On Spline Approximations for Bayesian Computations

Eugene Santos Jr.

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-92-47
October 1992

On Spline Approximations for Bayesian Computations[1]

Eugene Santos Jr.
Department of Computer Science
Box 1910, Brown University
Providence, RI 02912
esj@cs.brown.edu

August 6, 1992

KEYWORDS: probabilistic reasoning, constraint satisfaction, linear programming, least-squares fit

AMS CLASSIFICATIONS: 68T99 - Artificial Intelligence

[1]This work has been supported by the National Science Foundation under grant IRI-8911122 and by the Office of Naval Research, under contract N00014-88-K-0589.

Abstract

Probabilistic reasoning suffers from NP-hard implementations. In particular, the amount of probabilistic information necessary to the computations is often overwhelming. For example, the size of conditional probability tables in Bayesian networks has long been a limiting factor in the general use of these networks.

We present a new approach for manipulating the probabilistic information given. This approach avoids being overwhelmed by essentially compressing the information. Furthermore, unlike existing techniques, we do not need to explicitly access all the data in order to perform our computations. We have achieved this by transforming the problem into linear constraint satisfaction and using approximating functions.

1 Introduction

Probabilistic reasoning has become the mainstay of almost all inferencing systems. From expert systems for geological surveying [8] to medical diagnosis systems [32], the probabilistic approach provides a rich framework for knowledge representation that is essential in these domains. Unfortunately, because of one thing or another, the various approaches generally suffer from NP-hard implementations. The systems built tend to be tailored or refitted around these problems. This often results in various behavioral anomalies and general ad-hocness. Yet, we must keep in mind as always that these problems will persist only until efficient algorithms are found.

Probably one of the most popular models for probabilistic reasoning is Pearl's Bayesian networks [19]. This approach provides a handy visualization of our knowledge through the means of a directed acyclic graph of random variable relationships. Basically, each node in the graph represents a "discrete" random variable. The directed arcs between the nodes represent probabilistic conditional (in)dependencies. Furthermore, joint probabilities on the random variables can be readily computed via Bayes' Theorem and the given conditional independence assumptions. These computations take the form of multiplying conditional probability tables which have been associated with each node in the network.

Bayesian networks have been applied to various domains such as story comprehension [9, 10, 1, 3], circuit fault detection [7, 19], medical diagnosis [32] and planning systems [15]. However, computing with these networks has proven to be NP-hard as we might have expected [5]. This has generally prevented problem formulations from utilizing the full representational capabilities of Bayesian networks.

There are two key factors which prevent the general use of Bayesian networks. The first is *network topology*. Most of the existing algorithms are quite sensitive to this and often end up in a "do or die" situation (See [19, 31]). For example, Pearl's message-passing scheme [19] is restricted to singly-connected networks, that is, there can only exist at most one directed path between any two nodes in the graph. Unfortunately, for those which aren't hindered by topological considerations, they invariably run into the second factor, *conditional probability table size*.

Very little work, if any, has been done to address the issue of table size. Most have simply taken for granted that this problem is unavoidable. Up until recently, table size has not been a major factor simply because use of Bayesian networks were generally restricted to "toy"/prototype domains.

Table size is directly correlated with the amount of discretization or precision used in the problem formulation. Basically, the size of the table is exponential with respect to the number of instantiations required by each random variable. Thus, in the "toy" domains, precision is generally sacrificed resulting in relatively small tables. Extremely coarse approximations of information is typically sufficient for this relatively low level of prototyping.

Although there have been arguments that human beings reason well with only coarse approximations [19], domains such as robotic navigation require rather precise problem formulations to handle quantitative data such as sonar readings and map locations [15]. The higher degree of sensitivity needed in this data means a finer grained discretization of random variable instantiations. In a related vein, consider the domain of medical diagnosis where we are modeling the correlations between diseases and symptoms. As Charniak and McDermott point out in [2], when you start considering relationships between diseases and simultaneous multiple symptoms, that is,

$$P(\text{disease A}|\text{symptom } S_1, \ldots, \text{ symptom } S_n),$$

the explosion becomes quite apparent since we have to keep a probability around for each combination of disease and symptoms.

The explosive growth of probability tables for Bayesian networks is obviously a special case of a more general problem encountered in probabilistic systems. There is the need of having large amounts of data on hand such as correlations, statistical frequencies, conditional probabilities etc., in order to satisfy the axioms of probability theory so that computations can be performed. Assumptions on the problem are often made in an effort to reduce this data-explosion. For example, we might assume that 2 symptoms are sufficient for determining a disease or we may choose to quantize distance information to something like near, medium, far. However, as we can easily guess, we run into the classical dilemma of trading off representational power for computability. More often than not though, at least in terms of our existing computing capabilities, we end up with overly simple models which themselves are just barely computable. Much of the problems as we shall see involving table size spawn from the necessity of referencing each and every probability. This occurs in all the existing algorithms.

In this paper, we will focus on how to deal with explosive amounts of data. In particular, we present an approach which may be used to solve the conditional table size problem in Bayesian networks.

Our approach begins with the following recent result: Santos introduced a new technique for probabilistic computations [23, 25, 24, 3, 26]. Bayesian-style reasoning could be modeled as a linear constraint satisfaction problem. Once this was achieved, extremely efficient tools and techniques can be applied from Operations Research to perform our computations. Experimental results demonstrated that this approach was superior to existing techniques. Whereas the old approaches had an expected-case exponential run-time, this method exhibited an expected-case polynomial ($O(x^2)$) run-time over the same problem-sets. Furthermore, the linear constraints approach was basically insensitive to network topology.

Since linear constraint satisfaction already takes care of the network topology problem for Bayesian networks, this will provide us with a promising launching point for tackling the table size problem. We hope to show how the problem can be approached through linear constraint satisfaction. Basically, we will develop a model that will allow us to avoid explicitly storing the tables by replacing them with approximations.

We begin in Section 2 by briefly describing Bayesian networks and their uses. This section should provide a clear idea of the table size problem. In Section 3, we will discuss the various existing approaches for performing computations on Bayesian networks. In Sections 4 and 5, we will provide some of our definitions and notations and then briefly summarize our earlier constraints approach. Finally, Sections 6 and 7 will present our new approach which utilizes approximation functions.

2 Bayesian Networks

In probabilistic reasoning, random variables (abbreviated, r.v.) are used to represent events and/or objects in the world. By making various instantiations to these r.v.s, we can model the current state of the world. For example, consider the Mobile Target Localization problem (abbreviated MTL) [15]. It involves a robot equipped with sonars to track some given mobile target and reporting its location in the coordinate system of a global map. The robot, itself, must also move around in order to keep the target within sensor range.

We begin our formulation with a set of *locations* L corresponding to the regions that are used to encode the location of both the robot and target. Thus, L represents the discretization of our global map. We now use the following r.v.s in order to represent our world and its uncertainties:

- S_{R_i} represents the location of the robot at time i ranging over L.
- S_{T_i} represents the location of the target at time i also ranging over L.
- O_{R_i} represents our sensor observations of the robot's surroundings at time i.
- O_{T_i} represents our sensor observations of the target's location relative to the robot at time i.
- A_{R_i} represents the action our robot takes such as movement at time i.

So, the robotic sequence works as follows: The robot initially has some idea as to where it is and where the target was. It then takes set of sonar readings on its immediate surroundings and another set for locating the target. Based on this information, it decides the best sequence of actions and executes the first one. Once we finish executing the first action, we start over again and choose a new sequence, In essence, the sequence is used to help predict what the world should look like after a given action.

As we can clearly see, by making various instantiations to the above r.v.s, we are creating different states of the world. In this problem, our goal is to find the best action for the robot to perform given the other information such as sonar readings and object locations. Thus, this will involve computing joint probabilities of the given r.v.s. Unfortunately, the task is nearly impossible without additional information concerning relationships between the r.v.s. In the worst case, we would need the probabilities of every instantiation combination which is combinatorially explosive to say the least.

On the other hand, we look at Bayes' Theorem as follows:

$$\begin{aligned} &P(S_{T_i}, S_{R_i}, O_{R_i}, O_{T_i}, A_{R_i}) = \\ &\quad P(S_{T_i}|S_{R_i}, O_{R_i}, O_{T_i}, A_{R_i}) P(S_{R_i}|O_{R_i}, O_{T_i}, A_{R_i}) \\ &\quad P(O_{R_i}|O_{T_i}, A_{R_i}) P(O_{T_i}|A_{R_i}) P(A_{R_i}). \end{aligned} \tag{1}$$

Bayesian networks [19] take this a step further by making the important observation that certain r.v. pairs may become uncorrelated once information concerning some other r.v.(s) is known. More precisely, we may have the following independence condition:

$$P(A|C_1, \ldots, C_n, U) = P(A|C_1, \ldots, C_n) \tag{2}$$

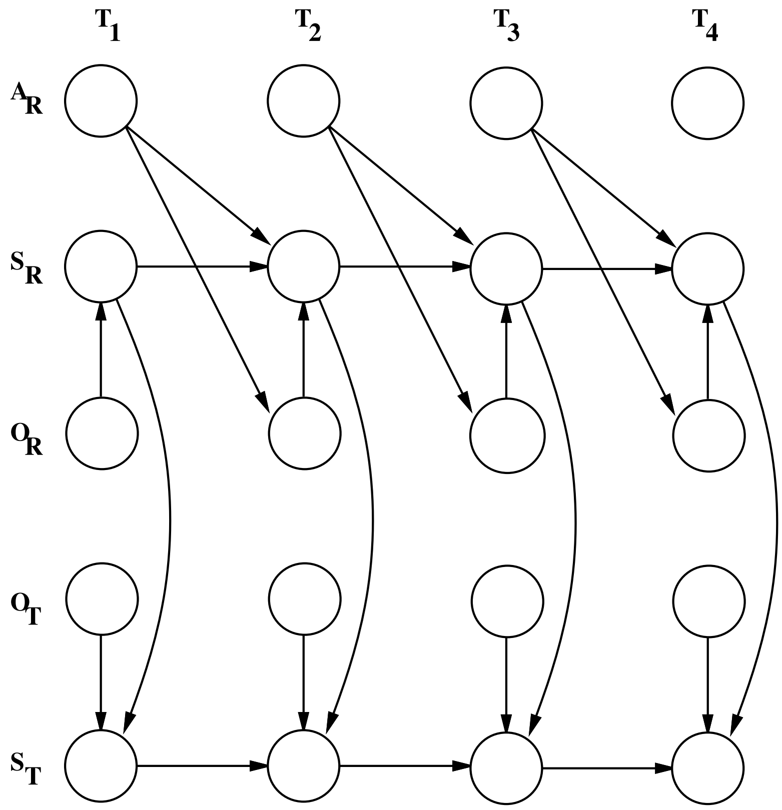

FIG. 2.1. Bayesian network for MTL problem with 4 time slices.

for any collection of r.v.s U. Intuitively, we can interpret this as saying that A is completely determined by $C_1, \ldots, C_n$.

Combined with Bayes' Theorem, these conditional independencies allow us to replace the terms in (1) with the smaller conditionals (2). Thus, instead of explicitly keeping the joint probabilities, all we need are smaller conditional probability tables which we can then use to compute the joint probabilities. Furthermore, these independence conditions provide us with a handy graphical visualization of the r.v. relationships at hand.

Using Bayesian networks, the MTL problem is modeled in Figure 2.1. We can now rewrite equation (1) as follows for the r.v.s involved in time step

T_1:[2]

$$P(S_{T_1}, S_{R_1}, O_{R_1}, O_{T_1}, A_{R_1}) = \\ P(S_{T_1}|S_{R_1}, O_{T_1})P(S_{R_1}|O_{R_1})P(O_{R_1})P(O_{T_1})P(A_{R_i}). \quad (3)$$

What we have is a directed acyclic graph of r.v. relationships. Directed arcs between r.v.s represent conditional dependencies. When all the parents of a given r.v. are instantiated, that r.v. is said to be conditionally independent of the remaining r.v.s given it's parents (See (2)).

Our goal in the MTL problem is to determine the best course of action for the robot such that it continues to keep track of its target. Basically, the robot must base it's decision on current sonar readings plus its own internal knowledge of the world. Having formulated the MTL problem using Bayesian networks, determining the best action can be modeled as follows: Assuming we are only working with 4 time slices as seen in Figure 2.1, this will imply that we are looking for a 3 action sequence. Hence, we are given the following information:

- O_{R_1} which represents the current/actual sonar readings on the robot's position.
- O_{T_1} which represents the current/actual sonar readings on the target's position.

Our goal is to determine an action a^* for A_{R_1} such that

$$P(A_{R_1} = a^*|O_{R_1}, O_{T_1}) = \max_a P(A_{R_1} = a|O_{R_1}, O_{T_1}). \quad (4)$$

[2] We can isolate the r.v.s in time step T_1 from the remaining r.v.s because of the following observation on the conditional independence relationships: Let U be the remaining r.v.s in the network. We know that

$$P(S_{T_1}, S_{R_1}, O_{R_1}, O_{T_1}, A_{R_1}) = \\ \sum_U P(S_{T_1}, S_{R_1}, O_{R_1}, O_{T_1}, A_{R_1}, U).$$

Furthermore, according to our independence assumptions

$$P(S_{T_1}, S_{R_1}, O_{R_1}, O_{T_1}, A_{R_1}, U) = \\ P(U|S_{T_1}, S_{R_1}, O_{R_1}, O_{T_1}, A_{R_1})P(S_{T_1}, S_{R_1}, O_{R_1}, O_{T_1}, A_{R_1})$$

Since the second term on the right hand side is constant, summing over the different possible instantiations for U reduces the first term to the value of 1. Hence, we can simply ignore the r.v.s in U.

This type of computation performed with Bayesian networks is classified as *belief updating* [19]. In general, belief updating involves updating our beliefs about the different possible instantiations of a r.v. given certain instantiations of other r.v.s as evidence/observations.[3]

Belief updating is one of two common computations performed with Bayesian networks. The second is called *belief revision* [19]. Belief revision is best used for modeling explanatory/diagnostic tasks. Basically, some evidence or observation is given to us, and our task is to come up with a set of hypothesis that together constitute the most satisfactory explanation/interpretation of the evidence at hand. This process has also been considered *abductive reasoning* in one form or another [12, 30, 20, 4]. More formally, if W is the set of all r.v.s in our given Bayesian network and e is our given evidence[4], any complete instantiations to all the r.v.s in W which is consistent with e will be called an *explanation* or *interpretation* of e. Our problem is to find an explanation w^* such that

$$P(w^*|e) = \max_w P(w|e). \tag{5}$$

Intuitively, we can think of the non-evidence r.v.s in W as possible hypothesis for e. For example, consider the probabilistic medical diagnosis model presented in [20]. It is a 2-layer Bayesian network consisting of a layer of manifestations (symptoms) and a layer of disorders (diseases). The manifestations are conditionally dependent on different sets of disorders. Given a set of manifestations, we must determine a set of disorders which maximizes (5).

The Bayesian networks approach can achieve an enormous reduction in the amount of probability data needed. Unfortunately, even identifying all the conditional independencies is usually not enough. Looking at the MTL problem, we consider the the following conditional table:

$$P(S_{T_2}|S_{R_2}, O_{T_2}, S_{T_1}). \tag{6}$$

The size of this table is $|L|^3|O|$ where O is the range of instantiations for the sonar readings O_{T_2}. Clearly, the higher the precision we desire in our robot world, the larger both L and O become. Even if we had decided that there

[3]Typically, we are only interested in finding the best instantiation.

[4]That is, e represents a set of instantiations made on a subset of W.

are only 100 possible locations in our world, the size of this table alone would be in the millions. This remains the predominant prohibiting factor of trying to use Bayesian networks for modeling complex tasks.

3 Computational Methods

Barring the fact that we may have a combinatorial explosion in the number of conditional probability entries, belief revision and updating on Bayesian networks, however, is also NP-hard with respect to the size of the network [5, 31]. Various techniques are available for performing the computations, but we must point out that although they may have reasonable expected run-times, these complexity measures are made with respect to the number of nodes/edges in the network and consider the table sizes to be some multiplicative constant. Obviously, when we have a table with an exponential number of entries, the run-times will generally be exponential for these techniques.

Probably the two most popular approaches for performing belief updating are *message-passing schemes* [19] and *stochastic simulations* [29, 6, 33, 11]. In message-passing, the basic idea is to treat each node in the network as an individual processor. Information is then propagated back and forth between the nodes until equilibrium is achieved. The manner of communication defined by message-passing guarantees that the results obtained when the network stabilizes will correspond precisely to belief updating. This algorithm is known to be efficient on a restricted network topology called *singly-connected network*. A singly-connected network requires that there exists at most one directed path between any two nodes. However, message-passing fails on the more general class of multiply-connected networks. Although there are methods such as *clustering* and *conditioning* [19] for transforming a multiply-connected network into an equivalent singly-connected one [28, 13], the transformations have also been shown to be quite difficult to accomplish.

Stochastic simulation, on the other hand, has no such topological restriction. The method basically works as follows: Assume that we are given evidence e, A is the r.v. we wish to update and X is the set of remaining r.v.s not including those in e or A. We know that

$$P(A|e) = \frac{P(Ae)}{P(e)}. \tag{7}$$

Furthermore,

$$P(Ae) = \sum_X P(AeX), \text{ and} \tag{8}$$

$$P(e) = \sum_Y P(eY) \tag{9}$$

where $Y = X \cup \{A\}$. By making various instantiations for X, we can easily compute the joint distribution $P(AeX)$ and apply it to (8). This process is called taking *sample points* in order to construct the joint probabilities. Stochastic simulation uses various approaches for sampling points to achieve better and better estimates on the probabilities while also guaranteeing convergence. Unfortunately, while stochastic simulation is invariant under network topology, it is quite sensitive to the distribution in the conditional tables. For example, if we take the simplest approach to sampling by doing it randomly, then in a network with a slightly skewed distribution, say one of the conditional tables is sparse, then we may sample a large number of points which have a joint probability of 0. Thus, the estimate will be very poor, indeed.

Obviously, belief revision is a very different computational task from updating. In essence, updating is a summation over joint probabilities, whereas revision is a maximization over those same probabilities. It is this maximization property that prompts us to interpret revision as some form of explanatory or diagnostic reasoning. Basically, the r.v.s being instantiated are likened to hypothesis. The joint probability is then used to measure how well a set of hypotheses supports the given evidence. Since, we are now explicitly defining the notion of a "best set"[5], it becomes necessary that we not only be able to compute the best, but also any other "good" alternatives. This is especially true in domains such as medical diagnosis where we must consider alternative diagnoses to the most-likely one.

Finding the alternatives also has implications for belief updating. From equation (8), it has been observed that $P(AeX)$ and $P(eY)$ for any A, X and Y are simply explanations/diagnoses for evidence e [26]. We can consider the alternative solutions to be sample points for use in (8). By starting with the "best" explanation, we now have the sample point with the largest probability, hence, avoiding the problem encountered with random sampling we saw above.

[5]Pearl in [19] calls this the *most-probable-explanation* (abbreviated, MPE) criterion.

One of the earliest methods for performing belief revision on Bayesian networks was a message-passing variant of the one mentioned above for belief updating [19]. Again, unfortunately, the scheme was restricted to singly-connected networks. Also, the scheme had no mechanisms for computing alternative solutions beyond the second most-probable. Sy in [34] introduces another variant of message-passing which has been empirically shown to be more efficient than Pearl's method. This variant changes the message passing into a strictly feed-forward manner, hence, requiring only one pass through the network. Furthermore, generating the alternative solutions is possible due to his architecture.[6] However, Sy's approach also stumbles on multiply-connected networks.

Shimony and Charniak in [31] showed that belief revision can be solved using a best-first search strategy. Hence, the problem of network topology is absorbed in the choice of a best-first search heuristic. Furthermore, through back-tracking, the alternative solutions could be generated. The biggest problem with this approach, though, is the explosive amount of search required as you progress to larger networks.

Most recently, Santos in [25, 26] introduced a new approach which reduces the problem of belief revision into linear constraint satisfaction. Having accomplished this reduction, efficient Operations Research tools such as the *Simplex Method* [17, 27, 18] could be used to solve the problem. This approach is invariant under network topology and also provides a mechanism for generating the alternative solutions. Although the newly transformed problem remains NP-hard, empirical studies indicate the approach to exhibit an expected-case polynomial run-time [24, 23, 26], easily outperforming the best-first search technique of [31, 4].

Although the above methods are radically different in their approaches, they all generally suffer from a common problem. Namely, if the conditional tables are exponentially large, the computation times are also exponential. This is due to the fact that each and every entry in the table will be explicitly referenced at some point in each of the algorithms. It becomes even worse for the methods restricted to singly-connected networks. Aside from the fact that reducing multiply-connected ones can be extremely expensive, one of

[6] In essence, a notion of backtracking is now present in this system. By carefully choosing an alternative branch from which to feed-forward, you could generate other solutions in order of probability.

the techniques, clustering, actually increases the table sizes combinatorially.

Our goal in this paper is to show how we can avoid having to reference every entry in the conditional table to perform our computations. Basically, our approach exploits the structure of the formulation by ignoring entire chunks of entries once some given entry has been explored. Careful compression of the tables should permit the realization of this approach.

4 Terms, Definitions and Notations

In this section, we provide terms and definitions which will be used throughout the remainder of this paper.

We first observe that a Bayesian network can be completely described by a finite collection of random variables and a finite set of conditional probabilities based on the r.v.s.[7]

NOTATION. Throughout the remainder of this thesis, upper case italicized letters such as $A, B, \ldots$ will represent r.v.s and lower case italicized letters such as $a, b, \ldots$ will represent the possible assignments to the associated upper case letter r.v., in this case, $A, B, \ldots$. Subscripted upper case letters which are not italicized are variables in a constraint system which explicitly represent the instantiation of the associated r.v. with the item in the subscript. For example, A_{a} denotes the instantiation of r.v. A with value a.

NOTATION. Given a r.v. A, the set of possible values for A called the *range* of A will be denoted by $R(A)$.

Given a Bayesian network, we can construct an ordered pair (V, P) where V is the set of r.v.s in the network and P is a set of conditional probabilities associated with the network. $P(A = a | C_1 = c_1, \ldots, C_n = c_n) \in P$ iff $C_1, \ldots, C_n$ are all the immediate parents of A and there is an edge from C_i to A for $i = 1, \ldots, n$ in the network. We can clearly see that (V, P) completely describes the Bayesian network.

DEFINITION 4.1. *Given a Bayesian network $\mathcal{B} = (V, P)$, an* instantiation *is an ordered pair (A, a) where $A \in V$ and $a \in R(A)$. (An instantiation (A, a) is also denoted by $A = a$ and A_{a}.) A collection of instantiations w is called an* instantiation-set *iff $(A, a), (A, a')$ in w inplies $a = a'$.*

[7]We consider prior probabilities to be degenerate cases of conditional probabilities, i.e., $P(A = a) = P(A = a|\phi)$ where ϕ is the empty set.

An instantiation represents the event when a r.v. takes on a value from its range. Given an instantiation-set, we can define the notion of the *span* of an instantiation-set.

DEFINITION 4.2. *Given an instantiation-set w for a Bayesian network $\mathcal{B} = (V, P)$, we define the* span *of w,* $\mathsf{span}(w)$*, to be the collection of r.v.s in the first coordinate of the instantiations. Furthermore, an instantiation-set w is said to be* complete *iff* $\mathsf{span}(w) = V$.

Straightforwardly, the span of an instantiation-set simply denotes the r.v.s which have been instantiated.

NOTATION. For each r.v. A, we define $\mathsf{cond}(A)$ as follows: $B \in \mathsf{cond}(A)$ iff there exists a conditional probability in P of the form $P(A = a|\ldots, B = b, \ldots)$.

Intuitively, $\mathsf{cond}(A)$ is the set of r.v.s which are the parents of A.

NOTATION. Given an instantiation-set w for $\mathcal{B}$ such that $\mathsf{cond}(A) \subseteq \mathsf{span}(w)$, $w(A)$ denotes the instantiation $A = a$ where $(A, a) \in w$.

$$w(\mathsf{cond}(A)) = \{w(B)|B \in \mathsf{cond}(A)\}.$$

DEFINITION 4.3. *Given an instantiation-set $w = \{(A_1, a_1), \ldots, (A_n, a_n)\}$ for a Bayesian network $\mathcal{B} = (V, P)$, we define the probability of w to be*

$$P(w) = P(A_1 = a_1, \ldots, A_n = a_n).$$

5 Constraints Formulation

As we mentioned earlier, our new approach is based on the results of transforming belief revision into linear constraint satisfaction [25, 26]. This approach was shown to be topologically invariant, capable of generating alternative solutions and perform efficiently on moderately-connected networks. In this section, we give a brief summary of the concepts and methodology used in the transformation that will be shared with our new approach.[8]

The transformation involves mapping the notion of r.v. instantiations into some multi-dimensional space which we will denote by $\Re^n$. A subspace of

[8] Precise details and theoretical proofs can be found in [25, 26].

$\Re^n$ will represent "valid" instantiations where valid includes things like being consistent to the given evidence e, each r.v. has at most one instantiation, etc. In particular, we are interested in transforming it into a *polyhedral convex set.*[9] Such a set can be described by a collection of linear inequalities. As it turns out, these inequalities will intuitively correspond to the restrictions/constraints required in making valid instantiations of the r.v.s. Finally, we would like to define a *linear energy function* such that by minimizing it over the convex set, the resulting answer will be the best explanation after we make the appropriate inverse mapping. Thus, we would have the makings of a linear constraint satisfaction problem.

We begin our transformation by mapping r.v. instantiations. Let A be a r.v. and a be a possible instantiations for A. If A is instantiated to a, that is, $A = a$, then we would like to set a real variable $\mathsf{A_a}$ to the value 1. If $A \neq a$, then $\mathsf{A_a} = 0$. This holds for every possible instantiation of A for every r.v. A.

Having mapped instantiations to $\Re^n$, we must now define our polyhedral convex set. We begin with the simplest of constraints: Each r.v. must have exactly one instantiation. This can be achieved with the linear constraint

$$\sum_{a \in R(A)} \mathsf{A_a} = 1 \tag{10}$$

where $R(A)$ is the set of all possible instantiations for A. Next, for each conditional probability $P(A = a | B = b, C = c)$ in the network,[10] construct a real variable $q[\mathsf{A_a} \mid \mathsf{B_b}, \mathsf{C_c}]$ (which for convenience, we will simply call it q for now) such that $q = 1$ if the conditional probability $P(A = a | B = b, C = c)$ is used in computing the joint probability. Otherwise, $q = 0$. Obviously, the conditional will be used only when we have the instantiations $A = a$, $B = b$ and $C = c$. To guarantee this property for q, we simply use the following constraints:

$$q \leq \mathsf{B_b}, \tag{11}$$

$$q \leq \mathsf{C_c} \tag{12}$$

[9] "Polyhedral" refers to the fact that the boundaries of the subspace are composed of hyperplanes.

[10] Generalizing this to other than two conditionals is straightforward.

and

$$\sum_{q \in Q(\mathsf{A}_\mathsf{a})} q = \mathsf{A}_\mathsf{a} \tag{13}$$

where $Q(\mathsf{A}_\mathsf{a})$ are all the q's whose associated conditional probability is of the form $P(A = a|\ldots)$.

Finally, given evidence e, if $A = a$ is an instantiation in e, then we also include the constraint $\mathsf{A}_\mathsf{a} = 1$.

Taking all these constraints together, we can prove that the resulting polyhedral convex set is indeed the the set of valid explanations for e. All that is left for us is to define the appropriate energy function. We will simply form a linear function from the q's, also called *conditional variables*, as follows: If q is true, this implies that the associated conditional probability must have been used to compute the joint probability. Thus, for each q, we have the following term in the energy function:

$$-\log(P(A = a|B = b, C = c))q[\mathsf{A}_\mathsf{a} \mid \mathsf{B}_\mathsf{b}, \mathsf{C}_\mathsf{c}]. \tag{14}$$

By taking $\exp^{-(\text{energy function})}$, we will get back the correct joint probability. Hence, minimizing it will get us our most-probable explanation.

We have now reduced belief revision into linear constraint satisfaction. In particular, what we now have is an *0-1 integer linear programming* problem [17, 27, 18]. We can efficiently solve this problem by combining the *Dual Simple Method* with *Branch and Bound* [24, 25, 26]. Furthermore, by using a *Cutting Plane* approach [25, 26], we can generate all the alternative solutions in order of decreasing probability. The individual methods themselves are well-understood in Operations Research allowing for many other variations to be used which may be even more effective.

Unfortunately, this approach also suffers from table size explosions. For each entry in a conditional table, we must associate a unique real variable q. As we shall see, we present a new approach founded on this linear constraint satisfaction method that should effectively avoid the table size problem.

6 Approximation Functions

To avoid the possible exponential explosion in our conditional table size, our approach is to compress the information. Through the use of approximation functions we call *splining functions*, we would reduce the entire table and store it as a simple function. Ideally, we would like to have a table reduced to a single splining function, but even multiple functions are still more desirable than having the explicit table. These splining functions are real-valued functions whose range will be from some $\Re^n$ and domain be $\Re$. Obviously, the domain will correspond to the conditional probabilities whereas the range will be used as indices in fetching/computing the appropriate conditional probabilities. Unfortunately, instantiations for r.v.s need not be numeric in nature. They can be objects, names, etc. Hence, it is necessary to map them into real values for our purposes.

NOTATION. $\mathcal{Q}$ is the set of rational numbers. $\mathcal{Q}^+$ is the set of positive rational numbers.

Let $\mathcal{B} = (V, P)$ be a Bayesian network where V is the set of r.v.s and P are the associated conditional probabilities in the network.

DEFINITION 6.1. *Given a r.v. A in V, a one-to-one and onto mapping E_A from $R(A)$ to $\mathcal{Q}^+$ is called an* encoder *for A.*

Intuitively, encoders provide a total ordering on the possible instantiations for a r.v. Furthermore, it provides a mechanism for flexibly transforming/reorganizing our instantiations to a possibly more useful form or interpretation as we shall see.

Clearly, the choice of encoders will have a major impact on the success of our compression/approximation. The problem is certainly easier if we just arbitrarily choose an encoder to work with. As we shall see in the next section though, it is possible to determine an optimal encoder.

One final note on encoders before we proceed with our formulation is on the possibilities of using multiple encoders. Currently, we will just assume a single encoder to simplify our discussion. At the end of this section, the benefits of multiple encoders should become clear. A method will also be presented for handling them.

Let $\mathcal{E}_\mathcal{B}$ be a collection of encoders for the r.v.s in V such that there is exactly one encoder for each r.v.

NOTATION. Let T be a collection of conditional probabilities from P.

$$\text{RV}(T) = \{A | P(C_0 = c_0 | C_1 = c_1, \ldots, C_n = c_n) \in T \text{ and } C_i \equiv A \text{ for some } i\}$$

If we have multiple splining functions, then we must have a way of determining which one to use for computing probabilities given an instantiation-set. For example, given an instantiation-set d and some table T such that $\mathsf{span}(d) \subseteq \text{RV}(T)$, only some of the splining functions will be applicable to d. The remaining are not defined for these particular instantiations. By restricting which entries in a table may be grouped with other entries, we can automatically provide such a mechanism.

DEFINITION 6.2. *Let T be a conditional table in $\mathcal{B}$ and assume*

$$\text{RV}(T) = \{C_0, C_1, \ldots, C_n\}.$$

Let $\rho(T)$ be a partition on T. We say the $\rho(T)$ is contiguous *with respect to $\mathcal{E}_{\mathcal{B}}$ if and only if for each cell σ in $\rho(T)$, there exists α_i, β_i for $i = 0, \ldots, n$ such that*

$$\sigma = \{P(C_0 = c_0 | C_1 = c_1, \ldots, C_n = c_n) \in T | \alpha_i \leq E_{C_i} \leq \beta_i \text{ for } i = 0, \ldots, n\}.$$

We call each σ in $\rho(T)$ a sub-table *of T.*

Intuitively, we can view a contiguous partition as follows: For each r.v. B involved in T, somehow uniquely number the instantiations $R(B)$. We now construct a hypercube whose axes correspond to each r.v. B in T bounded by the minimum and maximum values of the numbering on $R(B)$. A contiguous partition of T will cut up the hypercube into a set of smaller hypercubes based on the instantiation numbering. Each of these sub-tables will be replaced by a splining function.

DEFINITION 6.3. *Let $\rho(T)$ be a contiguous partition with respect to $\mathcal{E}_{\mathcal{B}}$. For each cell σ in $\rho(T)$, we associate a function $S_{\mathcal{E}_{\mathcal{B}},\sigma}$ from $\Re^{n+1}$ to $\Re$ such that*

$$S_{\mathcal{E}_{\mathcal{B}},\sigma}(E_{C_0}(c_0), E_{C_1}(c_1), \ldots, E_{C_n}(c_n)) = P(C_0 = c_0 | C_1 = c_1, \ldots, C_n = c_n) \tag{15}$$

where $E_{C_0}, E_{C_1}, \ldots, E_{C_n}$ are the encoders found in $\mathcal{E}_{\mathcal{B}}$ and

$$c_0 \in R(C_0), \ldots c_n \in R(C_n).$$

We call $S_{\mathcal{E}_{\mathcal{B}},\sigma}$ a spline *for σ. Let $\mathcal{S}_{\mathcal{E}_{\mathcal{B}},\rho(T)}$ be a collection of splines associated with partition $\rho(T)$. We call $\mathcal{S}_{\mathcal{E}_{\mathcal{B}},\rho(T)}$ a* spline-set *for T.*

Let $\mathcal{S}_{\mathcal{E}_{\mathcal{B}}}$ be a set of spline-sets such that each table T in $\mathcal{B}$ is associated with exactly one spline-set.

NOTATION. Given $\mathcal{S}_{\mathcal{E}_{\mathcal{B}},\rho(T)}$, since $\rho(T)$ is a partition, for any $c_0 \in R(C_0), \ldots, c_n \in R(C_n)$,

$$S_{\mathcal{E}_{\mathcal{B}},\rho(T)}(E_{C_0}(c_0), \ldots, E_{C_n}(c_n))$$

will unambiguously refer to the appropriate spline function defined in

$$\mathcal{S}_{\mathcal{E}_{\mathcal{B}},\rho(T)}.$$

Given $\mathcal{S}_{\mathcal{E}_{\mathcal{B}}}$, since all our conditional probability tables are unique, for any $c_0 \in R(C_0), \ldots c_n \in R(C_n)$,

$$S_{\mathcal{E}_{\mathcal{B}}}(E_{C_0}(c_0), \ldots, E_{C_n}(c_n))$$

will unambiguously refer to the appropriate spline function defined in $\mathcal{S}_{\mathcal{E}_{\mathcal{B}}}$.

We now redefine our notion of a Bayesian network.

DEFINITION 6.4. *Given a Bayesian network $\mathcal{B} = (V, P)$, let $\mathcal{E}_{\mathcal{B}}$ be a collection of encoders and $\mathcal{S}_{\mathcal{E}_{\mathcal{B}}}$ be a collection of spline-sets. We define a* splined Bayesian network *to be a 3-tuple $\overline{\mathcal{B}} = (V, \mathcal{E}_{\mathcal{B}}, \mathcal{S}_{\mathcal{E}_{\mathcal{B}}})$.*

DEFINITION 6.5. *Given a complete instantiation set w for $\mathcal{B}$, we define the* spline probability, *P_s for $\overline{\mathcal{B}}$ as*

$$P_s(w) = \prod_{A \in \mathsf{span}(w)} S_{\mathcal{E}_{\mathcal{B}}}(w(A), w(\mathsf{cond}(A))). \tag{16}$$

PROPOSITION 6.1. *Given a complete instantiation set w, $P(w) \equiv P_s(w)$.*

The above proposition shows that our splining probability is equivalent to our normal probability function. Thus, we can substitute splining probability functions into our computations for belief revision.

THEOREM 6.2. *Any Bayesian network can be modeled as a splined Bayesian network.*

(Proofs can be found in Appendix A.)

We now proceed to show how we can transform splined Bayesian networks into linear constraint satisfaction problems.

Clearly, a splined Bayesian network $\overline{\mathcal{B}} = (V, \mathcal{E}_{\mathcal{B}}, \mathcal{S}_{\mathcal{E}_{\mathcal{B}}})$ induces a partition on the original conditional probabilities P. Let $[P]_{\overline{\mathcal{B}}}$ denote this induced partitioning. Furthermore, a splining function from $\mathcal{S}_{\mathcal{E}_{\mathcal{B}}}$ is uniquely associated with each cell in the partition. If d is a cell in $[P]_{\overline{\mathcal{B}}}$, then S_d will uniquely denote the appropriate splining function.

We say that a r.v. instantiation $\{A = a\}$ *appears* in a cell in $[P]_{\overline{\mathcal{B}}}$ if there exists a conditional probability in the cell of the form $P(C_0 = c_0 | C_1 = c_1, \ldots, C_n = c_n)$ where for some $i = 1, \ldots, n$, $C_i \equiv A$ and $c_i \equiv a$.

Definition 6.6. *Given a r.v. A and a splined Bayesian network $\overline{\mathcal{B}}$, we define the partition on $R(A)$ induced by $\overline{\mathcal{B}}$ as follows: $a_1, a_2 \in R(A)$ both belong in the same partition if and only if for all cells, d in $[P]_{\overline{\mathcal{B}}}$, $\{A = a_1\}$ appears in d if and only if $\{A = a_2\}$ appears in d. We call this partitioning the* capsulation *of A by $\overline{\mathcal{B}}$ and denote it by $[A]_{\overline{\mathcal{B}}}$.*

Theorem 6.3. *Given any cell $d = \{a_1, \ldots, a_k\}$ in $[A]_{\overline{\mathcal{B}}}$ where $E_A(a_i) < E_A(a_{i+1})$, there does not exist $b \in R(A)$ such that $E_A(a_1) < E_A(b) < E_A(a_k)$ and $b \neq a_i$ for $i = 1, \ldots, k$.*

Definition 6.7. *Given a splining function $S \in \mathcal{S}_{\mathcal{E}_{\mathcal{B}}}$, we say that S is a* linear potential function (LPF) *if and only if*

$$S_{\mathcal{E}_{\mathcal{B}}}(E_A(a), E_{C_1}(c_1), \ldots, E_{C_n}(c_n)) \equiv e^{k_0 E_A(a) + k_1 E_{C_1}(c_1) + \ldots + k_n E_{C_n}(c_n) + k}$$

for some constants $k, k_0, k_1, \ldots, k_n$. We say that $\mathcal{S}_{\mathcal{E}_{\mathcal{B}}}$ is a linear potential space *if and only if all splining functions in it are all* LPF*s.*

Without going into detail, we must generalize our notion of constraint systems. Previously for belief revision, we restricted our variables to values of 0 and 1. We generalize this by allowing variables to be restricted to sets of real values. In doing so, we also generalize our notion of a 0-1 solution to the notion of a *permissible* solution as a solution which satisfies all the value restrictions as well as constraints.

We begin our construction as follows: Let $\overline{\mathcal{B}} = (V, \mathcal{E}_{\mathcal{B}}, \mathcal{S}_{\mathcal{E}_{\mathcal{B}}})$ be a splined Bayesian network and $\mathcal{S}_{\mathcal{E}_{\mathcal{B}}}$ be a linear potential space. For each r.v. A in V, construct a real variable x_A whose values are restricted to $\{E_A(a) | E_A \in \mathcal{E}_{\mathcal{B}}$ and $a \in R(A)\}$. Next, arbitrarily label all the cells in $[A]_{\overline{\mathcal{B}}}$ by

$$\{d_{A,1}, d_{A,2}, \ldots, d_{A,n}\}.$$

For each cell in $\{d_{A,1}, d_{A,2}, \ldots, d_{A,n}\}$, construct a new 0-1 restricted variable

$x_{d_{a,i}}$. Construct the following constraints:

$$\sum_{i=1}^{n} x_{d_{A,i}} = 1 \tag{17}$$

For each $d_{A,i}$,

$$x_A - \min_{a \in d_{A,i}} E_A(a) \geq -M(1 - x_{d_{A,i}}) \tag{18}$$

$$x_A - \max_{a \in d_{A,i}} E_A(a) \leq M(1 - x_{d_{A,i}}) \tag{19}$$

where M is some arbitrarily large positive constant. Intuitively, $x_{d_{A,i}}$ is used to "detect" whether x_A falls within a particular interval defined by the partitioning and x_A represents the instantiated value if any.

From our definitions above, we can unambiguously associate a splining function S in $\mathcal{S}_{\mathcal{E}_{\mathcal{B}}}$ with some table T. Furthermore, using our encoder mappings, we describe the domain of S as a cross product of intervals on real variables associated with the r.v.s in T. For example, let $\rho(T)$ be the partition on T associated with S. According to Definition 6.3, for each r.v. B in $\text{RV}(T)$, there exists a set of instantiations of B, $\{B = b_{i_1}, \ldots, B = b_{i_k}\}$ in the cell associated with S in $\rho(T)$ such that there does not exist an instantiation $\{B = c\}$ where $c \neq b_{i_j}$ for all i_j and $E_B(c)$ is in the interval from $\min_{i_j} E_B(b_{i_j})$ to $\max_{i_j} E_B(b_{i_j})$. Thus, we only need the min and the max values to describe the domain for S. Also, remember that we want to incorporate S into our probabilistic computations only when the r.v.s are instantiated within its restricted domain. Let us denote the interval for a r.v. A for splining function S by $[\min(A, S), \max(A, S)]$. We now proceed with our construction. For each splining function S involving the r.v.s $\{C_1, \ldots, C_n\}$, we construct the new variables $x_{C_i,S}$ which are virtual copies of x_{C_i} constructed above. For each $[\min(C_i, S), \max(C_i, S)]$, if there does not yet exist a 0-1 variable which detects whether x_{C_i} is in $[\min(C_i, S), \max(C_i, S)]$, create a new 0-1 variable $d_{C_i,S}$ with the following constraints: For each $d_{A,i}$,

$$x_{C_i} - \min(C_i, S) \geq -M(1 - d_{C_i,S}) \tag{20}$$

$$x_{C_i} - \max(C_i, S) \leq M(1 - d_{C_i,S}) \tag{21}$$

Now, continue and construct a new 0-1 real variable d_S adding the following constraints:

$$d_S \geq n - \sum_{j=1}^{n} d_{C_j,S} + 1 \tag{22}$$

$$d_S \leq \frac{1}{n}\sum_{j=1}^{n} d_{C_j,S} \tag{23}$$

For each i,

$$x_{C_i,S} \geq x_{C_i} - M(1-d_S) \tag{24}$$

$$x_{C_i,S} \leq x_{C_i} + M(1-d_S) \tag{25}$$

$$x_{C_i,S} \leq M d_S \tag{26}$$

d_S is used to indicate whether S will be used. If so, copy the values into the virtual copies and make the computations necessary based on the virtual copies.

We complete our transformation by defining an appropriate objective function. For each splining function S in $\mathcal{S}_{\mathcal{E}_\mathcal{B}}$, introduce the following terms into the objective function: Assume that

$$S_{l,\mathcal{E}_\mathcal{B}}(E_A(a), E_{C_1}(c_1), \ldots, E_{C_n}(c_n)) \equiv e^{k_0 E_A(a) + k_1 E_{C_1}(c_1) + \ldots + k_n E_{C_n}(c_n) + k}.$$

The terms to be added are

$$-\, k d_S - \sum_{i=1}^{n} k_i x_{C_i,S} \tag{27}$$

We now must show that the solution space defined by our induced constraint system is equivalent to the space of all complete instantiation-sets for the Bayesian network. We begin by providing a transformation from permissible assignments in our constraint systems to instantiation-sets. Let s be a permissible solution. We can construct a complete instantiation-set $w[s]$ as follows: For each r.v. A in V, A is instantiated if and only if $s(x_A) > 0$. Since our encoders are one-to-one and onto, then the inverse (or, called *decoder*) exists and we denote them by E_A^{-1}. If $s(x_A) > 0$, then $w[s](A) = E_A^{-1}(x_A)$.

Conversely, given a complete instantiation-set w, we can construct a permissible assignment $s[w]$ as follows: For each r.v. A in V, $s[w](x_A) = E_A(w(A))$. Furthermore, we properly activate the appropriate interval detectors. Finally, according to the instantiation-set w, we can easily determine which splining functions are active. $s[w](d_S) = 1$ if and only if S is an active splining function according to w. And, if S is active, then copy $s[w](x_{C_i,S}) = s[w](x_{C_i})$ for all i involved with S. Otherwise, $s[w](x_{C_i,S}) = 0$.

THEOREM 6.4. *w is a complete instantiation-set for $\mathcal{B}$ if and only $s[w]$ is a permissible solution for the induced constraint system.*

Having shown the equivalence, we can prove the following theorem on the probabilities being calculated.

THEOREM 6.5.

$$P(w) = e^{-\Theta(s[w])}.$$

Therefore, the optimal permissible solution for our induced constraint system will be the best complete instantiation set.

Finally, we must also incorporate the notion of evidence. Evidence, we recall, is the requirement that a r.v. be instantiated with a certain value. For the case where a r.v. A must be instantiated to a, we simply include the constraint $x_A = E_A(a)$. Thus, we can proceed with our belief revision computations like we did earlier in this section.

Since we have generalized our restrictions on what values a real variable may attain from simple 0 and 1, we must modify our original branch and bound algorithm found in [24, 26] to guarantee that we generate permissible solutions.

NOTATION. Let x be a real variable and $\{k_1, k_2, \ldots, k_n\}$ be its permissible values such that $k_i < k_{i+1}$. We define the following functions:

$$\lfloor k \rfloor_x = \max_{k_i \leq k} k_i$$

$$\lceil k \rceil_x = \min_{k_i \geq k} k_i$$

Similar to our original branch and bound algorithm, the basic idea is as follows: To find an optimal permissible solution, we solve a sequence of linear programs. This sequence can be represented by a tree where each node in the tree is identified with a linear program that is derived from the linear programs on the path leading to the root of the tree. The root of the tree is identified with the linear program induced by our constraint system. The linear programs along the nodes of the tree are generated using the following schema: Consider s_0, the optimal solution to our initial linear program denoted lp_0. If s_0 is a permissible solution, then we are finished. Otherwise, we choose some non-permissible variable assignment x in s_0 and define two new problems lp_1 and lp_2 as descendants of lp_0. lp_1 is identical to lp_0 except for the additional constraint $x \geq \lceil s^0(x) \rceil_x$, and lp_2 is identical to

lp_0 except for the additional constraint $x \leq \lfloor s^0(x) \rfloor_x$. Note that the two new problems do not have s_0 as their optimal solutions. Since we are looking for a permissible assignment, the optimal permissible solution must satisfy one of the additional constraints.

As we can clearly see, we now proceed in a similar fashion to our branch and bound method for 0-1 problems.

ALGORITHM 6.1. *Given a constraint system $L = (\Gamma, I, \psi)$, find its optimal permissible solution.*

1. *(*Initialization*) Set $CurrentBest := \phi$ and $ActiveNodes := \{(I, 0)\}$.*
2. *If $ActiveNodes = \phi$ then go to step 15. Otherwise, let* lp *be some linear program in ActiveNodes.*
3. *$ActiveNodes := ActiveNodes - \{\mathrm{lp}\}$.*
4. *Compute the optimal solution s^{opt} for* lp *using Simplex, etc.*
5. *If s^{opt} is a permissible solution, then go to step 12.*
6. *(*Bound*) If $CurrentBest \neq \phi$ and $\Theta_L(s^{opt}) > \Theta_L(CurrentBest)$, then go to Step 2.*
7. *(*Branch*) Choose some variable $x \in \mathrm{lp}$ whose value in s^{opt} is non-permissible.*
8. *Set $I_1 := I \cup \{x \leq \lfloor s^{opt}(x) \rfloor_x\}$ and $I_2 := I \cup \{x \geq \lceil s^{opt}(x) \rceil_x\}$*
9. *Create two new linear programs:*
   $\mathrm{lp}_1 := (I_1, \Theta_L(s^{opt}))$ and $\mathrm{lp}_2 := (I_2, \Theta_L(s^{opt}))$.
10. *$ActiveNodes := ActiveNodes \cup \{\mathrm{lp}_1, \mathrm{lp}_2\}$.*
11. *Go to step 2.*
12. *(*Permissible solution*)*
    If $CurrentBest = \phi$ or $\Theta_L(s^{opt}) < \Theta_L(CurrentBest)$, then
    $CurrentBest := s^{opt}$.
13. *(*Pruning*) Remove from ActiveNodes all linear programs whose lower bounds are greater than $\Theta_L(CurrentBest)$.*
14. *Go to step 2.*
15. *(*Solution*) Print CurrentBest.*

What we have attempted to do in this formulation is to avoid the combinatorial explosion of $O(|R(A)| \times |R(C_1)| \times \ldots \times |R(C_n)|)$ by compressing it to $O(|R(A)| + |R(C_1)| + \ldots + |R(C_n)|)$. Our goal is to find such an optimal compression by manipulating encoders and splining functions.

One final note for this section is on the use of multiple encoders for a single r.v. By associating a separate encoder to each conditional table, this will obviously increase the flexibility of our formulation to compress the tables. Furthermore, multiple encoders will only increase our complexity linearly, we just have to guarantee consistency between encoders. If one encoder says that A is instantiated to a, any other encoders must also instantiate A to a. For example, say we have two encoders E_A^1 and E_A^2 and let $a \in R(A)$. Let x_A^1 and x_A^2 be the encoder variables associated to the two encoders respectively. Basically, we must make sure that $x_A^1 = E_A^1(a)$ iff $x_A^2 = E_A^2(a)$. We can accomplish this by the following constraints:

$$\begin{aligned}
x_A^1 - E_A^1(a) &\geq -M(1 - y_{A=a}) \\
x_A^1 - E_A^1(a) &\leq M(1 - y_{A=a}) \\
x_A^2 - E_A^2(a) &\geq -M(1 - y_{A=a}) \\
x_A^2 - E_A^2(a) &\leq M(1 - y_{A=a})
\end{aligned}$$

where $y_{A=a}$ is a 0-1 detector variable. Obviously, this can be generalized to any number of encoders.

7 Computing Splines

Our formulation in the previous section provides an approach to eliminating the necessity of explicitly storing conditional probability tables. By viewing these tables as look-up functions, we aim to replace them with simple continuous real functions. The heart of our formulation lies in the effective identification of encoder functions and a splining function. In particular, we are interested in approximation functions of the form

$$S(E_A(a), E_{C_1}(c_1), \ldots, E_{C_n}(c_n)) = e^{k_0 E_A a + k_1 E_{C_1}(c_1) + \ldots + k_n E_{C_n}(c_n) + k} \tag{28}$$

called LPFs in Definition 6.7. Hence, to complete our formulation, we must find such a function by determining the appropriate encoders and the method to properly combine them.

We start with a conditional probability table T that we would like to replace by an approximation function S. Assume T are probabilities of the form

$$P(C_0 = c_0 | C_1 = c_1, \ldots, C_n = c_n).$$

Ideally, we would like

$$e^{k_0 E_{C_0}(c_0)+k_1 E_{C_1}(c_1)+\ldots+k_n E_{C_n}(c_n)+k} \equiv P(C_0 = c_0 | C_1 = c_1, \ldots, C_n = c_n). \quad (29)$$

This can be rewritten in a simpler form as

$$\begin{aligned} k_0 E_{C_0}(c_0) + k_1 E_{C_1}(c_1) + \ldots + k_n E_{C_n}(c_n) + k \equiv \\ \ln P(C_0 = c_0 | C_1 = c_1, \ldots, C_n = c_n). \end{aligned} \quad (30)$$

As we can easily see, our goal is to determine the constants $k, k_0, k_1, \ldots, k_n$ and to determine the mapping of each particular instantiation of a r.v. to some real number. Hence, we have the following variables we must solve for:

- The constants $k, k_0, k_1, \ldots, k_n$.
- For each r.v. C_i, for each $c_i \in R(C_i)$, the encoding $E_{C_i}(c_i)$.

We can accomplish this by minimizing the following sum over the entries in table T:

$$\sum_{i=0}^{n} \sum_{c_i \in R(C_i)} \left[\sum_{j=0}^{n} k_j E_{C_j}(c_j) + k - \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) \right]^2 . \quad (31)$$

Obviously, this equation is some form of *least-squares fit*. We can find the minimum by taking the partial derivatives over (31) with respect to all the variables we must solve for; and then set these derivatives to 0. Unfortunately, the resulting system of equations is quadratic making it quite difficult to solve. Instead, we take a careful look at our problem and make the following observations: First, the encoders $E_A(a)$ are already variables themselves. We find that we can simply "absorb" the multiplicative constants in front of them in (31). We can now reduce the sum to:

$$\sum_{i=0}^{n} \sum_{c_i \in R(C_i)} \left[\sum_{j=0}^{n} E_{C_j}(c_j) + k - \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) \right]^2 . \quad (32)$$

We have now eliminated a collection of variables simplifying our summation. As a result, by taking the partial derivatives over (32), our system of equations is smaller and most importantly, linear in nature. Furthermore, the partial derivative equation obtained for k is actually a linear combination of the other equations. Hence, we can eliminate k from our summation which leaves us

$$\sum_{i=0}^{n} \sum_{c_i \in R(C_i)} \left[\sum_{j=0}^{n} E_{C_j}(c_j) - \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) \right]^2 . \tag{33}$$

Intuitively, what we are seeing in these absorptions is the following: The multiplicative constants k_i simply scales our data while the constant k is a translation.

As it turns out, the nature of our minimization problem has a closed form solution. Thus, we can avoid having to explicitly solve the system of linear equations. The typical Gaussian elimination techniques take a polynomial amount of time to run.

THEOREM 7.1. *The minimal solution to (31) will be for each r.v. C_k in T, and for each $c_k \in R(C_k)$,*

$$E_{C_k}(c_k) = \tag{34}$$

$$\frac{1}{\prod^{n}_{\substack{i=0 \\ i \neq k}} m_i} \left\{ \sum_{\substack{j=0 \\ j \neq k}}^{n} \sum_{c_j \in R(C_j)} \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) \right\} -$$
$$\frac{n\xi(T)}{(n+1)\prod_{i=0}^{n} m_i}$$

where $m_i = |R(C_i)|$ for $i = 0, \ldots, n$ and

$$\xi(T) = \sum_{j=0}^{n} \sum_{c_j \in R(C_j)} \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots).$$

We will now show how we derived the closed form equation (34).

NOTATION.

$$\kappa_k = \sum_{c_k \in R(C_k)} E_{C_k}(c_k).$$

For some partial derivation with respect to $E_{C_k}(c_k')$, our equation is

$$E_{C_k}(c_k') \prod_{\substack{i=0 \\ i \neq k}}^{n} m_i + \tag{35}$$

$$\sum_{\substack{j=0\\j\neq k}}^{n}\ \sum_{c_j\in R(C_j)}\left[E_{C_j}(c_j)\prod_{\substack{l=0\\l\neq k\\l\neq j}}^{n} m_l-\ln P(C_0=c_0|\ldots,C_l=c_l,\ldots)\right]=0.$$

We begin by summing over all possible c_k's. This results in

$$\sum_{c_k\in R(C_k)} E_{C_k}(c_k)\prod_{\substack{i=0\\i\neq k}}^{n} m_i =$$

$$\sum_{c_k\in R(C_k)}\ \sum_{\substack{j=0\\j\neq k}}^{n}\ \sum_{c_j\in R(C_j)}\left[\ln P(C_0=c_0|\ldots,C_l=c_l,\ldots)-E_{C_j}(c_j)\prod_{\substack{l=0\\l\neq k\\l\neq j}}^{n} m_l\right]$$

which can be rewritten as

$$\begin{aligned}\kappa_k\prod_{\substack{i=0\\i\neq k}}^{n} m_i &= \xi(T)-\sum_{\substack{j=0\\j\neq k}}^{n}\ \sum_{c_j\in R(C_j)} E_{C_j}(c_j)\prod_{\substack{l=0\\l\neq j}}^{n} m_l\\ &= \xi(T)-\sum_{\substack{j=0\\j\neq k}}^{n}\kappa_j\prod_{\substack{l=0\\l\neq j}}^{n} m_l.\end{aligned}$$

Putting all this together implies that

$$\sum_{j=0}^{n}\kappa_j\prod_{\substack{i=0\\i\neq j}}^{n} m_i=\xi(T).$$

Now, assume

$$\kappa_j=\frac{\xi(T)}{(n+1)\prod^{n}_{\substack{i=0\\i\neq j}} m_i}. \tag{36}$$

We rewrite equation (35) as

$$
\begin{aligned}
E_{C_k}(c'_k) \prod_{\substack{i=0 \\ i \neq k}}^{n} m_i = & \sum_{\substack{j=0 \\ j \neq k}}^{n} \sum_{c_j \in R(C_j)} \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) \\
& - \sum_{\substack{j=0 \\ j \neq k}}^{n} \kappa_j \prod_{\substack{l=0 \\ l \neq k \\ l \neq j}}^{n} m_l \\
= & \sum_{\substack{j=0 \\ j \neq k}}^{n} \sum_{c_j \in R(C_j)} \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) \\
& - \sum_{\substack{j=0 \\ j \neq k}}^{n} \frac{\xi(T)}{(n+1) \prod^{n}_{\substack{l=0 \\ l \neq j}} m_l} \prod_{\substack{l=0 \\ l \neq k \\ l \neq j}}^{n} m_l \\
= & \sum_{\substack{j=0 \\ j \neq k}}^{n} \sum_{c_j \in R(C_j)} \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) - \sum_{\substack{j=0 \\ j \neq k}}^{n} \frac{\xi(T)}{(n+1) m_k} \\
= & \sum_{\substack{j=0 \\ j \neq k}}^{n} \sum_{c_j \in R(C_j)} \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) - \frac{n \xi(T)}{(n+1) m_k}.
\end{aligned}
$$

We can assert (36) above because our system of linear equations is actually under-constrained. Now that we have obtained a closed form solution, we can also provide necessary and sufficient conditions on when an approximation function will fit the data perfectly.

Our best approximation function will be a perfect fit if and only if

$$\sum_{j=0}^{n} E_{C_j}(c_j) = \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) \tag{37}$$

for all $c_j \in R(C_j)$ for $j = 0, \ldots, n$. Take the closed form solution we obtained in Theorem 7.1 and plug it into (37). This gives us the following condition on the probabilities:

$$
\sum_{j=0}^{n} m_j \left[\sum_{\substack{k=0 \\ k \neq j}}^{n} \sum_{c'_k \in R(C_k)} \ln P(C_0 = c'_0 | C_1 = c'_1, \ldots, C_j = c_j, \ldots, C_n = c'_n) \right] -
$$

$$\ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) \prod_{i=0}^{n} m_i = \frac{n^2 \xi(T)}{(n+1)} \tag{38}$$

From this, the next theorem is obvious.

THEOREM 7.2. *There is a perfect fit if and only if (38) holds.*

In our above formulation, we made two implicit assumptions. First, we allowed encoders to map to non-positive values whereas Definition 6.1 allowed only positive ones. This can be easily solved by selecting the translation constant k such that all encoders are positive. As we demonstrated above, the choice of k has no bearing on our optimality criterion. The second assumption involved having the encoders be invertible. This can also be easily solved by perturbing the encoder values.

8 Example

We now present an example transforming a Bayesian network into our constraints formulation. In particular, we wish to transform the MTL problem we presented in Section 2. Unfortunately, due to the table size problem, MTL currently employs extremely coarse discretizations. There are a total of 5 possible robot actions, 10 possible map locations and only 3 possible sonar readings. The sonars read either NEAR, FAR or LOST. Instead, we assume 100 possible map locations and 100 possible sonar readings.

From Figure 2.1, we have the following r.v.s: $A_{R_i}, S_{R_i}, O_{R_i}, O_{T_i}, S_{T_i}$ for $i = 1, 2, 3, 4$. Let $m_{A_{R_i}} = |R(A_{R_i})|$, etc. Assume we have one LPF splining function for each conditional table. Furthermore, assume that each table has its own separate set of encoders.

Given the conditional table associated node S_{R_2}, we denote the associated encoders by, $E_{S_{R_1}, S_{R_2}}$ and $E_{A_{R_1}, S_{R_2}}$. The second index of the encoders represent the associated node. This will be similarly handled for all nodes. Let $F_{S_{R_2}}$ be the splining function associated with node S_{R_2}. This is again similarly done for all nodes.

For our transformation, we begin by constructing the following real variables: For each node A in the network, for each $B \in \mathsf{cond}(A)$, let $x_{B,A}$ represent the instantiation of B with respect to node A. Thus, $x_{B,A}$ is the encoded value of instantiations to B in the context of $E_{B,A}$.

Now, since we have multiple encoders, we must guarantee that they are consistent with one another. Let B be a r.v.. For each $b \in R(B)$, construct a 0-1 detector $y_{B=b}$. For each encoder E_{B,A_i}, construct the following two constraints:

$$\begin{aligned} x_{B,A_i} - E_{B,A_i}(b) &\geq -M(1 - y_{B=b}) \\ x_{B,A_i} - E_{B,A_i}(b) &\leq M(1 - y_{B=b}) \end{aligned}$$

Assume the our evidence is $O_{R_1} = s_1$, $O_{T_1} = s_2$. Thus, we must add the two evidence constraints:

$$\begin{aligned} x_{O_{R_1}, S_{R_1}} &= E_{O_{R_1}, S_{R_1}}(s_1) \\ x_{O_{T_1}, S_{T_1}} &= E_{O_{T_1}, S_{T_1}}(s_2) \end{aligned}$$

Finally, all that needs to be done is to construct the objective function. Given our splining functions F, our function is

$$\sum_A \ln F'(A)$$

where $F'(A)$ is simply the original splining function F_A with the encoder elements replaced by the appropriate real variables representing the instantiations. In Appendix B, we show the complete transformation of the MTL problem.

Our transformation involves 24 representation variables, 1615 consistency variables, 2 evidence constraints and 4060 consistency constraints. From our construction, the number of variables and constraints is clearly linear to the number of nodes in the network and the size of each r.v.s state space. Compare this against the total number of probabilities we would need to maintain in all the conditional tables which is 3,172,015!

9 Conclusions

Manipulating the enormous amount of probabilistic information needed in modeling real-world domains has proved to be the stumbling block for all previously existing computational methods. Up till now, each of these methods must access almost all of the probabilities to achieve the desired computations. Obviously, this has been the main factor in the combinatorially explosive execution-times. In particular, we have taken a look at a popular model for probabilistic reasoning called Bayesian networks. This explosion comes in the form of conditional probability tables which it must maintain.

The size of the table can me measured as $O(|R(A_1)| \times |R(A_2)| \times \ldots \times |R(A_n)|)$ where A_i's are r.v.s.

Our approach begins with an earlier successful model for computing on Bayesian networks. Without considering table-sizes, manipulating Bayesian networks have already proven to be NP-hard with respect to network size [5]. The earlier approach [23, 25, 24, 3, 26] transforms Bayesian networks into linear constraint satisfaction problems. Although the problem remains NP-hard, high levels of computational efficiency could be achieved by using tools and techniques from Operations Research. Tools such as Simplex method and Karmarkar's projective scaling algorithm formed the implementational core. Empirical studies using a crude implementation of Simplex outperformed existing methods actually exhibiting an expected polynomial run-time, $O(n^2)$. Furthermore, this approach is also invariant under network topology.

Our new approach augments the earlier one by compressing the tables and reformulating them into linear constraint satisfaction. We have provided a scheme which does not require explicit storage of this data. The compression is achieved by using splining functions as approximations to the conditional tables. Unfortunately, although conditional tables are look-up functions, the parameters used, namely r.v. instantiations, may not be numerical values as one would like them to be in a real function. For example, r.v. may be instantiated to objects, names, etc. The most straightforward approach would be to choose an arbitrary mapping from the instantiations to real values. For example, given a r.v. C representing color, a unique integer can be associated to each possible color, like "red" to 1, "green" to 2, etc. Now, a least-squares fit criterion can be used to determine the best fitting spline. Unfortunately, this technique may be somewhat restrictive. Too much seems to rest on our initial choice of mappings.

Instead, we provide an alternative technique which extends the least-squares fit criterion to simultaneously determine the optimal mapping for the r.v. instantiations. This now provides us with an immense amount of flexibility for finding an excellent approximating spline. Furthermore, the solution for finding the best mapping and spline can be computed by applying a very simple set of closed form equations. Furthermore, we can provide necessary and sufficient conditions on when a perfect fit occurs.

Putting this altogether, we have effectively reduced an originally

$$O(|R(A_1)| \times |R(A_2)| \times \ldots \times |R(A_n)|)$$

to a

$$O(|R(A_1)| + |R(A_2)| + \ldots + |R(A_n)|).$$

Our examples quite clearly demonstrate the savings that can be achieved through our compression.

Finally, the tools used in our approach have long been studied and are well understood. Dual Simplex, branch and bound and least-squares methods are very common computational techniques. We would like to note though that these methods are general methods. There are other methods which exploit domain-dependent information that we can likely use. Methods which rival Simplex have actually exhibited expected linear and even log-linear run-times which are obviously superior to our initial polynomial time [21, 14, 16, 22]. Thus, this helps to provide further evidence for the expected success of our approach.

References

[1] Eugene Charniak and Robert Goldman. A logic for semantic interpretation. In *Proceedings of the AAAI Conference*, 1988.

[2] Eugene Charniak and Drew McDermott. *Introduction to Artificial Intelligence*. Addison Wesley, 1985.

[3] Eugene Charniak and Eugene Santos, Jr. Dynamic map calculations for abduction. In *Proceedings of the AAAI-92*, 1992.

[4] Eugene Charniak and Solomon E. Shimony. Probabilistic semantics for cost based abduction. In *Proceedings of the AAAI Conference*, 1990.

[5] Gregory F. Cooper. Probabilistic inference using belief networks is np-hard. Technical Report KSL-87-27, Medical Computer Science Group, Stanford University, 1987.

[6] Steve B. Cousins, William Chen, and Mark E. Frisse. Caben: A collection of algorithms for belief networks. Technical Report WUCS-91-25, Department of Computer Science, Washington University, St. Louis, Mo., 1991.

[7] R. Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24:347–410, 1984.

[8] R. O. Duda, P. E. Hart, and N. J. Nilsson. Subjective bayesian methods for rule-based inference systems. In *Proceedings of the National Computer Conference*, 1976.

[9] Robert P. Goldman. *A Probabilistic Approach to Language Understanding*. PhD thesis, Department of Computer Science, Brown University, 1990.

[10] Robert P. Goldman and Eugene Charniak. Probabilistic text understanding. In *Proceedings of the Third International Workshop on AI and Statistics*, Fort Lauderdale, FL, 1991.

[11] M. Henrion. Propagating uncertainty by logic sampling in bayes' networks. Technical report, Department of Engineering and Public Policy, Carnegie-Mellon University, 1986.

[12] Jerry R. Hobbs, Mark Stickel, Paul Martin, and Douglas Edwards. Interpretation as abduction. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, 1988.

[13] Eric J. Horvitz, J. Jacques Suermondt, and Gregory F. Cooper. Bounded conditioning: Flexible inference for decisions under scarce resources. In

Proceedings of the Conference on Uncertainty in Artificial Intelligence, 1989.

[14] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 206–13, 1982.

[15] Jak Kirman, Kenneth Basye, and Thomas Dean. Sensor abstractions for control of navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2812–2817, 1991.

[16] Philip Klein, Serge A. Plotkin, C. Stein, and Eva Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. Technical Report Technical Report 961, Schools of Operations Research and Industrial Engineering, Cornell University, 1991.

[17] C. McMillan. *Mathematical Programming*. John-Wiley & Sons, Inc., 1975.

[18] G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors. *Optimization: Handbooks in Operations Research and Management Science Volume 1*, volume 1. North Holland, 1989.

[19] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[20] Y. Peng and J. A. Reggia. *Abductive Inference Models for Diagnostic Problem-Solving*. Springer-Verlag, 1990.

[21] Serge A. Plotkin, David B. Shmoys, and Eva Tardos. Fast approximation algorithms for fractional packing and covering problems. In *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science*, pages 495–504, 1991.

[22] P. Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing intereger programs. *J. Comput. System Sciences*, 37:130–43, 1988.

[23] Eugene Santos, Jr. Cost-based abduction, linear constraint satisfaction, and alternative explanations. In *Proceedings of the AAAI Workshop on Abduction*, 1991.

[24] Eugene Santos, Jr. A linear constraint satisfaction approach to cost-based abduction. Submitted to Artificial Intelligence Journal, 1991.

[25] Eugene Santos, Jr. On the generation of alternative explanations with implications for belief revision. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 1991.

[26] Eugene Santos, Jr. *A Linear Constraint Satisfaction Approach for Abductive Reasoning.* PhD thesis, Department of Computer Science, Brown University, 1992.

[27] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons Ltd., 1986.

[28] Ross D. Shachter. Evidence absorption and propagation through evidence reversals. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 1989.

[29] Ross D. Shachter and Mark A. Peot. Simulation approaches to general probabilistic inference on belief networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 1989.

[30] Murray Shanahan. Prediction is deduction but explanation is abduction. In *Proceeding of IJCAI Conference*, 1989.

[31] Solomon E. Shimony and Eugene Charniak. A new algorithm for finding map assignments to belief networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 1990.

[32] M. Shwe, B. Middleton, D. Heckerman, M. Henrion, E. Horvitz, and H. Lehmann. Probabilistic diagnosis using a reformulation of the internist-1/qmr knowledge base: I. the probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30:241–255, 1991.

[33] Sampath Srinivas and Jack Breese. Ideal: Influence diagram evaluation and analysis in lisp documentation and users guide. Technical Report Technical Memorandum No. 23, Rockwell International Science Center, Palo Alto, CA, 1989.

[34] Bon K. Sy. Reasoning mpe to multiply connected belief networks using message passing. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 1992.

A Proofs

THEOREM 6.2. *Any Bayesian network can be modeled as a splined Bayesian network.*

Proof. In brief, we simply partition each $\nabla(A)$ into single element cells. The rest of the construction of a splined Bayesian network should follow straightforwardly. $\Box$

THEOREM 6.3. *Given any cell* $d = \{a_1, \ldots, a_k\}$ *in* $[A]_{\overline{\mathcal{B}}}$ *where* $E_A(a_i) < E_A(a_{i+1})$*, there does not exist* $b \in R(A)$ *such that* $E_A(a_1) < E_A(b) < E_A(a_k)$ *and* $b \neq a_i$ *for* $i = 1, \ldots, k$.

Proof. This follows from the fact that the partitioning induced on A results from a contiguous partitioning of the conditional probabilities P and from Definition 6.2. $\Box$

THEOREM 6.4. w *is a complete instantiation-set for* $\mathcal{B}$ *if and only* $s[w]$ *is a permissible solution for the induced constraint system.*

Proof. Given a complete instantiation-set w for $\mathcal{B}$, assume that $s[w]$ is a permissible assignment for the induced constraint system but not a permissible solution. This implies that one of the following constraints have been violated: (For simplicity, we denote $s[w]$ by s.)

1. $\sum_{i=1}^{n} x_{d_{A,i}} = 1$
2. $x_A - \min_{a \in d_{A,i}} E_A(a) \geq -M(1 - x_{d_{A,i}})$
3. $x_A - \max_{a \in d_{A,i}} E_A(a) \leq M(1 - x_{d_{A,i}})$
4. $x_{C_i} - \min(C_i, S) \geq -M(1 - d_{C_i,S})$
5. $x_{C_i} - \max(C_i, S) \leq M(1 - d_{C_i,S})$
6. $d_S \geq n - \sum_{j=1}^{n} d_{C_j,S} + 1$
7. $d_S \leq \frac{1}{n} \sum_{j=1}^{n} d_{C_j,S}$
8. $x_{C_i,S} \geq x_{C_i} - M(1 - d_S)$
9. $x_{C_i,S} \leq x_{C_i} + M(1 - d_S)$
10. $x_{C_i,S} \leq M d_S$

- CASE 1. $\sum_{i=1}^{n} x_{d_{A,i}} = 1$ is violated. This implies that

  $$s(x_{d_{A,1}}) = \ldots = s(x_{d_{A,n}}) = 0.$$

  From our construction, we see that A has been instantiated and for some i, $s(x_{d_{A,i}}) = 1$. Contradiction.

- CASE 2. $x_A - \min_{a \in d_{A,i}} E_A(a) \geq -M(1 - x_{d_{A,i}})$ has been violated. This implies that $s(x_A)$ is in the designated interval but $s(x_{d_{A,i}}) = 0$. This is a contradiction according to our construction.
- CASES 3, 4 AND 5. The reasoning is similar to CASE 2.
- CASE 6. $d_S \geq n - \sum_{j=1}^{n} d_{C_j,S} + 1$ is violated. This implies that $s(d_S) = 0$ and $s(d_{C_j,S}) = 1$ for all j. However, this violates our notion of an active splining function from our construction. Contradiction.
- CASE 7. $d_S \leq \frac{1}{n} \sum_{j=1}^{n} d_{C_j,S}$ is violated. This implies that $s(d_S) = 1$ and $s(d_{C_j,S}) = 0$ for all j. However, this violates our notion of an active splining function from our construction. Contradiction.
- CASE 8. $x_{C_i,S} \geq x_{C_i} - M(1 - d_S)$ is violated. This implies that $s(d_S) = 1$ and $s(x_{C_i,S}) < s(x_{C_i})$. But from our construction, when a splining function is active, $s(x_{C_i,S}) = s(x_{C_i})$. Contradiction.
- CASES 9 AND 10. Similar to CASE 8.

Contradiction. There $s[w]$ is a permissible solution.

Now, given a permissible solution s. Assume that $w[s]$ is not a complete instantiation-set. This implies that there exists a r.v. A in V such that B is not instantiated in $w[s]$. Let A be such a r.v.. From our construction, this implies that $s(x_{D_{d,i}}) = 0$ for all i. However, this violates constraint (17). Contradiction. □

THEOREM 6.5.

$$P(w) = e^{-\Theta(s[w])}.$$

Proof. Follows from our construction and Theorem 6.4. □

THEOREM 7.1. *The minimal solution to (31) will be for each r.v. C_k in T, and for each $c_k \in R(C_k)$,*

$$E_{C_k}(c_k) = \tag{39}$$

$$\frac{1}{\prod^{n}_{\substack{i=0 \\ i \neq k}} m_i} \left\{ \sum_{\substack{j=0 \\ j \neq k}}^{n} \sum_{c_j \in R(C_j)} \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) \right\} -$$

$$\frac{n\xi(T)}{(n+1)\prod_{i=0}^{n} m_i}$$

where $m_i = |R(C_i)|$ *for* $i = 0, \ldots, n$ *and*

$$\xi(T) = \sum_{j=0}^{n} \sum_{c_j \in R(C_j)} \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots).$$

Proof. We prove this by simply plugging the values for $E_{C_k}(c_k)$ into the linear equations. For some partial derivation with respect to $E_{C_k}(c_k')$, our equation is

$$E_{C_k}(c_k') \prod_{\substack{i=0 \\ i \neq k}}^{n} m_i +$$

$$\sum_{\substack{j=0 \\ j \neq k}}^{n} \sum_{c_j \in R(C_j)} \left[E_{C_j}(c_j) \prod_{\substack{l=0 \\ l \neq k \\ l \neq j}}^{n} m_l - \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) \right] = 0.$$

By substituting back our solutions on the lefthand side, we get

$$\sum_{\substack{j=0 \\ j \neq k}}^{n} \sum_{c_j \in R(C_j)} \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) -$$

$$\frac{n\xi(T)}{(n+1)m_k} - \sum_{\substack{j=0 \\ j \neq k}}^{n} \sum_{c_j \in R(C_j)} \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) +$$

$$\sum_{\substack{j=0 \\ j \neq k}}^{n} \sum_{c_j \in R(C_j)} \frac{1}{m_k} \left[\sum_{\substack{i=0 \\ i \neq j}}^{n} \sum_{c_i \in R(C_i)} \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) - \right.$$

$$\left. \frac{n\xi(T)}{(n+1)m_j} \right]$$

$$= -\frac{n\xi(T)}{(n+1)m_k} +$$

$$\sum_{\substack{j=0 \\ j \neq k}}^{n} \frac{1}{m_k} \left[\sum_{c_j \in R(C_j)} \sum_{\substack{i=0 \\ i \neq j}}^{n} \sum_{c_i \in R(C_i)} \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) - \right.$$

$$
\left.\frac{n\xi(T)}{(n+1)}\right] = -\frac{n\xi(T)}{(n+1)m_k} + \sum_{\substack{j=0 \\ j\neq k}}^{n} \left[\frac{\xi(T)}{m_k} - \frac{n\xi(T)}{(n+1)m_k}\right]
$$

$$
\begin{aligned}
&= -\frac{n\xi(T)}{(n+1)m_k} + \frac{n\xi(T)}{m_k} - \frac{n^2\xi(T)}{(n+1)m_k} \\
&= -\frac{n\xi(T)}{(n+1)m_k} + \frac{n(n+1)\xi(T)}{(n+1)m_k} - \frac{n^2\xi(T)}{(n+1)m_k} \\
&= -\frac{n\xi(T)}{(n+1)m_k} + \frac{n^2\xi(T)}{(n+1)m_k} + \frac{n\xi(T)}{(n+1)m_k} - \frac{n^2\xi(T)}{(n+1)m_k} \\
&= 0.
\end{aligned}
$$

□

THEOREM 7.2. *There is a perfect fit if and only if (38) holds.*

Proof. This follows immediately from our minimization problem and Theorem 7.1. □

B MTL Transformation

Assume the following:

- $R(A_{R_i}) = \{a_1, a_2, \ldots, a_5\}$ for $i = 1, 2, 3$.
- $R(S_{R_i}) = R(S_{T_i}) = \{l_1, \ldots, l_{100}\}$ for $i = 1, 2, 3, 4$.
- $R(O_{R_i}) = R(O_{T_i}) = \{s_1, \ldots, s_{100}\}$ for $i = 1, 2, 3, 4$.

We have the two evidence constraints:

$$\begin{aligned} x_{O_{R_1}, S_{R_1}} &= E_{O_{R_1}, S_{R_1}}(s_1) \\ x_{O_{T_1}, S_{T_1}} &= E_{O_{T_1}, S_{T_1}}(s_2) \end{aligned}$$

We have the following consistency constraints: For $i = 1, 2, 3$, for each A_{R_i}, for $j = 1, \ldots, 5$,

$$\begin{aligned} x_{A_{R_i}, S_{R_{i+1}}} - E_{A_{R_i}, S_{R_{i+1}}}(a_j) &\geq -M(1 - y_{A_{R_i}=a_j}) \\ x_{A_{R_i}, S_{R_{i+1}}} - E_{A_{R_i}, S_{R_{i+1}}}(a_j) &\leq M(1 - y_{A_{R_i}=a_j}) \\ x_{A_{R_i}, O_{R_{i+1}}} - E_{A_{R_i}, O_{R_{i+1}}}(a_j) &\geq -M(1 - y_{A_{R_i}=a_j}) \\ x_{A_{R_i}, O_{R_{i+1}}} - E_{A_{R_i}, O_{R_{i+1}}}(a_j) &\leq M(1 - y_{A_{R_i}=a_j}) \end{aligned}$$

For each $i = 1, \ldots, 4$, for each S_{R_i}, for each $j = 1, \ldots, 100$,

$$\begin{aligned} x_{S_{R_i}, S_{R_{i+1}}} - E_{S_{R_i}, S_{R_{i+1}}}(l_j) &\geq -M(1 - y_{S_{R_i}=l_j}) \\ x_{S_{R_i}, S_{R_{i+1}}} - E_{S_{R_i}, S_{R_{i+1}}}(l_j) &\leq M(1 - y_{S_{R_i}=l_j}) \\ x_{S_{R_i}, S_{T_i}} - E_{S_{R_i}, S_{T_i}}(l_j) &\geq -M(1 - y_{S_{R_i}=l_j}) \\ x_{S_{R_i}, S_{T_i}} - E_{S_{R_i}, T_{R_i}}(l_j) &\leq M(1 - y_{S_{R_i}=l_j}) \end{aligned}$$

For each $i = 1, \ldots, 4$, for each S_{T_i}, for each $j = 1, \ldots, 100$,

$$\begin{aligned} x_{S_{T_i}, S_{T_{i+1}}} - E_{S_{T_i}, S_{T_{i+1}}}(l_j) &\geq -M(1 - y_{S_{T_i}=l_j}) \\ x_{S_{T_i}, S_{T_{i+1}}} - E_{S_{T_i}, S_{T_{i+1}}}(l_j) &\leq M(1 - y_{S_{T_i}=l_j}) \end{aligned}$$

For each $i = 1, \ldots, 4$, for each O_{R_i}, for each $j = 1, \ldots, 100$,

$$\begin{aligned} x_{O_{R_i}, S_{R_i}} - E_{O_{R_i}, S_{R_i}}(s_j) &\geq -M(1 - y_{O_{R_i}=s_j}) \\ x_{O_{R_i}, S_{R_i}} - E_{O_{R_i}, S_{R_i}}(s_j) &\leq M(1 - y_{O_{R_i}=s_j}) \end{aligned}$$

For each $i = 1, \ldots, 4$, for each O_{T_i}, for each $j = 1, \ldots, 100$,

$$\begin{aligned} x_{O_{T_i}, S_{T_i}} - E_{O_{T_i}, S_{T_i}}(s_j) &\geq -M(1 - y_{O_{T_i} = s_j}) \\ x_{O_{T_i}, S_{T_i}} - E_{O_{T_i}, S_{T_i}}(s_j) &\leq M(1 - y_{O_{T_i} = s_j}) \end{aligned}$$

Finally, the objective function is

$$\begin{aligned}
& - \textstyle\sum_{i=1}^{3} \left[K_{A_{R_i}} + k_{A_{R_i}} x_{A_{R_i}, S_{R_{i+1}}} \right] \\
& - \left[K_{S_{R_1}} + k_{S_{R_1}} x_{S_{R_1}, S_{R_2}} + k_{O_{R_1}, S_{R_1}} x_{O_{R_1}, S_{R_1}} \right] \\
& - \textstyle\sum_{i=2}^{4} \left[K_{S_{R_i}} + k_{S_{R_i}} x_{S_{R_i}, S_{R_{i+1}}} + k_{A_{R_{i-1}}, S_{R_i}} x_{A_{R_{i-1}}, S_{R_i}} + k_{O_{R_i}, S_{R_i}} x_{O_{R_i}, S_{R_i}} \right] \\
& - \left[K_{O_{R_1}} + k_{O_{R_1}} x_{O_{R_1}, S_{R_1}} \right] \\
& - \textstyle\sum_{i=2}^{4} \left[K_{O_{R_i}} + k_{O_{R_i}} x_{O_{R_i}, S_{R_i}} + k_{A_{R_{i-1}}, O_{R_i}} x_{A_{R_{i-1}}, O_{R_i}} \right] \\
& - \textstyle\sum_{i=1}^{4} \left[K_{O_{T_i}} + k_{O_{T_i}} x_{O_{T_i}, S_{T_i}} \right] \\
& - \left[K_{S_{T_1}} + k_{S_{T_1}} x_{S_{T_1}, S_{T_2}} + k_{O_{T_1}, S_{T_1}} x_{O_{T_1}, S_{T_1}} \right] \\
& - \textstyle\sum_{i=2}^{4} \left[K_{S_{T_i}} + k_{S_{T_i}} x_{S_{T_i}, S_{T_{i+1}}} + k_{S_{R_i}, S_{T_i}} x_{S_{R_i}, S_{T_i}} + k_{O_{T_i}, S_{T_i}} x_{O_{T_i}, S_{T_i}} \right].
\end{aligned}$$