Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence

An Exact Algorithm for Computing the Same-Decision Probability

Suming Chen and Arthur Choi and Adnan Darwiche
Computer Science Department
University of California, Los Angeles
{*suming,aychoi,darwiche*}@*cs.ucla.edu*

Abstract

When using graphical models for decision making, the presence of unobserved variables may hinder our ability to reach the correct decision. A fundamental question here is whether or not one is ready to make a decision (stopping criteria), and if not, what additional observations should be made in order to better prepare for a decision (selection criteria). A recently introduced notion, the *Same-Decision Probability* (SDP), has been shown to be useful as both a stopping and a selection criteria. This query has been shown to be highly intractable, being $\mathrm{PP}^{\mathrm{PP}}$–complete, and is exemplary of a class of queries which correspond to the computation of certain expectations. We propose the first exact algorithm for computing the SDP in this paper, and demonstrate its effectiveness on several real and synthetic networks. We also present a new complexity result for computing the SDP on models with a Naive Bayes structure.

1 Introduction

When making any kind of decision under uncertainty, information gathering is crucial. Consider for example a physician who is examining an ill patient. The physician may perform a few tests and then strongly believe that the patient is suffering from substance-abuse induced depression. However, by not gathering more information in the form of further testing, the doctor could unwittingly be making a grave misdiagnosis. After all, 41% to 83% of patients being treated for psychiatric disorders, including depression, have been misdiagnosed and have an unresolved physical ailment that ranges from hypothyroidism to cancer [Klonoff and Landrine, 1997]. If the patient, say, was actually suffering from hypothyroidism, this misdiagnosis could have been prevented quite easily if the doctor had either checked the patient's neck carefully or performed a blood test. This example can also be seen as a warning to not make decisions that could easily change given some new information (i.e., decisions that may not be robust).

This paper is dedicated to the computation of a new probabilistic notion, called the *Same-Decision Probability* (SDP), which is mainly concerned with quantifying the robustness of a decision with respect to some hidden information. Intuitively, the SDP is the probability that one would make the same decision, had one observed the values of some hidden variables of interest.

The SDP is formulated in the context of probabilistic graphical models, which have often been used to model a variety of decision problems, e.g., in medical diagnosis [van der Gaag and Coupé, 1999], troubleshooting [Heckerman *et al.*, 1995], and in classification [Friedman *et al.*, 1997]. In these applications, there is often some unobservable variable, such as the state of a patient's health. There are also often some hidden variables which could influence our belief on this variable of interest. Before making a decision, there are two fundamental questions. The first question is whether, given the current observations, the decision maker is ready to commit to a decision. We will refer to this as the *stopping criteria* for making a decision.[1] Assuming the stopping criteria is not met, the second question is what additional observations should be made before the decision maker is ready to make a decision. This typically requires a *selection criteria* based on some measure for quantifying an observation's *value of information* (VOI).

The literature contains a number of proposals for both stopping and selection criteria. For stopping criteria, one may commit to a decision once the belief about a certain event crosses some threshold, as in [Pauker and Kassirer, 1980; Lu and Przytula, 2006]. Alternatively, we may simply perform as many observations as our budget allows, as in [Greiner *et al.*, 1996; Krause and Guestrin, 2009]. As for selection criteria, different observations may have different value of information (VOI) with respect to the decision we are interested in making. The value of an observation may depend upon how much it minimizes our expected uncertainty about an event, or it may depend on how much it raises the expected utility. Researchers have explored how to compute myopic VOI [Dittmer and Jensen, 1997] as well as non-myopic VOI [Heckerman *et al.*, 1993; Krause and Guestrin, 2009]. Additionally, [Bilgic and Getoor, 2011] have developed the Value of Information Lattice (VOILA), a framework in which all subsets of hidden variables are examined, and the

[1] [van der Gaag and Bodlaender, 2011] pose the **STOP** problem that asks whether or not the present evidence gathered is sufficient for diagnosis, or if there exists further relevant evidence that can and should be gathered.

most cost-efficient subset of features can be found.

The *Same-Decision Probability* (SDP) has been introduced recently and shown to be an effective stopping and selection criteria [Darwiche and Choi, 2010; Chen *et al.*, 2012]. The SDP is defined as a measure of potential stability in a decision given further observations of some set of hidden variables. In short, the SDP is the probability that we would have made the same decision, had we known the states of variables that have not yet been observed. Compared to calculating a low SDP, calculating a high SDP would indicate a higher degree of readiness to make a decision, as the chances of the decision changing post observation would be lower.

The SDP is hard to compute. For example, we show in this paper that computing the SDP is NP–hard even in Naive Bayes networks. In [Choi *et al.*, 2012], computing the SDP is shown to be generally $\mathrm{PP}^{\mathrm{PP}}$–complete. The $\mathrm{PP}^{\mathrm{PP}}$ class can be thought of as a counting variant of the $\mathrm{NP}^{\mathrm{PP}}$ class, for which the MAP problem is complete [Park and Darwiche, 2004]. Non-myopic value of information (VOI) is a similar query that is shown to be $\mathrm{PP}^{\mathrm{PP}}$-complete as well in [Chen *et al.*, 2012]. These problems are not only closely related in terms of complexity, but also, they both involve computing the expectation over some unobserved variables. Existing algorithms for computing the non-myopic VOI are either approximate algorithms [Heckerman *et al.*, 1993; Liao and Ji, 2008] or brute-force algorithms that are restricted to tree networks with few leaf variables [Krause and Guestrin, 2009].

The main contribution of this paper is in presenting the first *exact* algorithm for computing the SDP, which is based on a novel combination of branch-and-bound search with classical inference techniques for graphical models. As non-myopic VOI is in the same complexity class as SDP, our proposed algorithm can then serve as an example for developing similar exact algorithms for non-myopic VOI as well as other problems that are complete for $\mathrm{PP}^{\mathrm{PP}}$.

This paper is structured as follows. Section 2 provides a formal definition of the SDP and some background knowledge. Section 3 presents our algorithm and discusses its complexity. Section 4 exhibits empirical results. Section 5 closes with some concluding remarks.

2 Background

We use standard notation for variables and their instantiations, where variables are denoted by upper case letters X and their instantiations by lower case letters x. Sets of variables are then denoted by bold upper case letters $\mathbf{X}$ and their instantiations by bold lower case letters $\mathbf{x}$.

We will consistently use $\mathbf{X}$ to denote all variables in our model, $\mathbf{E} \subseteq \mathbf{X}$ to denote *evidence* variables (i.e., observed ones), and $\mathbf{U} \subseteq \mathbf{X}$ to denote the set of all *hidden* variables (i.e., unobserved ones). Hence, $\mathbf{E} \cap \mathbf{U} = \emptyset$ and $\mathbf{E} \cup \mathbf{U} = \mathbf{X}$. We will also use $D \in \mathbf{U}$ to denote a binary *hypothesis* variable with states d and $\overline{d}$, which is sometimes referred to as the *decision* variable.

Given some evidence $\mathbf{e}$, we assume that a decision will be made depending on whether $Pr(d \mid \mathbf{e}) \geq T$, for some threshold T. We are particularly interested in measuring the stability of this decision with respect to the state of some variables $\mathbf{H} \subseteq \mathbf{U} \setminus D$, which are called *query variables.*

D	$Pr(D)$
$+$	0.3
$-$	0.7

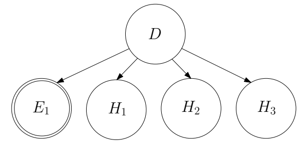

Figure 1: A Naive Bayes network (more CPTs in Table 1).

Table 1: $Pr(H_3 \mid D)$, $Pr(E_1 \mid D)$ and $Pr(H_2|D)$ are equal.

D	H_1	$Pr(H_1 \mid D)$
$+$	$+$	0.80
$+$	$-$	0.20
$-$	$+$	0.10
$-$	$-$	0.90

D	H_2	$Pr(H_2 \mid D)$
$+$	$+$	0.70
$+$	$-$	0.30
$-$	$+$	0.30
$-$	$-$	0.70

The Same-Decision Probability (SDP) is defined in [Darwiche and Choi, 2010] for this purpose.

Definition 1. *Given hypothesis variable D, evidence $\mathbf{e}$, unobserved variables $\mathbf{H}$, and threshold T, suppose we are making a decision that is confirmed by $Pr(d \mid \mathbf{e}) \geq T$. The* ***same-decision probability*** *(SDP) is defined as*

$$SDP(d, \mathbf{H}, \mathbf{e}, T) = \sum_{\mathbf{h}} [Pr(d \mid \mathbf{h}, \mathbf{e}) \geq T] Pr(\mathbf{h} \mid \mathbf{e}) \quad (1)$$

where $[Pr(d \mid \mathbf{h}, \mathbf{e}) \geq T]$ is an indicator function that $= 1$ when $Pr(d \mid \mathbf{h}, \mathbf{e}) \geq T$ and $= 0$ otherwise.

In other words, SDP is the probability that we would have made the same decision had we known the variables $\mathbf{H}$.

Consider the simple Naive Bayes network presented in Figure 1, which will serve as a running example through this paper. We will make a decision here if $Pr(d \mid \mathbf{e}) \geq T$, where d is the event $D = +$ and T is 0.50. For example, we would make this decision under evidence $E_1 = +$, since $Pr(D = + \mid E_1 = +) \geq 0.5$. Our goal now is to quantify the robustness of this decision with respect to hidden variables $\mathbf{H} = \{H_1, H_2, H_3\}$. That is, we wish to compute the probability that we would still make the same decision after having observed $\mathbf{H}$.

Consider Table 2 which enumerates all possible instantiations $\mathbf{h}$. We see that in 5 of the 8 cases, $Pr(d \mid \mathbf{h}, \mathbf{e}) \geq 0.50$. We take the sum of the probability of those instantiations and we find that $SDP(d, \{H_1, H_2, H_3\}, \{E_1\}, 0.50) = 0.5395$. Hence, according to this example, there is a good chance that we would make a different decision after having observed $\mathbf{H}$.

Computing the SDP helps us in quantifying how ready we are to make a decision, and can also help us decide which variables to observe next. There is an in-depth discussion of how SDP is useful as a stopping criteria and as a selection criteria in [Chen *et al.*, 2012]. In particular, the SDP is contrasted with classical notions such as entropy and value of information and is shown to be a useful decision-making tool that can quantify the robustness of a decision in ways that are not evident when we consider beliefs and utilities alone.

3 Computing the SDP Exactly

Computing the SDP involves computing an expectation over the hidden variables $\mathbf{H}$. The naive brute-force algorithm

Table 2: Scenarios $\mathbf{h}$ for the network in Figure 1. The cases where $Pr(d \mid \mathbf{h}, \mathbf{e}) \geq 0.5$ are bolded.

H_1	H_1	H_3	$Pr(\mathbf{h} \mid \mathbf{e})$	$Pr(d \mid \mathbf{h}, \mathbf{e})$
$+$	$+$	$+$	0.2005	**0.977**
$+$	$+$	$-$	0.0945	**0.888**
$+$	$-$	$+$	0.0945	**0.888**
$+$	$-$	$-$	0.0605	**0.595**
$-$	$+$	$+$	0.0895	**0.547**
$-$	$+$	$-$	0.1155	0.181
$-$	$-$	$+$	0.1155	0.181
$-$	$-$	$-$	0.2295	0.039

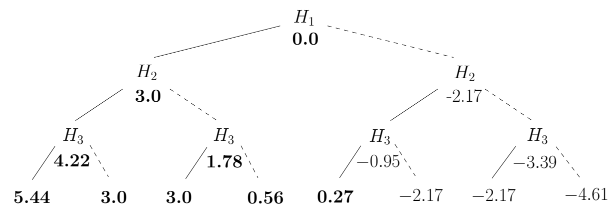

Figure 2: The search tree for the network of Figure 1. A solid line indicates $+$ and a dashed line indicates $-$. The quantity $\log O(d \mid \mathbf{q}, \mathbf{e})$ is displayed next to each node $\mathbf{q}$ in the tree. Nodes with $\log O(d \mid \mathbf{q}, \mathbf{e}) \geq \lambda = 0$ are shown in bold.

Table 3: Weights of evidence for the attributes in Figure 1.

i	w_{h_i}	$w_{\overline{h}_i}$
1	3.0	-2.17
2	1.22	-1.22
3	1.22	-1.22

would enumerate and check whether $Pr(d \mid \mathbf{h}, \mathbf{e}) \geq T$ for all instantiations $\mathbf{H}$. We now present an algorithm that can save us the need to explore every possible instantiation of $\mathbf{h}$. To make the algorithm easier to understand, we will first describe how to compute the SDP in a Naive Bayes network, which we also show to be NP–hard. We then generalize our algorithm to arbitrary networks.

3.1 Computing the SDP in Naive Bayes Networks

We will find it more convenient to implement the test $Pr(d \mid \mathbf{h}, \mathbf{e}) \geq T$ in the *log-odds* domain, where:

$$\log O(d \mid \mathbf{h}, \mathbf{e}) = \log \frac{Pr(d \mid \mathbf{h}, \mathbf{e})}{Pr(\overline{d} \mid \mathbf{h}, \mathbf{e})} \tag{2}$$

We then define the *log-odds threshold* as $\lambda = \log \frac{T}{1-T}$ and, equivalently, test $\log O(d \mid \mathbf{h}, \mathbf{e}) \geq \lambda$.

In a Naive Bayes network with D as the class variable, $\mathbf{H}$ and $\mathbf{E}$ as the leaf variables, and $\mathbf{Q} \subseteq \mathbf{H}$, the posterior log-odds after observing a *partial instantiation* $\mathbf{q} = \{h_1, \ldots, h_j\}$ can be written as:

$$\log O(d \mid \mathbf{q}, \mathbf{e}) = \log O(d \mid \mathbf{e}) + \sum_{i=1}^{j} w_{h_i} \tag{3}$$

where w_{h_i} is the *weight of evidence* h_i and defined as:

$$w_{h_i} = \log \frac{Pr(h_i \mid d, \mathbf{e})}{Pr(h_i \mid \overline{d}, \mathbf{e})} \tag{4}$$

The weight of evidence w_{h_i} is then the contribution of evidence h_i to the quantity $\log O(d \mid \mathbf{q}, \mathbf{e})$ [Chan and Darwiche, 2003]. Note that all weights can be computed in time and space linear in $|\mathbf{H}|$. Table 3 depicts the weights of evidence for the network in Figure 1.

One can then compute the SDP by enumerating the instantiations of variables $\mathbf{H}$ and then using Equation 3 to test whether $\log O(d \mid \mathbf{h}, \mathbf{e}) \geq \lambda$. Figure 2 depicts a search tree for the Naive Bayes network in Figure 1, which can be used for this purpose. The leaves of this tree correspond to instantiations $\mathbf{h}$ of variables $\mathbf{H}$. More generally, every node in the tree corresponds to an instantiation $\mathbf{q}$, where $\mathbf{Q} \subseteq \mathbf{H}$.

A brute-force computation of the SDP would then entail: 1.) Initializing the total SDP to 0, 2.) Visiting every leaf node $\mathbf{h}$ in the search tree, 3.) Checking whether $\log O(d \mid \mathbf{h}, \mathbf{e}) \geq \lambda$, and if so, adding $Pr(\mathbf{h} \mid \mathbf{e})$ to the total SDP. Figure 2 depicts the quantity $\log O(d \mid \mathbf{q}, \mathbf{e})$ for each node $\mathbf{q}$ in the tree, indicating that five leaf nodes (i.e., five instantiations of variables $\mathbf{H}$) will indeed contribute to the SDP.

We now state the key observation underlying our proposed algorithm. Consider the node corresponding to instantiation $H_1 = +$ in the search tree, with $\log O(d \mid H_1 = +, \mathbf{e}) = 3.0$. All four completions $\mathbf{h}$ of this instantiation (i.e., the four leaf nodes below it) are such that $\log O(d \mid \mathbf{h}, \mathbf{e}) \geq \lambda = 0$. Hence, we really do not need to visit all such leaves and add their contributions $Pr(\mathbf{h}|\mathbf{e})$ individually to the SDP. Instead, we can simply add $Pr(H_1 = +|\mathbf{e})$ to the SDP, which equals the sum of $Pr(\mathbf{h}|\mathbf{e})$ for these leaves. More importantly, we can detect that all such leaves will contribute to the SDP by computing a lower bound using the weights depicted in Table 3. That is, there are two weights for variable H_2, the *minimum* of which is -1.22. Moreover, there are two weights for variable H_3, the *minimum* of which -1.22. Hence, the lowest contribution to the log-odds made by any leaf below node $H_1 = +$ will be $-1.22 - 1.22 = -2.44$. Adding this contribution to the current log-odds of 3.0 will lead to a log-odds of .56, which still passes the given threshold.

A similar technique can be used to compute upper bounds, allowing us to detect nodes in the search tree where no leaf below them will contribute to the SDP. Consider for example the node corresponding to instantiation $H_1 = -, H_2 = -$, with $\log O(d \mid H_1 = -, H_2 = -, \mathbf{e}) = -3.39$. Neither of the leaves below this node will contribute to the SDP as their log-odds do not pass the threshold. This can be detected by considering the weights of evidence for variable H_3 and computing the *maximum* of these weights (1.22). Adding this to the current log-odds of -3.39 gives -2.17, which is still below the threshold. Hence, no leaf node below $H_1 = -, H_2 = -$ will contribute to the SDP and this part of the search tree can also be pruned.

If we apply this pruning technique based on lower and upper bounds, we will actually end up exploring only the partition of the tree shown in Figure 3. The pseudocode of our

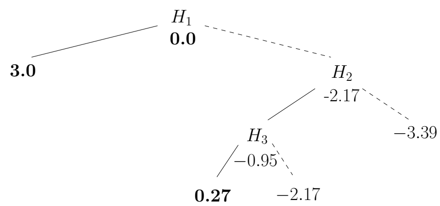

Figure 3: The reduced search tree for the network of Figure 2.

Algorithm 1 Computing the SDP in a Naive Bayes network. Note: For $\mathbf{q} = \{h_1, \ldots, h_j\}$, $w_{\mathbf{q}}$ is defined as $\sum_{i=1}^{j} w_{h_i}$.

input:
$\mathcal{N}$: Naive Bayes network with class variable D
$\mathbf{H}$: attributes $\{H_1, \ldots, H_k\}$
λ: log-odds threshold
$\mathbf{e}$: evidence

output: Same-Decision Probability p

main:

```
p ← 0.0 (initial probability)
q ← φ (initial instantiation)
DFS_SDP(q, H, 0)
return p
procedure DFS_SDP(q, H, d)
    if (log O(d | e) + w_q + Σ_{i=d+1}^{k} max_{h_i} w_{h_i}) < λ then return
    else if (log O(d | e) + w_q + Σ_{i=d+1}^{k} min_{h_i} w_{h_i}) ≥ λ then
        add Pr(q | e) to p, return
    else
        if d < k then
            for each value h_{d+1} of attribute H_{d+1} do
                DFS_SDP(q h_{d+1}, H \ H_{d+1}, d + 1)
```

final algorithm is shown in Algorithm 1.[2]

3.2 The SDP for Naive Bayes is Hard

SDP is known to be $\mathrm{PP}^{\mathrm{PP}}$–complete [Choi *et al.*, 2012]. We now show that SDP remains hard for Naive Bayes networks.

Theorem 1. *Computing the Same-Decision Probability in a Naive Bayes network is* NP*-hard.*

Proof. We reduce the number partition problem [Karp, 1972] to computing the SDP in a Naive Bayes model. Suppose we are given a set of positive integers $c_1, \ldots, c_n$, and we wish to determine whether there exists $I \subseteq \{1, \ldots, n\}$ such that $\sum_{j \in I} c_i = \sum_{j \notin I} c_j$. We can solve this by considering a Naive Bayes network with a binary class variable D having uniform probability, and binary attributes $H_1, \ldots, H_n$ having CPTs leading to weights of evidence $w_{H_i=T} = c_i$ and $w_{H_i=F} = -c_i$. In particular, the set of integers can be partitioned if there is an instantiation $\mathbf{h} = \{h_1, \ldots, h_n\}$ with $\sum_{i=1}^{n} w_{h_i} = 0$ since I would then include all indices i with $h_i = T$ in this case.

The Naive Bayes network satisfies a number of properties that we shall use next. First, $\sum_{i=1}^{n} w_{h_i}$ is either 0, ≥ 1, or ≤ -1 since all weights w_{h_i} are integers. Next, if $\sum_{i=1}^{n} w_{h_i} = c$, then $\sum_{i=1}^{n} w_{h'_i} = -c$ where $h'_i \neq h_i$. Finally, $Pr(h_1, \ldots, h_n) = Pr(h'_1, \ldots, h'_n)$ when $h'_i \neq h_i$.

Consider now the following SDP (the last step below is based on the above properties):

$$
\begin{aligned}
&SDP(D = T, \{H_1, \ldots, H_n\}, \{\}, 2/3) \\
&= \sum_{h_1,\ldots,h_n} [Pr(D = T \mid h_1, \ldots, h_n) \geq 2/3] Pr(h_1, \ldots, h_n) \\
&= \sum_{h_1,\ldots,h_n} [\log O(D = T \mid h_1, \ldots, h_n) \geq 1] Pr(h_1, \ldots, h_n) \\
&= \sum_{h_1,\ldots,h_n} \left[\sum_{i=1}^{n} w_{h_i} \geq 1 \right] Pr(h_1, \ldots, h_n) \\
&= \frac{1}{2} \sum_{h_1,\ldots,h_n} \left[\sum_{i=1}^{n} w_{h_i} \neq 0 \right] Pr(h_1, \ldots, h_n)
\end{aligned}
$$

We then have $\sum_{i=1}^{n} w_{h_i} = 0$ for some instantiation $h_1, \ldots, h_n$ iff $\sum_{h_1,\ldots,h_n} [\sum_{i=1}^{n} w_{h_i} \neq 0] Pr(h_1, \ldots, h_n) < 1$. Hence, the partitioning problem can be solved iff $SDP(D = T, \{H_1, \ldots, H_n\}, \{\}, 2/3) < 1/2$. ☐

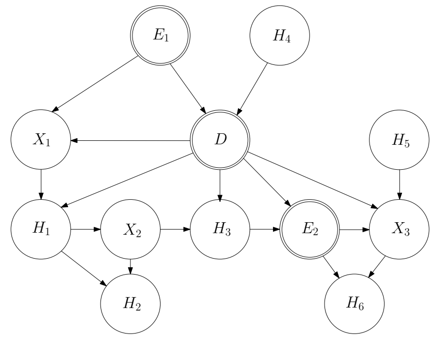

Figure 4: The partition of $\mathbf{H}$ given D and $\mathbf{E}$ is: $\mathbf{S}_1 = \{H_1, H_2, H_3\}$ $\mathbf{S}_2 = \{H_4\}$, $\mathbf{S}_3 = \{H_5, H_6\}$.

3.3 Computing the SDP in Arbitrary Networks

We will generalize our algorithm to arbitrary networks by viewing such networks as Naive Bayes networks but with aggregate attributes. For this, we first need the following notion.

Definition 2. *A partition of* $\mathbf{H}$ *given* D *and* $\mathbf{E}$ *is a set* $\mathbf{S}_1, \ldots, \mathbf{S}_k$ *such that:* $\mathbf{S}_i \subseteq \mathbf{H}$*;* $\mathbf{S}_i \cap \mathbf{S}_j = \emptyset$*;* $\mathbf{S}_1 \cup \ldots \cup \mathbf{S}_k = \mathbf{H}$*; and* $\mathbf{S}_i$ *is independent from* $\mathbf{S}_j$*,* $i \neq j$*, given* D *and* $\mathbf{E}$*.*

Figure 4 depicts an example partition.

The intuition behind a partition is that it allows us to view an arbitrary network as a Naive Bayes network, with class variable D and aggregate attributes $\mathbf{S}_1, \ldots, \mathbf{S}_k$. That is, each aggregate attribute $\mathbf{S}_i$ is viewed as a variable with states $\mathbf{s}_i$,

[2] The specific ordering of $\mathbf{H}$ in which the search tree is constructed is directly linked to the amount of pruning. We use an ordering heuristic that ranks each query variable H_i by the difference of its corresponding upper and lower bound — $\mathbf{H}$ is then ordered from greatest difference to lowest difference.

allowing us to view each instantiation $\mathbf{h}$ as a set of values $\mathbf{s}_1, \ldots, \mathbf{s}_k$. We now have:

Theorem 2. *For a partial instantiation* $\mathbf{q} = \{\mathbf{s}_1, \ldots, \mathbf{s}_j\}$,

$$\log O(d \mid \mathbf{q}, \mathbf{e}) = \log O(d \mid \mathbf{e}) + \sum_{i=1}^{j} w_{\mathbf{s}_i}, \tag{5}$$

where

$$w_{\mathbf{s}_i} = \log \frac{Pr(\mathbf{s}_i, \mid d, \mathbf{e})}{Pr(\mathbf{s}_i \mid \overline{d}, \mathbf{e})} \tag{6}$$

Proof. $\log O(d \mid \mathbf{q}, \mathbf{e}) = \log \frac{Pr(d|\mathbf{q},\mathbf{e})}{Pr(\overline{d}|\mathbf{q},\mathbf{e})}$

$= \log \frac{Pr(d|\mathbf{e}) Pr(\mathbf{s}_1|d,\mathbf{e}) \ldots Pr(\mathbf{s}_j|d,\mathbf{e})}{Pr(\overline{d}|\mathbf{e}) Pr(\mathbf{s}_1|\overline{d},\mathbf{e}) \ldots Pr(\mathbf{s}_j|\overline{d},\mathbf{e})}$

$= \log O(d \mid \mathbf{e}) + \sum_{i=1}^{j} w_{\mathbf{s}_i}$ □

Since Equations 5 and 6 are analogous to Equations 3 and 4, we can now use Algorithm 1 on an arbitrary network. This usage, however, requires some auxiliary computations that were not needed or were readily available for Naive Bayes networks. We discuss these computations next.

Finding a Partition

We first need to compute a partition $\mathbf{S}_1, \ldots, \mathbf{S}_k$, which is done by pruning the network structure as follows: we delete edges outgoing from nodes in evidence $\mathbf{E}$ and hypothesis D, and delete (successively) all leaf nodes that are neither in $\mathbf{H}$, $\mathbf{E}$ or D. We then identify the components $\mathbf{X}_1, \ldots, \mathbf{X}_k$ of the resulting network and define each non-empty $\mathbf{S}_i = \mathbf{H} \cap \mathbf{X}_i$ as an element of the partition. This guarantees that in the original network structure, $\mathbf{S}_i$ is d-separated from $\mathbf{S}_j$ by D and $\mathbf{E}$ for $i \neq j$ (see [Darwiche, 2009]). In Figure 4, network pruning leads to the components $\mathbf{X}_1 = \{X_1, X_2, E_2, H_1, H_2, H_3\}$, $\mathbf{X}_2 = \{D, E_1, H_4\}$ and $\mathbf{X}_3 = \{X_3, H_5, H_6\}$.

Computing $O(d \mid \mathbf{e})$, $Pr(\mathbf{q} \mid \mathbf{e})$ and $w_{\mathbf{s}_i}$

These quantities, which are referenced on Lines 2–4 of the algorithm, have simple closed forms in Naive Bayes networks. For arbitrary networks, however, computing these quantities requires inference which we do using the algorithm of variable elimination as described in [Darwiche, 2009]. Note that network pruning, as discussed above, guarantees that each factor used by variable elimination will have all its variables in some component $\mathbf{X}_i$. Hence, variable elimination can be applied to each component $\mathbf{X}_i$ in isolation, which is sufficient to obtain all needed quantities. We omit the details of these computations, however, for space limitations.

Computing the Min and Max of Evidence Weights

We finally show how to compute $\max_{\mathbf{s}_i} w_{\mathbf{s}_i}$ and $\min_{\mathbf{s}_i} w_{\mathbf{s}_i}$, which are referenced on Lines 2 and 3 of the algorithm. These quantities can also be computed using variable elimination, applied to each component $\mathbf{X}_i$ in isolation. In this case, however, we must eliminate variables $\mathbf{X}_i \setminus \mathbf{S}_i$ first and then variables $\mathbf{S}_i$. Moreover, the first set of variables is summed-out, while the second set of variables is max'd-out or min'd-out, depending on whether we need $\max_{\mathbf{s}_i} w_{\mathbf{s}_i}$ or $\min_{\mathbf{s}_i} w_{\mathbf{s}_i}$. Finally, this elimination process is applied twice, once with evidence $d, \mathbf{e}$ and a second time with evidence $\overline{d}, \mathbf{e}$. Again, we leave out the details of this computation for space limitations.

3.4 Complexity Analysis

Let n be the number of variables in the network, $h = |\mathbf{H}|$, and $w = \max_i w_i$, where w_i is the width of constrained elimination order used on component $\mathbf{X}_i$. The best-case time complexity of our algorithm is then $\mathrm{O}\big(n \exp w\big)$ and the worst-case time complexity is $\mathrm{O}\big(n \exp (w + h)\big)$. The intuition behind these bounds is that computing the maximum and minimum weights for each aggregate attribute takes time $\mathrm{O}\big(n \exp w\big)$. This also bounds the complexity of computing $O(d|\mathbf{e})$, $Pr(\mathbf{q}|\mathbf{e})$ and corresponding weights $w_{\mathbf{s}_i}$. Moreover, depending on the weights and the threshold T, traversing the search tree can take anywhere from constant time to $\mathrm{O}\big(\exp h\big)$. Since depth-first search can be implemented with linear space, the space complexity is $\mathrm{O}\big(n \exp w\big)$.

4 Experimental Results

We performed several experiments on both real and synthetic networks to test the performance of our algorithm across a wide variety of network structures, ranging from simple Naive Bayes networks to highly connected networks. Real networks were either learned from datasets provided by the UCI Machine Learning Repository or provided by HRL Laboratories and CRESST.[3] For the majority of the real networks, it was clear which variable should be selected as the decision variable. For the unclear cases, the decision variable was picked at random. Query and evidence variables were selected at random for all real networks.

Besides this algorithm, there are two other options available to compute the SDP: 1. the *naive* method to brute–force the computation by enumerating over all possible instantiations or 2. the *approximate* algorithm developed by [Choi *et al.*, 2012]. To compare our algorithm with these two other approaches, we compute the SDP over the real networks. For each network we selected at least 80% of the total network variables to be query variables so that we could emphasize how the size of the query set greatly influences the computation time. Each computation was given 20 minutes to complete. As we believe that the value of the threshold can greatly affect running time, we computed the SDP with thresholds $T = [0.01, 0.1, 0.2, \ldots, 0.8, 0.9, 0.99]$ and took the worst-case time. The results of our experiments with the three algorithms are shown in Table 4. Note that $|\mathbf{H}|$ is the number of query variables and $|\mathbf{h}|$ is the number of instantiations the naive algorithm must enumerate over. Moreover, ϕ indicates that the computation did not complete in the 20 minute time limit and * indicates that there was not sufficient memory to complete the computation. The networks $\{$*car,ttt,voting,nav,chess*$\}$ are Naive Bayes networks whereas the others are polytree networks.

Given the real networks that we tested our algorithm on, it is clear that the algorithm outperforms both the naive implementation and the approximate algorithm for both Naive Bayes networks and polytree networks. Note that the approximation algorithm is based on variable elimination but can only use certain constrained orders. For a Naive Bayes network with hypothesis D being the root, the approximation

[3] http://www.cse.ucla.edu/

Table 4: Algorithm comparison on real networks. We show the time, in seconds, it takes each algorithm, *naive*, *approx*, and *new* to compute SDP in different networks.

Network	source	$\lvert\mathbf{H}\rvert$	$\lvert\mathbf{h}\rvert$	*naive*	*approx*	*new*
car	UCI	6	144	0.131	0.118	0.049
emdec6g	HRL	8	256	0.407	0.245	0.294
tcc4e	HRL	9	512	0.470	0.257	0.149
ttt	UCI	9	19683	6.234	0.133	0.091
caa	CRESST	14	16384	6.801	0.145	0.167
voting	UCI	16	65536	21.35	0.176	0.128
nav	CRESST	20	1572864	642.88	0.856	0.178
fire	CRESST	24	16777216	ϕ	0.183	0.508
chess	UCI	30	1610612736	ϕ	*	15.53

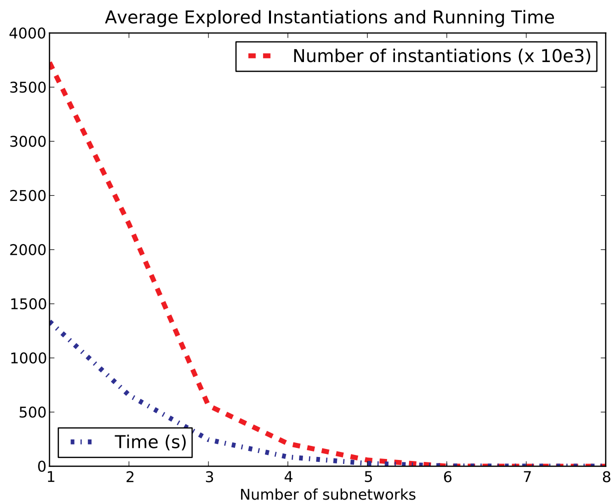

Figure 5: Synthetic network average running time and average number of instantiations explored by number of connected components.

algorithm will be forced to use a particularly poor ordering, which explains its failure on the `chess` network.

To analyze how a more general network structure and the selected threshold affects the performance of our algorithm, we generated synthetic networks with 100 variables and varying treewidth using *BNGenerator* [Ide *et al.*, 2004]. For each network, we randomly selected the decision variable, 25 query variables, and evidence variables.[4] We then generated a partition for each network and grouped the networks by the size of obtained partition (k). Our goal was to test how our algorithm's running time and ability to prune the search–space depends on k. The average time and average number of instantiations explored are shown in Figure 5.

In general, we can see that as k increases, the number of instantiations explored by the algorithm decreases and its runtime improves. The network becomes more similar to a Naive Bayes structure with increasing k. Moreover, the larger k is, the more levels there are in the search tree, which means that our algorithm will have more opportunities to prune. In the worst case, a network may be unable to be disconnected at all ($k = 1$). However, even in this case our algorithm is still, on average, more efficient compared to the brute-force implementation as for some cases, after computing the maximum and minimum weight of observing $\mathbf{H}$, it will find that there does not exist any $\mathbf{h}$ that will change the decision. We found that, given a time limit of 2 hours, the brute-force algorithm could not solve any synthetic networks whereas our approach solved greater than 70% of such networks.

[4] As the synthetic networks are binary, a brute–force approach would need to explore 2^{25} instantiations.

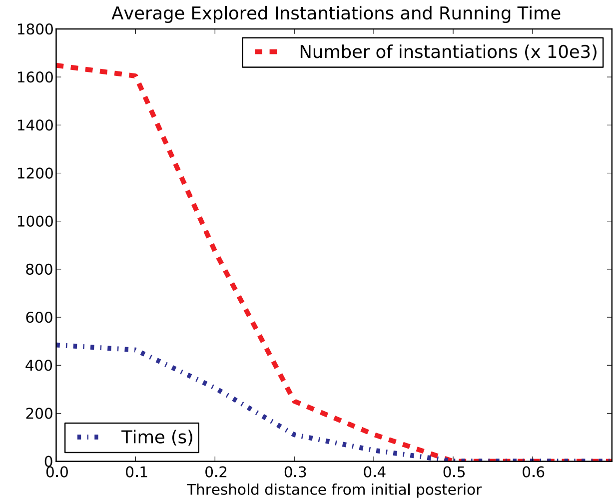

Figure 6: Synthetic network average running time and average number of instantiations explored by threshold distance from the initial posterior probability.

We also test how the threshold affects computation time. Here, we calculate the posterior probability of the decision variable and then run repeatedly our algorithm with thresholds that are varying increments away. The average running time for all increments can be seen in Figure 6. It is evident that when the threshold is set to be further away from the initial posterior probability, the algorithm finishes much faster, which is perhaps expected since the usage of more extreme thresholds would allow for more search space pruning.

Overall, our experimental results show that our algorithm is able to solve many SDP problems that are out of reach of existing methods. We also confirm that our algorithm completes much faster when the network can be disconnected or when the threshold is far away from the initial posterior probability of the decision variable.

5 Conclusion

We have introduced the first exact algorithm for computing the SDP. Experimental results show that this algorithm has comparable running time to the previous approximate algorithm and is also much faster than the naive brute-force algorithm. Our algorithm is able to compute the SDP for query sets that previous algorithms are unable to solve.

Acknowledgements

This work has been partially supported by ONR grant #N00014-12-1-0423, NSF grant #IIS-1118122, and NSF grant #IIS-0916161.

References

[Bilgic and Getoor, 2011] Mustafa Bilgic and Lise Getoor. Value of information lattice: Exploiting probabilistic independence for effective feature subset acquisition. *Journal of Artificial Intelligence Research (JAIR)*, 41:69–95, 2011.

[Chan and Darwiche, 2003] Hei Chan and Adnan Darwiche. Reasoning about bayesian network classifiers. In *UAI*, pages 107–115, 2003.

[Chen *et al.*, 2012] Suming Chen, Arthur Choi, and Adnan Darwiche. The same-decision probability: A new tool for decision making. In *Proceedings of the Sixth European Workshop on Probabilistic Graphical Models*, pages 51–58. PGM, 2012.

[Choi *et al.*, 2012] Arthur Choi, Yexiang Xue, and Adnan Darwiche. Same-decision probability: A confidence measure for threshold-based decisions. *International Journal of Approximate Reasoning (IJAR)*, 2, 2012.

[Darwiche and Choi, 2010] Adnan Darwiche and Arthur Choi. Same-decision probability: A confidence measure for threshold-based decisions under noisy sensors. In *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models*, pages 113–120. PGM, 2010.

[Darwiche, 2009] Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[Dittmer and Jensen, 1997] Soren Dittmer and Finn Jensen. Myopic value of information in influence diagrams. In *Proceedings of the Thirteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 142–149, 1997.

[Friedman *et al.*, 1997] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. In *Machine Learning*, pages 131–163, 1997.

[Greiner *et al.*, 1996] Russell Greiner, Adam J. Grove, and Dan Roth. Learning active classifiers. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML96*, 1996.

[Heckerman *et al.*, 1993] David Heckerman, Eric Horvitz, and Blackford Middleton. An approximate nonmyopic computation for value of information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(3):292–298, 1993.

[Heckerman *et al.*, 1995] David Heckerman, John S. Breese, and Koos Rommelse. Decision-theoretic troubleshooting. *Commun. ACM*, 38(3):49–57, March 1995.

[Ide *et al.*, 2004] Jaime S. Ide, Fabio G. Cozman, and Fabio T. Ramos. Generating random bayesian networks with constraints on induced width. In *In Proceedings of the 16th Eureopean Conference on Artificial Intelligence*, pages 323–327, 2004.

[Karp, 1972] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[Klonoff and Landrine, 1997] Elizabeth Adele Klonoff and Hope Landrine. Introduction. In *Preventing Misdiagnosis of Women: A Guide to Physical Disorders That Have Psychiatric Symptoms*, page xxi. SAGE Publications, Inc., 1997.

[Krause and Guestrin, 2009] Andreas Krause and Carlos Guestrin. Optimal value of information in graphical models. *Journal of Artificial Intelligence Research (JAIR)*, 35:557–591, 2009.

[Liao and Ji, 2008] Wenhui Liao and Qiang Ji. Efficient non-myopic value-of-information computation for influence diagrams. *Int. J. Approx. Reasoning*, 49(2):436–450, 2008.

[Lu and Przytula, 2006] Tsai-Ching Lu and K. Wojtek Przytula. Focusing strategies for multiple fault diagnosis. In *FLAIRS Conference*, pages 842–847, 2006.

[Park and Darwiche, 2004] James D. Park and Adnan Darwiche. Complexity Results and Approximation Strategies for MAP Explanations. *J. Artif. Intell. Res. (JAIR)*, 21:101–133, 2004.

[Pauker and Kassirer, 1980] Steven G. Pauker and Jerome P. Kassirer. The threshold approach to clinical decision making. *N Engl J Med*, 302(20):1109–17, 1980.

[van der Gaag and Bodlaender, 2011] Linda C. van der Gaag and Hans L. Bodlaender. On stopping evidence gathering for diagnostic bayesian networks. In *ECSQARU*, pages 170–181, 2011.

[van der Gaag and Coupé, 1999] Linda C. van der Gaag and Veerle M. H. Coupé. Sensitive analysis for threshold decision making with bayesian belief networks. In *AI*IA*, pages 37–48, 1999.