Deterministic Approximation of Marginal Probabilities in Bayes Nets

Eugene Santos, Jr., *Member, IEEE*, and Solomon Eyal Shimony

***Abstract*—Computation of marginal probabilities in Bayes nets is central to numerous reasoning and automatic decision making systems. This paper presents a deterministic approximation scheme for this hard problem that supplies provably correct bounds by aggregating probability mass in independence-based (IB) assignments. The scheme presented refines recent work in belief updating for Bayes networks: attempts to approximate posterior probabilities by finding a small number of the highest probability complete (or perhaps evidentially supported) assignments. Under certain assumptions, the probability mass in the union of these assignments is sufficient to obtain a good approximation. Such methods are especially useful for highly connected networks, where the maximum clique size or the cutset size make many existing algorithms intractable. Since IB assignments contain fewer assigned variables, the probability mass in each assignment is greater than in the respective complete assignment. Thus, fewer IB assignments are sufficient, and a good approximation can be obtained more efficiently. Two classes of algorithms for finding the high probability IB assignments are suggested: best-first heuristic search and a special-purpose integer linear program (ILP). Since IB assignments may be overlapping events in probability space, accumulating the mass in a set of assignments may be hard. In the ILP variant, it is easy to avoid the problem by adding equations that prohibit overlap. In the best-first search algorithm, other schemes are necessary, but experimental results suggest that using inclusion–exclusion (potentially exponential-time in the worst case) in the overlap cases is not too expensive for most problem instances.**

***Index Terms*—Anytime algorithms, approximate belief updating, approximating marginal probabilities, Bayesian belief networks, decision making systems, probabilistic reasoning.**

I. Introduction

PROBABILISTIC reasoning is a paradigm used in numerous reasoning and automatic decision making systems to handle uncertain world knowledge, uncertainty of cause and effect, and other sources of uncertainty [31]. Since probability theory requires the assignment of a probability measure to a prohibitively large space, various structuring methods are used to take advantage of the many independencies in the world, in hopes of curtailing the combinatorial explosion in required information. Of particular interest are influence diagrams [31], [39] which compactly represent large decision trees, and Bayesian (belief) networks, which are a necessary element in influence diagrams. In this paper, we focus on the Bayes network component of influence diagrams.[1]

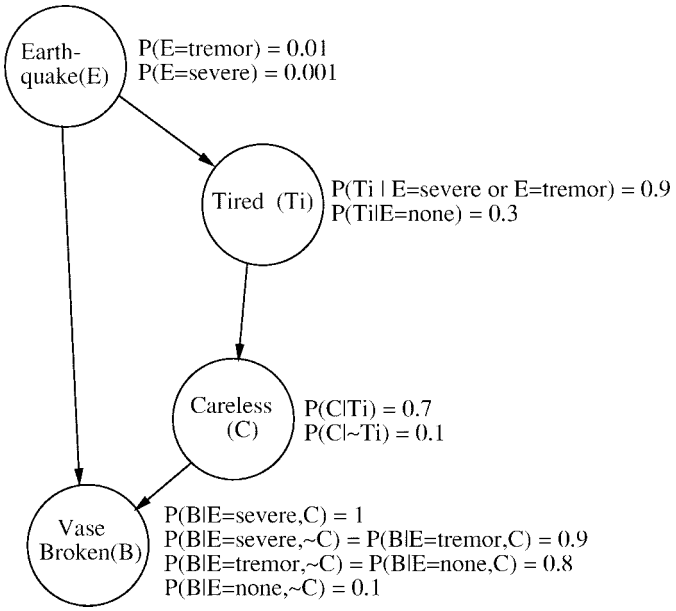

Fig. 1. Common sense reasoning example.

Bayes networks are directed acyclic graphs that describe (probabilistic) cause and effect relationships in the world [30], [31]. Each effect is a node (also called a variable), and a distribution of the state of the node given all possible states of its direct predecessor nodes (its parents, which are the immediate causes) is provided. See Fig. 1 for an example of Bayes net for common sense reasoning, and Fig. 2 for a Bayes net (topology only) for medical diagnosis.

These local distributions, together with the independence assumptions inherent to Bayes nets, uniquely determine the joint distribution over the entire sample space

$$P(V) = \prod_{v \in V} P(v | \text{ parents}(v)).$$

For most applications, this is a marked improvement over providing the joint probability of an unstructured domain. Bayes networks and influence diagrams have been used as the reasoning and decision making core in systems for computer vision [2], natural language understanding [6], planning

Manuscript received March 18, 1995; revised October 22, 1997. This work was supported in part by AFOSR under Project 940006, and by the Paul Ivanier Center for Robotics Research and Production Management, Ben-Gurion University.

E. Santos, Jr. is with the Computer Science and Engineering Department, University of Connecticut, Storrs, CT 06269-3155 USA (e-mail: eugene@eng2.uconn.edu).

S. E. Shimony is with the Department of Mathematics and Computer Science, Ben-Gurion University of the Negev, 84105 Beer-Sheva, Israel (e-mail: shimony@cs.bgu.ac.il).

Publisher Item Identifier S 1083-4427(98)04359-8.

[1] Alternately, we can look at Bayes networks as a special case: as an influence diagram that contains no value or decision nodes, and thus consisting only of chance nodes.

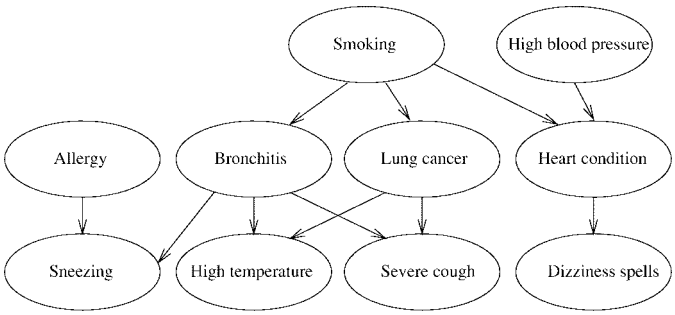

Fig. 2. Topology of simplified Bayes net for medical diagnosis.

[14], fault detection in electronic circuits [32], and medical diagnosis [33]. The above is by no means an exhaustive list of applications; see [31] for pointers.

The most commonly used operation on Bayes networks is the computation of marginal probabilities, usually after observing the state of some variables (introduction of *evidence*). For example, we might want to know the probability that the vase is broken given that a tremor has occurred, in the model of Fig. 1 (a prediction problem). In the model of Fig. 2, we might be interested in the probability of lung cancer given that severe cough and sneezing were observed (a diagnostic problem). (This operation is also called probability updating, or belief updating, in the literature.) It is thus unfortunate that, although a polynomial-time algorithm for computing the probabilities exists for polytrees [24], the problem was proved to be NP-hard in the general case in [9]. Several categories of exact algorithms exist for computing posterior probabilities: conditioning [10], clustering and junction trees [23], [26], term evaluation [27], and arc reversal [39]. Several variants of these algorithms attempt various refinements of these schemes, e.g., [15]. All of these algorithms are exponential-time in the worst case, where the runtime is a function of the topology and the number of states each variable can assume. (In this paper, we refer only to networks where each variable has a finite number of states.)

In the hope of avoiding an exponential runtime, a host of approximation algorithms have emerged. As it turns out, theoretically, even *approximating* marginal probabilities in belief networks is NP-hard, and thus there *is no* polynomial-time (deterministic) approximation algorithm unless $P = NP$ [12]. Most approximation algorithms are less affected by network topology, and are dependent on the actual probabilities as to their runtimes and quality of approximation. If the topology of a given network is such that exact algorithms are expected to take a long runtime, it may be advisable to run an approximation algorithm and hope that the probabilities are such that we can get a good approximation in reasonable time for the problem instance at hand. In addition, most approximation algorithms have an *anytime* behavior, which facilitates trading off time for precision in a graded manner.

Two major categories of marginal probability approximation algorithms exist: randomized approximation algorithms and deterministic approximation algorithms. In [21], approximation is achieved by stochastically sampling instantiations of the network variables. Later work in randomized approximation algorithms attempts to increase sampling efficiency [4], and to handle the case where the probability of the evidence is very low [17], which is a serious problem for most sampling algorithms. In what follows, we focus on the second category, *deterministic* approximation algorithms. In *bounded* conditioning [22], one uses the conditioning method, but conditions only on a small, high probability, subset of the (exponential size) set of possible assignments to the cutset variables. Other approximation algorithms attempt to simplify the network by removing arcs between nodes that are *almost* independent, to produce a network that is hopefully tractable topologically. An exact algorithm is then run on the "approximate" network, to produce an approximate answer [25]. Another source of complexity is the large number of states per node in various applications. To alleviate that problem, an approximation based on merging states was suggested [45]. The scheme begins by making all variables unary valued, and successively refining the states of variables, while performing probability updating on the approximate network and thus getting a successively better approximation in each step.

Another category of deterministic approximation algorithms is based on deterministic enumeration of terms or assignments to variables in the network. The idea is to enumerate a set of high-probability complete assignments to all the variables in the network (but frequently partial assignments suffice, as will be demonstrated). The probability of each such assignment can be computed quickly: in O(n), or sometimes even (incrementally) in O(1). The probability of a particular instantiation to a variable v (say $v = v_1$) is approximated by simply dividing the probability mass of all assignments which contain $v = v_1$ by the total mass of enumerated assignments. If the enumerated assignments have a sufficiently large probability mass, we get a good approximation.

Incremental operations for probabilistic reasoning, among them a suggestion for approximating marginal probabilities by enumerating high-probability terms, are presented in [13]. Of particular interest is the skewness result: if a network has a distribution such that every row in the distribution arrays has one entry greater than $(n-1)/n$, then collecting only $n+1$ assignments, we also have at least $2/e$ of the probability mass. Taking the topology of the network into account, and using term computations, this can presumably be achieved efficiently. However, the skewness assumption as is seems somewhat restrictive. The assumption may hold in some domains, such as circuit fault diagnosis, but not in medical diagnosis, or in the randomly generated networks on which we tested our algorithms. Nevertheless, [16] presented initial theoretical and empirical results that even weak skewness leads to a favorable overall result (i.e., large probability mass in a relatively small number of assignments) on the average. This paper extends the above empirical results to sets of partial assignments.

In [32], partial assignments to nodes in the network are created from the root nodes down. The probability of each such assignment is easily computable. Much saving of computational effort is achieved by not bothering about irrelevant nodes, i.e., nodes not above some node that is in the query set, or nodes that are d-separated from the evidence nodes. Later

in that paper, an assumption of *extreme probabilities* is made. This is similar to the skewness assumption above. In fact, in the circuit fault diagnosis experiment in [32], the numbers actually used are well within the bounds of the skewness assumption. The algorithm makes use of a conflict scheme in order to narrow the search.

As suggested in [19] and [41], belief networks frequently have independence structure that is not represented by the topology.[2] Sometimes independence holds given a *particular assignment* to a set of variables V, rather than to all possible assignments to V. In such cases, the topology is no help in determining independence (e.g., d-separation might not hold), the actual distributions might have to be examined. In [41], the idea of independence-based (IB) assignments (see below) was introduced.

Formally, an assignment is a set of (node, value) pairs, which can also be written as a set of $node = value$ instantiations. An assignment is consistent if each node is assigned at most one value. Two assignments are compatible if their union is consistent. Each assignment denotes a (sample space) event, and we thus use the assignment and the event it denotes as synonymous terms whenever this does not lead to ambiguity. An assignment $\mathcal{A}$ subsumes assignment $\mathcal{B}$ iff $\mathcal{A} \subseteq \mathcal{B}$. Among all assignments, we are usually interested only in relevant nodes—those supported by evidence or query nodes. A node v is supported by a set of nodes V if $v \in V$ or if v is an ancestor of some node in V. An assignment is *properly* supported by a set of nodes V if all the nodes in the assignment have a directed path of *assigned nodes* to a node in V.

The *IB condition* holds at a node v with respect to an assignment $\mathcal{A}$ if the value assigned to v by $\mathcal{A}$ is independent of all possible assignments to the ancestors of v given $\mathcal{A}_{\text{parents}(v)}$, the assignment made by $\mathcal{A}$ to the immediate predecessors (parents) of v. An assignment is IB if the IB condition holds at every $v \in \text{span}(\mathcal{A})$, where $\text{span}(\mathcal{A})$ is the set of nodes assigned by $\mathcal{A}$. A *hypercube* $\mathcal{H}$ is an assignment to a node v and some of its parents. In this case, we say that $\mathcal{H}$ is *based* on v. $\mathcal{H}$ is an IB hypercube if the IB condition holds at v with respect to $\mathcal{H}$. The *conditional probability* associated with a hypercube $\mathcal{H}$ is called the hypercube probability [denoted $P'(\mathcal{H})$]. It is *not* necessarily the same as the prior probability of the assignment, denoted $P(\mathcal{H})$.

Consider, for example, the network of Fig. 2, and suppose (for simplicity) that sneezing occurs with probability 1 whenever allergy occurs, independent of "Bronchitis." Then the assignment $\mathcal{A} = \{\text{Allergy = present, Sneezing = present}\}$ is an IB assignment, because the IB condition holds at every node in the span of $\mathcal{A}$: at the "Sneezing" node because given allergy the probability of sneezing is 1, independent of any other ancestor (in this case "Bronchitis" and "Smoking"), and at the "Allergy" node—vacuously (since "Allergy" has no parents). Assignment $\mathcal{A}$, being an assignment to the "Sneezing" node and some of its parents, also happens to be an IB hypercube based on the "Sneezing" node, and its hypercube probability, $P'(\mathcal{A})$, is 1. However, the prior probability $P(\mathcal{A})$, in this case is equal to the prior probability that allergy is present, which is usually less than 1. The nodes "Allergy," "Bronchitis," and "Smoking" are supported by the "Sneezing" node, and no other nodes are supported by "sneezing." Assignment $\mathcal{A}$ is *properly* supported by "Sneezing," as the path ("Allergy," "Sneezing") is completely within the span of $\mathcal{A}$.

In [41], IB assignments were the candidates for relevant explanation. We suggest that computing marginal probabilities (whether prior or posterior), can be done by enumerating high-probability IB assignments, rather than complete assignments. Since IB assignments usually have fewer variables assigned than complete assignments (as evident from the above example), each IB assignment is expected to hold more probability mass than a respective complete (or even a query and evidence supported) assignment. The probability of an IB assignment is also easy to compute [41]

$$P(\mathcal{A}) = \prod_{v \in \text{span}(\mathcal{A})} P(\mathcal{A}_{\{v\}} | \mathcal{A}_{\text{parents}(v)}) \tag{1}$$

where $\mathcal{A}_S$ is the assignment $\mathcal{A}$ restricted to the set of nodes S. The terms in the product can each be efficiently retrieved from the conditional distribution array (or other representation) of the node conditional distribution.

One might argue that searching for high-probability assignments for approximating marginal distributions is a bad idea, since coming up with the highest-probability assignment is NP-hard [42]. Thus, we are using the solution to an NP-hard problem to find an approximate solution to an NP-hard problem, where we might expect that a polynomial time algorithm can be sufficient to compute approximations. However, as noted above, [12] showed that this problem is also NP-hard. Therefore, using this kind of approximation algorithm is a reasonable proposition, provided that some subclasses of the problem that are bad for existing algorithms can be shown to behave well, either theoretically or by empirical results that show good behavior on the average. Since runtimes of our algorithms depend in a complicated manner on the conditional probabilities, it is very hard to get any theoretical bounds on the runtime for interesting classes of networks. In this paper, we take the experimental route to justify our performance claims.

The rest of the paper is organized as follows. Section II discusses the details of how to approximate posterior probabilities from a set of high-probability IB assignments. Section III reviews the IB MAP search algorithm of [41], and discusses a faster heuristic best-first algorithm for finding the high-probability IB assignments, based on the cost-sharing heuristic presented in [7]. Section IV reviews the reduction of IB MAP computation to linear systems of equations [41], incorporating improvements that reduce the number of equations. Searching for next-best assignments using linear programming is discussed, as is a method for avoiding overlaps of IB assignments. Section V presents results of experiments on random networks: mass distribution, timing of search for the IB MAP, and handling overlapped IB assignments (for the cost-sharing algorithm). We conclude with a discussion of related work, and on applying IB assignments to approximation algorithms presented in the literature.

[2] This type of independence structure has recently been renamed "context-specific independence" (CSI) [5].

II. Approximating Marginal Probabilities

The probability of a certain node instantiation, $v = v_1$, is approximated by the probability mass in the IB assignments containing $v = v_1$ divided by the total mass. If we need to find the probability of v, we call v a *query* node. Nodes where evidence is introduced are called *evidence* nodes. We also assume that the evidence is conjunctive in nature, i.e., it is an assignment of values to the evidence nodes. We need to assume that each enumerated IB assignment $\mathcal{A}$ contains *some* assignment to query node v. Otherwise, it might be impossible to tell which part of the mass of $\mathcal{A}$ supports $v = v_1$. Let us assume for now that this is indeed the case,[3] i.e., we have a set I containing IB assignments, and if $\mathcal{A} \in I$ then $v \in \text{span}(\mathcal{A})$. Thus, to approximate the probability of $v = v_1$, use

$$P_a(v = v_1) = \frac{P(\{\mathcal{A} | \mathcal{A} \in I \wedge \{v = v_1\} \in \mathcal{A}\})}{P(\{\mathcal{A} | \mathcal{A} \in I\})}$$

where the probability of a set of assignments is the probability of the event that is the union of all the events standing for all the assignments (not the probability of the union of the assignments). If we are computing the prior probability of $v = v_1$, we can either assume that the denominator is 1 (and not bother about assignments assigning v a value other than v_1), or use $1 - P(\{\mathcal{A} | \mathcal{A} \in I\})$ as an error bound. If all IB assignments are disjoint, the probability of the union is easily computable, and is simply the sum of probabilities of the IB assignments. However, since IB assignments are partial, it is possible for the events denoted by two different IB assignments to overlap. Thus, to compute the probability of a set, some other method must be used.

Computing the union of the IB assignments in a representation that makes computation of the probabilities easy is nontrivial. It turns out that we can use the inclusion–exclusion principle, due to the fact that the union of compatible IB assignment is also an IB assignment [44]. For example, let $\{u, v, w\}$ be nodes, each with a domain $\{1, 2, 3\}$. Then $\mathcal{A} = \{u = 1, v = 2\}$ has an overlap with $\mathcal{B} = \{u = 1, w = 1\}$. The overlap $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$ is also an assignment: $\mathcal{C} = \{u = 1, v = 2, w = 3\}$. (Note that for two assignments $\mathcal{A}, \mathcal{B}$, the *union* of $\mathcal{A}$ and $\mathcal{B}$ denotes the event that is the *intersection* of the events denoted by $\mathcal{A}$ and $\mathcal{B}$.)

Despite this property, evaluating the probability of a set of IB assignments may require the evaluation of an exponential number of terms. That is due to the equation for implementing the inclusion–exclusion principle

$$P(\cup_{1 \le i \le m} E_i) = \sum_{k=1}^{m} (-1)^{k+1} \sum_{1 \le a_1 < \cdots < a_k \le m} \cap_{i=1}^{k} E_{a_i}$$

where E_i is the ith event. It is always possible to force all IB assignments to be nonoverlapping, by using only nonoverlapping hypercubes in the search algorithms [38], [44]. However, this comes at a cost: the resulting IB assignments will then have more variables assigned, and thus much less probability mass. Several other ways exist to overcome this problem, but the best method depends on the algorithm for generating the IB assignment. We thus postpone this issue until the following sections, where the details of the IB assignment generators are discussed, and assume for the moment that there are no overlaps.

Given a set of query and evidence nodes, all nodes not supported by either the query or evidence nodes (redundant nodes) need never appear in an IB assignment in our search. A node (properly) supported by the evidence nodes is called (respectively, properly) evidentially supported, and a node supported by a query node is called query supported. Before we start searching for IB assignments, drop all irrelevant nodes, i.e., evidence nodes that are d-separated from the query nodes by other evidence nodes, as well as all the nodes that are not either query supported or supported by one of the remaining evidence nodes.

The basic approximation algorithm is described below. The existence of a *generator* is assumed. Each time the generator is called, it returns the next-best (next highest probability) IB assignment consistent with a set of initial assignments. Some variants of the algorithm use more than one generator instance.

- *Input:* A Bayesian belief network B, evidence $\mathcal{E}$ (a consistent assignment), a query node q.
- *Output:* Successively improved approximations for $P(q = q_i)$, for each value q_i in the domain of node q.

1) Preprocessing
   - Sort the nodes of B such that no node appears after any of its ancestors.
   - Initialize the IB hypercubes for each node $v \in B$.
2) Initializing: remove redundant nodes, and for each q_i in the domain of q do:
   a) Set up an empty result set for q_i.
   b) Add the assignment $\mathcal{E} \cup \{q = q_i\}$ to the initial assignment set for the generator.[4]
3) Repeat until time limit or generator returns null:
   a) Get next-best IB assignment $\mathcal{A}$ from the generator.
   b) Add $\mathcal{A}$ to the result set of q_i, where $\{q = q_i\} \in \mathcal{A}$.
   c) Update the posterior probability approximation.

The simplest generator is a best-first search with the current probability heuristic, which is exactly the inner loop of the algorithm in [41], (also described in Section III). In this paper, we also look at two other generators: a best-first search algorithm based on the cost-sharing heuristic, and an integer linear program scheme, modified from [41].

The posterior probability approximation for $q = q_i$ given the evidence is

$$P_a = \frac{P(\text{result set for } q_i)}{\sum_i P(\text{result set for } q_i)}.$$

As before, for null evidence, $1 - \sum_i P(\text{result set for } q_i)$ is the unassigned probability mass, and can be used to bound the error, as in [32]. For a discussion on how to bound error probabilities for the nonnull evidence case, as well as how to

[3] In subsequent work [38] this assumption was relaxed for special cases, such as query nodes which are root nodes.

[4] We need to enforce some assignment to the query node, in order to comply with our assumption.

generalize the algorithm to handle several query nodes, see [44]. In [38] and [44], the issue of using several generators is explored. In particular, we might want to use one generator for each possible query node instantiation, and an additional generator for the negation of the evidence. In this case, each generator would get an initialization set of size 1. Selection between the generators would be based on a meta-reasoning mechanism, where the criterion for selection would be to use the generator where the expected result would provide better bounds on the conditional probabilities (this is not necessarily the one providing the IB assignment with the greatest mass), possibly in a parallelized (or distributed) search scheme.

III. Shared-Costs Heuristic Search

In this section, we discuss best-first heuristic search generators for the marginal probability approximation algorithms. We begin by reviewing the simple, current probability (also called "cost-so-far") heuristic algorithm [41], [44], and then discuss the better cost-sharing heuristic.

A. Review of the Cost-So-Far Heuristic Search

The best-first algorithm keeps a sorted agenda of states, where a state is an assignment, a node last expanded, and a probability estimate:

- *Input:* A Bayesian belief network B, a set of consistent assignments $\mathbf{E}$.
- *Output:* The next best IB assignment that subsumes some $\mathcal{E} \in \mathbf{E}$.

1) Initializing: for each $\mathcal{E}$ in $\mathbf{E}$, push into the agenda the assignment $\mathcal{E}$ with a probability estimate of 1.
2) Repeat until empty agenda:
   a) Pop assignment with highest estimate $\mathcal{A}$ from the agenda, and remove duplicate assignments (they will all be at the top of the agenda).
   b) If $\mathcal{A}$ is IB, return it.
   c) Otherwise, expand $\mathcal{A}$ at v, the next node, into a set of assignments $\mathcal{S}$, and for each assignment $\mathcal{A}^j \in \mathcal{S}$ do:
      i) Estimate the probability of $\mathcal{A}^j$.
      ii) Push $\mathcal{A}^j$ with its probability estimate and last-expanded node v into the agenda.

When the generator is resumed (i.e., called after it returns the first time), the resumption point is at step 2. Expanding a state and the probability estimate is exactly as in [41]: $\mathcal{A}^j = \mathcal{A} \cup \mathcal{H}^j$, where $\mathcal{H}^j$ is the jth IB hypercube based on v that is maximal with respect to subsumption and consistent with $\mathcal{A}$. The probability estimate is the product of hypercube probabilities for all nodes where the IB condition holds.

B. Dealing with Overlapping Assignments

One way to deal with overlapping assignments is to approximate the inclusion–exclusion formula, by ignoring high order terms in the computation. That makes sense because low-probability assignments are going to be ignored in the approximation algorithm anyway. Theoretically, this is a bad idea. As shown in [28], we need a very large number of terms (about $2^{\sqrt{n}}$ in the worst case) to get a good approximation of the inclusion–exclusion formula, in the general case. Still, this might be feasible in a practical algorithm.

Another way to handle overlap is to use inclusion–exclusion only for a *small* set of overlapping assignments, and prevent the occurrence of sets of overlapping assignments with cardinality strictly greater than some small integer constant k. The exact value of k depends on which algorithm variant we use. In the best-first heuristic search algorithms, it is hard to prevent an IB assignment from overlapping all other assignments. If an IB assignment $\mathcal{A}$ comes up that is not subsumed by some previously generated IB assignment (in which case it is thrown out), we can do the following test. If $\mathcal{A}$ overlaps $m > k$ IB assignments, we split it into several assignments (which are not necessarily IB anymore), and toss the new assignments back into the agenda.

Each "split" assignment is a copy of the current assignment augmented by an assignment to one or more additional nodes. The node (or nodes) must not be in the span of the current assignment, but appear in the span of some of the overlapped assignments. As a result of the split, most of the new assignments will overlap with less than m.

Two questions remain: 1) how to do the split (i.e., which nodes to select for assignment), and 2) how to evaluate a split assignment when returning it into the agenda. There are several possible answers to the first question.

1) Select an unassigned root node, or if none are available (all root node assignments are equal), a successively distant unassigned node down the acyclic graph. Such an unassigned node must exist, or the current assignment is subsumed by an IB assignment already in the result set, and should have been thrown out.
2) Select some other node. One possibility is a node as low down as possible.

In the first case, the resulting "split" assignments will all be IB assignments, in which case there are two possibilities: 1) try to use them in the result sets immediately or 2) push them back into the agenda. In the second case, the resulting assignments will usually not be IB, and they must be pushed into the agenda. Intuitively, method 1 will allow us to use the results faster, but may cause problems if m is large. Method 2 will defer the problem, and is likely to do better in such cases.

Heuristic evaluation of the split assignment requires no modifications from the standard heuristic function. The only change in the algorithm needed to handle these split assignments is to change the last expanded node to one less than the node selected for the split, unless we already know the assignment to be IB (e.g., when selecting a root node for the split), in which case no modification is necessary. In our experiments, however, assignment splitting turned out to be unnecessary (see Section V).

C. Cost-Sharing Heuristic Search

To improve the performance of the search algorithm, we need to use a different heuristic than cost-so-far, which gives little information early on in the search. Including costs that will be incurred later on in the search (higher up in the

DAG) give us a better estimate. However, one cannot just add the costs to be incurred in the future, because in multiply connected networks one node cost (negative logarithm of probability) would be counted multiple times, and we would no longer have an admissible heuristic. The idea of dividing the cost to be incurred by the number of children, the "cost-sharing" heuristic, was pursued in [7] for weighted proof graphs (Weighted AND/OR DAG's).[5] The cost sharing heuristic showed a marked improvement in performance over the cost-so-far heuristic when applied to graphs generated by WIMP [6]. Since the above generator is a best-first search algorithm that uses the cost-so-far heuristic, plugging in the cost-sharing heuristic ought to give us a great improvement. In order to take advantage of the cost-sharing heuristic, we need to reduce the Bayes network into a weighted AND/OR DAG (WAODAG), such that probabilities are monotonic in the costs in the WAODAG.

For example, consider the Bayes network of Fig. 3, a part of the medical diagnosis network that would be relevant if the evidence is that "High temperature" is present. Each node has two states: T (symptom or disease present), and F (symptom or disease absent). Its corresponding WAODAG appears in Fig. 4, where a curved arc across a set of edges denote that the node incident on these edges is an AND node. The number next to each node is its cost, with zero costs omitted for clarity. Node names are abbreviated, e.g., "lc" for "Lung cancer." Each step number in the figure refer to a construction step as defined below, and denote that all items (nodes or arcs) at the same height in the figure are constructed in that step.

1) Constructing the WAODAG: To convert our problem into the WAODAG formulation, we perform a construction similar to [8]. The algorithm is given a belief network $B = (V, A, P)$, and evidence $\mathcal{E}$. (Note that query nodes essentially become evidence nodes, in the context of searching for the best IB assignment.) Assume without loss of generality that all nodes are either evidence or query nodes, or ancestors of some such node (otherwise they can just be dropped from the diagram). The construction is as follows (refer to the examples, Figs. 3 and 4).

1) For each possible node-state $(v, d) \in \mathcal{E} \cup \{(v, d) \mid v \in V - \text{span}(\mathcal{E})\}$ where d is in D_v (domain of v), construct OR node N^{v^d}. Note that for each evidence node, only one state is possible.
2) Construct an AND node S, with parents N^{v^d} for all $(v, d) \in \mathcal{E}$, i.e., all the evidence node-states.
3) For each maximal IB hypercube based on any node v that assigns value d to v, construct an AND node $H_i^{v^d}$, where i is the index of the hypercube among all the hypercubes that are based on v and assign a value d to v (the actual order is immaterial). We call $H_i^{v^d}$ the node image of the hypercube, and use it as a synonym for the hypercube itself whenever unambiguous.
4) For each maximal IB hypercube as above, construct a node $SC_i^{v^d}$, the hypercube's "self cost" node.
5) Construct a directed edge from each hypercube $H_i^{v^d}$ to N^{v^d}.
6) Each hypercube assigns some state to some of a node's parents. For each $(w, d') \in H_i^{v^d}$ such that $w \neq v$, construct an edge from node $N^{w^{d'}}$ to node $H_i^{v^d}$.
7) From each self-cost node $SC_i^{v^d}$ construct a directed edge $ESC_i^{v^d}$ (self-cost edge) to the respective $H_i^{v^d}$. $SC_i^{v^d}$ is essentially an AND node with no parents.
8) The cost of each self cost node $SC_i^{v^d}$ is defined by the respective hypercube probability: $c(SC_i^{v^d}) = -\log P'(H_i^{v^d})$.
9) The cost of any other node is 0.

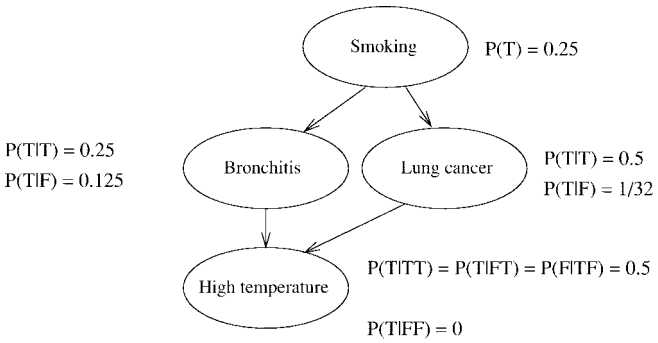

Fig. 3. Fragment of medical diagnosis network.

2) Admissibility and Edge-Based Search: In order to make the cost-sharing heuristic admissible, Charniak and Husain had to define the search in terms of sets of *edges*, rather than nodes. A search state (set of edges) has to obey the *minimal cut* property, as follows. A cut of an AND DAG is a set of edges E such that every path from any root node to a leaf node contains an edge from E. A cut is minimal if it is set-wise minimal, i.e., if no edge can be removed from E such that there is still a cut. (What we call a "minimal cut" here is called a "cut" in [7].) For an AND/OR DAG, E is a cut if there is contains a complete AND DAG (intuitively: completely specified proof) for which E is a cut. Likewise, for a minimal cut of an AND/OR DAG. In Fig. 4, $\{c11, c12\}$ would be a minimal cut of the AND DAG supported by $H_1^{ht^T}$, and thus a minimal cut of the entire WAODAG. In our reduction, we need to suggest a search operator that preserves the minimal cut property, and such that states corresponding to all (properly supported) IB assignments are reachable. If we do all of the above, we are assured by the results of [7], that the heuristic is admissible for our search, and that this algorithm variant indeed comes up with the highest-probability IB assignment.

Let AD be a complete AND DAG in our WAODAG, W, and let $\mathcal{H}$ be the set of hypercubes $H_i^{v^d}$ in AD. Define $a(AD)$ to be the assignment consisting of the union of all the hypercubes $H_i^{v^d} \in \mathcal{H}$. Let C be a (minimal) cut of W consisting only of the self-cost edges of a set of hypercubes $\mathcal{H}$. As above, define $a(C)$ as the assignment consisting of the

[5] In a weighted proof graph, each node v has a cost $c(v)$. The equivalent problem in this formalism is to find a least-total-cost proof of the evidence, a subgraph of least total cost. In such a proof, the evidence node S must be included. Additionally, if an AND node is in the proof, then all its parents must be in the proof. Similarly, if an OR node is in the proof, one of its parents must be included. Such a subgraph (a proof) is a complete AND DAG of the WAODAG.

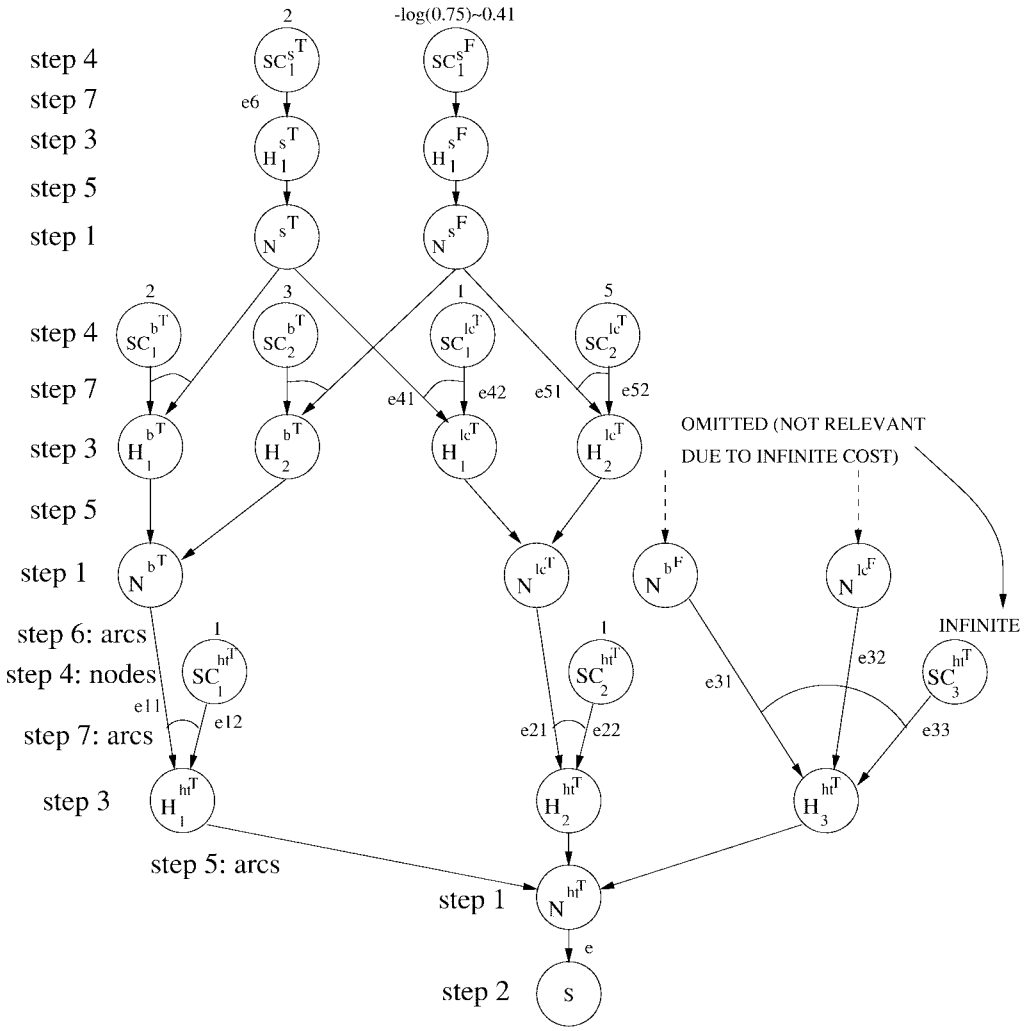

Fig. 4. WAODAG generated from medical diagnosis network.

union of all the hypercubes in $\mathcal{H}$. Likewise, if C is a set of edges (not necessarily a cut or a minimal cut), define $a(C)$ to be the union of all the hypercubes that have their self-cost edge in C.

Lemma 1: Let AD be a complete AND DAG in W, and let $\mathcal{A} = a(C)$, where the set of hypercubes in AD is $\mathcal{H}$. If $\mathcal{A}$ is consistent, then it is an IB assignment.

Proof: Let v be an arbitrary node in span($\mathcal{A}$). It is sufficient to show that the IB condition holds at v with respect to $\mathcal{A}$. There exists a unique (since $\mathcal{A}$ is consistent) value d such that $(v, d) \in \mathcal{A}$. Since (v, d) is in the assignment, then there must be a hypercube $H_i^{v^d} \in \mathcal{H}$ based on v (assigning it the value d). That is because for v to be assigned, either the above holds, or there is some hypercube $H_j^{u^{d'}} \in \mathcal{H}$ (for some d', j) based on some child u of v, that assigns v the value d. In the latter case, $H_j^{u^{d'}}$ is an AND node in AD, and one of its parents in W is N^{v^d} by construction. Since AD is a complete AND DAG containing $H_j^{u^{d'}}$, then $N^{v^d} \in AD$. Also, for AD to be complete, one parent of N^{v^d} must also be in AD. But all the parents of N^{v^d} are hypercubes of the form $H_i^{v^d}$ by construction. Thus, for some i, $H_i^{v^d}$ is in AD and thus also in $\mathcal{H}$. Since the IB condition holds at v with respect to $H_i^{v^d}$, and assigning other nodes (consistently) cannot affect the IB condition at v, then the IB condition holds at v with respect to $\mathcal{A}$. □

Lemma 2: Let C be a (minimal) cut of W consisting only of the self-cost edges of a set of jointly consistent hypercubes $\mathcal{H}$. Then the assignment $\mathcal{A} = a(C)$ is a consistent IB assignment subsumed by the evidence $\mathcal{E}$.

Proof: Let AD be the complete AND DAG for which C is a cut. AD contains a WAODAG node for each evidence node-state, and thus C must contain a self-cost edge for *some* hypercube $H(e)$ based on each evidence node e, and consistent with the node-state of e in W. Thus, the union of the above evidence hypercubes subsumes $\mathcal{A}$, and is a subset of $\mathcal{A}$, and therefore $\mathcal{A}$ subsumes $\mathcal{E}$. $\mathcal{A}$ is a consistent assignment by construction, and it is thus sufficient to prove that it is IB. We show first that $\mathcal{A} = a(AD)$. First, if $H_i^{v^d} \in AD$ then $ESC_i^{v^d} \in C$, as otherwise there is a path $SC_i^{v^d}, ESC_i^{v^d}, H_i^{v^d}, \cdots, S$ in AD, which obviously has no edge in C (contradiction). Likewise, if $ESC_i^{v^d} \in C$, then $H_i^{v^d} \in AD$, otherwise dropping $ESC_i^{v^d}$ from C does not create a path from root to sink (contradiction). Now, since

AD and C have the same hypercube sets, then $\mathcal{A} = a(C) = a(AD)$, and it thus follows immediately from Lemma 1 that $\mathcal{A}$ is IB. □

Lemma 3: Let $\mathcal{A}$ be a consistent IB assignment subsumed by the evidence, and $\mathcal{H}$ a set of hypercubes whose union is exactly $\mathcal{A}$, and such that $\mathcal{H}$ contains a hypercube for every node in span($\mathcal{A}$) (the fact that such a set exists for every IB assignment is proved in [40]; this set is not necessarily unique) then

$$C = \{ESC_i^{v^d} \mid H_i^{v^d} \in \mathcal{H}\}$$

is a (minimal) cut consisting only of self cost edges of jointly consistent hypercubes.

Proof: First, our AND DAG AD of which C is a cut is constructed as follows: a node-state $(v, d) \in AD$ just when $(v, d) \in \mathcal{A}$. A hypercube $H_i^{v^d} \in AD$ just when $H_i^{v^d} \in \mathcal{H}$. A self-cost node $ESC_i^{v^d} \in AD$ just when $H_i^{v^d} \in \mathcal{H}$. Clearly, AD is a complete AND DAG, as it includes all the sink nodes (the evidence node-states) and for every OR node in AD, i.e., node-state (v, d), one parent node (the $H_i^{v^d} \in \mathcal{H}$) is in AD. Likewise, for each hypercube in AD (an AND node) all of its parent nodes are in AD, because the hypercube subsumes $\mathcal{A}$, and its self-cost node is in AD by construction. For the self-cost nodes, since they have no parents, all of their parents are in AD, vacuously. Thus, AD is a complete AND DAG.

Also, C is a cut, because the only root nodes are self-cost nodes, and each of them only has one edge, which is in AD by construction. C is minimal because it consists only of self-cost edges (by construction), while and all the self cost edges are incident on root nodes. Thus, removing such an edge $ESC_i^{v^d}$ from C will allow a path from root node $SC_i^{v^d}$ to a sink node. Thus, C is a minimal cut of AD, which is a complete AND DAG in W, and is thus a minimal cut of W. □

We now proceed to the computation of the heuristic costs h, which are defined in a manner similar to [7], as follows:

- the heuristic cost of a self-cost edge is the cost of its source node, i.e., $h(ESC_i^{v^d}) = c(SC_i^{v^d})$;
- the heuristic cost of a hypercube $h(H_i^{v^d})$ is the sum of the heuristic costs of the incoming edges;
- the heuristic cost of each edge with a hypercube node as a source is the heuristic cost of its source hypercube;
- the heuristic cost of any other edge is the heuristic cost of its source node N^{v^d} divided by the number of children that v has in B;
- the heuristic cost of a node-state node N^{v^d} is the minimum of the heuristic costs of all of its incoming edges.

Since W is a DAG, and the above defines costs only in terms of the belief network or edges and nodes above, this definition is grounded.

Theorem 1: Let C be a (minimal) cut of W consisting only of consistent self-cost-node edges, and $\mathcal{A} = a(C)$. Then $\mathcal{A}$ is a consistent IB assignment subsumed by the evidence, and its probability satisfies

$$-\log P(\mathcal{A}) = \sum_{e \in C} c(e) = \sum_{e \in C} h(e).$$

Proof: By Lemma 2, $\mathcal{A}$ is a consistent IB assignment subsumed by the evidence. Thus, its probability is a product of conditional probability terms (1), which we rewrite as hypercube probability terms

$$P(\mathcal{A}) = \prod_{v \in \text{span}(\mathcal{A})} P'(H_i^{v^d})$$

where $H_i^{v^d}$ is the hypercube based on v in the set of hypercubes $\mathcal{H}$ from which $\mathcal{A}$ was constructed. Taking the logarithm of both sides and multiplying by -1, we get

$$\begin{aligned} -\log P(\mathcal{A}) &= -\log \prod_{v \in \text{span}(\mathcal{A})} P'(H_i^{v^d}) \\ &= \sum_{v \in \text{span}(\mathcal{A})} -\log P'(H_i^{v^d}) \\ &= \sum_{v \in \text{span}(\mathcal{A})} c(SC_i^{v^d}). \end{aligned}$$

But since the sum is over all the nodes in the assignment, and each node has its hypercube in $\mathcal{H}$, which in turn consists of exactly one hypercube for each self cost edge, we have

$$-\log P(\mathcal{A}) = \sum_{v \in \text{span}(\mathcal{A})} c(SC_i^{v^d}) = \sum_{e \in C} c(e).$$

□

3) Algorithm Summary and Correctness Proof: A search state is a set of edges, a minimal cut C. For convenience and efficiency, we also keep the hypercubes, last expanded node, current heuristic value, etc., but these can all be computed from the cut C. The heuristic value of a state is the sum of its heuristic edge costs. Our expansion operator σ is a function from a set of edges to a set of sets of edges, (i.e., essentially from a state to a set of next states), and is similar to that of [7]. The generator for next-best IB assignment follows.

- *Input:* A Bayesian belief network B, and evidence $\mathcal{E}$ (a consistent assignment).
- *Output:* IB MAP assignment.

1) Initializing:
   a) Remove redundant nodes.
   b) Create the WAODAG from the top down, while computing node and heuristic edge costs.
   c) Push the edge set $\{E\}$ onto the agenda, where E is the set of edges incident on S, with heuristic cost equal to $h(E)$.
2) (Resumption point): Repeat until empty agenda:
   a) Pop state C with lowest heuristic cost estimate from the agenda, and remove duplicate states (they will all be at the top of the agenda).
   b) If the assignment $a(C)$ is IB (all edges are self cost edges) return it.
   c) Otherwise, find the earliest node N^{v^d} which is a source of an edge e in C, compute $\sigma(C)$. That is, expand C at e (we also call this "expanding at node v") into a set of states $\mathcal{C}$. For each state $C' \in \mathcal{C}$ do the following:
      i) Find if C' corresponds to a consistent assignment, and if not, discard it.

ii) Compute the heuristic value for C' as the sum of edge costs.

iii) Push C' into the agenda.

Expanding a state C at e with source N^{v^d} (computation of σ) works as follows: Let E be the set of edges with source N^{v^d} in C. The parents of N^{v^d} are $H_i^{v^d}$ with $1 \leq i \leq$ indegree($N_i^{v^d}$). Then the new states $C_i, 1 \leq i \leq$ indegree(N^{v^d}) are $C_i = C - E +$incoming($H_i^{v^d}$). Note that by construction, each application of the expansion adds one self-cost edge to C, thereby adding a hypercube to the assignment defined by the cut. Each of the new states amounts to a different choice of hypercube at v, just like the cost-so-far algorithm.

A sample run of the algorithm on the Bayes network of Fig. 2 with evidence "high temperature present," would proceed as follows. Removal of the redundant nodes would result in the network of Fig. 3. The resulting WAODAG appears in Fig. 4. Push the state $(\{e\}, 3)$, where the singleton edge set $\{e\}$ is the state itself, and 3 is its heuristic cost. In the first pass through the loop, $(\{e\}, 3)$ is popped, and the states: $(\{e11, e12\}, 4)$, $(\{e21, e22\}, 3)$, and $(\{e31, e32, e33\}, \infty)$ are pushed. Second iteration pops $(\{e21, e22\}, 3)$, and pushes $(\{e22, e41, e42\}, 3)$ and $(\{e22, e51, e52\}, 6.2)$. Third iteration pops $(\{e22, e41, e42\}, 3)$ and pushes $(\{e22, e42, e6\}, 4)$. Fourth iteration (last one) pops $(\{e22, e42, e6\}, 4)$ which consists only of self cost edges, and returns. The result corresponds to the assignment smoking = T, lung cancer = T, high-temperature=T, (with "Bronchitis" uninstantiated), and a probability of $2^{-4} = 1/16$. (Note that in the last iteration, the equally cheap state $(\{e11, e12\}, 4)$ could have been selected instead, causing several more iterations, but with the same end result).

This algorithm diverges from that of [7] in that when an edge from an OR node is expanded to include an edge from a hypercube $H_i^{v^d}$ to N^{v^d}, we expand the hypercube node (which is an AND node) as well in the same expansion step. This does not affect either the heuristic value or the reachability of final states, since for all consistent self-cost-only cuts C, either C is reachable, or $a(C)$ is subsumed by some IB assignment $a(C')$, where C' is reachable. Hence, to prove that this heuristic is admissible, the results of [7] can be directly applied.

Theorem 2: Let $\mathcal{A}$ be a consistent IB assignment maximal with respect to subsumption among all IB assignments that are subsumed by the evidence. Then there exists a self-cost-only cut C that is reachable from the initial state such that $\mathcal{A} = a(C)$.

Proof: This theorem was proved for the expansion operator in [41]. We show that the expansion operator σ is equivalent to that of [41], in the following sense: for each operator application τ of [41], there is a sequence of one or more edge expansion (σ) steps.

Let $\mathcal{A}$ be an assignment that is τ-reachable from the evidence, where node v is the last expanded. Since τ always expands by selecting a hypercube and setting the next state to be the union of $\mathcal{A}$ with the hypercube, then $\mathcal{A}$ is a union of hypercubes $\mathcal{H}$. The order of node expansion using τ is such that children are always expanded before parents [i.e., all nodes that are descendants of v have already been expanded, if they are in span($\mathcal{A}$)], and we assume the same node ordering here. Note that τ always expands the minimal active fringe node, i.e., the earliest node in span($\mathcal{A}$) for which the IB condition does not hold with respect to $\mathcal{A}$. Let C be the set of edges defined as in Lemma 3 (C is not necessarily a cut, since $\mathcal{A}$ is not necessarily an IB assignment). Clearly, $a(C) = \mathcal{A}$. Now, applying τ at the next unexpanded node u selects a hypercube $H_i^{u^d}$ at u, and let $\mathcal{B} = \mathcal{A} \cup H_i^{u^d}$. If this occurs, then clearly $(u, d) \in \mathcal{A}$ for some $d \in D_u$.

The following assumes that u is indeed the next unexpanded node in the ordering. (This is not always the case, since it is possible that there are several nodes w_i in the ordering coming before u, which are not "active," i.e., the IB condition happens to hold at w_i with respect to $\mathcal{A}$ before they are expanded, and thus τ will never expand them. We will relax this assumption presently.) In the cost-sharing algorithm we let the next state (call it C') be C with all edges whose source is N^{u^d} removed, and all the arcs whose sink is $H_i^{u^d}$, including $ESC_i^{u^d}$, added. Thus, we have that $a(C') = \mathcal{B}$, and applying the σ operator is equivalent to applying τ.

Suppose, however, that there are several nodes W in the ordering before u that are in span($\mathcal{A}$) but where the IB condition holds with respect to $\mathcal{A}$. In this case, we apply σ at each of the $w \in W$, and since the IB condition holds at w, then at least one IB hypercube $H_i^{w^e}$ is subsumed by $\mathcal{A}$. Selecting these subsumed hypercubes for each $w \in W$ assures us that at the set of edges C_f at end of the series of expansions still obeys $a(C_f) = \mathcal{A}$. After this series of expansions, we can apply the expansion operator to u as above, and thus this series of expansions with the expansion of u can simulate any application of τ. (When $\mathcal{A}$ is and IB assignment, it may be necessary to apply σ several more times, until the only edges are self-cost edges. This can clearly be done without changing the assignment, as discussed above.) Thus, for any assignment $\mathcal{A}$ that is τ-reachable from the evidence, there exists a set of edges C that is reachable via σ, such that $a(C) = \mathcal{A}$, and such that all nodes in span($\mathcal{A}$) have been expanded by σ [thus if $(v, d) \in \mathcal{A}$ there is some self-cost edge $ESC_i^{v^d} \in C$, and since C contains only self-cost edges, it is a minimal cut]. Now, since all consistent IB assignments maximal with respect to subsumption (among all IB assignments that are subsumed by the evidence) are τ-reachable from the evidence, the theorem follows immediately. □

The algorithm also diverges from [7] in that the cost of an edge is the cost of its source node N^{v^d} divided by the out degree of v in B, rather than the out degree of N^{v^d} in W. That is permissible because of all the nodes $H_i^{u^{d'}}$ (where u is a child of v in B, and for any value d' and hypercube index i) only one appears in any AND DAG, and thus the cost is only shared among at most out-degree(v) edges. This argument is similar to the one presented in the conclusion of [7].

Theorem 3: The algorithm is correct. For example, it will always return with $a(C)$ being the next-probable IB assignment subsumed by the evidence $\mathcal{E}$.

Proof: All possible solution states are reachable from the initial state (Theorem 2), and the heuristic evaluation function

is admissible (Theorem 1 states that the function is exact for complete AND DAG's, and it is nonincreasing by a property shown in [7]). Thus, since we have a best-first heuristic search with all possible solutions reachable and an admissible heuristic, the algorithm will return when it finds a lowest cost consistent self-cost-only (minimal) cut C of $W(B, \mathcal{E})$. $a(C)$ is an IB assignment subsumed by the evidence (Lemma 2), and since probability is a strictly decreasing function of cost, it is of maximum probability. □

Note that it is possible for an edge e with source v to exist in C, where actually the IB condition holds at v with respect to $a(C)$. In this case, we still need to expand e, but it does not matter which next state is selected, they all result in the same assignment (counting only consistent assignments). In the actual implementation, to prevent the creation of too many duplicate states, the first one of these that is consistent is pushed into the agenda, and all the others are discarded. While the selection of edges may affect the search in that the cost estimation may be different, it cannot affect the final result in this case.

4) Handling Overlaps with Cost-Sharing: Essentially, the same schemes used for the current-cost search algorithms can be used here. As it turns out, however, if we add an assignment to some node that is lower numbered than all the expanded nodes, there is a problem as the heuristic will no longer be admissible. There *may* be a way to overcome this problem, but the easy way out is never to reach this case in the first place. By using the first method in Section III-B, i.e., selecting a node proceeding from the root down, the resulting split assignments are IB and we can use the exact value as a heuristic value, thus avoiding the problem. In fact, since inclusion–exclusion turns out to be benevolent in practice (see Section V), this issue is ignored in the implementation.

IV. IB Assignment Search as Integer Programming

In [35]–[37], a method of converting the complete MAP problem to a linear inequality system was shown. In [41] a similar method that converts the problem of finding the IB MAP to a linear inequality system was shown. We begin by reviewing the reduction, modify it to decrease the number of equations, and discuss the further changes necessary to make the system find the next-best IB assignments without overlaps.

A. Reduction of IB-MAP's: Review

The linear system of inequalities has a variable for each maximal IB hypercube. The inequality generation is reviewed below. A belief network is denoted by $B = (G, \mathcal{D})$, where G is the underlying graph and $\mathcal{D}$ the distribution. We usually omit reference to $\mathcal{D}$ and assume that all discussion is with respect to the same arbitrary distribution. For each node v and value in D_v (the domain of v), there is a set of k_{v^d} maximal IB hypercubes based on v (where $d \in D_v$). We denote that set by $\mathcal{H}^{v^d}$, and assume some indexing on the set. Member j of $\mathcal{H}^{v^d}$ is denoted $\mathcal{H}_j^{v^d}$, with $k_{v^d} \geq j \geq 1$.

A system of inequalities L is a triple (V, I, c), where V is a set of variables, I is a set of inequalities, and c is an assignment cost function.

Definition 1: From the belief network B and the evidence $\mathcal{E}$, we construct a system of inequalities $L = L_{IB}(B, \mathcal{E})$ as follows.

1) V is a set of variables $h_i^{v^d}$, indexed by the set of all evidentially supported maximal hypercubes $H_{\mathcal{E}}$ (the set of hypercubes H such that if H is based on w, then w is evidentially supported). Thus, $V = \{h_i^{v^d} | H_i^{v^d} \in H_{\mathcal{E}}\}$. [The superscript v^d states that node v is assigned value d by the hypercube (which is based on v), and the subscript i states that this is the ith hypercube among the hypercubes based on v that assign the value d to v.]
2) $c(h_i^{v^d}, 1) = -\log(P(H_i^{v^d}))$, and $c(h_i^{v^d}, 0) = 0$.
3) I is the following collection of inequalities:
   a) For each set of two inconsistent hypercubes $H_i^{v^d}, H_j^{w^{d'}} \in \mathcal{H}_{\mathcal{E}}$, such that $w \neq v$
   $$h_i^{v^d} + h_j^{w^{d'}} \leq 1. \tag{2}$$
   b) For each evidentially supported node v
   $$\sum_{d \in D_v} \sum_{i=1}^{k_{v^d}} h_i^{v^d} \leq 1. \tag{3}$$
   c) For each pair of nodes w, v such that $v \in \text{parents}(w)$, and for each value $d \in D_v$
   $$\sum_{i=1}^{k_{v^d}} h_i^{v^d} - \sum_{d' \in D_w \wedge (v,d) \in H_j^{w^{d'}}} h_i^{w^{d'}} \geq 0. \tag{4}$$
   d) For each $(v, d) \in \mathcal{E}$
   $$\sum_{i=1}^{k_{v^d}} h_i^{v^d} \geq 1. \tag{5}$$

The intuition behind these inequalities is as follows: inequalities of type **a)** enforce consistency of the solution. Type **b)** inequalities enforce selection of at most a single hypercube based on each node. Type **c)** inequalities enforce the IB constraint, i.e., at least one hypercube based on v must be selected if v is assigned. Type **d)** inequalities introduce the evidence.

Following [37], we define an assignment s for the variables of L as a function from V to $\mathcal{R}$. Furthermore

1) if the range of s is in $\{0, 1\}$ then s is a 0–1 assignment;
2) if s satisfies all the inequalities of types **a)**–**d)** then s is a solution for L;
3) if solution s for L is a 0–1 assignment, then it is a 0–1 solution for L.

The objective function to optimize is

$$\Theta_{L_{\text{IB}}}(s) = -\sum_{h_i^{v^d}} s(h_i^{v^d}) \log(P(H_i^{v^d})). \tag{6}$$

In [41] it was shown that an optimal 0–1 solution to the system of inequalities induces an IB MAP on the original belief network.

Note that the number of type **a)** inequalities is potentially quadratic in the maximum number of hypercubes per node

and in the out-degree of the belief network, and linear in the number of nodes. The number of type **b)** inequalities is equal to the number of supported nodes. The number of type **c)** inequalities is linear in the number of arcs between supported nodes and in the cardinality of the domain size of the nodes. The number of type **d)** inequalities is equal to the number of evidence nodes.

Clearly, the bottleneck is in the number of type **a)** inequalities. That number can be greatly reduced by collecting a larger number of mutually inconsistent hypercubes into the same equation, and eliminating several equations. Note that if node v has children x and y, then any hypercube based on x assigning some value to v is inconsistent with any hypercube based on y assigning some other value to v. And since only one hypercube based on x may be in the solution (because of the type **b)** inequalities), then all of the above hypercubes may be piled into the same equation, thus standing for a quadratic number of equations. All in all, the number of modified type **a)** inequalities is now linear in the number of (parent, first-child, second-child) triples and in the maximum cardinality of the domain of the nodes. Additionally, all type **d)** inequalities can be converted to equalities, and the type **b)** inequalities for the evidence nodes can be dropped. All variables standing for hypercubes inconsistent with the evidence can also be erased. To cater for a query node q, it is necessary to add an equation forcing an assignment to some hypercube variable based on q. This is like a type **b)** inequality, except that we need to use equality there. We call this a type **e)** equation. We end up with I, the set of inequalities as follows:

a) For each triple of nodes (v, x, y) s.t. $x \neq y$ and $v \in \text{parents}(x) \cap \text{parents}(y)$, and for each $d \in D_v$:

$$\sum_{(v,d)\in H_i^{x^e},\, e\in D_x} h_i^{x^e} + \sum_{(v,d')\in H_j^{y^f},\, f\in D_y,\, d\neq d'} h_j^{y^f} \leq 1. \quad (7)$$

b) For each evidentially supported node v that is not a query node and is not in $\text{span}(\mathcal{E})$

$$\sum_{d\in D_v} \sum_{i=1}^{k_{vd}} h_i^{v^d} \leq 1. \quad (8)$$

c) For each pair of nodes w, v such that $v \in \text{parents}(w)$, and for each value $d \in D_v$

$$\sum_{i=1}^{k_{vd}} h_i^{v^d} - \sum_{d'\in D_w \wedge (v,d)\in H_j^{w^{d'}}} h_i^{w^{d'}} \geq 0. \quad (9)$$

d) For each $(v, d) \in \mathcal{E}$

$$\sum_{i=1}^{k_{vd}} h_i^{v^d} = 1. \quad (10)$$

e) For each query node q

$$\sum_{d\in D_q} \sum_{i=1}^{k_{qd}} h_i^{q^d} = 1. \quad (11)$$

If the optimal solution of the system happens to be 0–1, we have found the IB MAP. Otherwise, we need to branch: select a variable h which is assigned a non 0,1 value, and create two sets of inequalities (subproblems), one with $h = 1$ and the other with $h = 0$. Each of these now needs to be solved for an optimal 0–1 solution, as in [36]. This branch and bound algorithm may have to solve an exponential number of systems, but in practice that is not the case. Additionally, the subproblems are always smaller in number of equations or number of variables.

To create a subproblem, h is *clamped* to some value in $\{0, 1\}$. The equations can now be further simplified: a variable clamped to 0 can be removed from the system. For a variable clamped to 1, the following reductions take place: Find the type **{b)** inequality, the type **{d)** equation (if any), and all the type **{a)** inequalities, in which h appears. In each such inequality clamp all the other variables to 0 (removing them from the system), and delete the inequality. After doing the above, check to see if any inequality contains only one variable, and if so clamp it accordingly. For example, if a type **{d)** equation has only one variable, clamp it to 1. Repeat these operations until no more reductions can be made.

Once the optimal 0–1 solution s is found, we need to add an equation prohibiting that solution, in order to prepare the generator for resumption (in order to get the next-best IB assignments later on), before returning the assignment induced by s. In [44], the following equation was used:

$$\sum_{v\in S} \sum_{\{H_i^{v^d} | (v,d)\in\mathcal{A}\}} h_i^{v^d} > |S|.$$

The above equation rules out any solution which induces an assignment $\mathcal{B}$ which assign $|S| = |\text{span}(\mathcal{A})|$ variables the same values as in $\mathcal{A}$. However, in order for $\mathcal{B}$ to be subsumed by $\mathcal{A}$, it must be the case that $|\text{span}(\mathcal{A})|$ are assigned the same values, the equation disallows any assignment subsumed by $\mathcal{A}$, as well as $\mathcal{A}$ itself.

B. Preventing Overlapped Solutions

After finding a solution to the optimization problem, we always add an equation disallowing that solution (and ones subsumed by that solution). It turns out that linear programs are sufficiently powerful to inhibit overlapping solutions altogether. The idea is to enforce the constraint that a new IB assignment be inconsistent with all previously generated IB assignments.

Theorem 4: Let $\mathcal{A}$ be an IB assignment corresponding to a solution to $L = L_{\text{IB}}(B, \mathcal{E}, Q)$. Then adding the equation

$$\sum_{v\in S-E} \sum_{\{H_i^{v^d} | (v,d)\notin\mathcal{A}\}} h_i^{v^d} \geq 1 \quad (12)$$

to L, disallows any solution where the induced assignment $\mathcal{B}$ overlaps with $\mathcal{A}$.

Proof: Given an IB assignment $\mathcal{A}$, it is evident, by inspection, that (12) enforces at least one hypercube to be chosen that assigns some variable in $S-E$ differently than in $\mathcal{A}$. Thus, for any new solution to the system, and induced assignment $\mathcal{B}$ must have at least one variable assigned differently from $\mathcal{A}$

	Min	Max	Avg	Med
States	4096	221184	63922	62208

Fig. 5. Ten node networks. States indicate number of complete assignments per network.

(and thus be inconsistent with $\mathcal{A}$). Consequently, this equation insures that no solution of the system will induce an IB assignment that overlaps with $\mathcal{A}$. □

V. Experimental Results

Empirical validation of our results for actually computing marginal probabilities was pursued in companion papers [38], [44]. Here we focus on the issue of mass collection, search efficiency, and the overlap problem.

As mentioned earlier, because they are partial assignments, each IB assignment is expected to gather more mass per assignment than the respective complete assignments. We studied this mass accumulation for IB MAP's by taking assignments one at a time in order of probability. By plotting the percentage of mass accumulated versus the number of assignments used, we can get a fair idea of mass accumulation rate. In particular, we extracted the top 50 IB assignments per problem instance from 50 randomly generated networks, each having ten nodes.[6] Fig. 5 gives a brief summary of our networks.

Looking at Fig. 6, plotting average mass versus number of assignments for skewed networks conditional probabilities generated uniformly form the range $[0, 0.1]$ and $[0.9, 1]$, we can see that mass is accumulated fairly quickly and is contained in a small set of assignments as we expected. After five IB assignments, we have already obtained (on average) roughly 85% of the total mass. Significantly greater mass is collected in a small number of IB assignments than the results of [16] predict for complete assignments. The latter are plotted, for a set of ten independent *binary* variables, where the prior probability of the states are 0.95 and 0.05. IB assignments collect more mass, despite the fact that for our IB multivalued variables, the additional states tend to disperse the probability mass.

We believe that this is due mainly to the moderate local independence in the networks. An assignment containing k uninstantiated variables is equivalent to 2^k complete instantiations (though not that much more mass, as many of the participating assignments will have orders of magnitude less mass). The difference in mass between complete assignments and IB assignments is expected to grow with network size, as corroborated by experiments such as the above for networks of sizes 5 and 15 (not shown). Furthermore, even if application networks contain significantly less local independence, an IB assignment will always contain at least as much mass as the respective complete assignment. Additionally, finding the high-probability IB assignments is no harder than finding high-probability complete assignments, and is typically somewhat easier. As noted above, overlap problems can be avoided *ab initio*, if desired. Thus, at worst, we lose nothing by using IB assignments in place of complete assignments.

We turn to the cost sharing and linear programming approaches. Timing results show that our constraints approach can solve for the IB MAP in networks of up to 2000 nodes (all our timing results are for a Sparc 2 or equivalent machine). Fig. 7 compares the timing results of the linear programming approach on 50 networks each consisting of 2000 nodes, with the current cost and shared cost methods. In this experiment evidence was generated for 1 to 3 (uniformly distributed) evidence nodes selected at random, and evidence state selected uniformly for each node. Average network size after removing redundant nodes was about 50. For these problem instances, cost sharing usually did much better than ILP, which in turn did much better than current cost. However, we expect that on larger diagram sizes, ILP will do better than cost sharing, which we intend to confirm in the near future. For the most part, we found the ILP solutions relatively quickly. We would like to note though, that our package for solving integer linear programs was crudely constructed by the authors without the additional optimizations such as sparse systems, etc. Furthermore, much of our computational process is naturally parallelizable and should benefit immensely from techniques such as parallel simplex [20] and parallel ILP [1], [3].

After running the generators to find the IB MAP, we turned to actually approximating the probabilities. In [44] the runtime of the cost-sharing scheme compared favorably with stochastic simulation (as implemented in CaBeN [11]), and our runtime was better than the stochastic simulation for 100 node networks. More importantly, cost-sharing bounds were at least as good as CaBeN's (numerically). Additionally, our search-based schemes are *deterministic* approximation algorithms, and thus our error bounds are *certain* bounds, unlike stochastic simulation, where these are only error estimates. For further details about the comparison, see [44].

We now focus on the overlap problem. We stated earlier that just using inclusion–exclusion turned out to work reasonably, which point we justify empirically. First, note that the approximation algorithm is anytime, and we would like to preserve that property in evaluating inclusion–exclusion. A recursive variant of the inclusion–exclusion formula is used, as follows. The formula looks at two events, one consisting of the new event (IB assignment), E_{m+1}, the second consisting of the union of all previous events

$$P(\cup_{1\le i\le m+1}E_i) = P(E_{m+1}) + P(\cup_{1\le i\le m}E_i) - P(E_{m+1}\cap\cup_{1\le i\le m}E_i). \quad (13)$$

In this equation, the first term is immediately available, while the second has already been computed, when the pre-

[6] Unless specified otherwise, networks were generated as follows. The in-degree was uniformly distributed between 0 and 3, by selecting first the number of parents, and adding arcs to nodes already processed. Each node has two to four states, selected randomly with uniform distribution. In order to get some local independence, we selected for each child node-state a parent on which it is always dependent, and then each hypercube was split along an additional dimension (made dependent on an additional parent) with probability 0.5, in which case the smaller hypercubes were again split with probability 0.5, etc., (see [41] for details). The number of hypercubes generated by this method is on the average more than for pure OR nodes, but less than for a distribution with conditional probabilities generated uniformly over the range $[0, 1]$.

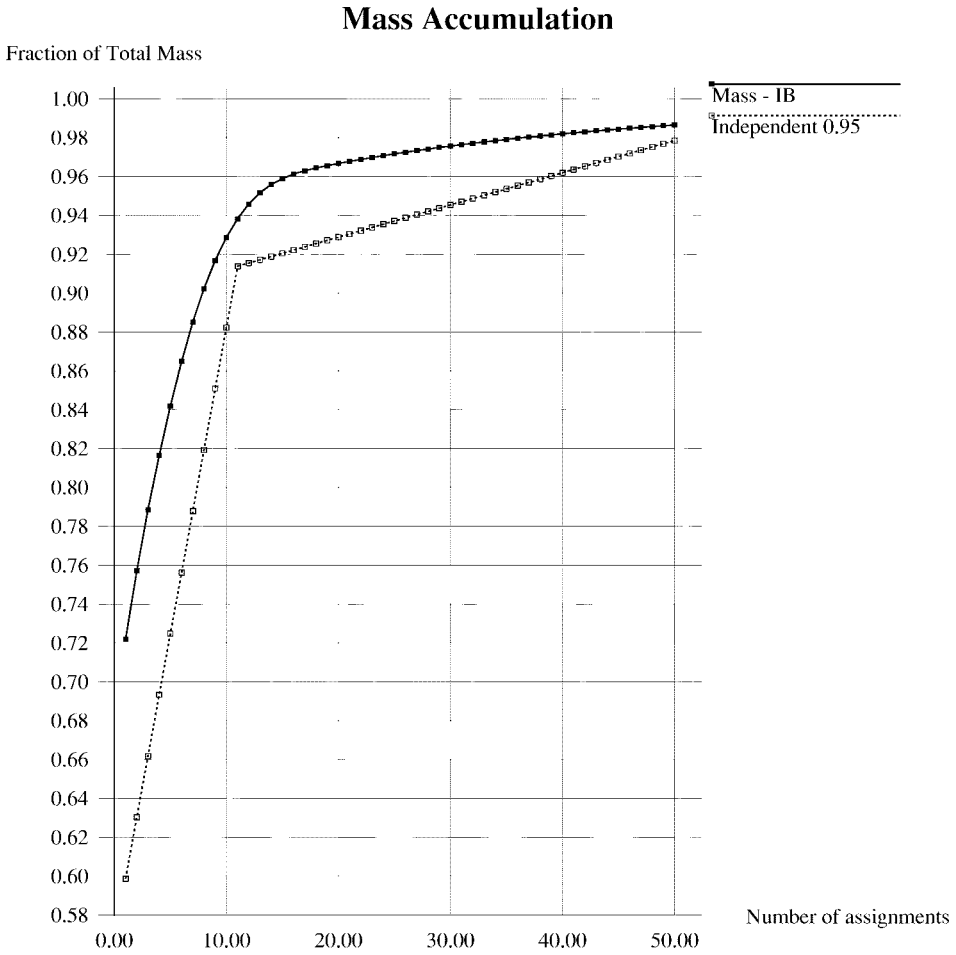

Fig. 6. Mass accumulation for ten node networks.

vious IB assignment (E_m) was considered. The last term is equivalent to the probability of the union of m events

$$P(E_{m+1} \cap \cup_{1 \le i \le m} E_i) = P[\cup_{1 \le i \le m} (E_{m+1} \cap E_i)] \quad (14)$$

which is computed by a recursive call to the inclusion–exclusion evaluator. Recursive computation levels are somewhat different, in that both compound terms in (13) need to be evaluated. Recursive evaluations of (13) therefore requires *two* recursive calls, which give rise to a runtime exponential in the recursion depth (the latter being equal to the overlap set size in the worst case). To decrease the number of calls to inclusion–exclusion, a test is first made whether E_{m+1} is subsumed by any of the previous events (in which case it is discarded). Events not overlapping E_{m+1} are ignored in the recursive call, and events with probability very near zero (10^{-30} in the experiment) are discarded.

We experimented on 20 randomly generated networks of each size: 50 and 100 nodes. Noting that our previously used network generation method did not provide a sufficient amount of assignment overlap, we did an extra pass on each node with two or more parents, and (with probability 0.5) converted it into a noisy OR.[7] Additionally, conditional probabilities were skewed, and picked uniformly from the ranges [0, 0.1] and [0.9, 1]. Average network sizes after removing redundant nodes was 21 and 24, respectively, (including intermediate nodes).

For each network, we produced 100 IB assignments by using cost-sharing heuristic search. (This is much more than usually required for approximation. For example, in the comparison to CaBeN, we used 15 IB assignments.) Runtime was not significantly different from networks generated without the noisy OR's in previous experiments. Overlap set sizes were anywhere from one to 41 assignments (that is, k from 0 to 40). We expected the runtime for inclusion–exclusion to be prohibitive in many cases. For example, if in the above large overlap set, every subset had an overlap, the recursion depth would reach 40, and the evaluator would not finish its computation in many years. Expecting the worst, we have limited recursion depth artificially to 5 (in which case, the overlapped mass is computed erroneously, but a bound on the additional error can be easily found). In about half the problem instances, overlap sets were small (under 5). In most remaining cases, even with overlap sets as large as 41, the recursion depth of 5 was not reached, and the incremental inclusion–exclusion evaluator ran in time significantly less than the runtime of the IB assignment search component. In three problem instances out of 40, recursion depth 5 was

[7] Since representing noisy OR with IB hypercubes is not efficient, we actually represented a noisy OR by a pure OR and additional intermediate nodes, for which the IB hypercube representation is efficient. This is a notational variant, with no effect on the network distribution.

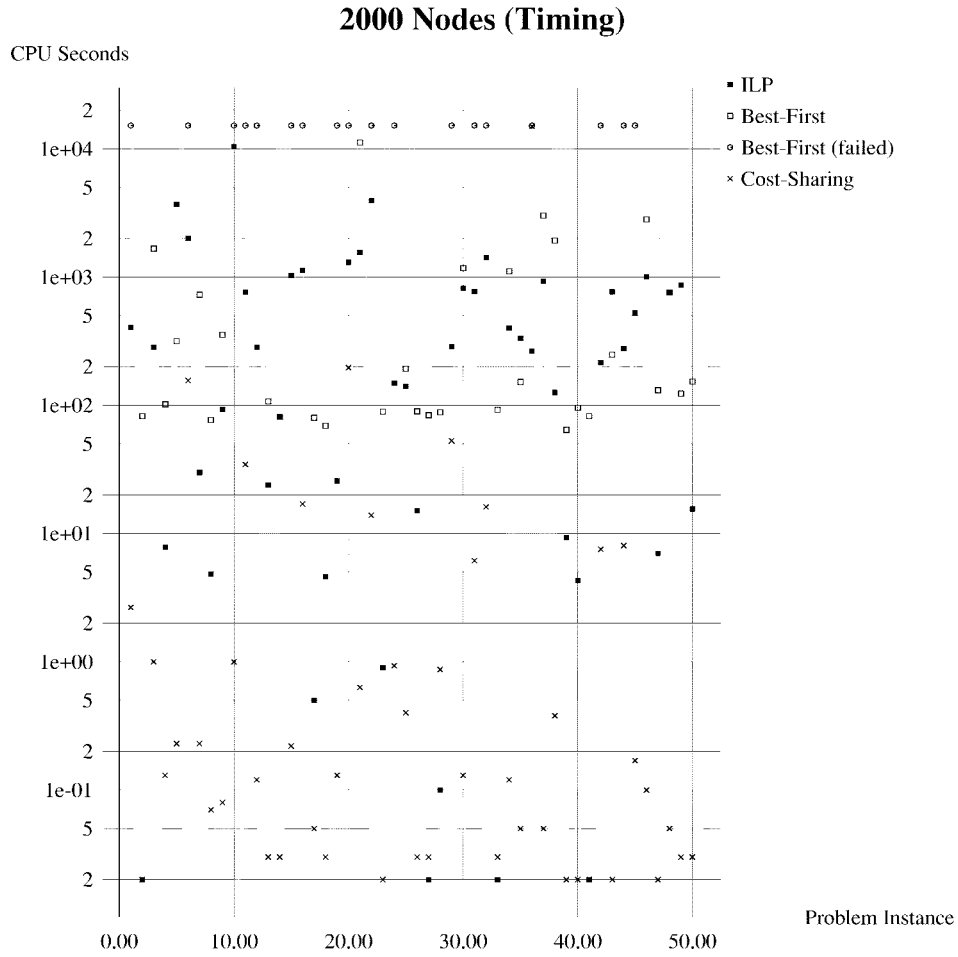

Fig. 7. 2000 node networks.

reached multiple times. This occurred even for relatively small overlap sets (starting at size 6).

Two conclusions follow from the latter experiment. 1) In many cases, it is *not* worthwhile to use any elaborate scheme to handle overlap (such as splitting assignments, or even avoiding overlaps in the ILP generator). 2) Overlap set size is not a good indicator of inclusion–exclusion runtime in this special case. Granted, it provides an upper bound on complexity, but a very poor one in practice. A better idea is to run the recursive inclusion–exclusion evaluator, and to resort to other methods (assignment splitting, reinitializing hypercubes to avoid overlaps altogether, etc.) only if the recursion depth exceeds a certain value, *and* the probability of the overlap at the maximum depth is significant compared to the rest of the probabilities and the current error bound.

VI. Discussion

We begin by discussing related work, and then suggest that converting complete assignments to IB assignments may make this scheme applicable to algorithms appearing in related work.

A. Related Work

The work on term computation [13] and related papers are extremely relevant to this paper. The skewness assumption made there, or a weaker version of it, also make our method applicable. In a sense, these methods complement each other, and it should be interesting to see whether IB assignments (or at least maximal IB hypercubes) can be incorporated into a term computation scheme.

This paper enumerates high probability IB assignments using a backward search from the evidence. Reference [32] also enumerates high probability assignments, but using a top down (forward) search. Backward constraints are introduced through conflicts. It is clear that the method is efficient for the example domain (circuit fault analysis), but it is less than certain whether other domains would obey the extreme probability assumption that makes this work. If that assumption does not hold, it may turn out that backward search is still better. On the other hand, it may still be possible to take advantage of IB hypercubes even in the forward search approach, as shown later on in this section. Note that among the algorithms presented here, the current probability heuristic ignores forward constraints, while the shared-cost heuristic does employ some form of forward reasoning by incorporating the costs from the top-down initialization. The ILP method uses global constraints that also include the top-down constraints, but what role the top-down constraints play in the search is unclear.

In [44], the deterministic algorithms were compared to stochastic simulation. Another class of randomized algorithms

computes the MAP (belief revision) rather than marginal probabilities (belief updating). For example, in [18] simulated annealing is used. It is not clear, however, how one might use it to enumerate a number of high-probability assignments.

A genetic algorithm for finding the MAP is presented in [34], with an experiment showing that the total probability mass of the population rises during the search and converges on some value. The authors do not say whether assignments in the population include duplicates, however, and make no mention of the possibility of approximating marginal probabilities with that population. It seems likely that if the algorithm can be modified to handle IB assignments, then the fact that a whole *population* is used, rather than a single candidate, may provide a ready source of near-optimal IB assignments. Of course, we are not guaranteed to get IB assignments in decreasing order of probability, so slightly different methods would have to be used to approximate the marginal probabilities. In addition, this method will change the algorithm from a deterministic approximation algorithm into a randomized approximation algorithm, with all that entails.

Finally, it should be possible to modify the algorithms presented in this paper to work on GIB assignments and δ-IB assignments, where an even greater probability mass is packed into an assignment [41], [43]. Some theoretical issues will have to be dealt with before we can do that, however.

B. Converting Complete Assignments to IB Assignments

Given a complete assignment (i.e., one that assigns values to all the variables), can we generate a compatible IB assignment from it? The answer to that is vacuously "yes," since a complete assignment is also IB. The interesting question is whether we can drop as many variables from the assignment as possible, such that the resulting assignment is still IB. That is, can we efficiently compute a compatible IB assignment that is maximal with respect to subsumption. A variant of this is to find a compatible IB assignment that has maximum probability (for strictly positively distributed networks, such an assignment is always also maximal with respect to subsumption).

It is clear that the maximal IB assignment compatible with a complete assignment is not unique, and that several of these maximal IB assignments may be much better than others (i.e., have higher probability). It thus appears better to find the highest probability IB assignment compatible with a complete assignment. Nevertheless, *any* maximal IB assignment with several variables unassigned is potentially better than a complete assignment. It is easy to find a maximal with respect to subsumption IB assignment, and we outline below one polynomial-time algorithm. Whether a maximum probability IB assignment compatible with a complete assignment can be found in polynomial time is an open question. The problem *seems* easier than finding the IB MAP *without* forcing compatibility with a certain complete assignment, but our current intuitions are that it is still NP-hard.

If we assume a constant in-degree, then the following algorithm modifies complete assignment $\mathcal{A}$ to a maximal with respect to subsumption IB assignment that subsumes the evidence and is consistent with the original $\mathcal{A}$ in polynomial time.

1) Make all nonevidentially supported nodes unassigned in $\mathcal{A}$.
2) Repeatedly, try to make nodes unassigned in $\mathcal{A}$, until no further modifications are possible, according to the following rule:
   - If v is not an evidence node, and for each child of u of v, either of the following conditions hold, then make v unassigned in $\mathcal{A}$:
     a) u is not assigned in $\mathcal{A}$.
     b) There exists a maximal hypercube H based on u, such that $H \subseteq \mathcal{A}$, and $v \notin$ span(H).

The resulting assignment $\mathcal{A}$ is consistent with the given assignment, by construction. $\mathcal{A}$ is also maximal with respect to subsumption (while still assigning a value to each evidence node), since if the assignment to any node v can be removed, then the IB condition must hold at each of its assigned children after it is removed. Thus there must be a maximal IB hypercube that does not assign a value to v, and is a subset of $\mathcal{A}$, for every child of v in span($\mathcal{A}$). That is exactly the condition checked in the loop. As to the runtime of the algorithm, since in every pass (except for the last pass) at least one node is removed from the assignment, then at most n passes are made. Each pass looks at every node at most a constant number of times, and thus the algorithm terminates in time $O(n^2)$. [Selecting hypercubes, checking consistency, membership, etc., are $O(1)$, given reasonable indexing and the constant in-degree assumption.] In fact, it is possible to decrease the runtime significantly [29], but this issue is beyond the scope of this paper.

Finding maximal IB assignments consistent with a complete assignment is potentially useful in at least two types of algorithms for approximating marginal probabilities. In Poole's top-down algorithm, (nearly) complete assignments are scored (deterministically) to approximate marginal probabilities. If each complete assignment is converted to a maximal IB assignment before scoring, each assignment scored will contain more mass, and thus the approximation method will converge faster. Another application for converting complete assignment to IB assignments is in using genetic algorithms to approximate marginals. For example, use the same search algorithm as in GALGO [34], but convert complete assignments to maximal IB assignments, and score them deterministically. The "fitness" value of each complete assignment is the probability of the maximal IB assignment generated from it. We intend to pursue these issues in forthcoming papers.

VII. Summary

Computing marginal (prior or posterior) probabilities in belief networks is hard. Approximation schemes are thus of interest. Several deterministic approximation schemes enumerate terms, or assignments to sets of variables, of high probability, such that a relatively small number of them contain most of the probability mass. This allows for an anytime approximation algorithm, whereby the approximation improves as

a larger number of terms is collected. IB assignments are partial assignments that take advantage of local independencies not represented by the topology of the network, to reduce the number of assigned variables, and hence the probability mass in each assignment. Empirical results show that this mass accumulation is indeed much faster than predicted for complete assignments. In any case, there is nothing to lose by using IB assignments: finding a most probable IB assignment is typically *easier* than finding the most-probable complete assignment, and an IB assignment will always have at least as much mass as the corresponding complete assignment.

What remains to be done is to come up with these IB assignments in a decreasing order of probability. This is also a hard problem in general, unfortunately. The factors contributing to complexity, however, are not maximum clique size or loop cutset, but rather the number of hypercubes. Under probability skewness assumptions, the search for high probability IB assignments is typically more efficient, and the resulting approximation (collecting a small number of assignments) is better.

Three algorithms for approximating marginal algorithms are presented: a modification of a node-based best-first search algorithm for finding the IB MAP, an edge-based best-first search algorithm with a cost-sharing heuristic, and an algorithm based on linear systems of inequalities. Experimental results show that using the cost-sharing heuristic improves performance of the best-first search algorithm by more than one order of magnitude. The problem of assignment overlaps, which is another source of possible exponential runtime, turns our to be benevolent in practice, when using the recursive version of inclusion–exclusion with the cost-sharing generator. In the ILP version of the algorithm, overlaps can be easily avoided altogether.

REFERENCES

[1] P. D. Bailor and W. D. Seward, "A distributed computer algorithm for solving integer linear programming problems," in *Proc. 4th Conf. Hypercubes, Concurrent Computers Applications*, 1989, pp. 1083–1088.

[2] T. O. Binford, T. S. Levitt, and W. B. Mann, "Bayesian inference in model-based machine vision," in *Proc. AAAI, Uncertainty AI Workshop*, July 1987.

[3] R. L. Boehning, R. M. Butler, and B. E. Gillett, "A parallel integer linear programming algorithm," *Eur. J. Oper. Res.*, vol. 34, pp. 393–398, 1988.

[4] R. R. Bouckaert, "A stratified simulation scheme for inference in Bayesian belief networks," in *Uncertainty Artificial Intelligence, Proc. 10th Conf.*, July 1994, pp. 110–117.

[5] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller, "Context-specific independence in Bayesian networks," in *Uncertainty Artificial Intelligence, Proc. 12th Conf.*, Aug. 1996, pp. 115–123.

[6] E. Charniak and R. P. Goldman, "A Bayesian model of plan recognition," *Artif. Intell. J.*, 1994.

[7] E. Charniak and S. Husain, "A new admissible heuristic for minimal-cost proofs," in *Proc. AAAI Conf.*, 1991, pp. 446–451.

[8] E. Charniak and S. E. Shimony, "Cost-based abduction and MAP explanation," *Artif. Intell. J.*, vol. 66, no. 2, pp. 345–374, 1994.

[9] G. F. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks," *Artif. Intell. J*, vol. 42, nos. 2/3, pp. 393–405, 1990.

[10] ——, "NESTOR: A computer-based medical diagnosis aid that integrates causal and probabilistic knowledge," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1984.

[11] S. B. Cousins, W. Chen, and M. E. Frisse, "CABeN: A collection of algorithms for belief networks," Tech. Rep. WUCS-91-25, Dept. Comput. Sci., Washington Univ., St. Louis, MO, 1991.

[12] P. Dagum and M. Luby, "Approximating probabilistic inference in Bayesian belief networks is NP-hard," *Artif. Intell.*, vol. 60, no. 1, pp. 141–153, 1993.

[13] B. D'Ambrosio, "Incremental probabilistic inference," in *Uncertainty Artificial Intelligence, Proc. 9th Conf.*, July 1993.

[14] T. Dean and K. Kanazawa, "A model for reasoning about persistence and causation," *Comput. Intell.*, vol. 5, no. 3, pp. 142–150, 1991.

[15] F. J. Diez, "Local conditioning in Bayesian networks," Tech. Rep. R-181, Cognitive Syst. Lab., Dept. Comput. Sci., Univ. Calif., Los Angeles, July 1992.

[16] M. Druzdzel, "Some properties of joint probability distributions," in *Uncertainty Artif. Intell., Proc. 10th Conf.*, July 1994, pp. 187–194.

[17] R. Fung and B. Del Favero, "Backward simulation in Bayesian networks," in *Uncertainty Artificial Intelligence, Proc. 10th Conf.*, July 1994, pp. 227–234.

[18] S. Geeman and D. Geeman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 6, pp. 721–741, 1984.

[19] D. Geiger and D. Heckerman, "Advances in probabilistic reasoning," in *Proc. 7th Conf. Uncertainty Artificial Intelligence*, 1991.

[20] B. E. Gillett, *Introduction to Operations Research: A Computer-Oriented Algorithmic Approach.* New York: McGraw Hill, 1976.

[21] M. Henrion, "Propagating uncertainty in Bayesian networks by probabilistic logic sampling," in *Proc. Uncertainty Artificial Intelligence*, 1988, pp. 149–163.

[22] E. J. Horvitz, H. J. Suermondt, and G. F. Cooper, "Bounded conditioning: Flexible inference for decisions under scarce resources," in *5th Workshop Uncertainty Artificial Intelligence*, Aug. 1989.

[23] F. V. Jensen, K. G. Olsen, and S. K. Andersen, "An algebra of Bayesian belief universes for knowledge-based systems," *Networks*, vol. 20, pp. 637–660, 1990.

[24] J. H. Kim and J. Pearl, "A computation model for causal and diagnostic reasoning in inference systems," in *Proc. 6th Int. Joint Conf. Artificial Intelligence*, 1983.

[25] U. Kjaerulff, "Reduction of computational complexity in Bayesian networks through removal of weak dependencies," in *Uncertainty Artificial Intelligence, Proc. 10th Conf.*, July 1994, pp. 374–382.

[26] S. L. Lauritzen and D. J. Speigelhalter, "Local computations with probabilities on graphical structures and their applications to expert systems," *J. R. Stat. Soc.*, vol. 50, pp. 157–224, 1988.

[27] Z. Li and B. D'Ambrosio, "An efficient approach to probabilistic inference in belief nets," in *Proc. Annu. Can. AI Conf.*, May 1992.

[28] N. Linial and N. Nisan, "Approximate inclusion–exclusion," *Combinatorica*, vol. 10, pp. 349–365, 1990.

[29] A. A. Melkman and S. E. Shimony, "Algorithms for parsimonious complete sets in directed graphs," *Inf. Process. Lett.*, vol. 59, pp. 335–339, 1996.

[30] R. E. Neapolitan, *Probabilistic Reasoning in Expert Systems.* New York: Wiley, 1990, ch. 8.

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

[32] D. Poole, "The use of conflicts in searching Bayesian networks," in *Uncertainty Artificial Intelligence, Proc. 9th Conf.*, July 1993.

[33] H. Pople, "The formation of composite hypotheses in diagnostic problem solving: An exercise in synthetic reasoning," in *Proc. IJCAI 5*, 1977, pp. 1030–1037.

[34] C. Rojas-Guzman and M. A. Kramer, "GALGO: A genetic algorithm decision support tool for complex uncertain systems modeled with Bayesian belief networks," in *Uncertainty Artificial Intelligence, Proc. 9th Conf.*, 1993.

[35] E. Santos, Jr., "Cost-based abduction, linear constraint satisfaction, and alternative explanations," in *Proc. AAAI Workshop on Abduction*, 1991.

[36] ——, "On the generation of alternative explanations with implications for belief revision," in *Proc. 7th Conf. Uncertainty Artificial Intelligence*, 1991, pp. 339–347.

[37] ——, "A linear constraint satisfaction approach to cost-based abduction," *Artif. Intell. J.*, vol. 65, no. 1, pp. 1–27, 1994.

[38] E. Santos, Jr., S. E. Shimony, and E. Williams, "Sample-and-accumulate algorithms for belief updating in Bayes networks," in *Uncertainty Artificial Intelligence, Proc. 12th Conf.*, Aug. 1996, pp. 477–484.

[39] R. D. Shachter, "Evaluating influence diagrams," *Oper. Res.*, vol. 34, no. 6, pp. 871–882, 1986.

[40] S. E. Shimony, "A probabilistic framework for explanation," Tech. Rep. CS-91-57, Brown Univ., Providence, RI, 1991.

[41] ——, "The role of relevance in explanation I: Irrelevance as statistical independence," *Int. J. Approx. Reas.*, vol. 8, pp. 281–324, June 1993.

[42] ——, "Finding MAP's for belief networks is NP-hard," *Artif. Intell. J.*, vol. 68, pp. 399–410, Aug. 1994.

[43] ——, "The role of relevance in explanation II: Disjunctive assignments

and approximate independence," *Int. J. Approx. Reas.*, vol. 13, no. 1, pp. 27–60, July 1995.
[44] S. E. Shimony and E. Santos, Jr., "Exploiting case-based independence for approximating marginal probabilities," *Int. J. Approx. Reas.*, vol. 14, no. 1, pp. 25–54, Jan. 1996.
[45] M. P. Wellman and C.-L. Liu, "State-space abstraction for anytime evaluation of probabilistic networks," in *Uncertainty Artificial Intelligence, Proc. 10th Conf.*, July 1994, pp. 567–574.

Eugene Santos, Jr. (M'93) received the B.S. degree in mathematics and computer science and the M.S. degree in mathematics, both from Youngstown State University, Youngstown, OH. He received the Sc.M. and Ph.D. degrees in computer science from Brown University, Providence, RI.

He is currently an Associate Professor of Computer Science and Engineering in the Computer Science and Engineering Department, University of Connecticut, Storrs. He is also an Adjunct Professor in the Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, where he also served as an Assistant Professor. His research interests include artificial intelligence, neural networks, automated reasoning, natural language processing, probabilistic reasoning and knowledge acquisition, and verification and validation.

Solomon Eyal Shimony received the B.S.E.E. degree from the Technion, Israel Institute of Technology, Haifa, Israel, in 1982, the M.Sc. degree in mathematics and computer science (computer graphics) from Ben-Gurion University, Beer-Sheva, Israel, in 1996, and the Sc.M. and Ph.D. degrees in computer science (artificial intelligence) from Brown University, Providence, RI, in 1989 and 1992, respectively.

He is currently a Senior Lecturer in the Computer Science Division, Mathematics and Computer Science Department, Ben-Gurion University. His areas of research include artificial intelligence, probabilistic reasoning, knowledge discovery in databases, computer vision, robotics, and constraint satisfaction problems.