IMPROVING THE DETERMINATION OF MINIMAL HITTING SETS IN MODEL-BASED DIAGNOSIS USING CONSTRAINT DATABASES

M. T. Gómez-López, R. M. Gasca and C. del Valle *,1

* *Escuela Técnica y Superior de Ingeniería Informática de Sevilla, Spain*
{*mayte, gasca, carmelo*} *@lsi.us.es*

Abstract: In model-based diagnosis, minimal hitting sets are usually used to identify which components may fail in a system. This work presents a set of algorithms to improve the determination of all Minimal Hitting Sets. Our proposal uses the minimal conflict sets to obtain, in an efficient way, the diagnosis of a system. The improvement consists of three algorithms which analyse only the relevant options. This proposal builds an equivalent system in order to obtain all minimal hitting sets depending on the location of the sensors. At the same time, all the information of the process is stored in a Constraint Database, keeping the information persistent and recoverable. Some empirical results are presented in order to show how the proposal improves the process in order to obtain all minimal hitting sets.*Copyright*

Keywords: Model-based Diagnosis, Constraint Databases, minimal hitting sets.

1. INTRODUCTION

A lot of theoretical and practical problems can be partly reduced to an instance of the minimal hitting set problem, as in model-based diagnosis how it is explained in (Reiter, 1987). Model-based diagnosis allows the identification of the parts which may fail in a system using a model. The models are based on the knowledge of the system to be diagnosed, and it can be represented by constraints associated to the components. Inputs and outputs of components are represented as variables in the constraints, and they can be observable and non-observable, depending on the allocation of the sensors. In function of the value of these sensors, it is possible to know which groups of components are failing looking for the minimal hitting sets. In this work, a new approach is proposed in order to automate and to improve the determination of all minimal hitting sets.

There are a significant amount of works that deals with minimal hitting sets, but most of them are only interested in finding single minimal hitting sets and for special types of problems. Our motivation is to find all minimal hitting sets, single or multiple. Another advantage of our technique is the possibility to store all the diagnosis process information using the relational calculus.

One of our previous works (Gómez-López *et al.*, 2004) shows how to obtain the possible minimal conflict sets in an efficient way. This paper starts at this point, and some algorithms are developed and implemented to obtain the minimal diagnosis. In this paper, we present how some issues concern-

[1] This work has been funded by the Ministerio de Ciencia y Tecnología of Spain (DPI2003-07146-C02-01) and the European Regional Development Fund (ERDF/ FEDER).

ing Constraint Databases (CDBs) can improve the efficiency in some stages of the model-based diagnosis. A novel methodology is presented in order to promote the advantages of this technology and its usage in industrial diagnosis problems.

Moreover the relational model is not able to represent the scientific and engineering data. For this reason, we consider CDBs as a good tool for our reasoning process. There are lots of works, as (Revesz, 2002) and (Kuper *et al.*, 1998), that utilise CDBs, but neither of them uses them for diagnosis. CDBs permit to treat and store the continuous information, as the behaviour of a system. It allows the use of all the power of relational databases in model-based diagnosis.

Our paper has been organised as follows: Section 2 presents other relevant proposals. Section 3 reviews some definitions in order to introduce the model-bases diagnosis. Section 4 explains how it is possible to take advantage of CDBs in the determination of minimal hitting sets. Section 5 describes the algorithms to improve the diagnosis, showing some empirical results. Finally the conclusions are presented.

2. BACKGROUND

Reiter (Reiter, 1987) presented one of the first solutions for the determination of minimal hitting sets using the so-called HS-trees. The problem with complete HS-trees is that the size of the tree grows exponentially with the size of the collection of sets. Minimal hitting sets can be efficiently obtained by a pruned HS-tree, but the construction of pruned HS-trees is complex due to unnecessary results are generated. The HS-tree was revised by Greiner (Greiner *et al.*, 1989) into an HS-DAG using a graph. Among the solutions that use trees, it is possible to find the BHS-trees that compute the hitting sets with a binary-tree, or the Wotawa's proposal (Wotawa, 2001), where the HST-tree algorithm is presented. The HST-tree algorithm constructs the tree in a unique way, where nodes are ordered from left to right depending on the size of their edge labels.

Other previous works have been centered in circuits diagnosis, as (Hou, 1994) or (Stumptner and Wotawa, 1997). (Hou, 1994) shows how to avoid visiting all subsets. The problem was that this proposal skips over some possible solutions, and it was corrected in (Han and Lee, 1999), where the number of subsets required to be explored was also reduced. The algorithm called structured-based abduction (SAB) of (Fattah and Dechter, 1995) is improved in (Stumptner and Wotawa, 1997), descending into the tree to find the possible combinations which can be responsible of an incorrect behaviour.

The determination of minimal hitting sets has also been approached without using trees. (de la Banda *et al.*, 2003) presents algorithms for the determination of all the minimal unsatisfiable subset of constraints. They use preprocessing steps to reduce the size of the set of constraints, allowing them to solve larger problems than other techniques.

3. DEFINITIONS

In order to obtain the minimal hitting sets, some definitions are necessary:

Definition 1. *Context Set (CS)*: Any subset of components which compose the system. There are $2^{ncomp} - 1$ possible CSs, where $ncomp$ is the number of components of the system.

Definition 2. *Context Network (CN)*: A graph formed by all the CSs according to the way proposed by ATMS (Kleer, 1986).

Definition 3. *Context Analytical Redundancy Constraint (CARC)*: A constraint derived from the system, in such a way that only observable variables (variables with sensors) are related.

Definition 4. *Observational Model (OM)*: A set of values for the observable variables.

Definition 5. *Possible Minimal Conflict Contexts (PMCC)*: CSs which have one or more than one $CARC$ associated, where its subcontexts are not $PMCCs$.

Definition 6. *Minimal Conflict Context (MCC)*: A $PMCC$ whose $CARCs$ are unsatisfiable for an OM. The $MCCs$ help us to find out which components are failing.

Definition 7. *Hitting Set (HS)* for a collection of sets $\mathcal{C}$ is a set $\mathcal{H} \subseteq \bigcup_{\mathcal{S} \in \mathcal{C}} \mathcal{S}$ such that $\mathcal{H}$ contains at least one element for each $\mathcal{S} \in \mathcal{C}$. A HS of $\mathcal{C}$ is minimal iff no proper subset of it is a HS of $\mathcal{C}$. The minimal HSs for a set of $MCCs$ are formed by $\{\mathcal{H}_1, \mathcal{H}_2, \ldots \mathcal{H}_n\}$, where $\mathcal{H}_i$ is a minimal HS of components. The cardinality of $\mathcal{H}_i$ ($|\mathcal{H}_i|$) is the number of components of $\mathcal{H}_i$.

In order to know the MCC, it is necessary to obtain the $CARCs$ of the system. Our proposal uses Gröbner Bases theory (Buchberger, 1985), which is the origin of many symbolic algorithms used to manipulate multiple variable polynomials. The main idea is to have the set of equality polynomial constraints of the form $P = 0$, Gröbner bases theory produces an equivalent system $G = 0$ which has the same solution as the original one, but substituting some variables for others.

In order to understand the Gröbner Bases' technique better, Figure 1 shows a well-know example, where M_i are multipliers, A_i are adders, and

$\{a, b, c, d, e, f, g\}$ are the observable variables. In this case, using Gröbner Bases, an equivalent set of constraints is obtained:

- CARC−1 $\{f = a * c + b * d\}$: Generated from the constraints of the components $\{M_1, M_2, A_1\}$
- CARC−2 $\{g = b * d + c * e\}$: Generated from the constraints of the components $\{M_2, M_3, A_2\}$
- CARC−3 $\{f - g = a * c - c * e\}$: Generated from the constraints of the components $\{M_1, M_3, A_1, A_2\}$

If a $CARC$ is unsatisfiable for an OM, it means than one or more than one component related with this $CARC$ is wrong.

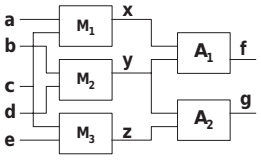

Fig. 1. A Well-known example of Model-based Diagnosis

4. CONSTRAINT DATABASES

One of the difficulties in model-based diagnosis is handling all the information, making the information persistent and recoverable. In this paper we propose to store all the information in a Relational CDB, facilitating the diagnosis process. It will allow us to store partial results and improve the diagnosis task.

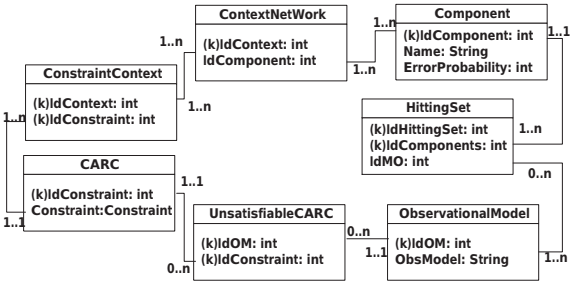

Fig. 2. Tables of the Constraint Database

In the Figure 2, the tables of the CDB are shown, and how the information is stored. The semantics of these tables are:

(1) **Component:** This table contains the names, the identifiers and the probability of error of each component.
(2) **ContextNetwork:** This table represents the CSs relating the Components. The table has potentially $2^{ncomp} - 1$ combinations of elements, but all of them do not have to be studied.
(3) **CARC:** This table stores all the constraints derived from the system.
(4) **ConstraintNetwork:** This table relates the CSs and the $CARCs$. For example, the context $\{M_1, M_2, A_1\}$ has the $CARC$ $\{f = a * c - b * d\}$ associated.
(5) **ObservationalModel:** This table contains the values of all the observable variables.
(6) **UnsatisfiableCARC:** This table stores what $CARCs$ fail for each OM.
(7) **HittingSet:** This table stores the components related to the minimal HSs for an OM.

The use of CDBs in the diagnosis process is a good decision because there are characteristics of model-based diagnosis related to the relational calculus used in relational databases. For example, the components are related among them using variables, the CSs and the HSs are formed by components, the CSs can have associated $CARCs$ ... All these relations can be represented and treated in an easy way using CDBs. Also, it is possible to store all the possible failures of the $CARCs$. It means that the determination of the minimal HSs will be linear in function of the number of $CARCs$. Fist of all, we can think that it is necessary to store all the possible values of OMs. But it is not true, because we only need to store the different types of failures. For the example shown in Figure 1, there are only three types of $CARCs$' failures for whatever OM: $\{CARC-1; CARC-3\}$, $\{CARC-2; CARC-3\}$ or $\{CARC-1; CARC-2; CARC-3\}$, other options are impossible. This process can be off-line and the diagnosis for an OM can be on-line and immediate.

5. THE IMPROVEMENTS

In this Section, we propose some improvements to avoid the study of all minimal HSs once we have the $MCCs$ for an OM. The $PMCCs$ are only created once and stored in the CDB. In order to obtain the $MCCs$, the algorithms described in (Gómez-López *et al.*, 2004) are used, where only the relevant contexts are analysed.

In order to determine all the minimal HSs in an efficient way, we build a bidimensional table that represents the relations between the components and the $MCCs$ *(tableCompsMCCs)*. If the CDB had not been used, the obtaining of the information would be much more difficult.

The *tableCompsMCCs* for the example of Figure 1 is shown in Table 1, where the MCC-i is related to the $CARC$-i. For this example, all the $PMCC$ are $MCCs$, it means that all the $CARCs$ are unsatisfiable for an OM. The table has *1* in the position {i, j} if the MCC-i is related to the component j, and *0* in otherwise.

	M_1	M_2	M_3	A_1	A_2
MCC-1	1	1	0	1	0
MCC-2	0	1	1	0	1
MCC-3	1	0	1	1	1

Table 1. Table of Figure 1

5.1 First Improvement: Searching the Representative Components

In this subsection, we will study how to determine equivalent components, proposing the following definitions:

Definition 8. *MCCs affected by a component c (MCCsAffect(c))* are the set of $MCCs$ that can fail when the behaviour of component c is wrong.

Lemma: If a component A affects the same $MCCs$ than another one B, it means that if there exists a minimal HS M such as $M = A \cup N$ (N is a set of components where neither A nor B is included in N) another minimal HS $M' = B \cup N$ exists. In this case, it is not necessary to study both possibilities, only one.

Proof: Let us reason by contradiction. If M is a minimal HS, it means that A affects some $MCCs$ which are not affected by neither component of N. If B does not affect the same components than A, it means that B could affect more $MCCs$ than A (M' would not be minimal), less or different $MCCs$ than A (M' would not be a HS). Contradiction.

Definition 9: *Group of Equivalent Components (GEC)* is a set of components which affects the same $MCCs$. Therefore, it is only necessary to study one component for each GEC.

If all the possible minimal HSs are studied, it will be necessary to analyse $2^{ncomp} - 1$ possibilities. However, using this improvement, the number of possible minimal HSs can be reduced. Figure 3 shows for several random examples, the percentage of the possibilities eliminated, studying only one component of each GEC.

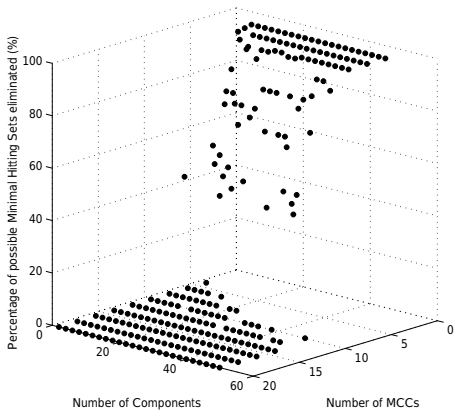

Fig. 3. Percentage of possibilities eliminated

When the percentage is zero, it means that all the $GECs$ have only one component, and there are not components which affect the same $MCCs$. Figure 3 also shows how the percentage of eliminations decreases when the number of $MCCs$ increases. It can be explained, due to if there are few components and a lot of $MCCs$ will be less probable that two components affect the same $MCCs$.

5.2 Second Improvement: Searching Single Minimal HSs

Before starting the analysis of the next proposal, a new case of study is going to be introduced, because the example shown in Figure 1 is not big enough to see all the types of problems that can happen in real systems. The Table 2 shows the *ComponentsMCCs* table for the new example. There are 8 $GECs$: {{s_0}, {s_1}, {s_2, s_3, s_7, s_8, s_{11}}, {s_4}, {s_5, s_9, s_{12}, s_{14}}, {s_6}, {s_{10}, s_{13}}, {s_{15}}}.

	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}
MCC-1	0	0	0	0	1	1	1	0	0	1	1	0	1	1	1	1
MCC-2	1	1	1	1	1	0	1	1	1	0	1	1	0	1	0	1
MCC-3	0	1	1	1	1	1	0	1	1	1	0	1	1	0	1	1
MCC-4	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
MCC-5	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1

Table 2. Table Example

There is another solution (de la Banda *et al.*, 2003), that looks for minimal single HSs at the beginning of the HSs determination. This solution determines the HSs of one component studying if there is a component c such as $\{\mathcal{C} - c\}$ is satisfiable, where $\mathcal{C}$ is the set of unsatisfiable constraints. Our solution does not need to study the satisfiability of the constraints, only determines if there is a component which affects all the $MCCs$. Thereby if a column j only has 1s, it means that the component represented by the column j affects every $MCCs$ and it forms a minimal HS of only one component (single minimal HS). Therefore this component will not be involved in any other minimal HSs. In our example there is a single minimal HS ($\{s_{15}\}$). Therefore, our problem has been reduced from 16 to 7, and the *representative components* are: $\{s_0, s_1, s_2, s_4, s_5, s_6, s_{10}\}$.

5.3 Third improvement: To avoid studying redundant or unpromising combinations of components

Before continuing the study of minimal HSs, it is possible to know the greatest number of components of the minimal HSs of a system (Max ($|\mathcal{H}_i|$)). Using the example of Table 2 again, it is possible to know that the biggest size of any

minimal HS for the example is 3, because the components that less affect the $MCCs$ are s_0, s_1 and s_6. For example s_0 does not affect MCC-1 or MCC-3, meaning that in the worst case only two components more would be necessary.

For this reason, it is possible to guarantee that:

Let $\mathcal{H} = \{h_1, \ldots, h_x\}$ be hitting sets and
$\mathcal{C} = \{c_1, \ldots, c_n\}$ be components then
Max($|h_i|$) =
Number of MCCs−Min(|MCCsAffected(c_j) |) +1
for i in 1 ... x, j in 1 ... n

Starting with the *representative components* of each GEC, we need to combine these components to know if they make up a minimal HS, but these combinations will not be a blind search. Let $\{r_1, r_2, \ldots, r_n\}$ be the representative components, the following pseudo-code helps to store in the variable $\mathcal{H}$ all the minimal HSs of two components, and in the queue $\mathcal{Q}$ the couples of promising components which are not HSs will be stored.

For each couple of components $\{r_i, r_j\}|i < j$.
If $\{r_i, r_j\}$ is a minimal HS then
$\mathcal{H} := \mathcal{H} \cup \{r_i, r_j\}$
else
If ¬(MCCsAffect(r_i)⊂MCCsAffect(r_j)) AND
¬(MCCsAffect(r_j) ⊂ MCCsAffect(r_i)) then
$\mathcal{Q} := \mathcal{Q} \cup \{r_i, r_j\}$
endif
endif
endforeach

If $\{r_i, r_j\}$ *is not a minimal HS*, but these two components could participate in a minimal HS with more components, they will be included in $\mathcal{Q}$ for future searches. But if the $MCCs$ affected by a component are included in the influenced $MCCs$ for another component, it means that these two components will never participate in a minimal HS together, and they will not be added to $\mathcal{Q}$. If (MCCsAffect(r_i) ⊂ MCCsAffect(r_j)) or (MCCsAffect(r_j) ⊂ MCCsAffect(r_i)), it means that the behaviour of the component r_i is included in r_j. In order to generalise this idea, it is necessary the following definition.

Definition 10: *Behaviour of a Component f Included in other components* C=$\{c_i, \ldots, c_j\}$ (BehaviourIncluded(f,C)) iff the behaviour of the component f is included in the components c_i or ... or c_j.

In order to know if MCCsAffect(f) is included in MCCsAffect(c_i), we use the binary AND operator between the columns f and c_i. If (f AND c_i) = f, it means that the behaviour of f is a subset of the c_i behaviour. For example, the s_0 behaviour is included in s_2, so these components are never going to be together in a minimal HS.

In our example, some of the minimal HSs of two components are: $\{(s_0, s_4)$, (s_0, s_5), (s_1, s_4), (s_1, s_5), (s_1, s_{10}), (s_2, s_4), (s_2, s_5), (s_2, s_6), (s_2, s_{10}), (s_4, s_5), (s_4, s_6), (s_4, s_{10}), (s_5, s_6), $(s_5, s_{10})\}$. The rest of combinations of two components are added to the queue $\mathcal{Q}$, because perhaps they can form minimal HSs combining them with other components. For our example, the couples added to the queue are: $\{(s_0, s_1)$, (s_0, s_6), $(s_1, s_6)\}$.

The search will not end until $\mathcal{Q}$ is empty. With the set of components taken out from $\mathcal{Q}$, we try to create minimal HSs of n+1 components, where n is the number of components obtained from $\mathcal{Q}$. If these $n+1$ components are not a minimal HS and $(n+1)$=(Max ($|\mathcal{H}_i|$)), these will not be added to $\mathcal{Q}$.

In order to avoid possible non minimal HSs, the algorithm combines the components C taken out from the queue C =$\{c_i, \ldots, c_j\}$ with another $\{c_k\}$ if and only if $k > j$. The following algorithm is executed for each of the mentioned possibilities and the components obtained from the queue.

If(¬ BehaviourIncluded(c_k,C))
If (MinimalHittingSet(C, c_k))
$\mathcal{H} := \mathcal{H} \cup \{C \cup c_k\}$
else
$\mathcal{Q} := \mathcal{Q} \cup \{C \cup c_k\}$
endif
endif

In order to understand the above algorithm, it is necessary to explain the following method:

The MinimalHittingSet(Components C, Component c) method returns *true* if $C \cup c$ is a minimal HS, avoiding n-tuples like $\{s_0, s_1, s_4\}$. These three components are a HS, but not a minimal HS. The idea of the method is to determine if all the components of a set $\mathcal{C}$ are essential to form a minimal HS analysing each MCC. It means that if we set aside a component, the set of remaining components would not be a HS. For example, if we want to know if $\{s_0, s_1, s_4\}$ is a minimal HS, the Table 3 will be studied.

	s_0	s_1	s_4
MCC-1	0	0	1
MCC-2	1	1	1
MCC-3	0	1	1
MCC-4	1	1	0
MCC-5	1	0	1

Table 3. Example for minimal HS

In order to store the indispensable components, we will use a set $\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$, such that if $\mathcal{I}_i$ has one component, it means that this component is indispensable. But if $\mathcal{I}_i$ has several components, it means that one of these components are indispensable. In general the set $\mathcal{I}$ = $\{\{s_i \ldots s_j\}, \{s_k \ldots s_h\}, \ldots, \{s_l \ldots s_m\}\}$ means that the indispensable components are $\{(s_i \vee \ldots \vee s_j) \wedge (s_k \vee \ldots \vee s_h) \wedge (s_l \vee \ldots \vee s_m)\}$. For each MCC, $\mathcal{I}$ will be modified as follow:

- *If MCC-i is affected only by one component c:* It means that this component is indispensable to obtain a minimal HS. And $\mathcal{I}$ is updated as $\mathcal{I}{:=}\mathcal{I} \cup \{c\}$, where c is the indispensable component.
- *If MCC-i is affected by a set of more than one component called* $\mathcal{C}'$*:*
  - If $\forall \mathcal{I}_i :_{i:1\ldots n} \mathcal{I}_i \cap \mathcal{C}' \neq \emptyset \Rightarrow \mathcal{I}_i := \mathcal{I}_i \cap \mathcal{C}'$
  - If $\exists \mathcal{I}_i :_{i:1\ldots n} \mathcal{I}_i \cap \mathcal{C}' = \emptyset \Rightarrow \mathcal{I} := \mathcal{I} \cup \mathcal{C}'$
- *MCC-i is affected by all the components* $\mathcal{C}$*:* Neither of them are indispensable.

If after the study of all $MCCs$, $\mathcal{I} = \mathcal{C}$, it means that $\mathcal{C}$ is a minimal HS. In other cases $\mathcal{C}$ would not be a minimal HS. The trace for the example shown in Table 3 is:

for MCC-1 : $\mathcal{I} = \{s_4\}$
for MCC-2 : $\mathcal{I} = \{s_4\}$ neither component is added because MCC-2 is affected by all components
for MCC-3 : $\mathcal{I} = \{s_4\}$
for MCC-4 : $\mathcal{I} = \{s_4\}, \{s_0, s_1\}$
for MCC-5 : $\mathcal{I} = \{s_4\}, \{s_0\}$

As $\mathcal{I}$ does not have $\{s_0, s_1, s_4\}$, it means that not all components are indispensable, therefore $\{s_0, s_1, s_4\}$ is not a minimal HS.

With all these improvements the number of possible minimal HSs is reduced in an important way. Figure 4 shows the percentage of the possibilities eliminated with this technique, related to the first improvement. And Figure 5 shows the computational time for several random examples, with different number of components and $MCCs$.

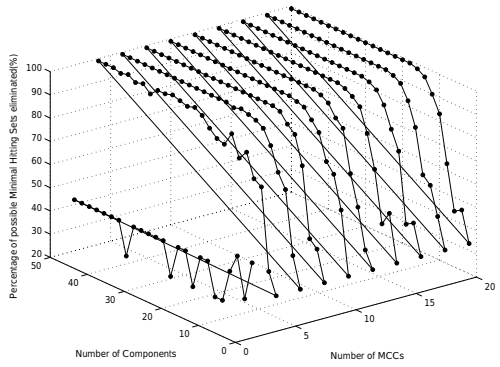

Fig. 4. Percentage of possibilities eliminated

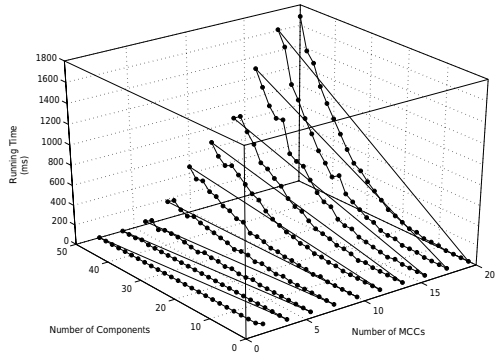

Fig. 5. Execution time(Pent. IV, 512 M memory)

6. CONCLUSIONS

Our work proposes to build an equivalent system with observable variables, which will be generated once for each system and validated for each OM. Moreover, three improvements have been presented in order to avoid the study of unpromising combinations of components.

In order to store all the information, we use a CDB. It allows us to obtain and store the partial data and all the possible combination of $MPCCs$ and the minimal HSs. It makes possible to obtain a diagnosis in linear time in function of the number of $CARCs$, because the obtaining of the minimal HSs can be an off-line process. In run time only will be necessary to validate each $CARC$ to know what $MCCs$ are failing, and to query the database about the minimal HSs.

REFERENCES

Buchberger, B. (1985). Gröbner bases: An algorithmic method in polynomial ideal theory. D. Reidel Publishing Co.. pp. 184–232.

de la Banda, M. G., P. J. Stuckey and J. Wazny (2003). Finding all minimal unsatisfiable subsets. In: *PPDP '03*. ACM Press. pp. 32–43.

Fattah, Yousri El and Rina Dechter (1995). Diagnosing tree-decomposable circuits.. In: *IJCAI*. pp. 1742–1749.

Gómez-López, M. T., R. Ceballos, R. M. Gasca and C. Del Valle (2004). Constraint databases technology for polynomial models diagnosis.. In: *15th International Workshop on Principles of Diagnosis*. pp. 215–220.

Greiner, R., B. A. Smith and R. W. Wilkerson (1989). A correction to the algorithm in reiter's theory of diagnosis. Vol. 41. Elsevier Science Publishers Ltd. pp. 79–88.

Han, B. and S. Lee (1999). Deriving minimal conflict sets by cs-trees with mark set in diagnosis from first principles. Vol. 29-2.

Hou, Aimin (1994). A theory of measurement in diagnosis from first principles. Vol. 65. Elsevier Science Publishers Ltd.. pp. 281–328.

Kleer, J. De (1986). An assumption-based truth maintenance system. Vol. 2. pp. 127–161.

Kuper, G., L. Libkin and J. Paredaes (1998). *Constraint Databases*. Springer.

Reiter, R. (1987). A theory of diagnosis from first principles. Vol. 1. pp. 57–96.

Revesz, P. (2002). *Introduction to constraint databases*. Springer-Verlag New York, Inc.. New York, NY, USA.

Stumptner, Markus and Franz Wotawa (1997). Diagnosing Tree-Structured Systems. Nagoya, Japan.

Wotawa, Franz (2001). A variant of reiter's hitting-set algorithm.. Vol. 79. pp. 45–51.