*DX-2004*

# *A probabilistic-based algorithm to diagnose complex electronic systems*

*Mohamed-Amine MAALEJ* and *Véronique DELCROIX* and *Sylvain PIECHOWIAK*

**Address:** LAMIH UMR CNRS 8530, Université de Valenciennes et du Hainaut Combrésis,59313 Valenciennes Cedex 9, France.

**E-mail:** {Mohamed-Amine.Maalej, Veronique.Delcroix, Sylvain.Piechowiak}@univ-valenciennes.fr

**Fax:** +33 3.27.51.13.16

**Tel:** +33 3.27.51.14.73

## Abstract

*The complexity of electronic devices increases continually; that is why diagnosing actual circuits is more and more difficult. Different diagnosing methods exist. The aim of this paper is to present a new model-based diagnosis method that uses the Bayesian network to describe the system and to compute the probability of the most likely diagnoses. This method determines one of the most probable sets of components which explain the abnormal behavior of a faulty system. When the list of diagnoses is too long, it is difficult to determine the set of components that must be repaired. In those cases additional observations must be used in order to refine diagnosis. Our method has the advantage of calculating diagnoses fast and thus allows a rapid decision about the necessity of supplementary observations. We will present its steps and some diagnosis examples and the different tests performed.*

**Key words:** Uncertain Reasoning, Model based diagnosis, Bayesian network

***15th International Workshop on Principles of Diagnosis***

***June 23-25, 2004 Carcassone, France***

# A probabilistic-based algorithm to diagnose complex electronic systems

**M. A. Maalej** and **V. Delcroix** and **S. Piechowiak** [1]

**Abstract.** [1] The complexity of electronic devices increases continually; that is why diagnosing actual circuits is more and more difficult. Different diagnosing methods exist. The aim of this paper is to present a new model-based diagnosis method that uses the Bayesian network to describe the system and to compute the probability of the most likely diagnoses. This method determines one of the most probable sets of components which explain the abnormal behavior of a faulty system. When the list of diagnoses is too long, it is difficult to determine the set of components that must be repaired. In those cases additional observations must be used in order to refine diagnosis. Our method has the advantage of calculating diagnoses fast and thus allows a rapid decision about the necessity of supplementary observations. We will present its steps and some diagnosis examples and the different tests performed.

## 1 INTRODUCTION

The problem of diagnosis in general has interested researchers in *Artificial Intelligence* (AI). It is a complex problem which involves several scientific domains. At present, industrial systems are becoming increasingly sophisticated and include massively electronic subsystems for control and command. In this paper, the purpose of the diagnosis is to look for the sets of components that explain global failure of the device. Among various diagnosis approaches stemming from the AI field, *Model Based Diagnosis* (MBD) is particularly interesting in the diagnosis of reliable and complex systems. Indeed, it allows identifying diagnoses even though insufficient expertise (knowledge from the exploitation) is available. The electronic devices are examples of these situations.

The AI community often describes a system as a set of components. The foremost methods use a *System Description* (SD) in order to describe the behavior of each component and their interconnections. When the system is faulty and diagnoses must be found, the description of the system is used to compute diagnoses.

In order to compute a diagnosis, different methods exist. A diagnosis is a state affectation of all the components of the system which is consistent with observations [2]. We define a diagnosis for a faulty device as a set of component(s) that, if considered as faulty and all other components are good working, explain the abnormal behavior of the device. A diagnosis is called minimal if there is no diagnosis containing a strict subset of its failure components.

Different diagnosis methods exist. We present here the main methods of model-based diagnosis. The majority of the approaches have emerged from advances in Automatic science and AI. These approaches elaborate their diagnosis from the knowledge of the system to be diagnosed. The devices models can be elaborated from information stemming from the conception of the system and the knowledge of physics laws. Model-based diagnosis approaches have been extensively explored over the past thirty years. A large number of different approaches for detection, diagnosis and control of failures have been developed. An ATMS-based computational multiple fault diagnosis framework is provided by the *General Diagnosis Engine* (GDE) [3] and Sherlock [4]. To determine logically possible diagnoses from a broken device, the GDE and Sherlock processes call for three steps: *prediction*, *conflict generation* and *candidate generation*.

*Prediction* step consists in using the system model and a set of given values (often in entrance of the system) to simulate expected values during the correct functioning of the system. In the *conflict generation* step expected values and really observed values are compared. A conflict is a set of components which cannot all be in a good working state at the same time without contradicting observations. In a conflict, one component at least is failing. The *candidate generation* step makes it possible to generate the diagnoses which have a non empty intersection with all the conflicts.

Our article focuses on a GDE approach and proposes a new algorithm which determines the most likely diagnoses. To determine the sets of diagnosis, we use a Bayesian network as a model for the device.

In Section 2 we concentrate on the Bayesian networks diagnosis. We will present the method used to build the Bayesian networks of various circuits. In the third part, we propose an algorithm that computes one amongst the most likely diagnoses. Then, in section 4, we give some examples of results returned by our algorithm and an illustration of the computing time needed to diagnose some circuits. Finally we conclude with a summary and a statement of future research.

## 2 USING THE BAYESIAN NETWORK AS A MODEL

The Bayesian network formalism was proposed in the early eighties in order to allow efficient representation of, and rigorous reasoning with uncertain knowledge. Bayesian network-based diagnosis takes advantage of probabilistic models of devices. Each model is represented by a collection of nodes that describes the behaviors of components of the system. An algorithm of inference propagates the available knowledge and computes posterior probabilities using the Bayes rule in probability theory.

[1] LAMIH, Université de Valenciennes et du Hainaut Cambrésis, France email : mohamed-amine.maalej@univ-valenciennes.fr

## 2.1 Definition of Bayesian networks

*Bayesian Networks*[2](BN) or Bayesian belief networks are directed acyclic graphs in which nodes represent the variables of the system. They also include conditional probabilities of the values of each child node given the values of its parent nodes, and simple probabilities for nodes without parents.

A Bayesian network is defined by an acyclic directed graph G = (V, E), where V is the set of nodes, and E the set of arcs. Bayesian network contains also a set of random variables corresponding to those defined on the space of probability (W, Z, P), such as:

$$P(v_1, v_2, \ldots, v_n) = \prod_{i=1..n} p(v_i \mid C(v_i))$$

Where $C(v_i)$ is the set of causes (parents) of $v_i$ in the graph G.

Bayesian network nodes represent the variables of the system, thus they represent influent parameters like the temperature of a system, the state of a component, the occurrence of an event… Every node has distribution of probability. In the discrete Bayesian network case, a probability table is used to represent this distribution. The size of this table depends on the parent number. In the case of certainty (case of observations), said to be evidence variables, the prior probability is fixed to one.

$$P(v_i = x_i) = 1 \text{ } if \text{ } the\, value\ x_i \text{ } is\ observed \quad for\ v_i$$

The links in the network represent informative or causal dependences among the variable. Dependences are quantified by conditional probabilities. These allow the definition of probability at the level of a node, considering the knowledge available on the similar nodes.

All information, observation or knowledge permits the update of other probabilities in the network. This principle is named inference in the Bayesian network. Inference or update of probabilities in a Bayesian network consists in computing the posterior probability distribution for a set of query variables from the probabilities of the other variables observed. The belief update in Bayesian networks has been shown to be a NP-hard problem [1].

There are several exact and approached inference algorithms. Among the first exact algorithms we find the clustering algorithm [9] and probabilistic logic sampling [8] is the first and simplest forward sampling algorithm.

A survey and a good classification of BN inference algorithms are available in [7].

## 2.2 Example of Bayesian network

In order to build a Bayesian network we need knowledge about the components of the system such as inputs, outputs, fault conditions and component states. The associated graph can be used to exploit generic knowledge of an expert of the domain. This graph can be seen as a computational architecture for storing factual knowledge and manipulating the flow of knowledge in the network. Then, we associate compact representation of the joint probability distribution of the various variables that describe our system.

In order to generate the global Bayesian network of a circuit, we associate a node for every system input or output. Every component is represented by a sub model with a set of input nodes, a set of output nodes and a state node so that output nodes are directly influenced by the input and state node (see figure 1).

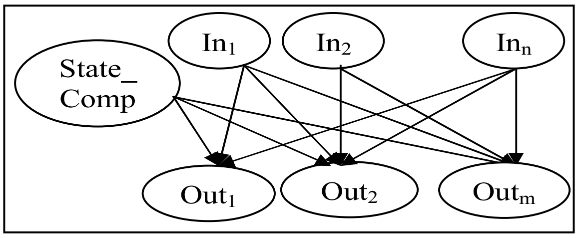


**Figure 1.** Component model [10]

Figure 2 below shows the logical diagram for a one bit-adder circuit. This example is often found in the research literature [5] [6] and [11]. It is a simple example, which will serve to explain the different steps in the construction of a Bayesian network for a circuit.

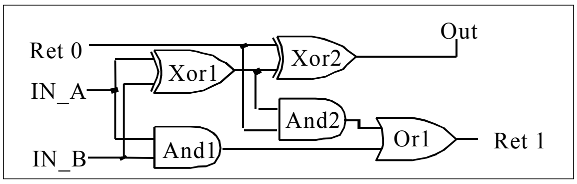


**Figure 2.** A one bit-adder circuit

The Bayesian network of the one-bit adder Bayesian network presented in figure 3 is built as follows: each input or output variable is represented by a node in the graph, each gate is represented by one node called "State of the gate", the input nodes of a gate and its state node constitute the parents of the output node for this gate. In our example we consider two possible values for the state of a gate: "Good-working" and "Abnormal" noted {ok, ab}. This states set can be widened if the behaviors of other failure modes are known.

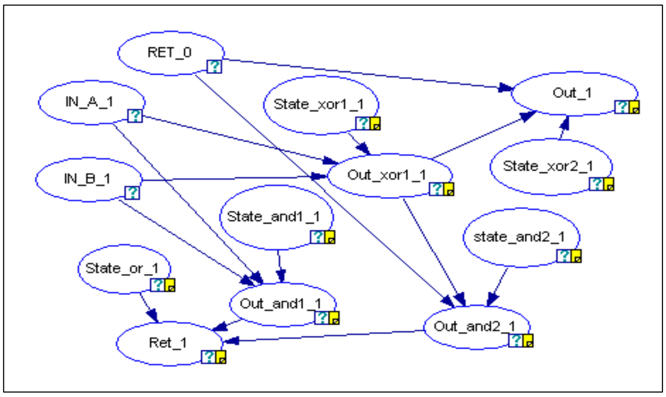


**Figure 3.** Graph of the Bayesian network for the one-bit adder

[2] These are the most common names, but we find in the literature belief network, probabilistic network, causal network, and knowledge map.

A probability distribution is attributed to every variable. In the case of a node without predecessor, such as nodes "state of a component" or input variables of the system, it is a prior probability distribution. Otherwise, if the node has one or several predecessors, it is conditional probability distribution. It represents the probability of the values of the node knowing the values of its immediate predecessors in the graph.

| state_and2_1 | OK | | | |
|---|---|---|---|---|
| RET_0 | on | | off | |
| Out_xor1_1 | on | off | on | off |
| on | 1 | 0 | 0 | 0 |
| off | 0 | 1 | 1 | 1 |
| state_and2_1 | ab | | | |
| RET_0 | on | | off | |
| Out_xor1_1 | on | off | on | off |
| on | 0.5 | 0.5 | 0.5 | 0.5 |
| off | 0.5 | 0.5 | 0.5 | 0.5 |

**Figure 4.** Conditional probability table of Out_and2

Figure 4 gives the conditional probability table of the node Out_and2 given the values of its three parents State_and2 , Ret_0 and Out_Xor1.

| | |
|---|---|
| OK | 0.99999 |
| ab | 1e-005 |

**Figure 5.** Probability table of the node State_and2

The values of State_and2 do not depend on any other variable. The probability table of this variable is given in figure 5.

# 3 A NEW ALGORITHM FOR COMPUTING ONE AMONG THE MOST LIKELY DIAGNOSES

## 3.1 Working hypotheses

In this section, we propose an algorithm to seek the most likely minimal diagnoses for a faulty device. We suppose that the components of the system work independently and the circuit gives wrong output(s). The set of observed variables (called observations) must show the breakdown of the whole device. In other words, the observations must be incompatible with the good working state of the device. A diagnosis is a set of components whose faults explain these observations, i.e. conditional probability of a diagnosis given that any evidence is not null: $P(D|Obs) \neq 0$.
Every diagnosis has a non null probability and the sum of all the diagnosis of a faulty device equal one: $\sum P_D$.
The size of a diagnosis is the number of faulty components explaining some observations, recalling that all other components are ok.
Here are some notations and notions in terms of probability which will be used afterward:

- *Obs*: the set of observations (affectations of some variables of the device).
- $\mathcal{C}$ : the set of components of the device.
- Smax: the maximum size of a diagnosis (larger diagnoses will not be sought).
- $\mathcal{C}_K = \{C_k \in \mathcal{C}, k \in K\}$ and $\mathcal{C}_{NK}$ its complement.
- $ab(\mathcal{C}_K)$ : all the components in $\mathcal{C}_K$ are failing.
- $ok(\mathcal{C}_K)$ : all $\mathcal{C}_K$ components are good working.
- $D_K$ : short notation for the diagnosis $ab(\mathcal{C}_K) \wedge ok(\mathcal{C}_{NK})$.
- $P_{DK}=P(D_K \mid Obs)$ : posterior probability of the diagnosis $D_K$.
- $P_C$ : posterior probability of failure of the component $C$: $P_C = P(ab(C) \mid Obs)$.

The goal of our algorithm is to determine the most probable minimal diagnosis for each less than or equal to *Smax* size. To calculate these diagnoses, our algorithm initially needs the Bayesian network of the system BN, the size of researched diagnoses limit *Smax* and a set of observations *Obs*.
The first step consists of determining the suspect components (candidates) and adding them in decreasing order of their posterior probabilities; the produced list is called *List of Suspect Component* (*LSC*). In the second step we compute the first simple diagnosis (diagnosis with a single faulty component) with its associated probability. We use *LSC* and we test each component. At this step, other algorithms propose to truncate this list [5] and [6] with a fixed rate. When one criterion is used to stop generating candidates, very low probable diagnoses can be eliminated. We propose to test the entire list until the first simple diagnosis is found, by this we find the most probable simple diagnosis even if it has a very low probability. Testing a component consists in verifying if it corresponds to a diagnosis for the faulty device. In some particular cases these tests increase computing time, for example when there is neither simple nor double diagnosis. Since our method determines the simplest and most likely diagnosis that fits with the observed data, it can be seen as an application of the Occams Razor logical principle.
When a simple diagnosis is found, it is saved in a *Diagnoses List* (*DL*). This list contains all computed diagnoses and their corresponding posterior probabilities.
When we compute a diagnosis, we calculate its posterior probability $P_{DK}$ as proposed by Delcroix in [6].

$$P_{DK} = P(ab(C_K) \wedge ok(C_{NK}) | Obs)$$

For a simple diagnosis, the equation above takes this form:

$$P_{DK} = P(ab(C_k) \wedge ok(C_i) \, for\ every\ i \neq k | Obs)$$

i.e.

$$P_{DK} = P(ab(C_k)|Obs).\prod_{i \neq k} P(ok(C_i)|ab(C_k) \wedge ok(C_j) \forall j < i)$$

This requires performing as many inferences as there are good working components. In order to avoid computing this probability which needs an important number of inferences, we use this approximation of $P_{DK}$:

$$P(ok(C_{NK}) \mid ab(C_K),\ Obs) \ \cong \ \ P(ok(C_{NK}))$$

Thus we assume diagnosis probability to be:

$$P_{DK} \cong P(ab(\mathcal{C}_K) \big| Obs). \prod_{C_k \in \mathcal{C}_{NK}} P(ok(C_k))$$

This approximation simply translate that "knowing that components of $\mathcal{C}_K$ are faulty and that $ab(\mathcal{C}_K)$ explains *Obs*, we clear components of $\mathcal{C}_{NK}$".

After finding the first diagnosis, we cut the candidates list; thus only upper probable faulty components remain in the *LSC*.

The following propositions justify the *LSC* truncation: Suppose that $LSC = \{C_1, C_2, \dots C_n\}$ in decreasing order of their posterior probabilities and let $P_{okMin} = Min\ P(ok(C_k))$. $P_{okMin}$ represents the probability of the component having the lower good working prior probability.

**Proposition 1**

*Let $Pc_i \geq Pc_j$ and $D_i$, $D_j$ are the two simple diagnoses associated to $C_i$ and $C_j$.*
*Using the approximation when computing $P_{Di}$ and $P_{Dj}$,*
*Then $P_{Di} \geq P_{Dj} \times P_{okMin}$.*

**Proof**

According to $P_{okMin}$ definition:

$$P(ok(C_i) \in [P_{okMin},\ 1]$$

Then, for each $P(ok(C_i), P(ok(C_j)$:

$$\frac{P(ok(C_i))}{P(ok(C_j))} \in [P_{okMin}, \frac{1}{P_{okMin}}] \quad (*)$$

Furthermore, using Delcroix's approximation

$$\frac{P_{D_i}}{P_{D_j}} \cong \frac{P_{C_i}}{P_{C_j}} \times \frac{P(ok(C_j))}{P(ok(C_i))} \times \frac{\prod_{k \neq i,j} P(ok(C_k))}{\prod_{l \neq i,j} P(ok(C_l))}$$

since $Pc_i \geq Pc_j$ and as defined by Eq $^{(*)}$, it follows: $P_{Di} \geq P_{Dj} \times P_{okMin}$.

**Proposition 2**

*Let $Pci \geq Pcj$ and let $D_i$ be the simple diagnosis composed of Ci. Let Djk be a double diagnosis holding $C_j$ and $C_k$.*
*Using Delcroix's approximation when computing $P_{Di}$ and $P_{Djk}$,*
*Then $P_{Di} \geq P_{Djk} \times PokMin^2$.*

**Proof**

We define $P_{Di}$ as follow:

$$P_{D_i} \cong P_{C_i} \times P(ok(C_j)) \times P(ok(C_k)) \times \prod_{l \neq i,j,k} P(ok(C_l))$$

Also $P_{Djk}$ is defined:

$$P_{D_{jk}} \cong P_{C_j} \times P(ab(C_k) \big| ab(C_j), Obs) \times P(ok(C_i)) \times \prod_{l \neq i,j,k} P(ok(C_l))$$

We denote,

$$P'_{C_k} \cong P(ab(C_k) \big| ab(C_j), Obs)$$

Then,

$$\frac{P_{D_i}}{P_{D_{jk}}} \cong \frac{P_{C_i}}{P_{C_j}} \times \frac{P(ok(C_j))}{P'C_k} \times \frac{P(ok(C_k))}{P(ok(C_i))}$$

Knowing that

$$\frac{P_{C_i}}{P_{C_j}} \geq 1 \ and \ \frac{P(ok(C_j))}{P'C_k} \geq P_{okMin} \ and \ \frac{P(ok(C_k))}{P(ok(C_i))} \geq P_{okMin}$$

This implies

$$P_{D_i} \geq P_{D_{jk}} \times P_{okMin}^{\ 2}$$

These two results allow us to claim that when a simple diagnosis is found, any other simple or double diagnoses are less probable or, in few cases, are slightly more probable. That's why we truncate the *LSC* just after the first found diagnosis.

In the third step, we consider the first element of *LSC* and we attempt to complete diagnosis with the other components in *LSC*. In this step we use another list *TCDL* (To be Completed Diagnosis List). When a diagnosis is found, it is stored it in *DL* and we look for a diagnosis of upper size. This looping procedure ends if the size of diagnosis equals *Smax* or if there are no more components in *LSC* .

To complete *TCDL* we use a particular scroll through *LSC* manner. We illustrate this course by this example. Let's consider a list with six components, *LSC*, with $Pc_1 \geq Pc_2 \geq \dots \geq Pc_6$. The research of the simple diagnoses is done for $C_1$ then $C_2$, $C_3$, $C_4$... If it is supposed that *C5* is a simple diagnosis, the couples which will be considered for the research of the double diagnoses will be: $(C_1, C_2)$, $(C_1, C_3)$, $(C_1, C_4)$ in continuation $(C_2, C_3)$, $(C_2, C_4)$ then $(C_3, C_4)$. If there is not double diagnosis, the tested sets of components of size *3* are: $(C_1, C_2, C_3)$, $(C_1, C_2, C_4)$, $(C_1, C_3, C_4)$ then $(C_2, C_3, C_4)$.

## 3.2 The algorithm

Inputs: BN, Obs, *Smax*
Output: *DL*
We denote $C_n$=*LSC[n]* the n[th] component in *LSC*

1. Compute *LSC*:
   - Change the probabilities of observed nodes (insert evidence(s) in the BN)
   - Make an inference to compute the posterior probabilities of all the components of the circuit
   - Add components and their posterior probabilities to *LSC*
   - Order *LSC* in decreasing order of posterior probabilities
2. Search for one of most probable simple diagnosis:
   $n \leftarrow 1$
   While ($n \leq |LSC|$ *And LSC not empty*)

- If $C_n$ constitutes a diagnosis (no conflicts in the BN when we introduce Obs and assign ab to the state of the component *LSC[n]* and ok to all other components)
  - Compute $P_{Cn}$
  - Add $C_n$ and $P_{Cn}$ to *DL*
  - Cut *LSC* : remove $C_n$ and all $C_j$ such that $P_{CJ} \leq P_{Cn}$

  Else $n \leftarrow n+1$

  End if

  End while

3. Search upper size diagnoses:

   $n \leftarrow 1$

   $S \leftarrow 2$

   While ($S \leq Smax$ *and* $n \leq |SDL|$ *and LSC not empty*)

   While (*end of LSC not reached and no diagnosis found for the size S*)
   - Complete *TCDL* from *LSC* until *its size equals S*
   - If *D* is a diagnosis (no conflicts in the BN if we introduce Obs and assign ab to all components of *D* and all other components as *ok*)
     - Compute $P_D$
     - Add *D* and $P_D$ to *DL*
   - Cut *LSC* : remove $C_k$ and all $C_j$ such that $Pc_j \leq Pc_k$, $C_k$ is the component of *D* with the lowest probability

   End if

   End while

   $S \leftarrow S+1$

   End while

End.

# 4 COMPUTING EXAMPLES AND FIRST RESULTS

We developed a C++ application, CBDiag, that implements this algorithm. It allows building and diagnosing digital circuits like adders and multipliers. This application uses the Smile[3] library to build the Bayesian networks and to make inferences. We run CBDiag on multiplier of different sizes (cf. table1) and with different sets of observations. The faulty behaviors of used circuits correspond to wrong arithmetic operations such as 2x0≠0 or 7x6=700. In order to decrease the inference computing time, we tried to use approached algorithms of inference (using stochastic sampling). It gives no result because of the very low observations probabilities used. Even with very high sampling rate, obtained results have very high error rate and a computing time higher than the one obtained using the exact inference algorithm.

## 4.1 Comparison with other methods

We confirmed the validity of the diagnoses computed by our algorithm by comparing them with those obtained by other applications that diagnose on the same circuits [6] and [11].

We have also tested the two algorithms in an n-bit ripple-carry adder. We use the same observations as in [5] and we obtain the same five diagnoses as GDE.

Like GDE we use Bayes rule to determine the posterior probabilities. de Kleer and Williams [3] determine the probabilities of candidates in incremental process. Our algorithms determine a list of suspect components knowing a fixed set of observations. We use this computed list as a base to compute the posterior probabilities of the best diagnosis.

**Table1.** Circuits used for the tests and their features.

| Circuits and their features | | |
|---|---|---|
| BN Name | Number of component | Number of nodes |
| Multi1x1bit | 6 | 16 |
| Multi2x2bit | 24 | 76 |
| Multi2x4bit | 48 | 146 |
| Multi2x4bit | 96 | 280 |
| Multi2x16bit | 192 | 566 |
| Multi8x8bit | 384 | 1072 |
| Multi2x64bit | 768 | 2246 |
| Multi16x16bit | 1536 | 4192 |

The tests performed below focus on running time of CBDiag. We search for the relation between execution time and the number of nodes in the network. The tests are made on a Dell Optiplex GX1 PC machine, with a 500Mhz Intel Pentium III processor and 256 Mb RAM.

## 4.2 Example 1

We use the BN of a 2x4-bit multiplier[4], this circuit contains 48 components. Its associate BN holds 146 nodes. We suppose that the prior faulty probabilities are of $10^{-5}$ for all components. The considered circuit fault represents the wrong arithmetic inequality 0x0 ≥ 32. In order to simulate this faulty behavior, we set all inputs to zero and we attribute one as a value for the fifth bit output corresponding node.

The exhaustive list of diagnoses which sizes are less or equal to 3 contains 3 Simple diagnoses with probability 0.333, 18 Double diagnoses with probabilities about $10^{-6}$ and 117 Triple diagnoses with probabilities of order of $10^{-12}$. These results were obtained with another algorithm [6] with a truncation of *LSC* when $Pc_1/Pc_i > 10^6$; the complete computing time is about 3242 seconds.

With CBDiag we obtain only one simple diagnosis having the probability of 0.333 and the computing time is only 0.19s. The diagnoses of upper size are not looked for with CBDiag in this case, since the list of suspect components *LSC* is truncated after computing the first simple diagnosis. So that $LSC=\varnothing$. As explained before, CBDiag looks for upper size diagnoses only when *LSC* holds other possible diagnoses.

Focusing on CBDiag result, we can predict, in this example, that it isn't necessary to ask for a size of diagnoses superior to 1. Indeed, the sum of probabilities of all diagnoses of a faulty system is equal to one. This CBDiag result (a simple diagnosis with a probability of 0.333) implies that all other simple diagnoses have probabilities less or equal to 0.333/0.9999 ≅ 0.33303 (cf proposition 1) and that every probability of any double diagnosis can not exceeds 0.33307.

In order to decrease computing time without appreciably decreasing the sum of probabilities of computed diagnoses, we can use superior *LSC* truncation rate.

## 4.3 Example 2

In this example, we use the Bayesian network of a 2x32-bit multiplier that has the same prior faulty probabilities for all its

[3] Structural Modeling, Inference, and Learning Engine, a platform independent library of C++ classes, developed in the Decision Systems Laboratory at the University of Pittsburgh. http://www.sis.pitt.edu/~genie/

[4] We use the logic diagram of MOTOROLA Semiconductor Technical Data (MC14554B) in order to construct Multipliers BNs.

components (equal $10^{-5}$). This circuit is used with this faulty observation ($x$x$y=z$, *when* $x\geq 2$, $y\geq 2$ *and* $z\leq 3$): multiplying two numbers that are superior or equal to 2, we obtain a number that is less or equals to 3. All we have to do is to set the second bit of each circuit-input to one and assign all circuit-out bits to zero except the first and second bits. CBDiag computes only one simple diagnosis in a short time (less than 2 seconds), this diagnosis has a probability of 0.09. This result is indicative of the big size of diagnosis set (certainly more than 10 diagnoses). This claimed behavior was confirmed by the use of another algorithm [6]; even with a low *LSC* truncate rate (equals to 1000) and a limited diagnosis size (equals to 2), we obtain more than 50 diagnoses in 756s computing time.

CBDiag computes only one simple diagnosis in a short time (less than 2 seconds). Guided by CBDiag result, we realize the necessity of additional observations. Indeed, since the most probable diagnosis has a low probability value and knowing that the sum of all probability of any faulty system is equal to one it is quite justified to think that the set of diagnosis is too large.

## 4.4 Computing time

Realized tests confirm that CBDiag gives exact results and helps us to improve the quality of our diagnosis. In a very short time it reveals the lack of observations, if need be, and helps to avoid long and useless lists of possible diagnoses.

We present some computing times of CBDiag for different sizes of circuits. The used observations point to wrong operations consisting of multiplication of even numbers and odd result: it is enough to leave first bits of all inputs at zero and the first out bit at one. We have tested our algorithm for different sized multipliers.

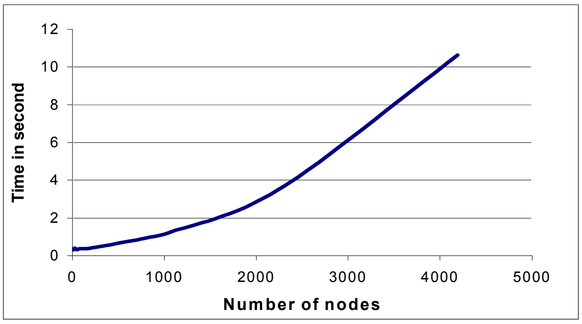


**Figure 6.** Running time in second vs. size in node (of a multiplier)

The running time does not exceed 12 seconds even at a system of more than 4000 nodes (16x16 bit Multiplier). Indeed, each computing time includes not only algorithm running time but also two inferences. One inference when constructing *LSC* and another is necessary when computing the diagnosis probability.

# 5 SUMMARY AND FUTURE WORK

We have presented a brief survey of model-based diagnosis approaches. Based on these approaches, we have provided an algorithm for computing most probable diagnoses for a faulty system. The three steps of our algorithm permit fast computing diagnoses and give an idea about the necessity or otherwise of other observations. We have detailed two tests; the first exposes the interest that our algorithm has in presenting the maximal diagnosis size to search for the best diagnosis, the second example demonstrates the importance that our algorithm has when the diagnosis set is too large and the most likely diagnosis has a small probability.

In future work, we want to separate clearly the time due to inferences in CBDiag and Delcroix's algorithm. Other tests must be done to measure the influence of observations on computing time. In this scope, we intend to study the effect of prior probability on the use of approached inference algorithms. We will also investigate new techniques to construct the functional and behavior models of discrete semiconductors like diodes, rectifiers or transistors and passive electronic components such as capacitors, inductors or resistors, in order to construct the Bayesian networks of those kinds of systems. It appears that the approach used to represent logical components cannot be used for discrete or passive components. Moreover, those components cannot be represented separately in elementary Bayesian networks. Other works show that it is possible to use qualitative reasoning to represent those types of component. We plan to use systemic or functional model to represent analogical system. Our aim is to propose Bayesian models for this kind of circuits that could be used by our algorithm.

# 6 REFERENCES


[1] G.F. Cooper, *The computational complexity of probabilistic* inference *using Bayesian belief networks*, Artificial Intelligence, 42: 393-405, 1990.

[2] A. Darwiche, *Model-based Diagnosis using Causal Networks*, Proceedings of the 14th IJCAI, 211-217, 1995.

[3] J. De Kleer and B.C. Williams, *Diagnosing multiple faults*, Artificial Intelligence, 32(1): 97-130, 1987.

[4] J. De Kleer and B.C. Williams, *Diagnosis with be-havioural* modes, in: proceedings of the 11th IJCAI-89: 1324-1330, 1989.

[5] J. De Kleer, *Focusing on Probable diagnoses*, Proceedings of the 9th AAAI, 842-848. 1991.

[6] V. Delcroix, S Piechowiak, and M.A. Maalej, *Calcul des diagnostics les plus probables a posteriori*, Revue d'intelligence artificielle 17(4): 627-654, 2003.

[7] H. Guo and W. H. Hsu, *A Survey of Algorithms for Real-Time Bayesian Network Inference*, AAAI/UAI/KDD-2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems. 2002.

[8] M. Henrion, *Propagating Uncertainty in Bayesian Networks by Probabilistic Logic Sampling*, In J.F. LEMMER AND L.N. Kanal, editor, Uncertainty in Artificial Intelligence, 2: 149-163, 1988.

[9] J. Pearl, *Fusion, propagation, and structuring in belief networks*, Artificial Intelligence, 29: 241-288, 1986.

[10] J. Pearl, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann, San Mateo, CA, 1988.

[11] S. Piechowiak and J. Rodriguez, *A constraint network manager for diagnosing*, 7th IAAI, Duster, Texas, 1994.