Computer Science Department Technical Report
Cognitive Systems Laboratory
University of California
Los Angeles, CA 90024-1596

CONSTRAINT-DIRECTED APPROACH TO DIAGNOSIS

Rina Dechter
Judea Pearl

November 1988
CSD-880093

TECHNICAL REPORT
R-72-I
September 1988

CONSTRAINT-DIRECTED APPROACH TO DIAGNOSIS*

Rina Dechter** and Judea Pearl
Cognitive Systems Laboratory
UCLA Computer Science Department
Los Angeles, California 90024
dechter@oahu.cs.ucla.edu, judea@lanai.cs.ucla.edu
Phone: (213) 825-3243

ABSTRACT

The paper presents a constraint-satisfaction framework for formulating diagnosis problems and an algorithm for identifying the minimal sets of faulty components that explain an observed behavior. The algorithm runs in linear time whenever the constraint graph is singly connected. In general, the complexity is exponential in the clique-size of any chordal graph in which the constraint graph can be embedded. Examples of circuit diagnosis problems are worked out and illustrated.

Topic: Principles, diagnosis
Status: research
Domain: circuit diagnosis

* This work was supported in part by the National Science Foundation, Grant #DCR 85-01234 and Air Force, Grant #AFOSR 88-0177.

** Current address: Computer Science Department, Technion, Haifa, Israel

1. INTRODUCTION

The behavior of a physical system (e.g., electronic circuit) is usually specified in terms of an assembly of interconnected components, each complying with some local, input-output constraint. For example, a circuit can be modeled by the following system of constraints: measurable quantities, including intermediate (unseen by the user) inputs and outputs, are real *variables*, and the status of each component is represented by a binary-valued variable. The value "0" to a component-variable indicates that it is functioning properly, while a value "1" indicates a faulty component. A constraint is an i-ary relation between a component and its input and output variables.

To illustrate this formulation, consider the example of Figure 1, treated in [De Kleer and Williams 86] [Davis 84] and [Genesereth 84]. M_1, M_2, and M_3 are multipliers, while A_1 and A_2 are adders. The inputs appear on the left, the outputs on the right. The numbers in brackets are the expected values at all potentially-observed points. The problem is, given the digital circuit depicted in Figure 1, to find the set of malfunctioning components which most likely would have caused the observed behavior.

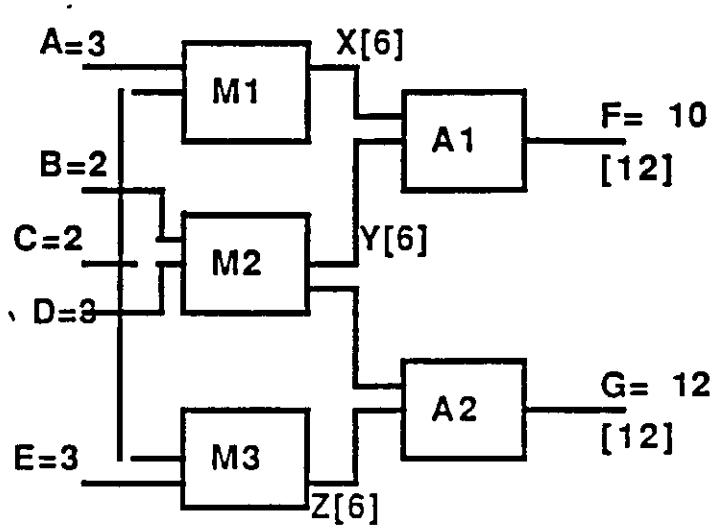


Figure 1.

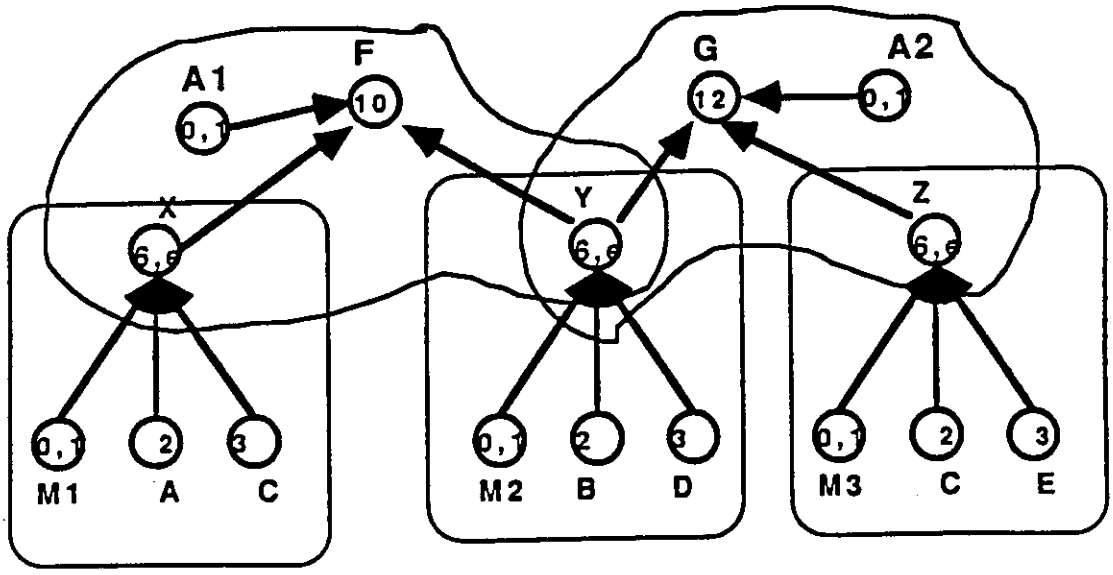


Figure 2.

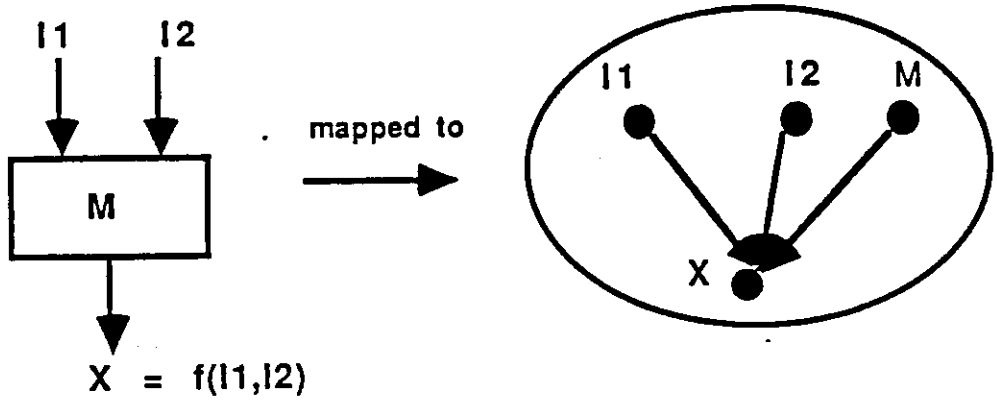


Figure 3.

The constraint network corresponding to this circuit is shown in Figure 2. Constraints are represented by a dark circle over the associated arcs (see Figure 3). The arcs can be directed or undirected. In the present example the arcs are directed indicating the functional dependence between inputs and outputs. The constraint representing multiplier M_1 is given by the following table:

A	C	M	X
2	3	0	6
2	3	1	e

where e stands for "any value".

Initially, when no observation is available, the "expected values" of the output variables are determined by a solution consistent with the assumption that all components are functioning properly. Typically, there is only one solution to the initial problem, and this which determines an unique set of "expected values" for all variables. The diagnostic task is initiated when one of the expected values differs from the observed value, and the task is to find a new solution consistent with the observed value. In the example above, the expected value 12 of F differs from the observed value 10. A new consistent solution will have one or more of the component-variables with value "1", indicating a set of malfunctioning components which constitutes a possible diagnosis of the problem.

We are normally not interested in all diagnoses, but with those that are *minimal,* i.e., solutions for which no subset of component-variables can switch their values from "1" to "0" and still be consistent. More ambitiously, we may wish to identify a diagnosis with minimal *number* of faulty components, i.e., a consistent solution with minimal number of "1"s for component-variables.

Constraint Satisfaction Problems (CSPs) are, in general, NP-hard, so the diagnosis problem may, in the worst case, require exponential complexity. However, in cases where the constraint graph is sparse, efficient techniques are available, based on tree-decomposition [Dechter and Pearl, 1987, 1988], which find a consistent solution in a reasonable time.

In this paper we extend these techniques to the task of finding minimal solutions. We first demonstrate the method on Tree-CSPs, i.e., CSPs having constraint graphs in the form of a tree of binary constraints, then extend the method to non-binary constraints. The extension to general CSPs is straightforward [e.g., Dechter and Pearl, 1988] and involves clustering compound variables to form a tree-structure (called a join-tree), then running the tree algorithm on the join-tree. The complexity of this method is exponential with the size of the largest cluster formed, namely, the size of the largest clique of the chordal graph in which we choose to embed the given constraint graph.

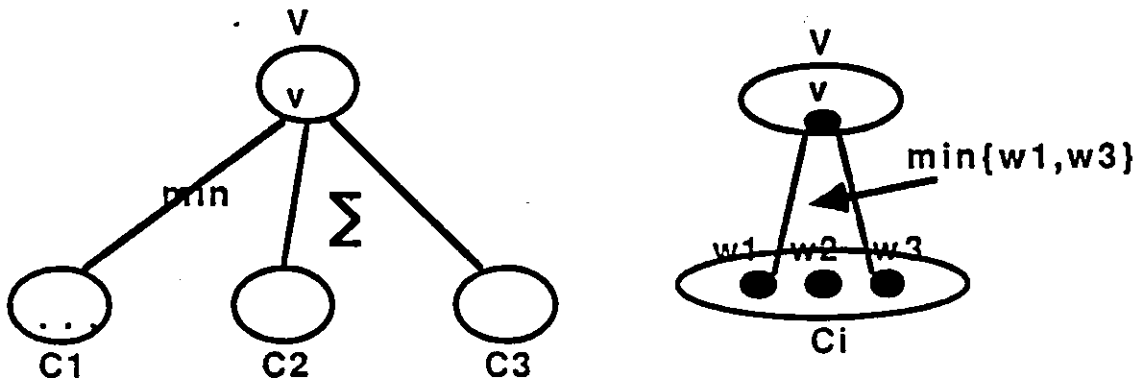
2. THE DIAGNOSIS PROCESS

We next provide a scheme for finding a solution with minimal numbers of "1"s on a Tree-CSP. The scheme bears many similarities to that developed by probabilistic considerations [e.g., Geffner and Pearl, 1987; Pearl, 1988], but casting it in constraint-based formalism facilitates the use of many techniques and languages developed for constraint processing.

2.1 Finding minimal-number-of-1's solutions in a tree-CSP

Given a Tree-CSP, select a root and direct the arcs from leaves to root. Associate with each value of each variable a **weight** w, defines as follows. Let V be a variable with a value v (v can be "0" or "1"). The weight $w(v)$ measures the minimal number of "1"s in a partial consistent solution of the subtree rooted at V, when the value of V is v. The weight of a "0" leaf values is "0" and the weight of a "1" leaf values is "1".

The weights of the rest of the values can be computed recursively from leaves to the root as follows: Given a variable V with value v and a set of child-nodes C_1, C_2, C_3 as in Figure 4, for each child node, C, compute the minimal weight among all its values-weights that are consistent with v in V. The sum of these minimal weights over all the child nodes is the weight of v. Each value also keeps pointers to all its children values that correspond to these minimum weights.



$$w(v) = v + \sum_{Ci} \min\{w(cij) \mid (v,cij) \text{ consistent}\}$$

Figure 4.

Clearly, the minimum weight associated with root node values is the desired minimal number of "1"s in a complete solution. Such a solution (or all such solutions) can be found by tracing the marked pointers going from the root back to the leaves. The complexity of marking the weights is $O(nk^2)$ when n is the number of variables and k is the number of values. The process of walking along the pointers to retrieve a solution with minimal number of "1's" is $O(nk)$.

When only a subset of variables (i.e., component-variables) are of interest, we want to find a solution with minimal number of "1"s in these designated variables. The scheme requires only a minor modification, ensuring that non-component variables will not add their own value to the weight calculation.

2.2 Handling Non-Binary Constraints

To handle non-binary constraints we assume that the underlying graph generated by all the constraints is a tree (as is the case in our example). We generate a different graphical representation of the problem in which nodes are either single nodes (as before) or compound nodes, containing a subset of single nodes. Arcs will be directed from a single or a compound node into a single node. Figures 5 and 6 and 7 show steps of this process:

1. Considering first only the tree-representation, induce direction on the tree to generate a directed tree. If the graph is directed already change a minimum number of directions to result in a rooted tree. (Compare figures 5 and 2, arrows point from children to parents).

2. For each circled subset of variables there is one variable which is an ancestor of all the rest in the directed tree generated at step 1. Make all the children nodes one compound variable (indicated by a circle) and make this circled node a child node of the ancestor variable.

In Figure 5, for the circled subset $\{A_1\ X\ F\ Y\}$, Y is an ancestor of X F and A_1 and therefore, its new graphic representation is given in Figure 6 and 7.

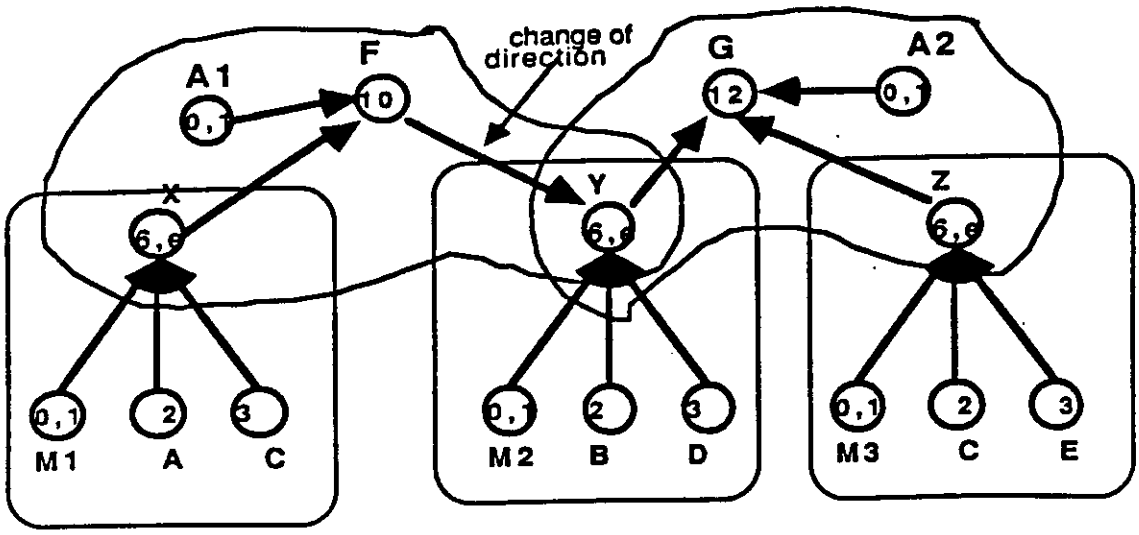


Figure 5. Redirecting arcs to generate a rooted tree.

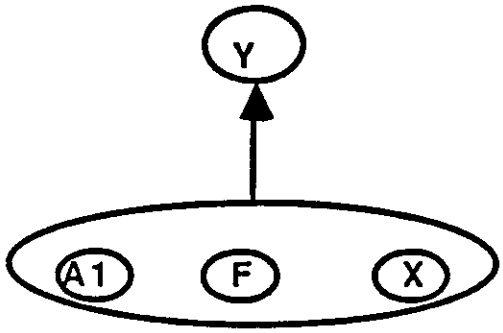


Figure 6.

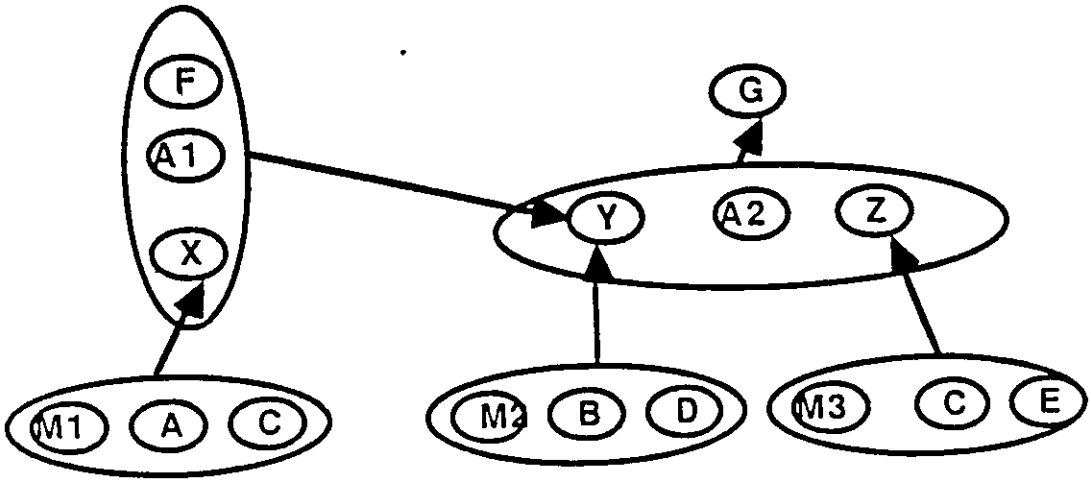


Figure 7.

In the new graph representation, arcs represent binary constraints between compound variables. In this specific example the constraints are between compound and single variables. The constraints themselves can be easily derived from the initial specifications, recalling that the set of values for a compound variable are all the possible tuples of the single variables' values that appear in the initial constraint. For instance, the set of values of the compound variable (M_1, A, C) are {(0, 2, 3), (1, 2, 3)} and the constraint with variable X will permit the value $X = 6$ with the value (0, 2, 3) and the value e with the value (1, 2, 3).

The weight computation is now defined as for binary trees, using summation to compute the weight of a tuple from the weights of its single values.

3. A DETAILED EXAMPLE

We give a step by step calculation of the weights of the root node G.

1. **Calculating Weights of Leafs:** Figure 7 shows the weights of all leaf nodes' values. For example the weights of the compound variable $M_1, A, C)$ are computed as follows:

   $w(M_1 = 0, A = 2, B = 3) = w(M_1 = 0) + w(A = 2) + w(B = 3)$. Only M_1 is a device variable and its value is 0, therefore: $w(M_1 = 0) = 0$ since A, and B are not device variables we have: $w(A = 2) = 0$, $w(C = 3) = 0$, and therefore $w(0, 2, 3) = 1$. Similarly we get: $w(M_1 = 1, A = 2, C = 3) = 1$.

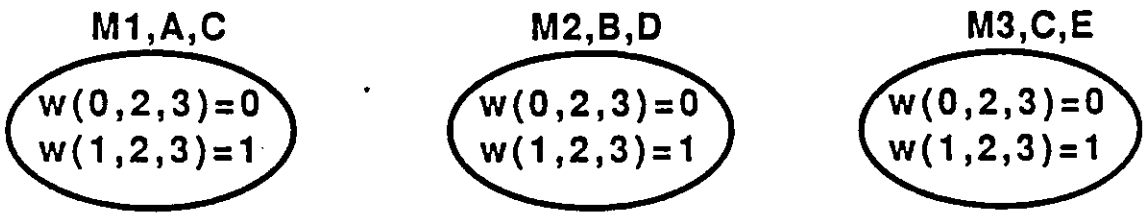


Figure 8

2. **Computing the Weights of X:** The possible values of X are $\{6, e\}$. The constraint between (M_1, A, C) and X are given in Figure 8 where the lines connecting values indicate consistent pairs of values. Since X has only one child node its weights are computed based on it alone and are given in the figure together with the pointers to be traced back later.

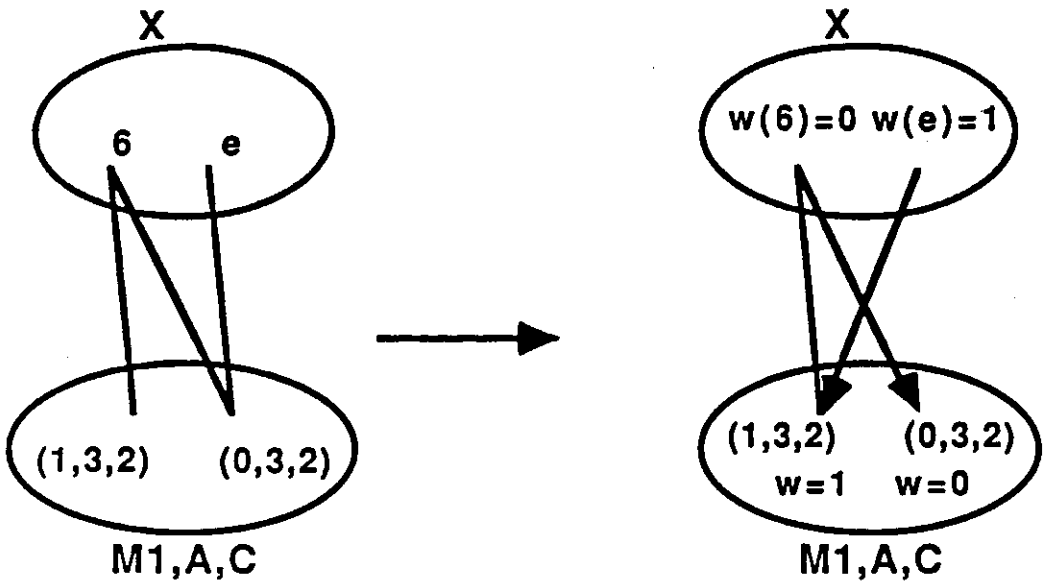
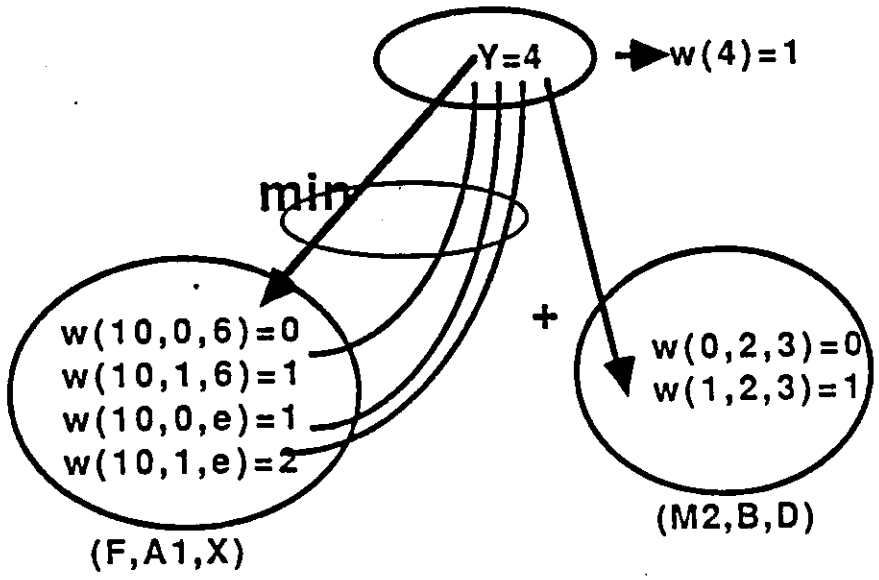


Figure 9.

3. **Computing the Weights of Y:** Figure 10 presents the details of weights calculation for each value of y separately. Y has two children nodes (F, A_1, X) and (M_2, B, D), the first has 4 compound values while the second has just two.



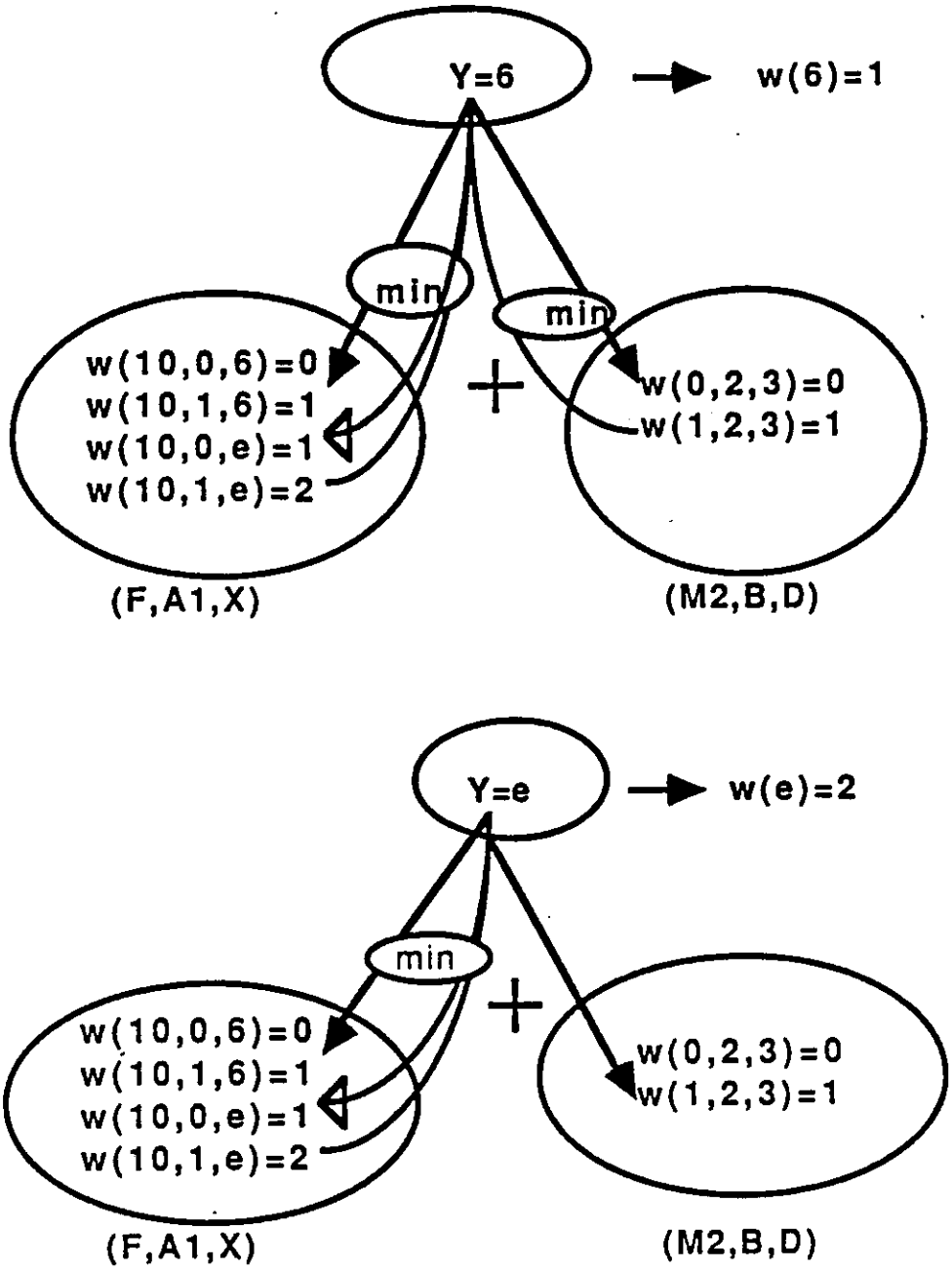


Figure 10.

The final step is to compute the weights associated with G based on the weights associated with Y, A_2, and Z. The possible values and the weights of the compound variable (Y, A_2, Z) and the constraint with variable G are given in Figure 11. Notice that the compound variable has 12 values since Y has three possible values while Z and A_2 have two values each. The variable G has only one value, e.g., the observed value 12.

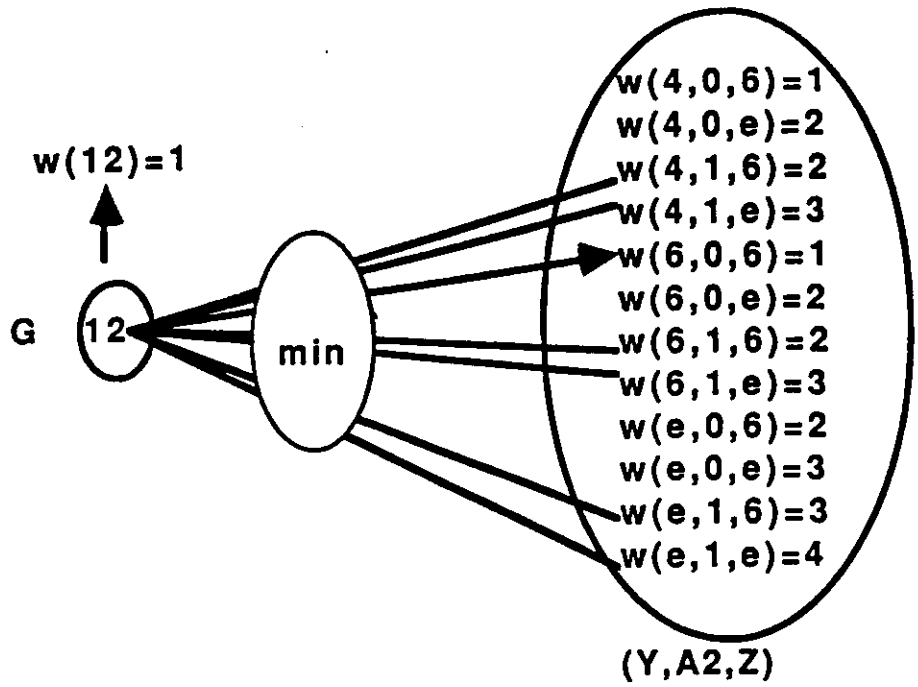


Figure 11.

Tracing back along the pointers, starting from variable G, we obtain two solutions each weighing 1, i.e., each representing one faulty component. One points to A_1 as the faulty component, the other points to M_1. In the case of several solutions we can prefer the one that participates in a higher number extensions.

In a similar way we can derive many other quantities of interest for diagnosis. For example,

1. For each component, find in how many consistent extensions it is faulty
2. Find all minimal diagnoses
3. Among all minimal diagnoses, find those with the highest likelihood.

4. FINDING ALL MINIMAL DIAGNOSES

If we need to identify all minimal diagnoses, the propagation process is different. Going back to the simplest case where all constraints are binary and all variables are component variables, a minimal diagnosis corresponds to a consistent solution in which no subset of "1"-values could be replaced by "0" and still preserve consistency. Considering a tree of binary constraints, each value, v, of a variable, V, is associated with all the **minimal** solutions consistent with it restricted to the subtree rooted in V. This association is implemented by attaching to each value a set **pointer-subsets** to all its neighbors' values from which these minimal partial solutions can be retrieved by following the pointers. We will illustrate the procedure on the simple example of Figure 12. In this example, the variable V has two child nodes with the constraints described explicitly in the graph.

In the first step of the algorithm, each value is assigned the set of all partial solutions among its neighbors, called **neighbor-solutions**, that is consistent with it. Let $S(V, v)$ stand for these set of tuples (see Figure 12), and assume the order of values in the tuples is according to the alphabetical order of variables.

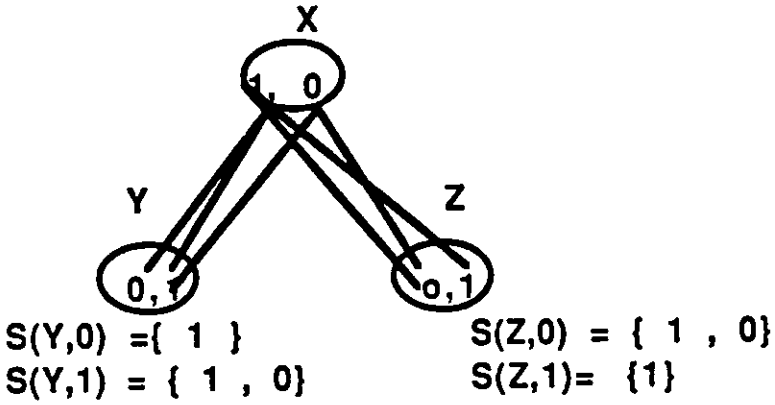


Figure 12.

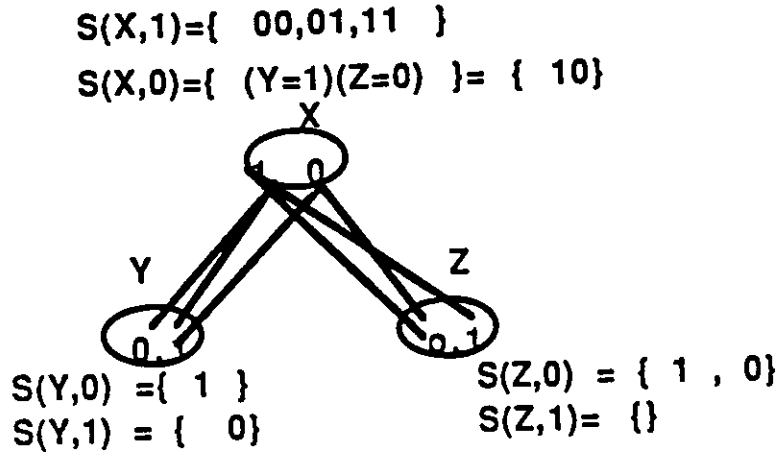


Figure 13.

If for a certain variable the value "1" has the same neighbor-solution as the value "0" this solution could not possibly be extended to a minimal solution and should be eliminated. The second step, therefore, modify the subsets $S(V, 1)$ accordingly (see Figure 13):

$$S(V, 1) \leftarrow S(V, 1) - S(V, 0)$$

These two steps, regarded as the first phase, can be performed distributedly without any particular order control, and in any network of constraints, not necessarily in a tree. Upon its completion, each value is associated with all **minimal neighbor-solutions.**

We can now define a criterion indicating when the neighbor-solutions of two adjacent variables are consistent with each other. Namely, we can view the remaining problem as a new Binary CSP in which each value, v, of V will correspond to a variable, $S(V, v)$, and its values are the set of neighbor-solutions. The constraint between variable $S(V, v)$ and variable $S(U, u)$ is called **handshaking constraints** and is defined as follows: IF subset $S(V, v)$ has the value u of U in one of its neighbor-solution in which the value of V is v, then the variable $S(V, v)$ has to have at least one neighbor-solution in which the value of V is v. Otherwise, we may eliminate the corresponding neighbor-solution from $S(V, v)$. Such local elimination of neighbor-solutions can be regarded as making the constraints of the new problem arc-consistent. The constraint from $S(V, v)$ to $S(U, u)$ is arc-consistent if for any of its neighbor-solutions of $S(V, v)$ that contain the value u of U, there exists in $S(U, u)$ a neighbor-solution with the value v of V.

In Figure 14, the constraints associated with X are made consistent w.r.t. Y and Z. We see that the only neighbor solution of $S(X, 0)$ should not be eliminated since it contains the value $Y = 1$ and indeed, $S(Y, 1)$ contains the value 0 for X, also its value for Z is 0 and $S(Z, 0)$ has the value 0 for X. In the case of $S(X, 1)$ two of its neighbor-solutions : 01, and 11 were eliminated since they contain the value "1" for Z while $S(Z, 1)$ is empty.

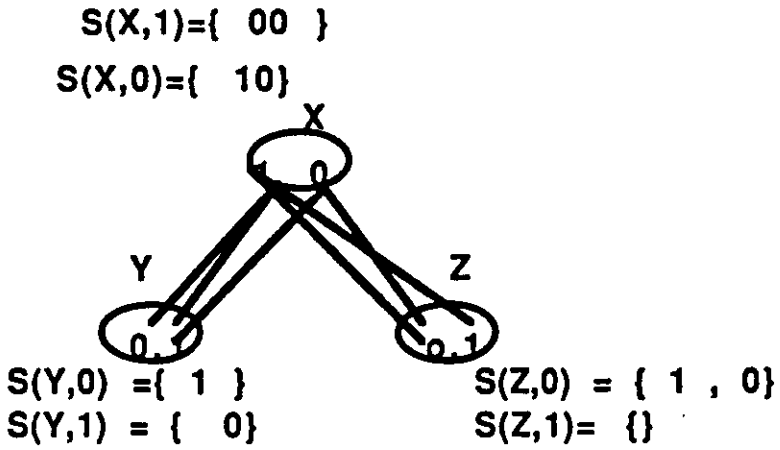


Figure 14.

The second phase of the algorithm performs this local consistency on the directed tree by eliminating neighbor-solutions from the variables while going from the leaves to the roots. At the end of this phase all variables $S(V, v)$ will be locally consistent w.r.t. their child variables.

In the third phase all the minimal solutions are generated by following the pointers created in the previous phases starting at the root node. In the above example only two minimal solutions exist: $\{(X = 0, Y = 1, Z = 0), (X = 1, Y = 0, Z = 0)\}$ which in this case are also the minimum-1's solutions. In Figures 15 and 16 we illustrated a more complex example of the algorithm. Figure 15 displays a CSP problem that has 5 binary-valued variables X_1, X_2, X_3, X_4, X_5 their constraints, and their $S(X, x)$ variables after the completion of the first phase of the algorithm. Figure 16 shows the remaining neighbor-solutions after phase 2.

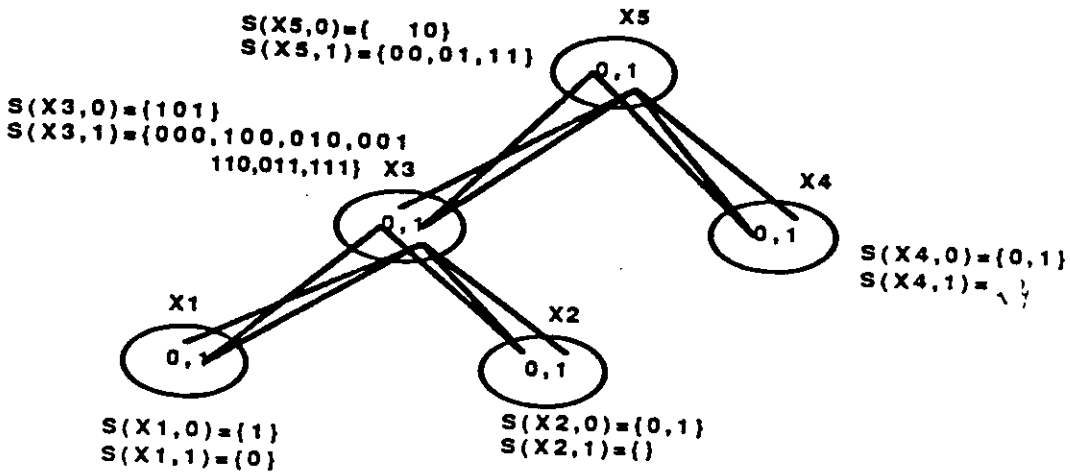


Figure 15.

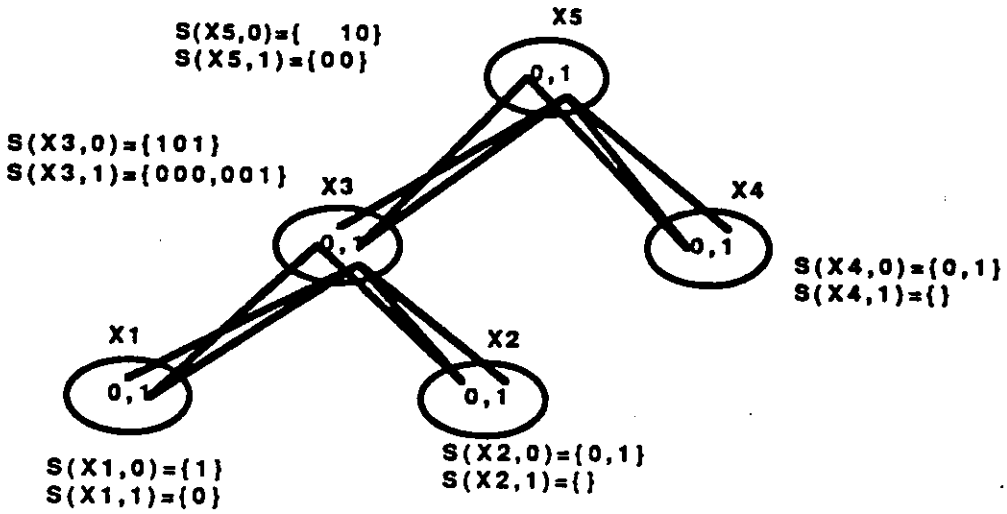


Figure 16.

The two minimal solutions generated in the last forward phase are:

$$(X_1, X_2, X_3, X_4, X_4) = \{00100, 1001\}$$

As in the previous section, the above procedure could be modified to work on non-binary constraints and on solutions in which only the subset of component variables are considered for the minimality criterion.

REFERENCES

[Davis, 1984] Davis, R., "Diagnostic Reasoning Based on Structure and Behavior," *Artificial Intelligence,* Vol. 24, 1984, pp. 347-410.

[de Kleer and Williams, 1986] de Kleer, J., and Williams, B. C., "Reasoning about Multiple-Faults," *Proc. 5th Natl. Conf. on AI (AAAI-86),* Philadelphia, 1986, pp. 132-39.

[Dechter and Pearl, 1987] Dechter, R., and Pearl, J., "Network-Based Heuristics for Constraint-Satisfaction Problems," *Artificial Intelligence,* Vol. 34, No. 1, 1987, pp. 1-38.

[Dechter and Pearl, 1988] Dechter, R., and Pearl, J., "Tree Clustering Schemes for Constraint-Processing," *Proceedings,* AAAI-88, Minneapolis, St. Paul, August 1988, pp. 150-154. To appear in *Artificial Intelligence.*

[Geffner and Pearl, 1987] Geffner, H., and Pearl, J., "A Distributed Approach to Diagnosis," *Technical Report R-66,* Cognitive Systems Laboratory, Unviersity of California, Los Angeles, Short version in *Proc. 3rd IEEE Conf. on AI Applications,* Orlando, FL., 1987, pp. 156-62.

[Genesereth, 1984] Genesereth, M. R., "The Use of Design Descriptions in Automated Diagnosis," *Artificial Intelligence,* Vol. 24, No. 1, 1984, pp. 411-36.

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