A Linear Constraint Satisfaction Approach for Abductive Reasoning

Eugene Santos Jr.

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-92-24
April 1992

A Linear Constraint Satisfaction Approach for Abductive Reasoning[1]

Eugene Santos Jr.

Ph.D. Dissertation

Department of Computer Science
Brown University, Box 1910
Providence, RI 02912

May 1992

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

A Linear Constraint Satisfaction Approach for Abductive Reasoning

by
Eugene Santos Jr.
B. S. Mathematics, Youngstown State University, 1985
B. S. Computer Science, Youngstown State University, 1985
M. S. Mathematics, Youngstown State University, 1986
Sc. M. Computer Science, Brown University, 1988

Thesis
Submitted in partial fulfillment of the requirements for the
Degree of Doctor of Philosophy in the Department of Computer Science
at Brown University

May 1992

Abstract

Abductive explanation has been formalized in AI as the process of searching for a set of assumptions that can prove a given observation. A basic problem which naturally arises is that there maybe many different possible sets available. Thus, some preferential ordering on the explanations is necessary to precisely determine which one is best. Unfortunately, any model with sufficient representational power is in general NP-hard.

Causal trees and AND/OR graphs are among the most commonly used for representing causal knowledge. Consequently, finding a best explanation has been treated as some heuristic search through the graph. However, this approach exhibits an expected exponential run-time growth rate.

In this thesis, we present a new approach to modeling abductive reasoning which admits an extremely efficient implementation. We treat the problem in terms of *constrained optimization* instead of graph traversal. Our approach models knowledge using linear constraints and finds a best explanation by optimizing some measure within these constraints. Although finding the best explanation remains NP-hard, our approach allows us to utilize the highly efficient tools developed in *operations research.* Such tools as the *Simplex method* and *Karmarkar's projective scaling algorithm* form the foundations for the practical realization of our approach. Experimental results strongly indicate that our linear constraint satisfaction approach is quite promising. Studies comparing our approach against heuristic search techniques has shown our approach to be superior in both time and space, and actually exhibiting an expected polynomial run-time growth rate.

Our goal is to show that our framework is both flexible and representationally powerful. We can model both cost-based abduction and Bayesian networks. Furthermore, it is possible for us to handle difficult problems such as alternative explanations, continuous random variables, consistency, partial covering and cyclicity which are commonly encountered in abductive (diagnostic) domains.

Vita

Eugene Santos, Jr. was born on January 28, 1968 in Columbus, Ohio. He grew up in Youngstown, Ohio where he was a 1984 graduate of Austintown Fitch High School. While attending high school, he also began his collegiate studies as an undergraduate at Youngstown State University. Subsequently, this early and extensive academic work would payoff on August 1985 in the form of a Summa Cum Laude with both a B. S. in Mathematics and a B. S. in Computer Science at Youngstown State University. With his tremendous love for computing machinery, his days of undergraduate studies were also accompanied by consulting and systems programming work at the Microcomputer Laboratory in Youngstown State University and with various businesses and organizations.

After receiving his Bachelor's degrees, Mr. Santos began to pursue his other fascination, namely mathematics. He would continue at Youngstown State University entering into the graduate program in mathematics. On August 1986, he would receive a M. S. in Mathematics specializing in numerical analysis working under Dr. John J. Buoni. Furthermore, during 1986, he was an Instructor at the university, teaching undergraduate mathematics and computer science courses.

Mr. Santos entered the graduate school at Brown University in the Fall of 1986 to pursue a doctorate in computer science. He received a Sc. M. in Computer Science on May 1988. His Master's thesis was on neural networks and it's application to natural language processing. It was during this time that he began working closely with Dr. Eugene Charniak as his advisor. Eventually, Dr. Charniak would carefully guide Mr. Santos towards his Ph. D.

While at Brown, Mr. Santos served as both a teaching and research assistant as well as various departmental jobs. These jobs included coordinating the comprehensive examination for new doctoral candidates as well as organizing recreational activities for both the students and staff.

Upon completion of his graduate studies, Mr. Santos intends to continue life in academia. In particular, he will seek employment as faculty in some college or university in order to teach computer science and pursue new research. His research interests include automated reasoning, machine learning, natural language understanding, error-correcting parsing, neural networks, expert systems, numerical analysis and object-oriented programming. Besides his interests in Computer Science and Mathematics, Mr. Santos enjoys listening to classical/jazz music, composing music, cooking, volleyball and playing the piano and trumpet.

Acknowledgements

First and foremost, I would like to thank my advisor Eugene Charniak. Through his tremendous support and philosophy of "beating me over the head with the given problem", he has helped me weather the worse of research storms. His guidance has helped me mature (in many more ways than just research) and acquire some of the insight and introspection necessary to doing research. Without his patience and faith in my ability (which I had lost at times), I never would have been able to produce the text you are now reading.

Many thanks to my other committee members, Tom Dean and Pascal van Hentenryck. Both are an incredible source of information and new ideas. If you ever wanted to find out about some topic or opinion, ask one of them. If they don't have the information on-hand, they will always come back with a pointer where to look!

The Department of Computer Science here at Brown University is probably one of the most student-friendly institutions around. You can walk into any faculty, staff, and student office at any time to ask a question or just to chit-chat. All effort is made at streamlining a graduate student's needs and helping him along toward getting his degree. The secretaries and technical staff are always great at handling those problems that crop up now and then in life. If you just look around, you can get that sort of familial-feel about the place.

Of great comfort were my good friends Moises Lejter, Tony Davis, Glenn Carroll, Jak and Kathy Kirman, Ken Basye and numerous others whom provided me with more human activities other than computer science (at least when we weren't all working or disagreeing on what social-type thing to do). I owe much of my sanity to these people.

To the people back at Youngstown State University, namely Dr. John J. Buoni, Dr. Richard Burden, long-time friend Bob Kramer and all the people in Computing Services, thank you for the encouragements and faith in my ability to do the best I can and help me achieve my goals no matter what it took.

Finally, to my mother, father and sister for the endless amounts of support, encouragement and love, thank yous are not enough.

This work has been supported by the National Science Foundation under grant IRI-8911122 and by the Office of Naval Research, under contract N00014-88-K-0589. The author was funded through various TA and RA appointments in the department.

Contents

List of Figures

List of Tables

1 Introduction

The majority of human reasoning tasks seems to be explanatory in nature. We constantly make observations about our environment and then attempt to explain their occurrence. Many of these tasks are often taken for granted. For example, we walk outside on a Spring day and find that the ground is wet. We quickly explain the wetness by assuming recent rain showers and then, just as quickly, we forget about it. Other situations which may have more impact occurs just as often. Consider the following scenario: "John visits his friend Mary's house and finds that the place is dark and quiet. He concludes that Mary is not home." Although we used the word "concludes", John is actually explaining why the house is dark and quiet. The distinct possibility exists that Mary may simply be sleeping inside.

Typically, reasoning has often been modeled in terms of deduction, that is, we try to prove our conclusions given the observations. However, this runs into a snag when dealing with explanatory tasks. Consider traditional deduction in propositional logic. We find that we are incapable of modeling John's reasoning. Our best (only) deductive conclusion would have been: "Mary is not home" $\vee$ "Mary is home" where "$\vee$" denotes logical disjunction. Common approaches to this problem have advocated augmenting propositional logic with certainty factors, probabilities, costs, etc. in an attempt to preserve deduction. This was often used in classical expert systems such as MYCIN [70, 71], PROSPECTOR [14] and INTERNIST [38, 46]. However, the resulting models were clumsy and restrictive. A case in point is Shortliffe's MYCIN system. Although a highly successful system within its restricted domain, its inferencing lacked a proper mathematical as well as semantic basis which stemmed from its treatment of diagnosis as deduction.[2]

Only until recently has explanatory reasoning been properly identified as being separate from deductive reasoning. Pople in [47, 45, 46] was one of the first researchers to point this out through his work on the Caduceus medical diagnosis system. Formally called *abduction*, it was not widely considered as a form of

[2]See [41] for discussion on the limitations of these approaches.

reasoning by the AI community until its popular introduction by Charniak and McDermott in [5]. Since then, many common problems have been identified in its terms. For example, such problems include medical diagnosis [8, 42], circuit fault detection [12, 11, 20] and story understanding [22, 23, 3].[3]

Clearly, we extensively use abductive reasoning in our everyday tasks from explaining why the ground is wet to performing sophisticated inferencing in medical diagnosis. Thus, we need an approach to modeling abduction which is representationally robust and permits a practical implementation. To our chagrin, however, it seems that abductive reasoning is an inherently difficult process. Indeed, various abductive models have been shown to be NP-hard [9, 41, 7, 64, 42].

To better understand the difficulty inherent in abduction, let us attempt to model John's situation above. The information John used to arrive at his conclusion can be described with the following set of propositions: We model causality as logical implication in order to build our knowledge-base using logical rules.

$$\begin{aligned}
\mathsf{house\text{-}dark} \wedge \mathsf{house\text{-}quiet} &\Longrightarrow \mathsf{house\text{-}dark\text{-}quiet}\\
\mathsf{lights\text{-}out} &\Longrightarrow \mathsf{house\text{-}dark}\\
\mathsf{no\text{-}one\text{-}home} \vee \mathsf{blackout} &\Longrightarrow \mathsf{lights\text{-}out}\\
\mathsf{tv\text{-}off} \wedge \mathsf{radio\text{-}off} &\Longrightarrow \mathsf{house\text{-}quiet}\\
\mathsf{no\text{-}one\text{-}home} \vee \mathsf{no\text{-}shows} \vee \mathsf{blackout} &\Longrightarrow \mathsf{tv\text{-}off}\\
\mathsf{no\text{-}one\text{-}home} \vee \mathsf{bad\text{-}songs} \vee \mathsf{blackout} &\Longrightarrow \mathsf{radio\text{-}off}
\end{aligned}$$

where "$\wedge$", "$\vee$" and "$\Longrightarrow$" denote conjunction, disjunction and implication, respectively. This abductive reasoning task can be viewed as a backward-chaining process on the propositions. In essence, we are traveling backwards through the implications in hopes of finding a set of assumptions which can serve as an explanation for the evidence. For example, assuming that no one is home is a possible explanation for the house being dark and quiet.

Abductive explanation has been formalized in AI as the process of searching for some set of assumptions that can prove the things to be explained [7, 26, 60, 64, 73, 31, 20, 43, 41]. We call each such set an *explanation* for the given evidence. A basic problem which naturally arises is that there maybe many different possible

[3] For a good general discussion of abduction, see [29].

explanations available. From traditional symbolic logic, the only measure of a set's viability as an explanation is the simple fact concerning whether the evidence can be deductively inferred from the set. Thus, even the most far-fetched set of assumptions can be a possible candidate as long as it implies the evidence. For example, the house may be dark and quiet because of a blackout which in general is a slightly less plausible possibility. In a related but slightly different problem, consider the explanation whereby John simply assumes that the house is dark and quiet. This is a perfectly legitimate answer but provides no useful information.

We can easily see that some preferential ordering on the explanations is necessary. This would serve to precisely define the notion of a best explanation as well as subsequent next best which is critical to have in domains such as medical diagnosis.

Several ordering measures are available such as *least specific abduction* [26, 73], *cost-based abduction* [7], *parsimonious covering theory* [43] and *belief revision* [41]. Each approach offers different perspectives on the problem and provides individual frameworks capable of modeling certain aspects of abductive reasoning.

The complexity of abduction quickly becomes apparent in that the problem now involves the search through a most likely exponential space of solutions for a single maximal or minimal. The knowledge representation used in abduction is generally rule-based and often has a graphical representation. Causal trees and AND/OR graphs are among the most commonly used. Explanations are thus subgraphs of these structures which explicitly detail the inferences used to prove the evidence.

Naturally, finding a best explanation has been treated as a heuristic search through the graph. However, these heuristics exhibit an expected exponential run-time growth rate. With the problem being NP-hard for any sufficiently sophisticated model and the inability of finding efficient graph search heuristics, the practical realization of abductive reasoning seems rather bleak.

In this paper, we present a new approach to modeling abduction. Since knowledge-bases are typically graphical in nature, all models have thus far been designed around a graph search engine for reasoning. We make the following observation: *Abductive reasoning is ultimately a constrained optimization problem.*

Basically, our approach models knowledge using *linear constraints*. We find a best explanation by optimizing some measure within these constraints. Reducing the highly structured problem of abduction into a seemingly less structured problem of linear constraint satisfaction might suggest some "loss" in the transformation. However, as will be quite evident, structures such as causal knowledge hierarchies are completely preserved in the reduction and can be straightforwardly retrieved.

Linear constraint satisfaction is a very well understood problem in *Operations Research*. Our reasoning engine is thus formed from highly efficient tools and techniques developed in OR. Such tools as the *Simplex method* and *Karmarkar's projective scaling algorithm* [36, 39, 59] provide us with a firm foundation to building a practical system. Experimental results strongly indicate that our linear constraint satisfaction approach is quite promising. Studies comparing our approach against heuristic search techniques on existing abduction problems has shown our approach to be superior in both time and space, and actually exhibiting an expected polynomial run-time growth rate [53, 55, 54].

Our goal is to show that our framework is both flexible and powerful enough to solve interesting problems in abductive reasoning. With our linear constraint satisfaction approach, we can completely model existing approaches such as cost-based abduction [53, 55, 54] and belief revision [57]. Especially in the case of cost-based abduction, thorough experimentation has shown that our approach has now made a computationally difficult problem extremely feasible for extensive use in existing applications such as the WIMP story comprehension system [6].

Furthermore, we consider some issues which remain unaddressed by the existing models. Mainly, this is due to the additional complexities imposed by these issues, thus making an already difficult problem impossible given their approach. For example, consider the problem that often crops up in the domains of the WIMP story understanding system. Our knowledge-base contains the following rules:

$$(\mathsf{foo}\ a) \wedge (=\ a\ b\ ) \implies (\mathsf{foo}\ b)$$
$$(\mathsf{foo}\ b) \wedge (=\ a\ b\ ) \implies (\mathsf{foo}\ a)$$

Since explanation is a backward chaining process, the existence of the above rules can throw explicit chaining algorithms into infinite loops. We call this problem and any "non-acyclic" knowledge bases, *cyclicity*. We can show that by using linear constraints, such issues are naturally handled within our framework [58, 56].

In Section 2, we briefly examine some of the existing frameworks for modeling abductive reasoning. We begin our linear constraints approach in Section 3 by modeling cost-based abduction. We will provide a detailed analysis of our approach plus extensive experimental data comparing it against existing graphical search solutions. In Section 4, we continue our approach by modeling Bayesian networks. In particular, we begin by concentrating on belief revision as an abductive model and proceed to formulating belief updating within our framework. Now having shown the flexibility and representational power of our approach through cost-based abduction and bayesian networks, we tackle the problem of cyclicity in Section 5.

2 Earlier Abduction Models

Knowledge for abductive reasoning is generally cast as propositions and rules operating on the propositions. The goal is to find a set of propositions which when operated on by some set of rules will result in a proof for the evidence. The propositions are considered to be the hypotheses or assumptions made to explain the observation.

In general, there are many different sets of propositions available as explanations. Early measures imposing a preferential ordering on these sets were based on the number and type of propositions in the set. One such approach was to simply maximize or minimize the number of hypotheses needed. However, consider the following case: Returning to our story above, it seems reasonable that "no one is home" is a better explanation than one requiring that both "the songs are awful" and "the shows are bad". Furthermore, these two different explanations are still better than the one which assumes a power failure. As we can easily see, the cardinalities of these explanations are 1, 2 and 1, respectively. Both maximizing and minimizing will fail to capture what we consider to be the best explanation.

Another simple approach is to designate some set of propositions as *assumable.* Thus, any set of hypotheses must consist only of assumable propositions. However, we often run into the problem of the explanations either being too detailed or not detailed enough.

2.1 Weighted Abduction

Hobbs and Stickel [26, 73] proposed an approach called *weighted abduction.* It involves levying numerical costs to making individual assumptions. The cost of an explanation is a function of the cost of the individual assumptions made in the explanation. These costs are used in an effort to guide the abductive system in favoring the intended explanations. The final choice for best explanation will be the one with *least cost.*

The main difficulty of this approach, however, is the lack of any clear semantics for the cost assignments. Appelt [1] attempted to provide semantics, but it was found to be incomplete and inextensible. Furthermore, it failed to give an intuitive feel for what the numbers really mean.

2.2 Cost-Based Abduction

Charniak and Shimony presented a minor variant of weighted abduction called *cost-based abduction* [7]. It has been shown in [7] that belief revision in Bayesian networks [41] can be accurately modeled by cost-based abduction.

In cost-based abduction, hypotheses have associated costs, and the cost of a proof is simply the sum of the costs of the hypotheses required to complete that proof. (Examples of such proofs can be found in [7, 4].) Central to this approach is the use of *directed acyclic graphs* called WAODAGs (or, weighted AND/OR directed acyclic graphs) [7, 4] to represent relationships between hypotheses and the evidence to be explained. Each node represents some piece of knowledge, and the connections explicitly detail the relationships between different pieces. Furthermore, each node in a WAODAG corresponds to a logical AND or OR operation on its immediate parents.

Assigning a truth value to each node is considered a proof if the assignment is consistent with respect to the boolean network and if the items we wish to explain have been explained, i.e., have been assigned a value of true. Furthermore, each hypothesis used in a proof will incur a cost. Consequently, each such proof will have an associated cost which is simply the sum of the hypothesis costs incurred. The goal is to find an assignment which has minimal cost (see Figure 2.1). Charniak and Shimony [7] also showed that by interpreting the costs as negative log probabilities, cost-based abduction can be reduced to belief revision in Bayesian networks. Thus, the cost semantics problem of weighted abduction is not encountered. Unfortunately, finding minimal cost proofs has been shown to be NP-hard [7].

Current approaches to finding the best proof have centered around using a best-first search technique and *expanding* partial proofs to search for the best

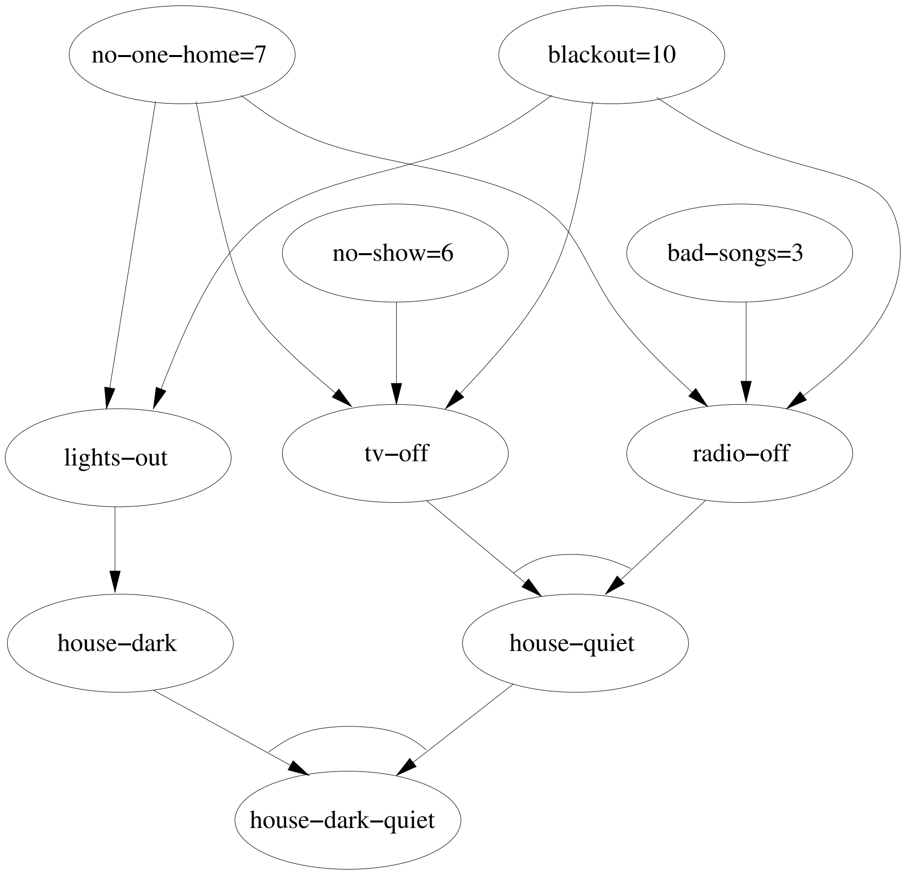


FIG. 2.1. A simple WAODAG. The AND-node house-dark-quiet is the observation. The nodes no-one-home, no-shows, blackout and bad-songs are the hypotheses with associated costs 7, 6, 10 and 3, respectively. The assignment of no-one-home to true and bad-songs, blackout and no-shows to false results in lights-out, radio-off, tv-off, house-dark and house-quiet to be true. This proof has a cost of 7 and is the minimal cost proof.

proof [4]. (We will continue with a more detailed discussion of cost-based abduction in Section 3.)

2.3 Belief Revision

Pearl presented an approach to modeling belief revision by using Bayesian networks [41]. Based upon the tenets of probability theory, events are represented by random variables and direct and indirect causal relationships between events are modeled by conditional probabilities and conditional independence. For example, if the lights are off, then the house is dark. This is a direct causal relationship which is modeled in this approach by the conditional probability $P(\textsf{house-dark} \mid \textsf{lights-out})$. Indirect causal relationships such as "if no one is home, then the house is dark" contains an intermediary relationship concerning the "lights being out". Furthermore, we know for a fact, that the status of the lights will completely determine whether the house is dark or not. Thus, if we are given the status of the lights being on or off, then any information about any one being home becomes irrelevant in determining the lighting conditions in the house. This sort of indirect causal relationship is modeled using conditional independence,

$$P(\textsf{house-dark} \mid \textsf{lights-out}, \textsf{no-one-home}) = P(\textsf{house-dark} \mid \textsf{lights-out}).$$

Properly constructed with a consistent set of conditional probability assignments, a Bayesian network will represent a unique probability distribution over the random variables. For our story, we may have the probability assignments in Figure 2.2.

The goal of belief revision on Bayesian networks is to find an instantiation of all the random variables which will maximize their joint probability. When evidence is given to be explained, an instantiation must be sought to maximize the joint probability given the evidence. The instantiation which maximizes this probability is called the best explanation. This measure is Pearl's *most-probable explanation criterion* (MPE).

This formulation of causal knowledge in terms of conditional probabilities admits a nice graphical representation which becomes central in determining the

$P(\text{no-one-home} = \text{true}) = .4$
$P(\text{blackout} = \text{true}) = .1$
$P(\text{no-shows} = \text{true}) = .7$
$P(\text{bad-songs} = \text{true}) = .8$
$P(\text{lights-out} = \text{true} \mid \text{no-one-home} = \text{true}, \text{blackout} = \text{true}) = 1$
$P(\text{lights-out} = \text{true} \mid \text{no-one-home} = \text{true}, \text{blackout} = \text{false}) = 1$
$P(\text{lights-out} = \text{true} \mid \text{no-one-home} = \text{false}, \text{blackout} = \text{true}) = 1$
$P(\text{lights-out} = \text{true} \mid \text{no-one-home} = \text{false}, \text{blackout} = \text{false}) = 0$
$P(\text{tv-off} = \text{true} \mid \text{no-one-home} = \text{true}, \text{no-shows} = \text{true}, \text{blackout} = \text{true}) = 1$
$P(\text{tv-off} = \text{true} \mid \text{no-one-home} = \text{true}, \text{no-shows} = \text{true}, \text{blackout} = \text{false}) = 1$
$P(\text{tv-off} = \text{true} \mid \text{no-one-home} = \text{true}, \text{no-shows} = \text{false}, \text{blackout} = \text{true}) = 1$
$P(\text{tv-off} = \text{true} \mid \text{no-one-home} = \text{false}, \text{no-shows} = \text{true}, \text{blackout} = \text{true}) = 1$
$P(\text{tv-off} = \text{true} \mid \text{no-one-home} = \text{true}, \text{no-shows} = \text{false}, \text{blackout} = \text{false}) = 1$
$P(\text{tv-off} = \text{true} \mid \text{no-one-home} = \text{false}, \text{no-shows} = \text{false}, \text{blackout} = \text{true}) = 1$
$P(\text{tv-off} = \text{true} \mid \text{no-one-home} = \text{false}, \text{no-shows} = \text{true}, \text{blackout} = \text{false}) = 1$
$P(\text{tv-off} = \text{true} \mid \text{no-one-home} = \text{false}, \text{no-shows} = \text{false}, \text{blackout} = \text{false}) = 0$
$P(\text{radio-off} = \text{true} \mid \text{no-one-home} = \text{true}, \text{bad-songs} = \text{true}, \text{blackout} = \text{true}) = 1$
$P(\text{radio-off} = \text{true} \mid \text{no-one-home} = \text{true}, \text{bad-songs} = \text{true}, \text{blackout} = \text{false}) = 1$
$P(\text{radio-off} = \text{true} \mid \text{no-one-home} = \text{true}, \text{bad-songs} = \text{false}, \text{blackout} = \text{true}) = 1$
$P(\text{radio-off} = \text{true} \mid \text{no-one-home} = \text{false}, \text{bad-songs} = \text{true}, \text{blackout} = \text{true}) = 1$
$P(\text{radio-off} = \text{true} \mid \text{no-one-home} = \text{true}, \text{bad-songs} = \text{false}, \text{blackout} = \text{false}) = 1$
$P(\text{radio-off} = \text{true} \mid \text{no-one-home} = \text{false}, \text{bad-songs} = \text{false}, \text{blackout} = \text{true}) = 1$
$P(\text{radio-off} = \text{true} \mid \text{no-one-home} = \text{false}, \text{bad-songs} = \text{true}, \text{blackout} = \text{false}) = 1$
$P(\text{radio-off} = \text{true} \mid \text{no-one-home} = \text{false}, \text{bad-songs} = \text{false}, \text{blackout} = \text{false}) = 0$
$P(\text{house-dark} = \text{true} \mid \text{lights-out} = \text{true}) = 1$
$P(\text{house-dark} = \text{true} \mid \text{lights-out} = \text{false}) = 0$
$P(\text{house-quiet} = \text{true} \mid \text{tv-off} = \text{true}, \text{radio-off} = \text{true}) = 1$
$P(\text{house-quiet} = \text{true} \mid \text{tv-off} = \text{true}, \text{radio-off} = \text{false}) = 0$
$P(\text{house-quiet} = \text{true} \mid \text{tv-off} = \text{false}, \text{radio-off} = \text{true}) = 0$
$P(\text{house-quiet} = \text{true} \mid \text{tv-off} = \text{false}, \text{radio-off} = \text{false}) = 0$
$P(\text{house-dark-quiet} = \text{true} \mid \text{house-dark} = \text{true}, \text{house-quiet} = \text{true}) = 1$
$P(\text{house-dark-quiet} = \text{true} \mid \text{house-dark} = \text{true}, \text{house-quiet} = \text{false}) = 0$
$P(\text{house-dark-quiet} = \text{true} \mid \text{house-dark} = \text{false}, \text{house-quiet} = \text{true}) = 0$
$P(\text{house-dark-quiet} = \text{true} \mid \text{house-dark} = \text{false}, \text{house-quiet} = \text{false}) = 0$

FIG. 2.2. A probability assignment for our story.

most-probable explanation. However, as the Bayesian networks become increasingly sophisticated, the current methods used to compute them also become extremely complicated. Furthermore, the best computational method which uses *message-passing schemes* [41] are incapable of generating the subsequent next best explanations beyond the second best one. (We will continue with a more detailed discussion of Bayesian networks including belief updating in Section 4.)

2.4 Parsimonious Covering Theory

Parsimonious covering theory is an approach presented by Peng and Reggia [43] for medical diagnosis. A *diagnostic problem* is defined as a two-layer network consisting of a layer of *manifestations* which are causally affected by a layer of *disorders*. Given a subset of the manifestations as evidence, a subset of disorders must be chosen to best explain the manifestations. The choice of best explanation is determined through a *covering set* approach. A collection of disorders which can explain the manifestations is called a *cover*. A cover is a best explanation if none of its proper subsets is also a cover. Such a cover is also said to be *irredundant*.

A limitation of this theory as pointed out by Peng and Reggia [43] is the large number of covers which are considered "best". In order to further select from these potential explanations, some additional criteria must be used. Thus, basic parsimonious covering theory is extended to incorporate probability theory. The potential of an explanation is now measured by some probability. With the addition of probabilities, care must be taken in choosing which covers are to be inspected. Peng and Reggia [43] proposed a 2-layer Bayesian network to probabilistically model their approach. However, extending their approach to more general problems is not readily obvious.

2.5 Coherence

Thagard [74] proposed an approach for modeling explanation in general. Called *explanatory coherence*, the theory consists of several principles that establish relations of local *coherence* between a hypothesis and other propositions. Vaguely:

> Propositions P and Q *cohere* if and only if there is some explanatory relation between them.

Accordingly, there are four possibilities as to what an explanatory relation might be:

- P is part of the explanation for Q.
- Q is part of the explanation for P.
- P and Q together are part of the explanation for some proposition R.
- P and Q are analogous in the explanation they respectively give for some R and S.

Satisfying any one of the four possibilities indicates a strong degree of confidence that P and Q can both be present. We measure our best explanation to be the set of propositions which can "cohere" together "best". *Incoherence* between two propositions occurs if they contradict each other or if they offer explanations that background knowledge suggests are incompatible.

As we can easily see, we may have many explanations which "cohere" best according to our current definition. Thagard continues by refining the four possibilities into seven distinct principles. In this way, it is hoped that further gradations can be made in "best". Although a seemingly sound theory, it is rather complex. A connectionist [52, 15, 35] implementation has been attempted, however, its feasibility in applications seems questionable.

2.6 Other Approaches

The above methods are directed mainly towards modeling abduction. Other approaches are certainly available. However, aside from the five methods we have just studied, the remaining ones handle abductive reasoning sort of as an afterthought to their main goals. Such systems include truth maintenance systems [12], influence diagrams [61], probabilistic logic [40], Dempster-Shafer theory [13, 63] and fuzzy logic [75].

2.7 Related Work

There are also various strands of work which, while somewhat related to the work described in this thesis, are nevertheless sufficiently distant not to warrant a full fledged review. In particular, we have in mind the following:

- Solving Constraint Satisfaction Problems (CPS's) such as the n-queens problems through Diophantine equations [51].
- Solving the relaxation labeling process through a Simplex-like algorithm [76].
- Updating deductive databases though linear programming techniques [2].
- Reduction of independence-based MAPs in Bayesian networks to linear constraint satisfaction [69].
- The problem of path planning in robotics in relation to routing and transportation problems [49].
- Work done with distributed intelligent agents [37].

3 Cost-Based Abduction

Our basic approach towards knowledge representation involves a mapping from objects and/or propositions in the world to real variables. The values that a real variable may attain are analogous to the changing states of the associated object or proposition. In the simplest case, we can map the truth or falsity of a proposition into the values 1 and 0 for the corresponding real variable. As we shall see later, more complicated mappings such as objects with multiple states can be accomplished by solely using the values 0 and 1 (see Section 4.1.1).

With the close correlation of real variables to objects in the world, we can model relationships such as causal/logical information between the various objects and/or propositions through linear constraints on the appropriate real variables. Taking all these linear constraints together, we can now mathematically define the space of possible solutions we wish to consider. A linear function on the real variables called, an *objective function* ,is then applied to this space to precisely determine the solution we desire.

Our goal in this section is to model cost-based abduction using this approach. This and the subsequent section which models Bayesian networks should demonstrate the representational capabilities of our linear constraints formulation.

3.1 WAODAGS

The keystone of *cost-based abduction* [7] is the *weighted* AND/ OR *directed acyclic graph* (abbreviated WAODAG) which models the relationships between objects and/or concepts in the world. Each node in the graph embodies some object or concept while each edge represents direct causal/logical relationships between nodes incident to the edge. For example, suppose we are in a house which contains a radio and a television set. Furthermore, suppose we also understand that having the radio off plays some role in the house being quiet and that the radio may be off because either there is no one home or the music is terrible to listen to. Assume we have nodes labeled radio-off, house-quiet, no-one-home and bad-songs representing the propositions that the radio is off, the house is quiet, no one

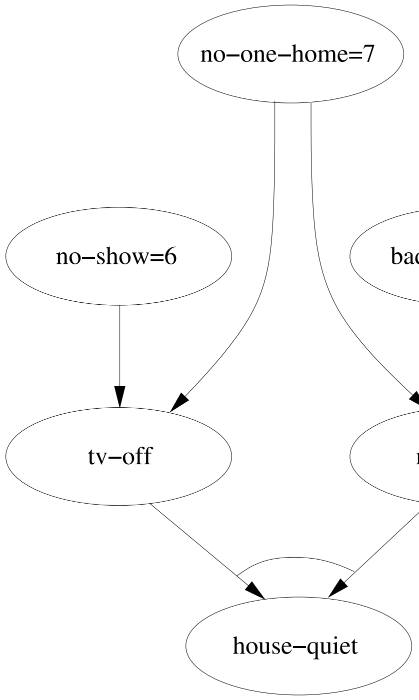


FIG. 3.1. A simpler WAODAG. The AND-node house-quiet is the observation. The nodes no-shows, no-one-home and bad-songs are the hypotheses with associated costs 6, 7 and 3, respectively.

is home and that the music is awful, respectively. An accurate indication of the relationships between these propositions can be made by introducing edges between radio-off and each of house-quiet, no-one-home and bad-songs as has been done in Figure 3.1. (This is similarly done for tv-off and no-shows.)

A dag alone, however, only represents the existence of relationships and does not specify their exact nature. In the above example, we know that if either the music is bad or no one is home, then the radio will be off. This relationship can be easily modeled by the rule no-one-home $\vee$ bad-songs $\Longrightarrow$ radio-off where "$\vee$" and "$\Longrightarrow$" represent disjunction and implication, respectively. Similarly, the relationship that the house is quiet if both the radio and television are off can be represented by radio-off $\wedge$ tv-off $\Longrightarrow$ house-quiet where "$\wedge$" denotes conjunction. Using this rule-based approach, the nodes in the dag can be augmented by the boolean functions "AND" and "OR" which take as input, the immediate parents of each node, if any. This AND/OR dag can now be used to completely specify the causal/logical relationships between the objects in our example.

Finally, since we are reasoning via abduction as opposed to deduction, our goal is to attempt to find the *best* set of hypotheses which can *prove* the given observation. Since we are utilizing a rule-based approach to represent our knowledge, the notion of a *proof* is straightforward. It simply consists of a set of hypotheses plus some set of rules. In cost-based abduction, the *cost* of a proof is a measure on the set of hypotheses it uses. Each hypothesis is assigned a *cost* and the cost of a proof is the sum of the costs of all the hypotheses used. The best proof is then the one with minimal attached cost.

We now formalize the minimum cost-based abduction problem:

DEFINITION 3.1. *A* WAODAG[4] *is a 4-tuple* (G, c, r, S), *where:*

1. G *is a directed acyclic graph,* $G = (V, E)$.
2. c *is a function from* $V \times \{\mathsf{true}, \mathsf{false}\}$ *to* $\Re$, *called the* cost function.
3. r *is a function from* V *to* $\{\text{AND}, \text{OR}\}$, *called the* label. *A node labeled* AND *is called an* AND-node, *etc.*
4. S *is a subset of nodes in* V *called the* evidence nodes.

NOTATION. V_H is the set of all nodes in V with indegree 0. The nodes in V_H are also called the *hypothesis nodes.*

The WAODAG is our graphical representation of the rule-based knowledge to be used by the cost-based abduction approach. The dag G represents our relationships, c represents the cost of assuming that a piece of knowledge is true or false, r indicates the type of boolean function associated and S represents the set of observations to be explained.

Clearly, an explanation is the same as a proof for cost-based abduction. We formally define a proof for a set of observations as an assignment of truth values to the different objects in the knowledge base such that the observations are true and the assignments are consistent with respect to the rule-based relationships between the objects.

In Figure 3.1, if $\{\mathsf{house\text{-}quiet} = \mathsf{true}\}$ is the observation to be explained, one

[4] Slight generalization of Charniak and Shimony [7].

possible proof would be the following assignment of truth values: {house-quiet, radio-off, bad-songs, tv-off, no-shows} are assigned true and {no-one-home} is assigned false. A second possibility is to assign all of them to true. And as a third possibility, we can assign {house-quiet, radio-off, tv-off, no-one-home} to true and {bad-songs, no-shows} to false. As we can easily see, all three assignments are internally consistent with the associated rule-based relationships and assign house-quiet to true.

More formally, we define this as follows:

DEFINITION 3.2. *A* truth assignment *for a* WAODAG $W = (G, c, r, S)$ *where* $G = (V, E)$ *is a function* e *from* V *to* $\{\mathsf{true}, \mathsf{false}\}$. *We say that such a function is* valid *iff (if and only if) the following conditions hold:*

1. *For all* AND*-nodes* q, $e(q) = \mathsf{true}$ *iff for all nodes* p *such that* (p, q) *is an edge in* E, $e(p) = \mathsf{true}$.
2. *For all* OR*-nodes* q, $e(q) = \mathsf{true}$ *iff there exists a node* p *such that* (p, q) *is an edge in* E *and* $e(p) = \mathsf{true}$.

Furthermore, we say that e *is an* explanation *iff* e *is valid and for each node* q *in* S, $e(q) = \mathsf{true}$.

Once we have the different possible explanations for some observation, we must now associate a cost to each one to impose some order reflecting the *goodness* of proofs. As we mentioned earlier, the cost of a proof is simply the sum of the individual costs of the hypotheses assumed.[5]

DEFINITION 3.3. *We define the* cost of an explanation e *for* $W = (G, c, r, S)$ *where* $G = (V, E)$ *as*

$$C(e) = \sum_{q \in V} c(q, e(q)) \tag{1}$$

An explanation e *which minimizes* C *is called a* best explanation *for* W.

From (1), we find that our three proofs above have costs 9, 16 and 7, respectively. Of the three our best proof is the third one with the cost of 7. This proof also turns out to be the best explanation (minimal-cost proof) for Figure 3.1.

[5]Although we have only discussed having costs associated with hypotheses, our approach permits costs to be associated with any node.

As we mentioned earlier, finding the minimal-cost proof can be a difficult task. Using brute force enumeration techniques is fine for small problems but grows exponentially in complexity. Charniak and Shimony [7] have proven that this problem is NP-complete by transforming it into the *vertex cover problem* [17]. Straightforwardly, this problem can be transformed into a search problem on AND/OR graphs. However, efficient admissible heuristics seem to be difficult to find. (In Section 3.4, we compare the computationally efficiency of our approach against a search heuristic designed for use on WAODAGs generated by the WIMP story understanding system [4].)

3.2 Constraint System Formulation

Basically, cost-based abduction is ultimately an *optimization* problem. So, instead of treating it as a traditional graph search problem, we consider the problem in terms of *constraint satisfaction.* In what follows, we will show how cost-based abduction can be directly transformed into a minimization problem on a collection of linear constraints. Central to this transformation is the formulation of *linear constraint systems* (or simply, *constraint systems*) which can be shown to correctly solve the minimum cost-based abduction by determining the minimal-cost proof.

NOTATION. $\Re$ denotes the set of real numbers.

DEFINITION 3.4. *A* constraint system *is a 3-tuple* (Γ, I, ψ) *where* Γ *is a finite set of real variables,* I *is a finite set of linear inequalities based on* Γ*, and* ψ *is a function from* $\Gamma \times \{0, 1\}$ *to* $\Re$ *called the* cost function.

NOTATION. For each node q in V, let $D_q = \{p|(p,q) \textit{ is an edge in } E\}$, the parents of q. $|D_q|$ is the cardinality of D_q.

From Definition 3.2, for a truth assignment to be a possible explanation, we must guarantee the internal consistency of the assignments required in the definition. This internal consistency is the same consistency required in boolean combinatorial circuits. We must guarantee the correct assignment of input values versus output values of each AND/OR node in the WAODAG.

Like values in boolean circuits, we can use numerical assignments instead of true or false. In general, we use 1 for true and 0 for false. By taking this viewpoint, we can now consider the internal consistency as some form of mathematical formulae to be satisfied where each node is actually a variable in the equation. Our purpose is now to show how these equations can be derived and then prove that they guarantee the internal consistency required.

We begin our derivation with the simplest of the requirements. Let q be an evidence node in our WAODAG. Associate the variable x_q with q. Since q is an evidence node, any explanation for q must assign q to true. This can be modeled by the equation

$$x_q = 1$$

Next, let q be an AND-node with parents D_q. We have the following: *q is* true *iff p is* true *for all nodes p in D_q*. Symmetrically, *q is* false *iff there exists a p in D_q such that p is* false. We can accomplish this with the following equations:

$$x_q \leq x_p \textit{ for each } p \in D_q$$

which guarantees that

1. q being true forces all p in D_q to be true, and
2. some p in D_q being false forces q to be false;

$$\sum_{p \in D_q} x_p - |D_q| + 1 \leq x_q$$

guaranteeing that

1. q being false forces some p in D_q to be false, and
2. if all p in D_q are true, then q must be true.

Note that at this time we are assuming that our variables may only take values of 0 or 1 although there is no upper or lower bound on the results of evaluating either side of the equation. For example, let $D_q = \{a, b, c, d\}$, $x_a = x_b = x_c = 0$ and $x_d = 1$. This implies that the summation side of the third equation above yields -2!

Now consider the OR-node. Let q be an OR-node with parents D_q. We have: *q is* false *iff p is* false *for all nodes p in D_q*. We can also accomplish this with the following equations:

$$\sum_{p \in D_q} x_p \geq x_q$$

which guarantees that

1. q being true forces some p in D_q to be true, and
2. if all p in D_q are false, then q must be false;

$$x_q \geq x_p \textit{ for each } p \in D_q$$

guarantees that

1. q being false forces all p in D_q to be false, and
2. some p in D_q being true forces q to be true.

Together, these equations will guarantee the internal consistency needed for a truth assignment to be an explanation. Also, any explanation is guaranteed to satisfy this set. We formalize our construction as follows:

DEFINITION 3.5. *Given a* WAODAG $W = (G, c, r, S)$ *where* $G = (V, E)$, *we can construct a constraint system* $L(W) = (\Gamma, I, \psi)$ *where:*

1. Γ *is a set of variables indexed by* V, *i.e.,* $\Gamma = \{x_q | q \in V\}$,
2. $\psi(x_q, X) = c(q, X)$ *for all* $q \in V$ *and* $X \in \{\mathsf{true}, \mathsf{false}\}$,
3. I *is the collection of all inequalities of the forms given below:*

$$x_q \leq x_p \in I \textit{ for each } p \in D_q \textit{ if } r(q) = \text{AND} \tag{2}$$

$$\sum_{p \in D_q} x_p - |D_q| + 1 \leq x_q \in I \textit{ if } r(q) = \text{AND} \tag{3}$$

$$\sum_{p \in D_q} x_p \geq x_q \in I \textit{ if } r(q) = \text{OR} \tag{4}$$

$$x_q \geq x_p \in I \textit{ for each } p \in D_q \textit{ if } r(q) = \text{OR} \tag{5}$$

We say that L(W) is induced *by W. Furthermore, by including the additional constraints:*

$$x_q = 1 \textit{ if } q \in S \tag{6}$$

we say that the resulting constraint system is induced evidentially *by W and is denoted by* $L_E(W)$.

Definition 3.6. *A* variable assignment *for a constraint system* $L = (\Gamma, I, \psi)$ *is a function* s *from* Γ *to* $\Re$. *Furthermore,*

1. *If the range of* s *is* $\{0, 1\}$, *then* s *is a* 0-1 assignment.
2. *If* s *satisfies all the constraints in* I, *then* s *is a* solution *for* L.
3. *If* s *is a solution for* L *and a 0-1 assignment, then* s *is a* 0-1 solution *for* L.

With our transformation of cost-based abduction into constraint systems, we must now prove that 0-1 solutions are equivalent to explanations. Given a 0-1 assignment s for $L(W)$, we can construct a truth assignment e for W as follows:

1. For all q in V, $s(x_q) = 1$ iff $e(q) = \mathsf{true}$.
2. For all q in V, $s(x_q) = 0$ iff $e(q) = \mathsf{false}$.

Conversely, given a truth assignment e for W, we can construct a 0-1 assignment s for $L(W)$.

Notation. e_s and s_e denote, respectively, a truth assignment e constructed from a 0-1 assignment s, and a 0-1 assignment s constructed from a truth assignment e.

We can show that all explanations for a given WAODAG W have corresponding 0-1 solutions for $L_E(W)$ and vice versa.

Theorem 3.1. *If* e *is an explanation for* W, *then* s_e *is a 0-1 solution for* $L(W)$.

(Proofs can be found in Appendix A.)

Corollary 3.2. *Let* L *be constructed from* $L(W)$ *by eliminating all constraints of the forms (3) and (5). If* e *is an explanation for* W, *then* s_e *is a solution of* L.[6]

Conditions (2) and (4) are called *bottom-up constraints* since they dictate the values of variables from the direction of the evidence nodes. Symmetrically, (3) and (5) are called *top-down constraints.*

Theorem 3.3. *If* s *is a 0-1 solution for* $L_E(W)$, *then* e_s *is an explanation for* W.

[6] L is later defined as a WAODAG *semi-induced constraint system.*

From Theorems 3.1 and 3.3, 0-1 solutions for constraint systems are the counterparts of explanations for WAODAGs. By augmenting a WAODAG induced constraint system with a cost function, the notion of the cost of an explanation for a WAODAG can be transformed into the notion of the cost of a 0-1 solution for the constraint system.

To complete the derivation, we must also be able to compute the costs associated with each proof. We can do this as follows:

DEFINITION 3.7. *Given a constraint system $L = (\Gamma, I, \psi)$, we construct a function Θ_L from variable assignments to $\Re$ as follows:*

$$\Theta_L(s) = \sum_{x \in \Gamma} \{s(x)\psi(x, \mathsf{true}) + (1 - s(x))\psi(x, \mathsf{false})\}. \tag{7}$$

Θ_L is called the objective function *of L.*

DEFINITION 3.8. *An* optimal 0-1 solution *for a constraint system $L = (\Gamma, I, \psi)$ is a 0-1 solution for L which minimizes Θ_L.*

As we can clearly see, (7) is identical to (1). From Theorems 3.1 and 3.3 and the relationship between node assignments and variable assignments, an optimal 0-1 solution in $L_E(W)$ is a best explanation for W and vice versa.

Taking the set of linear inequalities I and the objective function Θ_L, we observe that we have the elements known in operations research as a *linear program* [59, 36, 39]. The goal of a linear program is to minimize an objective function according to some set of linear constraints. Highly efficient methods such as the *simplex method*[7] and *Karmarkar's projective scaling algorithm* are used to solve linear programs [59, 36, 39]. Empirical studies have shown that the running time of the simplex method is roughly linear with respect to the number of constraints and the number of variables in the linear program [39].

PROPOSITION 3.4. *Given a WAODAG $W = (G, c, r, S)$ where $G = (V, E)$, if $L_E(W) = (\Gamma, I, \psi)$ is induced evidentially from W, then $|I| = |E| + |V - V_H| + |S|$.*

Although our constraint systems seem similar in nature to linear programs, linear programs are incapable of making restrictions which cannot be modeled by

[7] For a quick overview of the simplex method, see [25].

linear inequalities. Thus, solutions which minimize the objective function may not be strictly 0 and 1. However, if the solution is a 0-1 solution, then the best explanation has been found. (From our experiments, as we shall see later, the optimal solutions for many of these linear programs will in fact be 0-1 solutions. Thus, the best explanation can be found by just using straight Simplex methods on the problems.) If the solution is not a 0-1 solution, the value for the objective function generated by such a solution still provides an excellent lower bound to the cost of an optimal 0-1 solution. This lower bound will be used to direct our search for an optimal 0-1 solution as we shall see below.

For computing the lower bound, WAODAG induced linear programs are well suited for the simplex method. The constraint matrices for these types of linear programs are extremely sparse and consist of only three values: -1, 0, 1. Furthermore, detailed knowledge of the problem structure can be exploited to even further improve performance (see Section 3.5). The following theorem shows that the number of linear inequalities can be reduced under certain conditions.

Let $W = (G, c, r, S)$ where $G = (V, E)$ is a WAODAG such that only the hypothesis nodes have non-zero costs.

DEFINITION 3.9. *We can construct a constraint system* $\hat{L}(W) = (\Gamma, I, \psi)$ *where:*

1. Γ *is a set of variables indexed by* V*, i.e.,* $\Gamma = \{x_q | q \in V\}$.
2. $\psi(x_q, X) = c(q, X)$ *for all* $q \in V$ *and* $X \in \{\mathsf{true}, \mathsf{false}\}$.
3. I *is the collection of all inequalities of the forms given below:*

$$x_q \leq x_p \in I \textit{ for each } p \in D_q \textit{ if } r(q) = \text{AND} \tag{2}$$

$$\sum_{p \in D_q} x_p \geq x_q \in I \textit{ if } r(q) = \text{OR} \tag{4}$$

We say that $L(W)$ *is* semi-induced *by* W*. Furthermore, by including the additional constraints:*

$$x_q = 1 \textit{ if } q \in S \tag{6}$$

we say that the resulting constraint system is semi-induced evidentially *by* W *and is denoted by* $\hat{L}_E(W)$*. (Properties associated with induced constraint systems are easily generalizable to semi-induced ones.)*

A semi-induced constraint system is simply an induced constraint system lacking top-down constraints. From Corollary 3.2, the set of possible solutions for $L(W)$ is a superset of the set of possible explanations for W.

Before we can introduce the next theorem, we must now present the definition of AND-DAGs.

DEFINITION 3.10. *An* AND-DAG *is a* WAODAG *whose nodes which are labeled* OR *have at most indegree one. Construct* $W' = (G', c', r', S)$ *from* W *by removing all but one of the parent from every* OR *- node. Now, remove from* W'*, all nodes and associated edges which are not reachable from any evidence node in* S*. The resulting* AND-DAG W' *is said to be* induced *by* W*.*

PROPOSITION 3.5. *Let* W' *be an* AND-DAG *induced by* W*. For any truth assignment* e*, if* $e(p) = \mathsf{true}$ *for all nodes* p *in* W'*, then* e *is an explanation for* W*.*

THEOREM 3.6. *An optimal 0-1 solution for* $\hat{L}_E(W)$ *can be transformed into a best explanation for* W *in* $O(|E|)$ *steps if* $c(p, \mathsf{false}) \leq c(p, \mathsf{true})$ *for all nodes* p *in* V*.*

For the transformation, see the proof of Theorem 3.6 in Appendix. In general, transforming a 0-1 optimal solution for $\hat{L}_E(W)$ *requires at most* $2|E|$ *steps.*

COROLLARY 3.7. *An optimal 0-1 solution for* $\hat{L}_E(W)$ *is a best explanation for* W *if* $c(p, \mathsf{false}) < c(p, \mathsf{true})$ *for all nodes* p *in* V*.*

From the above theorem, a best explanation for W can be found by solving a smaller linear program.

Intuitively, we note that the information required to find an optimal 0-1 solution is propagated from true assignments which originate from the evidence nodes and thus, results in a bottom-up fashion of processing. In terms of our constraint system, constraints need only be sensitive to the information from one direction, namely from the evidence nodes. As we shall see later in this section, Corollary 3.7 will demonstrate certain enhancements and improvements which can be made to our approach.

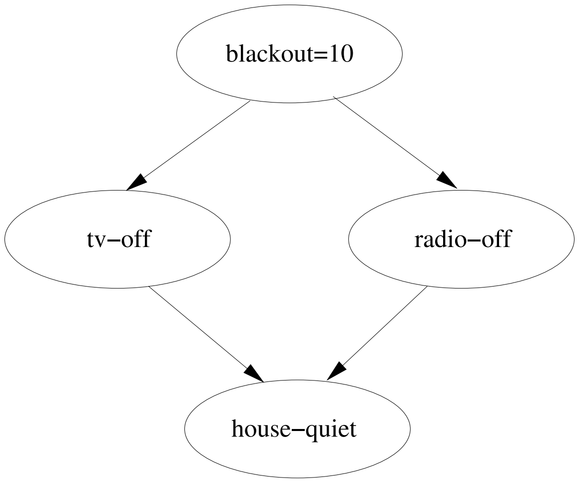


FIG. 3.2. In this simple WAODAG, the OR-node house-quiet is the observed evidence. blackout is the only hypothesis available.

PROPOSITION 3.8. *If $\hat{L}_E(W) = (\Gamma, I, \psi)$ is semi-induced evidentially by W, then*

$$|I| = |\{(p,q) \in E | r(p) = \text{AND}\}| + |V_O| + |S|$$

where V_O is the subset of all nodes in V which are labeled OR *and have nonzero outdegree.*

Many other types of improvements may also be employed. Some arise from the intimate knowledge of our problem like the one above while others are techniques used for general linear programming problems (see Sections 3.5 and 3.6).

Although only WAODAGs are used in the preceding discussions, one can easily generalize our linear constraint satisfaction approach to arbitrary boolean gate-only networks.

3.3 Branch and Bound

As we mentioned in the previous section, the solution to a WAODAG induced linear program need not consist of strictly 0s and 1s. For example, consider the simple WAODAG in Figure 3.2. (Note that we will use the terms constraint systems and linear programs interchangeably throughout this section.)

From this WAODAG, the following linear program is generated from our semi-induced constraint system $L = (\Gamma, I, \psi)$:

$$\begin{aligned} H &= 1 \\ H &\leq R + T \\ B &\geq R \\ B &\geq T \end{aligned}$$

and has objective function:

$$\Theta_L(s) = s(B)\psi(B, \mathsf{true}) + (1 - s(B))\psi(B, \mathsf{false})$$

where $H, R, T, B \in \Gamma$ respectively stand for house-quiet, radio-off, tv-off and blackout. Furthermore, assume $\psi(B, \mathsf{false}) = 0$ and $\psi(B, \mathsf{true}) > 0$.

We can easily show that the solution which minimizes the objective function is as follows: $H = 1$, $R = .5$, $T = .5$, and $B = .5$ with $\Theta_L(s) = \psi(B, \mathsf{true})/2$. We call B a *shared node* in our WAODAG. An OR-node such as H with assigned value strictly greater than its parent is called a *divide node*. (It is easy to show that either an OR-node is a divide node, or that all of its parent are 0 or the same value as the OR-node.)

Looking closely at Figure 3.2, we could easily remedy this problem by introducing the constraint house-quiet $\leq$ blackout. This new constraint reflects the fact that the value for house-quiet is ultimately determined by blackout. Simple patches like this one could be used to prevent this type of split node problem. However, most are non-trivial to identify and repair.

With small linear programs like the one above, using a brute force technique of simply trying each possible assignment maybe feasible. Of course, the run time grows exponentially with respect to the size of the problem.[8]

The technique to be presented avoids the necessity of searching the entire solution space by utilizing the lower bound computed by the linear program. This is a standard technique used in many domains to speed up processing time.

The basic idea is as follows: To find an optimal 0-1 solution, we solve a sequence of linear programs. This sequence can be represented by a tree where

[8] See integer programming techniques [59, 36, 39].

each node in the tree is identified with a linear program that is derived from the linear programs on the path leading to the root of the tree. The root of the tree is identified with the linear program induced by our constraint system. The linear programs along the nodes of the tree are generated using the following schema: Consider s_0, the optimal solution to our initial linear program denoted lp_0. If s_0 is a 0-1 solution, then we are finished. Otherwise, we choose some non-integral variable x in s_0 and define two new problems lp_1 and lp_2 as descendants of lp_0. lp_1 is identical to lp_0 except for the additional constraint $x = 1$, and lp_2 is identical to lp_0 except for the additional constraint $x = 0$. Note that the two new problems do not have s_0 as their optimal solutions. Since we are looking for a 0-1 assignment, the optimal 0-1 solution must satisfy one of the additional constraints. The two new nodes just defined are called *active nodes* and the variable x is called the *branching variable.*

Next, we choose one of the problems identified with an active node and attempt to solve it. It is not necessary to run a complete simplex method on the linear program. Using methods such as the *dual simplex algorithm* [59, 39], information is utilized in an incremental manner from other runs resulting in a quick and efficient computation. If the optimal solution is not a 0-1 solution, then two new problems are defined based on the current linear program. These new problems contain all the constraints of the parent problem plus the appropriate additional one.

When a 0-1 solution is found for some active node, the value of its objective function is compared against the current best, if any. The current best is the 0-1 solution which has the lowest cost of all the 0-1 solutions found so far. Thus, if the cost of the new solution is better than the current best, it is then used to prune those active nodes whose computed lower bounds exceed this value. This solution also now becomes the current best solution.

Branching continues in this manner until there are no active nodes in the tree. At the end, the current best solution is guaranteed to be the optimal 0-1 solution.

This technique is generally classified as a *branch and bound* technique in the area of *integer linear programming* [59, 39]. Also, it can be applied to any constraint system.

NOTATION. We denote an active node by an ordered pair (I, l) where I is a set of linear constraints and l is the value of the optimal solution for the associated linear program.

ALGORITHM 3.1. *Given a constraint system $L = (\Gamma, I, \psi)$, find its optimal 0-1 solution.*

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

To solve Figure 3.2, we choose tv-off to be our first branching variable. Following Algorithm 3.1 above, we generate two new linear programs:

$$\begin{aligned} L_1: \quad H &= 1 \\ H &\leq R+T \\ B &\geq R \\ B &\geq T \\ T &= 0 \end{aligned}$$

and

$$\begin{aligned} L_2: \quad H &= 1 \\ H &\leq R+T \\ B &\geq R \\ B &\geq T \\ T &= 1 \end{aligned}$$

both with objective function:

$$\Theta_L(s) = s(B)\psi(B, \mathsf{true}) + (1 - s(B))\psi(B, \mathsf{false})$$

. We first note that tv-off is now a fixed value in both L_1 and L_2. Since the original optimal value of tv-off was .5, both L_1 and L_2 now exclude this possibility which effectively eliminates the original optimal solution from their respective feasible solutions space. From simple observation, we find that the optimal solutions for L_1 and L_2 are $\{H = R = B = 1, T = 0\}$ for L_1 and $\{H = T = B = 1, R = 0\}$ for L_2. The cost for both assignments is $\psi(B, \mathsf{true})$. We can easily show that both assignments are optimal 0-1 solutions.

In this algorithm, two points were left deliberately vague: the choice of the next active node and the choice of branching variable. Several different options exist for both.

For the choice of the next active node, we have the following:

N1. Depth-first search of the branch and bound tree.

N2. Breadth-first search of the branch and bound tree.

N3. Choose the active node whose parent node has the best lower bound.

N4. Choose the active node whose parent's solution s^{opt} is *closest* to a 0-1 solution according to

$$\sum_{x \in \mathrm{lp}_k} \min\{s^{opt}(x), (1 - s^{opt}(x)\}.$$

For the choice of the next branching variable:

V1. Choose only variables corresponding to hypotheses nodes since a 0-1 assignment to these variables guarantees a 0-1 assignment throughout the remaining variables.

V2. Order the choice of variables such as shared nodes, then divide nodes, then hypothesis nodes, and so on.

V3. Choose the variable x which minimizes

$$\{\min\{s^{opt}(x), (1 - s^{opt}(x))\}\}.$$

V4. Choose the variable x which maximizes

$$\{\min\{s^{opt}(x), (1 - s^{opt}(x))\}\}.$$

V5. Choose the variable which has maximum associated cost.

V6. Use any combination of the above.

Obviously, many other techniques exist for making our choices, some based on the knowledge of our problem and others based on general techniques. Combinations of active node heuristics N1 and N3 together with branching variable heuristics V2 and V3 seem most promising.[9]

3.4 Experimental Results

We performed two experiments to measure the efficiency of our linear constraint satisfaction approach. The first involves a real application of our technique to solve cost-based abduction problems created by the story understanding system WIMP [22, 23]. This allows us to make a comparison against the search-style heuristic described in [4] to solve these graphs. Our second experiment involves

[9]Techniques V1 and V2 seem to work best when values can propagate in a top-down fashion. Thus, using the induced constraint system as opposed to the semi-induced constraint system is desired.

testing our approach on randomly generated WAODAGs as a gauge on how well the technique applies to the general class of WAODAGs. Also, WAODAGs larger than those found in the first experiment are used.

For both experiments, we employed active node method N1 and branching variable technique V3 above.

3.4.1 Experiment #1

WIMP is a natural language story comprehension system for parsing and understanding written English sentences [22, 23]. It utilizes belief networks to perform the necessary abductive inference tasks necessary to solve problems like pronoun reference and word-sense disambiguation. The belief networks can then be transformed into equivalent cost-based abductions problems as shown in [7].

The algorithm for determining the minimal cost proof in [4] is based on a best-first search of the WAODAG. The basic idea is that one starts with the partial proof consisting only of the evidence nodes in the WAODAG and then creating alternative partial proofs.[10] In each iteration, a partial proof is chosen to be expanded. It is expanded by adding some new nodes and edges to the existing partial proof which takes into consideration how one of its goals can be achieved locally according to the current partial proof and nearby nodes. This continues until all the goals (such as evidence) are satisfied and results in a minimal cost proof. How this is actually done is outlined in [7].

Naturally, the success of this algorithm depends on having a good heuristic function for deciding which partial proof should be worked on next. Furthermore, the heuristic function must be admissible to guarantee that the first proof generated is the minimal cost proof. Efficient heuristics have been difficult to find. One of the few basic admissible heuristics that has been used is *cost-so-far* in [7, 73]. Simply put, the partial proofs are weighted according to the costs of the hypotheses which they currently contain. A much more efficient heuristic has been found which utilizes a more sophisticated cost estimator [4]. The difficulty in the cost-so-far approach is that no estimation on the "goodness" of the partial

[10] A *partial proof* is a subgraph of the WAODAG.

proof can be made until hypothesis nodes have been reached. In brief, the new heuristic propagates the costs of the hypothesis nodes down the network to give some estimation of the "goodness" of each partial proof. The admissibility of this heuristic is guaranteed by the special care the heuristic takes in expanding partial proofs. (For precise details and the admissibility proof of this heuristic, see [4].)

Comparing our linear constraint satisfaction approach to the search heuristic is not an obvious task. The method and intuitions behind solving linear programming problems is radically different from those involved in best-first search. Our only course of action seems to be in making a direct CPU timing comparison. However, it is necessary to guarantee some invariance in our measure such as with respect to programming language used and internal algorithm optimization independent of the algorithm itself.

We find that a common factor in the run times of each approach is a complexity measure on the number of edges in the WAODAG. An expansion of a partial proof necessitates the traversal of the graph along its edges. The final minimal cost proof is a traversal such that each OR-node in the proof has exactly one outgoing edge and each AND-node includes all of its outgoing edges in the original WAODAG. Similarly for our constraint systems, Proposition 3.4 stipulates that the number of constraints required to solve a WAODAG is roughly the number of edges in the WAODAG.[11] And the run time of algorithms like the Simplex method to solve these systems is a function of the number of constraints.

We compare the efficiency of both the search heuristic and our linear constraint satisfaction approach by studying the runtime rate of growths. In essence, given a collection of ordered pairs of the form

$$(\text{WAODAG } \textit{complexity}, \textit{CPU usage}) \equiv (\textit{Number of } \text{WAODAG } \textit{edges}, \textit{CPU usage})$$

representing the runtime statistics, we attempt to perform a least-squares fit of the data on the function $t = e^{a+bx}$ where t is the CPU seconds used and x is the number of edges in the WAODAG to be solved. We can now compare the relative efficiency of each approach by comparing against the constant b for both fits.

[11]Generally, the number of edges is some multiple of the number of nodes.

	Nodes	*Edges*	*Hypotheses*	OR-*nodes*
Min	7	12	2	3
Max	158	375	41	59
Average	42.54	93.49	10.88	16.76
Median	34	75	9	14
Total	5955	13089	1523	2346

TABLE 3.1. WIMP WAODAG summary.

In this experiment, 140 WAODAGs generated by WIMP ranging in size from 7 nodes to 158 nodes and from 12 edges to 375 edges were presented to both approaches. Table 3.1 summarizes the set of WAODAGs generated by WIMP for our experiment. Figures 3.3 and 3.4 show the semi-logarithmic plot of our timings for the WIMP heuristic and for our linear constraint satisfaction approach.

We found that in our linear constraint satisfaction approach, roughly 61% of the WAODAGs generated by WIMP were solved using only linear programming without resorting to branch and bound. Furthermore, the number of active nodes actually used during branch and bound cases were only a small fraction of the total number of nodes involved. On average for the branch and bound cases alone, the number of active nodes was 6.2% of the nodes in the graph, and overall, the average number for all WAODAGs was 2.4%.

Performing a least-squares fit on the timings gives us the following

$$\begin{array}{lcl} \text{WIMP } \textit{heuristic} & : & e^{-5.32+.0245x} \\ \textit{Constraint system} & : & e^{-3.80+.0187x} \end{array}$$

which serves to verify some of our expectations on the growth rate of our linear constraint satisfaction approach as compared to the search technique.

Although we have just shown that our approach is better than the search heuristic found in WIMP, the best exponential fit does not actually describe our timings very well. Consider the logarithmic plot of our linear constraint satisfaction approach in Figure 3.4.

As we can clearly see, our linear constraint satisfaction approach actually exhibits a subexponential growth rate. By further attempting to fit our data to

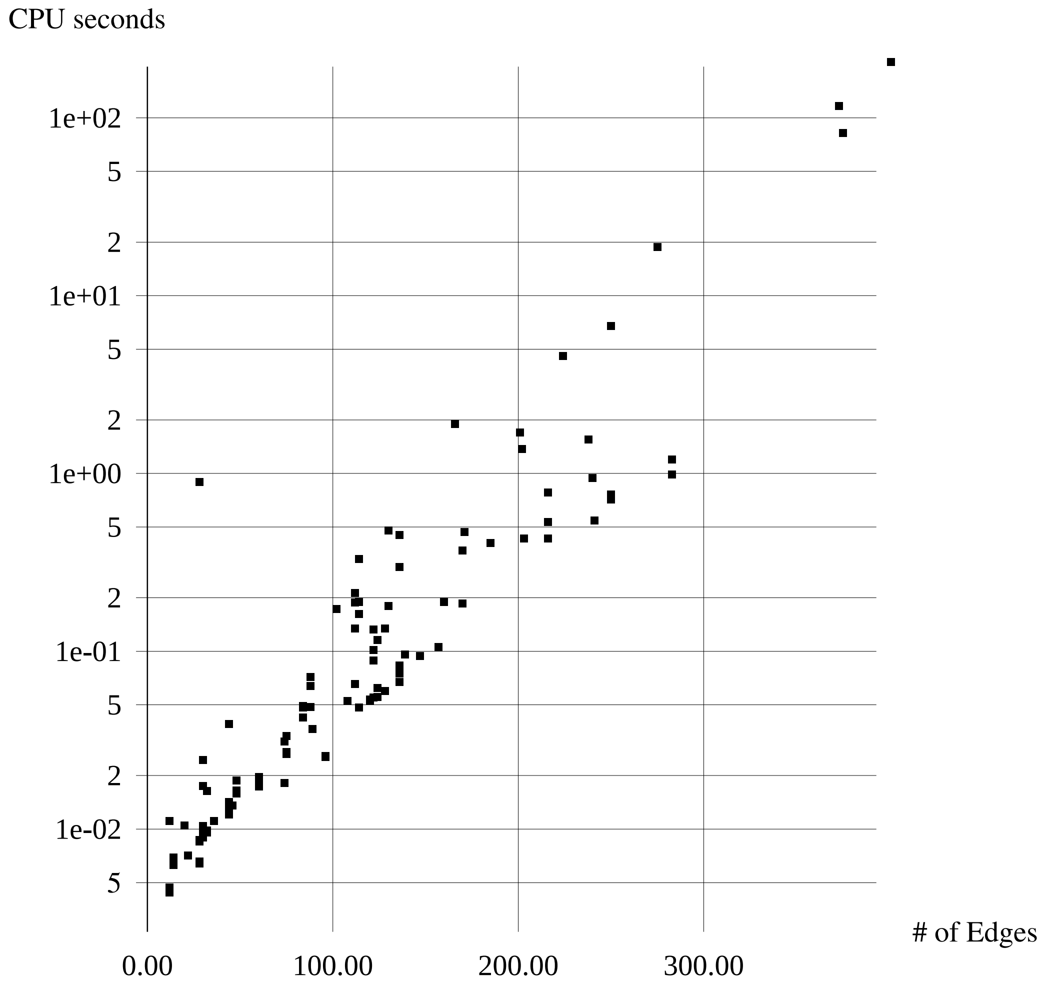


FIG. 3.3. Semi-logarithmic plot of WIMP heuristic timings.

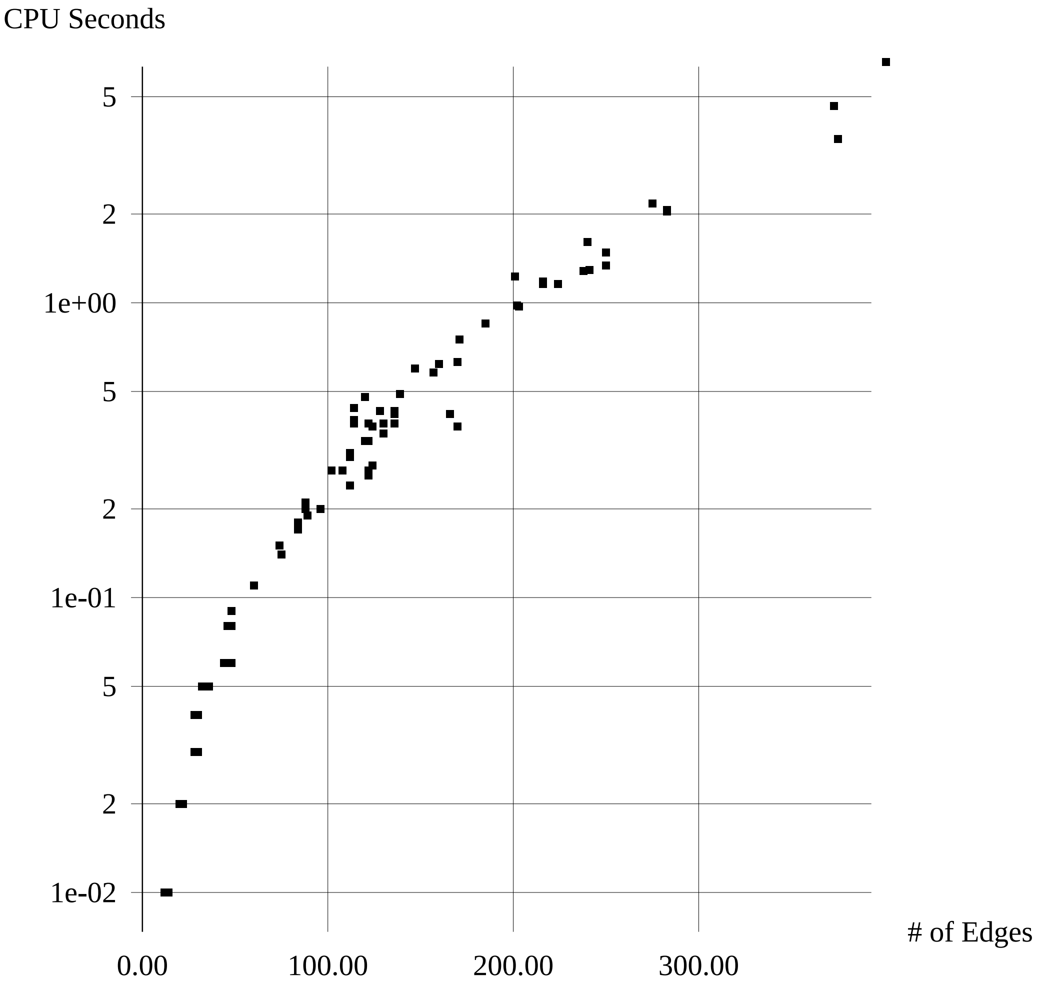


FIG. 3.4. Semi-logarithmic plot of linear constraint satisfaction timings.

ax^b, we get $.0001371x^{1.6484}$ as our growth curve. To show that the polynomial fit is better, we compare the two least square error fits, that is,

$$\sum_{e \in \Lambda} |\chi_e - F(e)|^2$$

where Λ is the set of all WAODAGs, χ_e is the amount of CPU seconds taken to solve WAODAG e and $F(e)$ is the amount of time predicted by the least-squares fit. Taking the error of the exponential fit and dividing it by the error of the polynomial fit, we roughly find a 10000% improvement of the polynomial fit over the exponential. When we attempted to perform a polynomial fit on the search heuristic, we found that the error actually tripled. Although the search heuristic is slightly faster than our approach on the very small (in terms of edges) WAODAGs, our approach seems to be quite fast and practical at solving all the WAODAGs generated by WIMP.

3.4.2 Experiment #2

Our purpose in this experiment is to further test the efficiency of our linear constraint satisfaction approach when faced with more general and larger WAODAGs than those generated by WIMP. In particular, we are interested in whether the subexponential growth rate exhibited for the WIMP generated WAODAGs remains to be the case for these randomly generated graphs.

100 WAODAGs were generated ranging from 36 to 387 nodes and from 39 to 699 edges. They were generated randomly[12] by first determining the number of nodes n from 1 to 400 and then instantiating said nodes. Next, the number of edges from n to 800 to be included in this graph is determined and the edges were randomly instantiated between two nodes.[13] Finally, hypothesis nodes are identified and are arbitrarily assigned some non-negative cost. Table 3.2 summarizes the set of randomly generated WAODAGs for this experiment.

We found that 97% of the randomly generated WAODAGs were solved using only linear programming without branch and bound.[14] Performing a least-squares

[12]By random, we mean uniform distribution.

[13]To guarantee that our resulting graph is acyclic, we initially imposed a random topological ordering on the nodes.

[14]We are currently investigating why this differs so much from WIMP generated WAODAGs.

	Nodes	*Edges*	*Hypotheses*	OR-*nodes*
Min	36	39	8	7
Max	387	699	154	135
Average	224.2	328.4	74.2	75.6
Median	225	323	76	79
Total	22424	32844	7423	7559

TABLE 3.2. Random WAODAG summary.

exponential fit gives us $e^{-5.49+.0614x}$.

Consider the logarithmic plot of our linear constraint satisfaction approach in Figure 3.5. Again, we can clearly see that our linear constraint satisfaction approach actually exhibits a subexponential growth rate. By further attempting to fit our data to ax^b, we get $.0079188x^{2.0308}$ as our growth curve. Again, the error fit actually improved roughly 2300%.

3.4.3 Discussion

From our two experiments above, our linear constraint satisfaction approach seems very promising. A very surprising result is that the optimal solution found for the linear program without branch and bound was either already a 0-1 optimal solution or a very close approximation as indicated by the relatively small number of active nodes used. Furthermore, due to the incremental nature of branch and bound, the additional computational effort required beyond solving the initial linear program was rather small as we could use methods such as the dual simplex algorithm.

Instinctively, we would have guessed that the bulk of our problems would have centered around the branch and bound process since it seems unlikely that our linear program should have an optimal solution which is also integral. We are currently attempting to understand why this is the case. However, it seems to be a very difficult problem.

It has been frequently observed that matching and set covering problems on graphs are very amenable to linear programming formulations in that they very

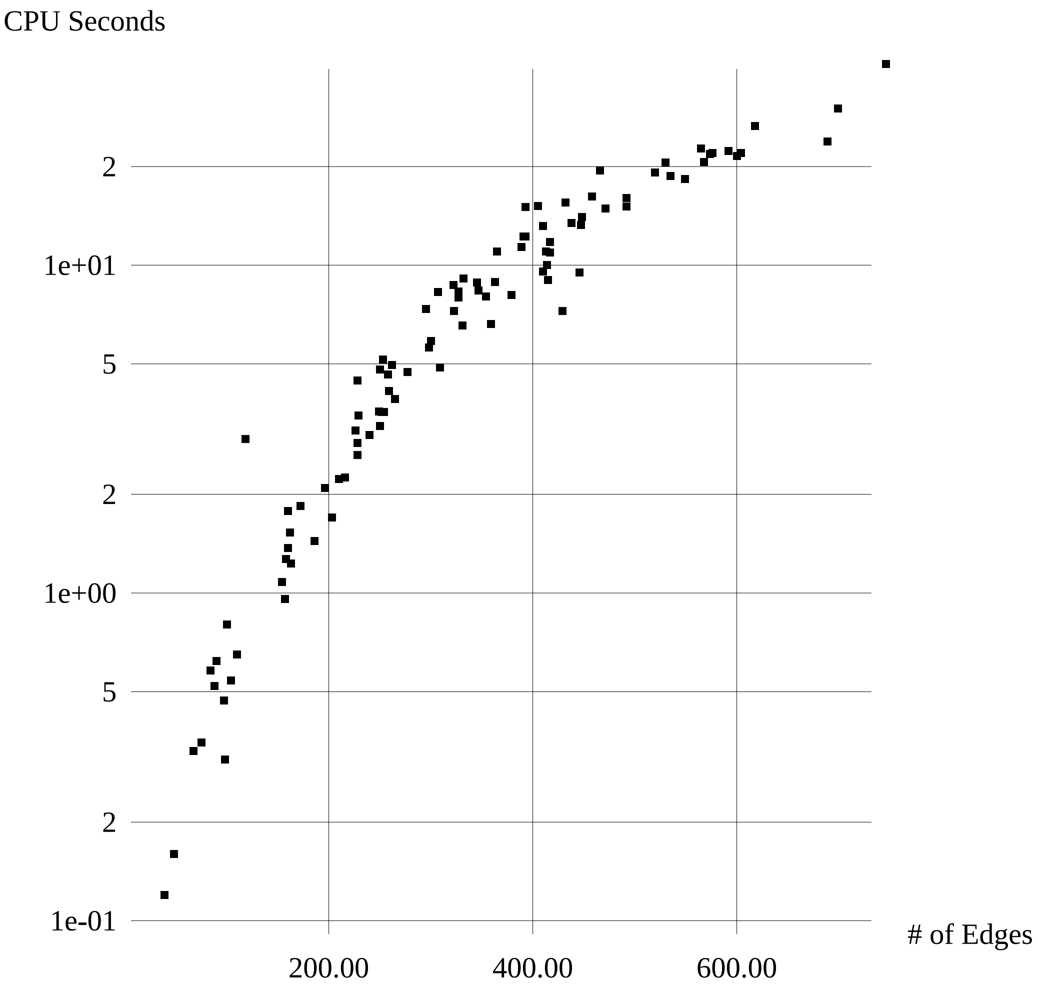


FIG. 3.5. Logarithmic plot of linear constraint satisfaction timings on random WAODAGs.

often have integral optimal solutions [18]. The abduction model for medical diagnoses presented in [42] is such a set-covering approach. Thus, the associated linear programs to solve problems using this model will generally have integral optimal solutions. This set-covering approach can be straightforwardly modeled using cost-based abduction where the rigid graphical structures play a key role in defining the solution spaces of the associated linear programs.

We also point out that in the original formulation of cost-based abduction presented in [7], the proof of NP-completeness was accomplished by transforming the *vertex covering problem* into cost-based abduction, thus strongly suggesting a close relationship between cost-based abduction and set covering.

3.5 Domain-Dependent Optimization

Since the success of branch and bound techniques depends on the ability to prune the active nodes early and as many as possible, we observe that pruning occurs whenever we have a 0-1 solution for the linear program. Although not every linear program results in a 0-1 solution, it is possible to build a 0-1 solution from the non-integer optimal solution. In fact, the construction is fairly straightforward and computationally cheap.

THEOREM 3.9. *Let s be the optimal solution of some* WAODAG *semi-induced linear program. Construct a variable assignment s' from s by changing all non-zero values in s to 1. s' is a 0-1 solution for the constraint system.*

For both WAODAG induced as well as semi-induced constraint system L, we consider the following construction from s: s' is constructed from s by changing all nonzero hypothesis node values in s to 1. Now, enforcing the boolean gate-like properties of the WAODAG, we propagate the boolean values from the hypothesis nodes up this boolean graph.

THEOREM 3.10. *s' constructed from s above is a 0-1 solution for L.*

On another point, we note that like traditional graph-based search heuristics, the constraints approach starts from scratch in its quest for the optimal solution. If by some fashion, we could incorporate a "good" explanation, then the gains are obvious. Finding a "good" explanation is generally quick and simple. Many fast non-admissible search heuristics will generate decent explanations.

In the rest of this subsection, we provide an approach for incorporating initial explanations. We will show theoretically how this approach naturally fits into our constraints formulation.

Let $W = (G, c, r, S)$ be a WAODAG and assume we have an explanation e. An immediate piece of information we can infer from e is that we should not consider any other explanations which are worse than e, that is, has a higher cost. We can use this additional cost information to help constrain our space of possible solutions for the evidentially induced $L_E(W) = (\Gamma, I, \psi)$. We can simply restrict our solution space to those solutions whose costs are less than or equal to $C(e)$. Basically, we introduce the following new constraint in I:

$$\sum_{x_q \in \Gamma} \{s(x_q)\psi(x_q, \mathsf{true}) + (1 - s(x_q))\psi(x_q, \mathsf{false})\} \leq C(e). \tag{8}$$

We can easily see that the left hand portion corresponds to our objective function. This now eliminates any assignments which incur a higher cost.

We now consider a more explicit use of e.

A collection of linear constraints defines a *polyhedral convex set.*[15] In 2-space, $\Re^2$, this would refer to any polygons whose interior angles are less than 180° such as the regular polygons. In $\Re^3$, this would correspond to solid objects which have planar faces and no "indentations" such as cubes and pyramids.

When we attempt to maximize (minimize) a linear cost function within the polyhedral, we are guaranteed that an optimal solution can be found at a "vertex", more properly called an *extreme point*, of the polyhedral. The Simplex method uses this fact and performs a hill-climbing search on the extreme points. It proceeds from extreme point to extreme point increasing (decreasing) the cost function. Eventually, an extreme point will be reached in which there are no other extreme points which can improved the cost function. This point will be an optimal solution.

Before Simplex can proceed, it must have an initial extreme point. Typically called *two-Phase Simplex*, the first Phase involves a search for an initial point and

[15]"Polyhedral" refers to the fact that the boundaries of the constraint set are hyperplanes.

then continues to the Phase described above. The search procedure performed in Phase I is very similar to that of Phase II. It also proceeds from point to point until it finds an extreme point. We can immediately see that the first Phase can be avoided if we can obtain and incorporate an initial solution cheaply.

We now very briefly describe the Simplex method. Consider the following minimization problem:

$$\begin{array}{rl}
\text{Minimize} & g(x_1, \ldots, x_n) = c_1x_1 + \ldots + c_nx_n - d \\
\text{subject to} & a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \geq b_1 \\
& a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \geq b_2 \\
& \vdots \\
& a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \geq b_m \\
& x_1, \ldots, x_n \geq 0.
\end{array}$$

Now, let $t_1, \ldots, t_m \geq 0$ be such that

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n - b_1 &= t_1 \\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n - b_2 &= t_2 \\
&\vdots \\
a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n - b_m &= t_m.
\end{aligned}$$

$t_1, \ldots, t_m$ are said to be *slack* variables in that they "take up the slack" in the inequalities to make them equalities. With these new variables, we can easily see that we have an underconstrained system of linear equations, that is, we have more variables than equations. Thus, we can reformulate our problem as follows:

$$\begin{array}{rl}
\text{Minimize} & g(x_1, \ldots, x_n) = c_1x_1 + \ldots + c_nx_n - d \\
\text{subject to} & a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n - b_1 = t_1 \\
& a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n - b_2 = t_2 \\
& \vdots \\
& a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n - b_m = t_m \\
& t_1, \ldots, t_m \geq 0 \\
& x_1, \ldots, x_n \geq 0.
\end{array}$$

From these linear equations, the space of feasible solutions is in fact a polyhedral convex set. We call the system above a *canonical minimization problem.*

An important observation for Simplex is that if we choose m of the variables in the above problem and solve them in terms of the remaining n variables, we can possibly find an extreme point for the polyhedral. Looking at the above problem, we see that the slack variables, $t_1, \ldots, t_m$ are already solved in terms of the remaining variables. To see if we have an extreme point handy, we simply assign the remaining variables to 0 and see what the resulting m variable values become. In this case, $t_i = -b_i$ for $i = 1, \ldots, m$. If this assignment satisfies all the constraints, then we have an extreme point. All that is necessary to check for satisfaction is to see if the t_i's are non-negative for all i. (We call this the *feasibility test.*) The m variables are called the *basic variables* and the remaining n are called the *non-basic variables.*

Clearly, if the assignment above is an extreme point, we can begin with Phase II of Simplex. Regardless of whether we are in Phase I or Phase II of Simplex, the basic technique for point to point movement is the same. Our current point is determined by setting the nonbasic variables to zero and the basic ones appropriately. Next, choose a basic variable and a nonbasic variables. We then perform a simple linear transformation which makes the basic to nonbasic and vice versa. This will construct a new linear program "equivalent" to the original and by appropriately setting and clearing variables, we can extract a new point.

When we have an initial solution, incorporating it into the problem should allow us to avoid Phase I of Simplex. Mentioned earlier above, Simplex determines whether it should proceed with Phase I by using the feasibility test. Basically, our goal is to pass the feasibility and hence avoid Phase I.

Given a WAODAG $W = (G, c, r, S)$, let $L_E(W)$ be the evidentially induced constraint system and e be an explanation for W. From our previous theorems in Section 3.2, $s[e]$ is a 0-1 solution for $L_E(W)$. Let $L_E(W)$ be rewritten in the canonical minimization problem above. We incorporate $s[e]$ as follows: For each real variable x_p assigned 1 in $s[e]$, replace it by the expression $1 - y_p$ in all the equations and the objective function. Basically, we have defined that $x_p \equiv 1 - y_p$. We can easily rewrite this new system into canonical form again. Since nothing else has changed, the slack variables $t_1, \ldots, t_m$ are still the basic variables. We know set the nonbasic variables to 0 and compute the values for the basic variables. Notice that setting y_p to 0 actually says that x_p is set to 1. Furthermore, since $s[e]$ is an extreme point to begin with, the basic variables must be assigned non-negative values! Thus, we have successfully incorporated an initial solution e into our problem and Simplex will proceed immediately to Phase II of its computations.

3.6 Optimization Results (Initial Solutions)

In our experiment, we compared our original results against the new approach on a collection of WAODAGs. The WAODAGs were again generated by WIMP, a natural language story comprehension system for parsing and understanding written English sentences [3, 22, 23].

We generated initial solutions for each of the WAODAGs by using a quick (unfortunately, non-admissible) heuristic outlined in [4]. The basic idea is that one starts with the partial proof consisting only of the evidence nodes in the WAODAG and then creating alternative partial proofs.[16] In each iteration, a partial proof is chosen to be expanded. It is expanded by adding some new nodes and edges to the existing partial proof which takes into consideration how one of its goals can be achieved locally according to the current partial proof and nearby nodes. This continues until all the goals (such as evidence) are satisfied and results in an explanation, not necessarily the minimal one though.

As we outlined in Section 3.5, we incorporate the initial solutions. We then compare the new results (New A) against the original in terms of actual CPU usage and number of extreme points visited. We also tested against a further modification of our approach by changing (8) to be strictly less than (New B). This effectively rules out considering the given explanation and any other explanations of equal cost. If our new system of equations has no feasible solution, then we know that the given explanation was already a best explanation.

We begin by looking at the run-times of all three approaches. WIMP generated 136 WAODAGs ranging in size from 7 to 158 nodes and from 12 to 375 edges. Table 3.3 describes the WAODAGs.

[16] A *partial proof* is a subgraph of the WAODAG.

	Nodes	*Edges*
Min	7	12
Max	158	375
Average	42.91	94.21
Median	34	75
Total	5836	12812

TABLE 3.3. Summary of WIMP WAODAGs.

	Original	*New A*		*New B*	
Min	0.01	0.01		0.00	
Max	4.25	4.14		4.99	
Average	0.32	0.24		0.23	
Median	0.13	0.07		0.07	
Total	43.77	32.27	26.27%	30.61	30.06%
			% Improvement with New A		% Improvement with New B

TABLE 3.4. Summary of run-time results.

In our experiment, we found that our new method decreased the amount of CPU time required by about 26.27% where as the strict inequality addition further improved it to 30.06%. Table 3.4 summarizes our findings.

Furthermore, we compared against the number of extreme points visited by all three methods. We found that our new method reduced the number of pivots[17] by about 30.42% and the strict inequality addition by 29.43%. Table 3.5 summarizes our findings.

From our data we see that while the strict inequality method runs faster than the non-strict method, it seems to require slightly more pivots. To understand this seemingly strange anomaly, we again look closely at Simplex and the polyhedral space of solutions defined by the constraints. When we are at an extreme point

[17]Each movement from one point to the next is called a *pivot* in linear programming terminology.

	Original	*New A*		*New B*	
Min	8	7		7	
Max	405	369		448	
Average	61.85	43.03		43.65	
Median	44	29		27	
Total	8411	5852	30.42%	5936	29.43%
			% Improvement with New A		% Improvement with New B

TABLE 3.5. Summary of extreme point results.

in the polyhedral, we can commence directly with Phase II. Hill-climbing can be applied since we have a clear idea of which direction to proceed in. Unfortunately, when we have a point outside the polyhedral, knowing how to get to an extreme point is much less clear. Just looking at a polygonal area in $\Re^2$, we can see that there are a huge number of possible directions to approach the polygon depending upon where the current point is. Hence, unlike the distinct search pattern involved in Phase II, Phase I is somewhat different. In general, specialized optimizations only applicable to Phase I renders its computations much faster than those of Phase II. Having said all this, we can now explain the pivots anomaly from our data. Simply put, the strict inequality used in the second method made our initial solution infeasible. Thus, we had to go back and perform Phase I. Although more pivots were required (which included branch and bound nodes), the computational time for the process was much smaller for these pivots as compared to the pivots used in Phase II for the first optimization method.

3.7 Alternative Explanations

In abductive explanation, having alternative explanations is often useful and sometimes necessary such as in medical diagnosis. For example, suppose I have the following symptoms: continual vomiting, acute stomach pains, acute back pains, diarrhea and yellowing of skin and eyes. One very likely diagnosis is Hepatitis B since this seems to mainly involve the liver. A less likely diagnosis is extensive gall stones in the gall bladder which also affects the liver and exhibits the same symptoms. However, without proper treatment, the gall stone condition can lead to perforation of the gall bladder and extremely serious complications. Thus, it is insufficient to simply select the more likely explanation of Hepatitis B without considering the possibility of gall stones. Furthermore, having the 2nd best, 3rd best, and so on, can provide a useful gauge on the *quality* of the best explanation. In this section, we present techniques for extracting alternative explanations in order of their associated costs [54, 57].

To generate the alternative explanations, we solve a sequence of constraint systems. This sequence consists of constraint systems each of which are derived from the constraint systems earlier in the sequence. The initial constraint system is the original constraint system which determines the first optimal solution. The subsequent constraint systems are generated using the following schema: Consider $L_1 = (\Gamma, I_1, \psi)$, our initial constraint system. Let s_1 be the optimal 0-1 solution of L_1. We define a new problem L_2 as the successor of L_1. L_2 is identical to L_1 except for the additional constraint

$$\sum_{x \in \Gamma} F(s_1, x) \leq |\Gamma| - 1$$

where for each $x \in \Gamma$,

$$F(s_1, x) = \begin{cases} x & \text{if } s_1(x) = 1 \\ (1 - x) & \text{if } s_1(x) = 0 \end{cases}$$

Note that the new problem does not have s_1 as its optimal 0-1 solution since the variable assignment would violate the new constraint.

Let s_2 be the optimal 0-1 solution, if any, to L_2. This will be the second best 0-1 solution. To continue the search for the next best explanation, we simply define a successor to the last constraint system, in this case, L_2. When the current constraint system does not yield any solution, all possible explanations have been generated and we are finished.

ALGORITHM 3.2. *Given a constraint system $L = (\Gamma, I, \psi)$, generate all the 0-1 solutions for L in order of cost.*

1. *(*Initialization*) Set $I_1 := I$, $L_1 := (\Gamma, I_1, \psi)$ and $k := 1$.*

2. Compute the optimal 0-1 solution for L_k. If there is no feasible solution, then go to step 7. Otherwise, let s_k be the solution.

3. $k := k + 1$.

4. Let $I_k := I_{k-1} \cup c_{k-1}$ where c_{k-1} contains the single constraint

$$\sum_{x \in \Gamma} F(s_{k-1}, x) \leq |\Gamma| - 1$$

where for each $x \in \Gamma$,

$$F(s_{k-1}, x) = \begin{cases} x & \textit{if } s_{k-1}(x) = 1 \\ (1 - x) & \textit{if } s_{k-1}(x) = 0 \end{cases}$$

5. Let $L_k := (\Gamma, I_k, \psi)$.

6. Go to step 2.

*7. (*Solutions*) Print $s_1, s_2, \ldots, s_{k-1}$.*

The above method we have just described can be classified as a *cutting plane method* in operations research [36, 59, 39]. Since each derived constraint system differs only in an additional constraint from some previously solved problem, efficient incremental techniques such as the dual simplex method can be applied here in a fashion similar to the one used in the branch and bound algorithm.

THEOREM 3.11. *Constraint system L_n in Algorithm 3.2 determines the n-th best 0-1 solution for L.*

The algorithm we have just presented can be applied to any constraint system regardless of whether it was WAODAG induced or not.

Although we may simply use the alternative generation method in Algorithm 3.2 to generate all possible explanations, there are certain situations where generating all of them may not be particularly desirable. Consider the following scenario: Suppose I decide to phone my friend Tony at the office. After several rings, no one has answered the phone. Without any additional information, I conclude that Tony is not at the office. However, suppose that we also know that Tony usually disconnects the phone and takes naps in the office. Furthermore, Tony has been known to be able to sleep in any environment whenever he desires

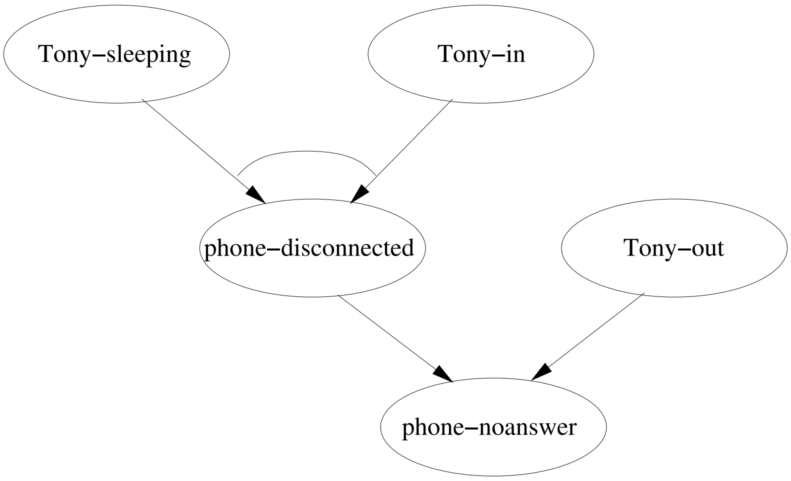


FIG. 3.6. Tony's office habits.

and is as likely to be awake as asleep at any time. This implies that for the hypothesis that Tony is awake, the difference in the cost of being true and it being false is negligible. For the sake of argument, let the cost of being true be equal to the cost for false. If we look at our original explanation that Tony is not in the office, we must augment it with our guess as to whether he is asleep or not. With our assumptions, there is no way to choose between asleep and awake since the costs of the augmented explanations are equal. However, since Tony is not in the office, the hypothesis involving his consciousness has no impact towards explaining the observation (see Figure 3.6).

If the algorithm first chooses that Tony is asleep, then the next alternative would be the same set of assignments except for Tony being awake. However, this new alternative explanation is uninteresting and does not contribute anything more to gauging our explanations. It may be the case that we may run into an overly large number of these types of uninteresting explanations. We now proceed to present an approach to deal with this problem.

DEFINITION 3.11. *Given a* WAODAG $W = (G, c, r, S)$ *where* $G = (V, E)$ *and* $H \subseteq V_H$*, an explanation* e *for* W *is said to be* consistent *with* H *iff for all* h *in* H*,* $e(h) = \mathsf{true}$. *The* base set $H(e)$ *of* e *is the subset of* V_H *consisting of all* h *in*

V_H *where* $e(h) = \mathsf{true}$.

In WAODAGs, finding the best explanation is tantamount to finding the best set of hypotheses we need to assume.

DEFINITION 3.12. *The* support-set $K(e)$ *of an explanation* e *is the set consisting of all nodes* m *in* V *such that* $e(m) = \mathsf{true}$.

PROPOSITION 3.12. *For every explanation* e *for* W, $H(e) = K(e) \cap V_H$.

The following propositions follow immediately from the properties found in WAODAGs:

PROPOSITION 3.13. *Let* e_1 *and* e_2 *be explanations for* W. $H(e_1) = H(e_2)$ *iff* $K(e_1) = K(e_2)$.

PROPOSITION 3.14. *Let* e *be an explanation for* W. *For each* $H(e) \subseteq H \subseteq V_H$, *there exists an explanation* e' *for* W *such that* $H(e') = H$.

THEOREM 3.15. *Let* e_1 *and* e_2 *be explanations for* W.

1. $H(e_1) \subseteq H(e_2)$ *iff* $K(e_1) \subseteq K(e_2)$.
2. $H(e_1) \subset H(e_2)$ *iff* $K(e_1) \subset K(e_2)$.

THEOREM 3.16. *There exists a 1-1 and onto mapping between* 2^{V_H} *and the set of all possible truth assignments for* W.

THEOREM 3.17. *If* e *is an explanation for* W, *then there exists at least* $2^{|V_H - H(e)|}$ *explanations for* W *which are consistent with* $H(e)$.

In general, we see that there are an exponential number of explanations for a given WAODAG. However, from Theorem 3.17, it seems that the majority of these explanations are formed from a possibly small number of "simpler" explanations which utilizes smaller numbers of hypotheses. The following question naturally arises: Do these additional explanations provide any new or important information? The answer is "yes" only if the explanation has a lower cost associated with it than the "simpler" explanation. Typically, this will occur if we consider negative costs.

We now consider a class of WAODAGs we call *monotonic*. In general, most of the WAODAGs used in systems like WIMP will fall into this class. In this class, "simpler" explanations will imply lower costs. We will provide a method for extracting only the "simple" explanations.

DEFINITION 3.13. *A* WAODAG W *is* monotonic *iff for every explanations* e_1 *and* e_2 *for* W, $K(e_1) \subseteq K(e_2)$ *implies* $C(e_1) \leq C(e_2)$. W *is* strictly monotonic *iff* W *is monotonic, and for every explanations* e_1 *and* e_2 *for* W, $K(e_1) \subset K(e_2)$ *implies* $C(e_1) < C(e_2)$.

PROPOSITION 3.18. *If* $c(v, \mathsf{true}) \geq c(v, \mathsf{false})$ *for all* v *in* V, *then* W *is monotonic. If* $c(v, \mathsf{true}) > c(v, \mathsf{false})$ *for all* v *in* V, *then* W *is strictly monotonic.*

THEOREM 3.19. *A* WAODAG W *is monotonic iff for every explanations* e_1 *and* e_2 *for* W, $H(e_1) \subseteq H(e_2)$ *implies* $C(e_1) \leq C(e_2)$. W *is strictly monotonic iff* W *is monotonic, and for every explanations* e_1 *and* e_2 *for* W, $H(e_1) \subset H(e_2)$ *implies* $C(e_1) < C(e_2)$.

Proposition 3.18 and Theorem 3.19 together demonstrate that in a monotonic WAODAG, "simpler" explanations are preferred due to the lower associated costs. The assumption of monotonicity is reasonable in many cases as pointed out by [7] and characterized in [3]. Thus, we are only interested in the "simplest" explanations. Our goal is to generate these explanations in order of cost without having to consider the remaining exponential number of explanations.

DEFINITION 3.14. e *is* cardinal *iff there does not exist an explanation* e' *such that* $H(e') \subset H(e)$.

Intuitively, a cardinal explanation is among the "simplest" of explanations we wish to consider. Our notion of cardinal explanations is equivalent to the notion of *irredundancy* found in *parsimonious covering theory* for modeling medical diagnosis [43]. A cover is said to be irredundant if none of its proper subsets is also a cover.

THEOREM 3.20. *If* W *is strictly monotonic, then any best explanation for* W *is cardinal.*

All the definitions given above involving WAODAGs can be carried over to WAODAG induced constraint systems.

Similar to the previous method, the best cardinal explanation, 2nd best, 3rd best, etc. may be generated by constructing a sequence of constraint systems $L_1, L_2, \ldots$ Each constraint system in the sequence is derived from those earlier in the sequence. The initial constraint system L_1 is the one which determines the optimal 0-1 solution s_1. L_2 is a new constraint system identical to L_1 except for the additional constraint

$$\sum_{x_q \in H(s_1)} x_q \leq |H(s_1)| - 1.$$

We can show that L_2 will determine the 2nd best cardinal solution. We can continue to find the other cardinal solutions by adding new constraints.

ALGORITHM 3.3. *Given a strictly monotonic* WAODAG *induced constraint system* $L = (\Gamma, I, \psi)$, *generate all the cardinal 0-1 solutions for* L *in order of cost.*

1. *(*Initialization*) Set* $I_1 := I$, $L_1 := \Gamma, I_1, \psi)$ *and* $k := 1$.
2. *Compute the optimal 0-1 solution for* L_k. *If there is no feasible solution, then go to step 7. Otherwise, let* s_k *be the solution.*
3. $k := k + 1$.
4. *Let* $I_k := I_{k-1} \cup c_{k-1}$ *where* c_{k-1} *contains the single constraint*

$$\sum_{x_q \in H(s_{k-1})} x_q \leq |H(s_{k-1})| - 1.$$

5. *Let* $L_k := (\Gamma, I_k, \psi)$.
6. *Go to step 2.*
7. *(*Solutions*) Print* $s_1, s_2, \ldots, s_{k-1}$.

LEMMA 3.21. *Let* W *be strictly monotonic. If* s_n *is the optimal 0-1 solution for the constraint system* L_n *in Algorithm 3.3, then* s_n *is a cardinal solution for* L.

THEOREM 3.22. *Let* W *be strictly monotonic. The constraint system* L_n *in Algorithm 3.3 determines the* n*-th best cardinal solution.*

Although this algorithm works only for W being strictly monotonic, we can modify any non-strictly monotonic problem to make it applicable. In essence, the strict monotonicity simply implies that we should always have a preference for a false assignment over a true assignment. By introducing an arbitrarily small positive difference between the cost for true and the cost for false in the original problem, we can now determine the cardinal solutions of the new problem which turns out to be identical to those of the original.

In our above formulation, we defined monotonicity only for WAODAG induced systems. Thus, the accompanying algorithm is not applicable to all constraint systems. We now provide a generalization of monotonicity for constraint systems.

In the rest of this section, $L = (\Gamma, I, \psi)$ will represent a constraint system and Q a subset of Γ.

DEFINITION 3.15. *Given a 0-1 solution s for L, the* support-set $K(s)$ *of s is the set of all $x_q \in \Gamma$ where $s(x_q) = 1$.*

DEFINITION 3.16. *Q is* dominant *iff for every 0-1 solutions s_1 and s_2 for L, $(K(s_1)) \cap Q) = (K(s_2) \cap Q)$ implies $K(s_1) = K(s_2)$. Q is* strongly dominant *iff Q is dominant and no proper subset of Q is dominant.*

PROPOSITION 3.23. *Γ is dominant.*

PROPOSITION 3.24. *Let Q' be a subset of Γ such that $Q \subseteq Q'$. If Q is dominant, then Q' is also dominant.*

Intuitively, a set of variables Q is dominant if any assignment of 0,1 values to Q completely determines the assignment of 0,1 values to the remaining variables. For WAODAGs, the only strongly dominant set is the set corresponding to V_H. Similarly for dags representing boolean gate functions, the only strongly dominant set is the set corresponding to the set of all nodes with zero indegree. For diagnostic problem [43], the only strongly dominant set is the set corresponding to the set of all disorders.

DEFINITION 3.17. *L is Q-*monotonic *iff for every 0-1 solutions s_1 and s_2 for L, $(K(s_1) \cap Q) \subseteq (K(s_2) \cap Q)$ implies $\Theta_L(s_1) \leq \Theta_L(s_2)$. L is* strictly Q-monotonic *iff L is Q-monotonic and for every 0-1 solutions s_1 and s_2 for L, $(K(s_1) \cap Q) \subset (K(s_2) \cap Q)$ implies $\Theta_L(s_1) < \Theta_L(s_2)$.*

Intuitively, monotonicity says that the more information we require in an explanation, the more costly it will be.

THEOREM 3.25. *Let C be the set of variables that appear in Θ_L with non-zero coefficient.*

1. *If C is a subset of Q and all the coefficients in Θ_L are positive, then L is Q-monotonic.*
2. *If $C = Q$ and all the coefficients in Θ_L are positive, then L is strictly Q-monotonic.*

DEFINITION 3.18. *A 0-1 solution s for L is Q-*cardinal *iff there exist no 0-1 solution s' for L such that $(K(s') \cap Q) \subset (K(s) \cap Q)$ and $\Theta_L(s') \leq \Theta_L(s)$.*

Q is the set of variables whose assignment we are mainly interested in. For WAODAGs, the set of variables corresponding to the hypothesis nodes would be such a set.

THEOREM 3.26. *Let L be strictly Q-monotonic. Every optimal 0-1 solution for L is Q-cardinal.*

Our interests now lie in generating the Q-cardinal solutions for L in order of cost. This can be accomplished by a method similar to Algorithm 3.3. We construct a sequence of constraint systems $L_1, L_2, \ldots$ which respectively compute the best cardinal solution, 2nd best, 3rd best, and so on. Each constraint system in the sequence is derived from those earlier in the sequence. The initial constraint system L_1 is the one which determines the optimal 0-1 solution s_1. L_2 is identical to L_1 except for the additional constraint

$$\sum_{x_q \in (K(s_1) \cap Q)} x_q \leq |K(s_1) \cap Q| - 1.$$

We can show that L_2 will determine the representative of the 2nd best Q-cardinal solution and continue to find the others by adding new constraints.

ALGORITHM 3.4. *Given a set of variables Q which is dominant and a strictly Q-monotonic constraint system $L = (\Gamma, I, \psi)$, generate all the cardinal 0-1 solutions for L in order of cost.*

1. *(*Initialization*) Set $I_1 := I$, $L_1 := (\Gamma, I_1, \psi)$ and $k := 1$.*

2. *Compute the optimal 0-1 solution for L_k. If there is no feasible solution, then go to step 7. Otherwise, let s_k be the solution.*
3. $k := k + 1$.
4. *Let $I_k := I_{k-1} \cup c_{k-1}$ where c_{k-1} contains the single constraint:*

$$\sum_{x_q \in (K(s_{k-1}) \cap Q)} x_q \leq |K(s_{k-1} \cap Q| - 1.$$

5. *Let $L_k := (\Gamma, I_k, \psi)$.*
6. *Go to step 2.*
7. *(*Solutions*) Print $s_1, s_2, \ldots, s_{k-1}$.*

LEMMA 3.27. *Let L be strictly Q-monotonic. If s_n is the solution of the constraint system L_n in the Algorithm 3.4, then s_n is a Q-cardinal solution for L.*

THEOREM 3.28. *Let Q be a dominant set and let L be strictly Q-monotonic. The constraint system L_n in the Algorithm 3.4 determines the n-th best Q-cardinal solution for L.*

As we can clearly see, by setting Q to be the set of variables corresponding to the hypothesis nodes of a strictly monotonic WAODAG W. We get the following results:

THEOREM 3.29. *Let W be a strictly monotonic* WAODAG *and $L(W)$ be the constraint system induced by W. Let Q be the set of variables corresponding to the hypothesis nodes of W in $L(W)$. We have the following:*

1. *Q is strictly dominant.*
2. *$L(W)$ is strictly Q-monotonic.*

3.8 Consistency

Finally, in this subsection, we consider extensions to cost-based abduction which can be naturally modeled under our constraint systems. For example, WAODAGs do not consider negation. This can be categorized in a class of problems we call *consistency*.

Consistency is concerned with the maintenance of relationships between two or more pieces of information. In the attempt to build sophisticated inference models, we often run into the difficulty of maintaining certain close relationships such as "(female a) iff $\neg$(male a)", or, from medical diagnosis, disorder d_1 is complementary to disorder d_2.

The formulation of WAODAGs in cost-based abduction can be modified to represent these relationships but the standard techniques for finding the best explanations may not be readily applicable.[18] As we shall see below, constraint systems can provide an intuitive and practical solution to handling these relationships.

We first consider the problem of negation. If we allow the existence of propositions p and q where p and q are derived (via rules) from a and $\neg a$, respectively, we have introduced a NOT-node in terms of boolean gate functions. However, given some partial truth assignment, arbitrarily assigning the remaining hypothesis nodes may violate some of the constraints.

Two approaches to handling negation in constraint systems are described below – the *implicit* formulation and the *explicit* formulation.

In the implicit formulation, when $\ldots a \ldots \Longrightarrow p$, no change is necessary and x_a is still used in the constraints. However, when $\ldots \neg a \ldots \Longrightarrow p$, $(1 - x_a)$ is used in place of x_a. For an example, see Figures 3.7 and 3.8.

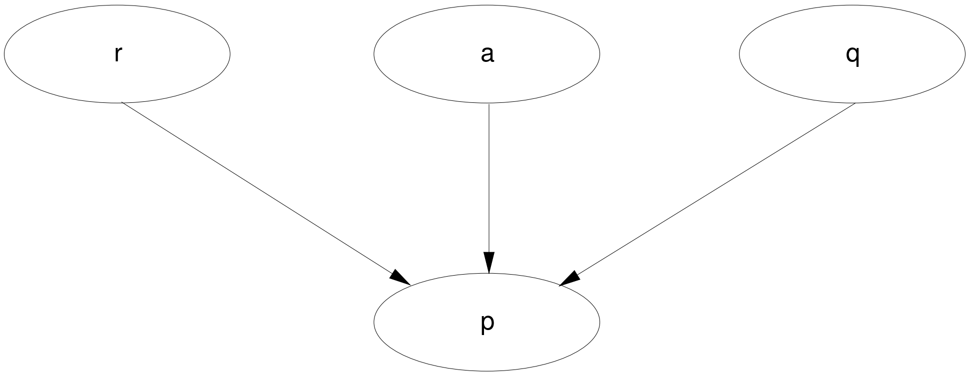
In the explicit formulation, a brand new variable $x_{\neg a}$ is used in place of x_a. (Referring to Figure 3.8, we replace "$(1 - x_a)$" with "$x_{\neg a}$".) In addition, the constraint $x_a = (1 - x_{\neg a})$ must be included in the constraint system.

We observe that the ability to handle negation in our approach permits us to easily generalize from WAODAGs to arbitrary boolean gate-only networks.

From negation, we can extend our formulation to handle the case involving *complementary* disorders mentioned earlier, where not both disorder d_1 and disorder d_2 can occur simultaneously. Actually, we can take this one step further.

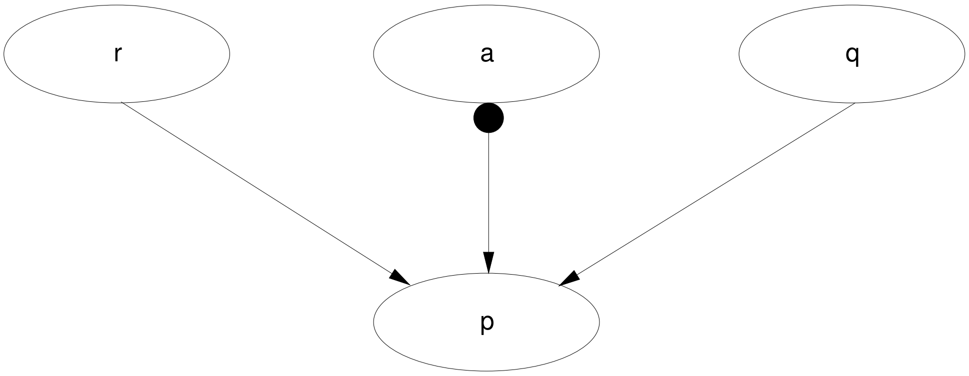
From Table 3.6, we can take any combination of the conditions and form a relationship between d_1 and d_2. We can easily see that the explicit formulation

[18] With the exception of brute force techniques which enumerate the possible solutions, current existing heuristics do not take consistency into account.



$$x_q + x_a + x_r \geq x_p$$
$$x_p \geq x_q$$
$$x_p \geq x_a$$
$$x_p \geq x_r$$

FIG. 3.7. Ordinary WAODAG.



$$x_q + (1 - x_a) + x_r \geq x_p$$
$$x_p \geq x_q$$
$$x_p \geq (1 - x_a)$$
$$x_p \geq x_r$$

FIG. 3.8. An AND/OR-graph with negation. A small circle of an arrow indicates negation of the parent.

Condition	*Constraint*
$\neg(d_1 \text{ and } d_2)$	$x_{d_1} + x_{d_2} \leq 1$
$\neg(\neg d_1 \text{ and } \neg d_2)$	$x_{d_1} + x_{d_2} \geq 1$
$\neg(d_1 \text{ and } \neg d_2)$	$x_{d_1} \leq x_{d_2}$
$\neg(\neg d_1 \text{ and } d_2)$	$x_{d_1} \geq x_{d_2}$

TABLE 3.6. d_1 and d_2 are disorders. The conditions we wish to have true are guaranteed by the addition of the associated constraint.

of the negation relationship is the combination of the first and second conditions in the table. We have shown how to handle two items. We can generalize this approach to handling various complicated relationships between three or more items.

4 Bayesian Networks

In cost-based abduction, information is arranged in a graphical form. The nodes and arcs of the graphs represent relationships between objects in the world. Similar to cost-based abduction, Bayesian networks have an underlying graphical foundation. The similarity of both approaches goes insofar as semantically defining nodes to correspond to some object or event and arcs to represent direct relationships between these nodes.

Bayesian networks provide an approach to modeling our knowledge in a probabilistic manner. Basically, objects and/or events are represented by random variables (abbreviated, r.v.s). We can then define the likelihood of a set of events by their joint probability. However, calculating the joint probability of a collection of random variables is often impossible unless you have additional information concerning the dependence relationships between each sets of random variables. Given a collection of r.v.s $A_1, A_2, \ldots, A_n$, we can rewrite their joint probability as

$$P(A_1 = a_1, \ldots, A_n = a_n) = \prod_{i=1}^{n} P(A_i = a_i | A_{i+1} = a_{i+1}, \ldots, A_n = a_n) \quad (9)$$

according to Bayes' theorem on conditional probabilities. However, we still require a large number of conditional probabilities of which most are not easily attainable.

Equation (9) suggests a topological ordering on the r.v.s. Bayesian networks consider this ordering as some sort of causality relationship. So, if an event A can cause an event B, then A should topologically dominate B. This would simply mean that if A and B were indexed in terms of $A_1, \ldots, A_n$, the index of A will be less than the index for B.

The strength of Bayesian networks lies in the assumption that for each r.v. A there exists a set of r.v.s $B_1, \ldots, B_k$ which topologically dominates A such that for any conditional probabilities

$$\begin{aligned} &P(A = a | B_1 = b_1, \ldots, B_k = b_k, C_1 = c_1, \ldots, C_m = c_m) = \\ &\quad P(A = a | B_1 = b_1, \ldots, B_k = b_k) \end{aligned} \quad (10)$$

where C_j topologically dominates A and $C_j \neq B_i$ for all $i = 1, \ldots, k$. We say that A is *conditionally dependent* on $B_1, \ldots, B_k$. Thus, instead of storing all the conditional probabilities required in (9), we can store a much more compact set according to (10). Furthermore, this provides us with a strong notion of causality for semantically constructing the knowledge-base of conditional probability tables. Now, we can compute the joint probability of $A_1, \ldots, A_n$ as follows:

$$P(A_1 = a_1, \ldots, A_n = a_n) = \prod_{i=1}^{n} P(A_i = a_i | A_{i_1} = a_{i_1}, \ldots, A_{i_j} = a_{i_j}) \qquad (11)$$

where

$$\{A_{i_1}, \ldots, A_{i_j}\} \subseteq \{A_{i+1}, \ldots, A_n\}$$

and A_i is conditionally dependent on $A_{i_1}, \ldots, A_{i_j}$.

From the above assumption of conditional independence, we can quickly transform the independence relationships into a graphical structure. In essence, we have a directed acyclic graph where each node represents a r.v. and the directed arcs represent topological dominance. A node A will be conditionally dependent on its immediate parents.

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

Assume that Mary used the following set of propositions:

$$\begin{aligned} \mathsf{rain} \vee \mathsf{sprinklers} &\implies \mathsf{street\text{-}wet} \\ \mathsf{rain} \vee \mathsf{sprinklers} &\implies \mathsf{lawn\text{-}wet} \\ \mathsf{lawn\text{-}wet} &\implies \mathsf{soil\text{-}moist} \\ \mathsf{soil\text{-}moist} &\implies \mathsf{roses\text{-}okay} \end{aligned}$$

Since the propositions explicitly indicate causality, we can again directly transform it into a graph. Now, by considering each logic variable as a r.v. with possible states of $\{\mathsf{true}, \mathsf{false}\}$, we can construct conditional probability tables for (11) which reflects our knowledge of the world (see Figure 4.1).

Bayesian networks have become an important tool in modeling probabilistic reasoning. The inherent representational power of these networks provides a

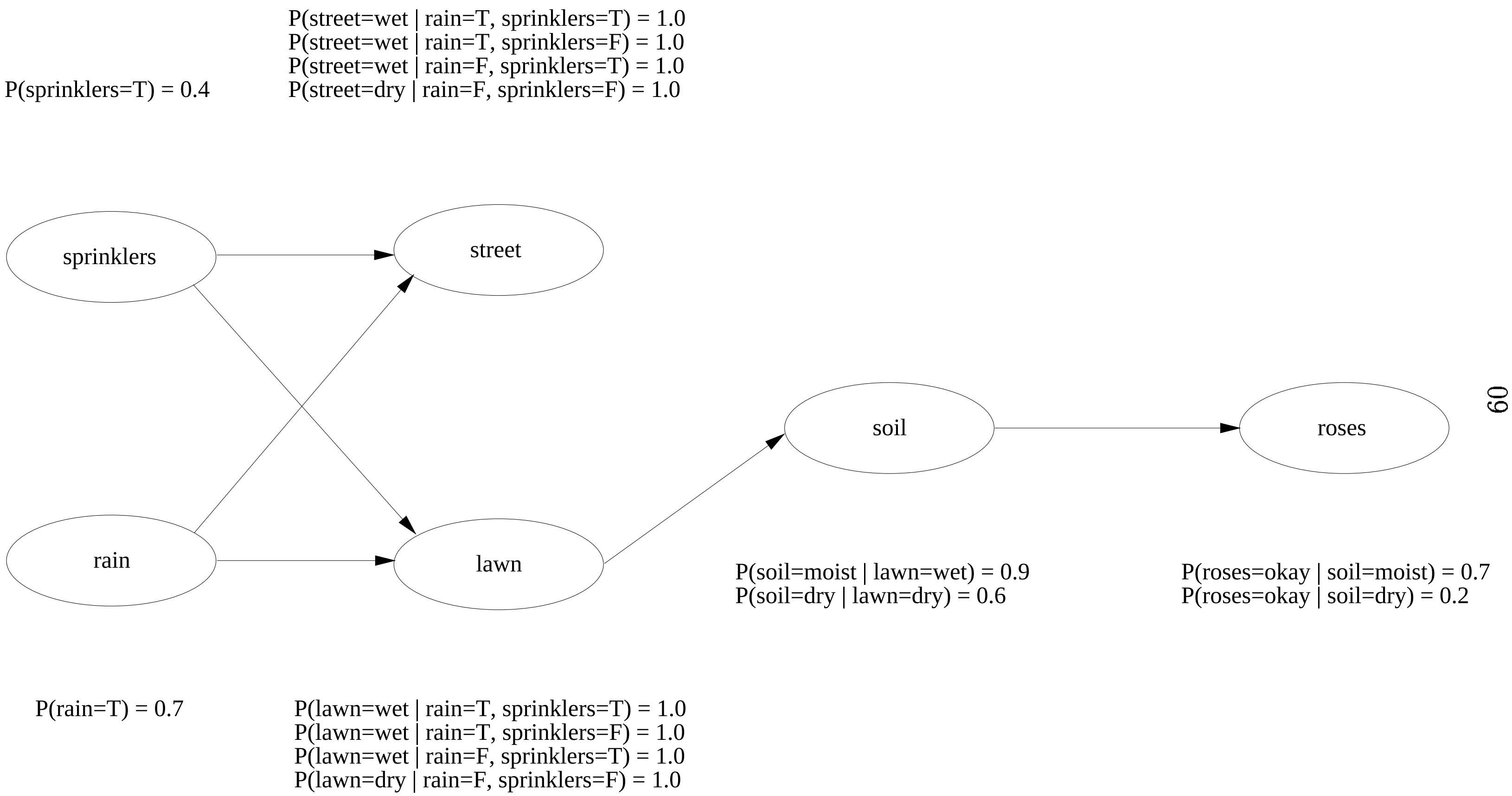


FIG. 4.1. Mary's Bayesian network.

very promising approach. In particular, *belief revision* in Bayesian networks is the process of finding the best interpretation for some given piece of evidence. This, of course, is a cornerstone of abductive explanation.

One algorithm for belief revision is given by Pearl in [41] which is based on a *message passing scheme*. However, except for simple networks such as polytrees, the method is rather complicated to apply. Also, as Pearl points out in Chapter 5 in [41], this algorithm cannot guarantee the generation of alternative explanations beyond the second best.

In this section, our goal is to apply our linear constraint satisfaction approach to *belief revision* as well as *belief updating* in Bayesian networks. This entails constructing a constraint system which is computationally equivalent to the Bayesian network.

4.1 Belief Revision

The goal of belief revision on Bayesian networks is to find an instantiation of all the random variables which will maximize their joint probability. When evidence is given to be explained, an instantiation must be sought to maximize the joint probability given the evidence. The instantiation which maximizes this probability is called the best explanation. This measure is Pearl's *most-probable explanation criterion* (MPE).

In this section, our goal is to apply our linear constraint satisfaction approach to belief revision in Bayesian networks. Although this could be done by first transforming the Bayesian network into a cost-based abduction graph [68] and then transforming the graph into a constraint system, a more natural and straightforward method will be given below. (See also [54, 57].) We will show how to directly transform a Bayesian network into an equivalent constraint system.

4.1.1 Constraints Formulation

We first observe that a Bayesian network can be completely described by a finite collection of random variables and a finite set of conditional probabilities based on the r.v.s.[19]

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

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

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

DEFINITION 4.1. *Given a Bayesian network $\mathcal{B} = (V, P)$, an* instantiation *is an ordered pair (A, a) where $A \in V$ and $a \in R(A)$. (An instantiation (A, a) is also denoted by $A = a$ and A_a.) A collection of instantiations w is called an* instantiation-set *iff are no two instantiations $(A, a), (A, a')$ in w such that $a \neq a'$.*

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

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

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

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

NOTATION. For each r.v. A and each a in $R(A)$, $v_{\mathsf{A_a}}$ is the set of all conditional probabilities in P of the form $P(A = a|C_1 = c_1, \ldots, C_n = c_n)$. For each r.v. A, we define $\mathsf{cond}(A)$ as follows: $B \in \mathsf{cond}(A)$ iff there exists a conditional probability in P of the form $P(A = a|\ldots, B = b, \ldots)$. Intuitively, $\mathsf{cond}(A)$ is the set of r.v.s which are the parents of A.

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

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

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

The goal of belief revision on Bayesian networks is to determine the complete instantiation-set which maximizes the associated probability under certain conditions. In general, these conditions, called *evidence*, imposes restrictions on what instantiations may be made. The instantiation-set satisfying the evidence with the highest probability is said to be the *most-probable explanation* for the evidence. We now formalize this as follows:

DEFINITION 4.4. *Given a Bayesian network* $\mathcal{B} = (V, P)$, evidence e *for* $\mathcal{B}$ *is said to be some instantiation-set for* $\mathcal{B}$.

DEFINITION 4.5. *Given instantiation-sets* w_1, w_2 *for a Bayesian network* $\mathcal{B}$, w_2 *is said to be* consistent *with* w_1 *iff* $w_1 \subseteq w_2$.

DEFINITION 4.6. *Given evidence* e *for* $\mathcal{B}$, *a complete instantiation-set* w *for* $\mathcal{B}$ *is an* explanation *for* e *iff* w *is consistent with* e. *Furthermore,* w *is said to be a* most-probable explanation *for* e *iff for all explanations* $w' \neq w$ *for* e, $P(w') \leq P(w)$.

Thus, the most-probable explanation for evidence e will be the explanation w^* which maximizes

$$P(w^*) = \max_{w \in W(e)} P(w)$$

where $W(e)$ is the set of all explanations for e.

Our basic approach in constructing a constraint system from a given Bayesian network is to represent and enforce the constraints that exist between any two or more random variables.

Given a Bayesian network $\mathcal{B} = (V, P)$, we construct a constraint system $L(\mathcal{B}) = (\Gamma, I, \psi)$ as follows:

1. For each random variable A in V, let $R(A) = \{a_1, \ldots, a_n\}$ and construct the variables $\mathsf{A}_{\mathsf{a}_1}, \ldots, \mathsf{A}_{\mathsf{a}_n}$ in Γ, set $\psi(\mathsf{A}_{\mathsf{a}_i}, \mathsf{false}) = \psi(\mathsf{A}_{\mathsf{a}_i}, \mathsf{true}) = 0$ and add the following constraint to I:

$$\sum_{i=1}^{n} \mathsf{A}_{\mathsf{a}_i} = 1. \tag{12}$$

2. For each random variable A and some a in $R(A)$, construct the following:
   (a) For each conditional probability $P(A = a | C_1 = c_1, \ldots, C_n = c_n)$ in $v_{\mathsf{A}_\mathsf{a}}$, construct a variable $q[\mathsf{A}_\mathsf{a} \mid \mathsf{C}_1 = \mathsf{c}_1, \ldots, \mathsf{C}_n = \mathsf{c}_n]$ in Γ such that (for notational convenience, we will denote $q[\mathsf{A}_\mathsf{a} \mid \mathsf{C}_1 = \mathsf{c}_1, \ldots, \mathsf{C}_n = \mathsf{c}_n]$ by q in the next two conditions)
      i. $\psi(q, \mathsf{false}) = 0$, $\psi(q, \mathsf{true}) = -\log(P(A = a | C_1 = c, \ldots, C_n = c_n))$, and,
      ii. Add the following constraints to I:

$$q \leq \mathsf{C}_{k_{\mathsf{c}_k}} \quad \text{for } k = 1, 2, \ldots, n. \tag{13}$$

   (b) Let $\Upsilon_{\mathsf{A}_\mathsf{a}}$ be all variables constructed by $v_{\mathsf{A}_\mathsf{a}}$ in (a). Add the following constraint to I:

$$\sum_{x \in \Upsilon_{\mathsf{A}_\mathsf{a}}} x = \mathsf{A}_\mathsf{a}\,. \tag{14}$$

Definition 4.7. *$L(\mathcal{B})$ constructed above is the constraint system* induced *by $\mathcal{B}$.*

Theorem 4.1. *Let $\mathcal{B} = (V, P)$ be a Bayesian network and $L(\mathcal{B}) = (\Gamma, I, \psi)$ be the constraint system induced by $\mathcal{B}$. Then*

1. *$|\Gamma| = |P| + \sum_{A \in V} |R(A)|$ and*
2. *$|I| = 2|V| + \sum_{p \in P} N(p) \leq 2|V| + |V||P| = (2 + |P|)|V|$ where $N(p)$ is the number of r.v.s which appear as conditionals in the probability.*

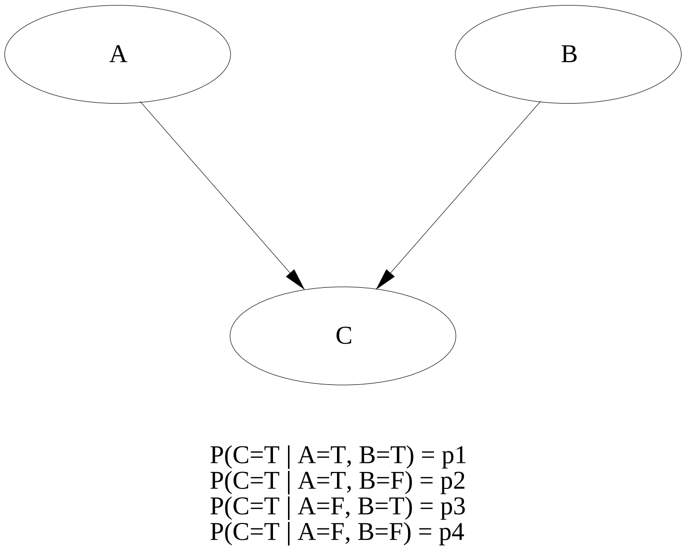


FIG. 4.2. Simple Bayesian network.

In our construction, (12) guarantees that any r.v. takes on exactly one value. (13) and (14) guarantees that the probability of any complete instantiation-set will be computed with the appropriate set of conditional probabilities. Variables of the form $q[\mathsf{A}_{\mathsf{a}} \mid \mathsf{C1}_{\mathsf{c}_1}, \ldots, \mathsf{C}n_{\mathsf{c}_n}]$ are called *conditional variables* in that they explicitly represent the dependencies between r.v.s and will be the mechanism for computing the probability for any instantiation-set.

For example, consider the simple Bayesian network in Figure 4.2. When we have the instantiations $\{A = \mathsf{true}, B = \mathsf{false}, C = \mathsf{true}\}$, its associated probability is $p_2 * p_5 * p_6$. In the induced constraint system, we expect our variables assignments to be $\mathsf{A}_{\mathsf{true}} = 1$, $\mathsf{B}_{\mathsf{false}} = 1$, $\mathsf{C}_{\mathsf{true}} = 1$, $q[\mathsf{C}_{\mathsf{true}} \mid \mathsf{A}_{\mathsf{true}}, \mathsf{B}_{\mathsf{false}}] = 1$, and all remaining variables to be 0. Since the only costs are associated with the variables $\mathsf{A}_{\mathsf{true}}$, $\mathsf{B}_{\mathsf{false}}$ and $q[\mathsf{C}_{\mathsf{true}} \mid \mathsf{A}_{\mathsf{true}}, \mathsf{B}_{\mathsf{false}}]$, the cost of this assignment is $-\log(p_5) - \log(p_6) - \log(p_2)$ which is equivalent to $-\log(p_2 * p_5 * p_6)$.

NOTATION. For each r.v. A, let $\Delta(A)$ be the set of variables in the induced constraint system constructed for A.

THEOREM 4.2. *Given a 0-1 solution s for $L(\mathcal{B})$, for each set of variables $\Delta(A)$, there exists some A_{a} in $\Delta(A)$ such that $\mathsf{A}_{\mathsf{a}} = 1$ and $\mathsf{A}_{\mathsf{a}'} = 0$ for all $\mathsf{A}_{\mathsf{a}'} \neq \mathsf{A}_{\mathsf{a}}$ in*

$\Delta(A)$.

THEOREM 4.3. *Given a 0-1 solution s for $L(\mathcal{B})$, for all variables $q[\mathsf{A_a} \mid \mathsf{C_1} = \mathsf{c_1}, \ldots, \mathsf{C}_n = \mathsf{c}_n]$, $q[\mathsf{A_a} \mid \mathsf{C_1} = \mathsf{c_1}, \ldots, \mathsf{C}_n = \mathsf{c}_n] = 1$ iff $\mathsf{A_a} = \mathsf{C}_{1_{\mathsf{c}_1}} = \ldots = \mathsf{C}_{n_{\mathsf{c}_n}} = 1$.*

Theorems 4.2 and 4.3 above verifies our expectations on the legitimate variable assignments. We must now show that calculations on the constructed constraint system are equivalent to those on the Bayesian network for belief revision.

Given a 0-1 solution s for $L(\mathcal{B})$, we can construct a complete instantiation-set w_s for $\mathcal{B}$ as follows: $s(\mathsf{A_a}) = 1$ iff $(A, a) \in w_s$. To convert from a complete instantiation-set to a 0-1 solution is slightly trickier. Given a complete instantiation-set w for $\mathcal{B}$, construct a 0-1 solution s_w for $L(\mathcal{B})$ as follows: $(A, a) \in w$ iff $s_w(\mathsf{A_a}) = 1$. For each conditional variable q in $\Upsilon_{\mathsf{A_a}}$, set the appropriate value according to w.

THEOREM 4.4. *If s is a 0-1 solution for $L(\mathcal{B})$, then w_s is an instantiation-set for $\mathcal{B}$.*

THEOREM 4.5. *If w is a complete instantiation-set for $\mathcal{B}$, then s_w is a 0-1 solution for $L(\mathcal{B})$.*

From Theorems 4.4 and 4.5, there is a 1-1 and onto mapping between 0-1 solution for the induced constraint system and complete instantiation-sets for the Bayesian network.

DEFINITION 4.8. *Let e be some evidence for $\mathcal{B} = (V, P)$. We construct $L_e(\mathcal{B}) = (\Gamma, I_e, \psi)$ from $L(\mathcal{B}) = (\Gamma, I, \psi)$ as follows: Let $I_e = I \cup I'$ where the constraint $\mathsf{A_a} = 1$ is in I' iff $(A, a) \in e$. We say that $L_e(\mathcal{B})$ is* induced *by $\mathcal{B}$* with evidence *e.*

PROPOSITION 4.6. $|I_e| = |I| + |e|$.

We can now show that all explanations for some given evidence e have corresponding 0-1 solutions for $L_e(\mathcal{B})$ and vice versa.

THEOREM 4.7. *If s is a 0-1 solution for $L_e(\mathcal{B})$, then w_s is an explanation for e.*

THEOREM 4.8. *If w is an explanation for e, then s_w is a 0-1 solution for $L_e(\mathcal{B})$.*

Theorems 4.7 and 4.8 above guarantees that the evidence properly restricts the set of possible 0-1 solutions we wish to consider. Now, we must show that the costs associated to each 0-1 solution are directly related to the probability of the corresponding instantiation-set.

For the following theorems, assume that L is induced by a Bayesian network $\mathcal{B}$, w is a complete instantiation-set for $\mathcal{B}$, and s is a 0-1 solution for $L(\mathcal{B})$.

THEOREM 4.9. $\Theta_L(s_r) = -\log(P(w))$.

Theorem 4.9 shows that our constraint system will properly calculate the probability of a complete instantiation-set for $\mathcal{B}$. Next, we must show that this calculation is sufficient for finding the most-probable explanation.

THEOREM 4.10. *There exists a constant α_e such that for all explanations w for e, $\Theta_{L_e}(s_w) = \alpha_e - \log(P(w|e))$.*

α_e in Theorem 4.10 above represents $-\log(P(e))$ which is a constant in all explanations (see proof of Theorem 4.10 in Appendix).

THEOREM 4.11. *w is a most-probable explanation for e iff s_w is an optimal 0-1 solution for $L_e(B)$.*

Theorem 4.10 guarantees that the probabilistic ordering of instantiation-sets is exactly reversed from the cost ordering imposed on 0-1 solutions. Furthermore, computing the cost for a 0-1 solution immediately determines the probability of its associated instantiation-set. With the transformation of belief revision problems to constraint systems, we can now solve them using the highly efficient tools and techniques applicable to constraint systems.

4.1.2 Alternative Explanations

In Section 3.7, we saw the necessity for generating alternative explanations in domains like medical diagnosis. Unfortunately, as we mentioned earlier, message-passing schemes are incapable of generating beyond the second best.

We now present a generation technique for Bayesian networks which is similar to the cutting plane approach in Section 3.7 [54, 54].

Again, our approach is to solve a sequence of constraint systems. This sequence consists of constraint systems each of which are derived from the constraint systems earlier in the sequence. The initial constraint system is the original constraint system which determines the first optimal solution. The subsequent constraint systems are generated using the following schema: Consider $L_1 = (\Gamma, I_1, \psi)$, our initial constraint system. Let s_1 be the optimal 0-1 solution of L_1. We define a new problem L_2 as the successor of L_1. L_2 is identical to L_1 except for the additional constraint

$$\sum_{x \in \tau} F(s_1, x) \leq |\tau| - 1$$

where for each $x \in \tau$ and $\tau = \{x | x \in V \text{ and } x \in \Delta(\mathsf{A}) \text{ for some r.v.} \mathsf{A}\}$,

$$F(s_1, x) = \begin{cases} x & \text{if } s_1(x) = 1 \\ (1 - x) & \text{if } s_1(x) = 0 \end{cases}$$

Note that the new problem does not have s_1 as its optimal 0-1 solution since the variable assignment would violate the new constraint.

Let s_2 be the optimal 0-1 solution, if any, to L_2. This will be the second best 0-1 solution. To continue the search for the next best explanation, we simply define a successor to the last constraint system, in this case, L_2. When the current constraint system does not yield any solution, all possible explanations have been generated and we are finished.

ALGORITHM 4.1. *Given a constraint system $L = (\Gamma, I, \psi)$, generate all the 0-1 solutions for L in order of cost.*

1. *(*Initialization*) Set $I_1 = I$, $L_1 = (\Gamma, I_1, \psi)$ and $k = 1$.*
2. *Compute the optimal 0-1 solution for L_k. If there is no feasible solution, then goto step 7. Otherwise, let s_k be the solution.*
3. *$k = k + 1$.*
4. *Let $I_k = I_{k-1} \cup c_{k-1}$ where c_{k-1} contains the single constraint:*

$$\sum_{x \in \tau} F(s_{k-1}, x) \leq |\tau| - 1$$

where for each $x \in \tau$ *and* $\tau = \{x | x \in V$ *and* $x \in \Delta(\mathsf{A})$ *for some r.v.* $\mathsf{A}\}$,

$$F(s_{k-1}, x) = \begin{cases} x & \textit{if } s_1(x) = 1 \\ (1-x) & \textit{if } s_1(x) = 0 \end{cases}$$

5. *Let* $L_k = (\Gamma, I_k, \psi)$.
6. *Go to step 2.*
7. *(*Solutions*) Print* $s_1, s_2, \ldots, s_{k-1}$.

The above method can again be classified as a *cutting plane method* in operations research [36, 59, 39]. Likewise, since each derived constraint system differs only in an additional constraint from some previously solved problem, incremental computational methods such as the dual simplex method can be applied here in a fashion similar to the one used in the branch and bound algorithm.

THEOREM 4.12. *Constraint system* L_n *in Algorithm 4.1 determines the n-th best 0-1 solution for* L.

4.1.3 Circumscribing Explanations and Focusing

Consider the following problem with most-probable explanation (MPE) criterion pointed out by Pearl in [41]: Suppose a medical test reveals that my friend Glenn has an 80% chance of being totally healthy. If you are not healthy, then you are going to die. Since Glenn believes himself to likely be healthy, he begins to daydream about going on vacation and not staying at home. Our Bayesian network can be represented by the following conditional probabilities on the r.v.s $\mathsf{Healthy}$, Dead and $\mathsf{Location}$:

$P(\mathsf{Healthy} = \mathsf{true}) = .8$
$P(\mathsf{Healthy} = \mathsf{false}) = .2$
$P(\mathsf{Dead} = \mathsf{true} \,|\, \mathsf{Healthy} = \mathsf{true}) = 0$
$P(\mathsf{Dead} = \mathsf{false} \,|\, \mathsf{Healthy} = \mathsf{true}) = 1$
$P(\mathsf{Dead} = \mathsf{true} \,|\, \mathsf{Healthy} = \mathsf{false}) = 1$
$P(\mathsf{Dead} = \mathsf{false} \,|\, \mathsf{Healthy} = \mathsf{false}) = 0$
$P(\mathsf{Location} = \mathsf{at\ home} | \, \mathsf{Healthy} = \mathsf{true}) = 0$
$P(\mathsf{Location} = \mathsf{at\ home} | \, \mathsf{Healthy} = \mathsf{false}) = 1$

$P(\text{Location} = \text{Barbados} | \text{Healthy} = \text{true}) = .1$
$P(\text{Location} = \text{Barbados} | \text{Healthy} = \text{false}) = 0$
$P(\text{Location} = \text{Caracas} | \text{Healthy} = \text{true}) = .1$
$P(\text{Location} = \text{Caracas} | \text{Healthy} = \text{false}) = 0$
... plus 8 more vacation spots with .1 / 0 conditional probabilities ...

Our most-probable explanation for this network is

$$\{\text{Healthy} = \text{false}, \text{Dead} = \text{true}, \text{Location} = \text{at home}\}$$

with probability .2! Thus, although Glenn is most likely healthy, the MPE criterion says that our best scenario is for him to be dead.

Probabilistically, we made the following computations: Let e be the evidence to be explained which in this case is the empty set. Let W be the set of all complete instantiation-sets for this network. Our goal is to find w^* such that $P(w^*|e) = \max_{w \in W} P(w|e)$, but, $P(w|e) = P(w|\phi) = P(w)$ for all w in W where ϕ is the empty set. Thus, $P(w^*) = \max_{w \in W} P(w)$.

Now,

$$\begin{aligned} &P(\text{Healthy} = h, \text{Dead} = d, \text{Location} = l) = \\ &\quad P(\text{Dead} = d, \text{Location} = l | \text{Healthy} = h) P(\text{Healthy} = h) = \\ &\quad P(\text{Dead} = d | \text{Healthy} = h) P(\text{Location} = l | \text{Healthy} = h) P(\text{Healthy} = h). \end{aligned}$$

For any scenario in which we are not dead, that is, $\text{Dead} = \text{false}$, we have the following probabilities:

$P(\text{Healthy} = \text{false}, \text{Dead} = \text{false}, \text{Location} = \text{at home}) = 0 * 1 * .2 = 0$
$P(\text{Healthy} = \text{false}, \text{Dead} = \text{false}, \text{Location} = \langle\text{vacation-spot}\rangle) = 0 * 0 * .2 = 0$
$P(\text{Healthy} = \text{true}, \text{Dead} = \text{false}, \text{Location} = \text{at home}) = 1 * 0 * .8 = 0$
$P(\text{Healthy} = \text{true}, \text{Dead} = \text{false}, \text{Location} = \langle\text{vacation-spot}\rangle) = 1 * .1 * .8 = .08$

Thus, Glenn's happy daydreaming about lovely vacation spots has ended in morbid thoughts of dying.

As we can clearly see, this result is mathematically correct although somewhat counterintuitive. As Pearl points out [41], this has much to do with what sort of information we are currently thinking about or focusing on. In the above case, since we started with information concerning our health, we naturally expect all our attention to shift heavily to thoughts involving being healthy. However, if we had been previously thinking about what is going to happen to our entire life in general, then the explanations of being unhealthy and dying seem to be not so unreasonable anymore.

From the computations above, we can make the following observation: Since multiplication of probabilities is a *monotonically decreasing* function [16] and is strictly decreasing when probabilities are less than 1, it is expected that the more probabilistic information we have, the smaller the joint probabilities will become. So, if Glenn continued daydreaming like about what he is going to do at each particular vacation spot, then any individual scenario in which he is alive has an even smaller probability. The problem of determining what the solution should be seems to be highly dependent on what we are currently focused on.

Pearl in [41] presents one possible conservative approach which suggests that what we are focusing on is precisely the evidence we wish to explain. Furthermore, the evidence should be the determining factor in what we consider as information which is relevant to any explanation we may be interested in. In the above scenario, since there is no evidence needing to be explained, no conclusions can be drawn. In probabilistic terms, this seems to suggest that random variables which are "irrelevant" to the evidence are not instantiated. This approach of only instantiating a subset of the random variables is currently being explored by [41] and [7, 67, 66].

In our constraint system formulation of Bayesian networks, not assigning all the random variables is tantamount to weakening the constraint that each random variable must have exactly one assignment,

$$\sum_{i=1}^{n} \mathsf{A}_{\mathsf{a}_i} = 1. \tag{15}$$

to the constraint that each random variable takes on "at most" one value,

$$\sum_{i=1}^{n} \mathsf{A}_{\mathsf{a}_i} \leq 1. \tag{16}$$

When the summation is zero, this means that the associated random variable takes on no values and is considered "ignored". By simply replacing constraints of the form (15) with those of (16), we can predict certain results on the possible instantiations of the random variables.

DEFINITION 4.9. *Let $\hat{L}(\mathcal{B})$ be the constraint system constructed from $L(\mathcal{B})$ by replacing all constraints of the form (15) with those of (16). We say that $\hat{L}(\mathcal{B})$ is* weakly induced *from $\mathcal{B}$.*

THEOREM 4.13. *Given a 0-1 solution s for $\hat{L}(\mathcal{B})$, for each set of variables $\Delta(A)$, there exists at most one variable A_a in $\Delta(A)$ such that $\mathsf{A}_\mathsf{a} = 1$ and $\mathsf{A}_{\mathsf{a}'} = 0$ for all $\mathsf{A}_{\mathsf{a}'} \neq \mathsf{A}_\mathsf{a}$ in $\Delta(A)$.*

THEOREM 4.14. *Given a 0-1 solution s for $\hat{L}(\mathcal{B})$, for all variables $q[\mathsf{A}_\mathsf{a} \mid \mathsf{C}_{1_{\mathsf{c}_1}}, \ldots, \mathsf{C}_{n_{\mathsf{c}_n}}]$,*

$$q[\mathsf{A}_\mathsf{a} \mid \mathsf{C}_{1_{\mathsf{c}_1}}, \ldots, \mathsf{C}_{n_{\mathsf{c}_n}}] = 1 \textit{ iff } \mathsf{A}_\mathsf{a} = \mathsf{C}_{1_{\mathsf{c}_1}} = \ldots = \mathsf{C}_{n_{\mathsf{c}_n}} = 1.$$

Theorems 4.13 and 4.14 show us that certain expectations on the legitimate variable assignments still hold.

We now show that this weakening of constraints results in computations using the circumscribing explanation approach suggested by Pearl [41].

DEFINITION 4.10. *Let d_1 and d_2 be instantiation-sets for $\mathcal{B} = (V, P)$. d_1 and d_2 are said to be* compliments *iff $\mathsf{span}(d_1) \cup \mathsf{span}(d_2) = V$ and $\mathsf{span}(d_1) \cap \mathsf{span}(d_2) = \phi$.*

NOTATION. We denote the set of all compliments of instantiation-set d by $X(d)$.

Intuitively, the compliment of an instantiation-set d is the set of all r.v.s not instantiated in d.

DEFINITION 4.11. *For any two instantiation-sets d_1 and d_2 for $\mathcal{B}$ such that $\mathsf{span}(d_1) \cap \mathsf{span}(d_2) = \mathsf{span}(d_1 \cap d_2)$, we define the* join *of d_1 and d_2 to be the instantiation-set d constructed as follows:*

1. *For all r.v.s A in $\mathsf{span}(d_1)$, $(A, a) \in d$ iff $(A, a) \in d_1$.*
2. *For all r.v.s A in $\mathsf{span}(d_2)$, $(A, a) \in d$ iff $(A, a) \in d_2$.*

We denote this by $J(d_1, d_2)$.

PROPOSITION 4.15. *If d_1 and d_2 are complementary instantiation-sets for $\mathcal{B}$, then the join of d_1 and d_2, $J(d_1, d_2)$, is a complete instantiation-set for $\mathcal{B}$.*

PROPOSITION 4.16. *Given an instantiation-set d for $\mathcal{B}$,*

$$P(d) = \sum_{d' \in X(d)} P(J(d, d')).$$

In circumscribing explanations, r.v.s which do not causally affect the evidence remain uninstantiated so as to prevent the undesirable affect of decreasing the probabilities. In what follows, we present our formulation of circumscribing explanations by introducing the concept of *well-founded* instantiation-sets.

DEFINITION 4.12. *An instantiation-set w is said to be* well-founded *for $\mathcal{B} = (V, P)$ iff for all r.v.s $A \in \mathsf{span}(w)$, $\mathsf{cond}(A) \subseteq \mathsf{span}(w)$.*

Basically, an instantiation-set is well-founded when the instantiation of a r.v. A implies that the r.v.s which causally affect A are also instantiated.

PROPOSITION 4.17. *All complete instantiation-sets for $\mathcal{B}$ are well-founded for $\mathcal{B}$.*

The following theorems will be very important to our computation of probabilities for well-founded instantiation-sets.

PROPOSITION 4.18. *Let w be a well-founded instantiation-set for $\mathcal{B}$ and let w' be complimentary to w. The probability of the join of w and w' is*

$$\begin{aligned} P(J(w, w')) = & \prod_{A \in \mathsf{span}(w)} P(w(A) | w(\mathsf{cond}(A))) \\ & * \prod_{A \in \mathsf{span}(w')} P(w'(A) | J(w, w')(\mathsf{cond}(A))). \end{aligned}$$

NOTATION. Let v be a subset of V. $\kappa(v)$ is the set of all instantiation-sets for $\mathcal{B} = (V, P)$ whose span is v.

THEOREM 4.19. *Given a Bayesian network $\mathcal{B} = (V, P)$, let v be a subset of V and q be an instantiation-set for $\mathcal{B}$ whose span is $V - v$, then*

$$\sum_{w \in \kappa(v)} \prod_{A \in \mathsf{span}(w)} P(w(A) | J(w, q)(\mathsf{cond}(A))) = 1.$$

THEOREM 4.20. *If w is a well-founded instantiation-set for $\mathcal{B}$, then*

$$P(w) = \prod_{A \in \mathsf{span}(w)} P(w(A) | w(\mathsf{cond}(A))).$$

From Theorem 4.20, the probability of a well-founded instantiation-set for $\mathcal{B}$ can be computed in a straightforward manner like complete instantiation-sets for $\mathcal{B}$. We will now proceed to show that 0-1 solutions for weakly induced constraint systems are equivalent to well-founded instantiation-sets.

Given a 0-1 solution s for $\hat{L}(\mathcal{B})$, we can construct a well-founded instantiation-set w_s for $\mathcal{B}$ as follows: $s(\mathsf{A_a}) = 1$ iff $(A, a) \in w_s$. To convert from a well-founded instantiation-set to a 0-1 solution again is slightly trickier. Given a well-founded instantiation-set w for $\mathcal{B}$, construct a 0-1 solution s_w for $\hat{L}(\mathcal{B})$ as follows: (1) For all A in $\mathsf{span}(w)$, $(A, a) \in w$ iff $s_w(\mathsf{A_a}) = 1$, and (2) For all A in $V - \mathsf{span}(w)$, for all $\mathsf{A_a}$ in $\Delta(A)$, $s_w(\mathsf{A_a}) = 0$. For each conditional variable q in $\Upsilon_{\mathsf{A_a}}$, set the appropriate value according Theorem 4.14.

THEOREM 4.21. *If s is a 0-1 solution for $\hat{L}(\mathcal{B})$, then w_s is a well-founded instantiation-set for $\mathcal{B}$.*

THEOREM 4.22. *If w is a well-founded instantiation-set for $\mathcal{B}$, then s_w is a 0-1 solution for $\hat{L}(\mathcal{B})$.*

From Theorems 4.21 and 4.22, there is a 1-1 and onto mapping between 0-1 solution for the weakly induced constraint system and well-founded instantiation-sets for the Bayesian network.

From this point on, let w be a well-founded instantiation-set for $\mathcal{B}$.

THEOREM 4.23.

$$\Theta_L(s_w) = -\log(P(w)).$$

DEFINITION 4.13. *Let e be some evidence for $\mathcal{B} = (V, P)$. We construct $\hat{L}_e(B) = (\Gamma, I_e, \psi)$ from $\hat{L}(\mathcal{B}) = (\Gamma, I, \psi)$ as follows: Let $I_e = I \cup I'$ where the constraint $\mathsf{A_a} = 1$ is in I' iff $(A, a) \in e$. We say that $\hat{L}_e(\mathcal{B})$ is* weakly induced *by $\mathcal{B}$* with evidence *e.*

DEFINITION 4.14. *Given some evidence e for $\mathcal{B}$ and some instantiation-set w, we say that w is a* hypothesis *for e iff w is consistent with e.*

THEOREM 4.24. *If s is a 0-1 solution for $\hat{L}_e(\mathcal{B})$, then w_s is a hypothesis for e.*

THEOREM 4.25. *If w is a hypothesis for e, then s_w is a 0-1 solution for $\hat{L}_e(\mathcal{B})$.*

Similarly to our earlier work involving complete instantiation-sets, when there is some set of evidence given to be explained, we only want to consider those instantiation-sets which are hypotheses for the evidence. Theorems 4.24 and 4.25 above guarantee that the evidence also properly restricts the set of possible 0-1 solutions we wish to consider.

Now, we must show that the costs associated to each 0-1 solution are directly related to the probability of the corresponding instantiation-sets.

For the following theorems, assume that $\hat{L}$ is weakly induced by a Bayesian network $\mathcal{B}$, w is a well-founded instantiation-set for $\mathcal{B}$, and s is a 0-1 solution for $\hat{L}(\mathcal{B})$.

THEOREM 4.26. *There exists a constant α_e such that for all well-founded instantiation-sets w consistent to e, $\Theta_{L_e}(s_w) = \alpha_e - \log(P(w|e))$.*

THEOREM 4.27. *w is a maximal well-founded instantiation-set for $\mathcal{B}$ with evidence e iff s_w is an optimal 0-1 solution for $\hat{L}_e(\mathcal{B})$.*

Theorem 4.26 guarantees that the ordering imposed on the well-founded instantiation-sets according to decreasing probabilities is exactly identical to the ordering imposed on 0-1 solution according to ascending costs. Furthermore, computing the cost for a 0-1 solution immediately determines the probability of its associated well-founded instantiation-set.

In general, "realistically" designed Bayesian networks should work well under the well-foundedness formulation. However, there still exists networks which cause problems for well-foundedness. Our daydream problem is one such network. There is no instantiated evidence to be explained. Thus, by enlarging our set of possible explanations to well-founded instantiation-sets, we find that the best one for our daydreaming problem is to simply not make any instantiations.

While there is no evidence in our daydream problem, there is an idea that the person, Glenn, is concerned about dying and thus wishes to know the value of the node Dead. We will formalize this idea by saying that some nodes are the "focus" of the problem. Here, the focus is Dead. We then want to find the values of the focus nodes by constructing the most likely scenario.

DEFINITION 4.15. *A* focus f *in a given Bayesian network* $\mathcal{B} = (V, P)$ *is a subset of* 2^V. *An instantiation-set* w *for* $\mathcal{B}$ *is said to be* f-focused *iff* $\mathsf{span}(w) \in f$.

Simply put, a focus is the collection of r.v. sets whose instantiations we may be particularly interested in. In our daydreaming problem, if we wished to know the status of our being alive, each set in the focus would contain the r.v. Dead. On the other hand, if we are in the context of speaking with our travel agent, then our focus would have vacation spots for Location.

We now define our focused explanation problem as follows: Given a Bayesian network $\mathcal{B} = (V, P)$, some evidence e and some focus f, find a hypothesis w^* for e which satisfies

$$P(w^*|e) = \max_{w \in W} P(w|e)$$

where W is the set of all hypotheses for e such that $w \in W$ iff w is f-focused.

NOTATION. Let A be a r.v. in $B = (V, P)$. We define the $\mathsf{cond}(\mathsf{cond}(A))$ to be

$$\bigcup_{B \in \mathsf{cond}(A)} \mathsf{cond}(B).$$

Let $\mathsf{cond}^*(A)$ to be the transitive closure of this operation.

With this focusing approach, we can easily see that well-foundedness is a special case. The focus for well-foundedness is simply the single set

$$\bigcup_{A \in \mathsf{span}(e)} \mathsf{cond}^*(A).$$

The best explanation generated will be precisely the hypothesis which maximizes this focus.

A similar notion of focusing in circumscribing explanations has previously been discussed in [65]. However, the main difference between our approach and the approach presented in [65] lies in the treatment of focusing. In [65], focusing is considered a special type of evidence called the *(null) evidence.* This type of evidence simply requires that the r.v.s specified must be instantiated. The approach then uses these (null) evidences and combines it with circumscribing explanation in the hopes of solving problems like the daydream above. Basically, any r.v. which causally affects the special (null) evidence r.v. must be instantiated. However, as [65] points out, certain Bayesian networks do exist which results in counterintuitive solutions using their approach. For example, suppose we are interested in whether we will live or not which involves making the r.v. Dead into a (null) evidence. Also, suppose that our network was modified such that Dead also depended on Location. Our new Bayesian network can be represented by the following conditional probabilities:

$P(\mathsf{Healthy} = \mathsf{true}) = .8$
$P(\mathsf{Location} = \mathsf{at\ home} | \mathsf{Healthy} = \mathsf{true}) = 0$
$P(\mathsf{Location} = \mathsf{at\ home} | \mathsf{Healthy} = \mathsf{false}) = 1$
$P(\mathsf{Location} = \mathsf{Barbados} | \mathsf{Healthy} = \mathsf{true}) = .1$
$P(\mathsf{Location} = \mathsf{Barbados} | \mathsf{Healthy} = \mathsf{false}) = 0$
$P(\mathsf{Location} = \mathsf{Caracas} | \mathsf{Healthy} = \mathsf{true}) = .1$
$P(\mathsf{Location} = \mathsf{Caracas} | \mathsf{Healthy} = \mathsf{false}) = 0$
... plus 8 more vacation spots with .1/0 conditional probabilities ...
$P(\mathsf{Dead} = \mathsf{true} \,|\, \mathsf{Healthy} = \mathsf{true}, \mathsf{Location} = \mathsf{at\ home}) = 0$
$P(\mathsf{Dead} = \mathsf{true} \,|\, \mathsf{Healthy} = \mathsf{true}, \mathsf{Location} = \text{<vacation-spot>}) = 0$
$P(\mathsf{Dead} = \mathsf{true} \,|\, \mathsf{Healthy} = \mathsf{false}, \mathsf{Location} = \mathsf{at\ home}) = 1$
$P(\mathsf{Dead} = \mathsf{true} \,|\, \mathsf{Healthy} = \mathsf{false}, \mathsf{Location} = \text{<vacation-spot>}) = 1$

According to the MPE criterion, the most-probable explanations for this network is

$$\{\mathsf{Healthy} = \mathsf{false}, \mathsf{Dead} = \mathsf{true}, \mathsf{Location} = \mathsf{at\ home}\}$$

again with a probability of .2. Since we must include the r.v. Dead, we must also instantiate the r.v. Location thus returning us to our original problem of having to instantiate everything.

From our above discussion, our approach is different in that it considers focusing as the mechanism for determining which r.v.s to instantiate. It is a general approach which precisely defines the problem of computing incomplete instantiation-sets. As we also saw above, well-foundedness is a special type of focusing under our approach. Similarly, the approaches found in [65, 67, 66] can also be classified within our model.

4.2 Belief Updating

In the previous sections, we have demonstrated how belief revision can be modeled using our linear constraint satisfaction approach. We now show that we can also model belief updating based on our constraints formulation of belief revision.

The goal of belief updating is to determine the belief of an event given some evidence. Basically, given evidence e, we are attempting to find for each r.v. A and instantiation $a \in R(A)$,

$$\mathrm{Bel}(A = a) \equiv P(A = a|e).$$

4.2.1 Formulation

We begin our formulation by making a simple observation found in basic probability theory. Let $\mathcal{B} = (V, P)$ be a Bayesian network and let Q be a subset of r.v.s from V. Furthermore, let w be an instantiation-set for $\mathcal{B}$ such that $\mathsf{span}(w) = Q$.

PROPOSITION 4.28. *For any subset of r.v.s Q' such that* $Q \cap Q' = \phi$,

$$P(w) = \sum_{\mathsf{span}(w')=Q'} P(w \cup w').$$

From Proposition 4.28, we can see that it is possible to compute $\mathrm{Bel}(A = a)$ given evidence e in terms of

$$\mathrm{Bel}(A = a) \equiv \frac{P(A = a, e)}{P(e)}$$

where $P(A = a, e)$ and $P(e)$ can be computed by

$$\sum_{\mathsf{span}(w)=V-\{A\}\cup\mathsf{span}(e)} P(w \cup \{A = a\} \cup e)$$

and

$$\sum_{\mathsf{span}(w)=V-\mathsf{span}(e)} P(w \cup e)$$

respectively.

As we can easily see, the joint probabilities involved in the summations are complete instantiation-sets for $\mathcal{B}$. Thus, the probabilities can be computed in a straightforward manner in Bayesian networks.

Obviously, the difficulty involved in directly applying this approach is the immense number of complete instantiation-sets involved. The best we could hope for is an estimate to the actual value.

Basically, our approach is as follows: We are given evidence e and would like to determine $\text{Bel}(A = a)$. We generate a sequence of complete instantiation-sets $I_1, I_2, \ldots$. As each instantiation-set is generated, we will update our estimate to $\text{Bel}(A = a)$ which we denote by $\hat{\text{Bel}}(A = a)$. We continue this updating using each I_j generated until some criterion (such as a time limit) is met.

We now have a very general outline of what our system should do. In order to fill in the gaps, we must be able guarantee that the estimates will converge to the actual value and that the process of selecting and generating the complete instantiation-sets can be done in an efficient and somewhat "goal-directed" manner.

We compute our estimates as follows: Let $I_1, I_2, \ldots$ be a sequence of complete instantiation-sets. Define a function $q : W \times W \longrightarrow \{0, 1\}$ where W is the set of all instantiation-sets for $\mathcal{B}$ such that $q(w, \hat{w}) = 1$ iff $\hat{w} \subseteq w$. We define the jth estimate of $\text{Bel}(A = a)$ as

$$\hat{\text{Bel}}_j(A = a) = \frac{\hat{P}_j(A = a, e)}{\hat{P}_j(e)} \tag{17}$$

where

$$\hat{P}_j(A = a, e) = \hat{P}_{j-1}(A = a, e) + q(I_j, \{A = a\} \cup e)P(I_j) \text{ and } \hat{P}_0(A = a, e) = 0$$

and

$$\hat{P}_j(e) = \hat{P}_{j-1}(e) + q(I_j, e)P(I_j) \text{ and } \hat{P}_0(e) = 0.$$

The following proposition follows straightforwardly:

PROPOSITION 4.29. *If $I_j \neq I_k$ for all $j \neq k$, then the sequence must be of finite length, that is, $I_1, I_2, \ldots, I_n$.*

It is necessary to make the assumption that the sequence of instantiation-sets is unique. Otherwise, we would be incorrectly re-using information that had already been incorporated into our computations.

THEOREM 4.30. *If $I_j \neq I_k$ for all $j \neq k$, then there exists some integer N such that $\hat{Bel}_N(A = a) = Bel(A = a)$.*

The above theorem guarantees that our computations will converge to the correct value. Now having shown that the computations will work for any sequence of distinct instantiation-sets, we must now consider which sequence may best suit our needs.

The simplest method for generating sequences would be to arbitrarily make instantiations for each r.v. Basically, each possible instantiation of a r.v. would have an equal chance of being realized in any instantiation-set created. The main difficulty with this simple approach is that an inordinate amount of time may be wasted in generating instantiation-sets with "insignificant" probabilities. For example, there may only be a small number of complete instantiation-sets whose probabilities summed together constitute the "mass" of the probability distribution. For example, of a million possible complete instantiation-sets, only a thousand of them together constitute .75 of the distribution.

An improved approach would be to first start at root nodes, that is, nodes with no incoming arcs, and make instantiations which are weighted according to the prior probabilities. For example, let A be a r.v. with prior probabilities $P(A = \mathsf{true}) = 0.8$ and $P(A = \mathsf{false}) = 0.2$. Thus, there is an 80% chance of assign A to true and only 20% to false. After the root node instantiations have been made, we proceed in a top-down manner and consider instantiations for those nodes which are dependent only on root nodes. The possible instantiations are then weighted against the appropriate conditional probability entries. For example, let B be a r.v. dependent on A with conditional probabilities $P(B = \mathsf{true} | A = \mathsf{true}) = 0.6$ and $P(B = \mathsf{true} | A = \mathsf{false}) = 0.3$. Assuming that A has been instantiated to true, we would consider the instantiation to be with weights of 0.6 for true and 0.4 for false. As we can clearly see, a r.v.s instantiation cannot be determined until all the r.v.s it depends on have already been instantiated, thus forcing a top-down construction paralleling the causality information built into the topology of the network. This sort of approach was introduced in [24] called *logic sampling*.

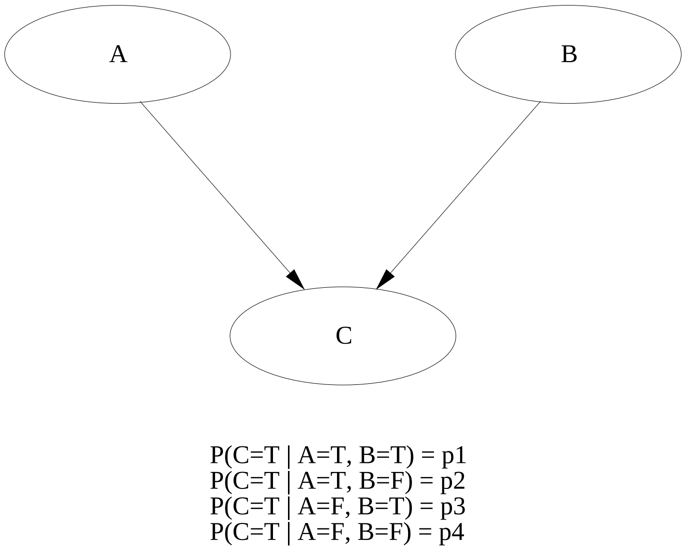


FIG. 4.3. Simple Bayesian network.

Still, eventhough logic sampling takes into account a fair amount of information inside the Bayesian network, its approach can also miss those "significant" instantiation-sets. For example, consider our simple Bayesian network in Figure 4.3 with actual probabilities:

$$
\begin{aligned}
&P(A = \mathsf{true}) = 0.1 \\
&P(B = \mathsf{true}) = 0.8 \\
&P(C = \mathsf{true}\,|A = \mathsf{true}, B = \mathsf{true}) = 0 \\
&P(C = \mathsf{true}\,|A = \mathsf{false}, B = \mathsf{true}) = 0 \\
&P(C = \mathsf{true}\,|A = \mathsf{true}, B = \mathsf{false}) = 1 \\
&P(C = \mathsf{true}\,|A = \mathsf{false}, B = \mathsf{false}) = 0
\end{aligned}
$$

Assume our evidence is that $C = \mathsf{true}$. If we choose according to the most likely instantiations, we initially get false for A and true for B. Since C must be true, our first complete instantiation-set would be $\{A = \mathsf{false}, B = \mathsf{true}, C = \mathsf{true}\}$. But it has a probability of 0! The best and only significant complete instantiation-set is in fact $\{A = \mathsf{true}, B = \mathsf{false}, C = \mathsf{true}\}$ with probability 0.02 and it is the least likely instantiation-set of the four to be chosen.

The above discussion strongly suggests that avoiding processing very low probability complete instantiation-sets is very desirable. Thus, we would wish to somehow only generate those instantiation-sets which are deemed significant. We could say that a complete instantiation-set is "most-significant" when it has the highest probability of those consistent with the evidence. Furthermore, if we could start with the most-significant and continue down the list in order of "next most-significant", then we could begin to accumulate the "mass" of the probabilities in the fastest manner. Indeed, we can prove this statement as follows.

DEFINITION 4.16. *Given a Bayesian network $\mathcal{B} = (V, P)$ and evidence e, a sequence of complete instantiation-sets $I_1, I_2, \ldots, I_n$ for $\mathcal{B}$ is said to be an* I-sequence of e *if and only if $e \subseteq I_j$ for $j = 1, \ldots, n$ and $I_j \neq I_k$ for all $j \neq k$. Furthermore, an I-sequence is said to be* complete *if any complete instantiation-set consistent with e can be found in the I-sequence.*

DEFINITION 4.17. *Given a Bayesian network $\mathcal{B} = (V, P)$, evidence e and a complete I-sequence of e, $I = I_1, \ldots, I_n$, we define the* mass *of e with respect to $I_1, \ldots, I_n$,*

$$M(e, I) = \sum_{j=1}^{n} P(I_j).$$

PROPOSITION 4.31. *Given a Bayesian network $\mathcal{B} = (V, P)$, evidence e and any two complete I-sequences of e, I and I', $M(e, I) = M(e, I')$.*

From Proposition 4.31, the mass of evidence e is independent of whichever complete I-sequence of e we choose. We will simply denote the mass of e by $M(e)$.

DEFINITION 4.18. *Given a Bayesian network $\mathcal{B} = (V, P)$, evidence e and two complete I-sequences of e, I and I', I is said to* mass-faster *than I' if and only if for all $j = 1, \ldots, n$ where n is the number of complete instantiation-sets in I or I',*

$$\sum_{k=1}^{j} P(I_k) \geq \sum_{k=1}^{j} P(I'_k).$$

Furthermore, we say that I is mass-fastest *if it is mass-faster than any other I-sequence for e.*

This now leads to our following theorem:

THEOREM 4.32. *Given a Bayesian network $\mathcal{B} = (V, P)$ and evidence e, an I-sequence of e which instantiation-sets are sorted in order of decreasing probabilities is mass-fastest.*

With the above theorem showing us which I-sequence we wish to generate, we must now consider how to go about doing it. So, we must somehow generate complete instantiation-sets which are consistent with e and in order of decreasing probability. As it turns out, the complete instantiation-set with the highest probability and is consistent to e is exactly the best explanation for e. And as we well know from above, we can solve for the best explanation using our linear constraint satisfaction approach. Furthermore, the remaining complete instantiation-sets that needs to be generated correspond to the alternative explanations which we can also generate. Therefore, we will perform belief updating by actually performing belief revision.

Combining (17) and our linear constraint satisfaction system for belief revision will provide us with estimates on the actual belief values.

4.2.2 Selective Updating

So far, we have assumed that we are interested in updating the beliefs of all the r.v.s. We now proceed to present a more specialized algorithm to handle the case where we are only interested in the updated beliefs of a single variable. Consider the following scenario: Suppose for a r.v. A, there exists an $a \in R(A)$ such that $P(A = a|e) = \alpha$ and $\alpha \gg P(A = b|e)$ for all $b \in R(A)$ and $b \neq a$. We also assume that $P(e) > 0$. So, for the purpose of our discussion, assume that $\alpha = .95$. Since, $P(A = a|e)$ is such a dominating value, we know $P(\{A = a\}, e)$ to be a significant portion of $P(e)$. Furthermore, $P(\{A = a\}, e)$ will generally be formed from a collection of explanations for e which constitutes almost all the mass for e and thus usually be one of the best explanations for e around. In terms of the mass-fastest I-sequence for e, the top I-sequence explanations will be those also consistent with $\{A = a\}$. Obviously, the estimates for $P(A = b|e)$ when $a \neq b$ will basically stay near zero. In general, we are interested in rough estimates of what our updated beliefs are. We tend to be just interested in which is the most likely instantiation and how dominating the estimate seems to be compared to the remaining instantiations. However, to allow for cases where better numerical estimations are desired, our approach can be naturally modified as follows:

Let A be the r.v. we are interested in finding the belief of given evidence e and $R(A) = \{a_1, \ldots, a_n\}$. Intuitively, if we find the best explanations for $e \cup \{A = a_i\}$ for each $i = 1, \ldots, n$, this should give us a good approximation to for $\mathrm{Bel}(A)$. Basically, we compute

$$\hat{\mathrm{Bel}}_0(A = a_i) = \frac{T_0(\{A = a_i\}, e)}{\sum_{j=1}^{n} T_0(\{A = a_j\}, e)} \tag{18}$$

where

$$T_0(\{A = a_i\}, e) = \max_{w \in W(\{A=a_i\} \cup e) \subseteq w} P(w)$$

and W is the set of all complete instantiation-sets. Obviously, the sum in the denominator is an approximation of $P(e)$. We can easily compute $T_0(\{A = a_i\}, e)$ by treating $\{A = a_i\} \cup e$ as new evidence e' and running our belief revision technique to find the best explanation for e'.[20]

After finding all the best explanations, we can further refine our initial estimates by considering the alternative explanations in a somewhat directed manner as follows: Of the best explanations, choose i such that $T_0(\{A = a_i, e)\}$ is maximal. Now, find the next best explanation for $\{A = a_i\} \cup e$. Let $T_1(\{A = a_i\}, e)$ be the probability of the next best explanation. We can take this new explanation and incorporate it into our estimation. The basic process continues by determining among which of the unselected explanations including this new one has the highest probability. We then proceed to find the next best explanation for it. Thus, we are again generating a complete I-sequence for e which is also made up of distinct elements. According to Theorem 4.30, our I-sequence will guarantee that our estimations will converge to the proper probabilities.

[20]When we have a probability of 0, which can certainly occur, in one of the $T_0(\{A = a_i\} \cup e)$, we know that the negative log is infinity. So, to deal with this problem, we simply change the 0 to some arbitrarily small yet positive value.

4.2.3 Quick Sampling

In large knowledge-bases, we often run into the phenomena where given an explanation, changing the state of a r.v. hardly affects the final joint probability, that is, the probability of the new explanation barely differs from that of the original. Return to our story about Mary: Mary walks outside and finds that the street and lawn are wet. She concludes that it has just rained recently. Furthermore, she decides that she does not need to water her climbing roses. We have the Bayesian network in Figure 4.1. Now, assume that we also included information about the color of Mary's blouse in a new r.v. Marys-blouse-color. Obviously, this information should have very small impact on the rest of our network and consequently is not attached to any other r.v. in the network. Furthermore, the color of her blouse is almost equally likely to be any color.

Since any explanation must be a complete instantiation-set, the addition of the new r.v. has increased the number of possible explanations by a factor of the number of r.v. states. Also, since the blouse color can be anything, there will be a multiple number of explanations with maximal probability.

Obviously, this can be solved by using the focusing methods found in Section 4.1.3. However, consider the domain of fault detection in circuit diagnosis where the r.v.s tend to be very deterministic in nature, that is, made up of logic gates such as OR-nodes. The phenomena arises when given an OR-node, we have multiple inputs (or, r.v.s) with which to activate a gate. These inputs each could have the same probability of being realized.

Observe that this phenomena often just involves some state changes. Also, calculating the new probability of such state changes can be done in a very local manner. We would simply divide out the original conditional probabilities and multiply in the new ones. So, given an explanation, we can go and generate a new set of explanations by taking the given one and "flipping" some of its states in different combinations. Of course, there is a limit on how many you generate since we could have conceivably gone through and tried all different combinations from the original which although may give us the exact solution to the updating, however, this becomes an exercise in combinatorics.

Basically, our goal is to use the current best explanation as a template and flip r.v.s in order to catch our somewhat "redundant" ones. The process is certainly inexpensive up to a certain point.

4.2.4 Hill-Climbing

By using the Simplex method in our search for the best-explanation, we are engaging in a hill-climbing process. We proceed towards our answer by continually looking for better solutions. With this in mind, it seems that our search for the best explanation is proceeding through explanations which are increasingly better. Up to this point, we have been interested only in the best explanation. Since these intermediate solutions might provide other "good" explanations, we should consider using this free information.

Although Simplex is performing a hill-climbing search, the solution space it uses is continuous and hence non-integral. (Recall our discussion in Section 3.2 concerning non-integral solutions.) Thus, the intermediate solutions our Simplex method generates may not be integral. This implies that we cannot directly construct an explanation.

Given a r.v. A with $R(A) = \{a_1, \ldots, a_n\}$, unless there exists an i such that $\mathsf{A}_{\mathsf{a}_i} = 1$ and $\mathsf{A}_{\mathsf{a}_j} = 0$ for all $j \neq i$, we cannot determine an instantiation for A according to our strict definitions. Yet, consider the following: Each assignment of values to the real variable is governed by the constraints and the objective function in Definition 4.7. The objective function itself is an embodiment of the joint probability distribution of the given Bayesian network. Thus, these partial solutions provide strong indicators as to what instantiations should be made.

Recall earlier our discussion of the logic sampling approach and its difficulty with evidence in the non-root r.v.s. Some global scheme is desirable to direct the instantiations. It seems that our partial solutions should provide us with such a mechanism.

Assume we have a non-integral solution Q for our Bayesian induced constraint system. Let $\mathsf{A}_{\mathsf{a}_1}, \ldots, \mathsf{A}_{\mathsf{a}_n}$ be the real variables associated with r.v. A such that there exists some i where $\mathsf{A}_{\mathsf{a}_i}$ is non-integral. In our transformation to constraint systems, we know that $\mathsf{A}_{\mathsf{a}_1} \geq 0$ and

$$\sum_{j=1}^{n} \mathsf{A}_{\mathsf{a}_j} = 1. \tag{19}$$

Thus, $\mathsf{A}_{\mathsf{a}_1}, \ldots, \mathsf{A}_{\mathsf{a}_n}$ directly provides us with a weighting scheme on how to choose the proper instantiations in order to create an explanation. From here, there are a variety of methods to construct the explanations. We could either simply choose the instantiation with the largest associated weight or use the weights as likelihoods and perform some limited stochastic simulations.

4.3 Other Models

In the above subsections, we demonstrated that Bayesian networks can be modeled through linear constraint satisfaction. Bayesian networks are by far one of the most commonly used models for probabilistic reasoning. This is mainly due to the model's success at formulating various tasks and actually solving the more simpler ones in a reasonable amount of time. Almost all alternative uncertainty models have failed to provide working implementations.

Models which are closely related to Bayesian networks can also be formulated using linear constraints as well. For example, Markov random fields [27, 33].

4.4 Discussion

We have just demonstrated that our linear constraint satisfaction approach can completely model Bayesian networks. In particular, the reasoning task of belief revision and belief updating can be solved using our tools and techniques. Furthermore, we have demonstrated that our approach can avoid the limitations currently suffered by existing methods. Generating alternative explanations are available through our formulation. This should provide us with some idea as to the representational capabilities of our linear constraint satisfaction approach.

Although our experiments in cost-based abduction were rather successful, our approach to belief revision may easily suffer the exponential explosions in the size of the conditional probability tables. For example, consider image processing where each pixel is represented by a r.v. [19].

Yet, of the other existing algorithms for performing belief revision, namely Pearl's message-passing scheme [41] and Shimony and Charniak's cost-based approach [68], message-passing is incapable of generating alternative explanations while the cost-based method suffers from the heuristic problems outlined in Section 3. Given this dearth of algorithms, we believe ours to be a plausible alternative. Through various tests, we have noticed that our method performs best on highly deterministic networks, that is, networks whose conditional probabilities are near 0 or 1.

For belief updating, we compared our approach against stochastic simulation methods [72, 34, 28, 10, 24, 62]. We found that our approach performed much better than stochastic simulation on highly deterministic networks. Basically, as we discussed earlier, simulation approaches could very rarely come up with an answer on these types of networks. On the other hand, our approach (without our suggested optimizations) is much slower on networks which have very flat distributions. We incur a higher overhead than the simulation techniques. Still, our formulation can easily solve a class of networks which have proven to be stumbling blocks for the simulation methods.

In the remainder of this section, we present some preliminary results on an alternative constraints formulation for Bayesian networks which seems quite interesting. This new formulation attempts to address some of the limitations found in the current one. New tools and techniques are also provided for solving this new system.

4.5 Near-Continuous Random Variables

The allure of Bayesian networks stems from its robust yet simple modeling of the world. As we have seen, it is capable of modeling abduction as well as belief updating. Furthermore, its graphical representation provides a firm foundation for modeling causality and causal inferencing.

Unfortunately, inherent in Bayesian networks is the potential for exponential explosions in the size of its conditional probability tables. Obviously, the main factor is the number of parents a node has plus the number of instantiations each parent has. Even having a small number of parents but a large number of instantiation sets for each can become quite prohibitive.

This implicit restriction has been the crux of Bayesian network's crudity at modeling quantitative information. Common information such as time and distance are often poorly modeled with discrete coarse approximations to avoid the combinatorial explosion.

We now present a method which may provide a finer grained discretization of quantized data without the exponential explosion. Although we are still dealing with discrete r.v.s, the level of discretization we wish to consider prompts us to call these *near-continuous* r.v.s.

4.5.1 Formulation

Basically, our current formulation for belief revision is firmly tied to the sizes of the conditional probability tables. Each entry in the table will be associated with some unique real variable in the constraint system. To avoid the possible exponential explosion, our approach is to compress the information found in the tables. Through the use of functions we call *splining functions*, we could reduce the entire table and store it as a simple function. Ideally, we would like to have a table reduced to a single splining function, but multiple functions are still more desirable than having the explicit table as we shall see. These splining functions map sets of real numbers to a single real number. The set of real numbers will be used as indices to fetch the appropriate conditional probabilities. But first, we must map the different possible instantiations for a r.v. to real values.

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

Let $\mathcal{B} = (V, P)$ be a Bayesian network.

DEFINITION 4.19. *Given a r.v. A in V, a one-to-one and onto mapping E_A from $R(A)$ to $\{q_1, q_2, \ldots, q_n\}$ where $q_i \in \mathcal{Q}^+$ is called an* encoder *for A.*

Let $\overline{E_{\mathcal{B}}}$ be a collection of encoders for the r.v.s in V such that there is exactly one encoder for each r.v. Intuitively, encoders provide a total ordering on the possible instantiations for a r.v. Furthermore, it provides a mechanism for flexibly transforming/reorganizing our instantiations to a possibly more useful form or interpretation as we shall see.

NOTATION.

$$\nabla(A) = \{P(A = a|C_1 = c_1, \ldots, C_n = c_n) \in P | a \in R(A)\}$$

If we have multiple splining functions, then we must have a way of determining which one to use given an instantiation. By restricting which entries in a table may be grouped with other entries, we can provide such a mechanism.

NOTATION. Let $\rho(A)$ be a partition on $\nabla(A)$ and $\wr$ to be a cell in $\rho(A)$. We define $\mathsf{extent}(A, \wr)$ to be be all the different instantiations of A which appear in $\wr$.

DEFINITION 4.20. *Given a r.v. A in V and $\nabla(A)$, assume* $\mathsf{cond}(A) = \{C_1, \ldots, C_n\}$*. Let $\rho(A)$ be a partition on $\nabla(A)$. We say the $\rho(A)$ is* contiguous *with respect to $\overline{E_B}$ if and only if for each cell $\wr$ in $\rho(A)$, the following condition holds:*

- *For all r.v. B in $\{A\} \cup \mathsf{cond}(A)$, there does not exist a $b \in R(B)$ such that b is not in $\wr$ and $E_B(b)$ is in the interval from $\min_{i_j} E_B(b_{i_j})$ to $\max_{i_j} E_B(b_{i_j})$.*
- *Let $\nabla(A)$ consists of conditional probabilities of the form*

$$P(A = a|C_1 = c_1, \ldots, C_n = c_n).$$

For all collections $\{a', c'_1, c'_2, \ldots, c'_n\}$ such that $a' \in \mathsf{extent}(A)$ and $c'_i \in \mathsf{extent}(C_i)$ for $i = 1, \ldots, n$,

$$P(A = a'|C_1 = c'_1, \ldots, C_n = c'_n) \in \wr.$$

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

DEFINITION 4.21. *Let $\rho(A)$ be a contiguous partition with respect to $\overline{E_B}$. For each cell $\wr$ in $\rho(A)$, we associate a function $S_{\wr,\overline{E_B}}$ from $\Re^{n+1}$ to $\Re$ such that*

$$S_{\wr,\overline{E_B}}(E_A(a), E_{C_1}(c_1), \ldots, E_{C_n}(c_n)) = P(A = a|C_1 = c_1, \ldots, C_n = c_n) \qquad (20)$$

where $E_A, E_{C_1}, \ldots, E_{C_n}$ *are the encoders found in*

$$\overline{E_{\mathcal{B}}} \text{ and } a \in R(A), c_1 \in R(C_1), \ldots_n \in R(C_n).$$

We call $S_{\wr,\overline{E_{\mathcal{B}}}}$ *a* spline *for* $\wr$. *Let* $\mathcal{S}_{\rho(A),\overline{E_{\mathcal{B}}}}$ *be a collection of splines associated with partition* $\rho(A)$. *We call* $\mathcal{S}_{\rho(A),\overline{E_{\mathcal{B}}}}$ *a* spline-set *for* A.

Let $S_{\overline{E_{\mathcal{B}}}}$ be a set of spline-sets such that each r.v. A in $\mathcal{B}$ is associated with exactly one spline-set.

NOTATION. Given $\mathcal{S}_{\rho(A),\overline{E_{\mathcal{B}}}}$, since $\rho(A)$ is a partition, for any $a \in R(A), c_1 \in R(C_1), \ldots c_n \in R(C_n)$,

$$S_{\wr,\overline{E_{\mathcal{B}}}}(E_A(a), E_{C_1}(c_1), \ldots, E_{C_n}(c_n))$$

will unambiguously refer to the appropriate spline function defined in $\mathcal{S}_{\rho(A),\overline{E_{\mathcal{B}}}}$.

Similarly, given $\mathcal{S}_{\overline{E_{\mathcal{B}}}}$, since all our conditional probability tables are unique, for any $a \in R(A), c_1 \in R(C_1), \ldots c_n \in R(C_n)$,

$$S_{\overline{E_{\mathcal{B}}}}(E_A(a), E_{C_1}(c_1), \ldots, E_{C_n}(c_n))$$

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

We now redefine our notion of a Bayesian network.

DEFINITION 4.22. *Given a Bayesian network* $\mathcal{B} = (V, P)$, *let* $\overline{E_{\mathcal{B}}}$ *be a collection of encoders and* $\mathcal{S}_{\overline{E_{\mathcal{B}}}}$ *be a collection of spline-sets. We define a* splined Bayesian network *to be a 3-tuple* $\overline{\mathcal{B}} = (V, \overline{E_{\mathcal{B}}}, \mathcal{S}_{\overline{E_{\mathcal{B}}}})$.

We now define the analogue of instantiation-sets for the splined Bayesian networks.

DEFINITION 4.23. *Given a well-founded instantiation set* w *for* $\mathcal{B}$, *we define the* spline probability, P_s *for* $\overline{\mathcal{B}}$ *as*

$$P_s(w) = \prod_{A \in \mathsf{span}(w)} S_{\overline{E_{\mathcal{B}}}}(w(A), w(\mathsf{cond}(A))) \tag{21}$$

where $w(\mathsf{cond}(A))$ *expands appropriately to* $\{w(C_1), \ldots, w(C_n)\}$.

THEOREM 4.33. *Given a well founded instantiation set* w, $P(w) \equiv P_s(w)$.

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

THEOREM 4.34. *Any Bayesian network can be modeled as a splined Bayesian network.*

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

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

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

DEFINITION 4.24. *Given a r.v. A and a splined Bayesian network $\overline{\mathcal{B}}$, we define the partition on $R(A)$ induced by $\overline{\mathcal{B}}$ as follows: $a_1, a_2 \in R(A)$ both belong in the same partition if and only if for all cells, d in $[P]_{\overline{\mathcal{B}}}$, $\{A = a_1\}$ appears in d if and only if $\{A = a_2\}$ appears in d. We call this partitioning the* capsulation *of A by $\overline{\mathcal{B}}$ and denote it by $[A]_{\overline{\mathcal{B}}}$.*

THEOREM 4.35. *Given any cell $d = \{a_1, \ldots, a_k\}$ in $[A]_{\overline{\mathcal{B}}}$ where $E_A(a_i) < E_A(a_{i+1})$, there does not exist $b \in R(A)$ such that $E_A(a_1) < E_A(b) < E_A(a_k)$ and $b \neq a_i$ for $i = 1, \ldots, k$.*

DEFINITION 4.25. *Given a splining function $S \in \mathcal{S}_{\overline{E_{\mathcal{B}}}}$, we say that S is* e-continuous *if and only if*

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

for some constants $k, k_0, k_1, \ldots, k_n$. We say that $\mathcal{S}_{\overline{E_{\mathcal{B}}}}$ is e-continuous *if and only if all splining functions in it are also e-continuous.*

Without going into detail, we must generalize our notion of constraint systems.

Previously in cost-based abduction and belief revision, we restricted our variables to values of 0 and 1. We generalize this by allowing variables to to be restricted to sets of real values. In doing so, we also generalize our notion of a 0-1 solution to the notion of a *permissible* solution as a solution which satisfies all the value restrictions as well as constraints.

We begin our construction as follows: Let $\overline{\mathcal{B}} = (V, \overline{E_{\mathcal{B}}}, \mathcal{S}_{\overline{E_{\mathcal{B}}}})$ be a splined Bayesian network and $\mathcal{S}_{\overline{E_{\mathcal{B}}}}$ be e-continuous. For each r.v. A in V, construct a real variable x_A whose values are restricted to $\{E_A(a) | E_A \in \overline{E_{\mathcal{B}}} \text{ and } a \in R(A)\}$ and a 0-1 restricted variable y_A. Next, arbitrarily label each cell in $[A]_{\overline{\mathcal{B}}}$ by $\{d_{A,1}, d_{A,2}, \ldots, d_{A,n}\}$. For each cell in $\{d_{A,1}, d_{A,2}, \ldots, d_{A,n}\}$, construct a new 0-1 restricted variable $x_{d_{a,i}}$. Construct the following constraints:

$$y_A + \sum_{i=1}^{n} x_{d_{A,i}} = 1 \tag{22}$$

For each $d_{A,i}$,

$$x_A - \min_{a \in d_{A,i}} E_A(a) \geq -M(1 - x_{d_{A,i}}) \tag{23}$$

$$x_A - \max_{a \in d_{A,i}} E_A(a) \leq M(1 - x_{d_{A,i}}) \tag{24}$$

$$x_A \leq M(1 - y_A) \tag{25}$$

where M is some arbitrarily large positive constant. Intuitively, $x_{d_{A,i}}$ is used to "detect" whether x_A falls within a particular interval defined by the partitioning, y_A indicates whether A is instantiated or not, and x_A represents the instantiated value if any.

We now must guarantee that when a r.v. A is instantiated, then all the r.v.s in $\mathsf{cond}(A)$ must also be instantiated. We can guarantee this by simply adding the following constraint for each r.v. A:

$$y_A \leq \frac{1}{|\mathsf{cond}(A)|} \sum_{B \in \mathsf{cond}(A)} y_B \tag{26}$$

From our definitions above, we can unambiguously associate a splining function S in $\mathcal{S}_{\overline{E_{\mathcal{B}}}}$ with some r.v.s $\{A\} \cup \mathsf{cond}(A)$. Furthermore, using our encoder mappings, we describe the domain of S as a cross product of intervals on real variables associated with the r.v.s. For example, let $\rho(A)$ be the partition on $\nabla(A)$

associated with S. According to Definition 4.21, for each r.v. B in $\{A\} \cup \mathsf{cond}(A)$, there exists a set of instantiations of B, $\{B = b_{i_1}, \ldots, B = b_{i_k}\}$ in the cell associated with S in $\rho(A)$ such that there does not exist an instantiation $\{B = c\}$ where $c \neq b_{i_j}$ for all i_j and $E_B(c)$ is in the interval from $\min_{i_j} E_B(b_{i_j})$ to $\max_{i_j} E_B(b_{i_j})$. Thus, we only need the min and the max values to describe the' domain for S. Also, remember that we want to incorporate S into our probabilistic computations only when the r.v.s are instantiated within its restricted domain. Let us denote the interval for a r.v. A for splining function S by $[\min(A, S), \max(A, S)]$. We now proceed with our construction. For each splining function S involving the r.v.s $\{C_1, \ldots, C_n\}$, we construct the new variables $x_{C_i,S}$ which are virtual copies of x_{C_i} constructed above. For each $[\min(C_i, S), \max(C_i, S)]$, if there does not yet exist a 0-1 variable which detects whether x_{C_i} is in $[\min(C_i, S), \max(C_i, S)]$, create a new 0-1 variable $d_{C_i,S}$ with the following constraints: For each $d_{A,i}$,

$$x_{C_i} - \min(C_i, S) \geq -M(1 - d_{C_i,S}) \tag{27}$$

$$x_{C_i} - \max(C_i, S) \leq M(1 - d_{C_i,S}) \tag{28}$$

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

$$d_S \geq n - \sum_{j=1}^{n} d_{C_j,S} + 1 \tag{29}$$

$$d_S \leq \frac{1}{n} \sum_{j=1}^{n} d_{C_j,S} \tag{30}$$

For each i,

$$x_{C_i,S} \geq x_{C_i} - M(1 - d_S) \tag{31}$$

$$x_{C_i,S} \leq x_{C_i} + M(1 - d_S) \tag{32}$$

$$x_{C_i,S} \leq M d_S \tag{33}$$

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

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

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

The terms to be added are

$$-kd_S - \sum_{i=1}^{n} k_i x_{C_i,S} \tag{34}$$

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

Conversely, given a well-founded instantiation-set w, we can construct a permissible assignment $s[w]$ as follows: For each r.v. A in V, if A is instantiated in w, then $s[w](y_A) = 0$ and $s[w](x_A) = E_A(w(A))$. If A is not instantiated in w, then $s[w](d_A) = 1$ and $s[w](x_A) = 0$. Furthermore, if A is instantiated, we properly activate the appropriate interval detectors. Finally, according to the instantiation-set w, we can easily determine which splining functions are active. $s[w](d_S) = 1$ if and only if S is an active splining function according to w. And, if S is active, then copy $s[w](x_{C_i,S}) = s[w](x_{C_i})$ for all i involved with S. Otherwise, $s[w](x_{C_i,S}) = 0$.

THEOREM 4.36. *w is a well-founded instantiation-set for $\mathcal{B}$ if and only $s[w]$ is a permissible solution for the induced constraint system.*

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

THEOREM 4.37.

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

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

One final note to our formulation is that we must also incorporate the notion of evidence. Evidence, we recall, can either be the requirement that a r.v. be instantiated with a certain value or that the r.v. simply be instantiated. For the first case where a r.v. A must be instantiated to a, we simply include the constraint $x_A = E_A(a)$. When it just needs to be instantiated to some value, we include the constraint $x_A >= \min_{a \in R(A)} E_A(a)$. Thus, we can proceed with our belief revision computations like we did earlier in this section.

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

4.5.2 Branch and Bound For Permissible Solutions

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

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

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

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

Similar to our original branch and bound algorithm, the basic idea is as follows: To find an optimal permissible solution, we solve a sequence of linear programs. This sequence can be represented by a tree where each node in the tree is identified with a linear program that is derived from the linear programs on the path leading to the root of the tree. The root of the tree is identified with the linear program induced by our constraint system. The linear programs along the nodes of the tree are generated using the following schema: Consider s_0, the optimal solution to our initial linear program denoted lp_0. If s_0 is a permissible solution, then we are finished. Otherwise, we choose some non-permissible variable assignment x in s_0 and define two new problems lp_1 and lp_2 as descendants of lp_0. lp_1 is identical to lp_0 except for the additional constraint $x \geq \lceil s^0(x) \rceil_x$, and lp_2 is identical to lp_0 except for the additional constraint $x \leq \lfloor s^0(x) \rfloor_x$. Note that the two new problems do not have s_0 as their optimal solutions. Since we are looking for a permissible assignment, the optimal permissible solution must satisfy one of the additional constraints.

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

ALGORITHM 4.2. *Given a constraint system $L = (\Gamma, I, \psi)$, find its optimal permissible solution.*

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

5 Cyclicity and Generalized Cost-Based Abduction

Cost-based abduction is restricted to those knowledge bases which are *acyclic* in nature. It requires that there cannot be two propositions A and B in the knowledge base where the following conditions occur:

- A can be used in a proof for B.
- B can be used in a proof for A.

In the most degenerate case where $A \Longrightarrow B$ and $B \Longrightarrow A$ are both in the knowledge base, if we had as evidence that B is true, then A can be assigned true to prove B. Furthermore, since B is already true, we can use it now to prove A. Thus, no other propositions need to be assigned true to explain B! Clearly, this explanation is counter-intuitive and provides little information. Also, since none of the hypotheses are used, no cost is incurred which can make this explanation, the best explanation.

A more sophisticated example involving cyclicity often occurs in the rule bases of the WIMP story understanding system [3, 23, 22]. In its knowledge base, you can find the logical rules:

$$(\mathsf{foo}\,a) \wedge (=\ a\ b\ ) \Longrightarrow (\mathsf{foo}\,b)$$
$$(\mathsf{foo}\,b) \wedge (=\ a\ b\ ) \Longrightarrow (\mathsf{foo}\,a)$$

A method is available in WIMP to eliminate this logical cyclicity. However, it is rather ad hoc.

A similar situation arises in Hobbs et al. [26] where we find the rules:

$$(\mathsf{dog}\ x) \Longrightarrow (\mathsf{mammal}\ x)$$
$$(\mathsf{mammal}\ x) \wedge (\mathsf{dog\text{-}features}\ x) \Longrightarrow (\mathsf{dog}\ x)$$

The second rule is needed by [26] (and probably by most cost-based schemes) to allow us to use the fact that "something is a mammal" as (weak) evidence that it is a dog. Our $(\mathsf{dog\text{-}features}\ x)$ corresponds to the *etcetera* attribute, $(\mathsf{etc}\ x)$, found in [26].

Finally, cyclicity can also occur in modeling causal information. Suppose we are modeling faulty electrical outlets. Furthermore, suppose that our television set and radio are both plugged into such an outlet. Being faulty, when the fuse is blown in one of the components, the accompanying surge causes the other fuse to also blow. In this case, it is possible that a better axiomatization could solve the problem. However, there is no such solution for the logical case above. Thus, cyclicity must somehow be faced.

Since abduction is a backward chaining process on the logical rules, the search for the best explanation in cost-based abduction can be performed as a graph searching problem. Starting from the evidence, we proceed backwards to the hypotheses through the implications. In this way, we build many partial proofs to use as guides for determining the least cost proof. Introducing cyclicity complicates the problem because explicitly chaining backwards though the implications can end up in an infinite loop.

In this section, we present an approach to the problem of cyclicity in cost-based abduction. We arrived at our solution by studying cyclicity under linear constraint satisfaction. The solution itself represents a natural extension of our constrained optimization approach and remains a linear constraint satisfaction formulation.

5.1 Generalized Cost-Based Abduction

We now address the issue of cyclicity and present a generalization to cost-based abduction. Consider the following set of rules:

$$\begin{aligned}
&(\mathsf{foo}\ a) \wedge (=\ \ a\ b\ ) \Longrightarrow (\mathsf{foo}\ b)\\
&(\mathsf{foo}\ b) \wedge (=\ \ a\ b\ ) \Longrightarrow (\mathsf{foo}\ a)\\
&\mathsf{a\text{-}stuff} \Longrightarrow (\mathsf{foo}\ a)\\
&\mathsf{b\text{-}stuff} \Longrightarrow (\mathsf{foo}\ b)
\end{aligned}$$

We can easily see that this set of rules is cyclic.

Again, we observe the following: If $(=\ \ a\ b\ )$ is true, then $(\mathsf{foo}\ a)$ could be explained by $(\mathsf{foo}\ b)$ and vice versa. Invariably, this is an explanation under cost-based abduction and will most likely be the best one.

Intuitively, to avoid this type of "self-supporting" anomaly, a proper explanation must guarantee that some "outside" agent be present, such as either a-stuff to explain (foo a) or b-stuff to explain (foo b), when (foo b), (foo a) and ($=$ a b) are all true. For this fairly simple case, we can easily enumerate the desired behavior as follows:

- When (foo a), (foo b) and ($=$ a b) are all true, then either b-stuff or a-stuff must be true.
- When (foo a) $=$ true and ($=$ a b) $= false$, then a-stuff must be true.
- When (foo b) $=$ true and ($=$ a b) $= false$, then b-stuff must be true.
- When (foo a) $=$ (foo b) $=$ false and ($=$ a b) is either true or false, then nothing special needs to be done.
- The remaining states are inconsistent and must be prevented from occurring.

From the above behavior list, we can make the following observation: Proper logical reasoning requires that propositions never support themselves. Acyclicity simply gives us a single unique partial ordering on the propositions. On the other hand, cyclicity can actually be viewed as providing multiple partial orderings. In our above example, consider the first behavioral item and pick a-stuff to be false. What we have effectively done is choose the following proof sequence:

1. b-stuff and ($=$ a b) are both true.
2. (foo b) is implied by b-stuff.
3. Since ($=$ a b) is also true, this implies that (foo a) is true.

In terms of causality, we can see this as: Causal reasoning is an inherently time-dependent process since obviously causes must temporally precede effects. Thus, when attempting to explain the occurrence of some given event, we are temporally ordering all the other events which lead up to it.

When we consider the problem in terms of our cost-based abduction graphs, we find that our WAODAGs clearly reflect unique partial orderings. Any explanations constructed under cost-based abduction properly determined the proof sequence. By adding cyclicity, these new more general graphs can be viewed as collections of WAODAGs. Our goal now is to be able to form proper explanations.

We now present our new model of cost-based abduction called *generalized cost-based abduction.*

DEFINITION 5.1. *A* cost-based graph *is a 4-tuple* (G, c, r, S)*, where*

1. *G is a directed graph,* $G = (V, E)$.
2. *c is a function from* $V \times \{\mathsf{true}, \mathsf{false}\}$ *to* $\Re$*, called the* cost function.
3. *r is a function from V to* $\{\text{AND}, \text{OR}\}$*, called the* label.
4. *S is a subset of* $V \times \{\mathsf{true}, \mathsf{false}\}$ *called the* evidence.

Clearly, a cost-based graph is a generalization of our original WAODAGs.

DEFINITION 5.2. *Given a cost-based graph* $Z = (G, c, r, S)$ *where* $G = (V, E)$*, a* solution graph $G_s = (V_s, E_s)$ *for Z is a subgraph of G which satisfies the following conditions:*

1. *G_s is acyclic.*
2. *If* $q \in V_s$ *and* $r(q) = \text{AND}$*, then* $D_q \subseteq V_s$ *and* $(p, q) \in E_s$ *for all* $p \in D_q$.
3. *If* $D_q \subseteq V_s$ *and* $r(q) = \text{AND}$*, then* $q \in V_s$ *and* $(p, q) \in E_s$ *for all* $p \in D_q$.
4. *If* $q \in V_s$ *and* $r(q) = \text{OR}$*, then there exists a node p in D_q such that* $p \in V_s$ *and* $(p, q) \in E_s$.
5. *If* $D_q \cap V_s \neq \phi$ *and* $r(q) = \text{OR}$*, then* $q \in V_s$ *and there exists a node* $p \in D_q \cap V_s$ *such that* $(p, q) \in E_s$.
6. *If* $(q, \mathsf{true}) \in S$*, then* $q \in V_s$.
7. *If* $(q, \mathsf{false}) \in S$*, then q is not in V_s.*

Basically, solution graphs correspond to explanations in that they represent all the nodes which are to be assigned true. Nodes not included in a solution graph are assumed to be set to false. The condition of acyclicity will guarantee that we have a proper proof sequence.

DEFINITION 5.3. *We define the* cost *of a solution graph* $G_s = (V_s, E_S)$ *for* $Z = (G, c, r, S)$ *where* $G = (V, E)$ *as*

$$C(G_s) = \sum_{q \in V_s} c(q, \mathsf{true}) + \sum_{q \in V - V_s} c(q, \mathsf{false}). \tag{35}$$

A solution graph G_s which minimizes C is called a best explanation *for Z.*

We have now completely defined our *generalized cost-based abduction* model. We can also easily see that our original cost-based abduction model is a special case of our new model.

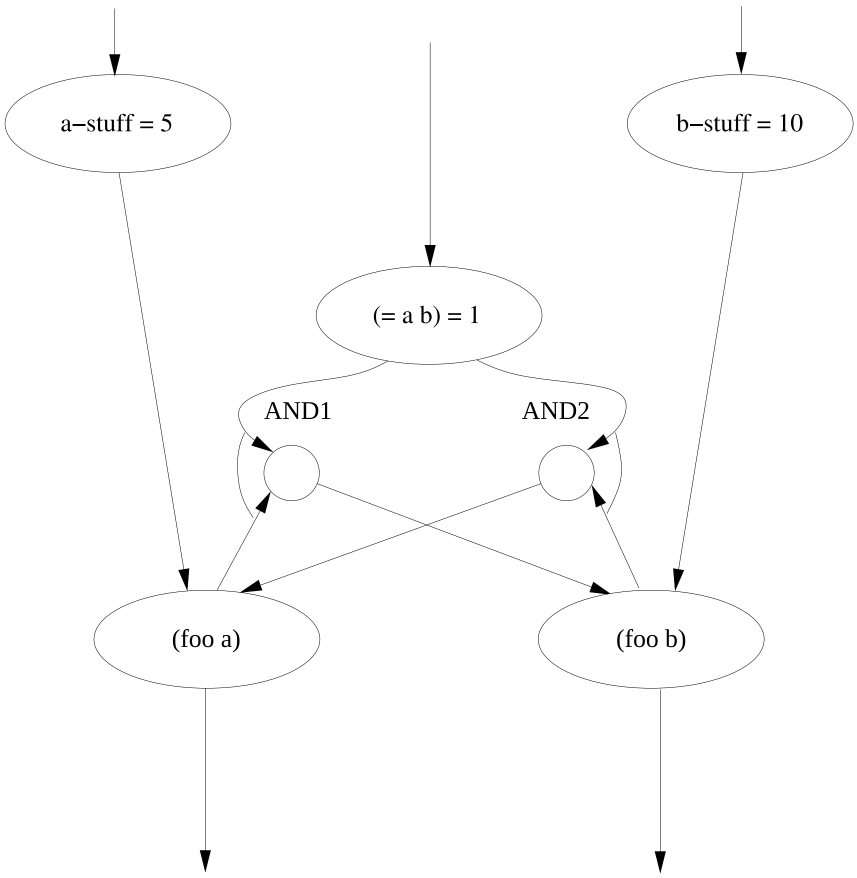


FIG. 5.1. A cost-based graph for our WIMP example.

Let's return to our example above and consider the associated cost-based graph (see Figure 5.1). Our "self-supporting" anomaly arises when we have the following truth assignment to our nodes: (foo a) = (foo b) = true, (= a b) = true and a-stuff = b-stuff = false. We can easily see that there is no solution graph of our cost-based graph which has this assignment. They all end up violating either the edge inclusion requirements or the acyclicity constraint.

5.2 Constraints Formulation - Cycles

We now consider how we might reduce generalized cost-based abduction to linear constraint satisfaction[21]. As we mentioned at the beginning of this thesis, our work on cyclicity was motivated by our constrained optimization view on cost-based abduction. We were trying to see how we might manipulate linear constraints in order to achieve our desired behaviors. We first found that we could properly satisfy the behavior list in the previous section by modifying the original set of constraints generated by Definition 3.5.

We first eliminate any constraints involving (foo a) and (foo b). We then introduce the following new constraints:

$$\begin{aligned}
(\mathsf{foo}\ a) + (=\ a\ b\ ) - 1 &\leq (\mathsf{foo}\ b) \\
(\mathsf{foo}\ b) + (=\ a\ b\ ) - 1 &\leq (\mathsf{foo}\ a) \\
(\mathsf{foo}\ a) + (\mathsf{foo}\ b) + (=\ a\ b\ ) - 2 &\leq \mathsf{a\text{-}stuff} + \mathsf{b\text{-}stuff} \\
(\mathsf{foo}\ a) - (=\ a\ b\ ) &\leq \mathsf{a\text{-}stuff} \\
(\mathsf{foo}\ b) - (=\ a\ b\ ) &\leq \mathsf{b\text{-}stuff} \\
(\mathsf{foo}\ a) &\geq \mathsf{a\text{-}stuff} \\
(\mathsf{foo}\ b) &\geq \mathsf{b\text{-}stuff}
\end{aligned}$$

Clearly, we can associate these constraints directly to the items in the behavior list. We can readily show that this new set of constraints will result in the behaviors we dictated. Thus, our solution demonstrates that our linear constraint satisfaction approach can be extended to handle at least the cyclicity problem in the previous section. However, when we proceed to more general cyclicity problems which may involve interactions between large numbers of propositions, using this sort of brute force enumeration of relationships is certainly infeasible. Even with moderate problems, this brute force approach grows exponentially in the number of constraints we would need to introduce. Just consider introducing (foo c) and other appropriate accoutrements.

Obviously, we need a general solution to cyclicity in linear constraint satisfaction which could compactly capture the essence of our problem.

[21]An earlier cycles formulation for generalized cost-based abduction can be found in [58]

Disallowing the presence of logical cycles in our solution graphs is tantamount to removing the associated 0-1 assignments from our 0-1 solution space. Intuitively, we introduce new variables in our construction to serve as indicators of whether or not the nodes are logically linked, that is, whether one node is directly used to prove the other. For an AND-node, when it is true, then all its parents must be used. However, for an OR-node, only some of its parents need be used. A situation can arise where although a parent is true, it is not used as a proof for the child. This must certainly occur in the presence of cycles. The new variables will detect this case. By taking these new variables plus the AND-nodes of a cycle, we can determine whether a causal chain is legitimate.

We now present our solution.

Let $Z = (G, c, r, S)$ where $G = (V, E)$ be a cost-based graph. Our goal is to construct an equivalent constraint system $L(Z) = (\Gamma, I, \psi)$.

NOTATION. Given a node q in V, we define the *in-degree* of q to be

$$|\{(p, q)|p \in V \text{ and } (p, q) \in E\}|.$$

We denote this by $\mathrm{IN}(q)$. We define the *out-degree* of q to be

$$|\{(q, p)|p \in V \text{ and } (p, q) \in E\}|$$

and denote it by $\mathrm{OUT}(q)$.

DEFINITION 5.4. *A subgraph $J = (V_J, E_J)$ of G is called a* cycle *if and only if J is strongly connected and for all $p \in V_J$,*

$$\mathrm{IN}(p) = \mathrm{OUT}(p) = 1.$$

Furthermore, we denote the set of all cycles for G by $\xi(G)$. An edge $(p, q) \in E$ is said to be encycled *if and only if there exists a cycle in $\xi(G)$ which also contains (p, q).*

We begin by first constructing all the real variables for $L(Z)$. For each $p \in V$, associate a real variable $x_p \in \Gamma$. Next, for each encycled edge $(p, q) \in E$ such that $r(q) = \text{OR}$, associate the real variable $m_{pq} \in \Gamma$. These are the new detection variables we mentioned above.

Now we construct the constraints I. For each node $q \in V$,

1. If $r(q) = \text{AND}$, then construct the following constraints:

$$x_q \leq x_p \text{ for each } p \in D_q \tag{36}$$

$$\sum_{p \in D_q} x_p - |D_q| + 1 \leq x_q. \tag{37}$$

   These two constraints are the same AND-node constraints found in our original formulation.

2. If $r(q) = \text{OR}$, then construct the variable set $t(q)$ as follows: For each $p \in D_q$, if there exists a cycle $J = (V_J, E_J)$ in $\xi(G)$ such that $(p,q) \in E_J$, then $m_{pq} \in t(q)$. Otherwise $x_p \in t(q)$. Next, construct the following constraints:

$$\sum_{x \in t(q)} x \geq x_q \tag{38}$$

$$x_q \geq x_p \text{ for each } p \in D_q \tag{39}$$

$$m_{pq} \leq x_p \text{ for each } m_{pq} \in t(q). \tag{40}$$

   Equations (38) and (39) are modified OR-node constraints to take into account encycled edges. Equation (40) is necessary to properly tie in these new variables.

For each $(p, b) \in S$, if $b = \mathsf{true}$, then construct the constraint $x_p = 1$, otherwise, $x_p = 0$. Finally, for each cycle $J = (V_J, E_J)$ in $\xi(G)$, construct the variable set $t(J)$ as follows: For each $(p,q) \in E_J$, if $r(q) = \text{AND}$, then $x_q \in t(J)$. Otherwise, $m_{pq} \in t(J)$. Construct the constraint,

$$\sum_{x \in t(J)} x \leq |t(J)| - 1. \tag{41}$$

This last constraint will be used to guarantee that we have no cycles in our solution graph. Consider the case where we only have encycled edge variables, m_{pq}, in $t(J)$. Clearly, if all the m_{pq}'s are 1, we have a cycle in the graph.

Lastly, we define ψ as follows: $\psi(x_q, X) = c(q, X)$ for all $q \in V$ and $X \in \{\mathsf{true}, \mathsf{false}\}$.

DEFINITION 5.5. *$L(Z)$ constructed above is the constraint system induced by Z.*

We now prove that our induced constraint system will indeed determine the best explanation in our cost-based graph. Given a solution graph $G_s = (V_s, E_s)$ for Z, we can construct a 0-1 assignment $s[G_s]$ for $L(Z)$ as follows (for notational convenience, $s[G_s](x) \equiv s(x)$): We begin by satisfying the condition that $p \in V_s$ if and only $s(x_p) = 1$. Next, for each $(p,q) \in E_s$, if (p,q) is encycled, then $s(m_{pq}) = 1$. Otherwise $s(m_{pq}) = 0$.

Conversely, given a 0-1 solution s, we can construct a subgraph $G_s[s] = (V_s[s], E_s[s])$ for Z as follows (for notational convenience, $G_s[s] \equiv G_s$): Again, we begin by satisfying the condition that $s(x_p) = 1$ if and only if $p \in V_s$. Next, for each node $q \in V$, if $r(q) = \text{AND}$ and $s(x_q) = 1$, then for all $p \in D_q$, $(p,q) \in E_s$. For each encycled edge $(p,q) \in E$, $(p,q) \in E_s$ if and only if $s(m_{pq}) = 1$.

We can now prove the following:

THEOREM 5.1. *$G_s = (V_s, E_s)$ is a solution graph for $Z = (G, c, r, S)$ where $G = (V, E)$ iff $s[G_s]$ is a 0-1 solution for $L(Z)$.*

THEOREM 5.2. *G_s is a best explanation iff $s[G_s]$ is an optimal 0-1 solution.*

The above theorems show that we can solve generalized cost-based abduction by reducing it to linear constraint satisfaction.

Let's return to our earlier cyclicity example in Figure 5.1. In our new set of constraints, we have the following to deal with the cyclicity:

$$\begin{aligned}
(\text{foo } a) &\leq \text{a-stuff} + m_1 \\
m_1 &\leq \text{AND}_1 \\
\text{AND}_1 &\leq (\text{foo } a) \\
(\text{foo } b) &\leq \text{b-stuff} + m_2 \\
m_2 &\leq \text{AND}_2 \\
\text{AND}_2 &\leq (\text{foo } b) \\
m_1 + \text{AND}_1 + m_2 + \text{AND}_2 &\leq 3
\end{aligned}$$

where AND_1 and AND_2 are the AND-nodes and m_1, m_2 are the special marker variables. We can easily see that the anomalous truth assignment $\text{AND}_1 = \text{AND}_2 = m_1 = m_2 = 1$ will violate the last constraint above, thus demonstrating that our linear constraints properly eliminates the "self-supporting" assignment.

The only additional requirement needed by our new linear constraint satisfaction formulation is generating all the cycles in the cost-based graph. However, we note that the problem of generating the cycles can be easily done in $O((|V|+|E|)(n+1))$ where n is the number of cycles generated. Basically, a depth-first search is used whereby through an elegant approach to adding edges will generate each cycle. Details and analysis of the algorithm can be found in [50].

5.3 Constraints Formulation - Topological

The cycles formulation of the previous section was simple and computationally efficient. It only required the additional process of identifying the cycles in the cost-based graph. This could be easily done in time linear to the size of the graph for each cycle. However, in the unlikely event that our graph is completely connected or approaches it, the number of cycles may be large with respect to the number of nodes.

In this section, we present an alternative constraints formulation for generalized cost-based abduction which does not need to identify the cycles in the graph.

A subgraph of a digraph is acyclic if and only if it provides a topological ordering of its nodes. Our basic approach, like the previous one, is to eliminate cyclic digraphs as possible solution graphs. Instead of working explicitly with cycles, we generalized our approach by requiring that there must be a topological ordering on nodes.

Intuitively, we determined whether there is a topological ordering for a given digraph by finding an assignment of real values to each node such that if a node p is an ancestor of a node q, then p's value must be less than q's. Recall, p is a parent of q if p is one of the causes of q.

In our formulation, we will associate with each node p a real variable t_p to hold p's topological number.[22] For a topological ordering to occur between two

[22] t_p does not need to be restricted to integer values.

nodes, we must have used a sequence of rules tying to two nodes together. Since implication is transitive, we could locally restrict ourselves to nodes which are adjacent to each other. For example, $p_1 \Longrightarrow p_2$ and $p_2 \Longrightarrow p_3$ implies the ordering $t_{p_1} < t_{p_2}$ and $t_{p_2} < t_{p_3}$. From transitivity, $p_1 \Longrightarrow p_3$ which must imply $t_{p_1} < t_{p_3}$ and is already the case from "$<$"'s transitivity.

We formed solution graphs from a given cost-based graph by essentially determining which rules to use. We consider a rule to be "conjunctive" if the precedent is a conjunction of one or more nodes and the antecedent is a single node,

$$p_1 \wedge \ldots \wedge p_n \Longrightarrow q.$$

Obviously, we can take any set of rules and create an equivalent set composing only of conjunctive rules. In terms of our cost-based graph, an AND-node with all of its parents would constitute a single conjunctive rule whereas an OR-node with represent one rule for each of its children.

Basically, if we used the conjunctive rule,

$$p_1 \wedge \ldots \wedge p_n \Longrightarrow q,$$

then we must have the following topological constraints:

$$t_{p_i} < t_q \text{ for } i = 1, \ldots, n.$$

Now, we can easily determine whether a collection of rules is acyclic. If the collection is cyclic, then there must exist some nodes p and q such that p is a descendant of q and vice versa. According to the transitivity property above, we have both $t_p < t_q$ and $t_q < t_p$ which is clearly a contradiction rendering our constraints infeasible.

We begin by first constructing all the real variables for $L(Z)$. For each $p \in V$, associate a real variable $x_p \in \Gamma$. Next, for each edge $(p, q) \in E$ such that $r(q) = \text{OR}$, associate the real variable $m_{pq} \in \Gamma$. Intuitively, the variable m_{pq} determines whether the logical rule $p \Longrightarrow q$ has been used or not. Furthermore, m_{pq} is directly related to the edge between p and q. Finally, for each $p \in V$, associate the topological variable t_p.

Now we construct the constraints I. For each node $q \in V$,

1. If $r(q) = \text{AND}$, then construct the following constraints:

$$x_q \leq x_p \text{ for each } p \in D_q \tag{42}$$

$$\sum_{p \in D_q} x_p - |D_q| + 1 \leq x_q \tag{43}$$

$$2|V|(1 - x_q) + t_q \geq t_p + 1 \text{ for each } p \in D_q. \tag{44}$$

   Again, equations (42) and (43) are our original AND-node constraints. On the other hand, equation (44) is used to model the following event: If AND-node q is true, then all the parents of q are true and must topological dominate q. Otherwise, no action is taken.

2. If $r(q) = \text{OR}$, then construct the following constraints:

$$\sum_{p \in D_q} m_{pq} \geq x_q \tag{45}$$

$$x_q \geq x_p \text{ for each } p \in D_q \tag{46}$$

$$m_{pq} \leq x_p \text{ for each } p \in D_q \tag{47}$$

$$2|V|(1 - m_{pq}) + t_q \geq t_p + 1 \text{ for each } p \in D_q. \tag{48}$$

   Equations (45) to (47) are similar to equations (38) to (40). Equation (48) is used to guarantee the appropriate topological ordering if necessary.

Finally, for each $(p, b) \in S$, if $b = \mathsf{true}$, then construct the constraint $x_p = 1$, otherwise, $x_p = 0$.

Lastly, we define ψ as follows: $\psi(x_q, X) = c(q, X)$ for all $q \in V$ and $X \in \{\mathsf{true}, \mathsf{false}\}$.

We now go about and prove that our induced constraint system will indeed determine the best explanation in our cost-based graph. Given a solution graph $G_s = (V_s, E_s)$ for Z, we can construct a 0-1 assignment $s[G_s]$ for $L(Z)$ as follows (for notational convenience, $s[G_s](x) \equiv s(x)$): We begin by satisfying the condition that $p \in V_s$ if and only $s(x_p) = 1$. Next, $(p, q) \in E_s$ and $r(q) = \text{OR}$ if and only if $s(m_{pq}) = 1$. Now, since G_s is acyclic, assign a topological ordering to V_s and set the appropriate values for $s(t_p)$ where $p \in V_s$. (A labeling of integers from 1 to $|V|$ is suggested.) For all $p \in V - V_s$, $s(t_p) = 0$.

Conversely, given a 0-1 solution s, we can construct a subgraph $G_s[s] = (V_s[s], E_s[s])$ for Z as follows (for notational convenience, $G_s[s] \equiv G_s$): Again, we begin by satisfying the condition that $s(x_p) = 1$ if and only if $p \in V_s$. Next, for each node $q \in V$, if $r(q) = $ AND and $s(x_q) = 1$, then for all $p \in D_q$, $(p, q) \in E_s$. For each edge $(p, q) \in E$, $(p, q) \in E_s$ and $r(q) = $ OR if and only if $s(m_{pq}) = 1$.

We can now prove the following:

THEOREM 5.3. *$G_s = (V_s, E_s)$ is a solution graph for $Z = (G, c, r, S)$ where $G = (V, E)$ iff $s[G_s]$ is a 0-1 solution for $L(Z)$.*

THEOREM 5.4. *G_s is a best explanation iff $s[G_s]$ is an optimal 0-1 solution.*

In the cycles formulation, we identified cycles in order to explicitly "break" them. This achieved the effect of properly ordering the propositions. We observed that this ordering actually corresponds to a topological ordering of all the nodes.

As we can easily see, our topological formulation is not dependent on the cycles in the digraph. The size of the constraint system is once again dependent only on the original size of the given digraph. Obviously, there are cases where the cycles formulation results in a smaller system of constraints, especially when we have a large number of nodes but a relatively small number of cycles.

5.4 Discussion

We have considered the problem of cyclicity with regards to cost-based abduction and presented generalized cost-based abduction to appropriately incorporate the phenomenon. This was accomplished by studying cyclicity from the standpoint of linear constraint satisfaction which allowed us to concretely identify the problem. Our goal was to determine whether we could manipulate the constraints from the original cost-based abduction problem to handle cyclicity. Indeed, we demonstrated that the problem could be solved through a natural extension of our linear constraint satisfaction approach. Furthermore, the extensions required seemed quite compact in size.

6 Conclusion

In this thesis, we presented a new approach to modeling abduction which may alleviate the computational difficulty in finding a best explanation.

First, we transformed cost-based abduction into linear constraint satisfaction. This allowed us to solve for minimal-cost proofs by using highly efficient tools and techniques developed in Operations Research. So far, experiments show that our technique outperforms the existing best-first heuristics by exhibiting an expected polynomial run-time growth rate as opposed to their expected exponential growth rate. Our tests were performed both on randomly generated problems and on ones generated by the WIMP story understanding system. Furthermore, we found that various optimizations such as incorporating an initial solution significantly improved our performance without increasing the complexity of our formulation. As a result, this now provided WIMP a practical means of performing abductive reasoning, a facility it was lacking earlier [6].

We further demonstrated that our approach could naturally generate the alternative explanations. This can be important in domains like medical diagnosis where the availability of alternatives is critical.

Next, we extended our approach and modelled Bayesian networks. In particular, we show that MPE problems for belief revision could also be transformed into linear constraints. Our formulation overcomes a limitation of the existing message-passing scheme. We are capable of readily generating all the alternative explanations in order of probability. Furthermore, our approach could address the additional issues of circumscribing explanations and focusing.

We also found that belief updating could be modelled based on our belief revision formulation. From our various studies, we found that our approach could solve the class of deterministic Bayesian networks which have been stumbling blocks for existing simulation techniques.

Preliminary work was presented which addressed the issue of combinatorial explosions of conditional probability table sizes in Bayesian networks. Through linear constraint satisfaction, we demonstrated that certain compressions may be achieved. This new class of constraint systems was also shown to properly subsume the formulations for belief revision. Thus, we can now hopefully solve Bayesian networks with near-continuous r.v.s.

Finally, we considered the problem of cyclicity. We provided a new framework called generalized cost-based abduction which permits cyclicity. Through our linear constraint satisfaction approach, we were able to naturally formulate and solve problems in this new framework.

Putting all this together, we believe our linear constraint satisfaction approach to be quite promising. Furthermore, our approach opens up some new and interesting lines of research for abduction and constraint satisfaction. Some are of a theoretical nature while others are more practical:

- From our cost-based abduction experiments, we found that a significant portion of the minimal-cost proofs had optimal solutions which were also 0-1 solutions. A question naturally arises as to whether there is something special about our domain which causes this. Can we predict when a 0-1 solution will be the optimal solution?
- There exists domain dependent methods for solving linear programming problems other than the general ones like Simplex [44, 30, 32, 48, 76]. These methods have been shown to outperform the general ones for their specific domains. For example, there are approximation methods for solving integer linear programming problems which have theoretically proven error bounds. Can we also find such a technique tailored to our abductive models?
- Parallel algorithms exist for solving linear programming problems [21, 25]. Can we get the expected sub-linear run-times from these algorithms?
- In systems like WIMP and TACITUS [26], the reasoning mechanisms are essentially broken down into two components. The first component involves constructing a network of propositional rules from a first-order logic knowledge-base. The second component then takes this network as the basis for its abductive computations. For example, WIMP constructs a network and then uses cost-based abduction. So far, we have only concentrated on the second component in our formulation since it has been the major stumbling block for systems like WIMP [23]. Obviously, we should now consider how we might model the first component with our approach with the

eventual goal of merging both within one constraint systems formulation.

- Splined Bayesian networks introduce a new flexibility in transforming and manipulating our data. We hope to continue to study them and attempt to determine the limits of this model.
- Finally, again, from our cost-based experiments, we found that our system worked very well on moderate size WIMP generated networks. Since linear programming methods were tailored for problems in the tens of thousands of variables, we would like to construct very large and hopefully realistic domains and attempt to apply our approach.

References

[1] Douglas E. Appelt. A theory of abduction based on model preference. In *Proceedings of the AAAI Symposium on Abduction*, 1990.

[2] Colin Bell, Anil Nerode, Raymond T. Ng, and V. S. Subrahmanian. Implementing deductive databases by linear programming. Technical Report CS-TR-2747, University of Maryland, 1991.

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

[4] Eugene Charniak and Saadia Husain. A new admissible heuristic for minimal-cost proofs. In *Proceedings of the AAAI Conference*, 1991.

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

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

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

[8] Gregory F. Cooper. *NESTOR: A Computer-based Medical Diagnostic Aid That Integrates Causal and Probabilistic Knowledge*. PhD thesis, Department of Computer Science, Stanford University, 1984.

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

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

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

[12] J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

[13] A. P. Dempster. A generalization of bayesian inference. *J. Royal Statistical Society*, 30:205–47, 1968.

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

[15] J. A. Feldman and D. H. Ballard. Connectionist models and their properties. *Cognitive Science*, 9:51–74, 1985.

[16] Watson Fulks. *Advanced Calculus*. John Wiley & Sons, Inc., 1978.

[17] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[18] Robert S. Garfinkel and George L. Nemhauser. *Integer Programming*. John Wiley & Sons, Inc., 1972.

[19] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distribution, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–41, 1984.

[20] Michael R. Genesereth. The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24:411–436, 1984.

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

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

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

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

[25] F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research*. Holden-Day, Inc., 1967.

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

[27] V. Isham. An introduction to spatial point processes and markov random fields. *International Statistical Review*, 49:21–43, 1981.

[28] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. Bayesian updating in recursive graphical models by local computations. Technical Report Report R 89-15, Institute for Electronic Systems, Department of Mathematics and Computer Science, University of Aalborg, Denmark, 1989.

[29] John R. Josephson. Abduction: Conceptual analysis of a fundamental pattern of inference. Technical Report 91-JJ-DRAFT, The Ohio State University, 1991.

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

[31] Henry A. Kautz and James F. Allen. Generalized plan recognition. In *Proceedings of the AAAI Conference*, 1986.

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

[33] S. L. Lauritzen. *Lectures on Contigency Tables*. University of Aalborg Press, 1982.

[34] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their applications to expert systems. *J. Royal Statistical Society*, 50(2):157–224, 1988.

[35] Richard Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, pages 4–22, April 1987.

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

[37] V. S. Mikhalevich, V. L. Volkovich, and G. V. Kolenov. An algorithm for coordination of solutions in a distributed system of interdependent problems with linear models. *Kibernetika*, 3:1–8+22, 1988.

[38] R. A. Miller, H. E. Poole, and J. P. Myers. Internist-1: An experimental computer-based diagnostic consultant for general internal medicine. *New England Journal of Medicine*, 307:468–70, 1982.

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

[40] N. J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.

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

[42] Y. Peng and J. A. Reggia. Plausibility of diagnostic hypothese: The nature of simplicity. In *Proceedings of the AAAI Conference*, 1986.

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

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

[45] Harry Pople. The formation of composite hypotheses in diagnostic problem solving: An exercise in synthetic reasoning. In *Proceedings of the IJCAI Conference*, 1977.

[46] Harry Pople. Heuristic methods for imposing structures on ill-structured problems. In P. Szolovits, editor, *Artificial Intelligence in Medicine.* Boulder, CO: Westview, 1982.

[47] Harry Pople, Jack Myers, and Randolph Miller. Dialog: A model of diagnostic logic for internal medicine. In *Proceedings of the IJCAI Conference*, 1975.

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

[49] Prabhakar Raghavan. A provably good routing in graphs: Regular arrays. *ACM*, pages 79–87, 1985.

[50] Edward M. Reingold, Jurg Nievergelt, and Narsingh Deo. *Combinatorial Algorithms: Theory and Practice.* Prentice-Hall, Inc., 1977.

[51] Igor Rivin and Ramin Zabih. An algebraic approach to constraint satisfaction problems. In *Proceedings of the IJCAI Conference*, 1989.

[52] David E. Rumelhart and James L. McClelland. *Parallel Distributed Processing: Explorations in the Microstrctures of Cognition, Volume 1.* The MIT Press, 1986.

[53] Eugene Santos, Jr. Cost-based abduction and linear constraint satisfaction. Technical Report CS-91-13, Department of Computer Science, Brown University, 1991.

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

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

[56] Eugene Santos, Jr. Modelling cyclicity and generalized cost-based abduction using linear constraint satisfaction. Submitted for publication, 1991.

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

[58] Eugene Santos, Jr. A linear constraint satisfaction approach to cyclicity. Technical Report CS-92-03, Department of Computer Science, Brown University, 1992.

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

[60] Bart Selman and Hector J. Levesque. Abductive and default reasoning: A computational core. In *Proceedings of the AAAI Conference*, pages 343–348, 1990.

[61] Ross D. Shachter. Evaluating influence diagrams. *Operations Research*, 36:871–82, 1986.

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

[63] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

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

[65] Solomon E. Shimony. On irrelevance and partial assignments to belief networks. Technical Report CS-90-14, Department of Computer Science, Brown University, 1990.

[66] Solomon E. Shimony. Algorithms for finding irrelevance-based map assignments to belief networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 1991.

[67] Solomon E. Shimony. Explanation, irrelevance and statistical independence. In *Proceedings of the AAAI Conference*, 1991.

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

[69] Solomon Eyal Shimony. *A Probabilistic Framework for Explanation*. PhD thesis, Department of Computer Science, Brown University, 1991.

[70] E. H. Shortliffe. *Computer-Based Medical Consultation: MYCIN*. Elsevier, 1976.

[71] E. H. Shortliffe and B. G. Buchanan. A model of inexact reasoning in medicine. *Mathematical Biosciences*, 23:351–379, 1975.

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

[73] Mark E. Stickel. A prolog-like inference system for computing minimum-cost abductive explanations in natural-language interpretation. Technical Report Technical Note 451, SRI International, 1988.

[74] Paul Thagard. Explanatory coherence. *Behavioral and Brain Sciences*, 12:435–502, 1989.

[75] R. Yager. Using approximate reasoning to represent default knowledge. *Artificial Intelligence*, 31:99–112, 1987.

[76] Xinhua Zhuang, Robert M. Haralick, and Hyonam Joo. A simplex-like algorithm for the relaxation labeling process. *IEEE Transactions on PAMI*, 11(12):1316–21, 1989.

A Proofs

THEOREM 3.1. *If e is an explanation for W, then s_e is a 0-1 solution for $L(W)$.*

Proof. Assume that s_e is not a solution of $L(W) = (\Gamma, I, \psi)$. This implies that there exists a constraint Q in I which is violated. (For notational convenience, we will denote $s_e(x_p) = a$ by $x_p = a$.)

Case 1. Q is of the form $x_p \leq x_q$.

Since s_e is a 0-1 assignment, $x_p = 1$ and $x_q = 0$. From (2) and (5), we get can conclude that either $r(p) =$ AND or $r(q) =$ OR. If $r(p) =$ AND, then q is a parent of p and x_q must equal 1 in s_e. If $r(p) =$ OR, then p is a child of q and x_p must equal 0 in s_e. Since neither is the case, Q cannot be violated.

Case 2. Q is of the form

$$\sum_{q \in D} x_q - |D| + 1 \leq x_p$$

where D is some set of nodes.

For Q to be violated,

$$\sum_{q \in D} x_q - x_p > |D| - 1.$$

This implies that $x_p = 0$ and for all $q \in D$, $x_q = 1$. From (3), we can conclude that $r(p) =$ AND and $D = D_p$. Since $r(p)$ is an AND node, if x_p equals 0, then all the parents of p must also equal zero. Thus, Q cannot be violated.

Case 3. Q is of the form

$$\sum_{q \in D} x_q \geq x_p$$

where D is some set of nodes.

This implies that $x_p = 1$ and for all $q \in D$, $x_q = 0$. From (4), we conclude that $r(p) =$ OR and $D = D_p$. Since $r(p)$ is an OR node, x_p equals 1 implies that there exists an $q \in D_p$ such that $x_q = 1$. Thus, Q cannot be violated.

Cases 1 to 3 cover every type of violations of the constraint system possible. Therefore, e cannot be an explanation for W. □

THEOREM 3.3. *If s is a 0-1 solution for $L_E(W)$, then e_s is an explanation for W.*

Proof. Assume e_s is not an explanation for W. This implies that one or more of the following conditions hold: (For notational convenience, we will denote $s(x_p) = a$ by $x_p = a$ and e_s by e.)

1. There exists an AND node p in W such that $e(p) = \mathsf{true}$ and there exists a $q \in D_p$ such that $e(q) = \mathsf{false}$.
2. There exists an AND node p in W such that $e(p) = \mathsf{false}$ and for all $q \in D_p$, $e(q) = \mathsf{true}$.
3. There exists an OR node p in W such that $e(p) = \mathsf{true}$ and for all $q \in D_p$, $e(q) = \mathsf{false}$.
4. There exists an OR node p in W such that $e(p) = \mathsf{false}$ and there exists a $q \in D_p$ such that $e(q) = \mathsf{true}$.
5. There exists an evidence node p in S such that $e(p) = \mathsf{false}$.

Case 1. From (2), $r(p) = $ AND implies that the constraint $x_p \leq x_q$ is in I. Since, $x_p = 1$ and $x_q = 0$, condition 1 does not hold.

Case 2. From (3), $r(p) = $ AND implies that $x_p \geq 1$. Thus, condition 2 does not hold.

Case 3. From (4), $r(p) = $ OR implies that $x_p \leq 0$. Thus, condition 3 does not hold.

Case 4. From (5), $r(p) = $ OR implies that $x_p \geq 1$. Thus, condition 4 does not hold.

Case 5. From (6), $p \in S$ implies that $x_p = 1$. Thus, condition 5 does not hold.

Therefore, s is not a 0-1 solution in $L_E(W)$. □

THEOREM 3.6. *Let $W = (G, c, r, S)$ be a* WAODAG *where $G = (V, E)$. An optimal 0-1 solution for $\hat{L}_E(W)$ can be transformed into a best explanation for W in $O(|E|)$ steps if and only if $c(p, \mathsf{false}) \leq c(p, \mathsf{true})$ for all nodes p in V.*

Proof. Let s be any optimal 0-1 solution for $\hat{L}_E(W)$. Assume e_s is not an explanation for W. This implies that one or more of the following conditions hold: (For notational convenience, let e denote e_s.)

1. There exists an AND node p in W such that $e(p) = \mathsf{true}$ and there exists a $q \in D_p$ such that $e(q) = \mathsf{false}$.

2. There exists an AND node p in W such that $e(p) = \mathsf{false}$ and for all $q \in D_p$, $e(q) = \mathsf{true}$.
3. There exists an OR node p in W such that $e(p) = \mathsf{true}$ and for all $q \in D_p$, $e(q) = \mathsf{false}$.
4. There exists an OR node p in W such that $e(p) = \mathsf{false}$ and there exists a $q \in D_p$ such that $e(q) = \mathsf{true}$.
5. There exists an evidence node p in S such that $e(p) = \mathsf{false}$.

From Definition 3.9, conditions 1, 3, and 5 cannot hold. We now only consider conditions 2 and 4.

We can view the process of finding a suitable 0-1 solution as the propagation of information from evidence nodes through AND / OR nodes to hypothesis nodes. The remaining conditions, 2 and 4, indicate that zero assignments do not propagate. Let M be the set of nodes p in W such that $e(p) = \mathsf{false}$ and whose childrens' assignment permits either condition 2 or 4 to hold. Let M' be the subset of M such that each node in M' does not have a descendant also in M.

Let p be any node in M' and D_p^{-1} be the immediate children of node p. For each q in D_p^{-1}, one of the following holds:

1. $e(q) = \mathsf{false}$ and $r(q) = $ AND.
2. $e(q) = \mathsf{false}$ and $r(q) = $ OR.
3. $e(q) = \mathsf{true}$ and $r(q) = $ OR.

$e(q) = \mathsf{true}$ and $r(q) = $ AND cannot both be true since it would violate the definition of M'. For the third combination, there exists some node $p' \neq p$ in D_q such that $e(p') = \mathsf{true}$.

Let W_M be the resulting WAODAG obtained by first removing all nodes and associated edges from W in M and then removing those which are not reachable from any evidence node in S. We can easily see that any AND-DAG obtained from W_M is an AND-DAG for W.

Since s is an optimal 0-1 solution for $\hat{L}_E(W)$ and $c(q, \mathsf{false}) \leq c(q, \mathsf{true})$ for any node q, for each hypothesis node p in W but not in W_M, $e(p)$ can be set to false. (It can be easily shown that if $e(p) = \mathsf{true}$, then $c(p, \mathsf{false}) = c(p, \mathsf{true})$.) Now, by propagating the truth values from the hypothesis nodes, another optimal 0-1 solution will be generated. This new solution will be a best explanation for W.

We can easily determine M and the final 0-1 solution in $O(|E|)$ steps. □

THEOREM 3.9. *Let s be the optimal solution of some* WAODAG *semi-induced linear program. Construct a variable assignment s' from s by changing all non-zero values in s to 1. s' is a 0-1 solution for the constraint system.*

Proof. Assume that s' is not a 0-1 solution for the constraint system. This implies that there exists a nonzero variable x_q such that by setting x_q to 1, this violates some constraint Q in the constraint system.

Case 1. $x_q \leq x_p$ is violated. Since x_q is nonzero, this implies that x_p was also nonzero to begin with. Thus x_p would have also been set to 1. Thus, this constraint is not violated.

Case 2. $\sum_{p \in D_q} x_p \geq x_p$ is violated. Since x_q is nonzero, this implies that some x_p where $p \in D_q$ is also nonzero. Thus, x_p would have also been set to 1. Thus, this constraint is not violated.

All the cases for constraint violations have been considered. No constraint is violated. Contradiction. □

THEOREM 3.10. *s' constructed from s above is a 0-1 solution for L.*

Proof. Follows from Theorems 3.1 and 3.3. □

THEOREM 3.11. *Constraint system L_n in Algorithm 3.2 determines the n-th best 0-1 solution for L.*

Proof. We can show that the additional constraint eliminates exactly one additional 0-1 solution from the set of 0-1 solutions in L. Also, the 0-1 solution eliminated will be the optimal 0-1 solution determined by L_{n-1}. □

THEOREM 3.15. *Let e_1 and e_2 be explanations for W.*

1. *$H(e_1) \subseteq H(e_2)$ iff $K(e_1) \subseteq K(e_2)$.*
2. *$H(e_1) \subset H(e_2)$ iff $K(e_1) \subset K(e_2)$.*

Proof. We observe that setting any node in W to true never results in some other node assignment in W to change from true to false. Combining this with Propositions 3.12 and 3.13, our theorem follows. □

THEOREM 3.16. *There exists a 1-1 and onto mapping between 2^{V_H} and the set of* *all possible truth assignments for W.*

Proof. Follows from Proposition 3.13. $\square$

THEOREM 3.17. *If e is an explanation for W, then there exists at least $2^{|V_H - H(e)|}$ explanations for W which are consistent with $H(e)$.*

Proof. Let e be an explanation for W and let H be a subset of V_H such that $H(e) \subset H$. By Proposition 3.14 and Theorem 3.16, there exists an explanation e' for W such that $H(e') = H$. $\square$

THEOREM 3.19. *A* WAODAG *W is monotonic iff for every explanations e_1 and e_2 for W, $H(e_1) \subseteq H(e_2)$ implies $C(e_1) \leq C(e_2)$. W is strictly monotonic iff W is monotonic, and for every explanations e_1 and e_2 for W, $H(e_1) \subset H(e_2)$ implies $C(e_1) < C(e_2)$.*

Proof. Follows from Theorem 3.15. $\square$

THEOREM 3.20. *If W is strictly monotonic, then any best explanation for W is cardinal.*

Proof. Let e be a best explanation for W. Assume e is not cardinal. Then there exists an explanation e' for W such that $H(e') \subset H(e)$. Since W is strictly monotonic, $C(e') < C(e)$. However, e is a best explanation for W which implies that $C(e) \geq C(e')$. Contradiction. $\square$

LEMMA 3.21. *Let W be strictly monotonic. If s_n is the optimal 0-1 solution for the constraint system L_n in Algorithm 3.3, then s_n is a cardinal solution for L.*

Proof. Suppose s_n is not a cardinal solution for L. Since W is strictly monotonic, there exists a 0-1 solution s for L such that $H(s) \subset H(s_n)$ and $\Theta_L(s) < \Theta_L(s_n)$. Since s has a lower cost than s_n and $s \neq s_n$, we must assume that s does not satisfy one or more of the new constraints in L_n. However, since every new constraint satisfied by s_n is also satisfied by s, it follows that s_n is not a solution for L_n. A contradiction. $\square$

THEOREM 3.22. *Let W be strictly monotonic. The constraint system L_n in Algorithm 3.3 determines the n-th best cardinal solution.*

Proof. Let s_n be the optimal 0-1 solution for the constraint system L_n.

Suppose s is a cardinal solution for L and n is the smallest integer such that $\Theta_L(s) < \Theta_L(s_n)$ and $s \neq s_k$ for $k = 1, 2, \ldots, n$. Then s must violate at least one of the new constraints of the form

$$\sum_{x_q \in H(s_k)} x_q \leq |H(s_k)| - 1.$$

Thus, $H(s_k) \subseteq H(s)$. If $H(s_k)$ is a proper subset of $H(s)$, then since W is strictly monotonic, $\Theta_L(s) < \Theta_L(s_k)$. Therefore, s is not cardinal. If $H(s_k) = H(s)$, then from Proposition 3.13, $s_k = s$. Contradiction. Hence, $s = s_k$ for some $k = 1, 2, \ldots, n-1$.

Finally, since the generation of cardinal solutions is ordered by cost, L_n will generate the n-th best cardinal solution. □

THEOREM 3.25. *Let C be the set of variables that appear in Θ_L with non-zero coefficient.*

1. *If C is a subset of Q and all the coefficients in Θ_L are positive, then L is Q-monotonic.*
2. *If $C = Q$ and all the coefficients in Θ_L are positive, then L is strictly Q-monotonic.*

Proof. Follows from the fact that if C is a subset of Q, then $(K(s_1) \cap Q) \subseteq (K(s_2) \cap Q)$ implies $(K(s_1) \cap C) \subseteq (K(s_2) \cap C)$. □

THEOREM 3.26. *Let L be strictly Q-monotonic. Every optimal 0-1 solution for L is Q-cardinal.*

Proof. Let s be the optimal 0-1 solution for L. Assume s is not Q-cardinal. There exists a 0-1 solution s' for L such that $(K(s') \cap Q) \subset (K(s) \cap Q)$. Since L is strictly Q-monotonic, $\Theta_L(s') < \Theta_L(s)$. Thus s is not optimal. Contradiction. □

LEMMA 3.27. *Let L be strictly Q-monotonic. If s_n is the solution of the constraint system L_n in the Algorithm 3.4, then s_n is a Q-cardinal solution for L.*

Proof. Similar to Lemma 3.21. □

THEOREM 3.28. *Let Q be a dominant set and let L be strictly Q-monotonic. The* *constraint system L_n in the Algorithm 3.4 determines the n-th best Q-cardinal solution for L.*

Proof. Let s_n be the solution of the constraint system L_n. Suppose s is a Q-cardinal solution for L and n is the smallest integer such that $\Theta_L(s) < \Theta_L(s_n)$ and $s \neq s_k$ for $k = 1, 2, \ldots, n$. Then s must violate at least one of the new constraints of the form

$$\sum_{x_q \in (K(s_k) \cap Q)} x_q \leq |K(s_k) \cap Q| - 1.$$

Thus, $(K(s_k) \cap Q) \subseteq (K(s) \cap Q)$. Since Q is a dominant set and $s \neq s_k$, $K(s_k) \cap Q$ is a proper subset of $K(s) \cap Q$. But L is strictly Q-monotonic, therefore, s cannot be Q-cardinal. Contradiction. □

THEOREM 3.29. *Let W be a strictly monotonic* WAODAG *and $L(W)$ be the constraint system induced by W. Let Q be the set of variables corresponding to the hypothesis nodes of W in $L(W)$. We have the following:*

1. *Q is strictly dominant.*
2. *$L(W)$ is strictly Q-monotonic.*

Proof. (1) Follows from Definition 3.16. (2) Since W is strictly monotonic, by Definition 3.13, for every two explanations e_1 and e_2 for W, $K(e_1) \subset K(e_2)$ implies $C(e_1) < C(e_2)$. Let s_1 and s_2 be any two 0-1 solutions for $L(W)$ such that $(K(s_1) \cap Q) \subset (K(s_2) \cap Q)$. Since Q is strictly dominant from (1), this implies that $K(s_1) \subset K(s_2)$. Furthermore, since there is a 1-1 and onto mapping between explanations for W and 0-1 solutions for $L(W)$, $\Theta_L(s_1) < \Theta_L(s_2)$. Therefore, $L(W)$ is strictly Q-monotonic. □

THEOREM 4.1. *Let $\mathcal{B} = (V, P)$ be a Bayesian network and $L(\mathcal{B}) = (\Gamma, I, \psi)$ be the constraint system induced by $\mathcal{B}$. Then*

1. *$|\Gamma| = |P| + \sum_{A \in V} |R(A)|$ and*
2. *$|I| = 2|V| + \sum_{p \in P} N(p) \leq 2|V| + |V||P| = (2 + |P|)|V|$ where $N(p)$ is the number of r.v.s which appear as conditionals in the probability.*

Proof. Follows from the above construction. □

THEOREM 4.2. *Given a 0-1 solution s for $L(\mathcal{B})$, for each set of variables $\Delta(A)$, there exists some $\mathsf{A_a}$ in $\Delta(A)$ such that $\mathsf{A_a} = 1$ and $\mathsf{A_{a'}} = 0$ for all $\mathsf{A_{a'}} \neq \mathsf{A_a}$ in $\Delta(A)$.*

Proof. Let s be a 0-1 solution for $L(\mathcal{B})$. Assume there exists a set of variables $\Delta(A)$ such that $\mathsf{A_a} = 0$ for all $\mathsf{A_a}$ in $\Delta(A)$. This implies that

$$\sum_{\mathsf{A_a} \in \Delta(A)} \mathsf{A_a} = 0$$

which violates the constraint formed by (12) for r.v. A. Contradiction.

Instead, assume there exists a set of variables $\Delta(A)$ such that $\mathsf{A_a} = \mathsf{A_{a'}} = 1$ for some $a - a'$. This implies that

$$\sum_{\mathsf{A_a} \in \Delta(A)} \mathsf{A_a} \geq 2$$

which also violates the constraint formed by (12) for r.v. A. Contradiction. $\square$

THEOREM 4.3. *Given a 0-1 solution s for $L(\mathcal{B})$, for all variables $q[\mathsf{A_a} \mid \mathsf{C}_1 = \mathsf{c}_1, \ldots, \mathsf{C}_n = \mathsf{c}_n]$,*

$$q[\mathsf{A_a} \mid \mathsf{C}_1 = \mathsf{c}_1, \ldots, \mathsf{C}_n = \mathsf{c}_n] = 1 \textit{ iff } \mathsf{A_a} = \mathsf{C}_{1_{\mathsf{c}_1}} = \ldots = \mathsf{C}_{n_{\mathsf{c}_n}} = 1.$$

Proof. Let s be a 0-1 solution for $L(\mathcal{B})$.

(*if*) Assume $q[\mathsf{A_a} \mid \mathsf{C}_{1_{\mathsf{c}_1}}, \ldots, \mathsf{C}_{n_{\mathsf{c}_n}}] = 1$. From (13), this implies $\mathsf{C}_{j_{\mathsf{c}_j}} = 1$. From (14),

$$\sum_{x \in \Upsilon_{\mathsf{A}_i}} x = \mathsf{A}_i = 1$$

since $q[\mathsf{A_a} \mid \mathsf{C}_{1_{\mathsf{c}_1}}, \ldots, \mathsf{C}_{n_{\mathsf{c}_n}}] \in \Upsilon_{\mathsf{A}_i}$.

(*only if*) Assume $\mathsf{A_a} = \mathsf{C}_{1_{\mathsf{c}_1}} = \ldots = \mathsf{C}_{n_{\mathsf{c}_n}} = 1$. From (14),

$$\sum_{x \in \Upsilon_{\mathsf{A}_i}} x = 1.$$

Let x be the variable $q[\mathsf{A_a} \mid \mathsf{C}_{1_{\mathsf{d}_1}}, \ldots, \mathsf{C}_{n_{\mathsf{d}_n}}]$. Assume $x = 1$ and for some l, $\mathsf{d}_l \neq \mathsf{c}_l$. From (13), this implies $\mathsf{C}_{l_{\mathsf{d}_l}} = 1$. Since $\mathsf{C}_{l_{\mathsf{c}_l}} = 1$,

$$\sum_{\mathsf{C}_{l_j} \in \Delta(C_l)} \mathsf{C}_{l_j} \geq 2$$

which violates (12). Contradiction. Therefore, for x to be 1, $\mathsf{d}_l = \mathsf{c}_l$ for $l = 1, \ldots, n$. □

THEOREM 4.4. *If s is a 0-1 solution for $L(\mathcal{B})$, then w_s is an instantiation-set for $\mathcal{B}$.*

Proof. Follows from above construction and Theorems 4.2 and 4.3. (□

THEOREM 4.5. *If w is a complete instantiation-set for $\mathcal{B}$, then s_w is a 0-1 solution for $L(\mathcal{B})$.*

Proof. Let w be a complete instantiation-set for $\mathcal{B}$. Assume s_w is not a 0-1 solution for $L(\mathcal{B}) = (\Gamma, I, \psi)$. This implies that there exists some constraint Q in I which is violated by s_w.

Case 1. Q is of the form

$$\sum_{\mathsf{A}_\mathsf{a} \in \Delta(A)} \mathsf{A}_\mathsf{a} = 1.$$

Since s_w is a 0-1 assignment, then from Theorem 4.2, either $\mathsf{A}_\mathsf{a} = 0$ for all A_a in $\Delta(A)$ or there exists $\mathsf{A}_\mathsf{a} = \mathsf{A}_{\mathsf{a}'} = 1$ for some $\mathsf{a} \neq \mathsf{a}'$. This implies that either the r.v. A has no instantiation or more that one instantiation which contradicts the fact that w is a complete instantiation-set for $\mathcal{B}$. Thus, Q cannot be violated.

Case 2. Q is of the form

$$\sum_{x \in \Upsilon_{\mathsf{A}_\mathsf{a}}} x = \mathsf{A}_\mathsf{a}\,.$$

From our construction of s_w above and Theorem 4.3, Q cannot be violated.

Case 3. Q is of the form $x \leq \mathsf{C}_{k_{\mathsf{C}_k}}$. From our construction of s_w above and Theorem 4.3, Q cannot be violated.

Cases 1 to 3 cover every type of constraint in $L(\mathcal{B})$. Contradiction. □

THEOREM 4.7. *If s is a 0-1 solution for $L_e(\mathcal{B})$, then w_s is an explanation for e.*

Proof. Assume there exists an instantiation (A, a) in e but not in w_s. Since w_s is a complete instantiation-set, there exists the instantiation (A, a') in w_s where $a \neq a'$. From our construction of w_s and Theorems 4.4 and 4.5, $\mathsf{A}_{\mathsf{a}'} = 1$. From Theorem 4.2. $\mathsf{A}_\mathsf{a} = 0$. However, Definition 4.8, $\mathsf{A}_\mathsf{a} = 1$. Contradiction. □

THEOREM 4.8. *If w is an explanation for e, then s_w is a 0-1 solution for $L_e(\mathcal{B})$.*

Proof. Assume s_w is not a 0-1 solution for $L_e(\mathcal{B})$. The only difference between $L_e(\mathcal{B})$ and $L(B)$ is in the addition of the constraints of the form $\mathsf{A_a} = 1$ where $(A, a) \in e$. If s_w is not a 0-1 solution for $L_e(\mathcal{B})$, then s_w must violate one of these constraints. Let $\mathsf{A_a} = 1$ be the constraint violated. This implies that $\mathsf{A_a} = 0$. However, this contradicts the construction of s_w. □

THEOREM 4.9. $\Theta_L(s_r) = -\log(P(w))$.

Proof.

$$\begin{aligned} -\log(P(w)) &= -\log\left\{\prod_{A\in V} P(A = w(a)|w(\mathsf{cond}(a)))\right\} \\ &= \sum_{A\in V} -\log(PA = w(A))|w(\mathsf{cond}(A))) \end{aligned}.$$

$$\Theta_L(s_w) = \sum_{\substack{\mathsf{A_a} \in \Delta \\ s_w(\mathsf{A_a}) = 1}} \sum_{x\in\Upsilon_{\mathsf{A_a}}} x\psi(x, \mathsf{true}).$$

From Theorem 4.3, for each $\mathsf{A_a} = 1$, only one conditional variable in $\Upsilon_{\mathsf{A_a}}$ has value 1 and $q[\mathsf{A_a} \mid \mathsf{C_1}_{\mathsf{c_1}}, \ldots, \mathsf{C}_{n\mathsf{c}_n}]$ is one such conditional variable iff $\mathsf{A_a} = \mathsf{C_1}_{\mathsf{c_1}} = \ldots = \mathsf{C}_{n\mathsf{c}_n} = 1$. From our construction of $L(\mathcal{B})$, cost is incurred only by conditional variables with values assigned 1. Since $q[\mathsf{A_a} \mid \mathsf{C_1}_{\mathsf{c_1}}, \ldots, \mathsf{C}_{n\mathsf{c}_n}]$ has cost $-\log(P(A = a|C_1 = c_1, \ldots, C_n = c_n))$, we can easily see that $\Theta_L(s_w) = -\log(P(w))$. □

THEOREM 4.10. *There exists a constant α_e such that for all explanations w for e, $\Theta_{L_e}(s_w) = \alpha_e - \log(P(w|e))$.*

Proof.

$$P(w|r) = \frac{P(w)P(e|w)}{P(e)}.$$

If w is consistent to e, then $P(e|w) = 1$, otherwise, $P(e|w) = 0$. Thus, all we need to consider is $P(e)$ which is constant.

$$P(w|e) = \frac{P(w)}{P(e)}$$

$$P(w) = P(e)P(W|e)$$

$$-\log(P(w)) = -\log(P(e)) - \log(P(w|e))$$

Since $\Theta_{L_e}(s_w) = \Theta_L(s_w)$, from Theorem 4.9, we get $\Theta_{L_e}(s_w) = -\log(P(w))$. Thus, $\alpha_e = -\log(P(e))$. □

THEOREM 4.11. *w is a most-probable explanation for e iff s_w is an optimal 0-1 solution for $L_e(B)$.*

Proof. Follows from Theorem 4.10. (□

THEOREM 4.12. *Constraint system L_n in Algorithm 4.1 determines the n-th best 0-1 solution for L.*

Proof. We can show that the additional constraint eliminates exactly one additional 0-1 solution from the set of 0-1 solutions in L. Also, the 0-1 solution eliminated will be the optimal 0-1 solution determined by L_{n-1}. □

THEOREM 4.13. *Given a 0-1 solution s for $\hat{L}(\mathcal{B})$, for each set of variables $\Delta(A)$, there exists at most one variable $\mathsf{A_a}$ in $\Delta(A)$ such that $\mathsf{A_a} = 1$ and $\mathsf{A_{a'}} = 0$ for all $\mathsf{A_{a'}} \neq \mathsf{A_a}$ in $\Delta(A)$.*

Proof. Assume there exists a set of variables $\Delta(A)$ such that $\mathsf{A_a} = \mathsf{A}_{sfa'} = 1$ for some $a \neq a'$. This implies that

$$\sum_{\mathsf{A_a} \in \Delta(A)} \mathsf{A_a} \geq 2$$

which violates the constraint formed by (16) for r.v. A. Contradiction. □

THEOREM 4.14. *Given a 0-1 solution s for $\hat{L}(\mathcal{B})$, for all variables*

$$q[\mathsf{A_a} \mid \mathsf{C}_{1_{\mathsf{C}_1}}, \ldots, \mathsf{C}_{n_{\mathsf{C}_n}}],$$

$$q[\mathsf{A_a} \mid \mathsf{C}_{1_{\mathsf{C}_1}}, \ldots, \mathsf{C}_{n_{\mathsf{C}_n}}] = 1 \textit{ iff } \mathsf{A_a} = \mathsf{C}_{1_{\mathsf{C}_1}} = \ldots = \mathsf{C}_{n_{\mathsf{C}_n}} = 1.$$

Proof. Let s be a 0-1 solution for $\hat{L}(\mathcal{B})$. (*if*) Assume $q[\mathsf{A_a} \mid \mathsf{C}_{1_{\mathsf{C}_1}} \ldots, \mathsf{C}_{n_{\mathsf{C}_n}}] = 1$. From (13), this implies $\mathsf{C}_{j_{\mathsf{C}_j}} = 1$. From (14),

$$\sum_{x \in \Upsilon_{\mathsf{A_a}}} = \mathsf{A_a} = 1$$

since $q[[\mathsf{A_a} \mid \mathsf{C}_{1_{\mathsf{C}_1}} \ldots, \mathsf{C}_{n_{\mathsf{C}_n}}] = 1 \in \Upsilon_{\mathsf{A_a}}$.

(*only if*) Assume $\mathsf{A}_{\mathsf{a}} = \mathsf{C_1}_{\mathsf{c}_1} = \ldots = \mathsf{C}_{n_{\mathsf{c}_n}} = 1$. From (14),

$$\sum_{x \in \Upsilon_{\mathsf{A}_{\mathsf{a}}}} = 1.$$

Let x be the conditional variable $q[\mathsf{A}_{\mathsf{a}} \mid \mathsf{C_1}_{\mathsf{d}_1}, \ldots, \mathsf{C}_{n_{\mathsf{d}_n}}] = 1$. Assume $x = 1$ and for some l, $\mathsf{d}_l \neq \mathsf{c}_l$. From (13), this implies $\mathsf{C}_{l_{\mathsf{d}_l}} = 1$. Since $\mathsf{C}_{l_{\mathsf{c}_l}} = 1$, r.v. C_l has two distinct instantiations which violates (16). Contradiction. Therefore, for x to be 1, $\mathsf{d}_l = \mathsf{c}_l$ for $l = 1, \ldots, n$. $\square$

THEOREM 4.19. *Given a Bayesian network $\mathcal{B} = (V, P)$, let v be a subset of V and q be an instantiation-set for $\mathcal{B}$ whose span is $V - v$, then*

$$\sum_{w \in \kappa(v)} U \prod_{A \in \mathsf{span}(w)} P(w(A)|J(w, Jq)(\mathsf{cond}(A))) = 1.$$

Proof. First, arbitrarily order the r.v.s in v resulting in the sequence of r.v.s, $A_1, A_2, \ldots, A_{|v|}$. We can now rewrite the sum as

$$\sum_{w \in \kappa(v)} \prod_{i=1}^{|v|} P(w(A_i))|J(w, Jq)(\mathsf{cond}(A_i)))$$

$$= e \sum_{w \in \kappa(v)} P(w(A_1))|J(w, Jq)(\mathsf{cond}(A_1)) \prod_{i=2}^{|v|} P(w(A_i))|J(w, Jq)(\mathsf{cond}(A_i))).$$

We partition $\kappa(v)$ with respect to A_1 as follows: instantiation-sets w, w' in $\kappa(v)$ belong in the same partition iff $(w \cup w') - (w \cap w') = \{(A_1, a_1), (A_1, a_1')\}$ where $a_1 \neq a_1'$. Let $v_1 = v - \{A_1\}$. For each d in $\kappa(v_1)$, $\vartheta_1(v, d)$ denotes the partition where each instantiation-set in $\kappa(v)$ is consistent with d. We can now rewrite our sum as

$$\sum_{d \in \kappa(v_1)} \sum_{w \in \vartheta_1(v, Jd)} P(w(A_1))|J(w, Jq)(\mathsf{cond}(A_1))U \prod_{i=2}^{|v|} P(w(A_i))|J(w, Jq)(\mathsf{cond}(A_i)))$$

Since all the instantiation-sets w in the product term are identical. This is constant and can be factored out of the inner sum.

$$= e \sum_{d \in \kappa(v_1)} \left\{ \prod_{i=2}^{|v|} P(d(A_i))|J(d, Jq)(\mathsf{cond}(A_i))) * \right.$$

$$\left. \sum_{w \in \vartheta_1(v, Jd)} P(w(A_1))|J(w, Jq)(\mathsf{cond}(A_1)) \right\}.$$

Note that the assignments to the conditionals of the probability in the innermost sum are all identical. Thus, the innermost sum is equal to 1. This leaves us with

$$= e \sum_{d \in \kappa(v_1)} \prod_{i=2}^{|v|} P(d(A_i)) | J(d, Jq)(\mathsf{cond}(A_i))).$$

We can continue and process A_2 in the same fashion. Eventually, we will reduce our sum to 1. $\square$

THEOREM 4.20. *If w is a well-founded instantiation-set for $\mathcal{B}$, then*

$$P(w) = U \prod_{A \in \mathsf{span}(w)} P(w(A) | w(\mathsf{cond}(A))).$$

Proof. By Proposition 4.15,

$$P(w) = \sum_{w' \in X(w)} P(J(w, Jw')).$$

From Proposition 4.18, we get

$$P(w) = \sum_{w' \in X(w)} \left\{ \prod_{A \in \mathsf{span}(w)} P(w(A)) | w(\mathsf{cond}(A))) * \prod_{A \in \mathsf{span}(w')} P(w'(A)) | J(w, Jw')(\mathsf{cond}(A))) \right\}.$$

By factoring out the common terms, we get

$$P(w) = \prod_{A \in \mathsf{span}(w)} P(w(A)) | w(\mathsf{cond}(A))) * \sum_{w' \in X(w)} \prod_{A \in \mathsf{span}(w')} P(w'(A)) | J(w, Jw')(\mathsf{cond}(A))).$$

By Theorem 4.19, the sum is equal to 1. $\square$

THEOREM 4.21. *If s is a 0-1 solution for $\hat{L}(\mathcal{B})$, then w_s is a well-founded instantiation-set for $\mathcal{B}$.*

Proof. From the above construction and Theorems 4.13 and 4.14, w_s is an instantiation-set for $\mathcal{B}$. Assume w_s is not well-founded. This implies that there exists a r.v. $A \in \mathsf{span}(w_s)$ such that r.v. C is not in $\mathsf{span}(w_s)$ for some $C \in$

$\mathsf{cond}(A)$. From the above construction, $\mathsf{A_a} = 1$ corresponding to some $A = a$ in w_s and $\mathsf{C_c} = 0$ for all $\mathsf{C_c} \in \Delta(C)$ since C is not in $\mathsf{span}(w_s)$. Furthermore, from the construction of weakly induced constraint systems, we have the following constraints:

$$q[\mathsf{A_a} \mid \ldots, \mathsf{C_c}, \ldots] \leq \mathsf{C_c} \text{ for all } \mathsf{C_c} \in \Delta(C) \text{ and} \tag{49}$$

$$\sum_{x \in \Upsilon_{\mathsf{A_a}}} x = \mathsf{A_a}\,. \tag{50}$$

Since $\mathsf{A_a} = 1$, (50) implies that there exists some $\mathsf{C_c} \in \Delta(C)$ such that

$$q[\mathsf{A_a} \mid \ldots, \mathsf{C_c}, \ldots] = 1.$$

(49) further implies that $\mathsf{C_c} = 1$. Contradiction. □

THEOREM 4.22. *If w is a well-founded instantiation-set for $\mathcal{B}$, then s_w is a 0-1 solution for $\hat{L}(\mathcal{B})$.*

Proof. Let w be a well-founded instantiation-set for $\mathcal{B}$. Assume s_w is not a 0-1 solution for $\hat{L}(\mathcal{B}) = (\Gamma, I, \psi)$. This implies that there exists some constraint Q in I which is violated by s_w.

Case 1. Q is of the form

$$\sum_{\mathsf{A_a} \in \Delta(A)} \mathsf{A_a} \leq 1.$$

Since s_w is a 0-1 assignment, then from Theorem 4.13, there exists $\mathsf{A_a} = \mathsf{A_{a'}} = 1$ for some $\mathsf{a} \neq \mathsf{a'}$. This implies that the r.v. A has more that one instantiation which contradicts the fact that w is an instantiation-set for $\mathcal{B}$. Thus, Q cannot be violated.

Case 2. Q is of the form

$$\sum_{x \in \Upsilon_{\mathsf{A_a}}} x = \mathsf{A_a}\,.$$

From our construction of s_w above and Theorem 4.14, Q cannot be violated.

Case 3. Q is of the form $x \leq \mathsf{C}_{k_{\mathsf{C}_k}}$. From our construction of s_w above and Theorem 4.14, Q cannot be violated.

Cases 1 to 3 cover every type of constraint in $\hat{L}(\mathcal{B})$. Contradiction. □

THEOREM 4.23.

$$\Theta_L(s_w) = -\log(P(w)).$$

Proof. Similar to Theorem 4.9. □

THEOREM 4.24. *If s is a 0-1 solution for $\hat{L}_e(\mathcal{B})$, then w_s is a hypothesis for e.*

Proof. Assume there exists a r.v. A such that $(A, a) \in e$ and $(A, a') \in w_s$ and $a \neq a'$. From our construction of w_s and Theorems 4.21 and 4.22, $\mathsf{A}_{\mathsf{a}'} = 1$. From Theorem 4.13, $\mathsf{A}_{\mathsf{a}} = 0$. However, from Definition 4.13, $\mathsf{A}_{\mathsf{a}} = 1$. Contradiction. □

THEOREM 4.25. *If w is a hypothesis for e, then s_w is a 0-1 solution for $\hat{L}_e(\mathcal{B})$.*

Proof. Similar to Theorem 4.8. □

THEOREM 4.26. *There exists a constant α_e such that for all well-founded instantiation-sets w consistent to e, $\Theta_{L_e}(s_w) = \alpha_e - \log(P(w|e))$.*

Proof.

$$P(r|e) = P(r)P(e|r)P(e).$$

If w is a hypothesis for e, then $P(e|w) = 1$. Otherwise, $P(e|w) = 0$. Thus, all we need to consider is $P(e)$ which in constant.

$$\begin{aligned} P(w|e) &= P(w)P(e) \\ P(w) &= P(e)P(w|e) \\ -\log(P(w)) &= -\log(P(e)) - \log(P(w|e)) \end{aligned}$$

Since $\Theta_{L_e}(s_w) = \Theta_L(s_w)$, from Theorem 4.23, we get

$$\Theta_{L_e}(s_w) = -\log(P(w))$$

Thus, $\alpha_e = -\log(P(e))$. □

THEOREM 4.27. *w is a maximal well-founded instantiation-set for $\mathcal{B}$ with evidence e iff s_w is an optimal 0-1 solution for $\hat{L}_e(\mathcal{B})$.*

Proof. Follows from Theorem 4.26. □

THEOREM 4.30. *If $I_j \neq I_k$ for all $j \neq k$, then there exists some integer N such that $\hat{Bel}_N(A = a) = Bel(A = a)$.*

Proof. Follows from Propositions 4.28 and 4.29. □

THEOREM 4.32. *Given a Bayesian network $\mathcal{B} = (V, P)$ and evidence e, an I-sequence of e which instantiation-sets are sorted in order of decreasing probabilities is mass-fastest.*

Proof. Follows from Definitions 4.16, 4.17, 4.18 and Proposition 4.31. □

THEOREM 4.33. *Given a well founded instantiation set w, $P(w) \equiv P_s(w)$.*

Proof. Follows from Theorem 4.20 and Definition 4.23. □

THEOREM 4.34. *Any Bayesian network can be modeled as a splined Bayesian network.*

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

THEOREM 4.35. *Given any cell $d = \{a_1, \ldots, a_k\}$ in $[A]_{\overline{\mathcal{B}}}$ where $E_A(a_i) < E_A(a_{i+1})$, there does not exist $b \in R(A)$ such that $E_A(a_1) < E_A(b) < E_A(a_k)$ and $b \neq a_i$ for $i = 1, \ldots, k$.*

Proof. This follows from the fact that the partitioning induced on A results from a contiguous partitioning of the conditional probabilities P and from Definition 4.20. □

THEOREM 4.36. *w is a well-founded instantiation-set for $\mathcal{B}$ if and only $s[w]$ is a permissible solution for the induced constraint system.*

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

1. $y_A + \sum_{i=1}^{n} x_{d_{A,i}} = 1$
2. $x_A - \min_{a \in d_{A,i}} E_A(a) \geq -M(1 - x_{d_{A,i}})$
3. $x_A - \max_{a \in d_{A,i}} E_A(a) \leq M(1 - x_{d_{A,i}})$
4. $x_A \leq M(1 - y_A)$
5. $y_A \geq |\mathsf{cond}(A)| + 1 - \sum_{B \in \mathsf{cond}(A)} y_B$

6. $x_{C_i} - \min(C_i, S) \geq -M(1 - d_{C_i,S})$
7. $x_{C_i} - \max(C_i, S) \leq M(1 - d_{C_i,S})$
8. $d_S \geq n - \sum_{j=1}^{n} d_{C_j,S} + 1$
9. $d_S \leq \frac{1}{n} \sum_{j=1}^{n} d_{C_j,S}$
10. $x_{C_i,S} \geq x_{C_i} - M(1 - d_S)$
11. $x_{C_i,S} \leq x_{C_i} + M(1 - d_S)$
12. $x_{C_i,S} \leq M d_S$

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

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

  From our construction, if $s(y_A) = 1$, then A has not been instantiated in w. If $s(y_A) = 0$, then A has been instantiated and for some i, $s(x_{d_{A,i}}) = 1$. Contradiction.
- CASE 2. $x_A - \min_{a \in d_{A,i}} E_A(a) \geq -M(1 - x_{d_{A,i}})$ has been violated. This implies that $s(x_A)$ is in the designated interval but $s(x_{d_{A,i}}) = 0$. This is a contradiction according to our construction.
- CASES 3, 4, 6 AND 7. The reasoning is similar to CASE 2.
- CASE 5. $y_A \leq \frac{1}{|\mathsf{cond}(A)|} \sum_{B \in \mathsf{cond}(A)} y_B$ is violated. This implies that $s(y_A) = 1$ and there exists a B in $\mathsf{cond}(A)$ such that $s(y_B) = 0$. This says that A is instantiated but B is not instantiated which violates the definition of well-foundedness for w. Contradiction.
- CASE 8. $d_S \geq n - \sum_{j=1}^{n} d_{C_j,S} + 1$ is violated. This implies that $s(d_S) = 0$ and $s(d_{C_j,S}) = 1$ for all j. However, this violates our notion of an active splining function from our construction. Contradiction.
- CASE 9. $d_S \leq \frac{1}{n} \sum_{j=1}^{n} d_{C_j,S}$ is violated. This implies that $s(d_S) = 1$ and $s(d_{C_j,S}) = 0$ for all j. However, this violates our notion of an active splining function from our construction. Contradiction.
- CASE 10. $x_{C_i,S} \geq x_{C_i} - M(1 - d_S)$ is violated. This implies that $s(d_S) = 1$ and $s(x_{C_i,S}) < s(x_{C_i})$. But from our construction, when a splining function is active, $s(x_{C_i,S}) = s(x_{C_i})$. Contradiction.
- CASES 11 AND 12. Similar to CASE 10.

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

Now, given a permissible solution s. Assume that $w[s]$ is not a well-founded instantiation-set. This implies that there exists a r.v. A in V such that A is instantiated in $w[s]$ and for some B in $\mathsf{cond}(A)$, B is not instantiated in $w[s]$. Let A be such a r.v.. From our construction, this implies that $s(x_A) > 0$ and $s(x_B) = 0$. Furthermore, from constraints (22), (23), (24) and (25), $s(y_A) = 0$ and $s(y_B) = 1$. However, this violates constraint (26). Contradiction. □

THEOREM 4.37.

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

Proof. Follows from our construction and Theorems 4.20 and 4.36. □

THEOREM 5.1. *$G_s = (V_s, E_s)$ is a solution graph for $Z = (G, c, r, S)$ where $G = (V, E)$ iff s is a 0-1 solution for $L(Z)$.*

Proof. Formally, we must first prove that given any solution graph $G_s = (V_s, E_s)$ for $Z = (G, c, r, S)$ where $G = (V, E)$ we can construct a 0-1 solution s for $L(Z) = (\Gamma, I, \psi)$ such that $p \in V_s$ if and only if $s(x_p) = 1$.

Assume s is not a 0-1 solution for $L(Z)$. This implies at least one of the following constraint forms have been violated by s:

1. $x = b$ where $x \in \Gamma$ and $b \in \{0, 1\}$.
2. $x \leq y$ where $x, y \in \Gamma$.
3. $x \leq y_1 + y_2 + \ldots + y_n$ where $x, y_1, \ldots, y_n \in \Gamma$.
4. $x \geq y_1 + y_2 + \ldots + y_n - n + 1$ where $x, y_1, \ldots, y_n \in \Gamma$.
5. $x_1 + x_2 + \ldots + x_n \leq n - 1$ where $x_1, \ldots, x_n \in \Gamma$.

- CASE 1: $x = b$ has been violated. Without loss of generality, assume that $b = 1$. This implies that $s(x) = 0$. Since this constraint is only generated by evidence in S, this implies that there is some node p in V such that $x \equiv x_p$ where x_p is the real variable associated with p in $L(Z)$. Now, $\{x_p = 1\} \in I \Rightarrow (p, \mathsf{true}) \in S \Rightarrow p \in V_s \Rightarrow s(x_p) = 1$. However, $x \equiv x_p$. Contradiction.
- CASE 2: $x \leq y$ has been violated. This implies that $s(x) = 1$ and $s(y) = 0$. This constraint can be generated from one of the following sources:

- From equation (36), we see that x and y are the real variables associated with some nodes q and p in V, respectively. Furthermore, $r(q) = \text{AND}$ and $(p, q) \in E$. Now, $s(x) = 1$ and $s(y) = 0$ imply $q = \mathsf{true}$ and $p = \mathsf{false}$ which further implies that $q \in V_s$ but p is not in V_s. However, this violates Definition 5.2 thus implying that G_s is not a solution graph. Contradiction.
- From equation (39), we see that x and y are the real variables associated with some nodes p and q in V, respectively. Furthermore, $r(q) = \text{OR}$ and $(p, q) \in E$. Now, $s(x) = 1$ and $s(y) = 0$ imply $p = \mathsf{true}$ and $q = \mathsf{false}$ which further implies that $p \in V_s$ but q is not in V_s. However, this violates Definition 5.2 thus implying that G_s is not a solution graph. Contradiction.
- From equation (40), we find that y is the real variable associated with some node p_i in V and x is a special marker variable created between node p_i and some node p_{i+1} in V and $x \equiv m_{p_i p_{i+1}}$. Now, $s(x) = 1$ and $s(y) = 0$ imply that $(p_i, p_{i+1}) \in E_s$ and $p_i = \mathsf{false}$. However, $(p_i, p_{i+1}) \in E_s$ implies that $p_i \in V_s$. Contradiction.

- Case 3: $x \leq y_1 + \ldots + y_n$ has been violated. This implies that $s(x) = 1$ and $s(y_i) = 0$ for $i = 1, \ldots, n$. We can easily see that this type of constraint is generated by the equation (38). We know that x is a real variable associated with some node p in V and the $r(p) = \text{OR}$. Furthermore, $s(x) = 1 \Rightarrow p \in V_s$. Some of the y_i's are each associated to some special marker variable $m_{q_i p}$. Now, $s(y_i) = 0 \Rightarrow s(m_{q_i p}) = 0$ which further implies that (q_i, p) is not in E_s. For the remaining y_i's which are associated to some node q_i in V, $s(y_i) = 0$ implies that q_i is not in V_s. This also implies that (q_i, p) is not in E_s. However, this violates Definition 5.2. Thus, G_s is not a solution graph. Contradiction.
- Case 4: $x \geq y_1 + \ldots + y_n - n + 1$ has been violated. This implies that $s(x) = 0$ and $s(y_i) = 1$ for all $i = 1, \ldots, n$. Since this constraint can only be generated by equation (37), x is the real variable associated with some node p in V, $r(p) = \text{AND}$ and y_i is associated with some q_i in V for all i. This implies that $q_i \in V_s$ for all i but p is not in V_s which violates Definition 5.2.

Thus, G_s is not a solution graph. Contradiction.

- CASE 5: $x_1 + \ldots + x_n \leq n - 1$ has been violated. This implies that $s(x_i) = 1$ for all $i = 1, \ldots, n$. Finally, this constraint is created by equation (41). It is directly associated with some cycle $\{p_1, \ldots, p_{n-1}\}$ in G. Now, for $i = 1, \ldots, n-1$ each x_i either represents the AND-node p_{i+1} or the special marker variable $m_{p_i p_{i+1}}$. If x_i is associated to the AND-node p_{i+1}, then $p_{i+1} \in V_s$ and according to Definition 5.2, $(p_i, p_{i+1}) \in E_s$. If x_i is associated to $m_{p_i p_{i+1}}$, then $s(x_i) = 1$ implies that $(p_i, p_{i+1}) \in E_s$. Finally, x_n is a special case joining p_{n-1} and p_1 which will imply that $(p_{n-1}, p_1) \in E_s$. However, this implies that G_s has a cycle. Contradiction.

All the cases have now been covered. Thus s is a 0-1 solution for $L(Z)$.

Now, we must finish our proof by showing that for each 0-1 solution s for $L(Z)$, we can construct a solution graph G_s for Z such that $s(x_p) = 1$ iff $p \in V_s$.

Assume that $G_s = (V_s, E_s)$ is not a solution graph for Z. This implies that one or more of the following conditions have been violated:

1. If $(q, \mathsf{true}) \in S$, then $q \in V_s$.
2. If $(q, \mathsf{false}) \in S$, then q is not in V_s.
3. If $q \in V_s$ and $r(q) = $ AND, then $D_q \subseteq V_s$ and $(p, q) \in E_s$ for all $p \in D_q$.
4. If $D_q \subseteq V_s$ and $r(q) = $ AND, then $q \in V_s$ and $(p, q) \in E_s$ for all $p \in D_q$.
5. If $q \in V_s$ and $r(q) = $ OR, then there exists a node p in D_q such that $p \in V_s$ and $(p, q) \in E_s$.
6. If $D_q \cap V_s \neq \phi$ and $r(q) = $ OR, then $q \in V_s$ and there exists a node $p \in D_q \cap V_s$ such that $(p, q) \in E_s$.
7. G_s is acyclic.

- CASE 1: If $(q, \mathsf{true}) \in S$, then $q \in V_s$.
  q not in V_s implies that $s(x_q) = 0$. However, according to our construction of $L(Z)$, if $(q, \mathsf{true}) \in S$, then $s(x_q) = 1$. Contradiction.
- CASE 2: If $(q, \mathsf{false}) \in S$, then q is not in V_s.
  $q \in V_s$ implies that $s(x_q) = 1$. However, according to our construction of $L(Z)$, if $(q, \mathsf{false}) \in S$, then $s(x_q) = 0$. Contradiction.

- CASE 3: If $q \in V_s$ and $r(q) = \text{AND}$, then $D_q \subseteq V_s$ and $(p, q) \in E_s$ for all $p \in D_q$.
  This implies that either $D_q - V_s \neq \phi$ or there exists some $p \in D_q$ such that (p, q) is not in E_s. If $D_q - V_s \neq \phi$, then let $p \in D_q - V_s$. Now, $s(x_p) = 0$ and $s(x_q) = 1$. However, since $r(q) = \text{AND}$, we have the constraint $x_q \leq x_p$ in $L(Z)$. Contradiction.
  D_q must be a subset of V_s. This implies that from our construction of G_s, for all $p \in D_q$, $(p, q) \in E_s$. Contradiction.
- CASE 4: If $D_q \subseteq V_s$ and $r(q) = \text{AND}$, then $q \in V_s$ and $(p, q) \in E_s$ for all $p \in D_q$.
  This implies that either q is not in V_s or there exists some $p \in D_q$ such that (p, q) is not in E_s. If q is not in V_s, then $s(x_q) = 0$. Also, $s(x_p) = 1$ for all $p \in D_q$. However, since $r(q) = \text{AND}$, we have the constraint $x_q \geq \sum_{p \in D_q} x_p - |D_q| + 1$. Contradiction.
- CASE 5: If $q \in V_s$ and $r(q) = \text{OR}$, then there exists a node p in D_q such that $p \in V_s$ and $(p, q) \in E_s$.
  This implies that for all $p \in D_q$, either p is not in V_s or (p, q) is not in E_s. We know that $s(x_q) = 1$ and $r(q) = \text{OR}$. This implies that we have a constraint of the form $x_q \leq y_1 + \ldots + y_n$ where each y_i represents one of the following:
  - y_i is the real variable associated with the node p_i in V.
  - y_i is the special marker variable associated with the nodes p_i and q in V and is called $m_{p_i q}$.

  Note, $D_q = \{p_1, \ldots, p_n\}$ where p_i is related to y_i.
  According to our constraint, there exists some $s(y_i) = 1$. If $y_i \equiv x_{p_i}$, then $p_i \in V_s$ and according to our construction of G_s, $(p_i, q) \in E_s$, since there is no special marker associated to p_i and q. Contradiction. If $y_i \equiv m_{p_i q}$, then we also have the constraint that $m_{p_i q} \leq x_{p_i}$ in $L(Z)$. This implies that $s(x_{p_i}) = 1$ and according to our construction of G_s, $(p_i, q) \in E_s$. Contradiction.
- CASE 6: If $D_q \cap V_s \neq \phi$ and $r(q) = \text{OR}$, then $q \in V_s$ and there exists a node $p \in D_q \cap V_s$ such that $(p, q) \in E_s$.

This implies that either q is not in V_s or for all $p \in D_q \cap V_s$, (p, q) is not in E_s. We know that for all $p \in D_q$, $s(x_p) = 1$ and $r(q) = \text{OR}$. From our construction of $L(Z)$, we have the constraints $x_p \leq x_q$ for all $p \in D_q$. Since $D_q \cap V_s \neq \phi$, $s(x_q) = 1$ which implies that $q \in V_s$. Thus, this leaves us with for all $p \in D_q \cap V_s$, (p, q) is not in E_s. Now, for each $p \in D_q \cap V_s$, if there is no m_{pq}, then $(p, q) \in E_s$. If there is an m_{pq}, then m_{pq} must be 0. Looking at the constraint for OR-node q, $x_q \leq y_1 + \ldots + y_n$ where y_i is either associated with a node in D_q or a special marker variable, we find that all the y_i's must be set to 0. Contradiction since $s(x_q) = 1$.

- CASE 7: G_s is acyclic.

  This implies that there exists a cycle $\{p_1, \ldots, p_n\}$ of nodes in G_s. Without loss of generality, assume that all the nodes are OR-nodes. A cycle in G_s is of course a cycle in G. By our cyclicity construction, we have special marker variables $m_{p_i p_{i+1}}$ between the adjacent nodes in the cycle. The cycle implies that $(p_i, p_{i+1}) \in E_s$ for all i including the edge (p_n, p_1). By our construction of G_s, $s(m_{p_i p_{i+1}}) = 1$ for all i. However, we have the constraint

$$m_{p_1 p_2} + \ldots + m_{p_{n-1} p_n} + m_{p_n p_1} \leq n - 1.$$

  Contradiction.

Contradiction. G_s is a solution graph for Z. □

THEOREM 5.2. *G_s is a best explanation iff s is an optimal 0-1 solution.*

Proof. Follows from Theorem 5.1 and the fact that our costs computations are identical. □

THEOREM 5.3. *$G_s = (V_s, E_s)$ is a solution graph for $Z = (G, c, r, S)$ where $G = (V, E)$ iff s is a 0-1 solution for $L(Z)$.*

Proof. Formally, we must first prove that given any solution graph $G_s = (V_s, E_s)$ for $Z = (G, c, r, S)$ where $G = (V, E)$ we can construct a 0-1 solution s for $L(Z) = (\Gamma, I, \psi)$ such that $p \in V_s$ if and only if $s(x_p) = 1$.

Assume s is not a 0-1 solution for $L(Z)$. This implies at least one of the following constraint forms have been violated by s:

1. $x = b$ where $x \in \Gamma$ and $b \in \{0, 1\}$.

2. $x \le y$ where $x, y \in \Gamma$.
3. $x \le m_1 + m_2 + \ldots + m_n$ where $x, m_1, \ldots, m_n \in \Gamma$.
4. $x \ge y_1 + y_2 + \ldots + y_n - n + 1$ where $x, y_1, \ldots, y_n \in \Gamma$.
5. $2|V|(1 - m_{pq}) + t_q \ge t_p + 1$.

- CASE 1: $x = b$ has been violated. Without loss of generality, assume that $b = 1$. This implies that $s(x) = 0$. Since this constraint is only generated by evidence in S, this implies that there is some node p in V such that $x \equiv x_p$ where x_p is the real variable associated with p in $L(Z)$. Now, $\{x_p = 1\} \in I \Rightarrow (p, \mathsf{true}) \in S \Rightarrow p \in V_s \Rightarrow s(x_p) = 1$. However, $x \equiv x_p$. Contradiction.
- CASE 2: $x \le y$ has been violated. This implies that $s(x) = 1$ and $s(y) = 0$. This constraint can be generated from one of the following sources:
  - From equation (42), we see that x and y are the real variables associated with some nodes q and p in V, respectively. Furthermore, $r(q) = $ AND and $(p, q) \in E$. Now, $s(x) = 1$ and $s(y) = 0$ imply $q = \mathsf{true}$ and $p = \mathsf{false}$ which further implies that $q \in V_s$ but p is not in V_s. However, this violates Definition 5.2 thus implying that G_s is not a solution graph. Contradiction.
  - From equation (46), we see that x and y are the real variables associated with some nodes p and q in V, respectively. Furthermore, $r(q) = $ OR and $(p, q) \in E$. Now, $s(x) = 1$ and $s(y) = 0$ imply $p = \mathsf{true}$ and $q = \mathsf{false}$ which further implies that $p \in V_s$ but q is not in V_s. However, this violates Definition 5.2 thus implying that G_s is not a solution graph. Contradiction.
  - From equation (47), we find that y is the real variable associated with some node p_i in V and x is a special marker variable created between node p_i and some node p_{i+1} in V and $x \equiv m_{p_i p_{i+1}}$. Now, $s(x) = 1$ and $s(y) = 0$ imply that $(p_i, p_{i+1}) \in E_s$ and $p_i = \mathsf{false}$. However, $(p_i, p_{i+1}) \in E_s$ implies that $p_i \in V_s$. Contradiction.
- CASE 3: $x \le m_1 + \ldots + m_n$ has been violated. This implies that $s(x) = 1$ and $s(m_i) = 0$ for $i = 1, \ldots, n$. We can easily see that this type of constraint is generated by equation (45). We know that x is a real variable associated

with some node p in V and the $r(p) = \text{OR}$. Furthermore, $s(x) = 1 \Rightarrow p \in V_s$. Now, $s(m_i) = 0 \Rightarrow s(m_{q_i p}) = 0$ which further implies that (q_i, p) is not in E_s. However, this violates Definition 5.2. Thus, G_s is not a solution graph. Contradiction.

- CASE 4: $x \geq y_1 + \ldots + y_n - n + 1$ has been violated. This implies that $s(x) = 0$ and $s(y_i) = 1$ for all $i = 1, \ldots, n$. Since this constraint can only be generated by equation (43), x is the real variable associated with some node p in V, $r(p) = \text{AND}$ and y_i is associated with some q_i in V for all i. This implies that $q_i \in V_s$ for all i but p is not in V_s which violates Definition 5.2. Thus, G_s is not a solution graph. Contradiction.
- CASE 5: $2|V|(1 - m_{pq}) + t_q \geq t_p + 1$ has been violated. If $s(m_{pq}) = 0$, then the constraint has not been violated since $s(t_p)$ and $s(t_q)$ are restricted to $\{1, \ldots, |V|\}$. Thus, $s(t_q) < s(t_p) + 1$ and $s(m_{pq}) = 1$. However, $s(m_{pq}) = 1$ implies that $(p, q) \in E_s$ which further implies that $s(t_q) > s(t_p)$. Contradiction.

All the cases have now been covered. Thus s is a 0-1 solution for $L(Z)$.

Now, we must finish our proof by showing that for each 0-1 solution s for $L(Z)$, we can construct a solution graph G_s for Z such that $s(x_p) = 1$ iff $p \in V_s$.

Assume that $G_s = (V_s, E_s)$ is not a solution graph for Z. This implies that one or more of the following conditions have been violated:

1. If $(q, \mathsf{true}) \in S$, then $q \in V_s$.
2. If $(q, \mathsf{false}) \in S$, then q is not in V_s.
3. If $q \in V_s$ and $r(q) = \text{AND}$, then $D_q \subseteq V_s$ and $(p, q) \in E_s$ for all $p \in D_q$.
4. If $D_q \subseteq V_s$ and $r(q) = \text{AND}$, then $q \in V_s$ and $(p, q) \in E_s$ for all $p \in D_q$.
5. If $q \in V_s$ and $r(q) = \text{OR}$, then there exists a node p in D_q such that $p \in V_s$ and $(p, q) \in E_s$.
6. If $D_q \cap V_s \neq \phi$ and $r(q) = \text{OR}$, then $q \in V_s$ and there exists a node $p \in D_q \cap V_s$ such that $(p, q) \in E_s$.
7. G_s is acyclic.

- CASE 1: If $(q, \mathsf{true}) \in S$, then $q \in V_s$.

q not in V_s implies that $s(x_q) = 0$. However, according to our construction of $L(Z)$, if $(q, \mathsf{true}) \in S$, then $s(x_q) = 1$. Contradiction.

- Case 2: If $(q, \mathsf{false}) \in S$, then q is not in V_s.

  $q \in V_s$ implies that $s(x_q) = 1$. However, according to our construction of $L(Z)$, if $(q, \mathsf{false}) \in S$, then $s(x_q) = 0$. Contradiction.

- Case 3: If $q \in V_s$ and $r(q) = \text{AND}$, then $D_q \subseteq V_s$ and $(p, q) \in E_s$ for all $p \in D_q$.

  This implies that either $D_q - V_s \neq \phi$ or there exists some $p \in D_q$ such that (p, q) is not in E_s. If $D_q - V_s \neq \phi$, then let $p \in D_q - V_s$. Now, $s(x_p) = 0$ and $s(x_q) = 1$. However, since $r(q) = \text{AND}$, we have the constraint $x_q \leq x_p$ in $L(Z)$. Contradiction.

  D_q must be a subset of V_s. This implies that from our construction of G_s, for all $p \in D_q$, $(p, q) \in E_s$. Contradiction.

- Case 4: If $D_q \subseteq V_s$ and $r(q) = \text{AND}$, then $q \in V_s$ and $(p, q) \in E_s$ for all $p \in D_q$.

  This implies that either q is not in V_s or there exists some $p \in D_q$ such that (p, q) is not in E_s. If q is not in V_s, then $s(x_q) = 0$. Also, $s(x_p) = 1$ for all $p \in D_q$. However, since $r(q) = \text{AND}$, we have the constraint $x_q \geq \sum_{p \in D_q} x_p - |D_q| + 1$. Contradiction.

- Case 5: If $q \in V_s$ and $r(q) = \text{OR}$, then there exists a node p in D_q such that $p \in V_s$ and $(p, q) \in E_s$.

  This implies that for all $p \in D_q$, either p is not in V_s or (p, q) is not in E_s. We know that $s(x_q) = 1$ and $r(q) = \text{OR}$. This implies that we have a constraint of the form $x_q \leq m_{p_1 q} + \ldots + m_{p_n q}$. According to our constraint, there exists some $s(m_{p_i q}) = 1$. Now, $p_i \in V_s$ and according to our construction of G_s, $(p_i, q) \in E_s$. Contradiction.

- Case 6: If $D_q \cap V_s \neq \phi$ and $r(q) = \text{OR}$, then $q \in V_s$ and there exists a node $p \in D_q \cap V_s$ such that $(p, q) \in E_s$.

  This implies that either q is not in V_s or for all $p \in D_q \cap V_s$, (p, q) is not in E_s. We know that for all $p \in D_q$, $s(x_p) = 1$ and $r(q) = \text{OR}$. From our construction of $L(Z)$, we have the constraints $x_p \leq x_q$ for all $p \in D_q$. Since $D_q \cap V_s \neq \phi$, $s(x_q) = 1$ which implies that $q \in V_s$. Thus, this leaves us with for all $p \in D_q \cap V_s$, m_{pq} must be 0. Looking at the constraint for OR-node q, $x_q \leq m_1 + \ldots + m_n$, we find that all the m_i's must be set to 0. Contradiction since $s(x_q) = 1$.

- CASE 7: G_s is acyclic.

  This implies that there exists a cycle $\{p_1, \ldots, p_n\}$ of nodes in G_s. Without loss of generality, assume that all the nodes are OR-nodes. A cycle in G_s is of course a cycle in G. By our cyclicity construction, we have special variables $m_{p_i p_{i+1}}$ between the adjacent nodes in the cycle. The cycle implies that $(p_i, p_{i+1}) \in E_s$ for all i including the edge (p_n, p_1). By our construction of G_s, $s(m_{p_i p_{i+1}}) = 1$ for all i. This further implies the following constraints:

  $$2|V|(1 - m_{p_i p_{i+1}}) + t_{p_{i+1}} \geq t_{p_i} + 1$$

  for all i. From transitivity, $t_{p_1} < t_{p_n}$. However, we also have $t_{p_1} > t_{p_m}$. Contradiction.

Contradiction. G_s is a solution graph for Z. □

THEOREM 5.4. *G_s is a best explanation iff s is an optimal 0-1 solution.*

Proof. Follows from Theorem 5.3 and the fact that our costs computations are identical. □