An Improved Constraint-Propagation Algorithm for Diagnosis*

Hector Geffner & Judea Pearl

Cognitive Systems Laboratory, UCLA
Los Angeles, CA 90024

Abstract

Diagnosing a system requires the identification of a set of components whose abnormal behavior could explain the 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.

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* arc 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* about the individual component's behavior which "best" explains the manifested system behavior is assembled.

While many proposed schemes fit within this paradigm [Reiter 86, Genesereth 84], the work of [de Kleer *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 of identifying the (minimal) set(s) of assumptions underlying a contradicted prediction (conflict sets) is facilitated by the ATMS machinery [de Kleer 86]. The diagnoses chosen as "best" are those minimal sets of assumptions (candidates) which, if removed, render the model behavior compatible with the manifested behavior. These are assembled from the conflict sets by a set-covering algorithm.

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

The diagnostic algorithm we propose including here consists of a single task: Instead of predicting only those behaviors which assume the proper functioning of components, the proposed algorithm generates explanations for *all possible* behaviors, including, in particular, the optimal diagnoses that account for each new observation. 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.*

The diagnostic algorithm reported here grew out of the analysis and comparison between the probabilistic scheme proposed in [Pearl 1986], and applied to diagnosis in [Geffher *et al.* 1987], and the non-probabilistic approach exemplified by [de Kleer *et al.* 1986]. As it will be discussed below, the resulting scheme can be viewed either as an enhancement of the latter or as a non-probabilistic simplification of the former.

We shall first introduce some notation and present the propagation algorithm for singly-connected constraint networks models (section II). Its application to the diagnosis of a simple digital circuit is discussed in section III. We then extend the propagation scheme to properly handle both multiply-connected networks as well as component models with different prior probabilities of failure (section IV). We conclude with a discussion of related work (section V), and a summary of the main results (section VI).

II The Proposed Scheme

For the purpose of illustrating the ideas underlying the proposed scheme, let us first introduce some convenient notation. Let *L(SjO)* denote a labeling that assigns a status to each component of a system *S*, compatible with a set of observations 0. 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 ,0)*, 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 0_1 and 0_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_x 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 values x_1 and x_2 that variables X_1 and X_2 may take.

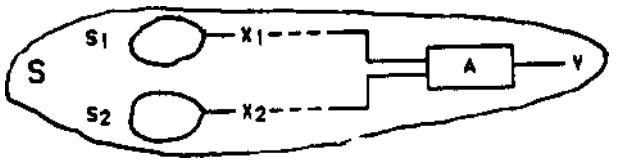


Figure 1

We can now consider the composite circuit formed by an adder *A* with output *Y* and inputs X_x and X_2. We are interested in finding 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 *yofY*. In other words, we want to determine the labeling $L(S, O \cup \{Y = y\})$ 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 labelings that were *already* found optimal overS, 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:

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

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 $A = -ok$, yielding:

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

The constant 1" stands for the penalty associated with component *A* being faulty, which in turn removes the constraint between the values of X_1 and X_2. Notice that the terms required to compute (1) and (3) depend only on the subcircuits S_1 and S_2, and were assumed to be available.

The overall best iabeling(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, in the simple adder circuit depicted in Figure 2, we have only two labelings, $\{A = ok\}$ and $\{A = -ok\}$. The best labeling compatible with *X* =6 is *(A mok)* while, for any other value of *X*, *{A =-ok*; is the best and, in fact, the *only* compatible labeling. Moreover, *{A =-ok}* is the best labeling compatible with *any value* of *X* except X=6. We will take advantage of this fact and will use the symbol Ω_X as a place

$L(\{A\}, \{Y=2, Z=8\} \cup \{X=6\}) = \{A=ok\}$
$L(\{A\}, \{Y=2, Z=8\} \cup \{X\neq 6\}) = \{A=-ok\}$

Figure 2 - Best Labelings far X

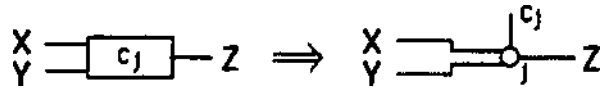
holder for all those, possible infinite, values of *X* which share the same optimal labelings. Refeming to the example above, we will say that *{A m=-ok}* is the best labeling compatible with $X = \Omega_X$, meaning that it is the best (and probably only) labeling compatible with *any* value of *X*. 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 possible value.

The example above 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 the form of constraint propagation; and secondly, that it is possible to incrementally predict all the possible model behaviors by encoding a finite (and usually small) set of "explanations," allowing the constraint propagation algorithm to accomplish the global identification task without appealing to nonlocal 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].

A. Notation

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 $C_j = -ok$. Since we are going to treat components as any other variable in the model, we will usually draw them as nodes as in Figure 3b. rather than depict them as blocks as in Figure 3a., where $C_j \in \{ok, -ok\}$ stands for the name of the variable corresponding to the component status and *j* denotes the undirectional constraint among the component inputs), its output(s) and its status.



a. *b.*

Figure 3 • Components as Nodes of a Constraint Network

The resulting network is composed of

- nodes (or variables), denoted by upper-case letters *(X,* y, *W,....* for observable system points and C,, *Cj,...,* for components);
- constraints, denoted by small letters *(ij,...)*; 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* ,...,c', c^j,..., will refer to the values of variables *X, Y,* IV,..., C' C^i, and subindices $x_1, x_2, \ldots$, wil be used to differentiate among the values of a single variable. We will extend this convention to refer to *px* as the values of the variables in the port *Px*. For example, the port *Px* in Figure 3 is the set of variables *{Y,Z,Cj},* while *px* stands for their associated values $\{y, z, c^j\}$. As stated before, the "value" fy 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 will use *messages,* denoted as $m(P_X^k \rightarrow X)$, where node *X* is the recipient and port P_X^k the origin, to encode the influence of the values of the variables in *Px* upon *X* through the constraint *k*.

B. The Algorithm

Messages: Messages m(Px x) are built from a vector of sub-messages $m(P_X^k \rightarrow x_i)$, where the x, '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 *Px* which would "best" support the potential observation *X* =x,- 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 each $m(P_X^k \rightarrow x_i)$ is a triplet:

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

If $N^k(x_i) = \infty$, for some value x, of *X*, we simply remove x, from the set of values *X* that needs 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 symbol '$\cap$' then, for each of its values x_t, *X* computes the states:

$$m(x_i) = \bigcap_{k \in C(X)} m(P_X^k \rightarrow x_i)$$

$$= (x_i, N(x_i), S(x_i)), \quad (4)$$

where

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

is called the *weight* associated with x, and represents the minimum number of faulty components necessary to account both 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: Let port *Px* contain the variables $Y_1, Y_2, \ldots, Y_n$. Then the parameters of the message $m(P_X^k \rightarrow x_i)$ are computed from:

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

and

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

where the y_i's range over the values of Y_j, and $C^k(x_i, y_1, y_2, \ldots, y_n)$ denotes a predicate indicating the compatibility of its arguments according to constraint *k*. The substraction of the component $N^k(y)$ from the weight *N(y)* in (7) and (8) amounts to subtracting the contribution to *N(y)* that originated in the port P^k. This "orthogonalization" is required to avoid counting the same fault more than once within a single diagnosis.

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, # 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 [de Kleer 86], [Oenesereth 84] and [Davis 84].

III Example

Let us consider the circuit depicted in Figure 4: Components $M.M_2$ and M_3 are multipliers, while A_1 and A_2 are adders. The former will correspond to constraints $C\backslash 1$ 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 new states of nodes *X* and *F*.

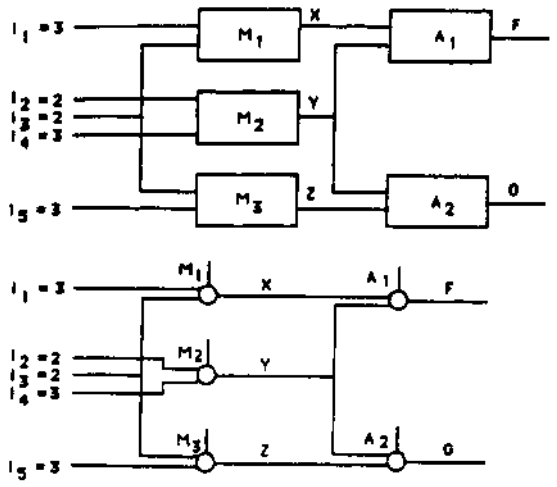
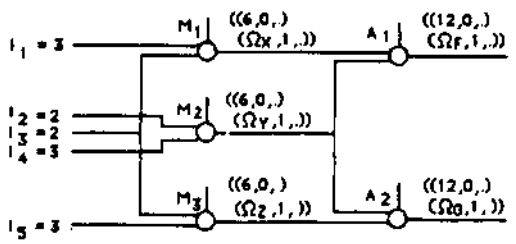


Figure 4 - A Simple Circuit and Its Constraint Network



$$m(X) = ((6,0,\{(I_1=3,I_2=2,M_1=ok)\}) \\ (\Omega_X,1,\{(I_1=3,I_2=2,M_1=\neg ok)\}))$$

$$m(F) = ((12,0,\{X=6,Y=6,A_1=ok\}) \\ (\Omega_F,1,\{(X=6,Y=6,A_1=\neg ok) \\ (X=\Omega_X,Y=6,A_1=ok) \\ (X=6,Y=\Omega_Y,A_1=ok)\}))$$

Figure 5 - (a) Pattern of Messages Generated by the Input (b) New states of 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:

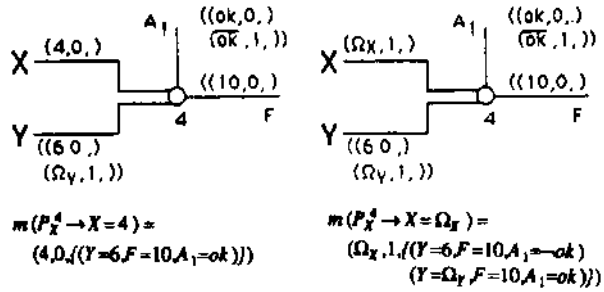
$$m(F) = m(P_F^4 \rightarrow F) \cap m(O_F \rightarrow F) \\ = ((12,0,\{X=6,Y=6,A_1=ok\}) \cap ((10,0,(\,))), \\ (\Omega_F,1,\{(X=\Omega_X,Y=6,A_1=ok) \\ (X=6,Y=\Omega_Y,A_1=ok) \\ (X=6,Y=6,A_1=\neg ok)\}))\ ,$$

which according to (4) and (5) becomes :

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

Since the value $F = 10$, has now 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 M \ = -*ok* associated, in turn, with the support of $X = \Omega_X$ (Fig.5b).

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 rinding 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, $m(P_X^4 \rightarrow X)$.

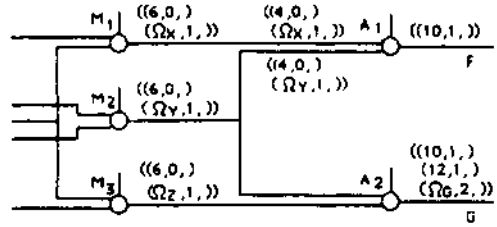


$$m(P_X^4 \rightarrow X=4) = \\ (4,0,\{(Y=6,F=10,A_1=ok)\})$$

$$m(P_X^4 \rightarrow X=\Omega_X) = \\ (\Omega_X,1,\{(Y=6,F=10,A_1=\neg ok) \\ (Y=\Omega_Y,F=10,A_1=ok)\})$$

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 originated from P^A, as specified by 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 received 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.



$$m(A_1) = ((ok,1,\{(X=\Omega_X,Y=6,F=10) \\ (X=6,Y=\Omega_Y,F=10)\}) \\ (\neg ok,1,\{(X=6,Y=6,F=10)\}))$$

$$m(G) = ((10,1,\{(Y=4,Z=6,A_2=ok)\}) \\ (12,1,\{(Y=6,Z=6,A_2=ok)\}) \\ (\Omega_G,2,\{(\ldots)\}))$$

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

Let us assume now that $G = 12$ is 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 for $G = 12$ is $(Y=6,Z=6,A_2=ok)$, and the best 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 useful information for selecting points to test. 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.

Let us also note here that the scheme proposed does not guarantee rinding all irredundant diagnoses [Peng 86], but only those with a *minimum* number of faults, i.e., the *optimal* diagnoses.

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.

A. 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.

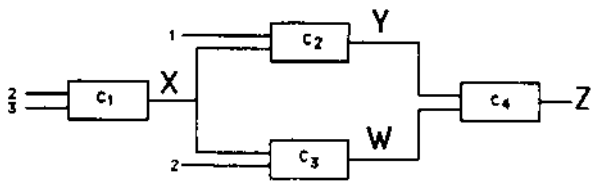


Figure 8 - A Circuit Containing a Loop.

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.

The solution that we will pursue is not new [Peart 86.a] [Dechter & Peari 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 duplicated into a set of identical nodes, each connected to one and only one of its original ports while still preserving the overall behavior. To illustrate the modifications needed to handle loops, let us consider constraint network *S* (Figure 10) in which the instantiation of assumption node *A* decomposes it into a pair of singly-connected networks, S, and S_2. Let *Oi* denote the set of observations gathered in the subnetwork j.t ^fo $i \in \{1,2\}$ and let $O = O_1 \cup O_2$ the complete set of observations. Recalling the notation introduced in Section II, *N(S,0)* stands for the number of faulty components needed to account for the set of observations *0* in system *S.N(y)* was then used as a shorthand for $N(S, O \cup \{Y=y\})$. We shall now use the abbreviations:

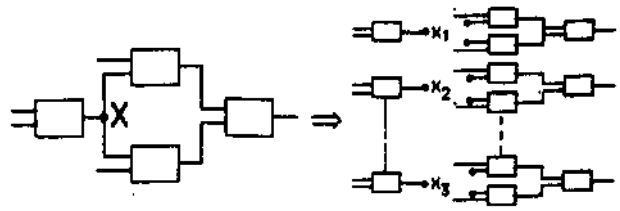


Fig. 9 - Loopy Network Treated as a Family of Singly-Connected Networks.

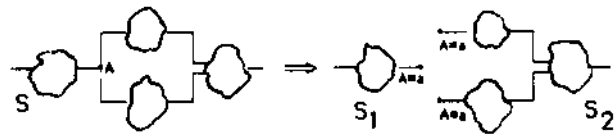


Figure 10 - Breaking Loops by Introducing Assumptions.

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

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

and

$$N'(a) = N(S_i, O_i \cup \{A=a\}). \tag{12}$$

Notice that since the S, 's are singly-connected, the measure W_a (x), i.e. the minimum number of faults in 5,, needed to account for the observations $O_i \cup \{A=a, X=x\}$, can be computed according to the procedure discussed above. 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:

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

where

$$N(a) = \sum_j N^j(a) \tag{14}$$

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 $A=a^*$ itself. From (10H14) it is clear that we will be able to compute *N(y)* for any node Y,if we can compute $W_a(x)$ for every value *a* of *A* and every value *x* of the nodes *X* in *S*.

B. The Algorithm

Each instantiation of the cutset variables identifies a set of singly-connected networks. We shall refer to the instantiated nodes as *assumption nodes,* to their instantiations as *assumptions* or *tags,* and to the states of the net compatible with a given set of assumptions 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 [de Kleer 86].

As with node values, the message-passing algorithm will not explicitly represent each of the possible values 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, we will use $\Omega(x_1,\ldots,x_n)$ as the place holder for values of X *other* than $x_1,\ldots,x_n$. This will render a tag containing the instantiation $X=x_i$ *incompatible* with any tag containing the instantiation $X=\Omega(\ldots,x_i,\ldots)$. The Ω-values assigned to non-assumption nodes are not required to keep this extra information explicit; so, for them, we use our previous, simpler notation.

Messages will now have the form :

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

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

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

where

$W_a^k(x)$ represents the *component* of $W_0(x)$ originated in the ik-th port of X;
$S_a^k(x)$ denotes its underlying *support* and
$N_X^k(a)$ stands for the *contribution* $N'(a)$ to $N(a)$, for $X \in S_i$, as received from the it-th port.

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

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:

$$m_a(x) = \bigcap_{\substack{i \in C(x) \\ h=a}} m_{h_i}(P_X^i \to x) \ ,$$

$$= (x, W_a(x), S_a(x), a, N_X(a)) \ , \qquad (15)$$

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:

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

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 computed according to Eq.(13):

$$N(y) = \min_a N_a(y)$$
$$= \min_a [N(a) - N_Y(a) + W_a(y)] \ . \qquad (17)$$

where the weight $N(a)$ is available at node A, and both $N_Y(a)$ and $W_a(y)$ are available at Y. 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:

$$m_a(O_Y \to Y) = ((y, 0, (), a, W_a(y))) \ . \qquad (18)$$

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

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 $C_1 = ok$ is $X = 4$. To arrive at this conclusion, we need either to solve a linear equation $2 \cdot X + 5 = 13$ 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.

For an illustration of how the proposed algorithm would handle the diagnosis of a circuit like the one depicted in Fig. 8, the reader might refer to [Gefrher *etal.* 1986].

C. 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 rates. 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,, to:

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

we can initialize it at:

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

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 Related Work

The diagnostic algorithm presented here grew out from the analysis and comparison of the scheme proposed in [de Kleer *et ai* 86] and the one reported in [Geffner *et al.* 87]. The former is a typical two stage algorithm. Predictions that assume the proper functioning of components whose failure was not established, are matched against the observations. A mismatch identifies a set of assumptions (conflicts) which cannot simultaneously hold. These sets are obtained from the tags attached to the predictions refuted. From these sett, a non-local set-covering algorithm constructs the set of minimal diagnosis, i.e. minimal sets of violated assumptions which explain the observations.

The diagnostic scheme reported in [Geffher *et al.* 87] is a probabilistic scheme in which both "right" and "wrong" behavior are considered on the same basis. Knowledge of the network topology (the singly-connectedness of the network is embedded in the algorithm) and a set of probabilistic goodness measures attached to hypotheses, allow copious prunning, avoiding the combinatorial explosion characteristic of unaided multi-hypothesis diagnostic algorithms. On the other hand, the scheme seems unnecessarely expensive for many applications, and it also requires the assessment of the probability distribution associated with the component's I/O which, generally, is not available.

The scheme proposed here borrows from the latter what we consider to be its two main features : knowledge of the network topology, and numeric qualification of candidate hypotheses. The algorithm however can be interpreted as embedding two further assumptions : very small failure rates and uniform probability over the possible output values of failed components. The resulting diagnostic scheme can be thought then, as the non-probabilitic version of [Oeffner *et al.* 87], in the sense that it can only accommodate a narrow set of probabilistic models. On the other hand, it only requires to propagate one type of message, it does not require the assessment of probabilities and it is sensibly less computationally expensive. Compared to [de Kleer *et al.* 86] and [Reiter 85], it does not require a non-local set covering procedure for identifying the culprits. Moreover in the singly-connected case, the knowledge of the network topology permits to tag messages only with a numeric measure rather than with the identity of the assumption set In the multiply-connected case however, tags identifying the cutset assumptions are needed. In this respect, the scheme reported in [de Kleer *et al.* 86] which does not presuppose any knowledge about the type of network, appears to be treating all variables as cutset variables. For sparse networks however, the preprocessing step needed to identify a cut set may be worthwhile.

VI 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(\ cycle-cutset* I) 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 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 the selection 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., Pearl J. [1987]. "The Cycle-Cutset Method for Improving Search Performance in AI Applications". *Third IEEE Corf, on AI Applications, 1987*

de Kleer, J. & Williams, B., [1986]. "Reasoning about Multiple-Faults," *Proceedings, AAAJ-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]. "An Improved Constraint Propragation Algorithm for Diagnosis," *Cog. Syst. Lab, TR-73,* UCLA.

Geffher, H. & Pearl, J., [1987]. "A Distributed Approach to Diagnosis," *Third IEEE Corf, of AI Applications, 1987*

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 Hypothesis", *Proceedings AAAI-86,* 5th Nat'l. Conf on AI, Phil., pp 140-145.

Reiter, R., [1985]. "A Theory of Diagnosis from First Principles," *Technical Report TR-187186,* 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 Journal 9,135-196*