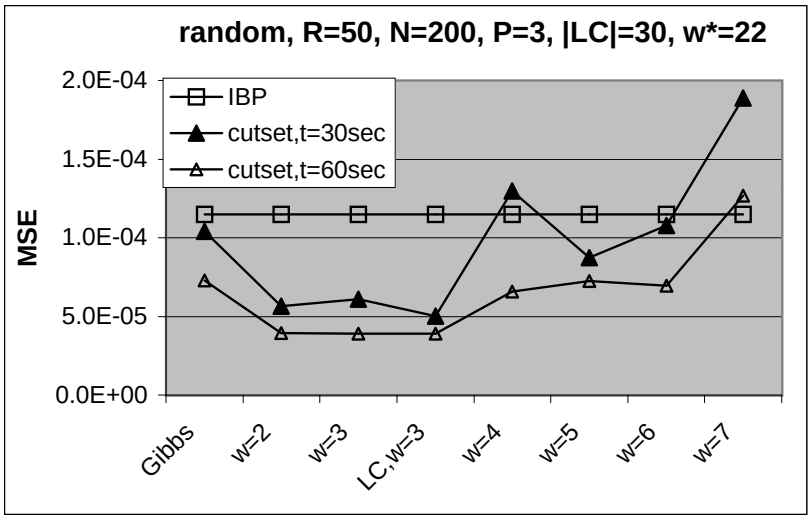
2.10.5 Summary

The empirical evaluation of the performance of w-cutset sampling shows that with the exception of grid networks, w-cutset sampling always outperforms Gibbs sampling (in some instances, generating samples even faster than Gibbs), outperforms cycle-cutset sampling for some w values, and offers a considerable improvement over IBP on several networks. We have discovered a class of networks where we can recommend with a degree of certainty Gibbs or cutset sampling. Gibbs appears to have an advantage in the networks with regular structure and small probability tables (such as grid networks). Cutset sampling performs well when it can take advantage of the underlying network structure and the network contains a few large probability tables (as in cpcs422b).

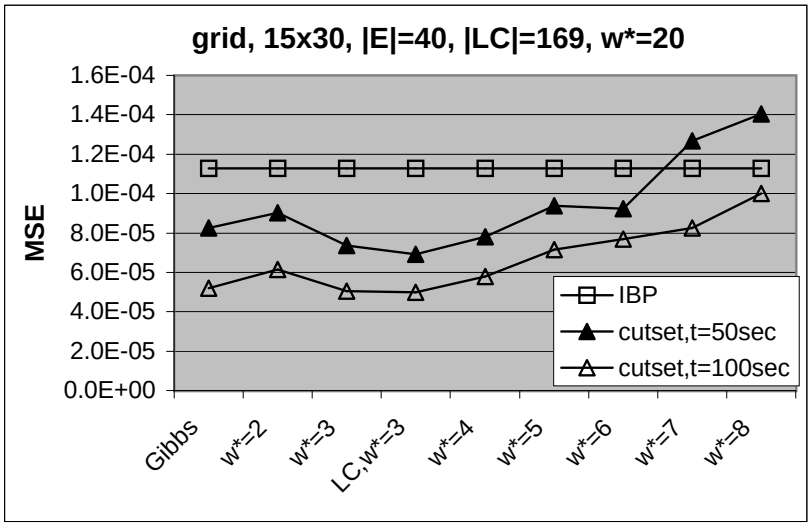
Overall, we can conclude that there exists a range of w values where w-cutset sampling achieves an optimal performance. In many instances, increasing w, up to some threshold value, compensates for the incurred overhead in exact inference due to variance reduction and, in some instances, also due to increased speed of generating samples (discussed in Section 2.8). The performance of w-cutset begins to deteriorate when increase in w results only in a small reduction of sampling set size. An example is cpcs360b network where starting with w=5, increasing w by 1 results in the reduction of sampling set only by 1 node (shown in in Figure 2.3) and leads to the reduction in the size of sampling chain (shown in in Figure 2.8).



(a) 2layer networks, time bound=25 sec.



(b) Random networks, time bound=60 seconds.



(c) Grid networks, 450 nodes (15x30), time bound=100 seconds.

Figure 2.13: Randomized networks , 10 instances each.

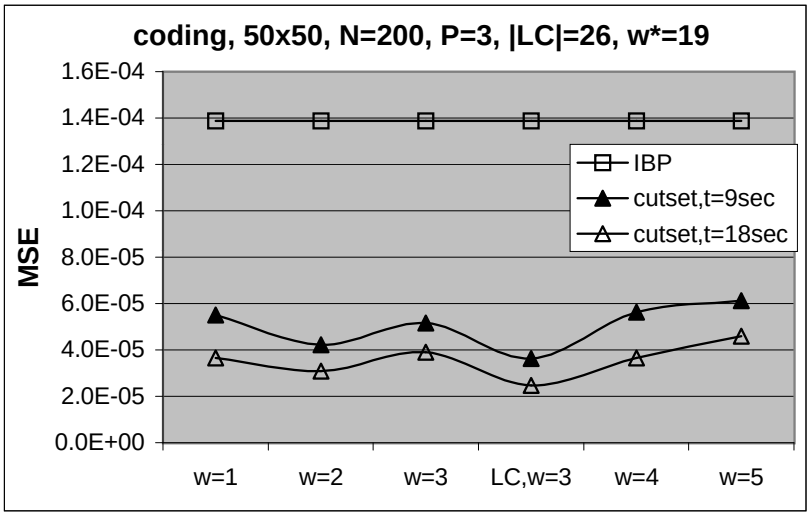


Figure 2.14: Coding networks, 50 code bits, 50 parity check bits, σ=0.4, 100 instances, time bound=6 minutes.

	Error	Gibbs	LC	w*=2	w*=3	w*=4	w*=5	w*=6
cpcs54	**Exact**	0.00117	0.000662	0.00076	0.000757	0.000703	0.00068	0.001028
	Est	0.00265	0.001308	0.00167	0.001314	0.001547	0.00177	0.001772
cpcs179	**Exact**	0.02386	0.002798	0.00204	0.002215	0.002479	0.00198	-
	Est	0.02277	0.002591	0.00146	0.001814	0.002229	0.00263	-
cpcs360b	**Exact**	0.00109	0.000178	0.00014	0.000189	0.000152	0.00025	0.000372
	Est	0.00282	0.000432	0.00049	0.000396	0.000402	0.0006	0.000719
cpcs422b	**Exact**	-	-	0.00028	0.000206	0.00032	0.00041	0.000345
	Est	-	-	0.00076	0.000642	0.000575	0.00064	0.000632
grid	**Exact**	0.00108	0.000992	0.00119	0.000909	0.000986	0.00109	0.001126
15x30	**Est**	0.00248	0.002145	0.00247	0.00205	0.002254	0.00222	0.002386
random	**Exact**	0.00091	0.000392	0.00039	0.000549	0.000644	0.00062	0.001024
N=200	**Est**	0.00199	0.000799	0.00089	0.001075	0.00124	0.00171	0.001996
2layer	**Exact**	0.00436	0.000655	0.00063	0.000815	0.001171	0.00134	0.001969
N=200	**Est**	0.00944	0.001445	0.00144	0.001846	0.002354	0.00302	0.003414
coding	**Exact**	-	0.00014	0.00019	0.000189	0.000174	-	-
N=200	**Est**	-	0.000296	0.00035	0.000336	0.000356	-	-

Figure 2.15: Average absolute error (exact) and estimated confi dence interval (est.) as a function of w.

Further, our empirical study conclusively indicates that in most cases cycle-cutset makes a good sampling set. It is not always the best, but usually it is comparable to the best w-cutset results for some w and maybe practical when we do not want to invest time in fi nding an optimal w-cutset.

Finally, the comparison of estimated confi dence interval for approximate posterior marginals to the exact absolute error indicates that confi dence interval accurately reflects the absolute error (most of the time, within the factor of 2-3 and, in case of cutset sampling, never exceeding the range of $[0, 0.003]$) and could be used as a measure of sampling algorithm performance.

2.11 Conclusions

In this chapter, we defi ned a general scheme for collapsing Gibbs sampler in Bayesian networks (see Section 2.6) named w-cutset sampling combining sampling and exact inference. We showed theoretically and empirically cutset sampling: (1) improves convergence rate due to sampling from lower-dimensional space

and (2) allows sampling from non-ergodic network that have ergodic subspace. Using the induced width w of the graph to control the complexity of exact inference and the notion of w-cutset, w-cutset sampling provides a mechanism for balancing the sampling and the exact inference to optimize the performance of the algorithm.

We investigated empirically the performance of w-cutset sampling as a function of the adjusted induce width parameter w. Our experiments over a range of randomly generated and real benchmarks demonstrate the power of the cutset sampling idea and in particular show that an optimal balance between inference and sampling benefits substantially from restricting the cutset size, even at the cost of more complex inference. Thus, user can control the trade-offs between sampling and inference by examining the w-cutset sampling for different w values. In this chapter, we have proposed several greedy schemes for finding a w-cutset for cutset sampling. This problem is investigated further in chapter 3 .

We examined the efficiency of Gibbs sampler when the sampling set is a cycle-cutset and, more generally, when the sampling set is a w-cutset of the network defined as a subset of variables such that, when instantiated, the induced width of the network is $\leq w$. We demonstrate empirically for several networks that we can compute a new sample faster using cutset sampling scheme than standard Gibbs sampler. Most importantly, fewer samples are needed for convergence.

The w-cutset sampling scheme is a simple yet powerful extension of sampling in Bayesian networks that is likely to dominate regular sampling for any sampling method. While we focused on Gibbs sampling, other sampling techniques, with better convergence characteristics, can be implemented with cutset sampling as long as they permit to exploit Bayesian network structure in a similar manner.

The research results presented in this chapter have been published in [Bidyuk & Dechter 2003b] and [Bidyuk & Dechter 2003a].

2.12 Future Work

So far we have investigated only one approach for reducing sampling variance, namely collapsing the sampling space. We know that grouping of the variables (sampling several of the variables simultaneously) can provide additional improvement in the convergence of Gibbs sampler [J. Liu & Kong 1994; Jensen, Kong, & Kjaerulff 1995]. When the collapsing and grouping of sampling space is no longer feasible and convergence rate of the Markov chain is still too slow, we may need to use other variance reduction methods. Therefore, the central objective is to improve efficiency of the sampling algorithms for Bayesian networks by exploiting the network structure. Additional reduction in sampling variance and increase in convergence speed can be expected from using additional memory structures to store and group intermediate results. In chapter 5, we explore theoretical foundations for several variance reduction schemes and their implications for Bayesian network sampling.

Beyond the problem if reducing sampling variance and improving convergence speed, we are interested in exploring different combinations of exact and approximate inference methods. For example, for large values of w, where the exact inference in the w-cutset sampling is no longer feasible, we can resort to approximate inference via Iterative Belief Updating (investigated in chapter 4). Other possibilities include combining exact conditioning and sampling. We explore those and other possibilities for hybrid inference methods in the chapter 5.

Chapter 3

On finding minimal w-cutset problem

The complexity of a reasoning task over a graphical model is tied to the induced width of the underlying graph. It is well-known that conditioning (assigning values) on a subset of variables yields a subproblem of the reduced complexity where instantiated variables are removed. If the assigned variables constitute a cycle-cutset, the rest of the network is singly-connected and therefore can be solved by linear propagation algorithms. A w-cutset is a generalization of a cycle-cutset defined as a subset of nodes such that the subgraph with cutset nodes removed has induced-width of w or less. In this chapter, we address the problem of finding a minimal w-cutset in a graph. We relate the problem to that of finding the minimal w-cutset of a tree-decomposition. The latter can be mapped to the well-known *set multi-cover* problem. This relationship yields a proof of NP-completeness on one hand and a greedy algorithm for finding a w-cutset of a tree decomposition on the other. Empirical evaluation of the algorithms is presented.

3.1 Introduction

A cycle-cutset of an undirected graph is a subset of nodes that, when removed, the graph is cycle-free. Thus, if the assigned variables constitute a cycle-cutset, the rest of the network is singly-connected and can be solved by linear propagation algorithms. This principle is at heart of the well-known cycle-cutset conditioning algorithms for Bayesian networks [Pearl 1988] and for constraint networks [Dechter 1990]. Recently, the idea of cutset-conditioning was extended to accommodate search on any subset of variables using the notion of w-cutset, yielding a hybrid algorithmic scheme of conditioning and inference paramterized by w [Rish & Dechter 2000]. The w-cutset is defined as a subset of nodes in the graph that, once removed, the graph has tree-width of w or less.

The hybrid *w-cutset-conditioning* algorithm applies search to the cutset variables and exact inference (e.g., bucket elimination [Dechter 1999]) to the remaining network. *Given a w-cutset C_w, the algorithm is space exponential in w and time exponential in $w + |C_w|$* [Dechter 2001]. The scheme was applied successfully in the context of satisfiability [Rish & Dechter 2000] and constraint optimization [Larossa & Dechter 2003]. More recently, the notion of conditioning was explored for speeding up sampling algorithms in Bayesian networks in a scheme called *cutset-sampling*. The idea is to restrict sampling to w-cutset variables only (peform inference on the rest) and thus reduce the sampling variance ([Bidyuk & Dechter 2003a; 2003b]).

Since the processing time of both search-based and sampling-based schemes grows with the size of the w-cutset it calls for a secondary optimization task for finding a minimal-size w-cutset. Also, of interest is the task of finding the full sequence of minimal w-cutsets, where w ranges from 1 to the problem's induced-width (or tree-width), so that the user can select the w that fits his/her resources. We call the former the *w-cutset problem* and the latter *the sequence w-cutset problem*. The w-cutset problem extends the task of finding minimum cycle-cutset (e.g. a 1-cutset), a problem that received fair amount of attention [Becker & Geiger 1996; Becker, Bar-Yehuda, & Geiger 1999; Vazirani 2001].

The paper addresses the minimum size w-cutset and, more generally, the minimum weight w-cutset problem. First, we relate the size of a w-cutset of a graph to its tree-width and the properties of its tree-decompositions. Then, we prove that the problem of finding a minimal w-cutset of a given tree decomposition is NP-complete by reduction from the *set multi-cover* problem [Vazirani 2001]. Consequently, we apply a well-known greedy algorithm (GWC) for set multi-cover problem to solve the minimum w-cutset problem. The algorithm finds w-cutset within $O(1 + \ln m)$ of optimal where m is the maximum number of clusters of size greater than $w + 1$ sharing the same variable in the input tree decomposition. We investigate its performance empirically and show that, with rare exceptions, GWC and its variants find a smaller w-cutset than the well-performing MGA cycle-cutset algorithm [Becker & Geiger 1996] (adapted to the w-cutset problem)

and a w-cutset algorithm (DGR) proposed in [Geigher & Fishelson 2003].

3.1.1 w-cutset of a graph

DEFINITION **3.1.1 (w-cutset of a graph)** *Given a graph $G=<X,E>$, $C_w \subset X$ is a* w-cutset *of G if the subgraph over $X \backslash C$ has tree-width $\leq w$. Clearly, a w-cutset is also a w'-cutset when $w' \geq w$. The cutset C_w is* minimal *if no w-cutset of smaller size exists.*

For completeness, we also define the weighted w-cutset problem that generalizes minimum w-cutset problem (where all node weights are assumed the same). For example, in w-cutset conditioning, the space requirements of exact inference is $O(d_{max}^w)$ where d_{max} is the maximum node domain size in graph G. The total time required to condition on w-cutset C is $O(d_{max}^w) \times |D(C)|$ where $|D(C)|$ is the size of the cutset domain space. The upper bound value $d_{max}^{|C|}$ on $|D(C)|$ produces a bound on the computation time of the cutset-conditioning algorithm: $O(d_{max}^w) \times d_{max}^{|C|} = O(d_{max}^{w+|C|})$. In this case, clearly, we want to minimize the size of C. However, a more refined optimization task is to minimize the actual value of $|D(C)|$:

$$|D(C)| = \prod_{C_i \in C} |D(C_i)|$$

Since the minimum of $|D(C)|$ corresponds to the minimum of $\lg(|D(C)|)$, we can solve this optimization task by assigning each node X_i cost $c_i = \lg |D(X_i)|$ and minimizing the cost of cutset:

$$cost(C) = \lg |D(C)| = \sum_{C_i \in C} \lg |D(X_i)| = \sum_i c_i$$

Similar considerations apply in case of the w-cutset sampling algorithm. Here, the space requirements for the exact inference are the same. The time required to sample a node $C_i \in C$ is $O(d_{max}^w) \times |D(C_i)|$. The total sampling time is $O(d_{max}^w) \times \sum_{C_i \in C} |D(C_i)|$. To minimize the total processing time, we assign each node X_i cost $c_i = |D(X_i)|$ and select the w-cutset of minimum cost:

$$cost(C) = \sum_{C_i \in C} |D(C_i)|$$

DEFINITION **3.1.2 (weighted w-cutset of a graph)** *Given a reasoning problem $<X,F>$ where each node $X_i \in X$ has associated $cost(X_i) = c_i$, the cost of a w-cutset C_w is given by: $cost(C_w) = \sum_{X_i \in C_w} c_i$. The minimum weight w-cutset problem is to find a min-cost w-cutset.*

In practice, we can often assume that all nodes have the same cost and solve the easier minimal w-cutset problem which is our focus here. In section 3.2, we establish relations between the size of w-cutset of a graph and the width of its tree-decomposition. In section 3.3, we show that the problem is NP-hard even when finding a minimum w-cutset of a chordal graph (corresponding to a tree-decomposition of a graph).

3.2 w-Cutset and Tree-Decompositions

In this section, we explore relationship between w-cutset of a graph and its tree-decomposition.

THEOREM **3.2.1** *Given a graph $G=<X,E>$, if G has a w-cutset C_w, then there is a tree-decomposition of G having a tree-width $tw \leq |C_w|+w$.*

Proof. If there exists a w-cutset C_w, then we can remove C_w from the graph yielding, by definition, a subgraph G' over $X \backslash C_w$ that has a tree decomposition T with clusters of size at most w+1. We can add the set C_w to each cluster of T yielding a tree-decomposition with clusters of size at most $w+1+|C_w|$ and tree-width $w+|C_w|$. ■

We can conclude therefore that for any graph $tw^* \leq |C_i| + i$ for every i. Moreover,

THEOREM **3.2.2** *Given a graph G, if c_i^* is the size of a smallest i-cutset C_i^*, and tw^* is its tree-width, then:*

$$c_1^* + 1 \geq c_2^* + 2 \geq \ldots \geq c_i^* + i \geq \ldots \geq tw^* \quad (3.1)$$

Proof. Let us define $\Delta_{i,i+1} = c_i^* - c_{i+1}^*$, then we claim that $\Delta_{i,i+1} \geq 1$. Assume to the contrary that $c_i = c_{i+1}$, that is $D_{i,i+1}$=0. Since C_i^* is an i-cutset, we can build a tree decomposition T with maximum cluster size (i+1). Pick some $X_j \in C_i^*$ and add X_j to every cluster yielding tree decomposition T' with maximum cluster size (i+2). Clearly, $C_i^* \backslash X_j$ is an (i+1)-cutset of size $c_i^* - 1 = c_{i+1}^* - 1$ which contradicts the minimality of C_{i+1}^*. ■

Given a graph $G = (V,E)$, the *w-cutset sequence* problem seeks a sequence of minimal j-cutsets where j ranges from 1 to the graph's tree-width: $C_1^*, ..., C_j^*, ..., C_{tw^*} = \phi$. Let C'_w be a subset-minimal w-cutset,

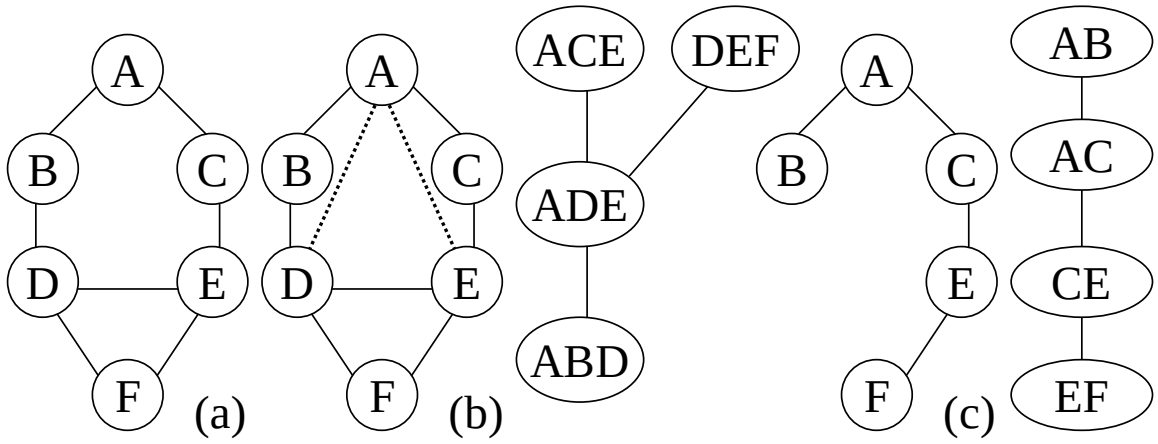


Figure 3.1: (a) Graph; (b) triangulated graph and corresponding tree decomposition of width 2; (c) graph with 1-cutset node $\{D\}$ removed and corresponding tree-decomposition.

namely one that does not contain another w-cutset. If we have a w-cutset sequence, we can reason about which w to choose for applying the w-cutset conditioning algorithm or w-cutset sampling. Given a w-cutset sequence we define a function $f(i) = |C_i| + i$ where i ranges from 1 to tw. This function characterizes the complexity of the w-cutset conditioning algorithms where for each i, the space complexity is exponential in i and the time complexity is exponential in $f(i)$. The time-complexity suggests operating with i as large as possible while space consideration suggests selecting i as small as possible. Notice that for various intervals of i, $f(i)$ is constant, if $|C_i| = |C_{i+1}| + 1$. Thus, given a w-cutset sequence, we have that whenever $f(i) = f(i+1)$, then $w = i$ is preferred over $w = i+1$. Alternatively, given a bound on the space-complexity expressed by r, we can select a most preferred w_p-cutset such that:

$$w_p(r) = \arg\min_j \{r = f(j)\}$$

In the empirical section 3.5, we demonstrate the analysis of function $f(i)$ and its implications.

THEOREM **3.2.3** *Given a tree-decomposition T=(V, E) where V=$\{V_1, ..., V_t\}$ is the set of clusters and given a constant w, a minimum w-cutset C^*_w of G satisfies:*

$$|C^*_w| \leq \sum_{i,|V_i|>w+1} (|V_i| - (w+1)) \tag{3.2}$$

Proof. From each cluster $V_i \in V$ of size larger than (w+1), select a subset of nodes $C_i \subset V_i$ of size $|C_i| = |V_i| - (w+1)$ so that $|V_i \backslash C_i| \leq w+1$.
Let $C_w = \cup_{i,|V_i|>w+1} C_i$.
By construction, C_w is a w-cutset of G and:
$c^*_w \leq |C_w| = |\cup_i C_i| \leq \sum_i |C_i| = \sum_{i,|V_i|>w+1} |V_i| - (w+1)$. ■

Since a w-cutset yields a tree-decomposition having $tw = w$, it looks reasonable when seeking w-cutset to start from a good tree-decomposition and find its w-cutset (or a sequence). In particular, this avoids the need to test if a graph has $tw = w$. This task is equivalent to finding a w-cutset of a chordal (triangulated) graph.

DEFINITION **3.2.1 (A w-cutset of a tree-decomposition)** *Given a tree decomposition T=$<V, E>$ of a reasoning problem $<X, F>$ where V is a set of subsets of X then $C^T_w \subset X$ is a* w-cutset *relative to T if for every i, $|V_i \backslash C^T_w| \leq w+1$.*

We should note upfront, however, that a minimum-size w-cutset of T (even if T is optimal) is not necessarily a minimum w-cutset of G.

Example 3.2.4 *Consider a graph in Figure 3.1(a). An optimal tree decomposition of width 2 is shown in Figure 3.1(b). This tree-decomposition clearly does not have a 1-cutset of size < 2. However, the graph has a 1-cutset of size 1, $\{D\}$, as shown in Figure 3.1(c).*

On the other hand, given a minimum w-cutset, removing the w-cutset from the graph yields a graph having $tw^* = w$. Because, otherwise, there exists a tree-decomposition over $X \backslash C_w$ having $tw < w$. Select such a tree and select a node in C_w that can be added to the tree-decomposition without increasing its tree-width beyond w. Such a node must exist, contradicting the minimality of C_w.

It is still an open question if every minimal w-cutset is a w-cutset of some minimum-width tree-decomposition of G.

3.3 Hardness of w-Cutset on Cluster-Tree

While it is obvious that the general w-cutset problem is NP-complete (1-cutset is a cycle-cutset known to be NP-complete), it is not clear that the same holds relative to a given tree-decomposition. We now show that, given a tree-decomposition T of a hyper-graph $\mathcal{H}$, the w-cutset problem for T is NP-complete.We use a reduction from *set multi-cover* (SMC) problem.

DEFINITION **3.3.1 (Set Cover (SC))** *Given a pair $< U, S >$ where U is universal set and S is a set of subsets S=$\{S_1, ..., S_m\}$ of U, find a minimum set $C \subset S$ s.t. each element of U is covered at least once: $\cup_{S_i \in C} S_i = U$.*

DEFINITION **3.3.2 (Set Multi-Cover(SMC))** *Given a pair $< U, S >$ where U is universal set and S is a set of subsets $S = \{S_1, ..., S_m\}$ of U, find a minimum cost set $C \subset S$ s.t. each $U_i \in U$ is covered at least $r_i > 0$ times by C.*

The SC is an instance of SMC problem when $\forall i, r_i = 1$.

THEOREM **3.3.1 (NP-completeness)** *The problem "*Given a tree-decomposition T=$<V, E>$ and a constant k, does there exist a w-cutset of T of size at most k ?*" is NP-complete.*

Proof. Given a tree decomposition T=$<V, E>$ over X and a subset of nodes $C \in X$, we can verify in linear time whether C is a w-cutset of T by checking if $\forall V_i \in V$, $|V_i \backslash C| \leq w + 1$. Now, we show that the problem is NP-hard by reduction from set multi-cover.

Assume we are given a set multi-cover problem $<U, S>$, where U=$\{X_1, ..., X_n\}$ and S=$\{S_1, ..., S_m\}$, a covering requirement $r_i > 0$ for each $U_i \in U$.

We define a cluster tree T=$<V, E>$ over S where there is a node $V_i \in V$ corresponding to each variable U_i in U that contains all subsets $S_j \in S$ that cover node X_i: $V_i = \{S_j \in S | X_i \in S_j\}$. Additionally, there is a node $V_S \in V$ that contains all subsets in S: $V_S = S$. Thus, $V = \{V_i | U_i \in U\} \cup V_S$. Denote $|V_i| = f_i$. The edges are added between each cluster $V_{i, i \neq s}$ and cluster V_S: $E = \{V_i V_S | U_i \in U\}$ to satisfy running intersection property in T.

Define w+1=$|S| - \min_i r_i = m - \min_i r_i$. For each $V_{i, i \neq s}$, since $|V_i|$=$f_i \leq m$ and $r_i > 0$, then $f_i - r_i \leq m - \min_i r_i \leq w + 1$. Consequently, $f_i \leq r_i + w + 1$.

For each V_i s.t. $f_i < r_i + w + 1$, define Δ_i=$r_i + w + 1 - f_i$ and augment cluster V_i with a set of nodes $\mathcal{Q}_i = \{Q_i^1 ... Q_i^{\Delta_i}\}$ yielding a cluster $V_i' = V_i \cup \mathcal{Q}_i$ of size $|V_i'|$=f_i'=$r_i + w + 1$.

We will show now that a set multi-cover $<U, S>$ has a solution of size k iff there exists a w-cutset of augmented tree decomposition T'=$<V', E>$ of the same size.The augmented tree for sample SMC problem in Figure 3.2(a) is shown in Figure 3.2(b).

Let C be a set multi-cover solution of size k. Then, $\forall U_i \in U$, $|C \cap V_i'| \geq r_i$ which yields $|V_i' \backslash C| \leq |V_i'| - r_i = f_i' - r_i = w + 1$. Since $|C| \geq \min_i r_i$, then $|V_S \backslash C| \leq |V_S| - \min_i r_i = m - \min_i r_i = w + 1$. Therefore, C is a w-cutset of size k.

Let C_w be a w-cutset problem of size k. If C_w contains a node $Q_j \in \mathcal{Q}_i$, we can replace it with some node $S_p \in V_i'$ without increasing the size of the cutset. Thus, without loss of generality, we can assume $C_w \subset S$. For each V_i' corresponding to some $U_i \in U$, let $C_i = C_w \cap V_i'$. By definition of w-cutset, $|V_i' \backslash C_w| \leq w + 1$. Therefore, $|C_i| \geq |V_i'| - (w + 1) = f_i' - (w + 1) = r_i$. By definition, C_w is a cover for the given SMC problem.

Minimum w-cutset problem is NP-hard by reduction from set multi-cover and is verifiable in linear time. Therefore, minimum w-cutset problem is NP-complete. ■

Example 3.3.2 *Let us demonstrate those steps for the SMC problem with U=$\{U_1, U_2, U_3\}$ and S=$\{S_1, ..., S_5\}$ shown in Figure 3.2(a). Define T=$<V, E>$, V=$\{V_1, V_2, V_3, V_s\}$, over S:*
V_1=$\{S_1, S_2, S_3\}$, f_1=3, V_2=$\{S_3, S_4, S_5\}$, f_2=3,
V_3=$\{S_2, S_4, S_5\}$, f_3=3, V_S=$\{S_1, ..., S_5\}$, f_S=5.
Then, $w = |S| - 1 - \min_i r_i = 5 - 1 - 1 = 3$. *Augment:*
V_1: Δ_1=w+1+$r_1 - f_1$=4+2-3=3, Q_1=$\{Q_1^1, Q_1^2, Q_1^3\}$.

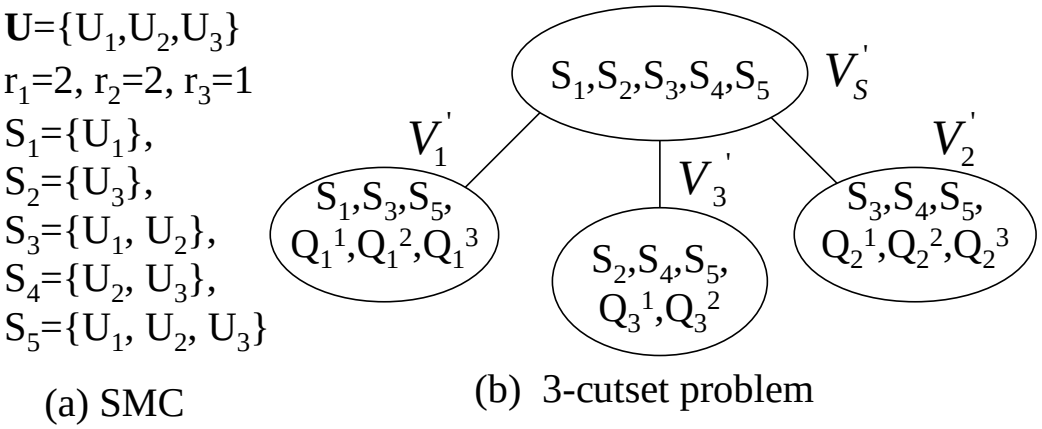


Figure 3.2: (a) A set multi-cover problem $<U, S>$ where U=$\{U_1, U_2, U_3\}$, S=$\{S_1, ..., S_5\}$, the covering requirements r_1=2, r_2=2, r_3=1. (b) Corresponding augmented tree decomposition T'=$<V', E>$ over S'=$\{S_1, ..., S_5, Q_1^1, Q_1^2, Q_1^3, Q_2^1, Q_2^2, Q_2^3, Q_3^1, Q_3^2\}$.

V_2: Δ_2=w+1+$r_2 - f_2$=4+2-3=3, Q_2=$\{Q_2^1, Q_2^2, Q_2^3\}$.
V_3: Δ_3=w+1+$r_3 - f_3$=4+1-3=2, Q_3=$\{Q_3^1, Q_3^2\}$.
The augmented tree decomposition T' is shown in Figure 3.2(b). Any SMC solution such as C=$\{S_3, S_5\}$ is a 3-cutset of T and vice versa.

In summary, we showed that when w is not a constant the w-cutset problem is NP-complete. This implies that the w-cutset sequence problem over tree-decompositions is hard.

3.3.1 An exact algorithm for minimal w-cutset of a tree-decomposition

Assume we are given a tree-decomposition T=$<V, E>$ over X. Assume that the tree-width of T is $w*$ and we are given a target bound value w. For each cluster V_i s.t. $|V_i| > w + 1$, defi ne $r_i = |V_i| - (w + 1)$. Let $C_w^*(T)$ denote minimum size w-cutset of T.

Before we defi ne an algorithm for fi nding an exact minimum size w-cutset, we will outline a few preprocessing steps that can be taken to solve the "trivial" part of the problem and reduce its complexity.

THEOREM **3.3.3** *Given a tree-decomposition T=$<V, E>$ over X of width $w*$ and a constant $w < w*$, assume there is a cluster $V_i \in V$ s.t. $|V_i| > w + 1$ and V_i is singly-connected in T. In other words, $r_i = |V_i| - (w + 1) > 0$ and V_i has only 1 neighbour V_j. Let S_{ij} be the separator between V_i and V_j: $S_{ij} = V_i \cap V_j$. If $|S_{ij}| \leq r_i$, then there is a minimum-size w-cutset $C_w^*(T)$ such that $S_{ij} \subset C_w^*(T)$.*

Proof. Assume some node $X_k \in S_{ij}$ is not in the minimum-size cutset. Then, $\exists X_q \in V_i$ s.t. $X_q \notin S_{ij}$ and $X_q \in C_w^*(T)$. Such a node must exist to satisfy r_i. If $X_q \notin S_{ij}$, then $\forall j \neq i$, $X_q \notin V_j$ and we can replace X_q with X_k in the cutset without increasing the size of the w-cutset. ■

As a consequence, whenever we have a singly-connected cluster V_i whose only separator $|S_{ij}| \leq r_i$, we can safely add to cutset any subset S_i of nodes in V_i such that $S_{ij} \subset S_i$ and $|S_i| = r_i$. Then, we remove from T all nodes in S_{ij} as well as the cluster V_i and obtain a tree T'=$<V', E'>$ where $V' = V \backslash S_{ij}$ where $C_w^*(T) = S_i \cup C_w^*(T')$. Thus, without loss of generality, we assume that for any singly-connected cluster V_i with some $r_i > 0$, its separator size $|S_{ij}| > r_i$.

A similar line of reasoning leads to the conclusion that given any singly-connected cluster V_i with neighbor V_j and their separator S_{ij} where $|S_{ij}| > r_i$, then there is a minimum size cutset $C_w^*(T)$ that contains r_i nodes from S_{ij}: $C_w^*(T) \cap V_i \subset S_{ij}$. Thus, in our search for a minimum w-cutset, we can limit the search for $C_w^*(T) \cap V_i$ to the subsets of S_{ij}. That is the main idea behind the proposed recursive algorithm: remove all those singly-connected clusters V_i that have $r_i \leq |S_{ij}|$, select one of the remaining sinlgy-connected clusters V_i that have $r_i > |S_{ij}|$ enumerate all possible subsets f_i of S_{ij} of size r_i and solve the w-cutset problem for each T' over $X' = X \backslash f_i$. The exact $MinSizeCutset(T, w)$ algorithm is given in Figure 3.3.

The maximum depth of the recursion of procedure $MinSizeCutset(T, w)$ equals the number of clusters in T whose size $> w + 1$. At each recursion iteration our state-space is multiplied by $|F_i| = (|S_{ij}|, r_i)$. Since

MinSizeCutset(T,w)
1. Remove all singly-connected clusters of size $\leq w+1$.

2. Solve all singly-connected clusters V_i s.t. $r_i \geq S_{ij}$ where S_{ij} is a separator between V_i and its neighbor V_j.

3. Select a singly-connected cluster V_i. Let S_{ij} be a separator between V_i and its neighbor V_j. Let F_i be a set of all possible subsets of size r_i from S_{ij}.
FOR EACH $f_i \in F_i$ DO
$T' = T \backslash f_i$
$\overline{f}_i = MinSizeCutset(T', w)$
END FOR
Let $m = \arg\min_{f_i \in F_i} |\overline{f}_i|$.
Return $C^*_w(T) = f_m \cup \overline{f}_m$.

Figure 3.3: Recursive minimum size w-cutset algorithm.

$|S_{ij}| \leq w*$, then:

$$\begin{aligned} |F_i| \leq (w*, r_i) &= \frac{w*!}{(w*-r_i)!r_i!} \\ &= w^*(w^*-1)...(w^*-r_i+1) \\ &\leq (w^*)^{\sum_i r_i} \end{aligned}$$

The total size of state-space explored is bounded by:

$$\prod_{i=1}^{|V|}(w^*)^{r_i} = (w^*)^{\sum_i r_i}$$

If we a looking to answer the decision problem *"Does the tree-decomposition T has a w-cutset of size k ?"*, we can stop the recursion when sum of the $r_i's$ of the processed clusters reaches k. Then, the total number of states explored is bounded by $(w^*)^k$.

3.4 Algorithm GWC for minimum cost w-cutset

Next, we show that the problem of finding w-cutset can be mapped to that of finding set multi-cover. The mapping suggests an application of greedy approximation algorithm for set multi-cover problem to find w-cutset of a tree decomposition. When applied to a tree decomposition T=<V, E> over X, it is guaranteed to find a solution within factor $O(1+\ln m)$ of optimal where m is the maximum # of clusters of size $> (w+1)$ sharing the same node. To avoid loss of generality, we consider the weighted version of each problem.

The mapping is as follows. Given any w-cutset problem of a tree-decomposition T=<V, E> over X, each cluster node $V_i \in V$ of the tree becomes a node of universal set U. A covering set S_{X_j}=$\{V_i \in V | X_j \in V_i\}$ is created for each node $X_j \in X$. The cost of S_{X_j} equals the cost of X_j. The cover requirement is $r_i = |V_i| - (w+1)$. Covering a node in SMC with a set S_{X_j} corresponds to removing node X_j from each cluster in T. Then, the solution to a set multi-cover is a w-cutset of T. Let C be a solution to the SMC problem. For each $U_i \in U$, the set C contains at least r_i subsets S_{X_j} that contain U_i. Consequently, since $U_i = V_i$, then $|V_i \cap C| \geq r_i$ and $|V_i \backslash C| \leq |V_i| - r_i = |V_i| - |V_i| + (w+1) = w+1$. By definition, C is a w-cutset. An example is shown in Figure 3.4. This duality is important because the properties of SC and SMC problems are well studied and any algorithms previously developed for SMC can be applied to solve w-cutset problem.

A well-known polynomial time greedy algorithm exists for weighted SMC [Vazirani 2001] that chooses repeatedly set S_i that covers the most "live" (covered less than r_i times) nodes f_i at the cost c_i: a set that minimizes the ratio c_i/f_i. In the context of w-cutset, f_i is the number of clusters whose size still exceeds $(w+1)$ and c_i is the cost of node X_i. As discussed earlier, c_i maybe defined as the size of the domain of node X_i or its log. When applied to solve the w-cutset problem, we will refer to the algorithm as GWC (Greedy W-Cutset). It is formally defined in Figure 3.5. We define here the approximation algorithm metrics:

DEFINITION **3.4.1 (factor δ approximation)** *An algorithm $\mathcal{A}$ is a* factor δ, $\delta > 0$, *approximation algorithm for minimization problem $\mathcal{P}$ if $\mathcal{A}$ is polynomial and for every instance $I \in D_{\mathcal{P}}$ it produces a solution s such that: $cost(s) \leq \delta * cost_{OPT}(s), \delta > 1$.*

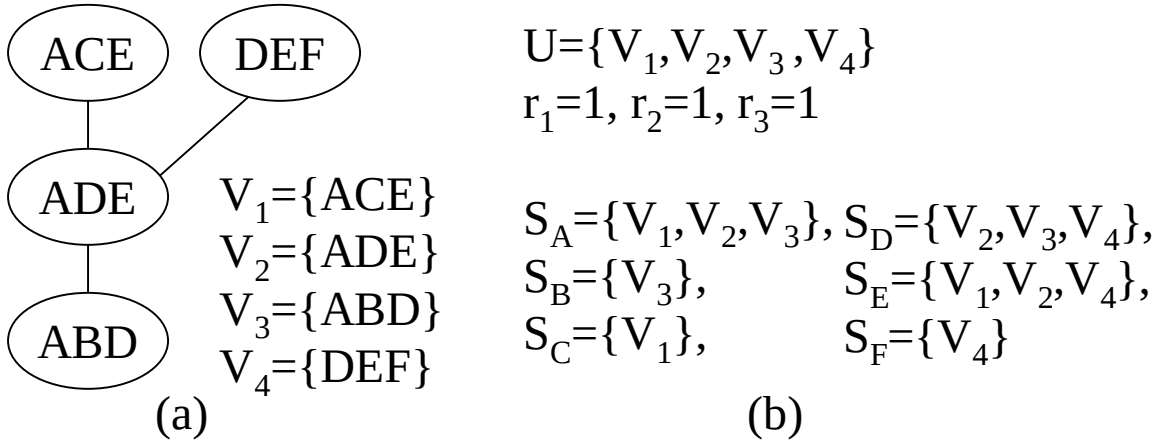


Figure 3.4: (a) A tree decomposition T=<V,E> where V=$\{V_1, ..., V_4\}$ over X=$\{A, B, C, D, E, F\}$; (b) the corresponding set multi-cover problem <U,S> where U=$\{V_1, V_2, V_3, V_4\}$ and S=$\{S_A, S_B, S_C, S_D, S_F\}$; here, set S_{X_i} contains a cluster V_j iff $X_i \in V_j$. The 1-cutset of T is a solution to the set multicover with covering requirements r_1=r_2=r_3=r_4=1: when node $V_i \in V$ is "covered" by set S_{X_i}, node X_i is removed from each cluster.

Greedy w-Cutset Algorithm (GWC)
Input: A set of clusters $V = \{V_1, ..., V_m\}$ of a tree-decomposition over $X = \{X_1, ..., X_n\}$ where $\forall V_i \in V$, $V_i \subset X$; the cost of each node X_i is c_i.
Output : A set $C \subset X$ s.t. $|V_i \backslash C| \leq w$.
Set $C = \emptyset$,t=0.
While $\exists V_i$ s.t. $|V_i| > w$ **do**
1. $\forall X_i \in X$, compute $f_i = |\{V_j\}|$ s.t. $|V_j| > w$ and $X_i \in V_j$.
2. Find node $X_i \in X$ that minimizes the ratio c_i / f_i.
3. Remove X_i from all clusters: $\forall V_i \in V, V_i = V_i \backslash X_i$.
4. Set $X = X \backslash X_i$, $C = C \cup \{X_i\}$.
End While
Return C

Figure 3.5: Greedy w-cuset Algorithm.

GWC is a factor $O(1 + \ln m)$ approximation algorithm [Rajagopalan & Vazirani 1998] where m is the maximum number of clusters sharing the same node (same as the maximum set size in SMC).

This bound is nearly the best possible for a polynomial algorithm due to strong inapproximability results for the set cover problem, the special case of set multi-cover problem. Approximation guarantee better than $O(\ln m)$ is not possible for any polynomial time algorithm unless P=NP [Bellare *et al.* 1993; Lund & Yannakakis 1994]. Furthermore, $\exists C, \Delta_0$ s.t. for all $\Delta \geq \Delta_0$ no polynomial-time algorithm can approximate the optimum within a factor of $\ln \Delta - C \ln \ln \Delta$ unless P=NP [Trevisan 2001].

3.5 Experiments

We use Bayesian networks as input reasoning problems. In all experiments, we started with a moral graph G of a Bayesian network $\mathcal{B}$ which was used as the input to the minimal w-cutset problem. The tree-decomposition of G was obtained using min-fi ll algorithm [Kjaerulff 1990].

Our benchmarks are two CPCS networks from UAI repository: cpcs360b with N=360 nodes and induced width $w*$=22 and cpcs422b with N=422 nodes and induced width $w*$=27, one instance each. Our other benchmarks are layered random networks, meaning that each node is assigned a set of parents selected randomly from previous layer. One set of random networks consisted of 4 layers of $L = 50$ nodes each, total of N=50x4=200 nodes, each node assigned $P = 3$ parents. The second set of random networks consisted of 8 layers of $L = 25$ nodes each, total of N=25x8=200 nodes, each node assigned $P = 3$ parents. For random networks, the results are averaged over 100 instances. We compare the performance of two greedy heuristic algorithms- MGA (Modifi ed Greedy Algorithm due to [Becker & Geiger 1996]) and DGR (Deter-

ministic Greedy Algorithm due to [Geigher & Fishelson 2003])- to our proposed algorithms: GWC (Greedy W-Cutset) and its variants.

The MGA algorithm is adapted from minimum cost cycle-cutset algorithm of [Becker & Geiger 1996] that iteratively removes all singly-connected nodes from the graph and adds to cutset the node that minimizes cost to degree ratio. The algorithm stops when remaining subgraph is cycle-free. The MGA is a factor 4 approximation algorithm. In [Vazirani 2001], a factor 2 approximation algorithm is defi ned based on layering. However, it can not be easily adapted to fi nding minimal w-cutset for $w > 1$. For MGA, the only modifi cation required to fi nd w-cutset is to stop when original graph with cutset nodes removed can be decomposed into a cluster tree of width w or less (using min-fi ll heuristics). In our implementation, MGA algorithm uses the GWC heuristics to break ties: if two nodes have the same degree, the node found in most of the clusters of size $> w$ is added to the cutset.

The DGR algorithm is the Deterministic Greedy Algorithm for fi nding an elimination order of the variables that yields a tree-decomposition of bounded width defi ned in [Geigher & Fishelson 2003]. DGR obtains a w-cutset while computing the elimination order of the variables. When eliminating some node X yields a cluster that is too large (size $> w + 1$), the algorithm uses greedy heuristics to pick a cutset node among all the nodes that are not in the ordering yet. Specifi cally, the deterministic algorithm adds to the cutset a node X that maximizes expression $\sqrt{|N_X|}C_X$, where N_X is a set of neighbours of X that are not eliminated yet and $C_X = \prod_{U_i \in N_X} |D(U_i)|$. As we ignore domain sizes in this empirical study, we defi ned $C_X = |N_X|$ in which case DGR adds to cutset a node of maximum degree in the subgraph over nodes that are not eliminated.

The GWC algorithm was implemented as described earlier picking at each iteration a node found in most clusters of size $> w + 1$ with a secondary heuristics (tie breaking) that selects the node contained in most of the clusters. Several variants of GWC with different tie breaking heuristics were tested that were allowed to rebuild a tree decomposition after removing a cutset node:

GWCA - breaks ties by selecting the node found in most of the clusters of the tree decomposition;
GWCM - breaks ties by selecting the node found in most of the clusters of maximum size;
GWCD - breaks ties by selecting the node of highest degree (the degree of the node is computed on the subgraph with all cutset nodes removed and all resulting singly-connected nodes removed). Note that GWC and GWCA only differ in that GWCA rebuilds a cluster-tree after removing a cutset node. Also note that MGA and GWCD have their primary and tie-breaking heuristics switched.

Table 3.1: w-cutset. Networks: I=cpcs360b, II=cpcs422b, III=4-layer random networks, L=50, N=200, P=3; IV =8-layer random networks, L=25, N=200, P=3.

	w	1	2	3	4	5	6	7	8	9	10
I	MGA	30	22	20	18	16	15	14	13	12	**10**
w*=20	DGR	36	22	19	18	16	**14**	**13**	**12**	**11**	**10**
	GWC	**27**	**20**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**
	GWCA	**27**	21	18	**16**	**15**	**14**	**13**	**12**	**11**	**10**
	GWCD	**27**	21	18	**16**	**15**	**14**	**13**	**12**	**11**	**10**
II	MGA	80	70	65	60	54	49	44	41	38	36
w*=22	DGR	84	70	63	54	49	43	38	32	27	23
	GWC	**78**	66	58	52	46	41	36	31	26	22
	GWCA	**78**	**65**	**57**	**51**	**45**	**40**	**35**	**30**	**25**	**21**
	GWCD	**78**	**65**	**57**	**51**	**45**	**40**	**35**	**30**	**25**	**21**
III	MGA	87	59	54	52	50	48	47	45	44	43
w*=49	DGR	80	57	52	50	48	46	44	43	42	40
	GWC	78	61	53	49	46	44	43	42	41	39
	GWCA	**74**	**56**	50	**47**	**44**	**42**	**41**	**39**	**38**	**37**
	GWCD	**74**	**56**	**49**	**47**	**44**	**42**	**41**	**39**	**38**	**37**
IV	MGA	99	74	69	66	63	61	59	56	54	51
w*=24	DGR	90	71	65	61	58	55	52	49	47	44
	GWC	93	77	68	63	59	55	52	49	46	43
	GWCA	87	**70**	**62**	**57**	**54**	**51**	**48**	**45**	**42**	**39**
	GWCD	**86**	**70**	**62**	**57**	**54**	**51**	**48**	**45**	**42**	**39**

	w	11	12	13	14	15	16	17	18	19	20
I	MGA	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
w*=20	DGR	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	GWC	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	GWCA	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	GWCD	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
II	MGA	33	30	28	**9**	**8**	7	6	5	4	**2**
w*=22	DGR	21	19	16	**9**	**8**	7	**5**	**4**	**3**	**2**
	GWC	19	16	13	10	**8**	**6**	**5**	**4**	**3**	**2**
	GWCA	**18**	**15**	**12**	**9**	**8**	**6**	**5**	**4**	**3**	**2**
	GWCD	**18**	**15**	**12**	**9**	**8**	**6**	**5**	**4**	**3**	**2**
III	MGA	41	40	39	37	36	35	34	33	31	30
w*=49	DGR	39	38	36	36	34	33	32	31	30	29
	GWC	38	37	36	35	34	33	32	31	30	29
	GWCA	**36**	35	**34**	**33**	**32**	**31**	**30**	**29**	**28**	**27**
	GWCD	**36**	**34**	**34**	**33**	**32**	**31**	**30**	**29**	**28**	**27**
IV	MGA	49	47	44	41	39	36	34	31	28	26
w*=24	DGR	41	38	36	33	31	28	25	23	21	19
	GWC	40	37	35	32	29	27	25	23	20	18
	GWCA	**37**	**34**	**32**	**30**	**27**	**25**	**23**	**21**	**19**	**17**
	GWCD	**37**	35	**32**	**30**	28	**25**	24	**21**	**19**	**17**

The results are presented in Table 3.1. For each benchmark, the table provides the fi ve rows of results corresponding to the fi ve algorithms (labelled in the second column). Columns 3-12 are the w-cutset sizes for the w-value. The upper half of the table entires provides results for w in range $[1, 10]$; the lower half of the

table provides results for w in range $[11, 20]$. The results for cpcs360b and cpcs422b correspond to a single instance of each network. The result for random networks are averaged over 100 instances. The best entries for each w are highlighted.

As Table 3.1 shows, it pays to rebuild a tree decomposition: with rare exceptions, GWCA finds a cutset as small as GWC or smaller. On average, GWCA, GWCM, and GWCD computed the same-size w-cutsets. The results for GWCM are omitted since they do not vary sufficiently from the others.

The performance of MGA algorithm appears to depend on the network structure. In case of cpcs360b, it computes the same size w-cutset as GWC variants for $w \geq 10$. However, in the instance of cpcs422b, MGA consistently finds larger cutsets except for w=20. On average, as reflected in the results for random networks, MGA finds larger cutset than DGR or any of the GWC-family algorithms. In turn, DGR occasionally finds a smaller cutset compared to GWC, but always a larger cutset compared to GWCA and GWCD.

We measured the GWC algorithm approximation parameter M in all of our benchmarks. In cpcs360b and cpcs422b we have $M = 86$ and $M = 87$ yielding approximation factor of $1 + \ln M \approx 5.4$. In random networks, M varied from 29 to 47 yielding approximation factor $\in [4.3, 4.9]$. Thus, if C is the w-cutset obtained by GWC and C_{opt} is the minimum size w-cutset, then on average:

$$\frac{|C|}{|C_{opt}|} \leq 5$$

Looking at the results as solutions to the sequence w-cutset problems, we can inspect the sequence and suggest good w's by analysing the function $f(i) = |C_i| + i$ as described in section 3.2. To illustrate this we focus on algorithm GWCA for CPC364, CPCS424 and 4-layer random networks (See Table 3.2).

Table 3.2: Function $f(i)$ for i=1...16, GWCA. Networks: I=cpcs360b, II=cpcs422b, III=4-layer random, L=50, N=200, P=3.

	f(i)									
i	1	2	3	4	5	6	7	8	9	10
I	28	23	21	20	20	20	20	20	20	20
II	79	67	60	55	50	46	42	38	34	31
III	75	57	53	51	49	48	48	47	47	47

	f(i)									
i	11	12	13	14	15	16	17	18	19	20
I	20	20	20	20	20	20	20	20	20	20
II	29	27	25	23	23	22	22	22	22	22
III	47	47	47	47	47	47	47	47	47	47

For cpcs360b we observe a small range of values for $f(i)$, namely $f(i) \in \{20, 21, 23, 28\}$. In this case the point of choice is $w = 4$ because $f(1) = 28$, $f(2) = 23$, $f(3) = 21$ while at $i = 4$ we obtain reduction $f(4) = 20$ which stays constant for $i \geq 4$. Therefore, we can have the same time complexity for w-cuset as for exact inference ($w* = 20$) while saving a lot in space, reducing space complexity from exponential in 20 to exponential in 4 only. For w-cutset sampling this implies sampling 20 variables (out of 360) and for each variable doing inference exponential in 4.

The results are even more interesting for cpcs422b where we see a fast decline in time complexity with relatively slow decline in space complexity for the range $i = 1, ..., 11$. The decline is more moderate for $i \geq 11$ but is still cost-effective: for $i = 16$ we get the same time performance as $i = 20$ and therefore $i = 16$ represents a more cost-effective point.

Finally, for the case of 4-layer random networks, on average the function $f(i)$ decreases for $i = 1...8$ and then remains constant. This suggests that if space complexity allows, the best point of operation is $w = 8$.

3.6 Related Work and Conclusions

In this chapter, we formally defined the minimal w-cutset problem applicable to any reasoning problem with graphical model such as constraint networks and Bayesian networks. The minimum w-cutset problem extends the minimum cycle-cutset problem corresponding to $w = 1$. The motivation for finding a minimal w-cutset is to bound the space complexity of the problem (exponential in the width of the graph) while minimizing the required additional processing time (exponential in the width of the graph plus the size of cutset). The cycle-cutset problem corresponds to the well-known weighted vertex-feedback set problem and can be approximated within factor 2 of optimal by a polynomial algorithm. We show that the minimal w-cutset problem is harder by reduction from the set multi-cover problem [Vazirani 2001]: the set multi-cover problem, and subsequently the w-cutset problem, cannot have constant-factor polynomial approximation algorithm unless P=NP. Empirically, we show that the minimal cycle-cutset heuristics based on the degree of a node is not competitive with the tree-decomposition of the graph.

To our knowledge, only heursitics related to the node elimination order were used before in finding a w-cutset. In [Rish & Dechter 2000; Larossa & Dechter 2003] and [Geigher & Fishelson 2003], the w-cutset is obtained while computing elimination order of the nodes. The next elimination node is added to the cutset in [Rish & Dechter 2000; Larossa & Dechter 2003] if its bucket size exceeds the limit. A similar approach was explored in [Geigher & Fishelson 2003] in DGR algorithm (presented in the empirical section) except that the cutset node was chosen heuristically among all the nodes that were not eliminated yet. The immediate drawback of either approach is that it does not permit to change the order of the nodes already eliminated. As the empirical results demonstrate, DGR usually finds smaller cutset than MGA but bigger than GWC/GWCA/GWCD.

The research results presented in this chapter have been accepted for publication in [Bidyuk & Dechter 2004].

3.7 Future Work

The main objective of our future work is to find good heuristics for w-cutset problem that are independent from tree-decomposition of a graph since the minimal w-cutset of a tree-decomposition provides only an upper bound on the minimal w-cutset of a graph. So far, we only looked at the degree of the node as possible heuristics and found emprically that GWC heuristics are usually superior. There are also open questions remaining regarding the relationship between w-cutset of a graph and a w-cutset of its tree-decomposition. As mentioned earlier, the w-cutset of a tree decomposition of a graph only provides an upper bound on the optimal w-cutset of the graph and it is not clear, for example, whether the minimal w-cutset of a graph is a w-cutset of one of its minimum width tree-decompositions.

Chapter 4

Epsilon-cutset effect on Iterative Belief Propagation

This chapter investigates the behavior of iterative belief propagation algorithm (IBP) in Bayesian networks with loops. In a multiply-connected network, IBP is only guaranteed to converge in linear time to the correct posterior marginals when evidence nodes form a loop-cutset. We propose an ϵ-cutset criteria that IBP will converge and compute posterior marginals close to correct when a single value in the domain of each loop-cutset node receives very strong support compared to other values thus producing an effect similar to the observed loop-cutset. We investigate the support for this criteria analytically and empirically and show that it is consistent with previous observations of IBP performance in multiply-connected networks.

4.1 Introduction

The paper investigates the correctness of iterative belief propagation (IBP) algorithm, also known as sum-product algorithm, in Bayesian networks with loops. Pearl [Pearl 1988] proposed iterative belief propagation algorithm for singly connected Bayesian networks and proved that the algorithm converges in number of iterations equals to the diameter of the network to the correct posterior values. Iterative belief propagation can be applied to networks with loops to derive approximate inference where exact methods such as loop-cutset conditioning, bucket-elimination, and tree-clustering [Pearl 1988; Dechter 1999; Lauritzen & Spiegelhalter 1988] become impractical due to exponential growth in time and memory as network width increases. In general, IBP does not always converge and does not produce correct posterior values for Bayesian networks with loops.

However, empirically it was demonstrated that IBP can be successfully applied to several classes of Bayesian networks with loops used in practical applications, especially for coding networks [R.J. McEliece & Cheng 1997; Kschischang & Frey 1998; Frey & MacKay 1997], where it was shown to outperform variational decoder [Frey & MacKay 1997] and mini-bucket approximation algorithm [I. Rish & Dechter 1998]. It also performs well on noisy-or networks (used in diagnostics) and pyramid networks (used in image recognition) [K. P. Murphy & Jordan 1999].

We now have gained better understanding of IBP behavior in networks with a single loop. Weiss [Weiss 2000] proved using the Markov network model that IBP always converges on a single-loop networks and defined the error in posterior marginals obtained by IBP as a function of eigenvalues of a matrix computed from the conditional probability tables of all the variables in a loop. Also, for a single loop networks, he established the correlation between the accuracy of posterior marginals computed by IBP and convergence rate. That is, the faster IBP converges, the more accurate the posterior marginals are.

The behavior of IBP in loopy networks of arbitrary topology so far is best understood in the context of minimizing system free energy. It was proved that in general case, stationary points of belief propagation algorithm correspond to the local minima of the Bethe free energy of the system [Yedidia, Freeman, & Weiss 2001]. That was an important result from which a modified version of belief propagation algorithm was derived, the generalized belief propagation, that minimizes the Kikuchi free energy [Yedidia, Freeman, & Weiss 2001]. Based on the same result, Welling and Teh proposed a method for optimizing Bethe free energy directly for binary networks.

Still, we lack the general criteria that would enable us to predict convergence of IBP algorithm for a given network and to evaluate the quality of posterior beliefs computed by IBP without computing exact posterior beliefs. While the relationship between IBP algorithm and free energy functions is important one, it does not provide us with any means of evaluating the quality of answers produced by IBP in general. Even for

a single-loop network, the parameters used by Weiss [Weiss 2000] to describe the convergence rate of IBP and to predict the error are only available after we have executed the algorithm and stored information from previous iterations.

We investigate the accuracy of IBP in directed Bayesian networks as a function of loop size, CPT values, prior beliefs, and evidence support. We derive anlytical expression for an error value in the posterior marginal belief computed by IBP in a single-loop Bayesian network without evidence (see section 4.6). For this special case, we conclusively prove from the derived expression that accuracy of IBP improves as:

1. Size of the loop increases.
2. Prior belief for a single value of root node approaches 1.

Those results are supported by empirical study of average error in posterior marginal beliefs of single-loop networks as a function of loop size and root priors (see section refsec:evidence).

For general class of directed Bayesian networks, we defi ne a concept of ϵ-support for a node X in a Bayesian network. We say that Bayesian network has ϵ-cutset when all of its loop-cutset receive ϵ-support. We prove (in section 4.5) that in a Bayesian network with ϵ-cutset, posterior marginals computed by IBP for a node X $P'(x|e)$ will converge to the correct posterior marginal $P(x|e)$ as ϵ converges to 0. A single loop without evidence and with extreme root priors is a special case of Bayesian network with ϵ-cutset. Thus, our results for a single-loop network from section 4.6 are a special case of ϵ-cutset effect.

Our theoretical results regarding ϵ-cutset effect on the accuracy of IBP are supported by the empirical evidence on 2-layer Noisy-Or and random Noisy-Or networks (see section 4.8).

4.2 Background

DEFINITION **4.2.1 (graph concepts)** *A* directed graph *is a pair $G = \{V, E\}$, where $V = \{X_1, ..., X_n\}$ is a set of nodes, or variables, and $E = \{(X_i, X_j)|X_i, X_j \in V\}$ is the set of edges. Given $(X_i, X_j) \in E$, X_i is called a* parent *of X_j, and X_j is called a* child *of X_i. The set of X_i's parents is denoted $pa(X_i)$, or pa_i, while the set of X_i's children is denoted $ch(X_i)$, or ch_i. The family of X_i includes X_i and its parents.*

The underlying graph G of a directed graph D is the undirected graph formed by ignoring the directions of the edges in D.

A node X in a directed graph D is called a root *if no edges are directed into X. A node X in a directed graph D is called a* leaf *if all of its adjacent edges are directed into X. A* cycle *in G is a path whose two end-points coincide. A* cycle-cutset *of undirected graph G is a set of vertices that contains at least one node in each cycle in G. A* loop *in D is a subgraph of D whose underlying graph is a cycle. A vertex v is a* sink *with respect to loop $\mathcal{L}$ if the two edges adjacent to v in $\mathcal{L}$ are directed into v. A vertex that is not a sink with respect to a loop $\mathcal{L}$ is called an* allowed *vertex with respect to $\mathcal{L}$. A* loop-cutset *of a directed graph D is a set of vertices that contains at least one allowed vertex with respect to each loop in D. (We borrowed loop-cutset definition from [Becker, Bar-Yehuda, & Geiger 1999]).*

A directed graph is acyclic if it has no directed *cycles. A graph is* singly connected *(also called a* polytree*), if its underlying undirected graph has no cycles. Otherwise, it is called* multiply connected*.*

DEFINITION **4.2.2 (belief networks)** *Let $X = \{X_1, ..., X_n\}$ be a set of random variables over multi-valued domains $D_1, ..., D_n$. A* belief network (BN) *is a pair (G, P) where G is a directed acyclic graph on X and $P = \{P(X_i|pa_i)|i = 1, ..., n\}$ is the set of conditional probability matrices associated with each X_i. An assignment ($X_1 = x_1, ..., X_n = x_n$) can be abbreviated as $x = (x_1, ..., x_n)$. The BN represents a joint probability distribution $P(x_1, ..., x_n) = \Pi_{i=1}^{n} P(x_i|x_{pa(X_i)})$, where x_S is the projection of vector x on a subset of variables S. An evidence E is an instantiated subset of variables.*

DEFINITION **4.2.3 (d-separation)** *If X, Y, and Z are three disjoint subsets of nodes in a DAG G, then Z is said to d-separate X from Y, denoted $d(X, Z, Y)_G$, if and only if there is not path from a node in X to a node in Y along which the following two conditions hold: (1) every node with converging arrows either is or has a descendant in Z, and (2) every other node is outside Z. A path satisfying the conditions above is to be* active*; otherwise it is said to be* blocked *(by Z). By path we mean a sequence of consecutive edges (of any directionality) in the DAG.*

4.3 Iterative Belief Propagation Algorithm

Iterative Belief Propagation (IBP) algorithm computes posterior belief $P(x|e)$, where E is set of evidence nodes, for every variable X in a Bayesian network. It applies Pearl's belief propagation algorithm [Pearl 1988], developed for singly-connected networks, to a multiply-connected networks, ignoring cycles. Belief is propagated by sending messages between the nodes:

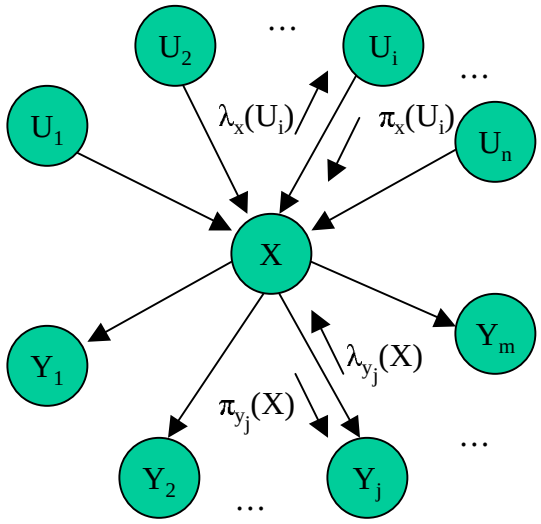


Figure 4.1: Flow of messages between variables X and its parent U_i and child Y_j during belief propagation.

Iterative Belief Propagation (IBP)
Input: A Belief Network $BN = \{P_1, ..., P_n\}$, evidence e, an activation schedule A, the number of iterations I.
Output: Belief in every variable.

1. **Initialize** λ and π.
For evidence node $x_i = j$, set $\lambda_{x_i}(k)$ to 1 for $j = k$ and to 0 for $j \neq k$.
For node x having no parents set π_x to prior $P(x)$.
Otherwise, set λ_x and π_x to $(1, ..., 1)$.

2. **For iterations** 1 to I:
For each node x along A, do:
/* α is a normalization constant */
• For each $x = j$, compute $\lambda_{x,u_i}(j) = \alpha \sum_j \lambda_x(j) \sum_{u_l : l \neq i} P(x = j \mid u_1, ..., u_m) \prod_{l \neq i} \pi_{u_l,x}$
• For each $x = j$, compute $\pi_{x,y_i}(j) = \alpha \prod_{k \neq i} \lambda_{y_k,x}(j) \sum_{u_l} P(x = j \mid u_1, ..., u_m) \prod_l \pi_{u_l,x}$.

3. **Belief update:** For each x along A
• For each $x = j$, compute
$\lambda_x(j) = \prod_i \lambda_{y_i,x}(j)$, where y_i are x's children.
• For each $x = k$, compute
$\pi_x(j) = \sum_{u_1,...,u_m} P(x = j \mid u_1, ..., u_m) \prod_i \pi_{u_i,x}$,
where u_i are x's parents.
• Compute $BEL(x) = \alpha \lambda_x \pi_x$.

Figure 4.2: Iterative Belief Propagation (IBP) Algorithm

During each iteration $(t+1)$, each node X sends *causal* support messages $\pi_{Y_j}^{(t+1)}(x)$ to each child Y_j (see figure 4.1):

$$\pi_{Y_j}^{(t+1)}(x) = \alpha \lambda_X(x) \prod_{k \neq j} \lambda_{Y_k}^{(t)}(x) \sum_u P(x|u) \prod_i \pi_X^{(t)}(u_i) \tag{4.1}$$

and *diagnostic* support messages $\lambda_x^t(u_i)$ to each parent U_i:

$$\lambda_X^{(t+1)}(u_i) = \alpha \sum_x \lambda_X(x) \prod_j \lambda_{Y_j}^{(t)}(x) \sum_{u_k, k \neq i} P(x|u) \prod_{k \neq i} \pi_X^{(t)}(u_k) \tag{4.2}$$

The flow of messages between node X and its parent U_i and child Y_j is shown in figure 4.1. Node X sends message $\lambda_X(U_i)$ to its parent U_i and message $\pi_{Y_j}(X)$ to its child Y_j. In turn, parent U_i sends message $\pi_X(U_i)$ to X and child Y_j sends message $\lambda_{Y_j}(X)$ to X. Messages exchanged between X and U_i are vectors of dimension equal to the domain size of U_i: $|\lambda_X(U_i)| = |\pi_X(U_i)| = |D(U_i)|$. Message exchanged between X and Y_j are vectors of dimension equal to the domain size of X: $|\lambda_{Y_j}(X)| = |\pi_{Y_j}(X)| = |D(X)|$.

In equations 4.1 and 4.2, α is a normalization constant such that $\sum_{u_i} \lambda_X(u_i) = 1$ and $\sum_X \pi_{Y_j}(X) = 1$. Message $\lambda_X(X)$ is introduced to incorporate evidence information into the equation (similar to [K. P. Murphy & Jordan 1999]). If node X is not observed, $\forall x_k \in D_X, \lambda_X(X = x_k) = 1$. If node X is observed and x_e is the evidence value, then $\forall x_k \in D_x, x_k \neq x_e, \lambda_X(X = x_k) = 0$ and $\lambda_X(x = x_e) = 1$.

The posterior belief is computed for each node X by combining $\pi_X(u_i)$ messages received from its parents u_i and $\lambda_{Y_j}(X)$ messages received from its children:

$$P^{(t)}(x) = \alpha\lambda_X(x) \prod_j \lambda^{(t)}_{Y_j}(x) \sum_u P(x|u) \prod_i \pi^{(t)}_X(u_i) \tag{4.3}$$

It is convenient to use the following notation in the context of belief propagation algorithm:

$$\lambda^{(t)}(x) = \lambda_X(x) \prod_j \lambda^{(t)}_{Y_j}(x) \tag{4.4}$$

$$\pi^{(t)}(x) = \sum_u P(x|u) \prod_i \pi^{(t)}_X(u_i) \tag{4.5}$$

Namely, causal supports from all parents and diagnostic support from all children are combined into vectors π_x and λ_x, respectively.

Then, we can rewrite the definition of a posterior belief for a varaible X after iteration t as follows:

$$P^{(t)}(x) = \alpha\lambda^{(t)}(X)\pi^{(t)}(X) \tag{4.6}$$

Normalization of π and λ messages is recommended to avoid numerical underflow although it does not affect the computation of the posterior beliefs. Namely, we can show that $P^n(X|E) = K * P^{un}(X|E)$ where $P^n(X|E)$ is a belief in X computed using normalized messages, $P^{un}(X|E)$ computed without normalization, and K is constant independent on the value of X, which will disappear by normalization (see appendix 1).

An *activation schedule* (variable ordering) specifies the order in which the nodes are processed (activated). After all nodes are processed, the next iteration of belief propagation begins, updating the messages computed during the previous iteration. Algorithm IBP(t) stops after t iterations. Figure 4.2 summarizes the algorithm.

4.4 IBP, loop-cutset and irrelevant subnetworks

In this section, we demonstrate that IBP *automatically* exploits conditional independence introduced by observed nodes and automatically ignores irrelevant portions of the network that contain no evidence.

Utilizing the notion of conditional independence, we can often reduce a seemingly complex Bayesian network to a singly-connected one. It is well-known that if evidence nodes form a loop-cutset, then we can transform multiply-connected Bayesian network to an equivalent singly-connected network which can be solved by belief propagation, leading to the loop-cutset conditioning method [Pearl 1988].

Also, unobserved nodes that have only unobserved descendents are irrelevant to the beliefs of the remaining nodes and therefore, processing can be restricted to the relevant subgraphs.

Combining the above two properties, it is clear that if evidence nodes constitute a loop-cutset of a relevant subgraph of a query node X, then its posterior belief can be computed by applying belief-propagation only to the relevant subgraph which can be *transformed* into a singly-connected network.

We show in subsections 4.4.1 and 4.4.2 that IBP exploits the above two properties, of observed and unobserved nodes, *automatically*, without requiring any outside action for network transformation. As a result, the correctness and convergence of IBP on node X in a multiply-connected Bayesian network will be determined by the structure of the relevant subgraph of node X as opposed to the structure of a complete network. If the relevant subnetwork of node X is singly-connected relative to the evidence, IBP will converge to correct posterior marginals for node X.

While these conclusions are fairly straightforward, we believe they deserve to be stated towards the understanding of IBP's boundaries as an approximate scheme.

In section 4.5, we will discuss the effect of increasing the level of support for a single value of a variable (for example, from its observed children) in the approximation of the conditional independence, effectively weakening dependence between the node's children and parents. We further argue that in a multiply-connected Bayesian network, as each loop-cutset node receives stronger support for one value, the closer the approximation to the loop-cutset condition becomes, leading to the improvements in IBP convergence and accuracy.

4.4.1 Evidence nodes

An observed node X in a Bayesian network G blocks the path between its parents and its children as defi ned in d-separation criteria. In other words, it creates conditional independence between its parents and its children.

In this section we establish that IBP automatically exploits evidence nodes blocking information flow between their parents and their children. Namely, messages that an observed node X sends to its children are independent from any messages that node X receives. Similarly, message that observed node X sends to its parents are independent from messages that node X receives from its children.

LEMMA **4.4.1 (Information flow blocking)** *Let X be an observed node in a Bayesian network G. Then for any child Y_j of node X, the $BEL(Y_j)$ computed by IBP is not dependent on the messages that X receives from its parents U_i or the messages that node X receives from its other children $Y_k, k \neq j$.* ■

Lemma 4.4.1 allows us to understand fully the behavior of IBP in a Bayesian network where observed nodes form a loop-cutset.

THEOREM **4.4.1 (IBP on loop-cutset)** *If evidence nodes constitute a loop-cutset, then IBP converges to the correct posteriors in linear time.* ■

The proof for lemma 4.4.1 and theorem 4.4.1 is given in appendix 2.

4.4.2 Irrelevant nodes

Unobserved nodes that have only unobserved descendents are irrelevant to the beliefs of the remaining nodes and therefore, processing can be restricted to the relevant subgraphs. In IBP, this property is expressed by the fact that irrelevant nodes (that are not observed and do not have observed descendents) send diagnostic support messages that equally support each value in the domain of a parent and thus do not affect the computation of marginal posteriors of its parents.

LEMMA **4.4.2** *Let X be a node in a loopy Bayesian network G such that it is not observed and it does not have observed descendents and let G' be a subnetwork obtained by removing node X and its descendents from the network. Then for any $Y \in G'$, the posterior belief of Y as computed by IBP over G is identical to the posterior belief of Y computed by IBP applied to G' only.* ■

We can immediately conclude that in a loopy network without evidence, IBP will always converge after 1 iteration because only propagation of π messages affects the computation of posterior beliefs and π messages do not change. Also in that case, IBP converges to the correct marginals for any node whose parents do not have common ancestors. This is because the relevant subnetwork that contains only the node and its ancestors is singly-connected. Therefore, by lemma 4.4.2 they are the same as the posterior marginals computed by applying IBP to the complete network. In summary,

THEOREM **4.4.2 (Irrelevant unobserved nodes)** *Let G be a Bayesian network and let G' be a network obtained by recursively eliminating all its unobserved leaf nodes. If observed nodes in G constitute a loop-cutset of G', then for all the nodes in G', IBP applied to G converges to the correct posterior marginals. For a node outside G', IBP will converge (not necessarily to correct posteriors) only if it converges for all of its parents.* ■

Lemma 4.4.2 and theorem 4.4.2 are proved in appendix 2.

THEOREM **4.4.3 (Corollary)** *If Bayesian network does not contain any observed nodes or only has observed root nodes, then IBP always converges.*

Corollary follows from theorem 4.4.2 is that IBP is guaranteed to converge for any Bayesian network that has no observed nodes or that has the observed root nodes only.

4.5 ϵ-cutset effect

Consider Bayesian network B. Let X be a node in B. Let $P_{IBP}(x|e)$ represent posterior beliefs computed by IBP. Let $P(x|e)$ denote correct posterior beliefs. As noted above, in multiply-connected networks, IBP is only guaranteed to converge to the correct posterior marginals for all nodes if evidence nodes form a loop-cutset.

Let $C = \{C_1, C_2, ...C_k\}$ be a subset of nodes that form loop-cutset in B. An observed loop-cutset node C_i effectively breaks the loop by blocking the flow of information between its parents and its children. Now, assume that node C_i is not observed but receives a very strong support for one value in its domain c_i' due to a strong prior for x' or strong evidential support form a child node, or a combination of those. We will call it ϵ-support. Then, when node C_i has strong ϵ support, we intuitively expect it to have an effect similar to

observed nodes. When all nodes in C have ϵ-support, we could expect IBP to compute values $P_{IBP}(x|e)$ that would be close to $P_{IBP}(x|e,c')$; at the same time, we could expect that $P(x|e,c')$ is close to $P(x|e)$. Now, let us formally defi ne ϵ-support criteria and ϵ-cutset.

DEFINITION **4.5.1 (ϵ-support)** *Let G be a Bayesian network. Let X be a node in a Bayesian network. Let $\pi(x)$ denote messages X sends to its children. Let $\lambda_x(pa(x))$ denote messages X sends to its parents. Let $\lambda'_x(pa(x))$ denote messages X would send if it was observed given the same input messages from its neighbours. We will say that we have ϵ-support for node X in G when following conditions hold: $\exists x \in D(X)$ such that*

$$P_{IBP}(x|e) > 1-\epsilon$$
$$\forall j, \pi_j(x) > 1-\epsilon$$
$$\exists j, s.t. \lambda_j(x) > 1-\epsilon$$

C is an ϵ-cutset of $\mathcal{B}$, if C is a loop-cutset and every $C_i \in C$ has ϵ-support in a value $c'_i \in D(C_i)$. Denote by $\{c'_1, c'_2, ...c'_k\}$ the tuple of corresponding values.

DEFINITION **4.5.2 (ϵ-cutset)** *A subset of nodes C in Bayesian network $\mathcal{B}$ is an ϵ-cutset of $\mathcal{B}$ if C is a loop-cutset of network $\mathcal{B}$ and ϵ-support conditions are satisfied for each loop-cutset node $C_i \in C$.*

Intuitively, when a Bayesian network $\mathcal{B}$ has an ϵ-cutset, we expect that $P(x|e)$ will be close to $P(x|e,c)$ for small ϵ. We prove that in following theorem 4.5.1.

THEOREM **4.5.1 (Observed Cutset)** *Let $\mathcal{B}$ be a Bayesian network, and let E be a set of observed variables. Let $C = \{C_1, C_2, ..., C_k\}$ be a subset of variables such that $C \bigcup E$ constitute a loop-cutset of $\mathcal{B}$. Let X_i be a variable in the network. Let c_i^j represent a j-th value in the domain of variable C_i. Let X_i be a variable in the network. Given $\epsilon > 0$, if $\forall C_i \in C$, $\exists c'_i$ such that:*

$$P(c'_i|e) \geq 1-\epsilon_i \geq 1-\epsilon$$
$$0 \leq \epsilon_i \leq \epsilon$$

then

$$|P(x|e) - P(x|e,c')| \leq \sum_i \epsilon_i \leq k\epsilon$$

The proof is provided in Appendix 4.

We establsihed that when loop-cutset nodes $C_i \in C$ are observed, IBP algorithm computes correct posterior belief $P(x|e,c)$. We denote by $P'(x|e)$ valus computed by BP in a given network if convergence is achieved. Assuming that C is an ϵ-cutset with extreme in $C = c$, we conjecture that $P'(x|e)$ converges to $P(x|e)$ for every node $X \notin C \bigcup E$ as ϵ decreases.

CONJECTURE **4.5.1 (ϵ-cutset effect)** *If C is an ϵ-cutset of Bayesian network $(\mathcal{B})$, then approximate posterior probability distribution $P'(x|e)$ will converge to the correct posterior probability distribution $P(x|e)$ as ϵ decreases.*

In the next section, we derive, in theorem 4.6.1, an analytical expression for error δ in the approximate posterior belief $P'(d|e)$ of sink node D in a single-loop Bayesian network without evidence. In section 4.7, we prove an error bound on the ndoes in $\mathcal{B}$ as a function of ϵ which proves the conjecture above for Bayesian networks without evidence. In this case, root node A consitutes an ϵ-cutset when prior $P(a) > 1-\epsilon$ for some value a in domain $D(A)$. Our empirical evidence further supports the ϵ-cutset effect theory as we evaluate average error $\delta = |P(x|e) - P'(x|e)|$ as a function of ϵ in single-loop networks and random networks with ϵ-cutset in section 4.8.

4.6 Single Loop Bayesian Network without Evidence

In this section, we will analyze the performance of belief propagation algorithm in a single-loop Bayesian network with binary nodes as shown in fi g 4.3. Without evidence, the posterior marginals of all nodes except the sink node D will be computed correctly due to theorem 4.4.2. Thus, we focus here on computing the error produced by IBP in the posterior marginal values of sink node D. Let $G(D)$ be correct posterior belief:

$$G(D) = \sum_{B_n, C_m} P(D|B_n, C_m) \sum_A P(B_n|A) P(C_m|A) P(A) \tag{4.7}$$

In a Bayesian network without observed nodes, all λ messages sent from children to parents provide same support for all parent values (λ will be vectors with all values set to 1). Therefore, only π messages contribute to the computation of the marginal beliefs. If we create activation schedule such that parents of a node are

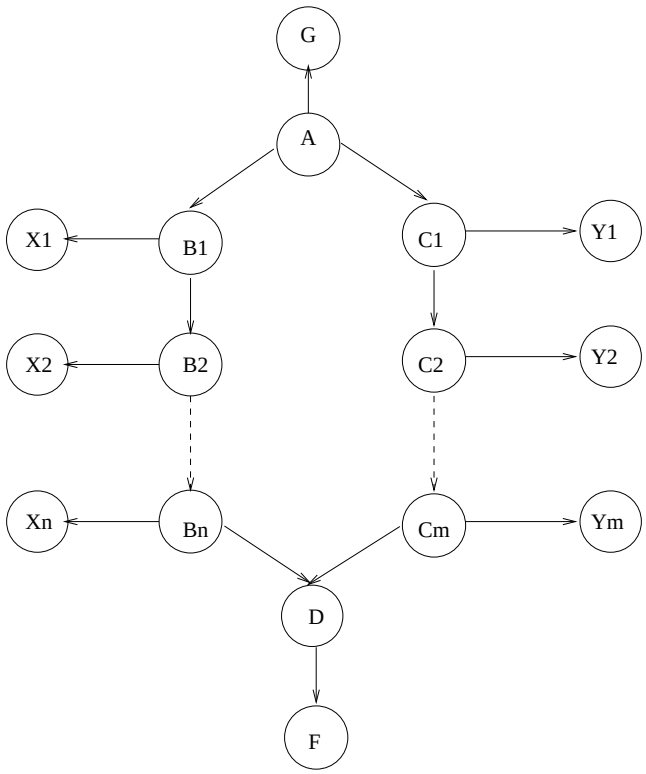


Figure 4.3: A single loop Bayesian network.

processed first, then one iteration of belief propagation is sufficient for convergence. The messages sent in future iterations would be identical to the messages sent during first iteration. Then, it is easy to derivethe posterior marginals of node D $G^*(D)$ computed by IBP:

$$G^*(D) = \alpha \sum_{B_n, C_m} P(D|B_n, C_m)\pi(B_n)\pi(C_m) =$$

$$= \alpha \sum_{B_n, C_m} P(D|B_n, C_m)(\sum_{A} P(B_n|A)P(A))(\sum_{A} P(C_m|A)P(A)) \tag{4.8}$$

where α is a normalization constant. We can show that $\alpha = 1$ when neither D nor any of its descendents are observed. In other words, $G^*(D)$ does not require normalization.

THEOREM **4.6.1** *Let G be a single-loop Bayesian network as shown in figure 4.3 Assume all nodes are binary and there are no observed nodes in the loop. Let us define:*

$$P(A) = (\epsilon, 1-\epsilon)$$

$$\beta_0^i = P(B_i = 0|B_{i-1} = 0), \beta_1^i = P(B_i = 0|B_{i-1} = 1)$$

$$\gamma_0^j = P(C_j = 0|C_{j-1} = 0), \gamma_1^j = P(C_j = 0|C_{j-1} = 1)$$

Then the error $\delta(D) = G^*(D) - G(D)$ in the posterior marginals of node D computed by Iterative Belief Propagation will be:

$$\delta(D) = (\epsilon - \epsilon^2)((-1)^{B_n}(-1)^{C_m} \sum_{B_n, C_m} P(D|B_n, C_m)) \prod_{i=1}^{n}(\beta_0^i - \beta_1^i) \prod_{j=1}^{m}(\gamma_0^j - \gamma_1^j) \tag{4.9}$$

■

We prove this theorem in Appendix 3. Let $d_{ij} = P(D = 0|B_n = i, C_m = j)$. Then, for D=0:

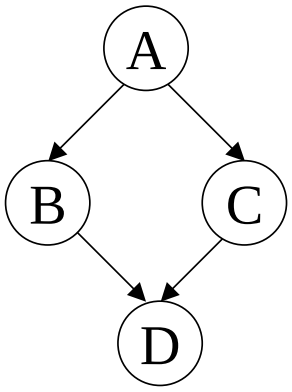
$$\delta(D) = \epsilon(1-\epsilon)(d_{00} - d_{01} - d_{10} + d_{11}) \prod_{i=1,n} (\beta_0^i - \beta_1^i) \prod_{j=1,m} (\gamma_0^j - \gamma_1^j) \tag{4.10}$$

From equation 4.10, it is clear that the accuracy of IBP improves:

1. As prior beliefs for a root node(s) approach boundary values: $\lim_{\epsilon \to 0,1} \delta = 0$
2. As λ support messages from the children of allowed nodes in the loop approach boundary values: $\lim_{\lambda(B),\lambda(C),\lambda(A) \to 0,1} \delta = 0$
3. As number of nodes in a loop increases: $\lim_{n \to \infty, m \to \infty} \delta = 0$
4. As conditional probabilities for the same value of X in different rows get closer:

$$\lim_{\beta_0^k - \beta_1^k \to 0, \gamma_0^k - \gamma_1^k \to 0} \delta = 0 \tag{4.11}$$

It is also easy to see that when one of the allowed nodes is observed, $\delta = 0$. When node A is initialized, either $\epsilon = 0$ or $(1 - \epsilon) = 0$. When one of the nodes B_i or C_j is observed, it is equivalent to having $\beta_0(B_i) = \beta_1(B_i) = 0|1$ or $\gamma_0(C_j) = \gamma_1(C_j) = 0|1$ yielding $\delta = 0$.



A	P(A)
0	0.5
1	0.5

A	P(B=0|A)	P(B=1|A)
0	0	1
1	1	0

A	P(C=0|A)	P(C=1|A)
0	0	1
1	1	0

BC	P(D=0|BC)	P(D=1|BC)
00	1	0
01	0	1
10	0	1
11	1	0

Result: D	P(D)	P'(D)
0	1	0.5
1	0	0.5
Error:	**0.5**	**0.5**

A	P(A)
0	0.5
1	0.5

A	P(B=0|A)	P(B=1|A)
0	0.1	0.9
1	0.9	0.1

A	P(C=0|A)	P(C=1|A)
0	0.1	0.9
1	0.9	0.1

BC	P(D=0|BC)	P(D=1|BC)
00	0.9	0.1
01	0.1	0.9
10	0.1	0.9
11	0.9	0.1

Result: D	P(D)	P'(D)
0	0.756	0.5
1	0.244	0.5
Error:	**0.256**	**0.256**

Figure 4.4: An example demonstrating the tightness of the worst-case bound $\delta(D) = 2\epsilon(1-\epsilon)$ on the absolute error $|P'(D) - P(D)|$, where $P'(D) = P_{ibp}(D)$ and $P(D)$ is exact. We show single loop Bayesian network (left), an instance of the network with deterministic CPTs where $\delta(D) = 2 * 0.5 * 0.5 = 0.5$ is exact; an instance of the network with relaxed CPTs where $\delta(D) = 0.256 << 0.5$.

In the worst case, when $d_{00} = d_{11} = 1$ and $d_{01} = d_{10} = 0$ and $\forall i, |\beta_0^i - \beta_1^i| = 1$ and $\forall j, |\gamma_0^j - \gamma_1^j| = 1$, the error bound obatined in theorem 4.6.1 evaluates to:

$$\delta(D) = 2\epsilon(1 - \epsilon)$$

Since the function $f(\epsilon) = \epsilon(1-\epsilon)$ reaches its maximum value of 0.25 at $\epsilon = 0.25$, the worst case error bound is 0.5. We simulate the worst case scenario above on the example of a minimal loop network consisting of four nodes as shown in Figure 4.6, left. Under the deterministic CPT assignments shown in Figure 4.6, middle, the exact absolute error in the posterior marginals of node D reaches the maximum value of 0.5.

Thus, the bound $2\epsilon(1 - \epsilon)$ is tight although in many instances the exact absolute error will be a lot smaller as any one small factor $|\beta_0^i - \beta_1^i|$ or $|\gamma_0^j - \gamma_1^j|$ or $|d_{00} - d_{01} - d_{10} + d_{11}|$ will reduce the bound on the error value. As we show in the Figure 4.6, right, relaxing the deterministic probabilities in the condition probability tables by replacing each 0 with 0.1 and, correspondingly, each 1 with 0.9, we obtain a much smaller absolute error 0.256.

The conditional probability table values for D can actually become the driving factor in the reduction of the error. It is easy to show that marginals obtained by IBP in absence of evidence are always bounded by $max_{pa(D)}P(D|pa(D))$ and $min_{pa(D)}P(D|pa(D))$:

$$min_{pa(D)}P(D|pa(D)) \leq P(D) \leq max_{pa(D)}P(D|pa(D))$$

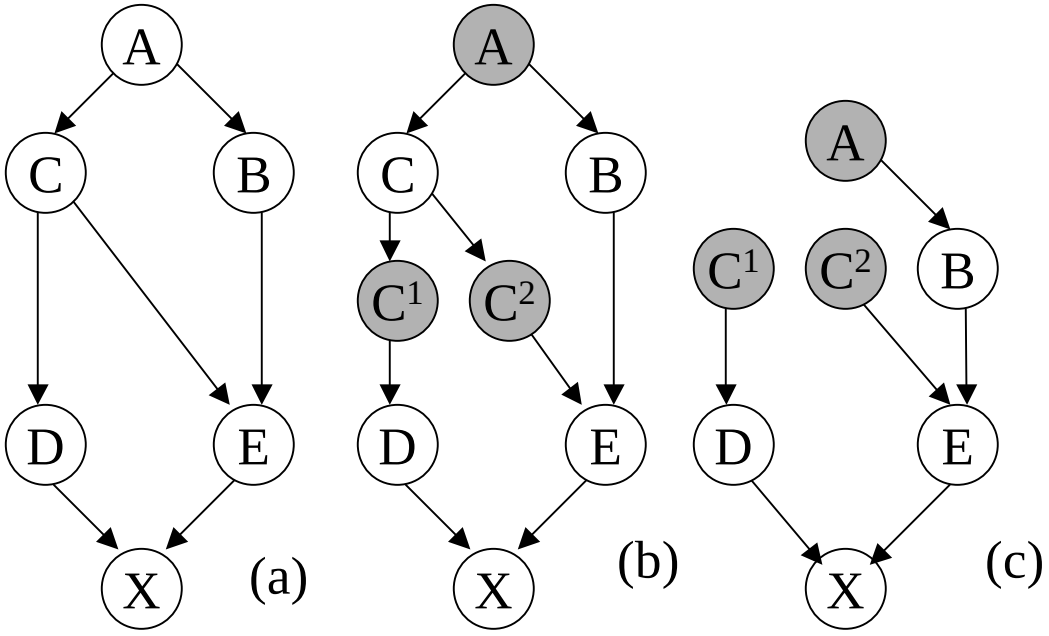


Figure 4.5: (a) A Bayesian network has a cycle-cutset consisting of node C. (b) An equivalent Bayesian network where two copies of C are created: C_1 and C_2. (c) The ancestor tree of node X over ancestor set $\{C_1, C_2, A\}$.

Thereofre, when the interval $[\min_{pa(D)} P(D|pa(D)), \max_{pa(D)} P(D|pa(D))$ is small, the $P_{ibp}(D)$ becomes relatively stable.

4.7 Bayesian Network Without Evidence

A more general result can be obtained for a Bayesian network without evidence and with bi-valued nodes. First, we give a recursive defi nition of the complete ancestor set of X.

DEFINITION **4.7.1 (complete set of ancestors)** *Parents of node X constitute a complete ancestor set of node X. If A=$\{A_1, ..., A_k\}$ is a* complete set of ancestors *of node X and P_i, $1 \leq i \leq k$, is a set of parents of node A_i, then $A' = (A \backslash A_i) \cup P_i$ is a complete ancestor set of X.*

DEFINITION **4.7.2 (ancestor subgraph)** *Let X be a node in graph G. Let A be a complete ancestor set of X. A subgraph containing node X, its complete ancestor set A, and every node on a directed path from each ancestor $A_i \in A$ to X and all the edges between those nodes is an* ancestor subgraph of X on A.

Now, we defi ne the transformation of the directed acycling graph G with respect to its cycle-cutset C. For every node $C_i \in C$ and for each child ch_j of C_i, create a new copy C_{ij} of C_i. Replace edge from C_i to ch_j with edge from C_{ij} to ch_j. Create edge from C_i to each C_{ij}. Defi ne a deterministic probability table for C_{ij} s.t. $P(C_{ij} = c|C_i = c) = 1$. It is easy to see that this transformation produces a new graph G' where all probabilities are preserved because $\forall j$, $P_{ibp}(C_i) = P(C_{ij})$. We will call G' a *cutset transformation* of G. The process is demonstrated in Figure 4.5a,b.

It is easy to see that C'=$\{C_{11}, ..., C_{1m_1}, C_{21}, ..., C_{2m_2}, ..., C_{k1}, ..., C_{km_k}\}$ is a cutset of G'. Any node X in G' has a complete set of ancestors A such that each $A_i \in A$ is either a root node or a node in cutset C' of G'. The proof is by construction. We start with an ancestor set consisting of parents of X. Replace each ancestor node that is not a cutset node and not a root node with its parents. Continue until no such nodes left. The cycle-cutset nodes will guarantee that we can obtain such ancestor set and that the ancestor subgraph of X over A is a tree. Proof by contradiction. Assume that a node Y in ancestor set of X is neither a cutset nor is a root node. Assume that we cannot replace node Y with its ancestor set without creating a cycle because one of its parents is already in the ancestor tree. That would indicate that there is a loop that does not cotain any of the cycle-cutset nodes which contradicts properties of the cycle-cutset. Thus, the ancestor subgraph of X over A is a tree and A contains only root nodes and cutset nodes in C'.

It is easier to analyze behavior of IBP in G' compared to G. Assume that C is an ϵ-cutset in G. Then $\forall C_i \in C \exists c'_i \in D(C_i)$ such that:

$$P_{ibp}(c'_i) > 1 - \epsilon_i$$

Let C' and G' be obtained from C and G using transformation above. Now, let A be a complete ancestor set of X in G where all nodes in A are either root nodes or are in C'. The $P_{ibp}(x)$ depends only on propagation of π-messages in the ancestor subtree of X over A. In absences of evidence, for root nodes $R_i \in A$, the $\pi(R_i) = P_{ibp}(R_i) = P(R_i)$. Then, we can obtain a closed form error bound on $P(X)$ in G' that is equivalent to the $P_{ibp}(X)$ in G. as expressed in the next theorem.

THEOREM **4.7.1 (IBP Error Bound)** *Given a Bayesian network $\mathcal{B}$ without evidence where all nodes have domain size 2 and given ϵ-cutset $\overline{C}$=$\{C_1, C_2, ..., C_k\}$ s.t $\forall C_i \in \overline{C}$:*

$$P(c_i) = (\epsilon_i, 1 - \epsilon_i)$$

$$P_{ibp}(c_i) = (\epsilon'_i, 1 - \epsilon'_i)$$

$$\epsilon_i - \epsilon'_i = \delta_i$$

Then, for any node X in $\mathcal{B}$:

$$|P(x) - P_{ibp}(x)| \leq 2 \sum_{c_i \in c} d_i \epsilon'_i (1 - \epsilon'_i) + \sum_{c_i \in c} d_i |\delta_i|$$

where d_i is the number of children of node $C_i \in C$.

The proof of the theorem is provided in Appendix 6. This theorem subsumes the single-loop case described above. Observing that in the instance of single loop $\delta_a = 0$ because $P(X) = \pi(A)$ bounding $(d_{00} - d_{01} - d_{10} + d_{11}) \leq 2$ and boudning the products of $\prod_i (\beta_0^i - \beta_1^i) \leq 1$ and $\prod_j (\gamma_0^j - \gamma_1^j) \leq 1$, we obtain for single loop with sink node D and a root node A with $P(A) = (\epsilon, 1 - \epsilon)$:

$$\delta_D \leq 2\epsilon(1 - \epsilon)$$

Theorem 4.7.1 above proves the ϵ-cutset conjecture for Bayesian networks without evidence with bi-valued nodes. The theorem bound is expressed via posterior marginals of cycle-cutset nodes computed by IBP and error $delta_i$ in each of those marginals. Each δ_i can be estimated recursively by arranging cutset nodes in topological order. Then:

$$\begin{aligned} \delta_1 &= 0 \\ \delta_2 &\leq 2d_1\epsilon'_1 + 3\delta_1 = 2d_1\epsilon'_1 \\ \delta_3 &\leq 2d_1\epsilon'_1 + 2d_2\epsilon'_2 + 3\delta_1 + 3\delta_2 = 7d_1\epsilon'_1 + 2d_2\epsilon'_2 \end{aligned}$$

And so on. This bound is likely to be impractical for empirical evaluation. However, it provides a theoretical foundation for ϵ-cutset effect in Bayesian networks without evidence as it clearly shows that as $\epsilon' \to 0$, then $\forall i$, $\delta_i \to 0$ and $\delta_x \to 0$. In general, we expect the error value δ_i to be small when ϵ'_i is small as demonstrated empirically in [Dechter & Mateescu 2003].

4.8 Empirical Results

In this section, we empirically investigate the accuracy and convergence of IBP in networks with loops. In all experiments, unless specifi ed otherwise, the conditional probabilities were represented by noisy-or:

$$P(Child = 0 | Parents) = e^{-\theta_0 - \sum_i \theta_i Parent_i} \tag{4.12}$$

where θ_0, the 'leak' term was fi xed at 0.005 and individual noise factors θ_i were chosen uniformly from the range [0,1]. The number of iterations for approximate inference was fi xed at 50: we established experimentally that 50 iterations was suffi cient for IBP to converge if it was going to converge at all. All experimental results are consistent with conclusions in sections 4.5, 4.6 (theorem 4.6.1), and 4.7 (theorem 4.7.1).

For each network used in empirical evaluation, we applied belief propagation algorithm to obtain approximate posterior marginals $P_{ibp}(x_i|e)$ and a variable elimination algorithm ([Kask *et al.* 2001; Dechter 2001]) to obtain exact posterior marginals $P(x_i|e)$. To evaluate the quality of approximations, we computed average absolute error.

4.8.1 Single Loop

The first set of experiments was focused on evaluating the performance of IBP on a small single-loop network with controlled parameters. Each network consisted of 1 root node A, 1 sink node D, and two directed pathes from root to node D: $\{A, B_1, ..., B_k, D\}$ and $\{A, C_1, ..., C_k, D\}$. Each loop node A, D, B_i, i=1...k, and C_j, j=1...k, was also assigned a child that was observed unless stated otherwise. Single loop networks (see fig 4.3) containing 4, 6, 8, and 10 nodes ($k \in [1, 2, 3, 4]$) were constructed.

The average absolute error in posterior marginals obtained by IBP was measure as a function of priors $P(A) = (\epsilon, 1-\epsilon)$, the size of the loop (expressed by parametr k), and the posterior probabilities of one of the loop nodes $C_i \in \{C_1, ..., C_k\}$ that was induced by extreme evidence support $\lambda(C_i)$ received from its child. All nodes were bi-valued with domain size 2 which implied that absolute error in $P_{ibp}(x_i^0|e)$ is the same as that in $P_{ibp}(x_i^1|e)$ The absolute error for each loop node was averaged over the instances of the network:

$$\Delta_{avg}(X_i) = \frac{1}{M} \sum_{X_i \in X^m \setminus E} |P_{ibp}(x_i^0|e) - P(x_i^0|e)|$$

The error value was averaged over 1000 instances. We did not measure convergence of IBP in the single-loop networks because IBP was shown to always converge when a network has only 1 loop ([Weiss 2000]).

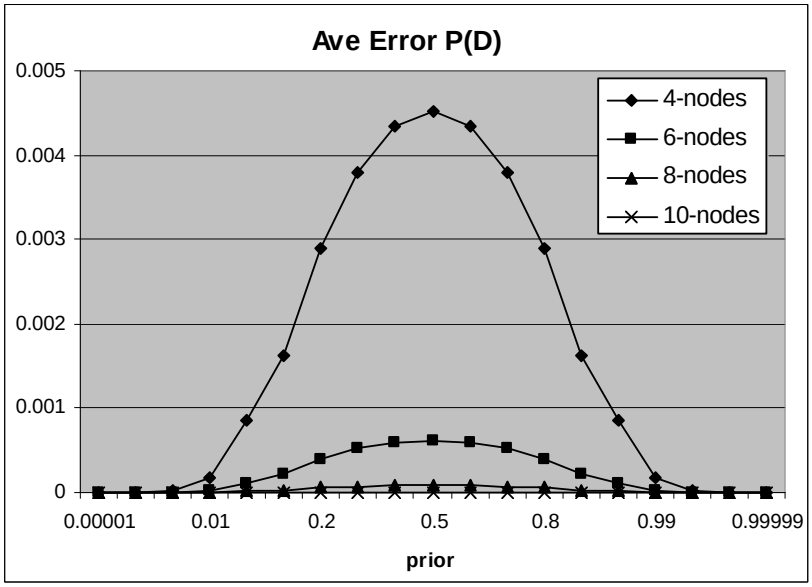


Figure 4.6: Average error $\Delta_{avg}(D)$ in a single loop, $k \in [1, 2, 3, 4]$; D is a sink node.

First, we collected emprical evidence in support of analytical result obtained in theorem 4.6.1 in loops of different sizes without evidence. We varied prior values $P(A) = (\epsilon, 1-\epsilon)$ of root node A and measured average absolute error in node D as function of ϵ. The results are shown in Figure 4.8. For all nodes, except root node, the average error value peaked at $P(A) = 0.5$, as expected, since $\epsilon = 0.5$ is the maximum of the function $f(\epsilon) = \epsilon(1-\epsilon)$. The average error for a root node consistently had a minimum at $P(A) = 0.5$ which we cannot explain analytically at this point and plan to investigate in the future. Note that the maximum average error is reduced by an order of magnitude as size of the loop increases. We do not show the bound $\epsilon(1-\epsilon)$ graphically as the estimated bound would be considerable bigger compared to actual error values.

The remainder of experiments with single-loop networks were conducted in presence of evidence in order to obtain the empirical support for ϵ-cutset conjecture. The figures 4.7 and 4.8 show the average error $\Delta_{avg}(X_i)$ as a function of priors (Figure 4.7):

$$\Delta_{avg} \sim f(\epsilon)$$

and support $\lambda(C_i)$ (Figure4.8):

$$\Delta_{avg} \sim f(\lambda(C_i)$$

for the 6-node loop (k=2) (the results are are typical of other loop networks which we leave out for conciseness). For all nodes and all loop sizes, the average error was approaching zero as $\epsilon \to 0, 1$ (Figure 4.7) and as $\lambda(C_i) \to 0, 1$ (Figure4.8) as we expected.

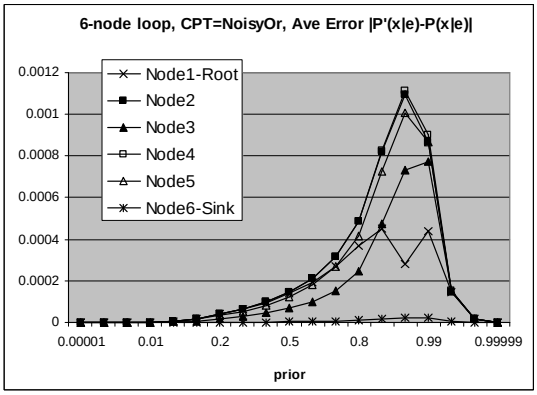


Figure 4.7: Average error in $P(X|E)$ is plotted against $P(A = 0)$.

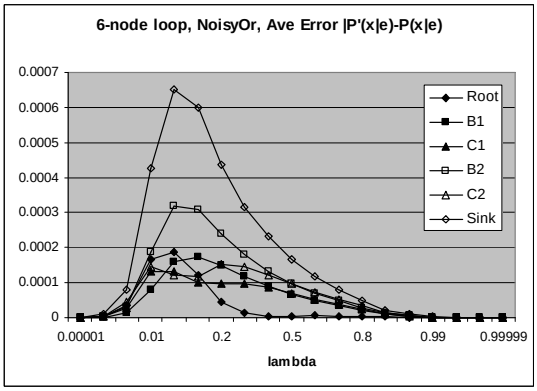


Figure 4.8: Average error in $P(X|E)$ is plotted against (C_2).

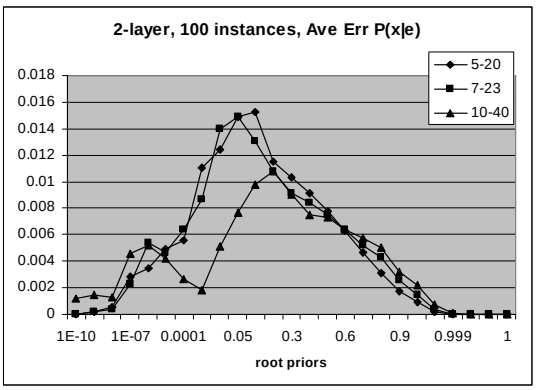
4.8.2 2-layer noisy-or networks, ϵ-cutset

In 2-layer networks, the absolute error was average over all nodes in the network and then over all the instances generated:

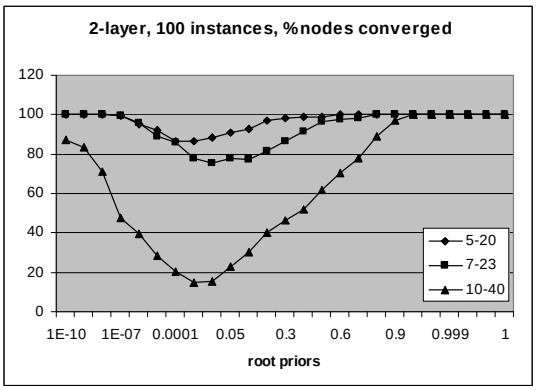
$$\Delta_{avg} = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{N}\sum_{X_i \in X^m \setminus E}\frac{1}{D(X_i)}\sum_{x_i \in D(X_i)}|P(x_i|e) - P(x_i|e)|$$

Additionally, we measure the convergence of IBP as a percent of nodes where IBP has converged. The IBP was considered as converged if the differenced between estimates obtained in the last two iterations did not exceed threshold θ=1E-10:

$$|P_{ibp}^{T-1}(x_i|e) - P_{ibp}^{T}(x_i|e)| \leq \theta$$



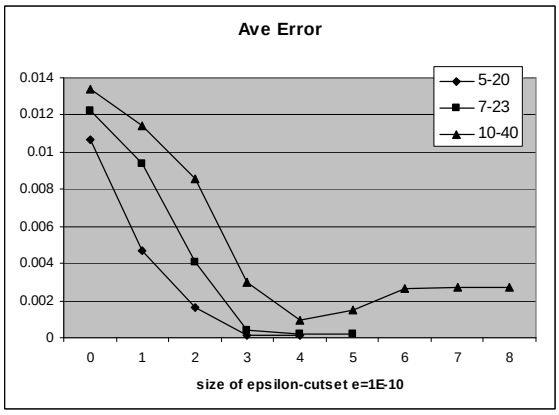
(a) Average error



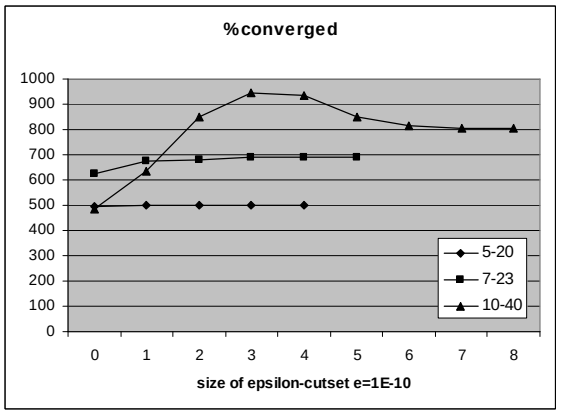
(b) Percent of converged nodes

Figure 4.9: 2-layer networks. Average error (a) and percent of converged nodes (b) are plotted agains root node priors.

We measured the accuracy and convergence of IBP in 2-layer noisy-or networks. Number of root nodes m and total number of nodes n was fixed in each test set (indexed $m - n$). Each leaf node Y_j was added to the list of children of root node U_i with probability 0.5. All leaf nodes were observed. Loop-cutset nodes were selected from root nodes using mga algorithm [Becker, Bar-Yehuda, & Geiger 1999]. The results (in Figures 4.9(a) and 4.9(b)) were averaged over 100 instances. We have observed in our experiments that initially, as priors of loop-cutset nodes become lower, the convergence and accuracy of IBP worsen. This effect was previously reported by Murphy, Weiss, and Jordan [K. P. Murphy & Jordan 1999] for 2-layer noisy-or networks with low root node priors. However, as priors of loop-cutset nodes continue to approach 0 and 1, the average error value approaches 0 and the number of converged nodes reaches 100%.



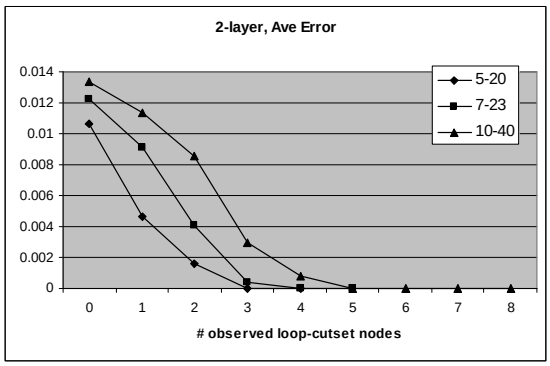
(a) Average error



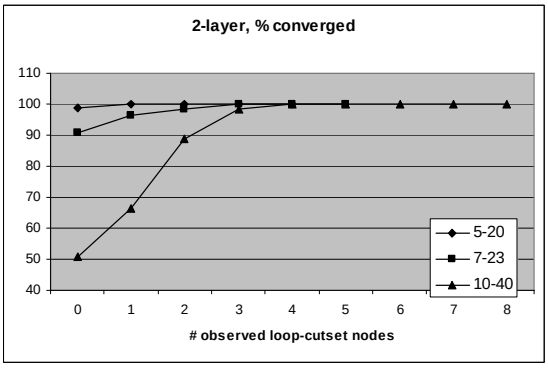
(b) Converged nodes

Figure 4.10: 2-layer Noisy-Or. Performance of IBP as the number of cutset nodes with ϵ-support increases towards loop-cutset.

4.8.3 2-layer networks, partial loop-cuset and ϵ-cutset



(a) Average error



(b) Converged nodes

Figure 4.11: 2-layer Noisy-Or. Average error and number of converged nodes are measured as number of observed loop-cutset nodes increases.

In a separate set of experiments with 2-layer networks, we have measured the average error and convergence of IBP while 1) increasing the number of observed loop-cutset nodes; 2) increasing the number of loop-cutset nodes with ϵ-support. We computed the loop-cutset of each network using mga algorithm [Becker, Bar-Yehuda, & Geiger 1999] which consisted of a susbset of root nodes.

In order to investigate the accuracy and convergence of IBP with respect to different subsets of loop-cutset nodes with ϵ-support, we fixed the prior values for some of the loop-cutset nodes (primary source of ϵ-support) to (1-ϵ, ϵ) and set the rest of the priors to (0.5,0.5). We computed the average error (over all nodes except evidence nodes) in the posterior marginals computed by IBP and percent of converged nodes while verying the size of the subset of loop-cutset nodes with ϵ-support. The results are presented in figure 4.8.3 with ϵ=1E-10. which show that IBP does not improve monotonically with the size of the subset with ϵ-support although eventually average error converges to some small value. We obtained similar results in experiments with other values of ϵ (two order of magnitude bigger and smaller).

Next, we fixed all priors to (0.5, 0.5) and computed the average error (over all nodes except evidence nodes) in the posterior marginals computed by IBP and percent of converged nodes while verying the number of observed loop-cutset nodes. Observed value was chosen at random with uniform probability The results are presented in figure 4.11. As the number of observed loop-cutset nodes increases, we observe the monotonic decrease in the average error value and increase in the number of points where IBP converged.

4.8.4 Random noisy-or networks, ϵ-cutset

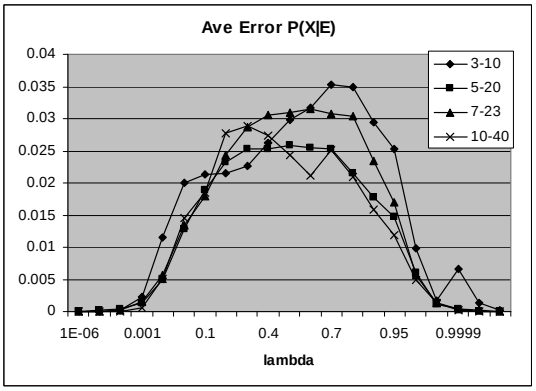


Figure 4.12: Average error in $P(X|E)$ is plotted against $\lambda_{Y_k}(X_k)$.

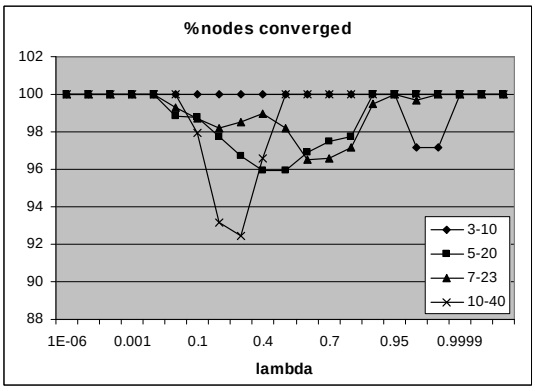


Figure 4.13: Percent of converged nodes is plotted against $\lambda_{Y_k}(X_k)$.

Random $m-n$ networks of size n were constructed by designating first m nodes as roots, last m nodes as leaves, and then adding each node X_i, $i > m$, to the list of children of node X_j, $j > i$, with probability 0.2. All leaf nodes were observed. Loop-cutset nodes were selected using mga algorithm proposed in [Becker, Bar-Yehuda, & Geiger 1999]. For each loop-cutset node, an extra child node Y_k was added with a symmetrical CPT. To control $\lambda_{Y_k}(X_k)$ messages, $P(Y_k|pa(Y_k))$ was varied. Results were averaged over 100 instances. As results demonstrate (figures 4.12 and 4.13), the average error approaches 0 and convergence of IBP reaches 100% as $\lambda_{Y_j}(X) \rightarrow 0, 1$.

4.8.5 Random Noisy-Or networks, partial loop-cuset and ϵ-cutset

We have generated a set of instances of Random Noisy-Or networks with binary variables, where, again, only some of the loop-cutset nodes were observed or only some of the loop-cutset nodes had ϵ-support. The goal of those experiments was to find out whether accuracy and convergence of belief propagation algorithm will improve monotonically as the number of observed loop-cutset nodes or the number of loop-cutset nodes with ϵ-support increases.

Our results demonstrate that performance of belief propagation algorithm may improve monotonically for some values of loop-cutset nodes. It appears that depending on which domain value of a loop-cutset node receives ϵ-support, the improvement in the IBP estimates may be monotonic (perhaps, ϵ-cutset effect cascades through the network increasing the number of nodes with strong ϵ-support) or have an opposit effect degrading performance of IBP.

Similar results were obtained for the case when we gradually increase the number of observed loop-cutset nodes. In figure 4.8.5, number of observed nodes gradually increases. And again the process lacks monotonicity. First, average error in posterior marginal increases as number of observed loop-cutset nodes increases. Average error decreases only when majority of loop-cutset nodes are observed. We observed the same for convergence. Apparently, even though observed loop-cutset nodes break some loops in the network, they can negatively affect the computation in the remaining loops.

4.9 Related work and conclusions

Empirical study of the performance of belief propagation algorithm [I. Rish & Dechter 1998; Frey & MacKay 1997] in different types of coding networks including Humming codes, low-density parity check, and turbocodes, demonstrated that accuracy of IBP is considerably better when noise level σ is low. The notion of ϵ-cutset effect defined in section 4.5 (which we prove for Bayesian networks without evidence) offers some explanation for those results. Lower noise level means that code nodes receive stronger support for a sinle value from their observed children $\lim_{\sigma \rightarrow 0} \lambda(U_i) \rightarrow 0, 1$. As a result, for small σ, root nodes become ϵ-cutset of the network.

An investigation into the distribution of cycle lengths in coding networks has demonstrated that a node has a low probability (less than 0.01) of being in a cycle of length less than or equal to 10 [Ge, Eppstein, & Smyth 1999]. Furthermore, the CPTs derived for edges with low Gaussian noise σ define very strong correlation between parent/child node values. Thus, observed child node will send quite strong support for the observed

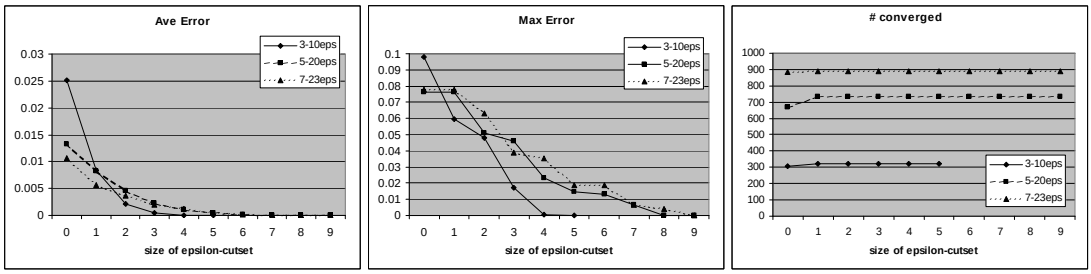


Figure 4.14: Random noisy-or networks. Average error, maximum error and number of converged nodes are plotted against the number of loop-cutset nodes with ϵ-support for value 1: $\lambda(X = 1) \rightarrow 1$ increases.

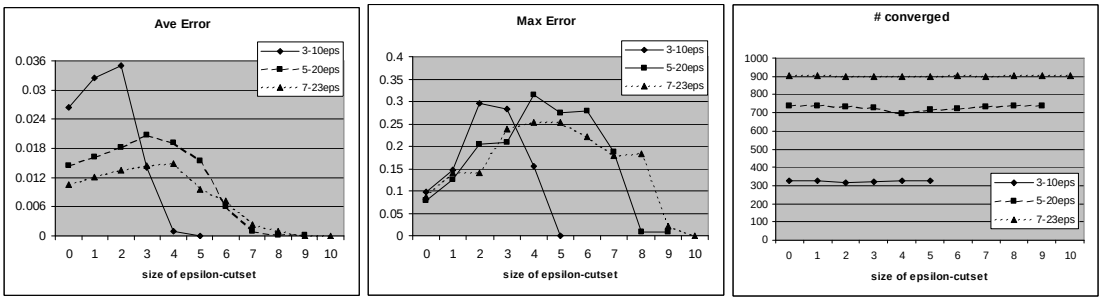


Figure 4.15: Random noisy-or networks. Average error, maximum error, and number of converged nodes are plotted agains the number of loop-cutset nodes with ϵ-support for value 0; $\lambda(X = 0) \rightarrow 1$.

value to the parent. Both of the above observations combined with our empirical findings, provide an insight into excellent performance of IBP in coding networks. Coding networks have good parameters for two different factors influencing convergence and accuracy of IBP: large loop size and strong ϵ-support.

The work presented here has two novelties. First, it provides a direct analytical connection between loop size, root priors, and evidence support and the error in the posterior marginals computed by IBP. We derived an exact expression for the error value only for a special case of a node in a single-loop network without evidence. However, the empirical evidence leads us to the same conclusions as our analytical findings which indicates that the mechanics behind the performance of IBP in single-loop network and multiple-loop networks with or without evidence is the same.

The second novelty is extending well-known loop-cutset criteria that guarantees convergence and correctness of IBP in loopy networks when evidence nodes constitute a loop-cutset to instances where loop-cutset nodes are not observed, but receive an ϵ-support. The proposed ϵ-cutset criteria guarantees the convergence and certain degree of accuracy when ϵ is sufficiently small. The next step in our research is to devise means of estimating the threshold ϵ value that will guarantee the convergence of IBP and desired degree of accuracy in posterior marginals computed by IBP.

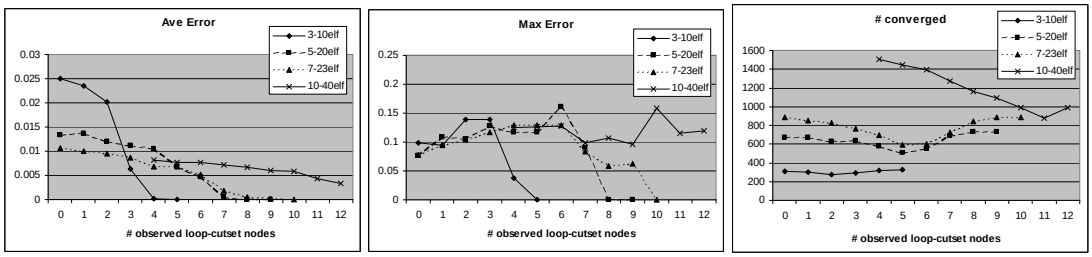


Figure 4.16: Random noisy-or networks. Average error,maximum error and number of converged nodes are plotted against the number of observed loop-cutset nodes increases.

Chapter 5

Future Work

In chapter 2, we have proposed a new sampling scheme called w-cutset that achieves a reduction in sampling variance and improvement in the convergence of the estimates via collapsing of the sampling space, specifically, sampling only a subset of variables. This scheme combines sampling with exact inference. The sampled variables are carefully selected to simplify the structure of the Bayesian network and control the complexity of exact inference. We demonstrated the efficiency of the proposed scheme empirically for a set of benchmark networks. In chapter 3, we investigated the related problem of finding minimal w-cutset.

The primary objective of the future work is to define alternative means of reducing variance and improving convergence of sampling in Bayesian networks by exploiting the structure of the graph. One possibility is to seek efficient implementation of the known statistical variance reduction methods. Alternatively, we can combine sampling with approximate inference such as Iterative Belief Propagation presented in chapter 4 when combination of sampling and exact inference is not feasible.

The auxiliary research question remaining is that of investigating the methods for finding minimal w-cutset with respect to the original graph of the Bayesian network as opposed to finding a w-cutset of its tree-decomposition.

Finally, we plan to continue work related to computing error bounds for IBP algorithm. The goal is to define criteria, computable in polynomial time, that will establish convergence of IBP and realistic bounds on the IBP estimates at least for some of the variables in the network. The promising results have been previously obtained on convergence points of IBP for Markov networks by [Tatikonda & Jordan 2002] and on posterior bound estimation by [Leisink & Kappen 2003].

5.1 Restricting Maximal Correlation in Gibbs Sampler

An obvious extension to the regular Gibbs sampling over variables X is to sample highly correlated variables simultaneously as proposed in [J. Liu & Kong 1994] resulting in the blocking Gibbs sampler. However, the grouping method previously proposed implementation in [Jensen, Kong, & Kjaerulff 1995] focuses on managing the complexity of simultaneously sampling the block of variables rather then identifying and grouping of highly correlated variables.

In w-cutset sampling, parameter w controls the complexity of computing necessary joint probabilities $P(x_1, ..., x_k, e)$. There is additional cost incurred by sampling a block of variables that grows exponentially with the size of the block. Consider sampling a group variables $\{X_1, \ldots, X_k\}$ with domains of size 2 and the task of generating new sample using systematic scan Gibbs sampler. Sampled individually, the variables will require computing their joint probability $2 \cdot \sum_{i=1}^{k} |D(X_i)| = 2 \cdot k$ times, 2 for each variable: $P(x_i^0|x_{-i}^{(t)})$ and $P(x_i^1|x_{-i}^{(t)})$. When sampled simultaneously, we need to compute and record $\prod_{i=1}^{k} |D(X_i)| = 2^k$ joint probability distributions. Hence, sampling very large blocks of variables simultaneously maybe prohibitive and in practical applications. Enforcing the limit on the maximum block size we can manage the additional cost of blocking Gibbs sampler and focus on identifying and grouping highly correlated variables.

By enforcing a bound on state transition probabilities $P(x_i|x_{-i}^{(t)})$ will can enforce the bound on the maximal correlation between two states. Given a systematic scan Gibbs sampler over three variables $\{X_1, X_2, X_3\}$, the transition probability p_{ij} from state $x^{(i)}$ is state $x^{(j)}$ can be evaluated as follows:

$$p_{ij} \leq \max_{x_1,x_2,x_3} P(x_1|x_2,x_3)P(x_2|x_1,x_3)P(x_3|x_1,x_2)$$

$$p_{ij} \leq \max P(x_1|x_2,x_3) \max P(x_2|x_1,x_3) \max P(x_3|x_1,x_2)$$

Blocking two variables together, for example X_1 and X_3, we obtain bound:

$$p'_{ij} \leq \max_{x_1,x_3,x_2} P(x_1,x_3|x_2)P(x_2|x_1,x_3)$$

$$p'_{ij} \leq \max P(x_3|x_1,x_2) \max P(x_2|x_1,x_3) \max P(x_1|x_2)$$

It is easy to show that $\max P(x_1|x_2) < \max P(x_1|x_2,x_3)$ when all probabilities are > 0. Therefore, in choosing to group X_1 and X_3, we should try to maximize the Δ-function:

$$\Delta = |\max p_{ij} - \max p'_{ij}| = \max P(x_2|x_1,x_3) \max P(x_3|x_1,x_2)(\max P(x_1|x_2,x_3) - \max P(x_1|x_2))$$

A more a rigorous analysis of this problem is necessary to obtain conclusive results. However, we believe that we can find good candidates for grouping by pre-sampling and evaluating $P(x_i|x_{-i}^{(t)})$.

A special of highly correlated variables is when the variables in fact have a deterministic relationship. As we have discussed in 2.2, when deterministic probabilistic relations are present, Gibbs sampler does not converge. On the contrary, when it is likely to get "stuck" in one area of the sampling space. Many applications yield networks where some of the probabilities are deterministic. In some networks with regular structure, we can easily identify an ergodic subspace. For example, in coding networks, it is easy to see that sampling space over coding bits of the network is ergodic. In other instances, as in the case of Hailfinder network used in experiments in

5.2 section:ccsExperiments

, we only know that some of the nodes have deterministic entries in their conditional probabilities table and cannot immediately if a cutset sampling space is ergodic or not.

Recent results obtained by Dechter and Mateescu in [Dechter & Mateescu 2003] indicate that IBP always correctly identifies zero marginals. In the course of empirical evaluation of the w-cutset sampling, we observed an improvement in the convergence of the full Gibbs sampler if we first identified nodes with zero marginals using IBP and instantiated them. In some instances, that was sufficient to obtain an ergodic sampling space although it is still not resolved whether IBP finds all zero marginals ([Dechter & Mateescu 2003]).

However, identifying zero marginals using IBP is not sufficient. More generally, we want to establish, when adding the next node C_k into into the w-cutset $\{C_1,...,C_{k-1}\}$ whether $\exists c_1,...,c_{k-1} \in D(C_1,...,C_{k-1})$ and $\exists c_k \in D(C_k)$ s.t. $P(c_k|c_1...c_{k-1}) = 0$.

5.3 Monte Carlo Sampling Variance Reduction Schemes

There is a number of statistical variance reduction methods for Monte Carlo sampling that modify the sampling procedure. Some of the more prominent are subsampling, multi-grid sampling, and stratified sampling ([Liu 2001]). In short, subsampling generally means discarding some of the samples. The multi-grid sampling creates a hierarchy of coarse to fine-grain grids over sampling space, performs sampling separately on each grid layers, and defines contraction and expansion operators to transfer information between grids. Finally, the stratified sampling is based on dividing the sampling space S into a set of subspace $S = S_1 \cup ... \cup S_q$ called *strata*, drawing samples from each strata independently, and then averaging the estimates obtained from each strata.

Each of those approaches guarantees a reduction in sampling variance and increase in the convergence speed of the sampling scheme but under special conditions. Furthermore, adapting those special sampling techniques to the Gibbs sampler is not well-defined.

The basic subsampling schemes attempt to reduce the correlation between samples by discarding some of the intermediate samples and using for example only every k^{th} sample or implement a randomized scheme for selecting samples that are included in the estimation. However, it was shown previously ([MacEachern & Berliner 1994]) we usually obtain better estimates if make use of all the samples. A more sophisticated subsampling scheme for a random scan sampler is discussed in ([MacEachern & Peruggia 2000]). Yet, we cannot obtain qunatifiable guarantees. Furthermore, it appears that comparative performance of random scan Gibbs sampler and systematic scan Gibbs sampler was never investigated rigorously enough (in particular, in context of Belief networks) to conclude that one is better than the other. The last study of random scan and system scan Gibbs sampler dates back to 1997 ([Roberts & Sahu 1997]). Therefore, there is no conclusive indication that a significant improvement (or any) can be achieve by subsampling.

The multi-grid Monte Carlo sampling was originally formulated for statistical physics problems in [Goodman & Sokal 1989] and then adapted to conditional probability-based sampling by [Liu & Sabatti 2000]. Roughly, the coarser grid is designed to combine slow evolving components and speed-up the propagation of

information. The potential advantage of this method in application to Bayesian networks is that we can avoid re-sampling variables whose values change rarely (which maybe in part to high correlation between variables) in the coarser-grid. However, the definition of the coarser grids as well as contraction and expansion operators so far have been constructed on the case by case basis (see [Liu & Sabatti 2000]) and the generalized implementation of multi-grid Monter Carlo sampling for Bayesian networkss requires further development.

In the ramainder of this section, we focus on stratified sampling and its implications in the context of Baysian networks.

Stratified sampling can lead to the reduction in sampling variance if the estimated function is "smoother" in each area S_i compared to the whole sampling space S, meaning the range of values of function of interest f $f(x)$ is considerably smaller over S_i than its total range of values over S. Assuming we can predict smooth regions of $f(x)$ and estimate the number of samples T_i that we need to draw from each region , we first obtain an estimate $\hat{f}_i(X)$ of $f(x)$ over each region S_i:

$$\hat{f}_i(X) = \frac{1}{T_i} f(x^{(t)})$$

and then compute their average:

$$\hat{f}(X) = \frac{1}{q} \sum_{i=1}^{q} \hat{f}_i(X)$$

In the basic population sampling, the number of samples T_i drawn from each subspace S_i is proportional to the size of the population in S_i. In special applications, the number of samples drawn from each region maybe proportional to the importance of that region to the objective function $f(x)$. In Bayesian networks, the number of samples should be proportional to the probabilistic mass of the sampled region. Therefore, we need to solve two problems: 1) predicting subspace S_i where the range of values of posterior marginals of interest $P(x_i|e)$ is smooth; 2) predicting the number of samples to be drawn from each region.

In [Bouchaert 1994], authors used stratified sampling in combination with likelyhood weighting. They defined strata as a set of disjoint interval covering the range between 0 and 1. Each instantiation of sample variables was assigned to an interval; the length of the interval determined the estimated probability mass of the instantion. A random number chosen uniformly between 0 and 1 determined the next sample instantiation. The proposed scheme has potential problem tracking instantiations and computing starta intervals when the number of variables in the sample increases.

We propose to associate each strata with an instatiation of a subset of variables Y such that their number of instances $D(Y)$ is tractable. An estimate obtained over subspace $y \in D(Y)$ is then an estimate of the conditional probabilitly distribution $P(x_i|y, e)$. The number of samples T_y from each subspace must be proportional to its probability mass $P(y|e)$. Therefore, the main problem is that of obtaining a good estimate of $P(y|e)$.

An easy special case is when we want to estimate the posterior marginal of variable X_i that is independent from Y conditional on evidence e and, as a result, $P(x_i|y, e) = P(x_i|e)$. Then, relative to X_i, we can draw equal number of samples from each subspace Y_i and compute an average. The subset of sampling variables Y such that X_i is independent from Y can be easily obtained by examining the structure of the Bayesian network.

Such an implementation of stratified sampling also relates to an observation made in [MacEachern & Peruggia 2000] that variance of a systematic scan Gibbs sampler can be reduced if we equally weight the samples corresponding to the instantiations of a subset Y such that function of interest $f(x)$ is independent from Y.

In a more general case, we can approximately predict that the smooth regions of a function $P(x_i, e)$ exploiting the structure of the graph. The nodes that are far from X_i in the network usually will have a lesser effect on its marginal probabilities. Therefore, we can associated stratas with instantiations of nodes Y in the its local neighborhood.

Now, we are faced with the problem of evaluating $P(y|e)$ before we can apply stratified sampling. We can estimate $P(y|e)$ by first pre-sampling the $P(e)$ and $P(y, e)$ for each instance y:

$$\hat{P}(y|e) = \frac{\hat{P}(y, e)}{\hat{P}(e)}$$

We can use a form of importance sampling such as Likelyhood Weighting to obtain estimates $P(e)$ and $P(y, e)$. However, further analysis of this problem is necessary.

5.4 Hybrid Inference Methods

The w-cutset sampling scheme proposed in chapter 2 combines sampling and exact inference. As we reduce sampling set size, induced width w increases until exact inference eventually can become infeasible. When exact inference is no longer feasible and further reduction in sampling is desired, we can replace exact computation of sampling probabilities $P(c_i|c_{-1}^{(t)}, e)$ with approximate inference. Iterative Belief Propagation algorithm is a good candidate for this scheme because it is fast (the inference time is linear in the size of the network) and it generally produces good approximations when it converges.

In particular, we have observed in our empirical evaluation of IBP in section 4.8 that IBP tends to improve monotonically (with respect to both the convergence and the precision of the estimates) as the number of observed cutset variables increases. Therefore, when estimates $P_{ibp}(x_i, e)$ are not satisfactory due to lack of convergence or lack of confi dence in the estimates, we maybe able to improve by combining IBP with conditioning. If selected cutset C is small, we can sum over all instances of C the estimated probabilities computed to IBP (similar to cutset conditioning):

$$P'(x_i|e) = \sum_c P_{ibp}(x_i|c, e)P_{ibp}(c|e)$$

When the cutset C becomes large, we can sample c from $P(c|e)$:

$$c_i \leftarrow P_{ibp}(c_{-i}^{(t)}, e)$$

resulting in estimates:

$$P'(x_i|e) = \frac{1}{T}\sum_{t=1}^{T} P_{ibp}(x_i|c^{(t)}, e)$$

The hybrid scheme combining sampling and IBP has a potential to improve over pure sampling or pure IBP. Previosly, a combination of importance sampling and IBP was proposed in [Yuan & j. Druzdzel 2003] where IBP is used to update the trial distribution. The resulting hybrid scheme reportedly outperformed previous state-of-the-art importance sampling algorithm [Cheng & Druzdzel 2000]. Therefore, it is likely that proposed combination of Gibbs sampling and IBP will perform well too.

5.5 Backward Cluster Sampling

The backward sampling scheme was proposed in [Fung & del Favero 1994] the nodes in a Bayesian network were processed and assigned a new sample value in reverse topological order, the opposite of the forward smpling-based methods (including Logic sampling Likelyhood Weighting) that process nodes in topological order and, as result, often sample from trial distribution very different from $P(x|e)$ because evidence nodes often are found at the bottom of the network (we typically observed the manifestations and symptoms of events as opposed to the events themselves). Backwards sampling helps to ensure that evidence nodes are processed early and trial distribution is close to $P(x|e)$.

Using the idea of backward sampling, namely processing evidence nodes early, we maybe able to improve the effi ciency of the cluster sampling as previously described in [Koller, Lerner, & Angelov 1998] and [Kjærulff 1995].

The exact algorithms relying on the cluster-tree decomposition [Lauritzen & Spiegelhalter 1988; Dechter 1999; Jensen, Lauritzen, & Olesen 1990; F. Jensen & Dittmer 1994] hult when the size of the separators between clusters requires recording prohibitively large probability distribution functions (exponential in the size of the separator). The cluster sampling scheme scheme defi ned in [Koller, Lerner, & Angelov 1998] (extending the work in [Kjærulff 1995]) samples the separator function estimating and recording only non-zero probability tuples generated while sampling. The sampling essentially helps us to identify tuples with high-probability mass. This cluster sampling scheme encounters problems with the evidence propagation because initially we sample separator function taking into account evidence information embedded in distant clusters.

On the other hand, a bucket elimination scheme ([Kask *et al.* 2001; Dechter 2001]) evidence nodes are eliminated fi rst and then the remaining nodes are organized into clusters called buckets. Otherwise, bucket elimination is quite similar to cluster-tree propagation and encounters similar problems when separator sizes between the buckets are too big. Previously, a mini-bucket approximation scheme was proposed to manage the networks with large separator sizes ([Larossa, Kask, & Dechter 2001]). However, we have no good heuristics for identifying mini-buckets and, as a result, the performance of mini-bucket scheme is non-monotonous: as we increase maximum separator function size, the performance of the algorithm in some instances will deteriorate. Alternatively, we can sample the separator function similar to [Larossa, Kask, & Dechter 2001],

but avoiding the problem of propagating evidence information. Furthermore, this quality of the estimates obtained in this scheme are likely to improve monotonically as we increase the maximum size of the recorded function and allow to record more samples.

The proposed combination of bucket elimination and sampling is so far only developed conceptually. Further analysis of the convergence and implementation trade-offs for this scheme is necessary before any conclusive results can be obtained.

Acknowledgments

Many thanks to my advisor, Professor Rina Dechter, for her inspiration and support in this research work. I would not be here without her patience and encouragement. This work was supported in part by the NSF grant IIS-0086529 and MURI ONR award N00014-00-1-0617.

References

A. Doucet, N. deFreitas, N. G. 2001. *Sequential Monter Carlo Methods in Practice*. Springer-Verlat, New York, Inc.

Abdelbar, A. M., and Hedetniemi, S. M. 1998. Approximating maps for belief networks in np-hard and other theorems. *Artificial Intelligence* 102:21–38.

Arnborg, S. A. 1985. Effi cient algorithms for combinatorial problems on graphs with bounded decomposability - a survey. *BIT* 25:2–23.

Becker, A., and Geiger, D. 1996. A suffi ciently fast algorithm for fi nding close to optimal junction trees. In *Uncertainty in AI*, 81–89.

Becker, A.; Bar-Yehuda, R.; and Geiger, D. 1999. Random algorithms for the loop cutset problem. In *Uncertainty in AI*.

Bellare, M.; Helvig, C.; Robins, G.; and Zelikovsky, A. 1993. Provably good routing tree construction with multi-port terminals. In *Twenty-Fifth Annual ACM Symposium on Theory of Computing*, 294–304.

Bidyuk, B., and Dechter, R. 2003a. Cycle-cutset sampling for bayesian networks. In *Sixteenth Canadian Conf. on AI*.

Bidyuk, B., and Dechter, R. 2003b. Empirical study of w-cutset sampling for bayesian networks. In *Uncertainty in AI*.

Bidyuk, B., and Dechter, R. 2004. On fi nding minimal w-cutset problem. In *Uncertainty in AI*.

Bouchaert, R. R. 1994. A stratifi ed scheme fro inference in bayesian belief networks. In *Uncertainty in AI*.

Casella, G., and Robert, C. P. 1996. Rao-blackwellisation of sampling schemes. *Biometrika* 83(1):81–94.

Cheng, J., and Druzdzel, M. J. 2000. Ais-bn: An adaptive importance sampling algorithm for evidenctial reasoning in large baysian networks. *Journal of Aritificial Intelligence Research* 13:155–188.

Cooper, G. 1990. The computational complexity of probabilistic inferences. *Artificial Intelligence* 393–405.

Dagum, P., and Luby, M. 1993. Approximating probabilistic inference in bayesian belief networks is np-hard. In *National Conference on Artificial Intelligence (AAAI-93)*.

Dechter, R., and Mateescu, R. 2003. A simple insight into properties of iterative belief propagation. In *Uncertainty in AI*.

Dechter, R., and Pearl, J. 1987. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence* 34:1–38.

Dechter, R. 1990. Enhancement schemes for constraint processing: Backjumping, learning and cutset decomposition. *Artificial Intelligence* 41:273–312.

Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence* 113:41–85.

Dechter, R. 2001. *Constraint Processing*. Morgan Caufmann.

Doucet, A.; de Freitas, N.; Murphy, K.; and Russell, S. 2000. Rao-blackwellised particle fi ltering for dynamic bayesian networks. In *Uncertainty in AI*, 176–183.

Escobar, M. D. 1994. Estimating normal means iwth a dirichlet process prior. *Journal of the American Statistical Aasociation* 89:268–277.

F. Jensen, F. J., and Dittmer, S. 1994. From influence diagrams to junction trees. In *Tenth Conference on Uncertainty in Artificial Intelligence*, 367–363.

Frey, B. J., and MacKay, D. J. C. 1997. A revolution: Belief propagation in graphs with cycles. In *Neural Information Processing Systems*, volume 10.

Fung, R., and Chang, K.-C. 1989. Weighing and integrating evidence for stochastic simulation in bayesian networks. In *Uncertainty in AI*, 209–219.

Fung, R., and del Favero, B. 1994. Backward simulation in bayesian networks. In *The Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-94)*, 227–234. Morgan Kaufmann Publishers.

Ge, X.; Eppstein, D.; and Smyth, P. 1999. The distribution of cycle lengths in graphical models for iterative decoding. Technical Report UCI-ICS 99-10, UCI.

Geigher, D., and Fishelson, M. 2003. Optimizing exact genetic linkage computations. In *7th Annual International Conf. on Computational Molecular Biology*, 114–121.

Gelfand, A. E., and Smith, A. F. M. 1990. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association* 85:398–409.

Geman, S., and Geman, D. 1984. Stochastic relaxations, gibbs distributions and the bayesian restoration of images. *IEEE Transaction on Pattern analysis and Machine Intelligence (PAMI-6)* 721–42.

Georg Gottlob, N. L., and Scarello, F. 1999. A comparison of structural csp decomposition methods. *Ijcai-99*.

Geyer, C. J. 1992. Practical markov chain monte carlo. *Statistical Science* 7:473–483.

Gilks, W. R., and Wild, P. 1992. Adaptive rejection sampling for gibbs sampling. *Applied Statistics* 41:337–348.

Gilks, W.; Richardson, S.; and Spiegelhalter, D. 1996. *Markov chain Monte Carlo in practice*. Chapman and Hall.

Goodman, J., and Sokal, A. D. 1989. Multigrid monte carlo method. conceptual foundations. *Phys. Review D* 45:2035–2072.

Groot, M. H. D. 1986. *Probability and Statistics, 2nd edition*. Addison-Wesley.

Hastings, W. K. 1970. Monter carlo sampling using markov chains and their applications. *Biometrica* 57(1):97–109.

Henrion, M. 1988. Propagating uncertainty in bayesian networks by probabilistic logic sampling. In *Uncertainty in AI*, 149–163.

Hogg, R. V., and Tanis, E. A. 2001. *Probability and Statistical Inference*. Prentice Hall.

Horvitz, E.; Suermondt, H.; and Cooper, G. 1989. Bounded conditioning: Flexible inference for decisions under scarce resources. In *Workshop on Unertainty in Artificial Intelligence*, 181–193.

I. Rish, K. K., and Dechter, R. 1998. Empirical evaluation of approximation algorithms for probabilistic decoding. In *Uncertainty in AI (UAI'98)*.

J. Liu, W. W., and Kong, A. 1994. Covariance structure of the gibbs sampler with applications to the comparison of estimators and augmentation schemes. *Biometrika* 81(1):27–40.

J. S. Liu, W. W., and Kong, A. 1995. Covariance structure and convergence rate of the gibbs sampler with various scans. *Journal of the Royal Statistical Society, Series B* 57(1):157–169.

Jensen, C.; Kong, A.; and Kjaerulff, U. 1995. Blocking gibbs sampling in very large probabilistic expert systems. *International Journal of Human Computer Studies. Special Issue on Real-World Applications of Uncertain Reasoning*. 647–666.

Jensen, F.; Lauritzen, S.; and Olesen, K. 1990. Bayesian updating in causal probabilistic networks by local computation. *Computational Statistics Quarterly* 4:269–282.

K. P. Murphy, Y. W., and Jordan, M. I. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Uncertainty in AI (UAI'99)*.

Kask, K.; Dechter, R.; Larrosa, J.; and Fabio, G. 2001. Bucket-tree elimination for automated reasoning. *Submitted -2001*.

Kjaerulff, U. 1990. *Triangulation of graphs - algorithms giving small total space*. Number R 90-09.

Kjærulff, U. 1995. Hugs: Combining exact inference and gibbs sampling in junction trees. In *Uncertainty in AI*, 368–375. Morgan Kaufmann.

Koller, D.; Lerner, U.; and Angelov, D. 1998. A general algorithm for approximate inference and its application to hybrid bayes nets. In *Uncertainty in AI*, 324–333.

Kschischang, F. R., and Frey, B. J. 1998. Iterative decoding of compound codes by probability propagation in graphical models. *IEEE Journal on Selected Areas in Communications* 16:219–230.

Larossa, J., and Dechter, R. 2003. Dynamic combination of search and variable-elimination in csp and max-csp. *Constraints*.

Larossa, J.; Kask, K.; and Dechter, R. 2001. Up and down mini-bucket: a scheme for approximating combinatorial optimization tasks. *Technical Report*.

Lauritzen, S., and Spiegelhalter, D. 1988. Local computation with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B* 50(2):157–224.

Leisink, M. A. R., and Kappen, H. J. 2003. Bound propagation. *Journal of Artificial Intelligence Research* 19:139–154.

Liu, J. S., and Sabatti, C. 2000. Generalized gibbs sampler and multigrid monte carlo for bayesian computation. *Biometrika* 87(2):353–369.

Liu, J. 1991. *Correlation Structure and Convergence Rate of the Gibbs Sampler*.

Liu, J. 1994. The collapsed gibbs sampler in bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association* 89(427):958–966.

Liu, J. S. 1996. Nonparametric hierarchical bayes via sequential imputations. *Annals of Statistics* 24(3):911–930.

Liu, J. S. 2001. *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag, New York, Inc.

Lund, C., and Yannakakis, M. 1994. On the hardness of approximating minimization problems. *J. of ACM* 41(5):960–981.

MacEachern, S. N., and Berliner, L. 1994. Subsampling the gibbs sampler. *American Statistician* 48:188–190.

MacEachern, S. N., and Peruggia, M. 2000. Subsampling the gibbs sampler: Variance reduction. *Statistics and Probability Letters* 47:91–98.

MacEachern, S. N. 1994. Estimating normal means with a conjugate style dirichlet process prior. *Communications in Statistics-Simulation and Computation* 23(3):727–741.

MacKay, D. 1996. Introduction to monte carlo methods. In *Proceedings of NATO Advanced Study Institute on Learning in Graphical Models. Sept 27-Oct 7*, 175–204.

Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H.; and Teller, E. 1953. Equations of state calculations by fast computing. *Journal of Chemical Physics* 21(6):1087–1091.

Park, J., and Darwiche, A. 2003. Complexity results and approximation strategies for map explanations. *JAIR*.

Park, J. 2002. Map complexity results and approximation methods. In *UAI*, 388–396. Morgan Kaufmann Publishes, Inc.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

Peot, M. A., and Shachter, R. D. 1992. Fusion and propagation with multiple observations in belief networks. *Artificial Intelligence* 299–318.

Rajagopalan, S., and Vazirani, V. 1998. Primal-dual rnc approximation algorithms for (multi)set (multi)cover and covering integer programs. *SIAM J. of Computing* 28(2):525–540.

Rish, I., and Dechter, R. 2000. Resolution vs. search; two strategies for sat. *J. of Automated Reasoning* 24(1/2):225–275.

R.J. McEliece, D. M., and Cheng, J.-F. 1997. Turbo decoding as an instance of pearl's belief propagation algorithm. *IEEE J. Selected Areas in Communication*.

Roberts, G. O., and Sahu, S. K. 1997. Updating schemes; correlation structure; blocking and parameterization for the gibbs sampler. *Journal of the Royal Statistical Society, Series B* 59(2):291–317.

Roth, D. 1996. On the hardness of approximate reasoning. *Artificial Intelligence* 82(1-2):273–302.

Schervish, M., and Carlin, B. 1992. On the convergence of successive substitution sampling. *Journal of Computational and Graphical Statistics* 1:111–127.

Shachter, R. D., and Peot, M. A. 1989. Simulation approaches to general probabilistic inference on belief networks. In *Uncertainty in Artificial Intelligene*, 221—231.

Shimony, S. E. 1994. Finding maps for belief networks is np-hard. *Artificial Intelligence* 68:399–410.

Shoiket, K., and Geiger, D. 1997. A proctical algorithm for fi nding optimal triangulations. In *Fourteenth National Conference on Artificial Intelligence (AAAI'97)*, 185–190.

Tatikonda, S. C., and Jordan, M. I. 2002. Loopy belief propagation and gibbs measures. In *Uncertainty in AI*.

Tierney, L. 1994. Markov chains for exploring posterior distributions. *Annals of Statistics* 22(4):1701–1728.

Trevisan, L. 2001. Non-approximability resu.lts for optimization problems on bounded degree instances. *In proceedings of 33rd ACM STOC*.

Vazirani, V. V. 2001. *Approximation Algorithms*. Springer.

Weiss, Y. 2000. Correctness of local probability propagation in graphical models with loops. *Neural Computation 12* 1–41.

Yedidia, J.; Freeman, W.; and Weiss, Y. 2001. Generalized belief propagation. In *Neural Information Processing Systems*, volume 13, 689–695.

Yuan, C., and j. Druzdzel, M. 2003. An importance sampling algorithm based on evidence pre-propagation. In *UAI*, 624–631.