A Topological-Based Method for Allocating Sensors by Using CSP Techniques

R. Ceballos, V. Cejudo, R. M. Gasca, and C. Del Valle

Departamento de Lenguajes y Sistemas Informáticos,
Universidad de Sevilla (Spain)
{ceballos, cejudo, gasca, carmelo}@lsi.us.es

Abstract. Model-based diagnosis enables isolation of faults of a system. The diagnosis process uses a set of sensors (observations) and a model of the system in order to explain a wrong behaviour. In this work, a new approach is proposed with the aim of improving the computational complexity for isolating faults in a system. The key idea is the addition of a set of new sensors which allows the improvement of the diagnosability of the system. The methodology is based on constraint programming and a greedy method for improving the computational complexity of the CSP resolution. Our approach maintains the requirements of the user (detectability, diagnosability,...).

1 Introduction

Model-based diagnosis (MBD)[1][2] allows to determine why a correctly designed system does not work as expected. In MBD, the behaviour of components is simulated by using constraints. Inputs and outputs of components are represented as variables of constraints. These variables can be observable and non-observable depending on the sensors allocation. The objective of the diagnosis process is to detect and identify the reason for any unexpected behaviour, and to isolate the parts which fail in a system.

The diagnosability of systems is a very active research area in the diagnosis community. A toolbox integrating model-based diagnosability analysis and automated generation of diagnostics is proposed in [3]. The proposed toolbox supports the automated selection of sensors based on the analysis of detectability and discriminability of faults. In this line, a methodology to obtain the diagnosability analysis using the analytical redundancy relations (ARR) was proposed in [4]. This approach is based on an exhaustive analysis of the structural information. The objective is the addition of new sensors to increase the diagnosability. In a previous work [5] a methodology to analyze the diagnosability of a system based in a process algebras was proposed. A framework for testing the diagnosability of a system is defined by using the available sensors, a model abstraction, and some snapshots of sensor readings.

In this work, a new approach is proposed in order to improve the computational complexity for isolating faults in a system. Our approach is based on the addition of new sensors. A constraint satisfaction problem (CSP) is obtained in

order to select the necessary sensors to guarantee the problem specification. We propose an algorithm for determining the bottleneck sensors of the system in order to improve the computational complexity of the CSP. A CSP is a framework for modeling and solving real problems as a set of constraints among variables. A CSP is defined by a set of variables $X=\{X_1, X_2,..., X_n\}$ associated with a set of discrete-valued, $\mathcal{D}=\{\mathcal{D}_1, \mathcal{D}_2,..., \mathcal{D}_n\}$ (where every element of $\mathcal{D}_i$ is represented by set of v_i), and a set of constraints $C=\{C_1, C_2,..., C_m\}$. Each constraint C_i is a pair (W_i, R_i), where R_i is a relation $R_i \subseteq D_{i1} \cdot ... \cdot D_{ik}$ defined in a subset of variables $W_i \subseteq X$.

The remainder of the paper is organized as follows. Section 2 provides the definitions and notation in order to clarify MBD concepts. Section 3 introduces the basis of our approach. Section 4 describes the CSP generation. Sections 5 shows the greedy method for improving the CSP resolution. Finally, conclusions are drawn and future work is outlined.

2 Notation and Definitions

In order to explain our methodology, it is necessary to establish some concepts and definitions from the model-based diagnosis theories.

Definition 1. A *System Model* is a finite set of equality constraints which determine the system behaviour. This is done by means of the relations between the non-observable and observable variables (sensors) of the system.

Definition 2. A *Diagnosis* is a particular hypothesis that shows the system differs from its model. Any component could be working or faulty, thus the diagnosis space for the system initially consists of 2^{nComp} - 1 diagnoses [2], where *nComp* is the number of components of the system. The goal of diagnosis is to identify and refine the set of diagnoses.

Definition 3. The *Discriminability Analysis* [3] determines whether and under which circumstances the considered (classes of) faults can be distinguished.

Definition 4. The *Diagnosability level* is the quotient of the number of the (classes of) faults which can be distinguished each other, and the number of all the possible faults. The size of the possible faults is initially 2^{comp} - 1.

Definition 5. A set of components T is a *Cluster of components* [6], (i) if it does not exist a common non-observable variable of any component of the cluster with any component outside the cluster, and (ii) if for all $Q \subset T$ then Q is not a cluster of components.

All common non-observable variables between components of the same cluster belong to the cluster, therefore, all the connections with components which are outside the cluster are monitored. A cluster of components is completely monitored, and for this reason the detection of faults inside the cluster is possible without any

information from other components which do not belong to the cluster. A more detailed explanation and the cluster detection algorithm appears in [6].

3 The Basis of the Algorithm

Our approach is based on the generation of new clusters of components by allocating sensors in some of the non-observable variables. These new clusters reduce the computational complexity of the diagnosis process since it enables the generation of the diagnosis of the whole system based on the diagnosis of the subsystems. Let C be a set of n components of a system, and C_1 and C_2 be clusters of n - m and m components such as $C_1 \cup C_2 = C$; then the computational complexity for detecting conflicts in C_1 and C_2 separately is lower than in the whole system C, since the number of possible diagnoses of the two clusters is $(2^{n\text{-}m}) + (2^m)$ - 2 $\leq 2^{n\text{-}m} \cdot 2^m$ - 2 which is less than 2^n-1.

The clustering process enables isolating the faults of the original system, since the multiple faults which include components of different clusters are eliminated. These kind of faults are transformed into single or multiple faults which belong to only one cluster. The computational complexity for detecting conflicts and discriminating faults in a system is always higher than for an equivalent system divided into clusters.

4 The CSP Problem Specification

The objective is to obtain the best allocation of sensors in order to generate new clusters. The allocation of the sensors will be formulated as a Constraint Satisfaction Problem (CSP). A CSP is a way of modeling and solving real problems as a set of constraints among variables.

The methodology was applied to the 74181 4-Bit ALU. It is one of the ISCAS-85 benchmarks [7]. It includes 61 components, 14 inputs and 8 outputs. Table 1 shows the set of variables and constraints for determining the number and location of sensors for this example. The following variables are included:

- *nNonObsVar*: This constant-variable holds the number of non-observable variables.

Table 1. CSP for the 74181 ALU sensors allocation

Variable (= initial value)	**Domain**
(1) nSensors = {free}	$\mathcal{D}$={1, ..., nNonObsVar}
(2) clusterOfComp$_i$ = {free}	$\mathcal{D}$={1, ..., nComp}
(3) clusterDist$_t$ = {free}	$\mathcal{D}$={1, ..., nComp}
(4) sensor$_k$ = {free}	$\mathcal{D}$={true, false}
Constraints	
(5) if (sensor$_{E01}$ = false) $\Rightarrow$ clusterOfComp$_{M11}$ = clusterOfComp$_{M32}$	
(6) if (sensor$_{E02}$ = false) $\Rightarrow$ clusterOfComp$_{M19}$ = clusterOfComp$_{M32}$	
...	

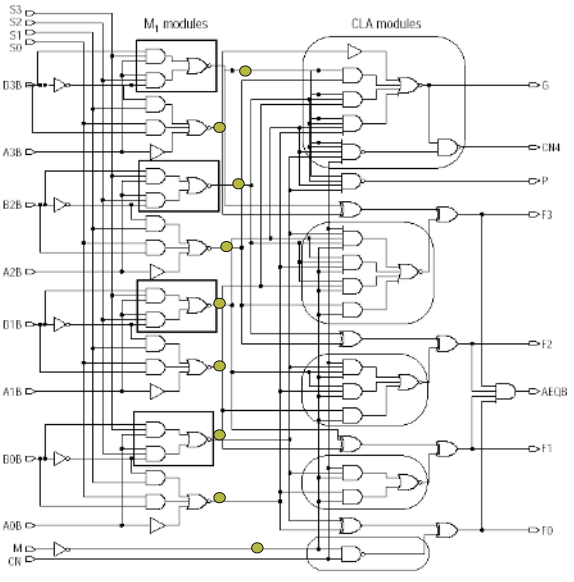

Fig. 1. 74181 ALU

- *nSensors*: This variable holds the number of new sensors. It must be smaller than the number of non-observable variables.
- $sensor_k$: This set of variables represents the possible new sensors of the system. They hold a boolean value in the interval {true, false}, where *true* implies that there must be a sensor, and *false* the opposite.
- $clusterOfComp_i$: This set of variables represents the cluster associated to each component i.
- $clusterDist_t$: This set of variables holds the number of components included in each cluster t.

For each common non-observable variable between two components a constraint is generated which guaranties that if there is not a sensor, the two components must belong to the same cluster. Table 1 shows the constraints (5),(6),... that hold this kind of information, and it is based on Figure 1. The final sensors allocation is stored in $sensor_k$, and the distribution of the clusters is stored in $clusterDist_i$. The optimization problem can have different objectives, depending on the user and the problem requirements. Two typical goals can be:

- To minimize the number of sensors (if the number of clusters is fixed).
- To minimize the maximal number of components in each cluster (if the maximal number of sensors is fixed).

It is possible to add other constraints in order to guarantee some properties of the solution. For example, in order to guarantee prices, to respect requirements of the customers, to store incompatibilities, to specify problems, ... We

have applied the limited discrepancy search (LDS) [8] algorithm in order to search the solution. This algorithm is based on the limitation of the number of discrepancies.

5 Improving the Algorithm: A Greedy Method

The computational complexity of a CSP is exponential in general. We propose a method to obtain the most important allocation of the new sensors in order to generate more clusters; that is, the bottlenecks of the system. Our method has two phases:

1. The calculation of the minimal paths: A graph where the nodes represent the components of the system, and the edges represent the connections between each two components (non-observable variables). Each edge has a weight calculated as the number of common non-observable variables between two components. By applying the Floyd's algorithm (dynamic programming), all the shortest paths between all pairs of nodes is stored.
2. In order to determine which are bottlenecks of the system, each minimal path will vote which sensors are more important. Figure 2 shows this algorithm.

```
sensorsOrder(P)
  componentVotes[nComp][nNonObsVar]
  sensorVotes[nNonObsVar]
  // All the components (1..nComp) votes the variables (sensors)
  // associated to the minimal paths
  forEach j between 1 to nComp
    forEach P_k from component i to component j
      forEach q between 1 to length(P_k)
        △ = (voteValue / (length(P_{k,i,j}) · length(P_k))
        forEach v includes in path[q]
          componentVotes[i][P_{k,i,j,q}] += △
        endForEach
      endForEach
    endForEach
  endForEach
  // Recounting of votes for each sensor
  forEach sensor_j between 1 to nSensors
    sensorVotes[j] = 0
    forEach i between 1 to nComp
      sensorVotes[j] += componentVotes[i][j] / ( △ · nComp )
    endForEach
  endForEach
  return sort(sensorVotes)
```

Fig. 2. Algorithm for obtaining the bottleneck sensors of the system ($O(n^2 \cdot m^2)$, where n is the number of components and m is the number of non observable variables)

Each minimal path will vote for the included non-observable variables of the minimal path. The number of votes are scaled in order to guarantee that each component generates the same total number of votes. These votes allow to generate a sorted list of non-observable variables. This list is composed of the most relevant sensors with the aim of generating new clusters.

The bottlenecks of the system represent the best sensors in order to isolate components and faults. The sorted list of sensors enables creating a CSP with less variables to find the solution of the problem in a limited time. Only the solutions included in the combinations of the m bottleneck sensors will be tested, and therefore, the number of possible solutions will be lower than 2^m. The optimal solution is not guaranteed, but the reduction of computational complexity enables finding a solution in a limited time.

Example: In the *Alu74181* example the most important sensors are (based on the number of votes): $E_{02}(930)$, $E_{03}(878)$, $X_{28}(773)$, $E_{01}(737)$, $E_{00}(583)$, $D_{00}(514)$, $D_{01}(463)$, $D_{02}(326)$, $D_{03}(301)$,... The other sensors have less than 166 votes. The first 9 sensors are represented by shaded circles in Figure 1. The possible diagnoses in the system are 2^{61} - 1. By using the first 9 selected sensors, the number of clusters is 17 (all with less than 6 components) and the computational complexity is reduced because of the reduction of possible diagnosis to less than 2^9.

6 Conclusions and Future Work

The objective of our approach is the allocation of a set of new sensors in order to improve the computational complexity and diagnosability of a system. The methodology was applied to an standard example, and the results are very promising. It is based only on topological properties. This enables applying this approach to different kinds of systems. As a future work, we are working on new greedy methods to improve the votes counting.

Acknowledgements

This work has been funded by the Spanish Ministry of Science and Technology (DPI2003-07146-C02-01) and the European Regional Development Fund.

References

1. Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence 32 **1** (1987) 57–96
2. de Kleer, J., Mackworth, A., Reiter, R.: Characterizing diagnoses and systems. Artificial Intelligence **2-3**(56) (1992) 197–222
3. Dressler, O., Struss, P.: A toolbox integrating model-based diagnosability analysis and automated generation of diagnostics. In: DX03, 14th International Workshop on Principles of Diagnosis, Washington, D.C., USA (2003) 99–104

4. Travé-Massuyés, L., Escobet, T., Spanache, S.: Diagnosability analysis based on component supported analytical redundancy relations. In: 5th IFAC Symposium on Fault Detection, EEUU (2003)
5. Console, L., Picardi, C., Ribaudo, M.: Diagnosis and diagnosability analysis using PEPA. In: ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence, Berlin, Germany, IOS Press (2000) 20–25
6. Ceballos, R., Gómez-López, M.T., Gasca, R., Pozo, S.: Determination of Possible Minimal Conflict Sets using Components Clusters and Grobner Bases. In: DX04, 15th International Workshop on Principles of Diagnosis, Carcassonne, France (2004) 21–26
7. Hansen, M.C., Yalcin, H., Hayes, J.P.: Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering. IEEE Design and Test of Computers **16**(3) (1999) 72–80
8. Harvey, W.D., Ginsberg, M.L.: Limited discrepancy search. In: Fourteenth IJCAI, Montreal, Canada, IOS Press (1995)