# Distributed Constraint Propagation for Diagnosis of Faults in Physical Processes

Ana L. C. Bazzan and Bruno C. da Silva*
Instituto de Informática, UFRGS
Caixa Postal 15064, CEP 91.501-970 Porto Alegre, RS, Brazil
{bazzan,bcs}@inf.ufrgs.br

## ABSTRACT

Most of the current research on distributed diagnosis in and for multiagent systems focuses on diagnosis of coordination failures. Proposed approaches for this problem are not efficient for diagnosing failures in physical devices. This paper proposes algorithms for distributed troubleshooting of physical devices and processes. The consequences of using distributed representation of the knowledge, ATMS, and distributed reasoning are discussed and algorithms are proposed to deal with the occurrence of conflicts and the computation of the set of candidate diagnosis.

## Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence
*Intelligent agents, Multiagent systems*

## General Terms

Algorithms

## Keywords

Model Based Diagnosis, Distributed Diagnosis

## 1. INTRODUCTION

Diagnosis reasoning is a well known problem in Artificial Intelligence. An effective way to do it is to use a description of the system, as for instance physical properties and laws. This description together with the behavior constitute a model. Model-based diagnosis (MBD) infers abnormalities of internal components of a system given the behavior of the system's input(s) and output(s).

In this paper we focus on troubleshooting of physical devices or processes. These are likely to involve a high number of components. However, the diagnosis problem framed in centralized way is computationally prohibitive: in [3], it is suggested that only the most probable candidates should be computed. Thus, due to the size and complexity of the biological processes, physical devices or industrial plants, it makes sense to do *diagnosis in a distributed way*. Hints of this can be found in the practice of industrial automation (which is inherently distributed due to the modern, distributed control systems).

There is a key difference between the centralized and the distributed versions of that problem. In the former, given the designer's knowledge of the device, it is more or less straightforward to compute the faulty values. In the latter case we cannot count on any agent knowing all inputs and outputs of the system. In traditional centralized MBD of hardware devices, symptoms are easily detected from the designer's knowledge. A symptom is any difference between a prediction and an observation, whereas a candidate is a hypothesis to explain the difference between the actual and the expected value. Symptoms are used to guide the computation of conflicts and candidate sets. Henceforth the term model-based diagnosis of processes (MBDP) is used for this class of problems.

## 2. DIAGNOSIS OF PHYSICAL DEVICES

MBD is a natural way of dealing with troubleshooting in devices because the model normally mirrors the causal structure of components [2, 3, 4]. Due to lack of space we omit the details of these approaches, as well as the description of research on diagnosis of lack of coordination, and refer the reader to a previous version of this paper [1].

Following the terminology used in [4], a diagnosis $\mathcal{D}$ of $(SD, OBS, COMP)$ is a set $\Delta \subseteq COMP$ such that $SD \cup OBS \cup \{AB(c)|c \in \Delta\} \cup \{\neg AB(c)|c \in COMP - \Delta\}$ is consistent. $SD$, $OBS$, and $COMP$ are the set of formulae describing the system, the observations, and the components of the system respectively.

The algorithms shown here are based on the method proposed in [3]: In summary, one must define an inference strategy $\mathbf{C}(OBS, ENV)$ which, given the set of observations and a set of assumptions (an environment), determines whether or not they are consistent. The method starts with an empty environment and inserts one parent at time. If an environment is inconsistent, then it is a minimal conflict and the supersets are not explored. Otherwise it is enlarged by using other combinations of components. In order to make the inference, strategy $\mathbf{C}$ is implemented using a function $\mathbf{P}(OBS, ENV)$ which returns all behavioral predictions following $OBS$ given the assumptions in $ENV$. $\mathbf{P}$ computes the outputs and/or inputs of component(s) *given that these*

*Authors partially supported by CNPq–Brazil


*AAMAS'07,* May 14–18, 2007, Honolulu, Hawai'i, USA.




978-81-904262-7-5 (RPS) 

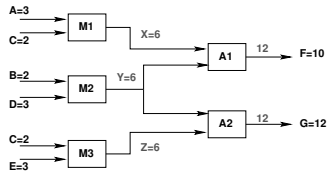


**Figure 1: Circuit Adder/Multiplier – Two Outputs**

*components are assumed as not faulty (not abnormal).* One output can be computed only if both inputs are known for sure; one input requires one output and the other input in order to be computed. If, for a given $ENV$, two different values of $\mathbf{P}$ are found, then $ENV$ is a conflict.

The classical example deals with the circuit shown in Figure 1: two adders and three multipliers. Outputs are measured at $G = 12$ and $F = 10$. The latter is faulty i.e. F is observed at 10, not 12 which is the correct value given the inputs. For the circuit in Figure 1, $\mathbf{P}(\{A = 3, B = 2, C = 2, D = 3\}, \{A1, M1, M2\})$ yields $\{A = 3, B = 2, C = 2, D = 3, X = 6, Y = 6, F = 12\}$. The minimal conflict set is computed: $\{\{A1, M1, M2\}, \{A1, A2, M1, M3\}\}$. In order to identify the minimal candidate set, here the basic approach by Reiter [4] is used, which is based on the computation of the *minimal hitting set* for the conflict set $CS$. The diagnosis is then: $\mathcal{D} = \{\{A1\}, \{M1\}, \{A2, M2\}, \{M2, M3\}\}$.

## 3. DISTRIBUTED DIAGNOSIS

In distributed MBDP, due to lack of global knowledge, no single agent can infer that a component is faulty. Thus either one assumes that agent $\mathcal{A}_i$ receives information regarding a symptom (e.g. F observed as 10, not 12), or this assumption is dropped and agents just try to find conflicts by exchanging values. Here no external communication regarding symptoms is made. Agents communicate to propagate the constraints (values of outputs of components).

### 3.1 Agents

Agents are represented by calligraphic letters: for example $\mathcal{M}_2$ is the agent in charge of $M2$, etc. Each agent $\mathcal{A}_i$ has a knowledge base $KB(i)$ to store the knowledge $K(i)$, where $K(i)$ denotes local information (assumed 100% trustworthy). Also computations performed by $\mathcal{A}_i$ over $K(i)$ are stored in $KB(i)$, as well as the list of agents related to $\mathcal{A}_i$. This relation is both physical (agents physically connected) and logical (agents inserted in a list called *agent_list* because they can help in the detection of conflicts and faults).

Both the knowledge and the beliefs are based on messages received from other agents. Values are computed locally from the knowledge of an agent about its component. Agents can perform the following actions: collect *input* and *output values* (only for agents in charge of input and output components respectively); compute output values (only if both inputs are known) and input value (given the output and one input); request knowledge and communicate values to agents in the agent_list; and communicate diagnosis (via broadcast to those in the agent_list).

### 3.2 Details of the Algorithms

Given a list of input and output components ($\mathcal{IN}$ and $\mathcal{OUT}$ respectively), assuming that each is associated with an agent, a list of components means in fact a list of agents. All components/agents are collected in the $\mathcal{AG}$ list. An important remark is that all agents can communicate concurrently. For examples of the computation of diagnosis using these algorithms see [1]. The schema of the approach is as follows:

**Algorithm 1** Computation of Conflicts, distributed version

given a list $\mathcal{AG}$ containing all agents
**for** $i \in \mathcal{AG}$ **do**
  initialize list $backward\_agents(i)$
  initialize list $forward\_agents(i)$
  $extra\_communications(i) = \{\}$
  $CS(i) = \{\}$ {// conflict sets detected by i}
  $structural\_knowledge(i) = \{i\}$ {// list of agents with whom $i$ has communicated}
  $K(i) = local\_observations(i)$
  $K(i) = P(i, K(i))$ {// expand local knowledge}
**end for**
**for** $i \in \mathcal{AG}$ **do**
  **for** $j \in backward\_agents(i)$ **do**
    COMM(i, j)
  **end for**
**end for**
**for** $i \in \mathcal{AG}$ **do**
  **for** $j \in forward\_agents(i)$ **do**
    COMM(i, j)
  **end for**
**end for**
**for** $i \in \mathcal{AG}$ **do**
  **while** $extra\_communications(i)$ not empty **do**
    j = i.pop()
    COMM(i, j)
  **end while**
**end for**
return $\bigcup_{CS(i)} \forall i \in \mathcal{AG}$

1. *representation of knowledge*: knowledge bases keep the variables values and constraints between variables;
2. *computation of conflicts* (Algorithms 1 and 2): this occurs in a finite number of cycles (in the worst case some agents have to communicate with all others). First each agent collects local knowledge to the agent_list (distinguishing between agents connected backwards and forwards, and expand the knowledge via the function $\mathbf{P}(OBS, ENV)$. Next communication with agents backwards and forwards occurs. During this, new agents can be added to a list (new_in_agent_list) as the list extra_communication grows. While these lists are not empty, this means that agent $i$ still has new relationships to explore. At the end of the communications, some agents have collected a list of conflicts.
3. *computation of candidates for diagnosis*: the hitting set is computed.

### 3.3 Larger Devices

In order to test the approach with devices larger than that in Figure 1, we use a generator of random adder–multiplier circuits. To compare performances, we use both the centralized and the distributed version of the algorithms. However,

**Algorithm 2** function COMM $(i, j)$ – (Communication for extension of knowledge between agents)

given: a function $enough\_conflicts(struc\_knowl, K)$ which determines whether all components in structural_knowledge belong to at least one conflict in K and a function $detect\_conflicts(K)$ which determines all agents involved in conflicting values in K
**if** not $enough\_conflicts(struc\_knowl(i), K(i))$ **then**
  $struc\_knowl(i) = struc\_knowl(i) \cup struc\_knowl(j)$
  $K(i) = K(i) \cup K(j)$
  $K(i) = P(struc\_knowl(i), K(i))$ {// expand knowledge}
  $CS(i) = CS(i) \cup detect\_conflicts(K(i))$
  **for** $k \in CS(i)$ **do**
    **if** $k! = i$ **then**
      **if** $k \notin backward\_agents(i)$ and $k \notin forward\_agents(i)$ **then**
        $extra\_comms(i) = extra\_comms(i) \cup k$
      **end if**
    **end if**
  **end for**
**end if**

since the runtime of the former is exponential in the number of components, it is not possible to make this comparison with a large number of them. We use here two metrics: rate of correct CS's (i.e. number of elements in the $CS$ yielded by the distributed algorithm divided by this number yielded by the centralized algorithm), and percentage of correct diagnosis (idem). Figure 2 depicts the correctly detected CS's, for 6 to 18 components in a device with two layers. Results are averages over 20 random circuits. Notice that the more faults, the higher the rate of successfully detected conflict sets. As the number of components increases, the percentage of correctly detected CS's decreases. In general, a rate of 75% correctly detected CS's is achieved with the increase of faulty outputs.

To give an idea of running times (cpu) of the algorithms, for 6 to 12 components in two layers, these range from 0.06 to 1.3 seconds and this is similar to the time required by the centralized algorithm. For larger number of components,

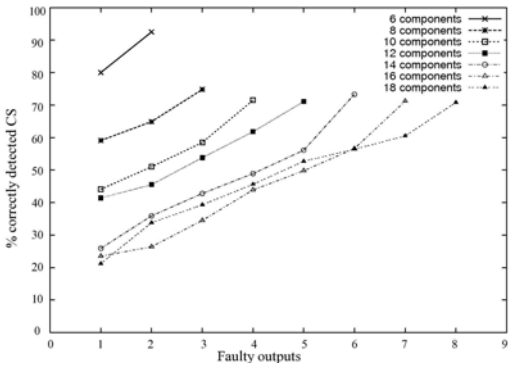


**Figure 2: Percentage of correctly detected conflict sets, for 6 to 18 components, varying the number of faulty outputs**

| l – c – o | % corr. HS | l – c – o | % corr. HS |
|---|---|---|---|
| 2 – 3 – 1 | 63 | 2 – 3 – 2 | 97 |
| 2 – 3 – 3 | 92 | 2 – 4 – 1 | 41 |
| 2 – 4 – 2 | 53 | 2 – 4 – 3 | 42 |
| 2 – 4 – 4 | 71 | 2 – 6 – 3 | 19 |
| 2 – 6 – 5 | 32 | 3 – 3 – 3 | 48 |

**Table 1: Percentage of correctly computed hitting sets, for different number of layers (l), components per layer (c), and faulty outputs (o)**

running times for the distributed algorithm are up to hundreds of seconds. In general the running times for the centralized are 2 to 3 times larger. However, two issues are important. We could not run the centralized algorithm for more than 18 components (2 layers of 9). Second, regarding the distributed algorithm, there is a high variance in those times because the circuits generated randomly connect adders and multipliers in very different flavors. This leads to very different situations in terms of conflict detection and need of communication. Many times agents communicate information which leads to already known conflicts. Thus, running time can be exceptionally high.

Regarding the second metric, Table 1 shows the percentage of correctly computed hitting sets (i.e. the diagnosis) for different number of components and layers. The tendency here is similar to the one discussed for the CS's: the higher the number of components, the lower the performance. A higher number of faulty outputs increases the performance though. The fact that circuits with more failures are better handled by the distributed algorithm might seem strange at first. These circuits are hard to be analyzed since the number of component sets which explain the observed faults tends to be big. However, one must also note that a higher number of failures causes conflicting predictions to occur more often. In other words, communications between agents tend to contain more conflicting values, which makes it easier to discover the correct conflict sets.

Finally, the approach was also tested with a boolean circuit as metaphor for a regulatory network with five genes [1]. Each can be ON or OFF. The idea here is slightly different from the devices previously seen. Adders and multipliers are man-made device, whose behavior is well-known. In biology, we have just theories or assumptions about how the components should behave. Thus, the goal here is to verify whether the theory leads to the observed behavior and, when this is not the case, to point out the candidates which could explain how or where the theory fails.

## 4. REFERENCES


[1] A. L. C. Bazzan. Distributed diagnosis of faults in circuits and biological systems. In F. W. et al., editor, *Proceedings of 3rd Workshop on Model-Based Systems (ECAI 2006)*, pages 16–20. ECAI, August 2006. www.inf.ufrgs.br/bazzan/downloads/mbs.pdf.

[2] R. Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24(1–3):347–410, December 1984.

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

[4] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, April 1987.