When Plans Distinguish Bayes Nets

Alex Dekhtyar
dekhtyar@cs.uky.edu
Department of Computer Science
University of Kentucky

Judy Goldsmith
goldsmit@cs.uky.edu
Department of Computer Science
University of Kentucky

Janice L. Pearce*
pearce@berea.edu
Department of Mathematics and Computer Science
Berea College

December 31, 2001

Abstract

We consider the complexity of determining whether differing probability distributions for the same Bayes net result in different plans, significantly different plan outcomes or optimal value functions.

Subarea: *belief networks*.

Keywords: *Bayes nets, information integration, computational complexity, stochastic planning*.

1 Introduction

Bayes nets model stochastic processes such as medical systems, military scenarios, academic advising, and a variety of industrial systems including nuclear power plant management and robotics applications. However, the process of modeling such a system often leads to interesting dilemmas, such as which differing expert opinion to believe, or how to reconcile the difference between expert opinion and knowledge extracted via automated means. The assumption that everyone is right is shown here to be computationally difficult to verify.

In order to model a system as a Bayes net, the system states must be factored into nodes (a.k.a. parameters or variables), so that each node depends

---

*Work done while Dr. Pearce was enrolled at the University of Kentucky, Department of Computer Science.

on some, preferably small, subset of other nodes. These dependencies are described by a directed, acyclic graph with conditional probability tables associated with each node having non-zero in-degree. (The probability tables may be represented by other data structures for ease of computation.)

All of the applications listed above assume a controller, whether a human decision maker or a software agent. While Bayes nets can be used for probabilistic inference or decision-theoretic planning, this work focuses on planning. *Probabilistic inference* is the means of determining the probable causes of observed effects. *Decision-theoretic planning* is the process of choosing actions in order to maximize the probability of the system achieving a desirable state or optimizing a utility function. The set of desirable states or the utility function must be defined and distinct actions must be specified. Each action determines its own conditional probabilities (described here as conditional probability tables) for the system. Thus, the choice of an action at each time step determines the probabilistic evolution of the system for that step.

A *plan* is a mapping from system states to actions; if the system states are not completely observable, then it is a mapping from beliefs about the current state of the system (probability distributions over the states) to actions. Plans are designed to maximize the probability of (eventually or quickly) reaching some prespecified goal states or optimizing some utility over time. A *planning algorithm* is a function that, given a Bayes net, produces a plan.

There are two basic approaches to building a Bayes net: automated learning from data or knowledge engineering from human experts. In some cases, fusing the information gleaned from both types of sources seems attractive. However, probabilities gleaned from different methods or even different experts may disagree. In fact, estimates from different experts might not only often differ from one another, but might vary from day to day from the same expert. In some situations, such as medical applications, there may be a plethora of data, though different data sets may yield different models. In others, such as military modeling, there are situations that should not be tested. In those cases, the modelers must depend on expert opinions. It seems reasonable that more than one expert should be consulted in matters of life or death (medicine, military applications). What if the experts disagree?

When a number of potentially conflicting probability estimates for the same Bayes net graph structure exists, it is the duty of Bayes net designers to produce a consistent and reasonable probability distribution. The first question that should be addressed is whether there is an actual conflict when probabilities disagree or whether the distributions are in some sense equivalent. For instance, one solution is to determine whether or not the differences are too minor to matter. In the case of a few scattered differences, one can use methods like sensitivity analysis [12, 4] (see also Section 7) to determine whether or not variations affect the consequent variables significantly. However, if the entire set of conditional probability tables differs from one source to another, we suggest other notions of equivalence.

Before we go into the details of these equivalence notions and their computational complexity, consider how the underlying problem might arise: How does one construct a Bayes net model of a stochastic system?

In many applications where Bayes nets are employed, the process of building the actual Bayes net is often a three step process:

1. The set of random variables in the application domain is specified.
2. The *structure* of the Bayes net is established: the dependences between individual random variables are established.
3. Given the Bayes net structure, the *probability distributions* associated with each of the nodes or random variables of the net are determined.

In a typical scenario, Bayes net designers use human experts to elicit probability estimates for the random variables in the network. When estimates differ, the data must be fused or integrated by somehow resolving conflicts. The methodology of this "data fusion" or "conflict resolution" process for uncertain data is not very well studied. The largest body of work on the subject is multisensor fusion used in vision, robotics and speech recognition [1, 7, 17, 15]. Typically the approaches described there, as well as in other research on fusing probabilities [6, 5, 18] work when applied to a particular problem or a narrow set of problems, and do not translate immediately to other problems or to the general case. Furthermore, much of the work in multisensor fusion assumes that the fusion is an ongoing, online process. Our work presumes an offline decision on whether or not to fuse static probability distributions.

How hard is it to determine whether the differences in the probability distributions affect the outcomes of particular planning algorithms? When the differences do not change the outcome, then we will say that they are not in conflict. Our main idea is that *if we can determine that the resulting plans or plan results for the Bayes nets are sufficiently similar, then we need not go through the complex process of integrating the probabilistic data.* Thus, the problem we address is the following:

> Given two sets of probability distributions for a Bayes net, when can we sidestep the fusion problem?

In order to answer this question, we need to formalize the notion of *equivalence* of Bayes nets. In what follows, we describe three such notions with respect to planning algorithms.

We ask when two sets of conditional probability tables for the same Bayes net structure are *equivalent* for planning purposes under the following notions of equivalence:

1. Does a given planning algorithm produce the same plan for each version of the Bayes net?
2. Does a given plan or the outputs of a planning algorithm have the same probability of success in each case?
3. Do the optimal plans for each case produce the same value function?

We show that the first problem is *coNP-complete*, and the second and third are $\mathrm{coNP}^{\mathrm{PP}}$*-hard*. We also give restrictions on either the type of plan considered or the structure of the Bayes net that make these problems easy.

The rest of the paper is organized as follows. Section 2 gives formal definitions, and Section 3 illustrates some of these definitions with a small example which suggests that telling two Bayes networks apart using our definitions of equivalence with respect to plans is not simple. In Sections 4 and 5 we consider the different notions of Bayes net equivalence mentioned above and provide the complexity results for the problem of determining whether two Bayes Nets are equivalent with respect to those notions. In Section 6 we note that simpler solutions may be developed for some restrictioned versions of our problems. Section 7 summarizes related work.

2 Definitions

In this section, we formally define Bayes nets for the purpose of planning and planning algorithms.

Definition 1 *Let* $V = \{v_1, \ldots, v_n\}$ *be a set of* random variables *with domains* $dom(v_1), \ldots, dom(v_n)$ *respectively, all finite.*

A Bayes net B *consists of a directed acyclic graph (DAG)* $G = (V, E)$*, and a set of conditional probability distribution functions* $P_v : dom(parents(v)) \times dom(v) \to [0, 1]$ *for each* $v \in V$*, such that for each set of values of* $parents(v)$*,* P_v *defines a probability distribution over* $dom(v)$*.*

A dynamic Bayes net *or two-phase temporal Bayes net is a Bayes net on a directed, acyclic* bipartite *graph* $G = (V, V, E)$*, where* $|V| = |V'|$ *and the nodes in* V *have in-degree 0. The nodes in* V' *represent the same state variables as those in* V*, but at a subsequent time step. This allows us to address the frame problem: How does a node's current state affect its next state? It also allows us to explicitly model Markov decision processes.*

A total state s *of* B *is a* total *function* $s : V \to \cup_{i=1}^{n} dom(v_i)$ *such that* $s(v_i) \in dom(v_i)$*, for all* $1 \le i \le n$*. Let* $\mathcal{S}$ *be the set of all possible states of the system.*

A partial state s' *of* B *is a* partial *function* $s' : V \to \cup_{i=1}^{n} dom(v_i)$ *such that* $s'(v_i) \in dom(v_i)$*, for all* $1 \le i \le n$*.*

We often abuse terminology by referring to both total and partial states as simply states. Note that $|\mathcal{S}|$ is generally exponentially larger than the number of random variables $|V|$.

We can interpret a function P_v as a probabilistic function from the values of $parents(v)$ to the values of v. In this interpretation, the collection of P_vs models the probabilistic evolution of states of the Bayes net.

The most common use of Bayes nets is for inference: given a set of values of some nodes, compute the most likely values of other nodes (either parents or children). However, we are interested in the use of Bayes Nets for *planning*.

We consider the extension of Bayes net models that allows us not only to model the probabilistic evolution of the system, but also to model an outside controller that can affect that evolution. As the means of modelling possible effects of an outside controller on the system, we employ *actions*, defined formally below.

Definition 2 *Let $G = (V, E)$ be a directed acyclic graph. An action a on G is a set of conditional probability distribution functions $P_{a,v} : dom(parents(v)) \times dom(v) \to [0, 1]$ for each $v \in V$.*

A Bayes net for planning purposes *B is a tuple $\langle G = (V, E), A, \mathcal{F}\rangle$, where G is a DAG, A is a set of stochastic actions on G and $\mathcal{F}$ is a set of states called* goal states*.*

A Bayes net may also model a Markov decision process, by including an explicit utility function *$r : \mathcal{S} \to \mathbf{R}$ instead of $\mathcal{F}$.*

While simple Bayes nets are useful in modelling situations for which we have no control, Bayes nets for the purpose of planning are useful when there is an opportunity to affect the outcome of a process. For the rest of this paper, we will use "Bayes net" to refer to a Bayes net for planning purposes. Note that the initial definition of a Bayes net corresponds to a Bayes net in the extended definition with only one action, where $\mathcal{F}$ is the set of all states.

We describe the set of choices a controller makes as a *plan*. For the purposes of this paper, we consider the simplest form of plans, namely mappings from states to actions. This imposes consistency on the controller's choices.

Definition 3 *Given a Bayes net B, a* plan *for B is a mapping $\pi : \mathcal{S} \to \mathcal{A}$.*

A planning algorithm $\mathcal{A}$ is a function that maps Bayes nets to plans for those Bayes nets.

Other definitions of plans exist in the literature, including time-dependent and history-dependent plans. The hardness proofs in this paper can be easily modified to give the same complexity results for time-dependent plans.

The task of a controller is either to maximize the probability of reaching a goal state or to optimize the expected reward over time. Note that, in many cases, the optimization criterion is a discounted expected reward: All rewards at future steps are discounted by a factor of $\gamma \in [0, 1]$.

Note that the problem of finding an optimal plan for general Bayes nets is exp-complete [13]. However, for this paper, we tend to assume that the planning algorithms, are tractable. This implies either that the Bayes nets fall into some special subclass, or that the planning algorithms are approximate.

In particular, we often assume tractable planning algorithms $\mathcal{A}$, those such that, for any Bayes net B and state s of B, computing the action $\mathcal{A}(B)(s)$ can be done in time polynomial in $|B|$.

3 Example

Consider the following example which features a small (fictitious) horseback riding summer resort. The resort consists of a number of trails and

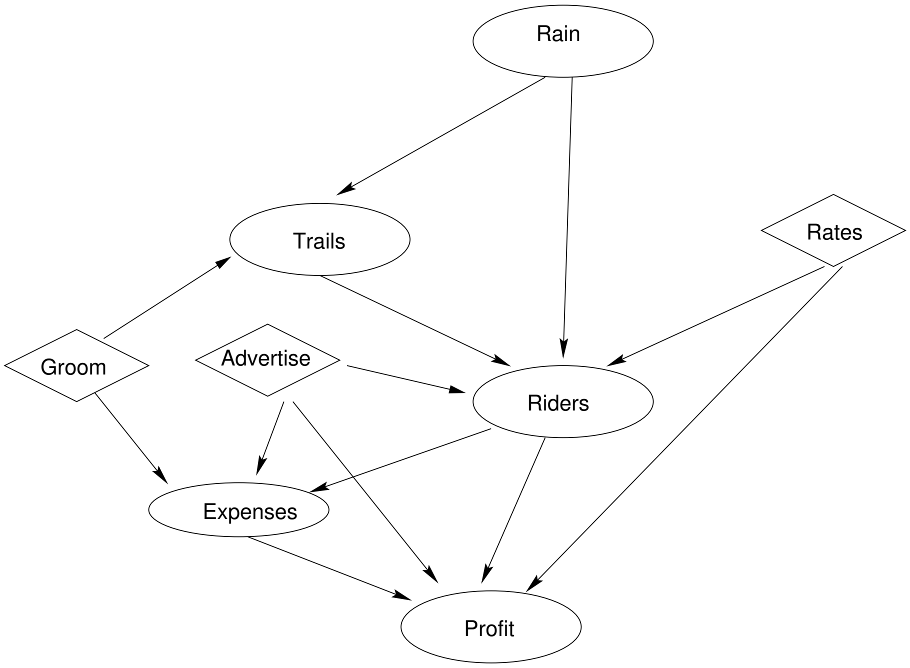


Figure 1: Sample Bayes net: Horseback Riding Resort

Node: rain		
$P(\text{rain} = yes)$		
Set 1	Set 2	Set 3
0.2	0.2	0.2

Node: trails				
		$P(\text{trails} = fair)$		
groom	rain	Set 1	Set 2	Set 3
no	*no*	0.6	0.6	0.6
no	*yes*	0.3	0.3	0.3
yes	*no*	1.0	1.0	1.0
yes	*yes*	0.8	0.8	0.8

Node: expenses					
			$P(\text{expenses} = high)$		
groom	advertise	riders	Set 1	Set 2	Set 3
no	*no*	*low*	0.35	0.35	0.35
no	*no*	*high*	0.5	0.5	0.5
no	*yes*	*low*	0.5	0.5	0.5
no	*yes*	*high*	0.65	0.65	0.65
yes	*no*	*low*	0.45	0.45	0.45
yes	*no*	*high*	0.6	0.6	0.6
yes	*yes*	*low*	0.65	0.65	0.65
yes	*yes*	*high*	0.75	0.75	0.75

Table 1: Sets of Conditional Probability Distributions for Horseback Riding Resort Bayes Nets (Part I).

Node: riders						
				P(riders $=$ *high*)		
rates	advertise	trails	rain	Set 1	Set 2	Set 3
low	*no*	*poor*	*no*	0.5	0.5	0.5
low	*no*	*poor*	*yes*	**0.3**	**0.35**	0.3
low	*no*	*fair*	*no*	**0.9**	**0.85**	0.9
low	*no*	*fair*	*yes*	**0.6**	**0.55**	0.6
low	*yes*	*poor*	*no*	**0.9**	**0.85**	0.9
low	*yes*	*poor*	*yes*	**0.65**	**0.7**	**0.75**
low	*yes*	*fair*	*no*	**0.95**	**0.9**	0.95
low	*yes*	*fair*	*yes*	**0.8**	**0.75**	0.8
high	*no*	*poor*	*no*	**0.25**	**0.3**	0.25
high	*no*	*poor*	*yes*	**0.2**	**0.25**	0.2
high	*no*	*fair*	*no*	0.7	0.7	0.7
high	*no*	*fair*	*yes*	0.4	0.4	0.4
high	*yes*	*poor*	*no*	**0.35**	**0.3**	0.35
high	*yes*	*poor*	*yes*	0.3	0.3	0.3
high	*yes*	*fair*	*no*	0.8	0.8	0.8
high	*yes*	*fair*	*yes*	0.5	0.5	0.5

Node: profits						
				P(profits $=$ *high*)		
rates	advertise	expenses	riders	Set 1	Set 2	Set 3
low	*no*	*low*	*low*	**0.45**	0.45	**0.4**
low	*no*	*low*	*high*	**0.65**	0.65	**0.6**
low	*no*	*high*	*low*	**0.35**	0.35	**0.3**
low	*no*	*high*	*high*	**0.45**	0.45	**0.4**
low	*yes*	*low*	*low*	**0.45**	0.45	**0.4**
low	*yes*	*low*	*high*	**0.65**	0.65	**0.6**
low	*yes*	*high*	*low*	**0.35**	0.35	**0.3**
low	*yes*	*high*	*high*	**0.55**	0.55	**0.5**
high	*no*	*low*	*low*	**0.55**	0.55	**0.65**
high	*no*	*low*	*high*	**0.75**	0.75	**0.85**
high	*no*	*high*	*low*	**0.4**	0.4	**0.45**
high	*no*	*high*	*high*	**0.5**	0.5	**0.55**
high	*yes*	*low*	*low*	**0.55**	0.55	**0.65**
high	*yes*	*low*	*high*	**0.75**	0.75	**0.85**
high	*yes*	*high*	*low*	**0.4**	0.4	**0.45**
high	*yes*	*high*	*high*	**0.6**	0.6	**0.65**

Table 2: Sets of Conditional Probability Distributions for Horseback Riding Resort Bayes Nets (Part II).

a stable compound which attract horse owners for recreational purposes. The owners of the resort are interested in maximizing their profit, which is determined by the rates they charge the riders for the use of their stables and trails as well as by the number of riders that visit the resort over a period of time. The profit is also adversely affected by the expenses that the owners incur. In this simple example, we assume that the number of riders visiting the resort is influenced by the weather conditions, state of the trails, and by the current trail and stable rates. The owners may choose to spend the money on advertising the resort, and on grooming trails: both actions possibly resulting in an increased number of visitors.

This situation can formally be represented as a Bayes net depicted in Figure 1. Two types of variables are present in the network: *random variables* and *action variables*, the latter identified by diamonds around them in the Figure. The values of random variables are determined stochastically based on the values of other variables in the network. The values of action variables are set by the *actions* (decisions) of resort owners and thus are deterministic. The random variables are:

- rain: the *presence* or *absence* of rain or other inclement weather;
- trails: *fair* or *poor* condition of trails;
- riders: *high* or *low* ridership on trails;
- expenses: *high* or *low* expenses to support the resort operation;
- profit: at the end of the period, the resort can turn in *low* or *high* profit.

The action variables are:

- rates: *high* or *low* rates charged for the use of the stable and trails;
- groom: indicates whether the trails are *groomed* or *not*.
- advertise: indicates whether the resort is *advertising* to the potential visitors or *not*.

All variables are binary for simplicity[1]. As mentioned above, the owners of the resort can take certain actions in order to affect the situation with the goal of maximizing the probability of turning high profit. These actions are represented in the model as the assignment of values to the action variables rates, groom and advertise. The actions affect the conditional probability distributions associated with certain nodes of the Bayes net. An atomic action is the assignment of a value to a single action variable. Values to different action variables are assigned independently, thus there are *eight* compound actions that the owners may choose. The effects of atomic actions are described in Table 3. Basically, each atomic action produces one

[1]The SPUDD planning algorithm [10, 9] that we have used in our example only works for binary variables.

Action	Implications
Rates	
Charge low rates	Increased probability of high ridership Decreased probability of high profit
Charge high rates	Increased probability of high profits Decreased probability of high ridership
Groom	
Do not groom trails	Decreased probability of fair condition of the trails Decreased probability of high expenses
Groom trails	Increased probability of fair condition of the trails Increased probability of high expenses
Advertise	
Do not advertise	Decreased probability of high ridership Decreased probability of high expenses
Advertise	Increased probability of high ridership Increased probability of high expenses

Table 3: Implications of Actions.

positive and one negative effect. Compound actions produce overlapping effects: e.g., *advertising* and *grooming trails* result in a high probability of incurring *high expenses* but also in a high probability of attracting *high ridership* (both probabilities higher than those for individual actions) as well as in trails being in *fair* condition.

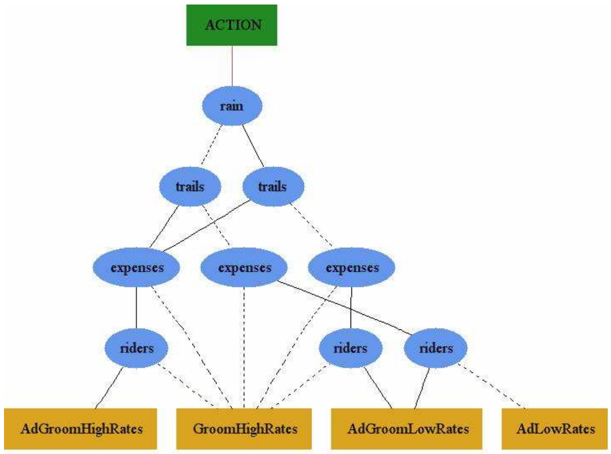
We have considered three different sets of conditional probability assignments associated with the Bayes net in Figure 1, which we will call Set 1, Set 2 and Set 3 CPTs. Tables 3 and 3 show all three sets of CPTs. The differences of Sets 2 and 3 from Set 1 are highlighted in boldface.

For our planning algorithm we have used the on-line system for planning under uncertainty SPUDD developed by Hoey, St.-Aubin, Hu and Boutiler [10, 9] to come up with the optimal plans for maximizing profit in our Horseback Riding Resort domain under each of the three sets of conditional probability tables[2].

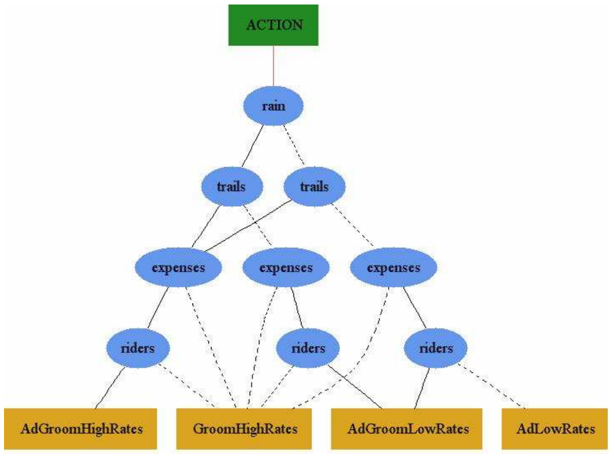
SPUDD output consists of two decision trees: the optimal plan DT and the value function (expected reward) DT. Figure 2 illustrates the optimal plans for all three sets of CPTs returned by SPUDD.

We can see that the first two CPTs (Sets 1 and 2) yield the same plan (the decision trees are rendered differently by SPUDD but are isomomorphic), whereas Set 3 yields a drastically different plan. On the other hand, the three value functions are all distinct. (They are not included in the text because the trees are wide enough that they cannot be rendered in readable

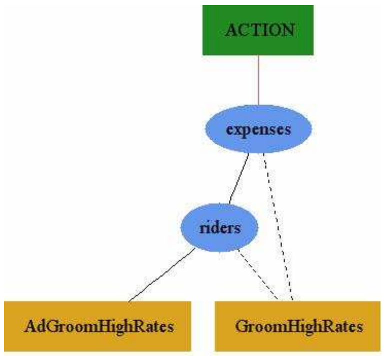
[2]SPUDD web page is http://www.cs.ubc.ca/spider/staubin/Spudd/index.html. At http://www.cs.ubc.ca/spider/staubin/Spudd/form.html one can run SPUDD interactively. We have used default SPUDD settings The data files describing our Horseback Riding Resort example, as well as SPUDD output can be obtained from http://www.cs.uky.edu/∼dekhtyar/dblab/fusion.html; the data file for Set 1 is also included in Appendix A.



(a)



(b)



(c)

Figure 2: Decision trees for SPUDD optimal plans for Sets 1 (a), 2 (b) and Set 3 (c) CPTs in Horseback Riding Resort example.

formats. Interested readers can find the decision trees for the optimal value functions at http://www.cs.uky.edu/ dekhtyar/dblab/fusion.html.)

Careful inspection of the three sets of probability tables might not reveal these outcomes. This is not surprising, given the intractibility of distinguishing Bayes nets, as shown in what follows.

It is not, however, surprising that the same plan applied to Bayes nets with different CPTs would yield different value functions. One might expect it to be difficult to find different CPTs that yield *the same* value function for any but the crudest plans.

It is also interesting to note how three different notions of Bayes net equivalence with respect to planning relate to this example. Suppose the owners ask three different consultants to evaluate the operation of the resort, resulting in Sets 1–3, respectively. The owners must make an information fusion decision: choose one set or combine them somehow. What they choose depends on what they want: advice on what actions to take, or expected profit under some pre-determined plan, or perhaps the optimal expected profit. (If they are considering bankruptcy, the bank might for instance be interested in the latter.) These three criteria correspond precisely to the three definitions of Bayes net equivalence under planning discussed in this paper.

Because the example considered here is so small, we have used a planning algorithm that finds optimal plans, so the optimal value and plan value are the same here.

4 Detecting Identical Plans

We first consider detecting *identical plans*.

Definition 4 *Let $B_1 = \langle G, P_1\rangle$ and $B_2 = \langle G, P_2\rangle$ be two Bayes nets over the same graph G. Let $\mathcal{A}$ be some planning algorithm. The triple $\langle B_1, B_2, \mathcal{A}\rangle$ is in $BNPlan$* ***iff*** *for every state s of B_1 and B_2,*[3]

$$\mathcal{A}(B_1, s) = \mathcal{A}(B_2, s).$$

Intuitively, two Bayes nets over the same structure belong to the class $BNPlan$ together with the planning algorithm $\mathcal{A}$ if the algorithm *does not distinguish* between the Bayes nets.

When the planning algorithm is fixed, we will denote as $BNPlan^{\mathcal{A}}$ the set of pairs of Bayes nets $\langle B_1, B_2\rangle$ such that for all states s, $\mathcal{A}(B_1, s) = \mathcal{A}(B_2, s)$.

Also, given the graph structure G of a family of Bayes nets BN_G, we can consider the sets $BNPlan_G \subset BNPlan$ and $BNPlan^{\mathcal{A}}_G \subset BNPlan^{\mathcal{A}}$ such that both Bayes nets come from BN_G.

First, we study the problem of the complexity of determining that a triple $\langle B_1, B_2, \mathcal{A}\rangle$ belongs to class $BNPlan$. The following results combined together show that this problem is coNP-complete.

[3]Since both B_1 and B_2 are Bayes nets over the same graph, their sets of nodes, and thus of states, coincide.

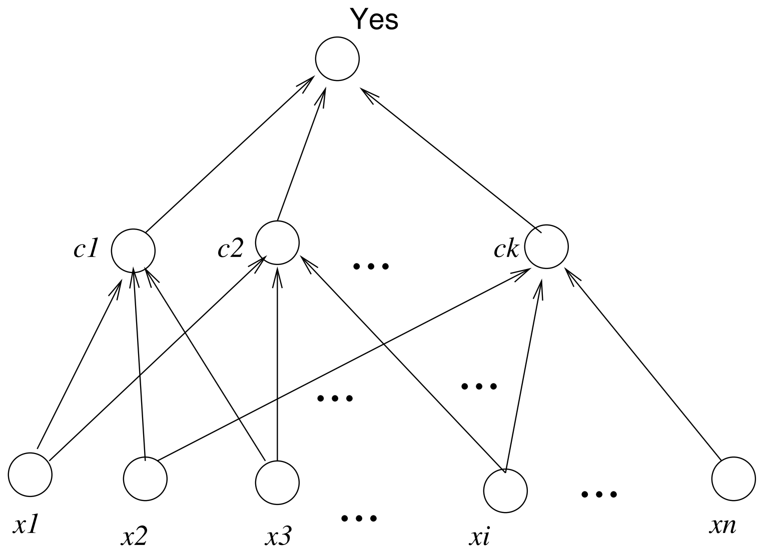


Figure 3: Bayes Net for Satisfiability problem.

Theorem 1 *The $BNPlan$ problem is* coNP-hard.

Proof We consider the dual problem of determining $\langle B_1, B_2, \mathcal{A}\rangle \notin BNPlan$ and will show that this problem is NP-hard.

Let φ be a propositional boolean formula over variables $x_1, \ldots, x_n$. Without loss of generality we can consider φ to be in 3CNF.

Given φ, we will construct a pair of Bayes nets B_1^φ and B_2^φ (we will omit the superscripts where this does not generate ambiguity) and a planning algorithm $\mathcal{A}$ such that $\varphi \in SAT$ **iff** $\langle B_1^\varphi, B_2^\varphi, \mathcal{A}\rangle \notin BNPlan$.

The Bayes net graph structure G underlying both B_1 and B_2 is shown in Figure 3. The net will consist of three levels of nodes which take on the values of 0 and 1 (false and true). At the lowest or initial level, a node will be associated with each of the variables $x_1, \ldots, x_n$ of φ. At the second or middle level, the nodes represent the clauses of φ. Each node c_i from the second level is connected to exactly three nodes representing boolean variables variables.

The third level of the network consists of one node, which we will label as Yes. Its parents will be all second level nodes $c_1, \ldots, c_k$.

The probability distribution P_1 of the Bayes net B_1 will faithfully simulate the computation of the truth value of φ. Each node $x_1, \ldots, x_n$ will have the same uniform distribution $P_1(x = 1) = P_1(x = 0) = 0.5$.

For each second level node c with parents x, x', and x'', the conditional probability table will simulate the truth table for the conjunct represented by c. E.g, if $c = x \vee \neg x' \vee x''$ the conditional probability table will be: $P_1(c = 0|x = 0, x' = 1, x'' = 0) = 1; P_1(c = 1|x = 0, x' = 1, x'' = 0) = 0$ (this is the only combination that makes c false) and $P_1(c = 0|x = a, x' = a', x'' = a'') = 0; P_1(c = 1|x = a, x' = a', x'' = a'') = 0$ for any other combination of truth values for x, x', and x''.

Finally, the probability distribution for the node Yes is defined as $P_1(\textsf{Yes} = 1|c_1 = 1, \ldots, c_k = 1) = 1; P_1(\textsf{Yes} = 0|c_1 = 1, \ldots, c_k = 1) = 0$ and $P_1(\textsf{Yes} = 1|c_1 = a_1, \ldots, c_k = a_k) = 0; P_1(\textsf{Yes} = 0|c_1 = a_1, \ldots, c_k = a_k) = 1$ for all other combinations of values of $c_1, \ldots, c_k$.

The probability distribution tables P_2 for the Bayes net B_2 are designed to mimic the probability distributions P_1 on the lower and middle layer of the nodes. For the node Yes, P_2 specifies that the probability of it becoming true is always 0.

The actions associated with these two Bayes nets correspond to setting the values of the lower level nodes (which represent the boolean variables). The goal states are those with the Yes node set to 1.

We can now specify the planning algorithm $\mathcal{A}$. Let state s of B_1 and B_2 be a sequence $(a_1, \ldots, a_n)$ of binary digits representing the truth values of variables $x_1, \ldots, x_n$. We denote the state of B_1 and B_2 which is a lexicographic successor of the binary string $a_1 a_2 \ldots a_n$ as $succ(s)$[4].

Algorithm $\mathcal{A}$ will leave the state s unchanged if s induces Yes = 1. Otherwise, it returns the set of actions that change current state of the BN B_1 or B_2 to the state $succ(s)$.

In order to see why this construction works we need the following:

1. The Yes node of B_1 is set to 1 iff the initial state s of B_1 corresponds to a satisfying assignment to the variables $x_1, \ldots, x_n$ of the formula φ.

2. The Yes node of B_2 will never be set 1.

3. The plans constructed by $\mathcal{A}$ for B_1 and B_2 will be identical on all states s unless there is a state s with Yes = 1.

From items 1 and 3 from the above we conclude that $\langle B_1^{\varphi}, B_2^{\varphi}, \mathcal{A}\rangle \notin BNPlan$ iff $\varphi \in SAT$. □

Theorem 2 *The* $BNPlan \in$ coNP.

Proof Notice that, for Bayes net B, $\mathcal{A}(B, s)$ is an action, and therefore $|\mathcal{A}(B, s)| \leq |B|$. To show that $\langle B_1, B_2, \mathcal{A}\rangle \notin BNPlan$, we need only guess state s such that $\mathcal{A}(B_1, s) \neq \mathcal{A}(B_2, s)$. This check is in polynomial time, and the guess is polynomial sized. Thus, $\overline{BNPlan} \in$ NP, so $BNPlan \in$ coNP. □

5 Detecting Identical Probabilities of Plan Success

In the previous section, we considered two Bayes nets to be equivalent, relative to a planning algorithm, if the algorithm produced the same plan. One can relax this definition and ask, does this planning algorithm or plan produce the same outcome, namely, the same probability of success?

We assume here that the specification of the Bayes net and/or the planning algorithm includes the specification of a set of goal states or utility

[4] We set $succ(1^n) = 0^n$.

function for each Bayes net. In what follows, we implicitly assume that the sets of goal states for the two Bayes nets in question are the same.

Definition 5 *Let $B = \langle G, P\rangle$ be a Bayes net with a given set of goal states. Let $\mathcal{A}$ be a planning algorithm, and s a state of B. Then $Succ(\mathcal{A}, B, s, h)$ is the defined to be the probability that the plan calculated by $\mathcal{A}$, when applied to state s of B and iterated for h steps, will reach a goal state.*

Rather than requiring that the success probabilities are identical for the two Bayes nets, we allow an extra parameter to specify how close they must be.

Definition 6 *Let $B_1 = \langle G, P_1\rangle$ and $B_2 = \langle G, P_2\rangle$ be two Bayes nets over the same graph G. Let $\mathcal{A}$ be some planning algorithm. The tuple $\langle B_1, B_2, \mathcal{A}, h, \theta\rangle$ is in $SBNPlan$ **iff** for every state s of B_1 and B_2, $|Succ(\mathcal{A}, B_1, s, h) - Succ(\mathcal{A}, B_2, s, h)| \leq \theta$.*

Note that we can also consider the value of a Bayes net relative to its optimal plan.

Definition 7 *Let $B = \langle G, P\rangle$ be a Bayes net with a given utility function. Let $V_B(s, h)$ be the optimal horizon h value function for B. Let $B_1 = \langle G, P_1\rangle$ and $B_2 = \langle G, P_2\rangle$ be two Bayes nets over the same graph G. The tuple $\langle B_1, B_2, h, \theta\rangle$ is in $BNValue$ **iff** for every state s of B_1 and B_2, $|V_{B_1}(s, h) - V_{B_2}(s, h)| \leq \theta$.*

Alternatively, one could further quantify $SBNPlan$ over all plans; depending on the complexity of plan descriptions allowed, this might easily increase the complexity of the problem.

Before stating the next theorem, we remind the reader of some facts about the class $\mathrm{NP}^{\mathrm{PP}}$.

$$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PH} \subseteq \mathrm{PP} \subseteq \mathrm{NP}^{\mathrm{PP}} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP}.$$

The set EMAJSAT is $\leq_m^P$-complete for $\mathrm{NP}^{\mathrm{PP}}$ [14]. A Boolean formula, natural number pair $\langle \varphi, k\rangle \in$ EMAJSAT iff there is a truth assignment to the first k variables of φ such that at least half of its completions satisfy φ.

Theorem 3 *The $SBNPlan$ problem is* $\mathrm{coNP}^{\mathrm{PP}}$-hard.

Proof To show this, we give a reduction from EMAJSAT. As in Theorem 1, we consider the dual problem of determining $\langle B_1, B_2, \mathcal{A}, h, \theta\rangle \notin SBNPlan$ and will show that this problem is $\mathrm{NP}^{\mathrm{PP}}$-hard.

Given a pair $\langle \varphi, k\rangle$, we construct an initial Bayes net B_1 as in the proof of Theorem 1. There are nodes in the Bayes net representing each variable and each clause, and an extra timekeeper node that turns the process off after one step.

Each clause node depends on the variable nodes for those variables it contains. The dependence is deterministic. Variable nodes have no parents.

The first k variable nodes are set by the user's action; the remaining variable nodes are set to 1 with probability $1/2$, independent of the action chosen or the prior state. The timekeeper node is dependent on all of the clause nodes. It has three states: starting, on and off. The system starts with it in starting. Any action sets it to on; any further actions set it to off. This enforces a horizon of 1.

The goal states are those in which all clause nodes are set to 1 and the timekeeper node is set to on.

In the first Bayes net, B_1, the clause nodes depend on the variable nodes in the manner indicated by φ. In the second Bayes net, B_2, the clause nodes are uniformly set to 0. Thus, B_2 can never reach a goal state.

Note that a plan for B_1 determines a setting of the first k variables. Furthermore, $\langle \varphi, k\rangle \in$ EMAJSAT if and only if there is some plan that reaches a goal state with probability at least $1/2$.

Consider the plan π which treats the current variable assignment $x_1, \ldots, x_k$ as a binary string and increments it by one, and maps 1^k to 0^k. If $\langle \varphi, k\rangle \in$ EMAJSAT then some setting of the first k variables witnesses that. Let s' the lexicographic predecessor of that setting (or let $s' = 1^k$, if that setting is 0^k), and let s be any state of the system that extends s'. Then $Succ(\mathcal{A}, B_1, s, 1) - Succ(\mathcal{A}, B_2, s, 1) \geq 1/2$.

Thus, $\langle \varphi, k\rangle \in$ EMAJSAT iff for some s, $Succ(\mathcal{A}, B_1, s, h) > \frac{1}{2}$ iff $\langle B_1, B_2, \mathcal{A}, h, \frac{1}{2}\rangle \notin SBNPlan$ for any $h \geq 1$. □

Corollary 1 *The problem $BNValue$ $\mathrm{coNP}^{\mathrm{PP}}$-hard.*

Proof Consider the construction in the proof of Theorem 3. Note that if A is an optimal plan for B_1, then $Succ(\mathcal{A}, B_1, s, 1) - Succ(\mathcal{A}, B_2, s, 1) \geq 1/2$ if and only if $\langle \varphi, k\rangle \in$ EMAJSAT. By definition, then, $V_{B_i}(s, h) = Succ(\mathcal{A}, B_i, s, 1)$. Thus, $\langle \varphi, k\rangle \in$ EMAJSAT if and only if $\langle B_1, B_2, h, \theta\rangle \in BNValue$. □

To show $\mathrm{coNP}^{\mathrm{PP}}$-completeness, we would need that $SBNPlan \in \mathrm{coNP}^{\mathrm{PP}}$. This requires that plan evaluation be in PP. Given that assumption, the theorem follows immediately from the appropriate characterization of the class $\mathrm{coNP}^{\mathrm{PP}}$.

In his dissertation, Jacobo Toran [19] considers the closure of the class P under polynomially length-bounded existential and universal quantifiers and under the counting quantifier $\mathbf{C}$. If K is a language class, then a language L is in $\mathbf{C}K$ if there is a $B \in K$, an $f \in$ FP, and a polynomial p such that

$$x \in L \Leftrightarrow |\{y : |y| \leq p(|x|) \text{ and } B(x, y)\}| \geq f(|x|).$$

In particular, for $f(x) = 1/2$, PP $= \mathbf{C}K$.

One can define the *Counting Hierarchy* over P analogously to the Polynomial Hierarchy: it is the closure of P under polynomially length bounded existential, universal, and counting quantifiers. Toran showed that, for any K in the counting hierarchy, $\mathrm{NP}^K = \exists^P K$. In particular,

$$\mathrm{NP}^{\mathrm{PP}} = \exists^P \mathrm{PP} = \exists^P \mathbf{C}P \text{ and } \mathrm{coNP}^{\mathrm{PP}} = \forall^P \mathbf{C}P.$$

This characterization gives us another method for showing membership in $\mathrm{NP}^{\mathrm{PP}}$ and $\mathrm{coNP}^{\mathrm{PP}}$, in terms of these quantifiers. Corollary 2 follows immediately from this.

Corollary 2 1. *If* $Succ(\mathcal{A}, B, s, 1) \in \mathrm{PP}$*, then* $SBNPlan \in \mathrm{coNP}^{\mathrm{PP}}$.

2. *If* $Val_B(s, h) \in \mathrm{PP}$ *then* $BNValue \in \mathrm{coNP}^{\mathrm{PP}}$.

6 Easier Cases

Although we have shown that, in the general case, it is computationally complex to decide whether two sets of conditional probability tables are equivalent for a Bayes net structure, there are some cases where these worst-case analyses do not apply. It is an open question to characterize other useful categories of Bayes nets and plans, but we give some preliminary thoughts on this.

For our analysis, let n be the number of nodes in the Bayes net in question. We consider Bayes nets where the number of actions available is $\mathcal{O}(n)$. We refer to any number $k = \mathcal{O}(\log n)$ as a *small number*.

Suppose for some network structure G, $\mathcal{A}$ generates a plan where each action is chosen based on a fixed, small set of nodes. Then for Bayes nets B_1 and B_2 over structure G, it is easy to determine whether $\langle B_1, B_2, \mathcal{A}\rangle \in BNPlan$: one merely enumerates all possible values of those few nodes and asks whether the plans are equivalent for those values.

For instance, if a plan in our Horseback Riding Resort example (see Section 3) is based solely on the number of riders then this case applies. This is perhaps too trivial an example. Imagine, instead, that plans are based on the number of riders and the weather. Then one need not consider the expenses incurred by the resort.

Suppose, instead, that we have a network structure G consisting of a number of disjoint subnetworks, each consisting of a small number of nodes. Suppose, in addition, that the function that determines the goal state can be expressed simply, for instance as the conjunction of a small number of independent node values. In this case, the probability of the conjunction is the minimum of the probabilities of achieving each given goal node value. Thus, $Succ(\mathcal{A}, B, s, h) \geq \theta$ if and only if the probability of "success" *for each goal variable* is at least θ. To determine this for all s, one need only consider the subnetworks containing goal variables, and evaluate the probability of success for each of the polynomially many states of that subnetwork.

Given two Bayes nets B_1 and B_2 over such a "factored" structure G, and given respective plans π_1 and π_2 generated by planning algorithm $\mathcal{A}$, it is not much harder to compute

$$\max_s |Succ(\mathcal{A}_1, B_1, s, h) - Succ(\mathcal{A}_2, B_2, s, h)|$$

than to compute $Succ(\mathcal{A}_i, B_i, s, h)$ for any particular s. We need only compute, for individual subnetwork C_i^j ($i \in \{1, 2\}$) over structure G^j,

$\max_s |Succ(\mathcal{A}_1, C_1^j, s, h) - Succ(\mathcal{A}_2, C_2^j, s, h)|$. If any of these values is greater than a given θ, then $\max_s |Succ(\mathcal{A}_1, B_1, s, h) - Succ(\mathcal{A}_2, B_2, s, h)| > \theta$, and $\langle \mathcal{A}, B_1, B_2, h \rangle \notin SBPlan$.

7 Related Work

Sensitivity Analysis

The most common notion of robustness is an insensitivity to variation of one or more parameters in a network. One technique for gaining insight into this notion of robustness is *sensitivity analysis*. The basic idea is to systematically vary one or more of the conditional probability values of the Bayes net and to study the resultant effects on the output [12]. One recent development, SAMIAM [2], is an implemented tool for actually computing the sensitivity of one parameter to another. Varying each probability in the network individually while studying each of the individual effects on the output is called one-way sensitivity analysis. In an n-way sensitivity analysis, n of the probability assessments are varied simultaneously, which demonstrates each of the individual effects of varying each of the n probabilities and also reveals joint or synergistic effects.

Unfortunately, the brute-force approach to one-way sensitivity analysis of a Bayes net is computationally time consuming, since for each probability value under study, a number of propagations must be investigated where each propogation requires computing the output from the network. Thus, using this approach, the computational burden for even a one-way sensitivity analysis can be prohibitive even for a small Bayes net [3].

Laskey was the first to address the computational complexity of sensitivity analysis of Bayes nets. She introduced a method for using the partial derivative to yield a first-order approximation of the effect of varying a single probability parameter. While her method requires considerably less computation than the brute-force approach, it provides insight only in the effect of a small variation of a probability; a larger variation will rapidly break the technique down [12].

Because of the computational difficulty inherent in undertaking a complete one-way sensitivity analysis, attention has been directed to using the graphical structure of the network to determine independences of each probability to the output and to then to use this information to eliminate the varying of probabilities that will not change the output [3]. Some recent progress has also been made in developing a methodology for n-way sensitivity analysis which requires fewer outward propagations of the network to determine upper and lower bounds on the probabilities [11].

However, Henrion et al. have found that diagnostic performance with Bayesian belief networks is often surprisingly insensitive to imprecision in the numerical probabilities [8]. Therefore it makes sense to consider how the probability data is obtained in a given application in choosing a technique for evaluating the robustness of the output. In the case when two or more experts are consulted to determine probabilities, their estimates of each of the probabilities in the Bayes net may differ, giving two different

Bayes nets on the same graph. This leads us to the problem considered in this paper: determining when two Bayes nets on the same graph are equivalent for the purposes of planning.

The notion of robustness in the sensitivity analysis literature that is closest to ours is that of Pradhan, et al., who consider the experimental effects of varying all of the parameters at once [16]. This is called *uncertainty analysis* by Kjælrulff and van der Gaag [11] and is also considered by Henrion, et al. [8]. Our work differs from theirs in three key aspects: we consider planning, rather than inference, and we introduce planning into the problem explicitly; we consider the algorithmic complexity of determining robustness, and we compare two explicit sets of probabilities, rather than varying the parameters of one.

Integration of Probabilistic Information

The problem of integrating probabilistic information that served as a starting point for this work has been considered by researchers in a variety of different fields of computer science and statistics. As mentioned in Section 1, the largest body of work on the fusion of uncertain data comes from the area of multisensor fusion (see [1, 7] for early summaries). The basic problem addressed in this field is similar to ours: a number of sensors provides the user/reasoning agent with a set of data streams carrying uncertain data which need to be combined together. For example, Rehg, Murphy, and Fieguth [17] study the problem of detecting whether a human, watched by four independent sensors, is speaking. Each sensor watches for particular features on the human face (face detection, skin color detection, skin texture, mouth motion) and the output of each sensor is a Bayes net. The reasoning agent then combines (fuses) the nets into one that allows it to detect whether a human is speaking. This work has been extended recently by Pavlovic et al. [15].

Different methods of fusion of probabilistic information have been proposed. Most of them fall into one of the two categories: (i) integration techniques for specific problems and (ii) so called “toolbox” approaches. The multisensor fusion work mentioned above belongs to the first category: the solutions proposed in [17, 15] are specific to the particular problem being investigated and to the particular sensors being used. Other examples of this approach include the work of Druzdzel and Diez [6] and Thurston and Ibrahim [18]. In [6] the problem of combining the knowledge from different sources when building Bayes nets is considered. This problem mirrors ours most closely. Druzdzel and Diez gave a detailed analysis of the integration problem for a particular Bayesian model. On the other hand, our work here looks at the problem of determining under which conditions there is no need to apply complex data integration methods.

An example of a “toolbox” approach is the work of Dekhtyar, Ross, and Subrahmanian on combining probabilities in Temporal Probabilistic databases [5]. They introduce a special *compaction* operation whose purpose is to combine together the probabilities from different probability distributions. This operation is parameterized by the combination function that dictates how the probability integration should proceed. Users are allowed to define their own combination functions to be used in the system. While more flexible, this approach factors the problem of developing the actual integration methods out of the Temporal Probabilistic database framework.

8 Discussion

The question of whether two sets of probability distributions for a Bayes net structure are equivalent (in any sense) is a natural one, and one that anyone building Bayes nets must address at some point. We have considered equivalence of Bayes nets with respect to planning. Although this question has been addressed in terms of sensitivity analysis, we are not familiar with any work that compares entire sets of conditional probability tables at once in the context of planning. This might be explained by our results, namely, that it is computationally infeasible (in the general case) to do so. However, Section 6 offers the first glimmerings of hope that this problem might in fact be tractable for some cases that occur in real life.

Because a growing number of heuristics for NP- and coNP-complete problems exists, as well as some heuristics for $\mathrm{NP}^{\mathrm{PP}}$-complete problems, there is an additional hope that the equivalence problems described here might be approachable heuristically. Future work could include an extension of the results in Section 6, heuristics for these equivalence problems, and further consideration of the notion of equivalence with respect to planning for Bayes nets.

9 Acknowledgements

The authors would like to thank the anonymous reviewers for suggesting ways to improve the paper. We are also grateful to Jesse Hoey for providing us with useful information about SPUDD.

References

[1] M.A. Abidi and R.C. Gonzalez, editors. *Data Fusion in Robotics and Machine Intelligence*. Academic Press, Inc., San Diego, CA, 1992.

[2] Hei Chan and Adnan Darwiche. When do numbers really matter? In *Proc. Uncertainty in AI*, 2001.

[3] Veerle M.H. Coupé and Linda C. van der Gaag. Practicable sensitivity analysis of Bayesian belief networks. In *M. Huskova, P. Lachout, J.A. Visek. Prague Stochastics '98 – Proceedings of the Joint Session of the 6th Prague Symposium of Asymptotic Statistics and the 13th Prague Conference on Information Theory, Statistical Decision Functions and Random Processes, Union of Czech Mathematicians and Physicists*, pages 81 – 86, 1998.

[4] Veerle M.H. Coupé, Linda C. van der Gaag, and J. Dik F. Habbema. Sensitivity analysis: an aid for probability elicitation. *Knowledge Engineering Review*, 15(3):1–18, 2000.

[5] Alex Dekhtyar, Robert Ross, and V.S. Subrahmanian. Probabilistic temporal databases, I: A lgebra. *ACM Transactions on Database Systems*, 26(1), 2001.

[6] Marek J. Druzdzel and F. Javier Diez. Criteria for combining knowledge from different sources in probabilistic models. In *Working Notes of the workshop on 'Fusion of Domain Knowledge with Data for Decision Support,' Sixteenth Annual Conference on Uncertainty in AI*, pages 23–29, 2000.

[7] D. Hall. *Mathematical Techniques in Multisensor Data Fusion*. Artech House, Boston, MA, 1992.

[8] Max Henrion, Malcolm Pradhan, Brendan Del Favero, Kurt Huang, Gregory Provan, and Paul O'Rorke. Why is diagnosis using belief networks insensitive to imprecision in probabilities? In *Proc. 12th Conference on Uncertainty in AI*, pages 307–314, 1996.

[9] Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutiler. SPUDD: Stochastic planning using decision diagrams. In *Proc. Uncertainty in AI*, 1999.

[10] Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutiler. Optimal and approximate stochastic planning using decision diagrams. Technical Report TR-00-05, University of British Columbia, 2000.

[11] Uffe Kjærulff and Linda C. van der Gaag. Making sensitivity analysis computationally efficient. In *Proc. Uncertainty in AI*, 2000.

[12] Kathryn Blackmond Laskey. Sensitivity analysis for probability assessments in bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(6):136–142, 1995.

[13] Michael L. Littman. Probabilistic propositional planning: Representations and complexity. In *Proceedings of 14th National Conference on Artificial Intelligence*. AAAI Press / MIT Press, 1997.

[14] Michael L. Littman, Judy Goldsmith, and Martin Mundhenk. The computational complexity of probabilistic plan existence and evaluation. *The Journal of AI Research*, 9:1–36, 1998.

[15] V. Pavlovic, A. Garg, J. Rehg, and T. Huang. Multimodal speaker detection using error feedback dynamic Bayesian networks. In *Computer Vision and Pattern Recognition*, 2000.

[16] Malcolm Pradhan, Max Henrion, Gregory Provan, Brendon del Favero, and Kurt Huang. The sensitivity of belief networks to imprecise probabilities: An experimental investigation. *Artificial Intelligence*, 85(1–2):363–397, 1996.

[17] James M. Rehg, Kevin P. Murphy, and Paul W. Fieguth. Vision-based speaker detection using Bayesian networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, Ft. Collins, CO, 1999.

[18] S.W. Thurston and J. Ibrahim. Informative prior specification for linear regression models using parameter decompositions. (in preparation), 2001.

[19] Jacobo Torán. Complexity classes defined by counting quantifiers. *Journal of the ACM*, 38(3):753–774, 1991.

A Appendix: Sample SPUDD Code

Provided for the convenience of the reader, here is the description of the Horseback Riding Resort problem (with Set 1 CPTs) from our example (see Section 3) in SPUDD format.

```
variables ( rain trails expenses riders profit)

action LowRates

rain (0.2)

trails (rain (0.3) (0.6))

expenses(riders (0.5) (0.35))

riders ( trails ( rain (0.6) (0.9))
                ( rain (0.3) (0.5))
        )

profit (expenses (riders (0.45) (0.35))
                 (riders (0.65) (0.45))
       )


endaction


action HighRates

rain (0.2)

trails (rain (0.3) (0.6))

expenses(riders (0.5) (0.35))

riders ( trails ( rain (0.4) (0.7))
                ( rain (0.2) (0.25))
       )

profit (expenses (riders (0.5) (0.4))
                 (riders (0.75) (0.55))
        )

endaction



action GroomLowRates

rain (0.2)
```

```
trails (rain (0.8) (1.0))

expenses(riders (0.6) (0.45))

riders ( trails ( rain (0.6) (0.9))
                ( rain (0.3) (0.5))
        )

profit (expenses (riders (0.45) (0.35))
                 (riders (0.65) (0.45))
       )


endaction


action GroomHighRates

rain (0.2)

trails (rain (0.8) (1.0))

expenses(riders (0.6) (0.45))

riders ( trails ( rain (0.4) (0.7))
                ( rain (0.2) (0.25))
       )

profit (expenses (riders (0.5) (0.4))
                 (riders (0.75) (0.55))
        )

endaction


action AdLowRates

rain (0.2)

trails (rain (0.3) (0.6))

expenses(riders (0.65) (0.5))

riders ( trails ( rain (0.8) (0.95))
                ( rain (0.65) (0.9))
        )

profit (expenses (riders (0.55) (0.35))
                 (riders (0.65) (0.45))
       )



endaction
```

```
action AdHighRates

rain (0.2)

trails (rain (0.3) (0.6))

expenses(riders (0.65) (0.5))

riders ( trails ( rain (0.5) (0.8))
                ( rain (0.3) (0.35))
       )

profit (expenses (riders (0.6) (0.4))
                 (riders (0.75) (0.55))
        )

endaction


action AdGroomLowRates

rain (0.2)

trails (rain (0.8) (1.0))

expenses(riders (0.75) (0.65))

riders ( trails ( rain (0.8) (0.95))
                ( rain (0.65) (0.9))
        )

profit (expenses (riders (0.55) (0.35))
                 (riders (0.65) (0.45))
       )


endaction


action AdGroomHighRates

rain (0.2)

trails (rain (0.8) (1.0))

expenses(riders (0.75) (0.65))

riders ( trails ( rain (0.5) (0.8))
                ( rain (0.3) (0.35))
       )
```

```
profit (expenses (riders (0.6) (0.4))
                 (riders (0.75) (0.55))
        )

endaction


reward (profit (1) (0))


discount 0.9
tolerance 0.01
```