Cost-Based Abduction and Linear Constraint Satisfaction

Eugene Santos Jr.

Department of Computer Science
Brown University
Providence, Rhode Island 02912

Technical Report No. CS-91-13
February 1991

Cost-Based Abduction and Linear Constraint Satisfaction*

Eugene Santos Jr.
Department of Computer Science
P.O. Box 1910, Brown University
Providence, RI 02912

Abstract

Abduction is the problem of finding the best explanation for a given set of observations. Cost-based abduction approaches attempt to prove the observation by assuming some hypotheses. Hypotheses have associated "costs" and the best proof is the one which assumes the least costly set. Previous approaches have formalized minimum cost-based abduction as a search problem. Heuristics were then sought to aid in the search for the best proof. However, efficient admissible heuristics seem to be difficult to find.

In this paper, we present an approach to the problem from the standpoint of optimization. By using linear constraints to represent causal relationships, we are able to provide a new framework capable of modeling cost-based abduction. Moreover, we are able to utilize the highly efficient optimization tools of operations research, thus yielding a computationally efficient method for solving cost-based abduction problems.

* 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.

1. Introduction

Most of the reasoning we employ in our everyday lives tend to be abductive in nature. For example, consider the following situation: "John visits his friend Mary's house and finds that the place is dark. He concludes that Mary is not home." John's conclusion is a form of abductive explanation and cannot be arrived at by purely deductive means.

Abductive explanation has been formalized in AI as the process of searching for some hypotheses that can prove the things to be explained [1, 6, 7, 8, 9, 10]. A basic problem which arises is that there maybe many different sets of hypotheses available for our proof. Some notion of a best explanation is necessary and implies that some measure must be found to order the possibilities.

Early measures based on minimizing the number of necessary hypotheses [6, 9] have been shown to be inadequate [1, 10], suggesting the use of a more sophisticated approach. One such approach involves the weighting of possible sets of hypotheses [1]. These weights are viewed as the "costs" of the proofs. The problem now becomes one of finding the minimal cost proof. The idea has been termed "cost-based abduction" and is presented in [1]. Hypotheses have associated costs, and the cost of a proof is simply the sum of the costs of the hypotheses required to complete our proof. Examples of such proofs can be found in [1, 10]. Central to these approaches is the use of directed acyclic graphs called WAODAGs [1, 10] 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 descendants.

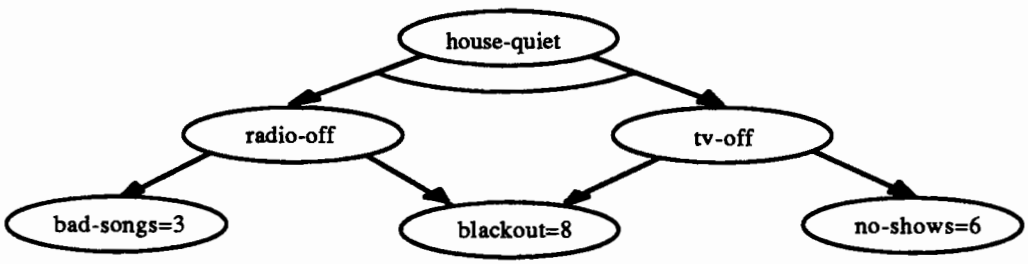

Figure 1.1. A simple WAODAG. The AND-node "house-quiet" is the observation. The nodes "bad-songs", "blackout", and "no-shows" are the hypotheses with associated costs 3, 8, and 6, respectively. The assignment of "blackout" to true and both "bad-songs" and "no-shows" to false results in "radio-off", "tv-off", and "house-quiet" to be true. This proof has a cost of 8 and is the minimal cost proof.

An assignment of a truth value to each node is considered a proof if it is consistent with respect to the boolean network and if the items we wish to explain have been explained, that is, have been assigned a value of true. Consequently, each such proof will have an associated cost. The goal is to find an assignment which has minimal cost (see Figure 1.1).

Current approaches to finding the best proof has been centered around using a best-first search technique and "expanding" partial proofs to search for the best proof [10].

Assorted heuristics maybe found to improve the performance of such techniques but what we ultimately have is an optimization problem.

In this paper, we present an approach that uses linear constraints to represent causal relationships. The highly efficient optimization tools developed in operations research can be used to manipulate these constraints which provide us with a natural way of modeling and solving the minimum cost-based abduction problem. In particular, we show that a WAODAG can be represented by a collection of linear constraints and that the minimal cost proof can be found by optimizing certain elements with respect to the constraints.

Our experimental results indicate that our approach when applied to minimum cost-based abduction seems to be computationally efficient.

In sections 2 and 3, we define the concept of constraint systems and describe the procedure used in solving for minimal cost proofs. In section 4, we present our experimental results on 98 randomly generated directed acyclic AND/OR dags.

2. Constraint Satisfaction

We begin by formalizing the minimum cost-based abduction problem.

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

Definition 2.1. A *WAODAG*[1] 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 x {T, F} to $\Re$, called the *cost function.*
3. r is a function from V to {AND, OR}, called the *label.* A node labeled AND is called an AND-node, etc.
4. S is a subset of nodes in V (the *evidence* nodes).

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

Definition 2.2. A *truth assignment* for a WAODAG W = (G, c, r, S) is a function e from V to {T, F}. We say that such a function is *valid* if the following conditions hold:

1. For all AND-nodes q, e(q) = T if and only if for all nodes p such that (q, p) is an edge in E, e(p) = T.

---

[1] Generalization of Charniak and Shimony [1].

2. For all OR-nodes q, e(q) = T if and only if there exists a node p such that (q, p) is an edge in E and e(p) = T.

Furthermore, we say that e is an *explanation* if and only if e is valid and for each node q in S, e(q) = T.

Definition 2.3. We define the *cost* of an explanation e for W = (G, c, r, S) as

$$C(e) = \sum_{q \in V} c(q, e(q)) \ . \tag{2.1}$$

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

We shall now introduce the notion of a constraint system and show how it could be used to solve the minimum cost-based abduction problem.

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

Definition 2.4. A *constraint system* is a 3-tuple (**V**, I, **c**) where **V** is a finite set of variables, I is a finite set of linear inequalities based on **V**, and **c** is a function from **V** x {T, F} to $\Re$. Given a WAODAG W = (G, c, r, S) where G = (V, E), we can construct a constraint system L(W) = (**V**, I, **c**) where:

1. **V** is a set of variables indexed by V, i.e., $\mathbf{V} = \{x_q \mid q \in V\}$.
2. $\mathbf{c}(x_q, X) = c(q, X)$ for all $q \in V$ and $X \in \{T, F\}$.
3. I is the collection of all inequalities of the forms given below:

a. $$x_q \le x_p \in I \text{ for each } p \in D_q \text{ if } r(q) = \text{AND}. \tag{2.2}$$

b. $$\sum_{p \in D_q} x_p - |D_q| + 1 \le x_q \in I \text{ if } r(q) = \text{AND}. \tag{2.3}$$

c. $$\sum_{p \in D_q} x_p \ge x_q \in I \text{ if } r(q) = \text{OR}. \tag{2.4}$$

d. $$x_q \ge x_p \in I \text{ for each } p \in D_q \text{ if } r(q) = \text{OR}. \tag{2.5}$$

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

e. $$x_q = 1 \text{ if } q \in S, \tag{2.6}$$

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

Definition 2.5. A *variable assignment* for a constraint system L = (**V**, I, **c**) is a function s from **V** 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 is a 0-1 assignment, then s is a *0-1 solution* for L.

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, $s(x_q) = 1$ if and only if e(q) = T.
2. For all q in V, $s(x_q) = 0$ if and only if e(q) = F.

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

Theorem 2.1. If e is an explanation for W, then s[e] is a solution of L(W).

Proof. Assume that s[e] is not a solution of L(W) = (V, I, c). 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,

$$x_p = 1 \text{ and } x_q = 0.$$

From (2.2) and (2.5), we get can conclude that either r(p) = AND or r(q) = OR. If r(p) = AND, then q is a child 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 (2.3), 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 children 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 (2.4), 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. ♦

Corollary 2.2. Let L be constructed from L(W) by eliminating all constraints of the forms (2.3) and (2.5). If e is an explanation for W, then s[e] is a solution of L.[2]

Conditions (2.2) and (2.4) are called *top-down* constraints since they dictate the values of variables from the direction of the evidence nodes. Symmetrically, (2.3) and (2.5) are called *bottom-up* constraints.

---

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

Theorem 2.3. If s is a 0-1 solution of $L_E(W)$, 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 $e(p) = T$ and there exists a $q \in D_p$ such that $e(q) = F$.
2. There exists an AND node p in W such that $e(p) = F$ and for all $q \in D_p$, $e(q) = T$.
3. There exists an OR node p in W such that $e(p) = T$ and for all $q \in D_p$, $e(q) = F$.
4. There exists an OR node p in W such that $e(p) = F$ and there exists a $q \in D_p$ such that $e(q) = T$.
5. There exists an evidence node p in S such that $e(p) = F$.

Case 1. From (2.2), $r(p) = \text{AND}$ implies that the constraint $x_p \le x_q$ is in I. Since, $x_p = 1$ and $x_q = 0$, condition 1 does not hold.

Case 2. From (2.3), $r(p) = \text{AND}$ implies that $x_p \ge 1$. Thus, condition 2 does not hold.

Case 3. From (2.4), $r(p) = \text{OR}$ implies that $x_p \le 0$. Thus, condition 3 does not hold.

Case 4. From (2.5), $r(p) = \text{OR}$ implies that $x_p \ge 1$. Thus, condition 4 does not hold.

Case 5. From (2.6), $p \in S$ implies that $x_p = 1$. Thus, condition 5 does not hold.

Therefore, s is not a 0-1 solution in $L_E(W)$. ♦

From Theorems 2.1 and 2.3, 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.

Definition 2.6. Given a constraint system $L(W) = (\mathbf{V}, \mathbf{I}, \mathbf{c})$ induced (evidentially) by a WAODAG W, we construct a function Θ_L from variable assignments to $\Re$ as follows:

$$\Theta_L(s) = \sum_{x_q \in \mathbf{V}} \{s(x_q)c(x_q, T) + (1 - s(x_q))c(x_q, F)\}. \qquad (2.7)$$

Θ_L is called the *objective function* of L(W).

Definition 2.7. An *optimal 0-1 solution* for a constraint system $L(W) = (\mathbf{V}, \mathbf{I}, \mathbf{c})$ induced (evidentially) by a WAODAG W is a 0-1 solution which minimizes Θ_L.

As we can clearly see, (2.7) is identical to (2.1). From Theorems 2.1 and 2.3 and the relationship between node assignments and variable assignment, 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* [2, 3, 4]. 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[2] and its variations are used

---

[3]For a quick overview of the simplex method, see [5].

to solve linear programs. 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 [3].

Proposition 2.4. Given a WAODAG $W = (G, c, r, S)$, if $L_E(W) = (\mathbf{V}, I, \mathbf{c})$ 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 cannot make restrictions which cannot be modeled by 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 is found. (From our experiments, as we shall see later, such 0-1 solutions will be found a significant number of times just by using linear programming techniques.) 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.[4] Furthermore, our detailed knowledge of the problem structure can be exploited to even further improve performance. The following theorem shows that the number of linear inequalities can be reduced under certain conditions.

Definition 2.8. Given a WAODAG $W = (G, c, r, S)$ where $G = (V, E)$, we can construct a constraint system $\underline{L}(W) = (\mathbf{V}, I, \mathbf{c})$ where:

1. $\mathbf{V}$ is a set of variables indexed by V, i.e., $\mathbf{V} = \{x_q \mid q \in V\}$.
2. $\mathbf{c}(x_q, X) = c(q, X)$ for all $q \in V$ and $X \in \{T, F\}$.
3. I is the collection of all inequalities of the forms given below:

   a. $$x_q \le x_p \in I \text{ if } r(q) = \text{AND}. \tag{2.2}$$

   b. $$\sum_{p \in D_q} x_p \ge x_q \in I \text{ if } r(q) = \text{OR}. \tag{2.4}$$

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

c. $$x_q = 1 \text{ if } q \in S, \tag{2.6}$$

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

---

[4] We can take a set of linear inequalities and simply represent them by $Ax \ge b$ where x is a vector of variables. The simplex method manipulates A towards determining the optimal solution. See [2, 3, 4].

A semi-induced constraint system is simply an induced constraint system lacking bottom-up constraints. From Corollary 2.2, the set of possible solutions for $\underline{L}(W)$ is a superset of the set of possible explanations for W.

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

Definition 2.9. An AND-DAG is a WAODAG whose nodes which are labeled OR have at most outdegree one. Given a WAODAG W = (G, c, r, S), construct W' = (G', c', r', S) from W by removing all but one of the children 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 WAODAG W' is said to be an AND-DAG induced by W.

Proposition 2.5. Let W' be an AND-DAG induced by W. For any truth assignment e, if e(p) = T for all nodes p in W', then e is an explanation for W.

Theorem 2.6. Let W = (G, c, r, S) be a WAODAG where G = (V, E). An optimal 0-1 solution for $\underline{L}_E(W)$ can be transformed into a best explanation for W in O(|E|) steps if and only if c(p, F) ≤ c(p, T) for all nodes p in V.

Proof. Let s be any optimal 0-1 solution for $\underline{L}_E(W)$. 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) = T and there exists a $q \in D_p$ such that e(q) = F.
2. There exists an AND node p in W such that e(p) = F and for all $q \in D_p$, e(q) = T.
3. There exists an OR node p in W such that e(p) = T and for all $q \in D_p$, e(q) = F.
4. There exists an OR node p in W such that e(p) = F and there exists a $q \in D_p$ such that e(q) = T.
5. There exists an evidence node p in S such that e(p) = F.

From Definition 2.8, 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) = F 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 an ancestor also in M.

Let p be any node in M' and D_p^{-1} be the immediate parents of node p. For each q in D_p^{-1}, one of the following holds:

1. e(q) = F and r(q) = AND.
2. e(q) = F and r(q) = OR.
3. e(q) = T and r(q) = OR.

e(q) = T 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 $e(p) = T$.

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 $\underline{L}_E(W)$ and $c(q, F) \leq c(q, T)$ 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 $e(p) = T$, then $c(p, F) = c(p, T)$.) 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. ♦

From the above theorem, a best explanation for W can be found by solving a smaller linear program. In general, transforming a 0-1 optimal solution for $\underline{L}_E(W)$ requires at most $2|E|$ steps.

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 top-down 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.

Proposition 2.7. Given a WAODAG $W = (G, c, r, S)$, if $\underline{L}_T(W) = (\mathbf{V}, \mathbf{I}, \mathbf{c})$ is semi-induced evidentially by W, then $|I| = |\{(p, q) \in E | r(p) = AND\}| + |V_O| + |S|$ 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 above while others are techniques used for general linear programming problems.

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. 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 DAG in Figure 3.1. (Note that we will use constraint systems and linear programs interchangeably throughout this section.)

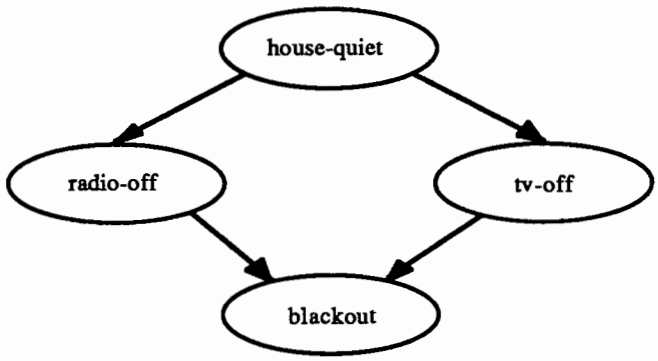

Figure 3.1. In this simple WAODAG, the OR-node "house-quiet" is the observed evidence. "blackout" is the only hypothesis available.

From this WAODAG, the following linear program is generated from our induced constraint system $L = (\mathbf{V}, I, \mathbf{c})$:

$$\begin{gathered} H = 1 \\ R + T \geq H \\ B \geq R \\ B \geq T \\ 0 \leq H, R, T, B \leq 1 \end{gathered}$$

and has objective function: $\Theta_L(s) = s(B)\mathbf{c}(B, T) + (1 - s(B))\mathbf{c}(B, F)$ where H, R, T, and $B \in \mathbf{V}$ respectively stand for "house-quiet", "radio-off", "tv-off", and "blackout". Furthermore, assume $\mathbf{c}(B, F) = 0$.

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 $\Theta_L(s) = \mathbf{c}(B, T)/2$. We call B a *shared* node in our WAODAG. An OR-node such as H with assigned value strictly greater than its children is called a *divide* node. (It is easy to show that either an OR-node is a divide node, or that all of its children are 0 or the same value as the OR-node.)

Certainly, simple patches exist to prevent these types of problems. However, most are non-trivial to identify and repair.

With small linear programs like 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.[5]

The technique to be presented avoids the necessity of searching the entire solution space by utilizing the lower bound computed by the linear program.

The basic idea is as follows: To find an optimal 0-1 solution, we solve a sequence of

---

[5]See integer programming techniques [2, 3, 4].

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 WAODAG. The linear programs along the nodes of the tree are generated using the following schema: Consider $s^{opt}(0)$, the optimal solution to our initial linear program (lp(0)). If $s^{opt}(0)$ is a 0-1 solution, then we are finished. Otherwise, we choose some non-integral variable q in $s^{opt}(0)$ 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 $q = 1$, and (lp(2)) is identical to (lp(0)) except for the additional constraint $q = 0$. Note that the two new problems do not have $s^{opt}(0)$ 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 q 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 [2, 3], information is utilized from other runs which results in quick and efficient calculations. If the optimal solution is not a 0-1 solution, then two new problems are defined based on this linear program. These new problems contains 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 used to prune those active nodes whose computed lower bounds exceed this value. This solution is also kept as the current best solution.

Branching continues in this manner until there are no active nodes in the tree while each 0-1 solution generated is compared against the current best. 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 [2, 3]. Also, it can be applied to any constraint system regardless of whether they are WAODAG induced.

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

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

1. Depth-first search of the branch and bound tree.
2. Breadth-first search of the branch and bound tree.
3. Choose the active node whose parent node has the best lower bound.
4. Choose the active node whose parent's solution s^{opt} is "closest" to a 0-1 solution according to

$$\sum_{q \in lp(k)} \min\{s^{opt}(x_q), (1 - s^{opt}(x_q))\}.$$

For the choice of the next branching variable:

1. Choose only variables corresponding to hypotheses nodes since a 0-1 assignment to these variables guarantees a 0-1 assignment throughout the remaining variables.
2. Order the choice of variables such as shared nodes, then divide nodes, then hypothesis nodes, and so on.
3. Choose the variable x_q which minimizes
$$\{\min\{s^{opt}(x_q), (1 - s^{opt}(x_q))\}\}.$$
4. Choose the variable $x_q$which maximizes
$$\{\min\{s^{opt}(x_q), (1 - s^{opt}(x_q))\}\}.$$
5. Choose the variable which has maximum associated cost.
6. Use any combination of the above.

Techniques 1 and 2 seem to work best when values can propagate in a bottom-up fashion. Thus, using the induced constraint system as opposed to the semi-induced constraint system is desired.

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 3 and 4 together with branching variable heuristics 2 and 4 seem most promising.

Finally, an improvement can be made in the algorithm by having a 0-1 solution early in order to prune the branch and bound tree.

Theorem 3.1. Let s^{opt} be the optimal solution of some WAODAG semi-induced linear program. Construct a variable assignment s from s^{opt} by changing all non-zero values in s^{opt} to 1. s is a 0-1 solution for the linear program.

Proof. Let s be a variable assignment constructed from s^{opt} by changing all non-zero values in s^{opt} to 1. Assume s is not a 0-1 solution for the linear program.

Since s is a 0-1 assignment, this implies that there exists some linear constraint Q in the linear program which is violated. (For notational convenience, we will denote $s(x_p) = a$ by $x_p = a$ and $s^{opt}(x_p) = a$ by $x_p{}^{opt} = a$.)

Case 1. Q is of the form $x_p \leq x_q$. Since s is a 0-1 assignment,
$$x_p = 1 \text{ and } x_q = 0.$$
Now, $x_p = 1$ only if $x_p{}^{opt} > 0$ and $x_q = 0$ only if $x_q{}^{opt} = 0$. Since $x_p{}^{opt} > 0$ would have also violated Q and s^{opt} is a solution, Q cannot be violated.

Case 2. Q is of the form $x_p = 1$. Since s is a 0-1 assignment, $x_p = 0$. Now, $x_p = 0$ only if $x_p{}^{opt} = 0$. Since $x_p{}^{opt} = 0$ would have also violated Q and s^{opt} is a solution, 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 $q \in D$, $x_q = 0$. Now,

$x_p = 1$ only if $x_p^{opt} > 0$ and $x_q = 0$ only if $x_q^{opt} = 0$ for all $q \in D$. Since $x_p^{opt} > 0$ would have also violated Q and s^{opt} is a solution, Q cannot be violated.

Cases 1 to 3 cover every type of violations of the constraint system possible. Contradiction. Therefore, s is a 0-1 solution for the linear program. ♦

For a WAODAG induced constraint system L, we consider the following construction from s^{opt}: s is constructed from s^{opt} by changing all nonzero hypothesis node values in s^{opt} 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 3.2. s constructed above from s^{opt} is a 0-1 solution for L.

Proof. Follows from Theorems 2.1 and 2.3. ♦

4. Experimental Results and Conclusion

Lastly, we report our results using depth-first choice of active nodes and heuristic 4 for branching variables. Ninety-eight WAODAGs were randomly generated each consisting of 12 to 369 nodes. Table 4.1 presents a summary of the results obtained from our algorithm on these problems. The headings are interpreted as follows:

#nodes, #edges - the number of nodes and edges per WAODAG.

#evid - the number of evidence nodes per WAODAG.

#hyp-node - the number of hypothesis nodes per WAODAG.

#OR-node - the number of OR-nodes per WAODAG.

ShF - the average indegree for each node per WAODAG. Measures the sharing that occurs in the graph. Higher values imply higher connectivity in the graph.

#Active - the number of active nodes utilized per WAODAG.

#Iters - the total number of simplex method iterations [2, 3, 4] required including, if any, additional iterations from the branch and bound procedure.

I/(n+e) - the ratio of iterations to the sum of nodes and edges in the WAODAG.

Min/Max, Avg, Median, Total - the minimum, maximum, average, median and total over all WAODAGs.

Ratio - the ratio of the total value divided by the total number of nodes.

	#nodes	#edges	#evid	#hyp	#OR	ShF	#Active	#Iters	I/(n+e)
Min	12	18	1	3	3	1.06	0	21	0.3156
Max	369	881	10	116	151	5.28	220	3660	3.7938
Avg	174.3	302.86	5.184	55.96	59.08	1.82	7.5	406.36	0.7637
Med	132	208	5	45	31	1.44	0		
Total	17078	29680	508	5484	5790		735	39823	
Ratio	1	1.7391	0.03	0.321	0.339			2.3318	

Table 4.1. Experimental results using depth-first search and branching variable heuristic 4.

67 of the 98 WAODAGs were solved using only the initial linear programming problem while in general, the remaining problems were solved by evaluating only a small number of active nodes. A possible correlation can be made between ShF and #Act, although, it is not as clear as we would have liked. (See appendix for a complete listing of results from our approach.)

Our algorithm seems to be quite fast. However, no obvious method, aside from simple CPU timing analysis, has been found to compare it to existing work such as in [6, 7, 8, 9, 10] due to the rather different approach we have taken.

In conclusion, this approach can be used to solve the minimum cost-based abduction problem in an efficient manner. The formalism of linear constraint satisfaction provided a natural framework for finding minimal cost proofs. It seems likely that our approach can be extended to other problems in explanation and reasoning. Some of the extensions currently being incorporated include handling consistency, partial explanation, and the generation of alternative explanations in order of cost. Also, we intend to make detailed comparisons of our approach to that of [10].

5. References

[1] E. Charniak and S. E. Shimony, Probabilistic semantics for cost based abduction, *Proceedings of the 1990 National Conference on Artificial Intelligence*, 1990.

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

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

[4] C. McMillan, Jr., *Mathematical Programming*, John Wiley & Sons, Inc., 1975.

[5] F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*, Holden-Day, Inc., 1967.

[6] J. Hobbs, M. Stickel, P. Martin, and D. Edwards, Interpretation as Abduction, *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, 1988.

[7] B. Selman and H.J. Levesque, Abductive and default reasoning: a computational core, *Proceedings of the Eight National Conference on Artificial Intelligence*, 1990.

[8] M. Shanahan, Prediction is deduction but explanation is abduction, *IJCAI*, 1989.

[9] M.E. Stickel, A prolog-like inference system for computing minimum-cost abductive explanations in natural language interpretation, SRI International, Technical Note 451, 1988.

[10] E. Charniak and S. Husain, A new admissible heuristic for minimal cost proofs, Computer Science Department, Brown University, Technical Report.

A. Appendix

Table A.1 presents the results of our algorithm run on ninety-eight randomly generated problems. The table headings are as follows:

node - the total number of nodes in the WAODAG.

edges - the total number of edges in the WAODAG.

evid - the number of evidence nodes.

hyp - the number of hypothesis nodes.

AND - the number of AND-nodes.

OR - the number of OR-nodes.

ShF - the average indegree of nodes in the graph. Measures the amount of sharing which occurs. Higher values imply higher connectivity.

Ph1 - the number of iterations required in Phase I of the simplex method [2,3, 4].

Ph2 - the number of iterations required in Phase II of the simplex method [2,3, 4].

Acts - the number of active nodes processed to find the optimal 0-1 solution.

Ph3 - the total number of iterations using the dual simplex method to process all the active nodes [2, 3, 4].

nodes	edges	evid	hyp	AND	OR	ShF	Ph1	Ph2	Acts	Ph3
43	56	1	15	16	12	1.45	44	23	-	-
53	78	1	14	24	15	1.23	82	31	-	-
17	24	1	5	6	6	1.31	48	15	-	-
79	112	1	23	31	25	1.33	54	76	4	8
53	72	1	17	23	13	1.42	39	43	-	-
84	118	1	25	32	27	1.48	43	53	-	-
54	80	1	14	21	19	1.47	64	29	-	-
17	22	1	6	6	5	1.44	13	8	-	-
47	76	1	9	20	18	1.28	77	42	-	-
20	28	1	6	8	6	1.47	25	31	3	7
47	62	1	16	19	12	1.46	77	21	-	-
40	56	1	12	14	14	1.44	60	21	-	-
58	82	1	17	24	17	1.46	16	36	-	-
57	72	1	21	22	14	1.25	41	16	-	-
37	54	1	10	17	10	1.25	69	32	3	9
95	126	1	32	35	28	1.56	36	84	-	-

12	18	1	3	6	3	1.45	30	9	-	-
66	88	1	22	25	19	1.42	68	36	-	-
20	28	1	6	8	6	1.37	36	17	6	77
68	106	1	15	27	26	1.61	50	57	-	-
270	378	8	81	109	80	1.44	189	184	29	347
243	332	7	77	100	66	1.41	188	123	8	106
299	424	8	87	123	89	1.46	231	147	-	-
260	362	7	79	112	69	1.43	257	94	44	261
312	450	6	87	135	90	1.47	216	195	8	40
311	442	8	90	128	93	1.46	143	198	-	-
369	554	8	92	161	116	1.53	378	311	28	399
354	526	8	91	150	113	1.52	419	327	7	118
280	396	6	82	115	83	1.45	142	80	7	11
289	412	6	83	126	80	1.46	208	121	-	-
282	396	4	84	88	110	1.42	214	215	-	-
213	286	4	70	63	80	1.37	134	119	-	-
240	332	4	74	70	96	1.41	149	160	121	475
246	326	4	83	71	92	1.35	165	115	-	-
293	420	5	83	96	114	1.46	179	193	133	2333
309	436	5	91	96	122	1.43	205	242	-	-
184	236	4	66	54	64	1.31	81	70	-	-
242	330	4	77	74	91	1.39	204	119	5	12
322	460	5	92	95	135	1.45	229	201	3	7
233	314	4	76	70	87	1.37	138	112	11	33
263	382	5	72	110	81	1.52	266	164	-	-
91	102	5	40	25	26	1.21	64	15	-	-
242	258	5	113	48	81	1.1	87	87	-	-
218	302	5	67	85	66	1.46	167	112	-	-
88	108	5	34	26	28	1.19	67	48	-	-
104	104	5	52	22	30	1.14	54	29	-	-
89	90	5	44	22	23	1.06	57	12	-	-
97	104	5	45	21	31	1.21	50	16	-	-
76	88	5	32	29	15	1.24	47	28	-	-
256	280	5	116	61	79	1.13	129	90	-	-
337	490	8	92	149	96	1.49	322	175	10	66

326	470	9	91	135	100	1.48	422	233	8	54
275	386	10	82	81	112	1.46	266	131	3	4
349	514	8	92	106	151	1.51	363	350	30	412
77	84	9	35	20	22	1.24	55	19	-	-
173	208	10	69	44	60	1.28	113	69	4	12
108	122	10	47	42	19	1.24	72	21	-	-
262	362	10	81	78	103	1.44	270	124	-	-
92	104	8	40	34	18	1.24	65	22	-	-
204	272	10	68	82	54	1.4	155	58	-	-
273	460	3	88	113	72	1.7	191	127	-	-
185	265	8	76	40	69	1.5	86	50	3	6
283	464	6	95	74	114	1.68	247	112	11	51
166	229	9	73	61	32	1.46	114	23	-	-
330	577	10	97	83	150	1.8	472	131	-	-
215	346	5	87	85	43	1.65	179	92	3	7
219	346	8	86	52	81	1.64	164	117	6	17
283	496	9	94	76	113	1.81	284	47	-	-
228	371	4	87	54	87	1.66	249	69	-	-
142	188	5	64	27	51	1.37	91	48	-	-
239	395	10	88	98	53	1.72	316	80	-	-
188	270	9	86	69	33	1.51	149	42	-	-
148	183	8	69	50	29	1.31	92	41	-	-
154	201	5	78	51	25	1.35	69	49	-	-
89	105	5	48	16	25	1.25	48	18	2	3
189	268	10	75	73	41	1.5	145	48	-	-
132	179	5	65	43	24	1.41	86	23	-	-
84	101	6	43	28	13	1.29	59	13	-	-
94	101	7	48	18	28	1.16	50	27	-	-
249	406	6	92	60	97	1.67	214	96	5	13
200	821	3	88	97	137	2.57	200	203	-	-
41	109	5	10	16	15	3.03	52	9	-	-
304	770	10	75	93	136	2.62	1096	72	3	8
293	714	3	79	87	127	2.46	567	269	220	2824
307	881	3	76	141	90	2.9	1125	90	-	-
132	410	3	16	62	54	3.18	386	144	-	-

271	759	4	61	83	127	2.84	1395	101	3	11
324	847	4	74	108	142	2.65	1336	136	-	-
97	314	6	8	42	47	3.45	333	9	-	-
52	147	10	10	22	20	3.5	63	16	-	-
152	674	6	31	72	49	4.62	2110	71	4	18
55	237	6	15	24	16	4.84	247	42	-	-
73	319	6	18	35	20	4.76	227	8	-	-
182	697	6	42	67	73	3.96	443	51	-	-
32	118	3	12	9	11	4.07	65	13	-	-
136	549	9	36	45	55	4.32	633	67	-	-
128	660	3	14	54	60	5.28	695	39	-	-
164	703	3	35	58	71	4.37	1220	42	-	-

Table A.1. Experimental results using depth-first search and branching variable heuristic 4.