Check for updates

On Linear Potential Functions for Approximating Bayesian Computations

EUGENE SANTOS, JR.

Air Force Institute of Technology, Wright-Patterson AFB, Ohio

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 using approximation functions called linear potential functions. We can potentially reduce the information from a combinatorial amount to roughly linear in the number of random variable assignments. Furthermore, we can compute these functions through closed form equations. As it turns out, our approximation method is quite general and may be applied to other data compression problems.

Categories and Subject Descriptors: E.4 [**Coding and Information Theory**]; G.1.2 [**Numerical Analysis**]: Approximation; G.1.6 [**Numerical Analysis**]: Optimization; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving; I.5 [**Pattern Recognition**]

General Terms: Algorithms, Design, Experimentation, Theory

Additional Key Words and Phrases: Artificial intelligence, data compaction and compression, integer programming, least squares approximation, pattern recognition, probabilistic reasoning, uncertainty

1. *Introduction*

Probabilistic reasoning has become the mainstay of many inferencing systems. From expert systems for geological surveying [Duda et al. 1976] to medical diagnosis systems [Shwe et al. 1991], the probabilistic approach provides a rich framework for knowledge representation that is essential in these domains. Unfortunately, probabilistic reasoning in general is NP-hard. Thus, to compensate for this, the systems built tend to use unrealistic assumptions such as independence, and hence, often have various behavioral anomalies.

Probably one of the most popular models for probabilistic reasoning is Bayesian networks [Pearl 1988]. This approach provides a handy visualization of

---

This research was supported in part by AFOSR Project #940006.

Author's address: Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH 45433-7765; esantos@afit.af.mil.

Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery (ACM), Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.
© 1996 ACM 0004-5411/96/0500-0399 $03.50

our knowledge using a directed acyclic graph of random variable relationships. In the discrete case, each node in the graph represents a "discrete" random variable.[1] The directed arcs between the nodes represent probabilistic conditional (in)dependencies. Joint probabilities on the random variables can be readily computed via the chain rule 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,[2] circuit fault detection [Davis 1984; Pearl 1988], medical diagnosis [Shwe et al. 1991], and planning systems [Kirman et al. 1991]. However, computing with these networks has been proven to be NP-hard as we might have expected [Cooper 1987; Shimony and Charniak 1990; Dagum and Luby 1993]. 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 the topology of the network and this is often the cause of combinatorial explosion (see Pearl [1988] and Shimony and Charniak [1990].) For example, Pearl's early message-passing scheme [Pearl 1988] is restricted to singly connected networks, that is, there can only exist at most one path between any two nodes in the graph.[3] Unfortunately, even for those which are not hindered by topological considerations, they invariably run into the second factor, *conditional probability table size*.

Recently, various approaches to the table size problem have been developed such as independence-based assignments [Shimony 1993; Santos, Jr. and Shimony 1994] and "Noisy-OR" models [Peng and Reggia 1986; Pearl 1988; and Srinivas 1993]. These approaches attempt to exploit domain dependent properties of the problem. The former uses "finer" independence assumptions and the latter disjunctive interactions or its generalizations. Unfortunately, the independence-based approach is too restrictive while the "Noisy-OR" approach relies on successfully identifying the interactions.

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 involved in this particular table. 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 [Pearl 1988], domains such as robotic navigation require rather precise problem formulations to handle quantitative data such as sonar readings and map locations [Kirman et al. 1991; 1993]. The higher the degree of

[1] For Bayesian networks based solely on continuous random variables, see Pearl [1988].

[2] See, for example, Goldman [1990], Goldman and Charniak [1991], Charniak and Goldman [1988], and Charniak and Santos, Jr. [1992].

[3] Other more recent schemes such as Sy's forward propagation [Sy 1992] are still limited to singly-connected networks.

sensitivity needed in this data implies an even finer grained discretization of random variable instantiations.

The explosive growth of probability tables for Bayesian networks is a special case of a more general problem encountered in probabilistic systems. There is a need of having large amounts of data on hand such as correlations, statistical frequencies, conditional probabilities, etc. 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 two symptoms are sufficient for determining a disease or we may choose to quantize distance information to something like near, medium, far. However, we run into the classic dilemma of trading off representational power for computability. Still, more often than not, we end up with overly simple models, which themselves are just barely computable. As we shall see, much of the problem involving table size spawns from the need to reference each and every probability during the computations. One such example is the search for the "most-probable answer" in belief revision [Pearl 1988; Sy 1992]. This search often involves finding the optimal conditional probability in each of these tables while under certain random variable assignment restrictions [Sy 1992]. For example, searching for the largest probability in which certain random variables are set to true. Since the random variables involved are discrete, this amounts to an enumeration type search. Compounding the problem is the fact that the values in the table are unordered hence requiring an exhaustive search.

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 is designed to avoid the necessity of an exhaustive enumeration search. Basically, we use an approximation function called a *linear potential function* (LPF) to capture all the values in a given table. The values for each particular entry in the table can be retrieved from the approximation by passing the appropriate random variable instantiations as the function's parameters. Once this is achieved, we merge it into an integer linear programming approach for belief revision.

The biggest problem arises in trying to map the different possible instantiations for the random variables to some real number. Values for random variables need not be numerical.[4] Often, they are symbolic in nature such as representing the color of some object. For example, random variable A is assigned **red**. Taking this a step further, the numerical values used by those random variables which discretize information such as distance may not be the best mappings. Hence, we have the additional problem of finding the optimal encoder (mapping) as well.

We present a model for determining the best LPF as well as the optimal encoding. On the surface, it seems that the only approach would be to first pick some "optimal" encoder and then construct an approximation function. However, such an approach is somewhat ad-hoc since the "best" approximations necessarily entwine both encoder choice and approximation simultaneously. In this paper, we have managed to develop a model that does unify both, but most importantly allows us to determine the optimal compression with a closed form solution. Hence, we have a fast and simple approach for computing an optimal LPF. Furthermore, we can provide necessary and sufficient conditions for when a

[4] The numerical values are necessary since we will recast our computations for linear programming.

perfect approximation can be found. This compression reduces a table that requires $O(|R(C_0)| \times |R(C_1)| \times \cdots \times |R(C_n)|)$ space to a linear potential function which only requires $O(|R(C_0)| + \cdots + |R(C_n)|)$, where the C_is are the random variables involved and $|R(C_i)|$ is the number of possible instantiations to C_i.

Once we determine the optimal approximations of the conditional tables, we must now somehow use it in our computations for the "most-probable answer". We demonstrate this by incorporating these approximations into an existing approach for probabilistic computations. Although the conditional tables of a Bayesian network share random variables, we can decouple them by allowing different encoders for different tables.

Recently, Santos, Jr. [1991; 1993; 1994] and Charniak and Santos, Jr. [1992] introduced a new technique for probabilistic computations using linear constraint satisfaction. Extremely efficient tools and techniques can be applied from Operations Research to perform the 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. Thus, we have an excellent launching point for exploiting our approximation functions.

We begin in Section 2 by briefly describing Bayesian networks and their uses. In Section 4, we present our approximation technique for the probability tables. Now with these LPFs, we provide a new approach for computing with them through linear constraint satisfaction in Sections 5 and 6.

2. *Bayesian Networks and Belief Revision*

In probabilistic reasoning, random variables are used to represent events and/or objects in the world. By making various instantiations to these random variables, we can model the current state of the world. Thus, this will involve computing joint probabilities of the given random variables. Unfortunately, the task is nearly impossible without additional information concerning relationships between the rvs. In the worst case, we would need the probabilities of every instantiation combination that is combinatorially explosive.

On the other hand, consider the chain rule as follows:

$$P(A_1, A_2, A_3, A_4, A_5) = P(A_1|A_2, A_3, A_4, A_5) * P(A_2|A_3, A_4, A_5)P(A_3|A_4, A_5)P(A_4|A_5)P(A_5). \quad (1)$$

Bayesian networks [Pearl 1988] take this process further by making the important observation that certain random variable pairs may become uncorrelated once information concerning some other random variable(s) is known. More precisely, we may have the following independence condition:

$$P(A|C_1, \ldots, C_n, U) = P(A|C_1, \ldots, C_n) \quad (2)$$

for some collection of random variables U. Intuitively, we can interpret this as saying that A is determined by $C_1, \ldots, C_n$ regardless of U.

Combined with chain rule, 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 that we can then use to compute the joint probabilities.

What we have is a directed acyclic graph of random variable relationships. Directed arcs between random variables represent conditional dependencies. When all the parents of a given random variable A are instantiated, that random variable is said to be conditionally independent of the remaining random variables, which are not descendants of A, given its parents (see (2)).[5]

For example, let's consider the following story: Mary walks outside and finds that the street and lawn are wet. She concludes that it has just rained recently. Furthermore, she decides that she does not need to water her climbing roses.

Assume that Mary used the following set of rules:

$$\text{rain} \vee \text{sprinklers} \Rightarrow \text{street-wet}$$

$$\text{rain} \vee \text{sprinklers} \Rightarrow \text{lawn-wet}$$

$$\text{lawn-wet} \Rightarrow \text{soil-moist}$$

$$\text{soil-moist} \Rightarrow \text{roses-okay}$$

We can directly transform these into a graph. Now, by considering each variable as a random variable with possible states of {true, false}, we can construct conditional probability tables for (1) which reflects our knowledge of the world (see Figure 1). Let's compute the joint probability of the world where the roses are okay, the soil is dry, the lawn is wet, the street is wet, the sprinklers are off and it is raining.

$$\begin{aligned} &P(\text{sprinklers} = F, \text{rain} = T, \text{street} = \text{wet}, \text{lawn} = \text{wet}, \\ &\quad \text{soil} = \text{dry}, \text{roses} = \text{okay}) \\ &\quad = P(\text{roses} = \text{okay} \mid \text{soil} = \text{dry}) \\ &\qquad * P(\text{soil} = \text{dry} \mid \text{lawn} = \text{wet}) \\ &\qquad * P(\text{lawn} = \text{wet} \mid \text{rain} = T, \text{sprinklers} = F) \\ &\qquad * P(\text{street} = \text{wet} \mid \text{rain} = T, \text{sprinklers} = F) \\ &\qquad * P(\text{sprinklers} = F) \\ &\qquad * P(\text{rain} = T). \end{aligned}$$

Substituting the appropriate numbers from the tables, we get $0.2 * 0.4 * 1.0 * 1.0 * 0.6 * 0.7 = 0.0336$ as the probability of this scenario.

(*For a more concrete application of Bayesian networks, see Appendix A for the Mobile Target Localization problem* [Kirman et al. 1991; 1993].)

There are two types of computations performed with Bayesian networks: *belief updating* and *belief revision* [Pearl 1988]. Belief updating concerns the computation of probabilities over random variables, while belief revision concerns finding

[5] For more details on this, see *d-separation* in Pearl [1988].

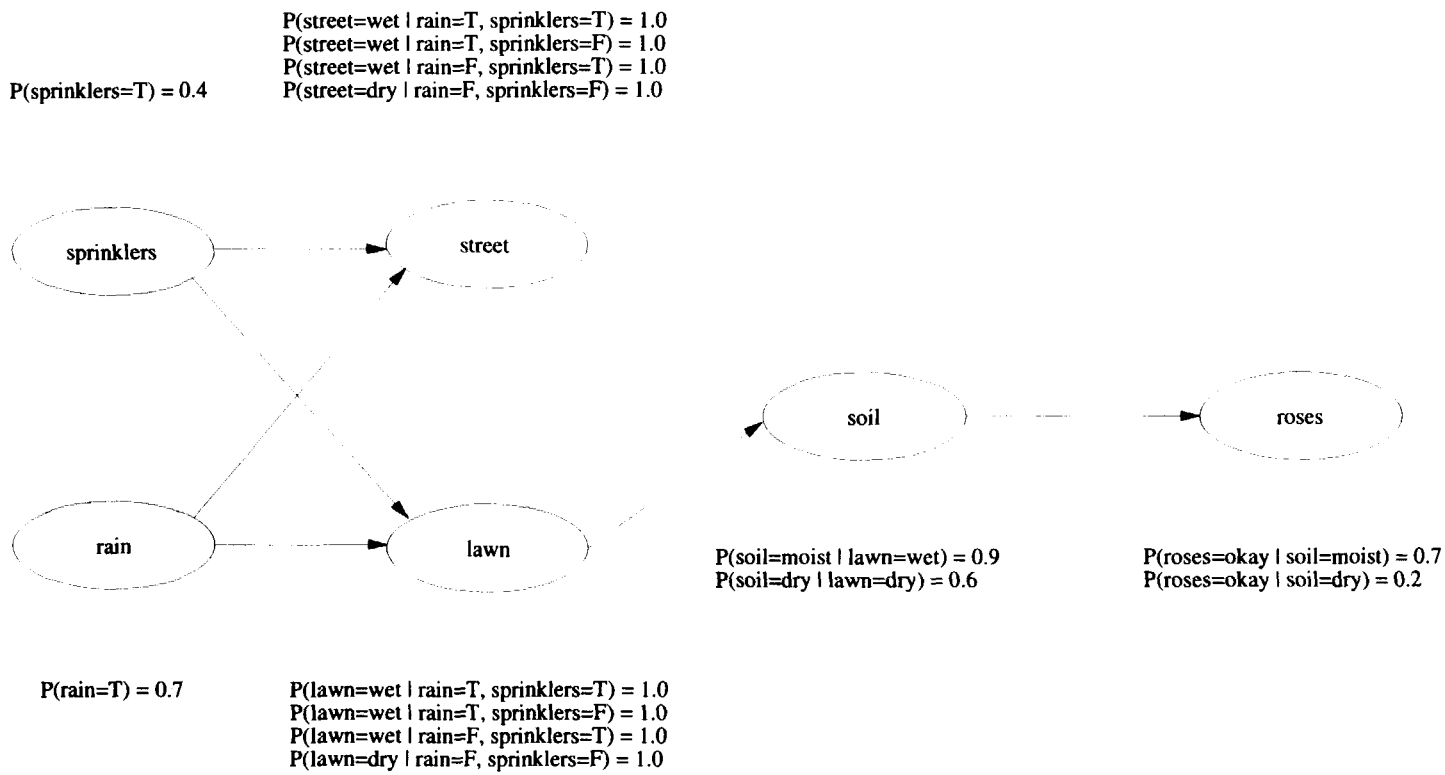

FIG. 1. Mary's Bayesian network. Conditional tables at each node must contain all possible combinations of assignments. For space purposes, we give a reduced collection. We can compute the missing information by taking 1 minus the appropriate probabilities given.

the maximally probable global assignment. In this paper, we concentrate on belief revision. We note, though, that belief updating can also be computed using belief revision [Kirman et al. 1993; Santos, Jr. and Shimony 1994].

Belief revision can be 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.[6] More formally, if W is the set of all random variables in our given Bayesian network and e is our given evidence,[7] any complete instantiations to all the random variables 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

$$P(w^*|e) = \max_w P(w|e). \tag{3}$$

w^* is called the "most-probable explanation" (MPE).[8] Intuitively, we can think of the nonevidence random variables in W as possible hypotheses for e.

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. Consider the Mobile Target Location (MTL) problem[9] [Kirman et al. 1991; 1993], which involves a robot locating and following a mobile target using Bayesian networks. We consider the following conditional table:

$$P(O_T|L_R, L_T) \tag{4}$$

where L represents the locations of the robot and target and where O represents the particular sonar readings on the target. The size of this table is $|L|^2|O|$, where $|L|$ is the number of possible map positions in the world and $|O|$ is the range of instantiations for the sonar readings O_T. 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 are only 100 possible locations in our world and possible sonar readings, the size of this table alone would be one million. This remains a major prohibiting factor in trying to use Bayesian networks for modeling complex tasks.

3. *Terms, Definitions, and Notations*

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

[6] See, for example, Hobbs et al. [1988], Shanahan [1989], Peng and Reggia [1990], and Charniak and Shimony [1990].

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

[8] Belief updating on the other hand is interested only in the marginal probabilities of a subset of random variables given the evidence. Typically, it is to determine the best instantiation of a single random variable given the evidence.

Also note that to compute the MPE for e, it is sufficient to determine the complete assignment consistent with e whose joint probability is maximal. In this case, $P(e)$ is simple a constant factor.

[9] See Appendix A.

Simply put, a Bayesian network is a directed acyclic graph whose nodes represent random variables and edges represent direct conditional dependencies, that is, a random variable A is conditionally independent of all nodes not descended from A given all its parents. Hence, each node A also has an associated table of all the conditional probabilities between A and its parents (see Figure 1).

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 random variables.[10]

Notation 3.1. Throughout the remainder of this paper, upper case italicized letters such as $A, B, \ldots$ will represent random variables and lowercase italicized letters such as $a, b, \ldots$ will represent the possible assignments to the associated uppercase-letter random variable, in this case, $A, B, \ldots$. Subscripted uppercase letters that are not italicized are variables in a constraint system which explicitly represent the instantiation of the associated random variable with the item in the subscript. For example, A_a denotes the instantiation of random variable A with value a.

Notation 3.2. Given a random variable 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 random variables 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. (V, P) completely describes the Bayesian network.

Definition 3.3. Given a Bayesian network $\mathfrak{B} = (V, P)$, 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 implies $a = a'$.

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

Definition 3.4. Given an instantiation-set w for a Bayesian network $\mathfrak{B} = (V, P)$, we define the *span* of w, $\mathsf{span}(w)$, to be the collection of random variables in the *first coordinate of the instantiations. Furthermore, an instantiation-set w is said to be complete iff* $\mathsf{span}(w) = V$.

The span of an instantiation-set simply denotes the random variables that have been instantiated.

Notation 3.5. For each random variable, A, we define $\mathsf{cond}(A)$ as follows: $B \in \mathsf{cond}(A)$ iff there exists a conditional probability in P of the form $P(A = a| \ldots, B = b, \ldots)$.

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

Intuitively, $\mathsf{cond}(A)$ is the set of random variables which are the parents of A.

Notation 3.6. 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 3.7. 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

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

4. *Approximation Functions*

To avoid the exponential explosion in our conditional table size, our approach is to compress the information. Through the use of approximation functions we call *linear potential functions* (LPFs), 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 approximating function, but even multiple functions are still more desirable than having the explicit table. These functions are real-valued functions from $\Re^n$ into $\Re$, where $\Re$ is the real line and $\Re^n$ is the n-dimensional Euclidean space. The range of these functions will correspond to the conditional probabilities whereas the domain will be used as indices in fetching/computing the appropriate conditional probabilities. Unfortunately, instantiations for random variables 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 4.1. $\mathcal{Q}$ is the set of rational numbers. $\mathcal{Q}^+$ is the set of positive rational numbers.[11]

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

Definition 4.2. Given a rv A in V, a one-to-one 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 random variable. Furthermore, it provides a mechanism for flexibly transforming/reorganizing our instantiations to a possibly more useful form or interpretation. For example, consider the simple table consisting of only one random variable, say A.[12] Assume $R(A) = \{\text{red, green, yellow, blue, purple}\}$ and the table is

$$P(A = \text{red}) = 0.02$$

$$P(A = \text{green}) = 0.50$$

$$P(A = \text{yellow}) = 0.00$$

$$P(A = \text{blue}) = 0.37$$

[11] Since we are intending to use LPFs in linear programs, variables in linear programs must have nonnegative values.

[12] This would correspond to a root node in our Bayesian network such as rain and sprinklers in Figure 1.

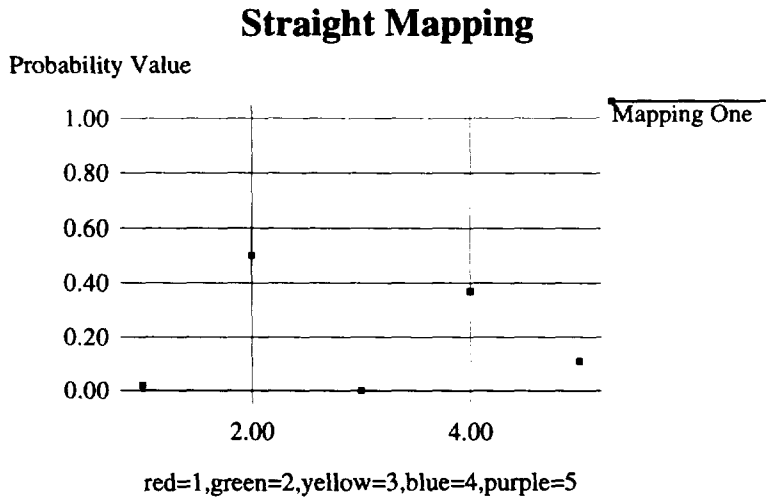

FIG. 2. Encoding 1 for random variable A.

$$P(A = \text{purple}) = 0.11.$$

First, let's use the simple encoding of the colors to 1, 2, 3, 4, 5 based in the order they were presented above. We get the resulting graph in Figure 2. Now, let's choose a more intelligent encoding scheme as in Figure 3. The approximation for our second encoding is simply a line through all the points as opposed to the more sophisticated interpolation curve we would need for the first encoding.

As we have just seen, the choice of encoders will have a major impact on the success of our compression/approximation. The problem is certainly easier if we could choose encoders for the random variables independently. However, in tables involving more than one random variable, the encoders must interact which can result in convoluted probability spaces worse than Figure 2.

By viewing these tables as look-up functions, our goal is to replace them with simple continuous real functions. The heart of our formulation lies in the effective identification of encoders and an approximation function. In particular, we are interested in approximation functions of the form

$$S(E_A(a), E_{C_1}(c_1), \ldots, E_{C_n}(c_n)) = \exp(k_0 E_A(a) + k_1 E_{C_1}(c_1) + \cdots + k_n E_{C_n}(c_n) + k). \quad (5)$$

Hence, we must find such a function by simultaneously determining the appropriate encoders and the method for properly combining them. Our choice of the LPF class is tied closely to our later integer linear programming approach for belief revision presented later.

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

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

Notation 4.3. If $P(C_0 = c_0 | C_1 = c_1, \ldots, C_n = c_n)$ is in T, we simply rewrite this as $(c_0, c_1, \ldots, c_n) \in T$.

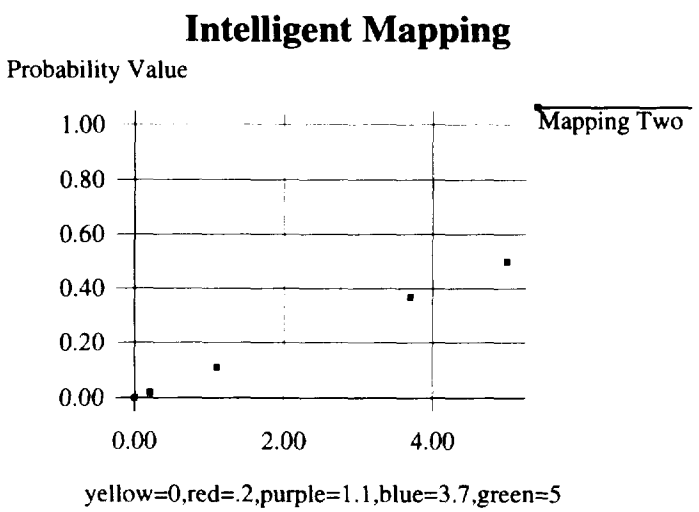

FIG. 3. Encoding 2 for random variable A.

Ideally, we would like

$$\exp(k_0 E_{C_0}(c_0) + k_1 E_{C_1}(c_1) + \cdots + k_n E_{C_n}(c_n) + k) = P(C_0 = c_0 | C_1 = c_1, \ldots, C_n = c_n). \quad (6)$$

This can be rewritten in a simpler form as

$$k_0 E_{C_0}(c_0) + k_1 E_{C_1}(c_1) + \cdots + k_n E_{C_n}(c_n) + k = \ln P(C_0 = c_0 | C_1 = c_1, \ldots, C_n = c_n). \quad (7)$$

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 random variable, to some real number. Hence, we have the following variables that we must determine:

—The constants $k, k_0, k_1, \ldots, k_n$.

—For each rv 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:

$$\sum_{(c_0, \ldots, c_n) \in T} \left[\sum_{j=0}^{n} k_j E_{C_j}(c_j) + k - \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) \right]^2. \quad (8)$$

This equation is some form of *least-squares fit*. We can find the minimum by taking the partial derivatives over (8) with respect to all the variables we must determine; 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 (8). (However, if the encoders were chosen ahead of time, this no longer holds.)

We can now reduce the sum to:

$$\sum_{(c_0, \ldots, c_n) \in T} \left[\sum_{j=0}^{n} E_{C_j}(c_j) + k - \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) \right]^2. \quad (9)$$

We have now eliminated a whole collection of variables and thus simplified our summation. As a result, by taking the partial derivatives over (9), 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 that leaves us

$$\sum_{(c_0, \ldots, c_n) \in T} \left[\sum_{j=0}^{n} E_{C_j}(c_j) - \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) \right]^2. \quad (10)$$

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 4.4. *The minimal solution to* (8) *will be for each random variable* C_k *in* T, *and for each* $c_k \in R(C_k)$,

$$E_{C_k}(c_k) = \frac{1}{\prod_{\substack{i=0 \\ i \neq k}}^{n} m_i} \left\{ \sum_{\substack{(c_0', \ldots, c_n') \in T \\ c_k' = c_k}} \ln P(C_0 = c_0' | \ldots, C_l = c_l', \ldots) \right\} - \frac{n\xi(T)}{(n+1) \prod_{i=0}^{n} m_i}. \quad (11)$$

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

$$\xi(T) = \sum_{(d_0, \ldots, d_n) \in T} \ln P(C_0 = d_0 | \ldots, C_l = d_l, \ldots).$$

We will now show how we derived the closed form eq. (11).

Notation 4.5

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

Differentiating (10) with respect to $E_{C_k}(d_k)$, we obtain

$$E_{C_k}(d_k) \prod_{\substack{i=0 \\ i \neq k}}^{n} m_i + \sum_{\substack{j=0 \\ j \neq k}}^{n} \sum_{d_j' \in R(C_j)} \left[E_{C_j}(d_j') \prod_{\substack{l=0 \\ l \neq k \\ l \neq j}}^{n} m_l \right]$$

$$- \sum_{\substack{(d_0', \ldots, d_n') \in T \\ d_k' = d_k}} \ln P(C_0 = d_0' | \ldots, C_l = d_l', \ldots) = 0. \tag{12}$$

We begin by summing over all possible d_ks. This results in

$$\begin{aligned} &\sum_{d_k \in R(C_k)} E_{C_k}(d_k) \prod_{\substack{i=0 \\ i \neq k}}^{n} m_i \\ &= \sum_{d_k \in R(C_k)} \sum_{\substack{(d_0', \ldots, d_n') \in T \\ d_k' = d_k}} [\ln P(C_0 = d_0' | \ldots, C_l = d_l', \ldots)] \\ &\quad - \sum_{d_k \in R(C_k)} \sum_{\substack{j=0 \\ k \neq k}}^{n} \sum_{d_j' \in R(C_j)} E_{C_j}(d_j') \prod_{\substack{l=0 \\ l \neq k \\ l \neq j}}^{n} m_l, \end{aligned}$$

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

$$\kappa_j = \frac{\xi(T)}{(n+1) \prod_{\substack{i=0 \\ i \neq j}}^{n} m_i}. \tag{13}$$

We rewrite eq. (12) as

$$\begin{aligned} E_{C_k}(d_k) \prod_{\substack{i=0 \\ i \neq k}}^{n} m_i &= \sum_{\substack{(d_0', \ldots, d_n') \in T \\ d_k' = d_k}} \ln P(C_0 = d_0' | \ldots, C_l = d_l', \ldots) \\ &\quad - \sum_{\substack{j=0 \\ j \neq k}}^{n} \kappa_j \prod_{\substack{l=0 \\ l \neq k \\ l \neq j}}^{n} m_l \\ &= \sum_{\substack{(d_0', \ldots, d_n') \in T \\ d_k' = d_k}} \ln P(C_0 = d_0' | \ldots, C_l = d_l', \ldots) \end{aligned}$$

$$
\begin{aligned}
&- \sum_{\substack{j=0 \\ j \neq k}}^{n} \frac{\xi(T)}{(n+1) \prod_{\substack{l=0 \\ l \neq j}}^{n} m_l} \prod_{\substack{l=0 \\ l \neq k \\ l \neq j}}^{n} m_l \\
&= \sum_{\substack{(d_0', \ldots, d_n') \in T \\ d_k' = d_k}} \ln P(C_0 = d_0' | \ldots, C_l = d_l', \ldots) \\
&\quad - \sum_{\substack{j=0 \\ j \neq k}}^{n} \frac{\xi(T)}{(n+1) m_k} \\
&= \sum_{\substack{(d_0', \ldots, d_n') \in T \\ d_k' = d_k}} \ln P(C_0 = d_0' | \ldots, C_l = d_l', \ldots) \\
&\quad - \frac{n \xi(T)}{(n+1) m_k}.
\end{aligned}
$$

We can assert (13) 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

$$
\sum_{j=0}^{n} E_{C_j}(c_j) = \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) \tag{14}
$$

for all $c_j \in R(C_j)$ for $j = 0, \ldots, n$. By taking the closed form solution we obtained in Theorem 4.4 and substituting it into (14), this results in the following condition on the probabilities:

$$
\begin{aligned}
&\sum_{j=0}^{n} m_j \left[\sum_{\substack{(c_0', \ldots, c_n') \in T \\ c_k' = c_k}} \ln P(C_0 = c_0' | C_1 = c_1', \ldots, C_n = c_n') \right] \\
&\quad - \ln P(C_0 = c_0 | \ldots, C_l = c_l, \ldots) \prod_{i=0}^{n} m_i = \frac{n^2 \xi(T)}{(n+1)},
\end{aligned} \tag{15}
$$

for all $(c_0, \ldots, c_n) \in T$.

From this, the next theorem follows:

THEOREM 4.6. *There is a perfect fit if and only if* (15) *holds.*

PROOF. This follows immediately from our minimization problem and Theorem 4.4. □

In our above formulation, we made two implicit assumptions. First, we allowed encoders to map to nonpositive values whereas Definition 4.2 allowed only positive ones. This can be easily resolved 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 since we need to be able to recover the original

values after being encoded. This can also be easily resolved by perturbing the encoder values by a small constant.

We now perform some experiments applying our approximations to randomly generated conditional tables. We generate two classes of tables: completely random and structured random. Both classes are also normalized to 1 in order to be valid conditional probability tables. We generate structured tables by arbitrarily ordering the possible assignments for each random variable. We then generate random values in the table with the following property: Let a_1 and a_2 be some instantiations for random variable A. For any two entries in the table which share exactly the same instantiations to all the random variables, except for A and A is either a_1 or a_2, then the value in the entry with a_1 should have a value less than that for a_2. Note, however, that after we normalize the table, this property may no longer hold. Structured tables attempt to capture some of the ordering relationships found in real world domains. For example, consider two random variables A and B denoting "Target Found" and "Radar Distance," respectively. The probability that a target is found given a radar reflection distance increases as the distance decreases.

Asides from our two different classes of tables, we varied several additional parameters including the size of the table, the number of random variables involved in a table as well as the skewness of values placed in the table. Increasing skewness implies that entries in the table will be closer to either 0 or 1. We note though, that in our experiments, skewness had little or no effect on the quality of our approximations.

There are several measures one can choose to determine how good a fit is. In our experiments, we measured the approximations against the following two metrics:

—Absolute worst fit:

$$\max_{c \in T} |P(c) - \bar{P}(c)| \tag{16}$$

—Relative average fit:

$$\frac{1}{|T|} \sum_{c \in T} \frac{|P(c) - \bar{P}(c)|}{P(c)}, \tag{17}$$

where T is the table, P is the actual probability, and $\bar{P}$ is our approximated probability. The relative fit will give us a percentage difference from the original table entry value.

For our experiments, we varied the table sizes from about 100,000 entries to 3,200,000 entries and varied the number of random variables from 3 to 12. For each combination of table size and number of random variables, we generated approximately 200 instances roughly three quarters of which were the structured tables.[13] In generating a table of size 100,000 with 6 random variables, we

[13] Note that by the definition of conditional probabilities, the average entry value of a given table is a function of the number of instantiations of the random variable being conditioned upon. For example, a $30 \times 100 \times 100$ table will have an average entry value of 1/30 instead of 1/300000 assuming that our random variable being conditioned on has 30 possible instantiations.

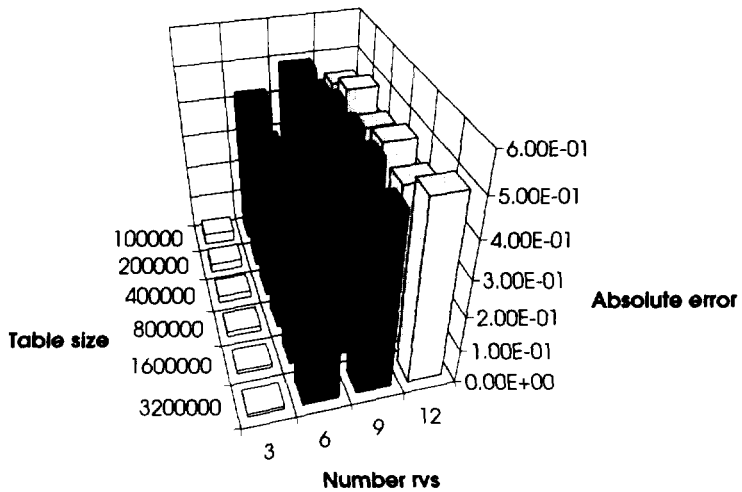

FIG. 4. Absolute worst fit for completely random tables.

selected one rv at a time and determined the number of instantiations for it. The number was chosen using a normal distribution with $\sqrt[6]{100{,}000}$ as its mean. We then determined the next random variable by choosing a number that was the 5th root of the remaining size-factor, etc.

Our approximation approach should fare worst when faced with completely random tables. By their very nature, we do not expect any patterns or structures to be found. In Figure 4, we can see the average absolute worst fit for each category of conditional probability tables. However, we must realize that most of the tables used in practice will by necessity be structured in some way.

In Figure 5, we can see that our approach performs better on the structured tables as opposed to the random tables. At worst, our approximations are off by 0.005. However, we must place this in perspective of the actual probability values which are a function of the number of instantiations for the random variable being conditioned on. Hence, with respect to individual probabilities, we measure the relative fits of our approximations according to $|P(c) - \bar{P}(c)|/P(c)$, where again, $P(c)$ is the actual probability entry and $\bar{P}(c)$ is the approximation. From Figure 6, the average relative error over an entire table is less than $\frac{1}{5}\%$, that is, the approximated value varies at most $\pm\frac{1}{5}\%$ from the actual value.

One final note about computing our LPFs: As it turns out, our method for finding the optimal LPFs is not restricted to Bayesian networks and their conditional probability tables. In fact, it can be applied to any table of values of any dimensions. Hence, we have provided a general scheme that can be used in many domains requiring data compression.

5. *Constraints Formulation*

As we mentioned earlier, our LPFs can be incorporated into an existing computational approach based on the results of transforming belief revision into linear constraint satisfaction [Santos, Jr. 1991; 1993; Charniak and Santos, Jr. 1992]. This approach could be applied to any Bayesian network regardless of network topology. Furthermore, this approach is capable of generating alternative

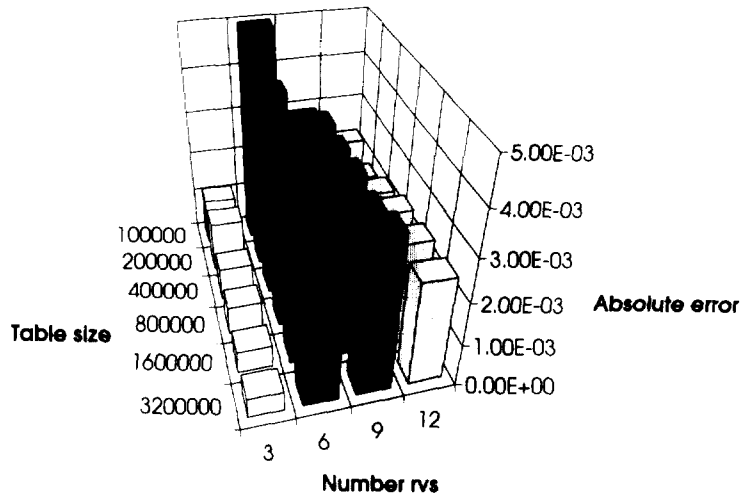

FIG. 5. Absolute worst fit for structured tables.

solutions and performs efficiently on moderately connected networks with moderate-sized tables. 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.[14]

The transformation involves mapping the random variable instantiations into some multi-dimensional Euclidean space $\Re^n$. A subset of $\Re^n$ will represent "valid" instantiations where valid includes things like being consistent to the given evidence e, each random variable has at most one instantiation, etc. In particular, we are interested in transforming it into a *polyhedral convex set.*[15] 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 random variables. 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 random variable instantiations. Let A be a random variable and a be a possible instantiation for A. If A is instantiated to a, that is, $A = a$, then we would like to set a real variable 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 random variable A.

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

$$\sum_{a \in R(A)} \mathsf{A}_a = 1, \tag{18}$$

[14] Precise details and theoretical proofs can be found in Santos, Jr. [1991].

[15] "Polyhedral" refers to the fact that the boundaries of the subset are composed of hyperplanes.

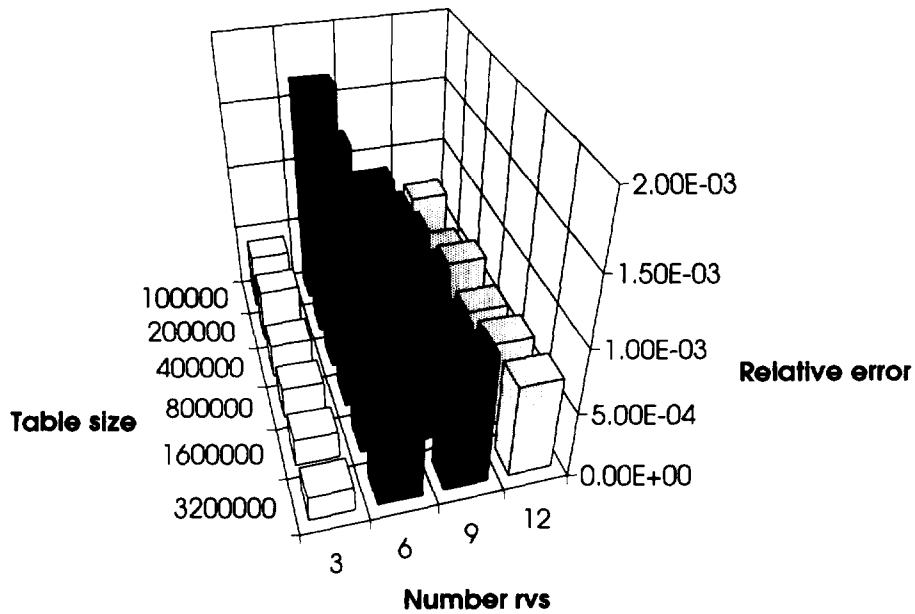

FIG. 6. Relative average fit for structured tables.

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,[16] construct a real variable $q[A_a|B_b, C_c]$ (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:

$$q \leq B_b, \tag{19}$$

$$q \leq C_c \tag{20}$$

and

$$\sum_{q \in Q(A_a)} q = A_a \tag{21}$$

where $Q(A_a)$ are all the qs whose associated conditional probability is of the form $P(A = a| \cdots)$.

Finally, given evidence e, if $A = a$ is an instantiation in e, then we also include the constraint $A_a = 1$.

Taking all these constraints together, we can prove that the resulting polyhedral convex set is indeed 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 qs, also called *conditional variables*, as follows: If q is true, this implies that the associated conditional probability must have been used to

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

compute the joint probability. Thus, for each q, we have the following term in the energy function:

$$-\log(P(A = a|B = b, C = c))q[\mathsf{A}_a|\mathsf{B}_b, \mathsf{C}_c]. \tag{22}$$

By taking exp − (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 a 0-1 *integer linear programming* problem [McMillan 1975; Schrijver 1986; Nemhauser et al. 1989]. We can solve this problem by combining the *Dual Simplex Method* with *Branch and Bound* [Santos, Jr. 1991; 1994]. Furthermore, by using a *Cutting Plane* approach [Santos, Jr. 1991], 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 suffers from table size explosions. For each entry in a conditional table, we must associate a unique real variable q. Overall, given N random variables $\{A_1, \ldots, A_n\}$, this formulation requires

—*Variables*:

$$\sum_{i=1}^{n} [|R(A_i)| + \delta(A_i)]$$

where $\delta(A_i)$ is the number of entries in the table associated with A_i.

—*Constraints*:

$$\sum_{i=1}^{n} [1 + |R(A_i)| + |\mathsf{cond}(A_i)|\delta(A_i)]$$

Our counts will clearly be dominated by $\delta(A_i)$ since each $\delta(A_i) = O(|R(A_i)| \times |R(A_{i1})| \times \cdots \times |R(A_{im})|)$ where $\mathsf{cond}(A_i) = \{A_{i1}, \ldots, A_{im}\}$.

6. *Computing with Linear Potential Functions*

We now present a new computational approach based on the previous section using LPFs.

Let $\mathscr{E}_{\mathscr{B}}$ be a collection of encoders for the random variables in V such that there is exactly one encoder for each random variable. To simplify our discussion, we first assume a single encoder for each random variable. Thus, the conditional tables which share the same random variable will also share the same encoder. However, we will also present appropriate modifications for multiple encoders.

Next, let T be some conditional table associated with node C_0 in V. We denote

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

to be all the random variables that C_0 is conditioned on including C_0.

Let T be a conditional table in $\mathscr{B}$ and assume

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

Definition 6.1. Let $S_{\mathscr{E}_{\mathscr{B}},T}$ be a function from $\Re^{n+1}$ to $\Re$ of the form

$$S_{\mathscr{E}_{\mathscr{B}},T}(E_{C_0}(c_0), E_{C_1}(c_1), \ldots, E_{C_n}(c_n)) \tag{23}$$

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

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

We call $S_{\mathscr{E}_{\mathscr{B}},T}$ an *approximator* for T. Furthermore, we say $S_{\mathscr{E}_{\mathscr{B}},T}$ is *perfect* if

$$S_{\mathscr{E}_{\mathscr{B}},T}(E_{C_0}(c_0), E_{C_1}(c_1), \ldots, E_{C_n}(c_n)) = P(C_0 = c_0 | C_1 = c_1, \ldots, C_n = c_n)$$

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

Let $\mathscr{S}_{\mathscr{E}_{\mathscr{B}}}$ be a set of approximators such that each conditional table T in $\mathscr{B}$ is associated with exactly one approximation function.

Given $\mathscr{S}_{\mathscr{E}_{\mathscr{B}}}$, since all our conditional probability tables are uniquely identified by the random variables involved, for any $c_0 \in R(C_0), \ldots, c_n \in R(C_n)$,

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

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

We now redefine our notion of a Bayesian network.

Definition 6.2. Given a Bayesian network $\mathscr{B} = (V, P)$, let $\mathscr{E}_{\mathscr{B}}$ be a collection of encoders and $\mathscr{S}_{\mathscr{E}_{\mathscr{B}}}$ be a collection of approximators. We define a *LPF-Bayesian network* to be a 3-tuple $\bar{\mathscr{B}} = (V, \mathscr{E}_{\mathscr{B}}, \mathscr{S}_{\mathscr{E}_{\mathscr{B}}})$.

Definition 6.3. Given a complete instantiation set w for $\mathscr{B}$, *we define the LPF-probability*, P_s for $\bar{\mathscr{B}}$ as

$$P_s(w) = \prod_{A \in \text{span}(w)} S_{\mathscr{E}_{\mathscr{B}}}(w(A), w(\text{cond}(A))). \tag{24}$$

PROPOSITION 6.4. *Given a complete instantiation set w, if all the approximators in* $\mathscr{S}_{\mathscr{E}_{\mathscr{B}}}$ *are perfect, then* $p(w) = P_s(w)$.

Our goal is to substitute our LPF into our computations for belief revision. We now proceed to show how we can transform LPF-Bayesian networks into linear constraint satisfaction problems.

Definition 6.5. Given an approximator $S \in \mathscr{S}_{\mathscr{E}_{\mathscr{B}}}$, we say that S is a *linear potential function* (*LPF*) if and only if

$$\begin{aligned} &S_{\mathscr{E}_{\mathscr{B}}}(E_A(a), E_{C_1}(c_1), \ldots, E_{C_n}(c_n)) \\ &\quad = \exp(k_0 E_A(a) + k_1 E_{C_1}(c_1) + \cdots + k_n E_{C_n}(c_n) + k). \end{aligned}$$

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

This is the approximator we computed in (5).

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 that satisfies all the value restrictions as well as constraints.

We begin our construction as follows: Let $\bar{\mathcal{B}} = (V, \mathcal{E}_{\mathcal{B}}, \mathcal{S}_{\ell_R})$ be a LPF-Bayesian network and $\mathcal{S}_{\ell_R}$ be a linear potential space. For each random variable 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 $a \in R(A)\}$. These will be the only variables in our linear constraints.

In order to properly constrain our overall space of possible solutions, for each random variable A, we must have the following two constraints:

$$x_A \geq \min_{a \in R(A)} E_A(a) \tag{25}$$

$$x_A \leq \max_{a \in R(A)} E_A(a). \tag{26}$$

We complete our transformation by defining an appropriate objective function: For each function S in $\mathcal{S}_{\ell_R}$,

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

must be part of our objective function.

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: Since our encoders are one-to-one and onto, then the inverse (or, called *decoder*) exists and we denote them by E_A^{-1}. $w[s](A) = E_A^{-1}(x_A)$.

Conversely, given a complete instantiation-set w, we can construct a permissible assignment $s[w]$ as follows: For each random variable A in V, $s[w](x_A) = E_A(w(A))$.

THEOREM 6.6. *w is a complete instantiation-set for $\mathcal{B}$ if and only $s[w]$ is a permissible solution for the induced constraint system.*

PROOF. Follows from the construction above. □

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

THEOREM 6.7

$$P_s(w) = \exp(-\Theta(s[w])),$$

where $\Theta(s[w])$ is a summation of the objective terms in Shanahan [1989].

PROOF. Follows from Theorem 6.6 and our construction above. □

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

Finally, we must also incorporate the notion of evidence. Evidence, we recall, is the requirement that a random variable be instantiated with a certain value. For the case where a random variable 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 as we did earlier in this section.

Now, given N random variables $\{A_1, \ldots, A_n\}$, this new formulation requires

—*Variables*:

$$\sum_{i=1}^{n} |R(A_i)|.$$

—*Constraints*:

$$2 \sum_{i=1}^{n} |R(A_i)|.$$

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 Santos, Jr. [1994] to guarantee that we generate permissible solutions.

Notation 6.8. Let x be a real variable and $\{v_1, v_2, \ldots, v_n\}$ be its permissible values such that $v_i < v_{i+1}$. We define the following functions:

$$\lfloor v \rfloor_x = \max_{v_i \leq v} v_i$$

$$\lceil v \rceil_x = \min_{v_i \geq v} v_i.$$

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 6.9. Given a constraint system L, find its optimal permissible solution.

1. (*Initialization*) Set *CurrentBest* := ϕ and *ActiveNodes* := $\{(I, 0)\}$.
2. If *ActiveNodes* = ϕ, then go to step 15. Otherwise, let lp be some linear program in *ActiveNodes*.
3. *ActiveNodes* := *ActiveNodes* − {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 $\Theta_L(s^{\text{opt}}) > \Theta_L(\textit{CurrentBest})$, then go to Step 2.
7. (*Branch*) Choose some variable $x \in$ lp whose value in s^{opt} is nonpermissible.
8. Set $I_1 := I \cup \{x \leq \lfloor s^{\text{opt}}(x) \rfloor_x\}$ and $I_2 := I \cup \{x \geq \lceil s^{\text{opt}}(x) \rceil_x\}$
9. Create two new linear programs:

$$\text{lp}_1 := (I_1, \Theta_L(s^{\text{opt}})) \text{ and } \text{lp}_2 := (I_2, \Theta_L(s^{\text{opt}})).$$

10. *ActiveNodes* := *ActiveNodes* $\cup$ $\{\text{lp}_1, \text{lp}_2\}$.
11. Go to step 2.
12. (*Permissible solution*)
    If *CurrentBest* = ϕ or $\Theta_L(s^{\text{opt}}) < \Theta_L(\textit{CurrentBest})$, then
    CurrentBest := s^{opt}.
13. (*Pruning*) Remove from ActiveNodes all linear programs whose lower bounds are greater than $\Theta_L(\textit{CurrentBest})$.
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 $O(|R(A)| \times |R(C_1)| \times \cdots \times |R(C_n)|)$ by compressing it to $O(|R(A)| + |R(C_1)| + \cdots + |R(C_n)|)$. Our goal is to find such an optimal compression by manipulating encoders and approximator functions.

In terms of how well LPFs perform, let's assume that some set $\{A_1, A_2, \ldots, A_n\}$ of random variables in our given Bayesian network are now replaced by their respective LPFs $\{S_1, S_2, \ldots, S_n\}$. We now consider the overall expected error of computing a joint probability when using our LPFs. Again, since multiplying the conditional probabilities is equivalent to taking the sum of the logarithms, the expected error is as follows:

$$\sum_{i=1}^{n} \sum_{w \in T_i} z_i(\ln P_i(w) - \ln \hat{P}_i(w)), \tag{28}$$

where $z_i = 1/|T_i|$, $|T_i|$ is the number of entries in the table associated with A_i, P_i is the conditional probability for A_i, and $\hat{P}_i(w)$ is the approximation via S_i.

THEOREM 6.10

$$\sum_{i=1}^{n} \sum_{w \in T_i} z_i(ln\ P_i(w) - ln\ \hat{P}_i(w)) = 0.$$

PROOF. This follows from substituting in the optimal encoder values into (28). □

One final note for this section is incorporating multiple encoders for a single random variable. We shall show this only increases 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:

$$x_A^1 - E_A^1(a) \geq -M(1 - y_{A=a})$$

$$x_A^1 - E_A^1(a) \leq M(1 - y_{A=a})$$

$$x_A^2 - E_A^2(a) \geq -M(1 - y_{A=a})$$

$$x_A^2 - E_A^2(a) \leq M(1 - y_{A=a}),$$

where $y_{A=a}$ is a 0-1 detector variable and M is some arbitrarily large positive constant. This can be generalized to any number of encoders. Again, using this technique for multiple encoders frees us to compute LPFs for each table independently. Appendix A provides an example transformation. Finally, we find that our problem maps straight back to pure 0-1 integer programming without needing to worry about permissibility since permissibility is automatically guaranteed by enforcing 0-1 values upon the detectors.

7. *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 probabilistic reasoning methods. Up till now, each of these methods must access virtually all of the probabilities to achieve the desired computations. 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 be measured as $O(|R(A_1)| \times |R(A_2)| \times \cdots \times |R(A_n)|)$ where A_i's are random variables.

Our new approach compresses the tables into approximation functions called *linear potential functions* and reformulates them into linear constraint satisfaction. We have provided a scheme that does not require explicit storage of this data. The compression is achieved by using these functions as approximations to the conditional tables. Unfortunately, although conditional tables are look-up functions, the parameters used, namely random variable instantiations, may not be numerical values as one would like them to be in a real function. For example, random variables 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 random variable 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 LPF. 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 random variable instantiations. This now provides us with an immense amount of flexibility for finding an excellent approximating function. Most importantly, the solution for finding the best mapping 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.

We also demonstrated that our LPFs can be fully merged into an existing computational method.

Putting this altogether, we have effectively reduced an originally

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

to a

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

Our examples demonstrate the savings that can be achieved. However, we do note that for small tables, say less than one hundred entries, there is a trade-off in using an approximation function. Currently, we have the computational resources to solve small problems using the tables explicitly [Santos, Jr. 1991]. Naturally, there will be a few tables that are small in a given Bayesian network. In these cases, we may not use a linear potential function. Instead, we would utilize a mix of approximation functions and explicit tables. As it turns out, such a mix can also be naturally formulated using integer linear programming.

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.[17] Thus, this helps to provide further support for the expected success of our approach.

Future work on approximation functions will include the possibility of using multiple LPFs on different portions of a single conditional table. In essence, we partition up the table and create a spline of LPFs over it. This way, if there are some clear and distinct patterns within a single table, we can take advantage of this and achieve an even better approximation. By carefully restricting the types of partitionings allowed, it should be possible to preserve the linear compressions gained from using a single LPF.

Appendix A. Mobile Target Location Problem

The Mobile Target Localization problem (abbreviated MTL) [Kirman et al. 1991; 1993] 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.

[17] See, for example, Plotkin et al. [1991], Karmarkar and Karp [1982], Klein et al. [1991], and Raghavan [1988].

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 random variables in order to represent our world and its uncertainties:

—L_{R_i} represents the location of the robot at time i ranging over L.
—L_{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.
—*Dist* represents the distance between the robot and target allowing us to use proximity as a desirable outcome.

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.

By making various instantiations to the above random variables, 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 random variables.

On the other hand, we again look at the chain rule as follows for the random variables at time 1:

$$\begin{aligned} &P(O_{R_1}, O_{T_1}, A_{R_1}, L_{R_1}, L_{T_1}) \\ &\quad = P(O_{R_1}|O_{T_1}, A_{R_1}, L_{R_1}, L_{T_1})P(O_{T_1}|A_{R_1}, L_{R_1}, L_{T_1}) \\ &\quad * P(A_{R_1}|L_{R_1}, L_{T_1})P(L_{R_1}|L_{T_1})P(L_{T_1}). \end{aligned} \tag{29}$$

Using Bayesian networks, the MTL problem is modeled in Figure A.1. We can now rewrite eq. (1) as follows for the random variables involved in time step 1:[18]

[18] We can isolate the random variables in time step 1 from the remaining random variables because of the following observation on the conditional independence relationships: Let U be the remaining random variables in the network. We know that

$$\begin{aligned} &P(O_{R_1}, O_{T_1}, A_{R_1}, L_{R_1}, L_{T_1}) \\ &\quad = \sum_U P(O_{R_1}, O_{T_1}, A_{R_1}, L_{R_1}, L_{T_1}, U). \end{aligned}$$

Furthermore, according to our independence assumptions

$$\begin{aligned} &P(O_{R_1}, O_{T_1}, A_{R_1}, L_{R_1}, L_{T_1}, U) \\ &\quad = P(U|O_{R_1}, O_{T_1}, A_{R_1}, L_{R_1}, L_{T_1})P(O_{R_1}, O_{T_1}, A_{R_1}, L_{R_1}, L_{T_1}). \end{aligned}$$

$$P(O_{R_1}, O_{T_1}, A_{R_1}, L_{R_1}, L_{T_1})$$
$$= P(O_{R_1}|L_{R_1})P(O_{T_1}|L_{R_1}, L_{T_1})P(A_{R_1}|L_{R_1})P(L_{R_1})P(L_{T_1}). \quad (30)$$

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 four time slices as seen in Figure A.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

$$P(A_{R_1} = a^*|O_{R_1}, O_{T_1}) = \max_a P(A_{R_1} = a|O_{R_1}, O_{T_1}). \quad (31)$$

This type of computation performed with Bayesian networks is classified as *belief updating* [Pearl 1988]. In general, belief updating involves updating our beliefs about the different possible instantiations of a random variable given certain instantiations of other random variables as evidence/observations.[19] However, as we noted in Section 2, belief updating can be performed with belief revision [Kirman et al. 1993; Santos, Jr. and Shimony 1994].

We now present a hypothetical example transforming a Bayesian network into our constraints formulation. For this problem, we assume 100 possible map locations and 100 possible sonar readings.

From Figure A.1, we have the following random variables: S_{R_i}, O_{R_i}, O_{T_i}, S_{T_i} for $i = 1, 2, 3, 4$, A_{R_i} for $i = 1, 2, 3$, and *Dist*. Let $m_{A_{R_i}} = |R(A_{R_i})|$, etc. Assume we have one LPF for each conditional table. Furthermore, assume that each table has its own separate set of encoders.

Given the conditional table associated node L_{R_2}, we denote the associated encoders by, $E_{L_{R_1},L_{R_2}}$, $E_{A_{R_1},L_{R_2}}$, and $E_{L_{R_2},L_{R_2}}$. The second index of the encoders represent the associated node. This will be similarly handled for all nodes. Let $F_{L_{R_2}}$ be the LPF associated with node L_{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 random variable. For each $b \in R(B)$,

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 random variables in U.

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

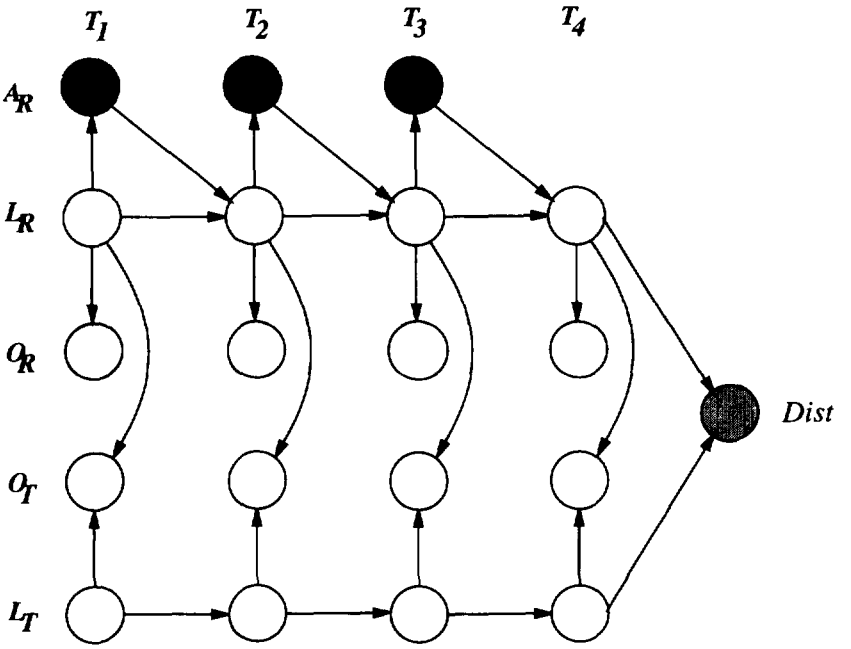

FIG. A1. Bayesian network for MTL problem with 4 time slices.

construct a 0-1 detector $y_{B=b}$. For each encoder E_{B,A_t}, construct the following two constraints:

$$x_{B,A_t} - E_{B,A_t}(b) \geq -M(1 - y_{B=b})$$

$$x_{B,A_t} - E_{B,A_t}(b) \leq M(1 - y_{B=b}).$$

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

$$x_{O_{R_1},O_{R_1}} = E_{O_{R_1},O_{R_1}}(s_1)$$

$$x_{O_{T_1},O_{T_1}} = E_{O_{T_1},O_{T_1}}(s_2).$$

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

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

where $F'(A)$ is simply the original function F_A with the encoder elements replaced by the appropriate real variables representing the instantiations.

Our transformation will involve 46 representation variables, 1715 consistency variables, 2 evidence constraints and 6700 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 random variables state space. Compare this against the total number of probabilities we would need to maintain in all the conditional tables, which is 5,221,700!

We now provide the resulting linear constraint system for the MTL problem. Assume the following:

—$R(A_{R_i}) = \{a_1, a_2, \ldots, a_5\}$ for $i = 1, 2, 3$.
—$R(L_{R_i}) = R(L_{T_i}) = \{l_1, \ldots, l_{100}\}$ for $i = 1, 2, 3, 4$.
—$R(O_{R_i}) = R(O_{T_i}) = \{s_1, \ldots, s_{100}\}$ for $i = 1, 2, 3, 4$.
—$R(Dist) = \{d_1, d_2, \ldots, d_{100}\}$.

We have the two evidence constraints:

$$x_{O_{R_1},O_{R_1}} = E_{O_{R_1},O_{R_1}}(s_1)$$

$$x_{O_{T_1},O_{T_1}} = E_{O_{T_1},O_{T_1}}(s_2).$$

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

$$x_{A_{R_i},L_{R_{i+1}}} - E_{A_{R_i},L_{R_{i+1}}}(a_j) \geq -M(1 - y_{A_{R_i}=a_j})$$

$$x_{A_{R_i},L_{R_{i+1}}} - E_{A_{R_i},L_{R_{i+1}}}(a_j) \leq M(1 - y_{A_{R_i}=a_j})$$

$$x_{A_{R_i},A_{R_i}} - E_{A_{R_i},A_{R_i}}(a_j) \geq -M(1 - y_{A_{R_i}=a_j})$$

$$x_{A_{R_i},A_{R_i}} - E_{A_{R_i},A_{R_i}}(a_j) \leq M(1 - y_{A_{R_i}=a_j}).$$

For each $i = 1, \ldots, 3$, for each L_{R_i}, for each $j = 1, \ldots, 100$,

$$x_{L_{R_i},L_{R_{i+1}}} - E_{L_{R_i},L_{R_{i+1}}}(l_j) \geq -M(1 - y_{L_{R_i}=l_j})$$

$$x_{L_{R_i},L_{R_{i+1}}} - E_{L_{R_i},L_{R_{i+1}}}(l_j) \leq M(1 - y_{L_{R_i}=l_j})$$

$$x_{L_{R_i},L_{R_i}} - E_{L_{R_i},L_{R_i}}(l_j) \geq -M(1 - y_{L_{R_i}=l_j})$$

$$x_{L_{R_i},L_{R_i}} - E_{L_{R_i},L_{R_i}}(l_j) \leq M(1 - y_{L_{R_i}=l_j})$$

$$x_{L_{R_i},A_{R_i}} - E_{L_{R_i},A_{R_i}}(l_j) \geq -M(1 - y_{L_{R_i}=l_j})$$

$$x_{L_{R_i},A_{R_i}} - E_{L_{R_i},A_{R_i}}(l_j) \leq M(1 - y_{L_{R_i}=l_j})$$

$$x_{L_{R_i},O_{T_i}} - E_{L_{R_i},O_{T_i}}(l_j) \geq -M(1 - y_{L_{R_i}=l_j})$$

$$x_{L_{R_i},O_{T_i}} - E_{L_{R_i},O_{T_i}}(l_j) \leq M(1 - y_{L_{R_i}=l_j}).$$

For each $j = 1, \ldots, 100$,

$$x_{L_{R_4},L_{R_4}} - E_{L_{R_4},L_{R_4}}(l_j) \geq -M(1 - y_{L_{R_4}=l_j})$$

$$x_{L_{R_4},L_{R_4}} - E_{L_{R_4},L_{R_4}}(l_j) \leq M(1 - y_{L_{R_4}=l_j})$$

$$x_{L_{R_4},O_{R_4}} - E_{L_{R_4},O_{R_4}}(l_j) \geq -M(1 - y_{L_{R_4}=l_j})$$

$$x_{L_{R_4},O_{R_4}} - E_{L_{R_4},O_{R_4}}(l_j) \leq M(1 - y_{L_{R_4}=l_j})$$

$$x_{L_{R_4},O_{T_4}} - E_{L_{R_4},O_{T_4}}(l_j) \geq -M(1 - y_{L_{R_4}=l_j})$$

$$x_{L_{R_4},O_{T_4}} - E_{L_{R_4},O_{T_4}}(l_j) \le M(1 - y_{L_{R_4}=l_j})$$
$$x_{L_{R_4},Dist} - E_{L_{R_4},Dist}(l_j) \ge -M(1 - y_{L_{R_4}=l_j})$$
$$x_{L_{R_4},Dist} - E_{L_{R_4},Dist}(l_j) \le M(1 - y_{L_{R_4}=l_j}).$$

For each $i = 1, \ldots, 3$, for each L_{T_i}, for each $j = 1, \ldots, 100$,

$$x_{L_{T_i},L_{T_{i+1}}} - E_{L_{T_i},L_{T_{i+1}}}(l_j) \ge -M(1 - y_{L_{T_i}=l_j})$$
$$x_{L_{T_i},L_{T_{i+1}}} - E_{L_{T_i},L_{T_{i+1}}}(l_j) \le M(1 - y_{L_{T_i}=l_j})$$
$$x_{L_{T_i},L_{T_i}} - E_{L_{T_i},L_{T_i}}(l_j) \ge -M(1 - y_{L_{T_i}=l_j})$$
$$x_{L_{T_i},L_{T_i}} - E_{L_{T_i},L_{T_i}}(l_j) \le M(1 - y_{L_{T_i}=l_j})$$
$$x_{L_{T_i},O_{T_i}} - E_{L_{T_i},O_{T_i}}(l_j) \ge -M(1 - y_{L_{T_i}=l_j})$$
$$x_{L_{T_i},O_{T_i}} - E_{L_{T_i},O_{T_i}}(l_j) \le M(1 - y_{L_{T_i}=l_j}).$$

For each $j = 1, \ldots, 100$,

$$x_{L_{T_4},L_{T_4}} - E_{L_{T_4},L_{T_4}}(l_j) \ge -M(1 - y_{L_{T_4}=l_j})$$
$$x_{L_{T_4},L_{T_4}} - E_{L_{T_4},L_{T_4}}(l_j) \le M(1 - y_{L_{T_4}=l_j})$$
$$x_{L_{T_4},O_{T_4}} - E_{L_{T_4},O_{T_4}}(l_j) \ge -M(1 - y_{L_{T_4}=l_j})$$
$$x_{L_{T_4},O_{T_4}} - E_{L_{T_4},O_{T_4}}(l_j) \le M(1 - y_{L_{T_4}=l_j})$$
$$x_{L_{T_4},Dist} - E_{L_{T_4},Dist}(l_j) \ge -M(1 - y_{L_{T_4}=l_j})$$
$$x_{L_{T_4},Dist} - E_{L_{T_4},Dist}(l_j) \le M(1 - y_{L_{T_4}=l_j}).$$

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

$$x_{O_{R_i},O_{R_i}} - E_{O_{R_i},O_{R_i}}(s_j) \ge -M(1 - y_{O_{R_i}=s_j})$$
$$x_{O_{R_i},O_{R_i}} - E_{O_{R_i},O_{R_i}}(s_j) \le M(1 - y_{O_{R_i}=s_j}).$$

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

$$x_{O_{T_i},O_{T_i}} - E_{O_{T_i},O_{T_i}}(s_j) \ge -M(1 - y_{O_{T_i}=s_j})$$
$$x_{O_{T_i},O_{T_i}} - E_{O_{T_i},O_{T_i}}(s_j) \le M(1 - y_{O_{T_i}=s_j}).$$

For each $j = 1, \ldots, 100$,

$$x_{Dist,Dist} - E_{Dist,Dist}(d_j) \ge -M(1 - y_{Dist=d_j})$$
$$x_{Dist,Dist} - E_{Dist,Dist}(d_j) \le M(1 - y_{Dist=d_j}).$$

Finally, the objective function is

$$
\begin{aligned}
&-\sum_{i=1}^{3}\left[x_{A_{R_i},A_{R_i}}+x_{L_{R_i},A_{R_i}}\right]\\
&-\sum_{i=2}^{4}\left[x_{L_{R_i},L_{R_i}}+x_{A_{R_{i-1}},L_{R_i}}+x_{L_{R_{i-1}},L_{R_i}}\right]\\
&-x_{L_{R_1},L_{R_1}}\\
&-\sum_{i=1}^{4}\left[x_{O_{R_i},O_{R_i}}+x_{L_{R_i},O_{R_i}}\right]\\
&-\sum_{i=1}^{4}\left[x_{O_{T_i},O_{T_i}}+x_{L_{T_i},O_{T_i}}+x_{L_{R_i},O_{T_i}}\right]\\
&-\sum_{i=2}^{4}\left[x_{L_{T_i},L_{T_i}}+x_{L_{T_{i-1}},L_{T_i}}\right]\\
&-x_{L_{T_1},L_{T_1}}\\
&-\left[x_{Dist,Dist}+x_{L_{R_4},Dist}+x_{L_{T_4},Dist}\right]\\
&+K,
\end{aligned}
$$

where K is the overall translation constant, which is required to ensure that our encodings are nonnegative values.

ACKNOWLEDGMENTS. Special thanks to the anonymous reviewers who helped improve this paper tremendously.

REFERENCES

CHARNIAK, E., AND GOLDMAN, R. 1988. A logic for semantic interpretation. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Menlo Park, Calif., pp. 87–94.

CHARNIAK, E., AND SANTOS, JR., E. 1992. Dynamic map calculations for abduction. In *Proceedings of the AAAI Conference*. AAAI Press, Menlo Park, Calif., pp. 552–557.

CHARNIAK, E., AND SHIMONY, S. G. 1990. Probabilistic semantics for cost based abduction. In *Proceedings of the AAAI Conference*. AAAI Press, Menlo Park, Calif., pp. 106–111.

COOPER, G. F. 1987. Probabilistic inference using belief networks is NP-hard. Technical Report KSL-87-27, Medical Computer Science Group, Stanford University, Stanford, Calif.

DAGUM, P., AND LUBY, M. 1993. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artif. Int. 60*, 1, 141–153.

DAVIS, R. 1984. Diagnostic reasoning based on structure and behavior. *Artif. Int. 24*, 347–410.

DUDA, R. O., HART, P. E., AND NILSSON, N. J. 1976. Subjective bayesian methods for rule-based inference systems. In *Proceedings of the National Computer Conference*.

GOLDMAN, R. P. 1990. *A Probabilistic Approach to Language Understanding*. Ph.D. dissertation. Department of Computer Science, Brown Univ.

GOLDMAN, R. P., AND CHARNIAK, E. 1991. Probabilistic text understanding. In *Proceedings of the 3rd International Workshop on AI and Statistics* (Fort Lauderdale, Fla.).

HOBBS, J. R., STICKEL, M., MARTIN, P., AND EDWARDS, D. 1988. Interpretation as abduction. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Menlo Park, Calif., pp. 95–103.

KARMARKAR, N., AND KARP, R. M. 1986. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, pp. 206–213.

KIRMAN, J., BASYE, K., AND DEAN, T. 1991. Sensor abstractions for control of navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, New York, pp. 2812–2817.

KIRMAN, J., NICHOLSON, A., LEJTER, M., SANTOS, JR., E., AND DEAN, T. 1993. Using goals to find plans with high expected utility. In *Proceedings of the 2nd European Workshop on Planning*.

KLEIN, P., PLOTKIN, S. A., STEIN, C., AND TARDOS, E. 1991. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. Tech. Rep. 961. Schools of Operations Research and Industrial Engineering, Cornell Univ.

MCMILLAN, C. 1975. *Mathematical Programming*. John-Wiley & Sons, Inc., New York.

NEMHAUSER, G. L., RINNOOY KAN, A. H. G., AND TODD, M. J., EDS. 1989. *Optimization: Handbooks in Operations Research and Management Science*. vol. 1. North-Holland, Amsterdam, The Netherlands.

PEARL, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan-Kaufmann, San Mateo, Calif.

PENG, Y., AND REGGIA, J. A. 1986. Plausibility of diagnostic hypothese: The nature of simplicity. In *Proceedings of the AAAI Conference*. AAAI Press, Menlo Park, Calif., pp. 140–147.

PENG, Y., AND REGGIA, J. A. 1990. *Abductive Inference Models for Diagnostic Problem-Solving*. Springer-Verlag, New York.

PLOTKIN, S. A., SHMOYS, D. B., AND TARDOS, E. 1991. Fast approximation algorithms for fractional packing and covering problems. In *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science*. pp. 495–504.

RAGHAVAN, P. 1988. Probabilistic construction of deterministic algorithms: Approximating packing intereger programs. *J. Comput. Syst. Sci. 37*, 130–143.

SANTOS, JR., E. 1991. On the generation of alternative explanations with implications for belief revision. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. Morgan-Kaufmann, San Francisco, Calif., pp. 337–347.

SANTOS, JR., E. 1993. A fast hill-climbing approach without an energy function for finding mpe. In *Proceedings of the 5th IEEE International Conference on Tools with Artificial Intelligence*.

SANTOS, JR., E. 1994. A linear constraint satisfaction approach to cost-based abduction. *Artif. Int. 65*, 1, 1–28.

SANTOS, JR., E., AND SHIMONY, S. E. 1994. Belief updating by enumerating high-probability independence-based assignments. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. Morgan-Kaufmann, San Francisco, Calif., pp. 506–513.

SCHRIJVER, A. 1986. *Theory of Linear and Integer Programming*. Wiley, New York.

SHANAHAN, M. 1989. Prediction is deduction but explanation is abduction. In *Proceedings of the International Joint Conference on Artificial Intelligence*. Morgan-Kaufmann, San Mateo, Calif., pp. 1055–1060.

SHIMONY, S. E. 1993. The role of relevance in explanation, I: Irrelevance as statistical independence. *Int. J. Approx. Reas.* (June).

SHIMONY, S. E., AND CHARNIAK, E. 1990. A new algorithm for finding map assignments to belief networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. Morgan-Kaufmann, San Francisco, Calif.

SHWE, M., MIDDLETON, B., HECKERMAN, D., HENRION, M., HORVITZ, E., AND LEHMANN, H. 1991. Probabilistic diagnosis using a reformulation of the internist-1/qmr knowledge base: I. the probabilistic model and inference algorithms. *Meth. Inf. Med. 30*, 241–255.

SRINIVAS, S. 1993. A generalization of the noisy-or model. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. Morgan-Kaufmann, San Francisco, Calif. pp. 208–215.

SY, B. K. 1992. Reasoning mpe to multiply connected belief networks using message passing. In *Proceedings of the 10th National Conference on Artificial Intelligence*. AAAI Press, Menlo Park, Calif., pp. 570–576.

RECEIVED SEPTEMBER 1993; REVISED SEPTEMBER 1995; ACCEPTED NOVEMBER 1995