

**AN IMPROVED CONSTRAINT-PROPAGATION ALGORITHM
FOR DIAGNOSIS**

**Hector Geffner
Judea Pearl**

**February 1987
CSD-870012**

AN IMPROVED CONSTRAINT-PROPAGATION ALGORITHM FOR DIAGNOSIS* †

Hector Geffner & Judea Pearl

Cognitive Systems Laboratory
UCLA Computer Science Department
Los Angeles, California 90024
judea@locus.ucla.edu (213) 825-3243
hector@locus.ucla.edu (213) 825-7367

ABSTRACT

Diagnosing a system requires the identification of a set of components whose abnormal behavior could explain faulty system behavior. Previously, model-based diagnosis schemes have proceeded through a cycle of *assumptions* → *predictions* → *observations* → *assumptions-adjustment*, where the basic assumptions entail the proper functioning of those components whose failure is not established. Here we propose a scheme in which every component's status is treated as a variable; therefore, predictions covering all possible behavior of the system can be generated. Remarkably, the algorithm exhibits a drastic reduction in complexity for a large family of system-models. Additionally, the intermediate computations provide useful guidance for selecting new tests. The proposed scheme may be considered as either an enhancement of the scheme proposed in [de Kleer, 1986] or an adaptation of the probabilistic propagation scheme proposed in [Pearl, 1986] for the diagnosis of deterministic systems.

† submitted for presentation at the **IJCAI-87 Conference** Milan, Italy August 1986.

Paper Type: long
Topic: Reasoning
Track: Science
Keywords: constraint propagation, diagnostic reasoning

* This work was supported in part by the National Science Foundation, Grant DSR 83-13875.

AN IMPROVED CONSTRAINT-PROPAGATION ALGORITHM FOR DIAGNOSIS*

Hector Geffner & Judea Pearl

Diagnosing a system requires the identification of a set of components whose abnormal behavior could explain faulty system behavior. Previously, model-based diagnosis schemes have proceeded through a cycle of assumptions \rightarrow predictions \rightarrow observations \rightarrow assumptions-adjustment, where the basic assumptions entail the proper functioning of those components whose failure is not established. Here we propose a scheme in which every component's status is treated as a variable; therefore, predictions covering all possible behavior of the system can be generated. Remarkably, the algorithm exhibits a drastic reduction in complexity for a large family of system-models. Additionally, the intermediate computations provide useful guidance for selecting new tests.

The proposed scheme may be considered as either an enhancement of the scheme proposed in [de Kleer, 1986] or an adaptation of the probabilistic propagation scheme proposed in [Pearl, 1986] for the diagnosis of deterministic systems.

I. Introduction

The diagnosis of a system exhibiting abnormal behavior consists of identifying those subsystems whose abnormal behavior *could produce* the manifested behavior. In *model-based diagnosis*, the system is treated as an idealized structure of components whose local behaviors interact to produce overall system behavior. Previous diagnostic schemes have tackled the task by partitioning the problem into two phases. First, making use of a set of observations and *assuming* the proper local behavior of components, *predictions* are generated about the behavior of unobserved points. In the second phase, the *assumptions underlying those predictions* contradicted by observations are identified, and a set of *hypotheses* which "best" explains the manifested system behavior is assembled about the individual component's behavior.

While many proposed schemes fit within this paradigm [Reiter 86, Genesereth 84], the work of [deKleer et al., 1986] has especially focused on algorithms for performing these two tasks efficiently. In their scheme, predictions are generated by constraint propagation [Stallman et al., 1977], while the process

*This work was supported in part by the National Science Foundation Grant DSR 83-13875.

of identifying the (minimal) set(s) of assumptions underlying a contradicted prediction (conflict sets) is facilitated by the ATMS [deKleer 86]. The diagnoses chosen as "best" are those minimal sets of assumptions (candidates) which, if removed, render the model behavior compatible with manifested behavior. These are assembled from the conflict sets by a set-covering algorithm.

The diagnostic algorithm we propose here consists of a single task: Instead of predicting only those behaviors which assume proper functioning of components, the proposed algorithm generates predictions about *all possible* behavior. Since the point value corresponding to a new observation is surely among the set of possible behaviors, the optimal diagnoses that account for the enhanced set of observations will be simply the set of assumptions underlying that value prediction.

The algorithm takes advantage of the fact that there is usually a small set of hypotheses compatible with the current set of observations that will best explain *any single new* observation. The resulting scheme

- diagnoses failures due to multiple faults;
- is incremental; and (in contrast to two-phase approaches)
- *fully exploits the topology of the system model under diagnosis.*

Interestingly enough, the algorithm can be viewed as either an enhancement of the scheme proposed in [de Kleer et al., 1986], a non-probabilistic adaptation of the propagation scheme proposed in [Pearl, 1986] or as a message-passing mechanism for solving constraint satisfaction problems with functional dependencies [Dechter, 1986].

In this paper we first describe the ideas underlying the scheme proposed (Section I), introduce some notation and present the propagation algorithm for models of singly-connected constraint networks (Section II) and (Section III) illustrate its application to the diagnosis of a simple digital circuit. We then extend the propagation scheme properly handle both multiply-connected networks as well as component models with different prior probabilities of failure. In Section V, we summarize the main results.

II. The Proposed Scheme

For the purpose of illustrating the ideas underlying the scheme being proposed, let us first introduce some convenient notation. Let $L(S, O)$ denote a particular labeling that assigns a specific status to

each component of a system S , compatible with a set of observations O . For the time being, we shall assume that the merit of a given diagnosis is determined by the number of faulty components involved; so, every labeling $L(S, O)$ will be assigned a figure of (de)merit denoted by $N(S, O)$, representing the number of S -components labeled "faulty" by $L(S, O)$.

Let us now consider two complex system circuits, S_1 and S_2 , with outputs X_1 and X_2 , and observation sets O_1 and O_2 , respectively, as depicted in Figure 1. Let us also assume that, for each of these circuits, we have computed a set of diagnoses accounting for both their observation sets and *each of the possible values* of their output variables, X_1 and X_2 . In other words, two sets of labelings, $L(S_1, O_1 \cup \{X_1 = x_1\})$ and $L(S_2, O_2 \cup \{X_2 = x_2\})$, are available, each associated with the range of values x_1 and x_2 that variables X_1 and X_2 may take.

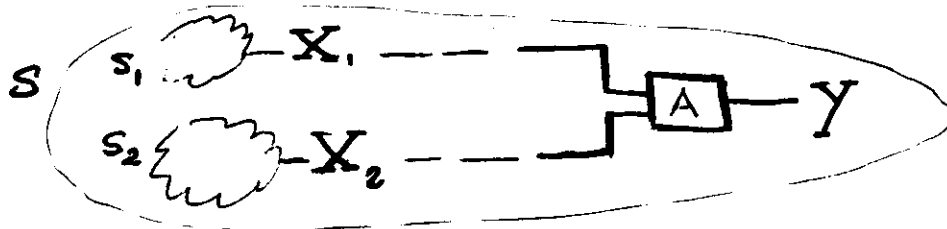


Figure 1

We next consider composite circuits formed by an adder A with output Y and inputs X_1 and X_2 and attempt to find the best labelings over the components of S compatible with the set of observations $O = O_1 \cup O_2$ and each of the possible values y of Y . In other words, we want to determine $L(S, O \cup \{Y = y\})$ for each value y of Y .

The main point to note is that, to determine $L(S, O \cup \{Y = y\})$, we need not reconsider all the combinations of labelings over the individual circuits S_1 and S_2 ; $L(S, O \cup \{Y = y\})$ is made up of exactly those labelings that were *already* found optimal over S_1 and S_2 for some values of X_1 and X_2 . In particular, assuming that the adder A is working properly (denoted by $A = ok$), the weight associated with the optimal labeling will be given by:

$$N_{A=ok}(S, O \cup \{Y = y\}) = \min_{x_1+x_2=y} [N(S_1, O_1 \cup \{X_1 = x_1\}) + N(S_2, O_2 \cup \{X_2 = x_2\})], \quad (1)$$

where x_1 and x_2 range over the possible values of X_1 and X_2 , respectively. Thus, if the minimum is achieved at values x_1^* and x_2^* , the optimal labeling for $Y = y$ under the assumption $A = ok$, can be constructed from:

$$L_{A=ok}(S, O \cup \{Y = y\}) = L(S_1, O_1 \cup \{X_1 = x_1^*\}) \cup L(S_2, O_2 \cup \{X_2 = x_2^*\}) \cup \{A = ok\}. \quad (2)$$

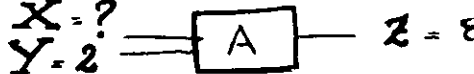
To find the overall optimal labeling, we must also include the possibility that A is not working properly, denoted by $A = \neg ok$, yielding:

$$N_{A=\neg ok}(S, O \cup \{Y = y\}) = 1 + \min_{x_1} N(S_1, O_1 \cup \{X_1 = x_1\}) + \min_{x_2} N(S_2, O_2 \cup \{X_2 = x_2\}), \quad (3)$$

The constant 1 stands for the penalty associated with the faulty component $A = \neg ok$ which, in turn, removes the constraint between the values of X_1 and X_2 . (The two rightmost terms of (3) were precomputed on the individual circuits S_1 and S_2 .)

The overall best labeling(s) compatible with $Y = y$ can now be obtained simply by comparing the weight computed from (1) with the one computed from (3) and choosing the labeling(s) corresponding to the lowest one. If $Y = y$ is now observed, $L(S, O \cup \{Y = y\})$ would emerge as the labeling defining the best diagnosis.

A question remains, however, of how to minimize over sets of values that, in principle, may have infinite cardinality. Its solution is based on the fact that there will be labelings which are compatible with *all* the values a variable can take. For example,



$$\begin{array}{l} L(\{A\}, \{Y=2, Z=8\} \cup \{X=6\}) = \{A=ok\} \\ L(\{A\}, \{Y=2, Z=8\} \cup \{X \neq 6\}) = \{A=\neg ok\} \end{array}$$

Figure 2 - Best Labelings for X

in the simple adder circuit depicted in Figure 2, we have only two labelings, $\{A = ok\}$ and $\{A = \neg ok\}$. The best labeling compatible with $X = 6$ is $\{A = ok\}$ while, for any other value of X , $\{A = \neg ok\}$ is the best and, in fact, the *only* compatible labeling. Moreover, $\{A = \neg ok\}$ is the best labeling compatible with *any value* of X except $X = 6$. Therefore, certain subsets of X values (e.g., $X \neq 6$) will be treated as special x -values, denoted by Ω_X and will be assigned labeling as though they were singletons. The introduction of these "special" values will allow us to keep the state representation of the problem concise; the algorithm, however, should behave as though an explicit representation were used for each value in the subset.

This example illustrates two important properties on which our algorithm is based: first, that the problem of determining the optimal labeling for a variable can be solved from information associated with neighboring variables, making feasible a distributed diagnostic inference engine in a form of constraint propagation; and secondly, that it is possible to predict all the possible model behaviors by handling a finite (and usually small) set of "explanations," allowing the constraint propagation algorithm to accomplish a global identification task without appealing to non-local set-covering procedures.

A critical assumption implicit in the example of Figure 1 is that circuits S_1 and S_2 are only connected via A or, more generally, that the constraint network induced by the composite circuit S is singly-connected. In Section IV this assumption will be relaxed and we shall generalize the algorithm to deal with multiply-connected constraint networks. These extensions will provide interesting points of comparison with [de Kleer, 1986].

Notation

As we stated before, the global behavior of the system-model is characterized by a pattern of interactions among a number of local behaviors that, in turn, correspond to the constraints induced by the components on their inputs and outputs. We assume that a component C_j can be in either of two states: it is working properly, denoted by $C_j = ok$, or it is failing, denoted by $C_j = \neg ok$. Since we are going to treat components as any other variable in the model, we choose to draw them as nodes in Figure 3 rather than depict them as blocks as in Figures 1 and 2:

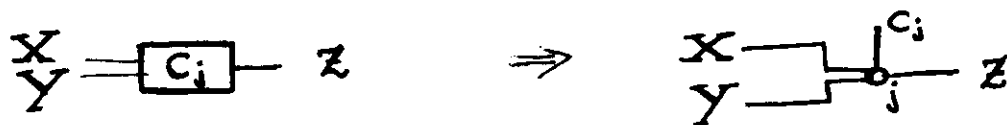


Figure 3 - Components as Nodes of a Constraint Network

where $C_j \in \{ok, \neg ok\}$ stands for the name of the variable corresponding to the component status and j is the unidirectional constraint among the component input(s), its output(s) and its status. The resulting network is composed of

- nodes (or variables), denoted by upper-case letters
(X, Y, W, \dots , for observable system points and C_i, C_j, \dots , for components);
- constraints, denoted by small letters (i, j, \dots); and
- links connecting nodes to constraints.

The set of nodes linked to a constraint k will be denoted by P^k . A more useful set is $P_X^k = P^k - X$, which reads as the *port k of node X* , and stands for the set of all other nodes different than X linked to constraint k . $C(X)$ will refer to the set of constraints linked to X . Lower-case letters $x, y, w, \dots, c^i, c^j, \dots$, will refer to the values of variables X, Y, W, \dots, C^i, C^j , and subindices x_1, x_2, \dots , will be used to differentiate among the values of a single variable. We will extend this convention to refer to p_X^k as the values of the variables in the port P_X^k . For example, the port P_X^j in Figure 3 is the set of variables $\{Y, Z, C_j\}$, while P_X^j stands for their associated values $\{y, z, c^j\}$. As stated before, the "value" Ω_Y will serve as a place holder for all those values of node Y that are not explicitly represented.

In constraint-propagation algorithms such as the one used here, the ports of a variable contain enough information for the variable to update its state. This updating is usually done incrementally, whenever a new state of a variable can be deduced from a neighbor constraint and its associated ports. We shall refer to such inferences as *messages*, denoted by $m(P_X^k \rightarrow X)$, where node X is the recipient and port P_X^k the origin. The specification of what information these messages carry, how they are generated and how they are combined is described in the following subsections.

The Algorithm

Messages: Messages $m(P_X^k \rightarrow X)$ are built from a vector of sub-messages $m(P_X^k \rightarrow x_i)$, where the x_i 's refer to values X may take. Each of these submessages is composed of three fields: x_i , $N^k(x_i)$ and $S^k(x_i)$, where the last parameter stands for the set of values p_X^k of P_X^k which would "best" support the potential observation $X = x_i$ and the second one for the weight associated with such support (i.e., the minimum number of faulty components, in the subnetwork behind the port P_X^k , necessary to account for $X = x$).

In short, a message $m(P_X^k \rightarrow X)$ can be expressed as:

$$m(P_X^k \rightarrow X) = \bigcup_i m(P_X^k \rightarrow x_i),$$

while $m(P_X^k \rightarrow x_i)$ assumes the form:

$$m(P_X^k \rightarrow x_i) = (x_i, N^k(x_i), S^k(x_i)).$$

If $N^k(x_i) = \infty$, for some value x_i of X , we simply remove x_i from the set of values X that need(s) to be considered.

Message Combination: The first requirement for message combination is that every node save the last message received from each of its ports. New messages override old messages from any one port. >From this set of messages, every node computes its state $m(X)$. That is, if we denote the combination of messages by the \cap , then (for each of its values x_i) X computes the states:

$$\begin{aligned} m(x_i) &= \bigcap_{k \in C(X)} m(P_X^k \rightarrow x_i) \\ &= (x_i, N(x_i), S(x_i)), \end{aligned} \quad (4)$$

where

$$N(x_i) = \sum_{k \in C(X)} N^k(x_i) \quad (5)$$

is called the *weight* associated with x_i and represents the minimum number of faulty components necessary to account for the current set of observations and the instantiation $X = x_i$, and

$$S(x_i) = \bigcup_{k \in C(X)} S^k(x_i), \quad (6)$$

the *support* of x_i , encodes the state(s) of the neighborhood of X that will best account for all the observations and the instantiation $X = x_i$.

Upon observing $X = x_i$, the set of optimal diagnoses can be easily retrieved by tracing the support associated with x_i . $N(x_i)$ represents the number of faults involved in any of these hypothetical diagnoses.

Message Assembling: To illustrate how messages are computed, let Y_1, Y_2, \dots, Y_n stand for the variables of P_X^k . The parameters of the message $m(P_X^k \rightarrow x_i)$ are computed from:

$$N^k(x_i) = \min_{\substack{y_1, y_2, \dots, y_n \\ \text{subject to } C^k(x_i, y_1, \dots, y_n)}} \sum_{j=1}^n [N(y_j) - N^k(y_j)], \quad (7)$$

and

$$S^k(x_i) = \min_{\substack{y_1, y_2, \dots, y_n \\ \text{subject to } C^k(x_i, y_1, \dots, y_n)}}^{-1} \sum_{j=1}^n [N(y_j) - N^k(y_j)], \quad (8)$$

where the y_j 's range over the values of Y_j , and $C^k(x_i, y_1, y_2, \dots, y_n)$ denotes a predicate indicating the compatibility of its arguments according to constraint k . The subtraction of the component $N^k(y)$ from the weight $N(y)$ amounts to subtracting the contribution of X to $N(y)$, and procedures counting any fault more than once.

Initialization: The propagation algorithm requires that nodes be properly initialized. A node, C , representing a component status will be initialized to:

$$m(C) = ((ok, 0, ()) \\ (\neg ok, 1, ())) ,$$

while any other variable Y will have an initial state:

$$m(Y) = (\Omega_Y, 0, ()) .$$

Observations: The observation $X = x$ is codified as a message from a virtual port O_X of X :

$$m(O_X \rightarrow X) = ((x, 0, ())), \quad (9)$$

where absence of sub-messages corresponding to other values of X is interpreted as $N(X = x_i) = \infty$ for any $x_i \neq x$. For the purpose of applying formulas (5)-(8), for any observable node X , virtual ports O_X are taken to be members of the set $C(X)$ of ports linked to X .

Control of the Propagation Process: While an orderly and incremental propagation scheme would be the most efficient implementation in serial machines, the algorithm can also work under distributed control, in which each node inspects the state of its ports at its own discretion. The final state reached at equilibrium would be the same.

We now proceed to illustrate the working of the algorithm on a simple circuit discussed in [deKleer 86], [Genesereth 84] and [Davis 84].

III. Example

Let us consider the circuit depicted in Figure 4: Components M_1 , M_2 and M_3 are multipliers, while A_1 and A_2 are adders. The former will correspond to constraints C_1 , C_2 and C_3 , and the latter to constraints C_4 and C_5 . Initially, all inputs are known and propagated through the rest of the network, generating the pattern of messages shown in Figure 5(a). Figure 5(b) displays the states of nodes X and F . Let us assume, now, that $F = 10$ is observed. From Eq.(9), a message $m(O_F \rightarrow F) = ((10, 0, ()))$ is posted at F so that the resulting state of F , according to Eqs. (4)-(6), becomes:

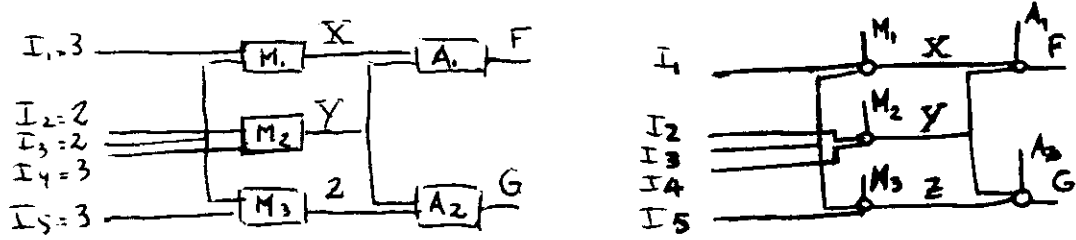


Figure 4 - A Simple Circuit and Its Constraint Network

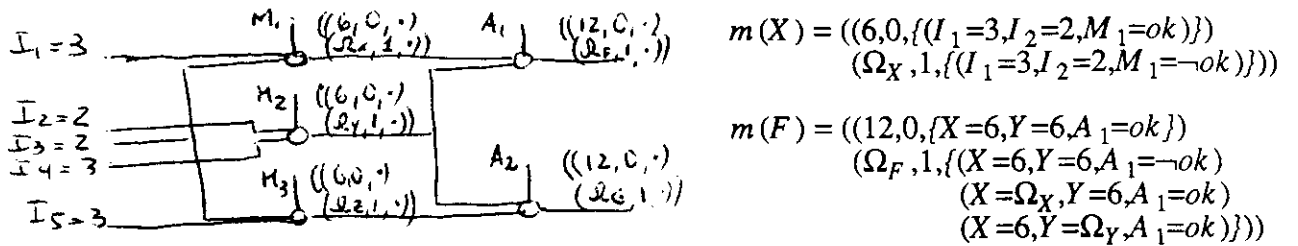


Figure 5 - (a) Pattern of Messages Generated by the Input (b) States of X and F

$$\begin{aligned}
 m(F) &= m(P_F^4 \rightarrow F) \cap m(O_F \rightarrow F) \\
 &= ((12, 0, \{X=6, Y=6, A_1=ok\}) \quad \cap \quad ((10, 0, ())) , \\
 &\quad (\Omega_F, 1, \{(X=\Omega_X, Y=6, A_1=ok) \\
 &\quad (X=6, Y=\Omega_Y, A_1=ok) \\
 &\quad (X=6, Y=6, A_1=\neg ok)\}))
 \end{aligned}$$

Since $F = 10$ was not explicitly represented in the previous state of F , the best "prediction" of the observed value is the one associated with Ω_F . Furthermore, since all values of F other than 10 are ruled out by the observation, we obtain:

$$\begin{aligned}
 m(F) &= ((10, 1, \{(X=\Omega_X, Y=6, A_1=ok) \\
 &\quad (X=6, Y=\Omega_Y, A_1=ok) \\
 &\quad (X=6, Y=6, A_1=\neg ok)\})) .
 \end{aligned}$$

Since the observed value $F = 10$, has three unit-weight supports, we can immediately assert that there are at least three different, equally meritorious, diagnoses accounting for the enhanced set of observations, each of which is encumbered by a single faulty component (A_1 , M_1 or M_2). We can uncover the identity of the faulty components involved in these diagnoses by simply tracing their respective supports. For instance, underlying the support $(X=\Omega_X, Y=6, A_1=ok)$ of $F = 10$, we find a single fault $C_1 = \neg ok$ associated, in turn, with the support of $X = \Omega_X$.

The new message $m(O_F \rightarrow F)$ also causes a pattern of additional messages to propagate throughout the network. However, this propagation is necessary not for finding the diagnoses, (these can be found by tracing the existing supports) but for preparing the support lists of the network to accommodate new observations. Figure 6 illustrates the computation of one of these messages, $m(P_X^4 \rightarrow X)$.

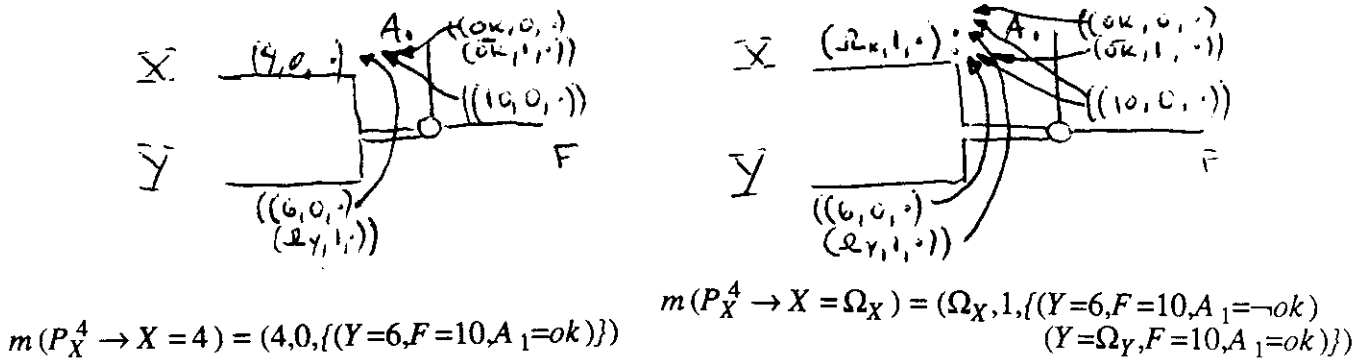


Figure 6 - Computing (a) $m(P_X^4 \rightarrow X = 4)$ (b) $m(P_X^4 \rightarrow X = \Omega_X)$ after Observing $F = 10$

Note that, for the purpose of computing $N^4(X)$, the weights associated with the variables in P_X^4 (i.e., Y , F and A_1) do not include the contributions of messages originating from P^4 , as specified in Eq.(7).

Figure 7(a) illustrates the new pattern of messages resulting after the observation $F = 10$. Each message is depicted next to its originating port; additionally, any node that has receiving messages from more than one port computes its new state according to Eq.(4). Figure 7(b) displays the updated state of nodes A_1 and G .

Assume that $G = 12$ is now observed. By simply tracing the supports associated with this value of G , the optimal diagnoses accounting for this and all previous observations is retrieved. Since the only support of $G = 12$ is $(Y=6,Z=6,A_2=ok)$, and the support of $Z=6$ involves no fault, the two best explanations are found encoded in the support of $Y=6$, namely, either $\{M_1 = \neg ok\}$ or $\{A_1 = \neg ok\}$, respectively.

The computations required by the algorithm also provide information useful in selecting test procedures. Not only do the resulting data structures encode the best diagnoses accounting for any potential observation at any point in the circuit, but they also encode, through the supports attached to component nodes, the network state most compatible with any component status. This turns out to be especially useful when we want to select test points to discriminate between different hypotheses.

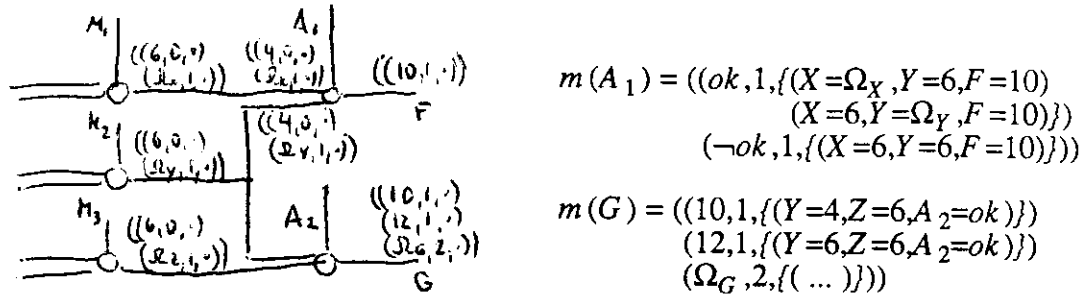


Figure 7(a) - Pattern of Messages after $F = 10$ (b) - New States of A_1 and G

Let us also note here that the scheme proposed does not guarantee finding all irredundant diagnoses [Peng 86], but only those with a *minimum* number of faults, i.e., the *optimal* diagnoses. A method of enumerating the former is described in [Dechter 1986].

IV. Enhancements

In this section we will discuss some modifications to the scheme introduced in Section II to enhance the types of models with which the algorithm can deal. We first discuss how to extend the message-passing algorithm to handle constraint networks containing loops and then propose a slight modification to accommodate models with specified component-failure probabilities.

Handling Constraint Networks with Loops

The strength of the proposed scheme lies in its ability to decompose global optimization problems into local ones. To achieve this, we assumed that messages received by a node from different ports carried independent information, i.e., do not emanate from common observations. It is easy to find systems in which this assumption is violated, as in the example of Figure 8. Clearly, the information carried by both inputs of C_4 will depend on the status of C_1 . Thus, knowing $m(Y)$ and $m(W)$ no longer suffices to compute $m(Z)$. The reason is simply that, for some pairs of values y and w of Y and W , respectively, $m(y)$ and $m(w)$ might involve a common set of faulty components; so, the weight associated with a value z of Z supported by y and w will not necessarily be the sum of the weights of its supporters. In addition, $m(y)$ and $m(w)$ may involve incompatible labelings of X ; so, we must ensure that they do not appear in a same support set.

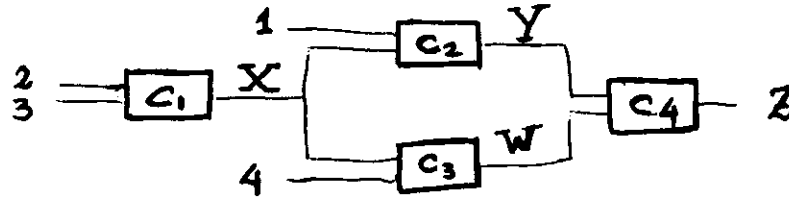


Figure 8 - A Circuit Containing a Loop.

The solution that we will pursue is not new [Pearl 86.a] [Dechter & Pearl 86] and rests on the idea of treating a "loopy" network as a family of singly-connected networks in which some of the variables (usually those corresponding to a cycle cutset and referred to here as "assumption" nodes) have been assigned a fixed value (Figure 9). A fixed-value node can be partitioned into a set of identical nodes, each connected to one and only one of its original ports while still preserving the overall behavior.

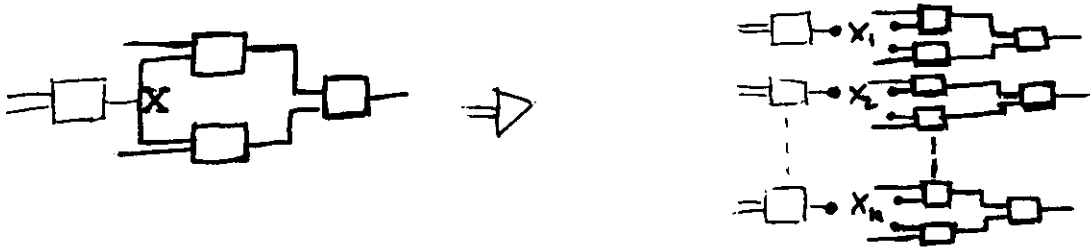


Figure 9 - A Loopy Network Treated as a Family of Singly-Connected Networks.

To illustrate the modifications needed to handle loops, let us consider (Figure 10) constraint network S where the instantiation of assumption node A decomposes it into a pair of singly-connected networks, S_1 and S_2 .

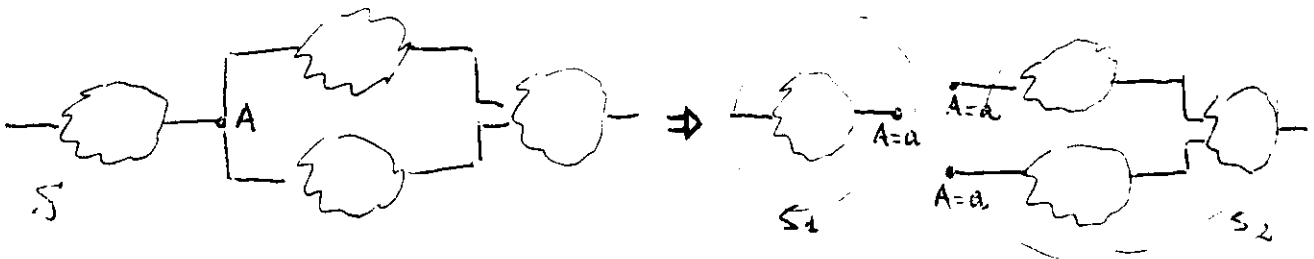


Figure 10 - Breaking Loops by Introducing Assumptions.

Let $O = O_1 \cup O_2$ represent the set of observations gathered and O_j correspond to observations in sub-network S_j . Recalling the notation introduced in Section II, $N(S, O)$ stands for the number of faulty

components needed to account for the set of observations O in system S , while $N(y)$ is simply shorthand for $N(S, O \cup \{Y=y\})$. We shall now use the abbreviations:

$$W_a(x) = N(S_i, O_i \cup \{A=a, X=x\}) \quad \text{for } X \in S_i, \quad (10)$$

$$N_a(x) = N(S, O \cup \{A=a, X=x\}). \quad (11)$$

and

$$N^j(a) = N(S_j, O_j \cup \{A = a\}) \quad (12)$$

By applying the algorithm previously described and assuming $A = a$, we are in a position to compute $W_a(x)$, for any node $X \in S_i$, i.e., the minimum number of faults in S_i , needed to account for the observations $O_i \cup \{A=a, X=x\}$. We are interested, however, in computing the weight $N(X=x) = N(S, O \cup \{X=x\})$, i.e., the total number of faults in S needed to account for the entire set of observations $O \cup \{X=x\}$ *without any assumption about the value of A* . This can be obtained by writing:

$$\begin{aligned} N(x) &= \min_a N_a(x) \\ &= \min_a N(S, O \cup \{A = a, X = x\}) \\ &= \min_a \left[N(S_i, O_i \cup \{X = x\} \cup \{A = a\}) + \sum_{j \neq i} N(S_j, O_j \cup \{A = a\}) \right] \\ &= \min_a \left[W_a(x) + N(a) - N^i(a) \right] \end{aligned} \quad (13)$$

where

$$N(a) = \sum_j N^j(a) \quad (14)$$

Thus, if the minimum is achieved at $A = a^*$, the optimal set of diagnoses will be obtained from the supports associated with $X=x$ under the assumption $A = a^*$ and from those associated with $A = a^*$ in itself, unconditioned by $X = x$. From Eq.(13) it is clear that the computation of $N(y)$ for any node Y can be reduced to the computation of $W_a(x)$ for every value a of A and every value x of the nodes X in S . These weights can, in turn, be computed by executing the procedure discussed in Section II. Independently, for each of the singly connected sub-networks S_i , we still need, though, to provide a mechanism to

keep track of the weight $N^i(a)$ associated with that subnet (S_i) which contains X .

The Algorithm

Since each instantiation of the cutset variables identifies a set of singly-connected networks, let us tag every message that is supposed to propagate in that set with the identity of such an instantiation. We shall refer to the instantiated nodes as *assumption nodes*, to their instantiations as *assumptions* or *tags*, and to each of the virtual singly-connected networks corresponding to particular tags as *contexts*. The manner in which tags are treated in this modified scheme bears a strong resemblance to the way assumptions are treated in the ATMS [deKleer 86].

As with node values, the message-passing algorithm will not explicitly represent each of the possible contexts but will appeal, instead, to the special Ω -values introduced in Section II. Now, however, tags containing Ω -values will be required to specify the range of values for which any Ω -value stands. For that purpose, to do this, we will use $\Omega(x_1, \dots, x_n)$ as the place holder for values of X other than x_1, \dots, x_n . This will render a tag containing the instantiation $X = x_i$ *incompatible* with any tag containing the instantiation $X = \Omega(\dots, x_i, \dots)$. The Ω -values assigned to non-assumption nodes are not required to keep this extra information explicit; so, for them, we preserve our previous, simpler notation.

Messages will now be formatted as:

$$m(P_X^k \rightarrow X) = \bigcup_i m_a(P_X^k \rightarrow x_i)$$

where a represents the tag of the message, and each submessage $m_a(P_X^k \rightarrow x)$ contains five fields:

$$m_a(P_X^k \rightarrow x) = (x, W_a^k(x), S_a^k(x), a, N_X^k(a)),$$

where

$W_a^k(x)$ represents the *component* of $W_a(x)$ originated in the k -th port of X ;

$S_a^k(x)$ denotes its underlying *support*; and

$N_X^k(a)$ stands for the *contribution* $N^i(a)$ to $N(a)$, for $X \in S_i$, as received from the k -th port.

Note that $N_X^k(a)$ does not depend on the value of X . We have included it in the sub-messages only to simplify the local computations.

The state of a node is computed by combining all messages with compatible tags received by the node from its different ports. For a value x of node X , the resulting state $m(x)$ is given by:

$$\begin{aligned} m_a(x) &= \bigcap_{\substack{i \in C(X) \\ \cap b_i=a}} m_{b_i}(P_X^i \rightarrow x) , \\ &= (x, W_a(x), S_a(x), a, N_X(a)) , \end{aligned} \quad (15)$$

where $W_a(x)$ and $S_a(x)$ are computed as usual, within the context defined by each a , while the weight $N_X(a)$ is computed from:

$$N_X(a) = \max_i N_X^i(a) . \quad (16)$$

Since $X \in S_i$, $N_X(a)$ is synonymous with $N^i(a)$, and we shall use the two notations interchangeably. Messages are now assembled as in the case of singly-connected networks, with the additional constraints imposed by the tags.

When a new observation, $Y = y$, is obtained, the optimal weight can be obtained according to Eq.(13):

$$\begin{aligned} N(y) &= \min_a N_a(y) \\ &= \min_a [N(a) - N_Y(a) + W_a(y)] . \end{aligned} \quad (17)$$

If $Y \in S_i$, and the minimum is achieved at $A = a^*$, the set of optimal diagnoses can be obtained by tracing the supports $S_{a^*}(y)$ and $S^j(a^*)$, for $j \neq i$. Additionally, the message posted by the observation $Y = y$ will have the form:

$$m_a(O_Y \rightarrow Y) = ((y, 0, (), a, W_a(y))) , \quad (18)$$

where the last component amounts to setting $N^i(a)$ to the current value of $W_a(y)$, as specified by Eqs.(10) and (12).

The complexity of the resulting algorithm depends on the topology of the constraint network. If X_1, X_2, \dots, X_n are the assumption nodes, the worst-case complexity will be attained when all the assumption combinations must be considered, yielding a total number of (tagged) messages proportional to:

$$\prod_i |X_i| ,$$

where $|X_i|$ refers to the number of values explicitly represented at node X_i . On the other hand, if loops

do not interact, a single assumption for each loop suffices, and the number of messages is reduced to:

$$\sum_i |X_i| .$$

Figure 11 shows examples of each of these cases.

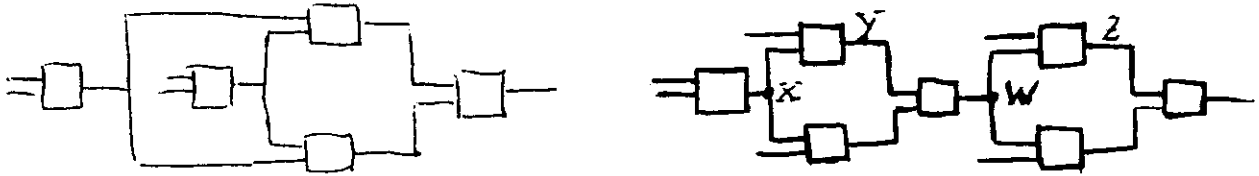


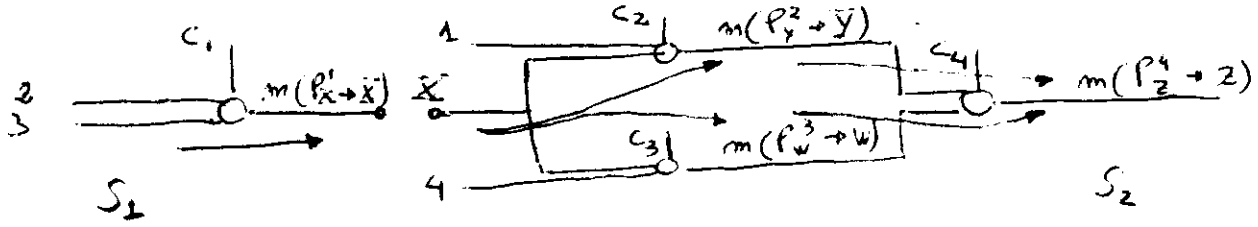
Figure 11 - (a) Complex Loopy Network (b) Simple Loopy Network

To take advantage of the topology of the network, a preprocessing step should both select the assumption nodes as well as delineate their scopes. In Figure 11(b), for instance, to decouple the information carried by the variables in the ports of Y , it is sufficient to instantiate only node X . Similarly, node Z requires only that node W be instantiated, while tagging $m(Z)$ with X will unnecessarily increase the number of messages propagated.

Another problem generated by the presence of loops in constraint networks is the inability of unaided local constraint-propagation methods to enumerate, in advance, all the distinguished instantiations of the cutset variables. For instance, if we regard the components in Figure 8 as adders, and we happen to observe $Z=13$ instead of the predicted $Z=15$, the only value of X compatible with $C_1 = \text{ok}$ is $X = 4$. To arrive at this conclusion, we need either to solve a linear equation $2 \cdot X + 5 = 13X$ or to step sequentially through all the values in the domain of X . One way to obtain these solutions would be to permit the engine to propagate symbolic values [Stallman 77]. This approach seems suitable for implementation in the scheme proposed here and corresponds to viewing the Ω -values of assumption nodes as symbolic values.

Example

To illustrate the extensions discussed in this section, let us consider the diagnosis of the four-adder circuit depicted in Figure 8. To render the constraint network singly-connected, it is sufficient to select X as an assumption node which breaks the circuit in two halves, S_1 and S_2 , as in Figure 12. Propagating the inputs yields the pattern of messages shown in Figure 12, where the middle field, corresponding to message support, was omitted for clarity. Within S_2 , messages have been split into two sets (depicted in brackets), corresponding to each of the assumptions: $X = 5$ and $X = \Omega_X(5)$ ($X \neq 5$). Within



$$m(P_X^1 \rightarrow X) = ([(5, 0, X=5, 0)] \\ [(\Omega_X(5), 0, X=\Omega_X(5), 1)])$$

$$m(P_W^3 \rightarrow W) = ([(9, 0, X=5, 0) \\ (\Omega_W, 1, X=5, 0)] \\ [(\Omega_X(5) + 4, 0, X=\Omega_X(5), 0) \\ (\Omega_W, 1, X=\Omega_X(5), 0)])$$

$$m(P_Y^2 \rightarrow Y) = ([(6, 0, X=5, 0) \\ (\Omega_Y, 1, X=5, 0)] \\ [(\Omega_X(5) + 1, 0, X=\Omega_X(5), 0) \\ (\Omega_Y, 1, X=\Omega_X(5), 0)])$$

$$m(P_Z^4 \rightarrow Z) = ([(15, 0, X=5, 0) \\ (\Omega_Z, 1, X=5, 0)] \\ [(2 \cdot \Omega_X(5) + 5, 0, X=\Omega_X(5), 0) \\ (\Omega_Z, 1, X=\Omega_X(5), 0)])$$

Figure 12 - Pattern of Messages Generated by the Input

each of the sub-networks delimited by X , the second field $W_x(V)$ denotes the number of faults (in the subnetwork containing V) needed to account for the observations and the assumption $X=x$. Note that the computations in S_2 , in the context $X=5$, proceed as if S_2 were the entire network and $X=5$ were the input. The difference with the singly-connected case is essentially that messages propagated in this context carry, besides the previous measures and the identifying tag, a weight $N^2(x)$ that quantifies the context itself (the last field). On the other hand, $\Omega_X(5)$ is propagated as a symbolic value, and its argument 5 encodes the range of X values for which it stands (i.e., any value except 5). The weights are combined according to Eq.(9), so that no single fault is counted more than once.

Let us assume now that $Z=13$ is observed. According to Eq.(18), the set of messages that will be posted at Z will have the form: $m_a(O_Z \rightarrow Z) = ((z, 0, (), a, W_a(Z=13)))$, where a is the corresponding tag. For the tag $X=5$, we have $W_{X=5}(Z=13)=0$, which is retrieved from $W_{X=5}(Z=\Omega_Z)$. For $X=\Omega_X(5)$, the requirement that $2 \cdot \Omega_X(5) + 5 = 13$ yields (by simple manipulation) the new tag $X=4$, with an associated weight $W_{X=4}(Z=13)=0$. For values of X other than 4 and 5, i.e., $\Omega_X(4,5)$, we have $W_{X=\Omega_X(4,5)}(Z=13) = W_{X=\Omega_X(5)}(Z=\Omega_Z) = 1$, so the resulting observation message posted becomes:

$$m(O_Z \rightarrow Z) = ([(13, 0, (), X=5, 1)] \\ [(13, 0, (), X=4, 0)])$$

$$[(13,0,(),X=\Omega_X(4,5),1)] .$$

The optimal weight $N(Z=13)$, as obtained from Eq.(15) is:

$$N(Z=13) = \min_x [N(x) - N_Z(x) + W_x(Z=13)] .$$

Since $N(X=5)=0$, $N(X=\Omega_X(5))=1$, while $N^2(X=5)=N^2(X=\Omega_X(5))=0$, the minimum is achieved at both $X=5$ and $X=4$, rendering $N(Z=13)=1$. At this point, the optimal diagnosis can be obtained from the supports $\{S_{X=5}(Z=13) \cup S^1(X=5)\}$, and from $\{S_{X=4}(Z=13) \cup S^1(X=4)\}$.

This observation generates a new pattern of messages to be propagated through the rest of the network. The computation of the message $m(P_Y^4 \rightarrow Y)$ is illustrated in Figure 13,

$$\begin{array}{l}
 m(P_Y^4 \rightarrow Y) = ([(4,0,X=5,1) \\
 \quad (\Omega_Y,1,X=5,1)] \\
 \quad [(5,0,X=4,0) \\
 \quad \quad (\Omega_Y,1,X=4,0)] \\
 \quad [(9 - \Omega_X(4,5), 0, X = \Omega_X(4,5), 0) \\
 \quad \quad (\Omega_Y,1,X=\Omega_X(4,5),0)])
 \end{array}
 \begin{array}{l}
 \longleftarrow m(C_4) = ((ok,0) \\
 \quad \quad \quad (\neg ok,1)) \\
 \longleftarrow m(O_Z \rightarrow Z) = ([(13,0,X=5,1) \\
 \quad \quad \quad [(13,0,X=4,0) \\
 \quad \quad \quad [(13,0,X=\Omega_X(4,5),1)]]])
 \end{array}$$

$$\begin{array}{l}
 m(P_W^3 \rightarrow W) = ([(9,0,X=5,0) \\
 \quad \quad \quad (\Omega_W,1,X=5,0)] \\
 \quad \quad \quad [(\Omega_X(5) + 4,0,X=\Omega_X(5),0) \\
 \quad \quad \quad (\Omega_W,1,X=\Omega_X(5),0)])
 \end{array}$$

Figure 13 - Computing $m(P_Y^4 \rightarrow Y)$ after Observing $Z=13$.

and the new state of Y , $m(Y)$, is computed as illustrated in Figure 14. If, now, $Y=5$ is observed, the optimal diagnosis weight is obtained by minimizing $N_x(Y=5)$ over x , according to Eq.(17), yielding the weight $N(Y=5) = 1$. That is, there exists at least one diagnosis involving a single faulty component that accounts for all the observations. While we did not show the supports in the previous computations, it is easy to show that the single optimal diagnosis obtained by tracing the supports, $\{S_{X=4}(Y=5) \cup S^1(X=4)\}$, corresponds to $\{C_1 = \neg ok\}$.

$$\begin{aligned}
m(Y) &= m(P_Y^2 \rightarrow Y) \cap m(P_Y^4 \rightarrow Y) \\
&= ([(6, 0, X=5, 0) \\
&\quad (\Omega_Y, 1, X=5, 0)] \\
&\quad [(\Omega_X(5) + 1, 0, X=\Omega_X(5), 0) \\
&\quad (\Omega_Y, 1, X=\Omega_X(5), 0)]) \\
&\quad ([(4, 0, X=5, 1) \\
&\quad (\Omega_Y, 1, X=5, 1)] \\
&\quad [(5, 0, X=4, 0) \\
&\quad (\Omega_Y, 1, X=4, 0)] \\
&\quad [(9 - \Omega_X(4, 5), 0, X=\Omega_X(4, 5), 1) \\
&\quad (\Omega_Y, 1, X=\Omega_X(4, 5), 1)]) \\
&= ([(4, 1, X=5, 1) \\
&\quad (6, 1, X=5, 1) \\
&\quad (\Omega_Y, 2, X=5, 1)] \\
&\quad [(5, 0, X=4, 0) \\
&\quad (\Omega_Y, 1, X=4, 0)] \\
&\quad [(9 - \Omega_X(4, 5), 1, X=\Omega_X(4, 5), 1) \\
&\quad (\Omega_X(4, 5) + 1, 1, X=\Omega_X(4, 5), 1) \\
&\quad (\Omega_Y, 2, X=\Omega_X(4, 5), 1)])
\end{aligned}$$

Figure 14 - Computing $m(Y)$ after Observing $Z=13$.

Varying Failure Rates

The criterion of minimizing the number of faulty components is reasonable in situations where there are no reasons to believe that different components fail with significantly different frequencies. If such is not the case, information about component failure rates could be easily integrated in the scheme proposed. One needs only to change the initial states of the component nodes. For example, instead of initializing component node C_i , to:

$$m(C_i) = ((ok, 0, ()) \\
(-ok, 1, ())) ,$$

we can initialize it at:

$$m(C_i) = ((ok, 0, ()) \\
(-ok, k \log P_i, ())) ,$$

where P_i is the probability of failure in component C_i , and k is any suitable constant.

For those cases (the majority) in which the components' failures are independent, the set of diagnoses obtained with this slight modification, will be those with the highest probability.

V. Conclusions

We have introduced a distributed diagnostic algorithm which fully exploits the topology of the network of the system being diagnosed. The algorithm has linear complexity for singly-connected networks and a worst-case complexity of $\exp(|\text{cycle-cutset}|)$ for multiply-connected networks.

The proposed scheme departs from previous work by treating each component status as a variable, thus facilitating the prediction of all possible model behaviors. This allows the message-passing algorithm to perform the whole diagnostic task without appealing to non-local set-covering procedures. It also simplifies probabilistic approaches like [Pearl 86] by taking advantage of the deterministic nature of the models analyzed.

The intermediate computations generated by the algorithm provide information useful for selecting of new tests. Additionally, information about component failure rates can easily be accommodated.

REFERENCES

- Davis R. [1984]. "Diagnosis from Structure and Behavior", *Artificial Intelligence Journal* 24,
- Dechter, R. [1986]. "Diagnosis with the CSP Model", UCLA Cognitive Systems Laboratory, Technical Report R-72, December, 1986.
- Dechter R., Pearl J. [1986]. "The Cycle-Cutset Method for Improving Search Performance in AI Applications", *To be presented at the Third IEEE Conf. on AI Applications, 1986*
- de Kleer, J. & Williams, B., [1986]. "Reasoning about Multiple-Faults," *Proceedings, AAAI-86, 5th Nat'l. Conf. on AI, Phila., PA., pp. 132-139.*
- de Kleer, J., [1986]. "An Assumption Based TMS", *Artificial Intelligence Journal* 28, 127-162.
- Geffner, H. & Pearl, J., [1986]. "A Distributed Approach to Diagnosis," *To be presented at the Third IEEE Conf. of AI Applications, 1986*
- Genesereth M.R., [1984]. "The Use of Design Descriptions in Automated Diagnosis", *Artificial Intelligence* 24, 411-436.
- Pearl, J. [1986a]. "A Constraint-Propagation Approach to Probabilistic Reasoning", in Kanal, L. N. & Lemmer, J. (Eds.) *Uncertainty in Artificial Intelligence*, North-Holland, Amsterdam, 1986, pp. 357-370.
- Pearl, J., [1986b]. "Distributed Revision of Belief Commitment in Multi-Hypotheses Interpretation," *Proceedings, AAAI-Workshop on Uncertainty in AI, Philadelphia, PA., Aug. 8-10, 1986, pp. 140-145.*
- Peng, Y. & Reggia, J., [1986]. "Plausibility of Diagnostic Hypotheses," *Proceedings, AAAI-86, 5th Nat'l. Conf. on AI, Philadelphia, PA., Aug. 11-15, 1986, pp. 140-145.*
- Reiter, R., [1985]. "A Theory of Diagnosis from First Principles," *Technical Report TR-187/86, Computer Science Department, University of Toronto.*
- Stallman, R.M. & Sussman, G.J. [1977]. "Forward Reasoning and Dependency Directed Backtracking in a System for Computed Aided Circuit Analysis". *Artificial Intelligence* 9, 135-196