ELSEVIER

Artificial Intelligence 86 (1996) 157–170

Artificial Intelligence

Research Note

Polynomial solvability of cost-based abduction ★

Eugene Santos Jr [a,*], Eugene S. Santos [b,1]

[a] *Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH 45433-7765, USA*

[b] *Department of Computer and Information Sciences, Youngstown State University, Youngstown, OH 44555, USA*

Received May 1995; revised June 1996

Abstract

In recent empirical studies we have shown that many interesting cost-based abduction problems can be solved efficiently by considering the linear program relaxation of their integer program formulation. We tie this to the concept of total unimodularity from network flow analysis, a fundamental result in polynomial solvability. From this, we can determine the polynomial solvability of abduction problems and, in addition, present a new heuristic for branch and bound in the non-polynomial cases.

1. Introduction

In cost-based abduction [3,9], 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 [1,3,9]. Central to this approach is the use of *directed acyclic graphs* called WAODAGs (or, weighted AND/OR directed acyclic graphs) to represent relationships between hypotheses and the evidence to be explained. Each node represents some proposition, and the connections explicitly detail the relationships between different propositions. It has been shown in [3] that belief revision in Bayesian networks [7] can be accurately modeled by cost-based abduction. The costs are interpreted as negative log probabilities. Unfortunately, computing the minimal cost explanation has also been shown to be NP-hard [3].

★ Special thanks to the anonymous reviewers who helped significantly improve this paper.

* This research was supported in part by AFOSR Project #940006. E-mail: esantos@afit.af.mil.

[1] E-mail: santos@cis.ysu.edu.

PII S0004-3702(96)00016-1

Recent empirical [2,9] studies have shown that many interesting cost-based abduction problems can be solved efficiently performing better than existing best-first search techniques by considering the linear program relaxation of their integer program formulation. In particular, a significant number of test cases [2,9] were solved through linear programming alone without resorting to branch and bound. Linear programming is known to be solvable in polynomial time [6,10].

We study this phenomenon and determine the conditions for solving abduction problems in polynomial time. Our approach ties the concept of total unimodularity from network flow analysis to our cost-based problem. Total unimodularity is a fundamental result in optimization on polynomial solvability [6,10]. Furthermore, for those abduction problems requiring branch and bound, we present a new heuristic based on our results which potentially reduces the overall number of branches that need to be explored.

2. Solvability

Cost-based abduction problems are defined as directed graphs, called WAODAGs (or, weighted OR / AND directed acyclic graphs), where the nodes represent propositions, the connections represent logical relationships, and the weights at each node represent the costs [3,9]. Semantically, these costs can be treated as the negative logarithms of conditional probabilities as shown in [3].

Notation. $\mathbb{R}^+$ denotes the positive reals.

Definition 2.1. A WAODAG is a 4-tuple (G, c, r, S), where:
(1) G is a directed acyclic graph, $G = (V, E)$.
(2) $c : V \to \mathbb{R}^+$ is called the *cost function*.
(3) $r : V \to \{\text{AND}, \text{OR}\}$ is called the *label*. Such nodes are called AND-*nodes* and OR-*nodes* respectively.
(4) $S \subseteq V$ is called the *evidence nodes*.

Notation. For each node $q \in V$, $D_q = \{p \mid (p, q) \in E\}$, the parents of q.

Notation. $|\cdot|$ denotes the cardinality of a set.

An *explanation* of the evidence is the same as a proof. More formally, we define this as follows:

Definition 2.2. A *truth assignment* for a WAODAG $W = (G, c, r, S)$ where $G = (V, E)$ is a function $e : V \to \{\textbf{true}, \textbf{false}\}$. We say that such a function is an *explanation* for W if and only if the following conditions hold:
(1) $\forall q \in V$ s.t. $r(q) = \text{AND}, e(q) = \textbf{true} \Rightarrow \forall p \in V$ s.t. $(p, q) \in E, e(p) = \textbf{true}$.
(2) $\forall q \in V$ s.t. $r(q) = \text{OR}, e(q) = \textbf{true} \Rightarrow \exists p \in V$ s.t. $(p, q) \in E$ and $e(p) = \textbf{true}$.
(3) $\forall q \in S$, $e(q) = \textbf{true}$.

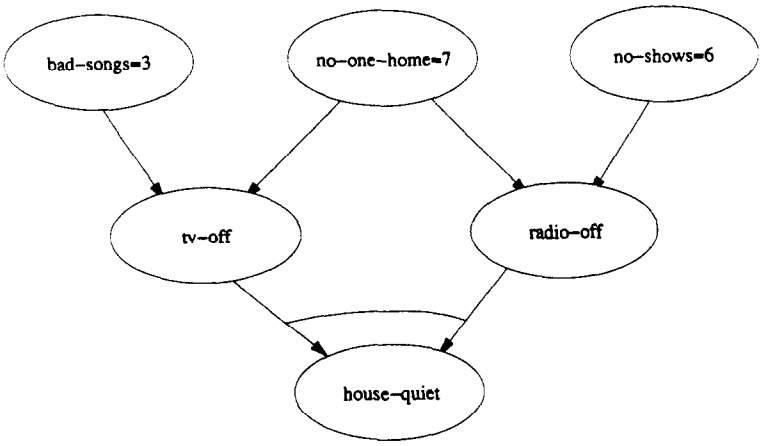

Fig. 1. A simpler WAODAG. The AND-node house-quiet is the observation. The nodes no-shows, no-one-home and bad-songs are the hypotheses with associated costs 6, 7 and 3, respectively.

To order the possible explanations, we define the *cost* of an explanation as follows:

Definition 2.3. The *cost* C of an explanation e for $W = (G, c, r, S)$ where $G = (V, E)$ is

$$C(e) = \sum_{\{q \in V \mid e(q)=\textbf{true}\}} c(q). \tag{1}$$

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

Note that conditions (1) and (2) in Definition 2.2 above are relaxed from the original definition found in [9]. According to Theorems 2.14 and 2.15 from [9], the best explanation defined here can be transformed into a best explanation for the original definition. Such a transformation can be achieved in $O(|E|)$ steps.

In Fig. 1, assume that house-quiet = **true** is the observation to be explained. One possible explanation 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**. We find that our three explanations above have costs 9, 16 and 7, respectively. Of the three, our best explanation is the third one with the cost of 7.

Given a WAODAG $W = (G, c, r, S)$ where $G = (V, E)$, construct a 0-1 *integer linear program*, $L(W)$, as follows: The real variables of $L(W)$ is the set of variables indexed by V, that is, $\{x_q \mid q \in V\}$. These variables are governed by the constraints

- For each x_q,
  (1) if $r(q) =$ AND, then

$$x_q \leqslant x_p \tag{2}$$

for each $p \in D_q$;

(2) if $r(q) = \text{OR}$, then

$$\sum_{p \in D_q} x_p \geqslant x_q. \tag{3}$$

- For each $q \in S$, $x_q = 1$.
- For each x_q, x_q is either 0 or 1.

The objective function being minimized is

$$\Theta_{L(W)} = \sum_{q \in V} x_q c(q).$$

Note: We have only taken a subset of the constraints from the original formulation [9] for our transformation. In particular, the *top-down* constraints in [9] are not considered.[2] Again, according to Theorems 2.14 and 2.15 from [9], this system is sufficient to find a best explanation for W.

Theorem 2.4. *An optimal solution for $L(W)$ is a best explanation for W.*

For Fig. 1, we have the following system of linear inequalities:

$$x_{\text{house-quiet}} \leqslant x_{\text{tv-off}},$$

$$x_{\text{house-quiet}} \leqslant x_{\text{radio-off}},$$

$$x_{\text{house-quiet}} = 1,$$

$$x_{\text{bad-songs}} + x_{\text{no-one-home}} \geqslant x_{\text{tv-off}},$$

$$x_{\text{no-shows}} + x_{\text{no-one-home}} \geqslant x_{\text{radio-off}},$$

and the following objective function

$$\Theta_{L(W)} = 3x_{\text{bad-songs}} + 7x_{\text{no-one-home}} + 6x_{\text{no-shows}}.$$

With the transformation to integer linear programming, we now examine the *total unimodularity* [6, 10] of the constraint system. Without loss of generality, assume that W is a binary graph, that is, each node has either 0 or 2 parents only. Any WAODAG can be directly transformed into a binary graph with at most a linear increase in the number of nodes and edges.

As mentioned in [9], the optimal solution to $L(W)$ may not be integral (or more specifically, 0-1). Consider the simple WAODAG in Fig. 2. From this WAODAG, we have the following system of linear inequalities:

$$x_{\text{tv-off}} + x_{\text{radio-off}} \geqslant x_{\text{house-quiet}},$$

$$x_{\text{house-quiet}} = 1,$$

$$x_{\text{tv-off}} \leqslant x_{\text{no-one-home}},$$

$$x_{\text{radio-off}} \leqslant x_{\text{no-one-home}},$$

[2] This corresponds to the semi-induced constraint system of [9].

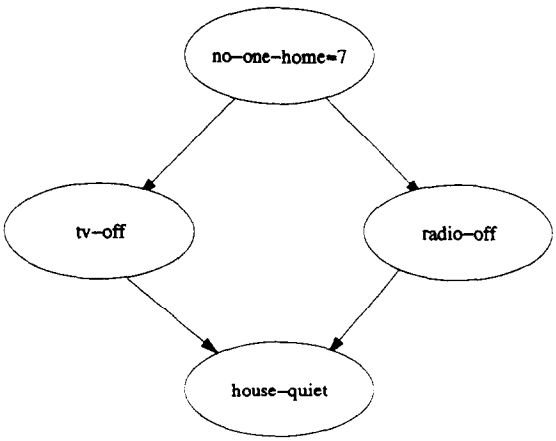

Fig. 2. A WAODAG with a non-integral optimal for $L(W)$.

and the following objective function

$$\Theta_{L(W)} = 7x_{\text{no-one-home}}.$$

We can easily show that the solution which minimizes the objective function is as follows: $x_{\text{house-quiet}} = 1$, $x_{\text{radio-off}} = 0.5$, $x_{\text{tv-off}} = 0.5$, and $x_{\text{no-one-home}} = 0.5$ with $\Theta_{L(W)} = 3.5$. The primary cause for the 0.5s arises from the inequality for the OR-node house-quiet.

Consider the following canonical form optimization problem:

$$\begin{aligned} \min_{x} \quad & \boldsymbol{cx} \\ \text{s.t.} \quad & \boldsymbol{Ax} \leqslant \boldsymbol{b}, \\ & x_i \text{ is 0 or 1.} \end{aligned} \tag{4}$$

The x_i correspond to our x_q where each q is a node above. $\boldsymbol{x}$ is called the *solution vector*. Each c_i in $\boldsymbol{c}$ corresponds to the cost of each node represented by x_i and $\boldsymbol{c}$ is called the *cost vector*. Finally, we can rewrite our above WAODAG linear inequalities as a matrix and vector multiplication where A is called the *constraints matrix* and $\boldsymbol{b}$ the *constraints vector*.

Observe that each column in the matrix A corresponds to a unique node in W and vice versa. For Fig. 1, we have the following system:

$$\boldsymbol{x} = \begin{bmatrix} x_{\text{house-quiet}} \\ x_{\text{tv-off}} \\ x_{\text{radio-off}} \\ x_{\text{bad-songs}} \\ x_{\text{no-one-home}} \\ x_{\text{no-shows}} \end{bmatrix}, \qquad \boldsymbol{c} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 3 \\ 7 \\ 6 \end{bmatrix},$$

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}$$

Next, we define the concept of *parity* as follows:

Definition 2.5. Given an undirected simple cycle,

$$\tau = \{(p_1, p_2), (p_2, p_3), \ldots, (p_{n-1}, p_1)\},$$

in G, the *parity* of τ is

$$|\{\rho \in V \mid r(\rho) = \text{OR},\ (p, \rho) \in \tau,\ (\rho, q) \in \tau,\ \text{and } D_\rho = \{p, q\}\}| \quad \text{modulo 2.} \quad (5)$$

Furthermore, if the parity equals 1, then τ is an *odd-cycle*. Otherwise, it is an *even-cycle*.

Intuitively, parity is a measure on the number of OR-nodes and their parents that are traversed by the cycle. We now extend this over W.

Definition 2.6. W is *parity-balanced* if for any undirected simple cycle τ in G, τ is an even-cycle.

Notation. For simplicity, we can also represent τ by the sequence of nodes $\{p_1, p_2, \ldots, p_n\}$ where $p_n = p_1$.

Definition 2.7. A is said to be *totally unimodular* if all of its square submatrices have determinant equal to either 0, +1, or −1.

Consider the following theorem on *total unimodularity* from [10, Theorem 19.3(iv)]:

Lemma 2.8. *A is totally unimodular iff each collection of columns of A can be split into two parts so that the sum of the columns in one part minus the sum of the columns in the other part is a vector with entries only* 0, +1, *and* −1.

Based on our derivation of A from W, the entries in A are either 0, +1, or −1. Furthermore, for any submatrix A' formed by arbitrarily selecting sets of columns from A, each row in A' will have one of the following configuration of non-zeros:

(1) All zeros.
(2) One ±1.
(3) One +1 and one −1.
(4) Two −1s.
(5) One +1 and two −1s.

With regards to Lemma 2.8 on columns splitting, the first two configurations will have no effect. The third configuration requires that the two columns involved must belong in the same grouping, otherwise, the results for that row will be ± 2. The fourth configuration requires that the two columns be separated. The last configuration prohibits placing the $+1$ column in one group and both the -1s in the other group.

Intuitively, Lemma 2.8 represents a dependency graph between the columns in the matrix. As long as this graph is bipartite, we can partition up the columns to satisfy the lemma. Our goal is to show that parity-balance guarantees a bipartite graph and vice versa. We now prove the following theorem on total unimodularity:

Theorem 2.9. *A is totally unimodular iff W is parity-balanced.*

Proof. ($\Rightarrow$) Let A be totally unimodular and assume that W is not parity-balanced. This implies that there exists an odd-cycle τ in G. Let $\tau = \{p_1, p_2, \ldots, p_n\}$ where $p_n = p_1$. An OR-node ρ is said to be completed in τ if it contributes to the parity of τ, i.e., $r(\rho) = \text{OR}$, $(p, \rho) \in \tau$, $(\rho, q) \in \tau$, and $D_\rho = \{p, q\}$. The number of completed nodes in τ is odd.

Rewrite τ as follows:

$$\{q_1, p_{1,1}, p_{1,2}, \ldots, p_{1,m_1}, q_2, p_{2,1}, \ldots, p_{2,m_2}, q_3, \ldots, q_k, p_{k,1}, \ldots, p_{k,m_k}, q_1\}$$

where $\{q_1, \ldots, q_k\}$ are all the completed OR-nodes in τ. Let $\boldsymbol{a}_{i,j}$ be the column in A associated with $p_{i,j}$ for $j = 1, \ldots, m_i$ and $i = 1, \ldots, k$. Form a collection of columns with this set and call the matrix A'. From Lemma 2.8, if the columns in A' can be split, then the columns of any submatrix of A' formed by removing rows from A' can also be split. There may be OR-nodes in τ whose parents are both also in τ but is not completed. Eliminate the rows relating such OR-nodes to their parents, i.e., those from inequality (3). Now, the remaining rows in A' can be in any configuration except the fifth one as described earlier.

Claim. *For each $i = 1, \ldots, k$, all $p_{i,j}$ for $j = 1, \ldots, m_i$ must belong in the same group.*

Consider any two consecutive nodes $p_{i,j}$ and $p_{i,j+1}$ in τ. One is necessarily the parent of the other. Without loss of generality, say $p_{i,j}$ is the parent. If $p_{i,j+1}$ is an AND-node, then there is a row in A' that satisfies configuration (3) with the $+1$ in $\boldsymbol{a}_{i,j+1}$ and -1 in $\boldsymbol{a}_{i,j}$. Similarly, this holds even if $p_{i,j+1}$ is an OR-node, since it cannot be a completed node. This implies that the two nodes must belong to the same group since the columns must belong in the same part in order to satisfy Lemma 2.8.

Denote these groups simply by P_i.

Claim. *P_i can never be grouped with P_{i+1} for $i = 1, \ldots, k-1$ and P_k cannot be grouped with P_1.*

For each completed node q_i, $p_{i-1,m_{i-1}}$ and $p_{i,1}$ are the parents of q_i for $i = 2, \ldots, k$, and for q_1, the parents are p_{k,m_k} and $p_{1,1}$. Also, there must exist a row of configuration (4) above such that the two -1s correspond to the parents. This implies that these two nodes must always be in separate groups to satisfy Lemma 2.8. Thus, P_i can never be grouped with P_{i+1} for $i = 1, \ldots, k-1$ and P_k cannot be grouped with P_1.

For $k = 1$, we have an inconsistency where $p_{1,1}$ and p_{1,m_1} must both be in the same group as well as both be separated.

For $k > 1$, this leaves us with the problem of merging the P_i until we only have two groups while simultaneously satisfying the grouping and separation constraints. We can represent this by an undirected graph Δ whose nodes correspond to the P_i and arcs between the nodes correspond to a separation condition. Our problem now reduces to whether Δ is bipartite.

Clearly, Δ is a simple cycle of odd length since the number of completed nodes is odd. Thus, Δ cannot be bipartite [5]. Contradiction.

($\Leftarrow$) Let W be parity-balanced and let $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n\}$ be any collection of columns in A. Let A' be the submatrix of A formed from the $\boldsymbol{a}_i$.

Let $V' = \{p_1, p_2, \ldots, p_n\}$ be the nodes associated with the columns $\boldsymbol{a}_i$. Construct the subgraph $G' = (V', E')$ from G as follows: For all i, j from 1 to n, if $(p_i, p_j) \in E$, then $(p_i, p_j) \in E'$.

We say an OR-node is weakly completed with respect to G' if all its parents are also in G'. We observe the following: Given $(p_i, p_j) \in E'$, (i) $r(p_j) =$ AND or (ii) $r(p_j) =$ OR and p_j is not weakly completed if and only if there exists exactly one row in A' of configuration (3) such that the non-zeros are only in $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$.

Define a relation R on V' such that $p_i R p_j$ iff $i = j$ or there exists an undirected path τ from p_i to p_j where τ does not contain any weakly completed OR-node. Clearly, R is an equivalence relation and let $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ be the partition on V' induced by R. Intuitively, these σ_k represent the necessary grouping of nodes/columns to guarantee total unimodularity.

Next, construct an undirected graph Δ whose nodes correspond to each σ_k. For each row of configuration (4) on A', if the -1s occur in $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$, then connect σ_{k_i} to σ_{k_j} where $p_i \in \sigma_{k_i}$ and $p_j \in \sigma_{k_j}$.

Claim. *σ_k is never directly connected to itself.*

If both p_i and p_j belong to σ_k, then there is an undirected path from p_i to p_j containing no weakly completed OR-nodes. By construction of A from W, p_i and p_j are the parents of some OR-node q outside of V'. We can construct a cycle by including q. By Definition 2.5, this is an odd-cycle. This would contradict our assumption of parity-balance for W.

Finally, we must consider rows of configuration 5 in A'. Such rows indicate weakly completed OR-nodes in G'. Assume p_i is such a node and p_{j_1} and p_{j_2} are the parents of p_i.

Claim. *Both p_{j_1} and p_{j_2} cannot belong in the same partition σ_k.*

Otherwise, there is an undirected path from p_{j_1} to p_{j_2} containing no weakly completed OR-nodes. We can construct a cycle in the path by including p_i. By Definition 2.5, this is an odd-cycle. This would contradict our assumption of parity-balance for W.

Since the parents of a weakly completed OR-node cannot be in the same partition, place an arc in Δ between the corresponding partitions containing the parents.

As before, we must now prove that Δ is bipartite. Assume that Δ has an odd-length cycle. Each arc in Δ corresponds to an OR-node in G. We can construct an undirected

cycle in G by first constructing path segments which are free of weakly completed OR-nodes in the nodes of Δ being traversed and join them via the OR-nodes represented by the arcs. Hence, we have an odd-cycle in G. Contradiction. Therefore, Δ is bipartite and A' satisfies the columns splitting of Lemma 2.8. □

The Simplex Method for solving linear programs searches along the vertices of the polyhedron defined by the constraint system. Combined with the following theorem from [10, Theorem 19.3(ii)]:

Theorem 2.10. *A is totally unimodular iff for each integral vector* $\boldsymbol{b}$, *the polyhedron* $\{\boldsymbol{x} \mid \boldsymbol{x} \geqslant 0,\ A\boldsymbol{x} \leqslant \boldsymbol{b}\}$ *has only integral vertices.*

This guarantees that the optimal solution found by Simplex will be integral. Hence, Simplex alone will solve WAODAG W if W is parity-balanced without resorting to branch and bound.

However, Simplex has an exponential worst-case run-time. Linear programming problems can be solved in polynomial time through methods such as Khachiyan's [10] but these methods do not necessarily find an optimal solution which is integral. It is possible for there to exist an integral and a non-integral solution both of which are optimal (same objective value). Simplex searches for an optimal solution by exploring the extreme points of the constraint space [4,6]. Total unimodularity guarantees that all extreme points are integral. Hence, Simplex will find an integral optimal. As it turns out, taking the solution generated with Khachiyan's method, the optimal integral solution can be found in polynomial time when W is parity-balanced. This follows from [10, Theorem 16.2].

Let's consider two simple WAODAGs in Figs. 3 and 4 where the first is of even parity and the other is of odd parity. Let

$$\boldsymbol{x} = \begin{bmatrix} x_{\text{house-quiet}} \\ x_{\text{tv-off}} \\ x_{\text{radio-off}} \\ x_{\text{no-one-home}} \end{bmatrix}.$$

The corresponding constraint matrices are

$$A_{\text{even}} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix}, \qquad A_{\text{odd}} = \begin{bmatrix} 1 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix}.$$

According to Lemma 2.8, we find that taking the last three columns of A_{odd}, there is no way to split them into two sets to satisfy the lemma. On the other hand, A_{even} does indeed satisfy the conditions for total unimodularity.

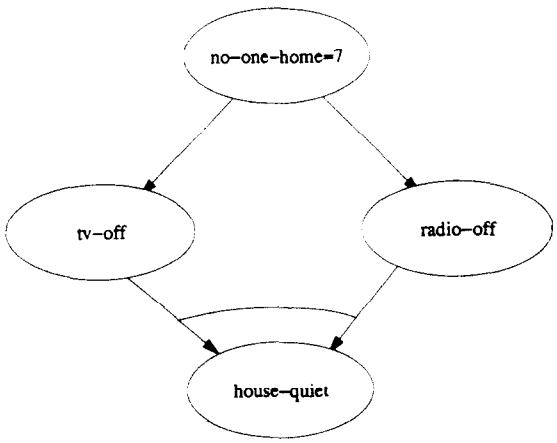

Fig. 3. A simple even parity WAODAG.

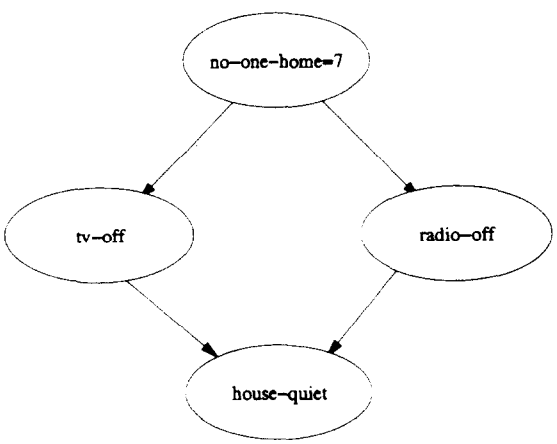

Fig. 4. A simple odd parity WAODAG.

Proposition 2.11. *Cost-based abduction with parity-balanced* WAODAG*s can be solved in polynomial time.*

The total unimodularity of a matrix can be determined in polynomial time according to [10, Theorem 20.3]. Hence, we can determine parity-balance also in polynomial time. An alternative to this approach is to check for parity-balance directly for each undirected cycle in the graph. 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 [8].

Now, consider the case when the constraint system is not totally unimodular. The vertex corresponding to the optimal solution in this polyhedron may still be integral.

Let n be the number of variables and m be the number of equations in our constraint system. For each vertex v in our polyhedron, there are n constraints which define v. Construct a subgraph G_v from G as follows:

- For each inequality like (3), we include the OR-node, its parents and the edges between them.
- For each inequality like (2), we include the AND-node, the single parent involved and the edge between them.

Constraints for evidence are ignored since they will have no impact on total unimodularity of the resultant matrix. (The row associated with such a constraint has a single non-zero value of 1.) Parity and parity-balanced can be naturally extended to G_v even though AND-nodes may only have one parent.

Theorem 2.12. *If G_v is parity-balanced, then v is integral.*

Proof. If G_v is parity-balanced, then it follows with slight modifications from Theorem 2.9, that the coefficient matrix of the n equations is totally unimodular. Thus, v is integral. □

The integrality of a vertex depends on its associated subgraphs. Hence, linear programming alone can also result in an integral optimal solution.

Let $W' = (G', c', r', S')$ where $G' = (V', E')$ be a WAODAG.

Definition 2.13. W' is said to be a *partial* WAODAG of W if the following conditions hold:

- $V' = V$.
- G' is a subgraph of G such that $\forall p \in V$, if $r(p) =$ OR then either
  - $\forall (q,p) \in E, (q,p) \in E'$, or
  - $\forall (q,p) \in E, (q,p)$ is not in E'.
- $c' \equiv c$.
- $r' \equiv r$.
- $S' = S$.

Theorem 2.14. *Given a* WAODAG *W, let e be a best explanation for W. We can determine a best explanation for W in polynomial time if there exists a partial* WAODAG *W' of W such that W' is parity-balanced and e is also a best explanation for W'.*

Proof. Observe that the polytope for W' is a superset of the polytope for W. In fact, the matrix for W' is a submatrix of the one for W. Furthermore, since e is also a best explanation for W', there cannot exist any solution $\boldsymbol{x}$ in W such that $\boldsymbol{cx} < C(e)$ regardless of whether $\boldsymbol{x}$ is integral or non-integral.

With Khachiyan's method, we can find $C(e) = \max\{\boldsymbol{cx} \mid \boldsymbol{Ax} \leqslant \boldsymbol{b}\}$ and a system of linear equations $\boldsymbol{A'x} = \boldsymbol{b'}$ determining a minimal face F of $\{\boldsymbol{x} \mid \boldsymbol{Ax} \leqslant \boldsymbol{b}\}$ so that each $\boldsymbol{x}$ in F satisfies $\boldsymbol{cx} = C(e)$ [10].

We can find an optimal integral solution for W in polynomial time according to Corollary 5.3b from [10] which states that: *Given a system of rational linear equations, we can decide if it has an integral solution, and if so, find one, in polynomial time.* □

From Theorem 2.14, it is not necessary for the original graph to be parity-balanced to ensure that an optimal solution to the derived constraint system is integral. As long as there exists a partial WAODAG which is parity-balanced and shares the same optimal integral solution as the original, Simplex alone can be used to determine a best explanation. This can serve to explain the empirical results in [2,9].

Finally, given the observations on parity-balance, the clamping of evidence nodes in W can also impact total unimodularity indirectly. Construct a new WAODAG $W_1 = (G', c, r, S)$ where $G' = (V, E')$ from W as follows: Initially, $E' = E$. Remove edge $e = (p, q)$ from E' if $D_q \cap S$ is not empty and $r(q) = \text{OR}$. Intuitively, q is automatically supported by one of its parents.

Proposition 2.15. *The best explanation for W_1 is the best explanation for W.*

W_1 reduces the size of the problem and eliminates potential odd parity cycles from W. More importantly, this generalizes to the branch and bound process further eliminating such cycles.

In the bounding process, a variable x_q can be clamped to either 0 or 1. Proceeding down the branch and bound tree, each problem instance has two associated sets of nodes V_0 and $V_1 \supseteq S$ which must be clamped to 0 and 1 respectively. Let $\overline{V_0}$ be defined recursively as follows:

- $V_0 \subseteq \overline{V_0}$.
- If $r(q) = \text{AND}$ and $D_q \cap \overline{V_0}$ is not empty, then $q \in \overline{V_0}$.
- If $r(q) = \text{OR}$ and $D_q \subseteq \overline{V_0}$, then $q \in \overline{V_0}$.

If $\overline{V_0} \cap V_1$ is not empty, the problem instance is infeasible. Otherwise, construct W' from W as follows:

(1) For each $p \in \overline{V_0}$, remove all edges incident on p from G.
(2) If $D_q \cap V_1$ is not empty and $r(q) = \text{OR}$, remove edge $e = (p, q)$.
(3) If $D_q \subseteq V_1$ and $r(q) = \text{AND}$, remove edge $e = (p, q)$.

Definition 2.16. e is a *consistent explanation* for W with respect to V_0 and V_1 if the following conditions hold:

- e is an explanation for W.
- $\forall q \in V_0$, $e(q) =$ **false**.
- $\forall q \in V_1$, $e(q) =$ **true**.

The *best* consistent explanation is one which minimizes the cost function $C(\cdot)$.

The following relationship between W and W' holds:

Theorem 2.17. *e is a consistent explanation for W with respect to V_0 and V_1 iff e is a consistent explanation for W' with respect to $\overline{V_0}$ and V_1.*

Proof. ($\Rightarrow$) Let e be a consistent explanation for W with respect to V_0 and V_1. By the construction of W' above as a subgraph of W, e is an explanation for W'. Also, $\forall q \in V_1$, $e(q) =$ **true**. Finally, from Definition 2.2 and the construction of $\overline{V_0}$ above, $\forall q \in \overline{V_0}$, $e(q) =$ **false**. Therefore, e is a consistent explanation for W' with respect to $\overline{V_0}$ and V_1.

($\Leftarrow$) Let e be a consistent explanation for W' with respect to $\overline{V_0}$ and V_1. Since V_0 is a subset of $\overline{V_0}$, for all q in V_0, $e(q) =$ **false**. Also, for all q in V_1, $e(q) =$ **true**.

If e' is not an explanation for W, then e' violates one of the following conditions from Definition 2.2:

(1) $\exists q \in V$ s.t. $r(q) =$ AND, $e(q) =$ **true**, and $\exists p \in D_q(W)$ s.t. $e(p) =$ **false**. $D_q(W)$ are the parents of q in W.

(2) $\exists q \in V$ s.t. $r(q) =$ OR, $e(q) =$ **true**, and $\forall p \in D_q(W)$, $e(p) =$ **false**.

(3) $\exists q \in S$ s.t. $e(q) =$ **false**.

Case (1). Let p and q be the two nodes. If $p \in D_q(W')$, then e' cannot be an explanation for W'. If p is not in $D_q(W')$, by our above construction, $p \in \overline{V_0}$ or $q \in \overline{V_0}$. If $q \in \overline{V_0}$, then e' cannot be a consistent explanation for W' with respect to $\overline{V_0}$ and V_1. If $p \in \overline{V_0}$, then q must also be in $\overline{V_0}$. Contradiction.

Case (2). Let q be the node. If $D_q(W')$ is not empty, then e' cannot be an explanation for W'. For each $p \in D_q(W)$, either $p \in \overline{V_0}$ or $p \in V_1$. p cannot be in V_1 since $e(p) =$ **false**. Thus, $D_q(W) \subseteq \overline{V_0}$ implying that $q \in \overline{V_0}$. Contradiction.

Case (3). $S = S'$. Contradiction.

Therefore, e' is an explanation for W. Thus, e is a consistent explanation for W with respect to V_0 and V_1. □

Thus, W and W' are equivalent WAODAGs under their respective clampings.

Theorem 2.18. *The best consistent explanation for W with respect to V_0 and V_1 is the best consistent explanation for W' with respect to $\overline{V_0}$ and V_1.*

Proof. Follows from Definition 2.16 and Theorem 2.17. □

This leads to the following new heuristic for branch and bound: Let $V_o \subseteq V$ be the set of all OR-nodes in G such that the OR-node contributes to the parity of some undirected simple cycle in G.

Theorem 2.19. *If all the nodes in V_o are clamped, then the resulting* WAODAG *W' constructed from W above can be solved in polynomial time.*

Proof. If all the nodes in V_o are clamped, then there cannot exist any cycles in G' constructed from G above with odd parity. Hence, W' is parity-balanced. From Theorem 2.9, the constraint system for W' is totally unimodular and thus solvable in polynomial time. □

Theorem 2.20. *The worst-case number of branches needed to find the optimal integral solution is $2^{|V_o|} \leqslant 2^{|V|}$.*

Proof. Follows from our above construction and Theorem 2.19. □

3. Conclusion

Parity-balance is a necessary and sufficient condition for guaranteeing the total unimodularity of a cost-based abduction problem for arbitrary cost functions. Thus, parity-balance also guarantees polynomial solvability by using our transformation to integer linear programming. Furthermore, the integrality of each vertex in our polyhedron relies on the associated subgraphs of the cost-based problem. Finally, a new heuristic for branch and bound motivated by our results provides a new tighter upper bound on the worst-case performance of integer linear programming for solving cost-based abduction.

Most work in cost-based abduction and related problem domains such as weighted abduction [11] have been mainly empirical in nature. However, the results from experimentation have indicated general characteristics from the problem domain that allow for more efficient computations. This work has strived to provide a theoretical explanation for such an observation.

References

[1] E. Charniak and S. Husain, A new admissible heuristic for minimal-cost proofs, in: *Proceedings AAAI-91*, Anaheim, CA (1991) 446–451.

[2] E. Charniak and E. Santos Jr. Dynamic map calculations for abduction, in: *Proceedings AAAI-92*, San Jose, CA (1992) 552–557.

[3] E. Charniak and S.E. Shimony, Cost-based abduction and MAP explanation, *Artif. Intell.* **66** (1994) 345–374.

[4] F.S. Hillier and G.J. Lieberman, *Introduction to Operations Research* (Holden-Day, San Francisco, CA, 1967).

[5] J.A. McHugh, *Algorithmic Graph Theory* (Prentice-Hall, Englewood Cliffs, NJ, 1990).

[6] 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, Amsterdam, 1989).

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

[8] E.M. Reingold, J. Nievergelt and N. Deo, *Combinatorial Algorithms: Theory and Practice* (Prentice-Hall, Englewood Cliffs, NJ, 1977).

[9] E. Santos Jr, A linear constraint satisfaction approach to cost-based abduction, *Artif. Intell.* **65** (1994) 1–28.

[10] A. Schrijver, *Theory of Linear and Integer Programming* (Wiley, New York, 1986).

[11] M. Shanahan, Prediction is deduction but explanation is abduction, in: *Proceedings IJCAI-89*, Detroit, MI (1989) 1055–1060.