A Framework for Decentralized Qualitative Model-Based Diagnosis*

Luca Console, Claudia Picardi
Università di Torino
Dipartimento di Informatica
Corso Svizzera 185, 10149, Torino, Italy
lconsole,picardi@di.unito.it

Daniele Theseider Dupré
Università del Piemonte Orientale
Dipartimento di Informatica
Via Bellini 25/g, 15100, Alessandria, Italy
dtd@mfn.unimpn.it

Abstract

In this paper we propose a framework for decentralized model-based diagnosis of complex systems modeled with qualitative constraints and whose models are distributed among their subsystems.

We assume that Local Diagnosers are associated with subsystems and are coordinated by a Supervisor which acts as the diagnoser for the complex system. The Local Diagnosers and the Supervisor communicate via a standard interface and share a common modeling ontology. In this diagnostic architecture, connections between subsystems only need to be known at runtime, thus allowing for dynamic (re)configuration of the system.

The approach is designed to compute *partial* hypotheses in order to avoid unnecessary queries to Local Diagnosers.

1 Introduction

Developing diagnostic software for complex systems is a challenging task especially when a system is composed of many subsystems coming from different suppliers (a very common situation in industry). Decomposition has been recognized as an important leverage to manage architectural complexity of the systems to be diagnosed. Most of the approaches, however, focus on hierarchical decomposition [Genesereth, 1984; Mozetic, 1991], while decentralization has been explored less frequently (see e.g. [Pencolé and Cordier, 2005]). On the other hand, new problems have to be faced when independently manufactured subsystems are assembled. Each subsystem may come with its own (possibly embedded) diagnostic software whose details should not be disclosed to the customers. A diagnostic software for a complex system has thus to take into account all its subsystems and their diagnosers. The structure of the system and the paths of interaction of the subsystems may be defined statically, or they may change dynamically, e.g. due to redundant design, and for software systems like Web Services.

In this paper we propose a decentralized supervised approach to diagnosis. We assume that a system is formed by subsystems each one having its Local Diagnoser, while a Supervisor is associated with the overall system, with the aim of coordinating Local Diagnosers and producing global diagnoses. The approach computes a set of *partial* diagnoses [de Kleer *et al.*, 1992] that represents all the consistency-based diagnoses; it is designed to avoid commitment on parts of the model that are unnecessary to explain the observations. In this way local diagnosers are not queried on global hypotheses where their contribution is irrelevant.

The paper is organized as follows: in the next section we describe the decentralized architecture we propose. Section 3 introduces the type of models used for diagnosis and a running example. Then, we discuss how the decentralized diagnostic process is carried out in our approach (Section 4) and analyze its complexity (Section 5). We finally conclude with a discussion on related work.

2 Architecture

We propose a supervised diagnostic architecture where:

- A Local Diagnoser is associated with each subsystem. The model of the subsystem is private.
- A Supervisor of the diagnostic process is associated with the system. It coordinates the work of the diagnosers optimizing their invocations during a diagnostic session.

In the definition we consider the general case where:

(i) The Supervisor needs no a-priori knowledge about the subsystems and their paths of interaction (they can be reconstructed at run time), and Local Diagnosers do not necessarily know each other. This ensures that subsystems can be added/removed/replaced independently of each other at any time.

(ii) Each Local Diagnoser must implement a given communication interface with the Supervisor (which is reasonable in an assembler-supplier relationship).

(iii) It should be possible to have a hierarchical architecture of Supervisors corresponding to a hierarchical decomposition of the system.

The architecture defines a communication interface between the Local Diagnosers and the Supervisor. They must share a modeling ontology even though this allows each Local Diagnoser to use its own modeling language and assump-

*This work has been partially supported by EU (Ws-Diamond Project, IST-516933) and by MIUR-PRIN (Quadrantis Project).

tions, provided that it is able to map it over the common one during communications.

Local diagnosers are awakened by the subsystem they are in charge of whenever an anomaly (fault) is detected. They may explain the fault locally or may put the blame on an input received from another subsystem. They communicate to the Supervisor their explanations (without disclosing the details of the possible local failure(s)). The Supervisor may invoke the Local Diagnoser of the blamed subsystem asking it to generate explanations. The Supervisor may also ask a Local Diagnoser to check some predictions.

3 Models

In this section we discuss the modeling ontology used in the interface. First of all, as it is common in model-based diagnosis, we consider *component-oriented* models, where each component has one correct behavior and possibly one or more faulty behaviors. Each behavior of a component is modeled as a relation over a set of variables.

We consider *qualitative models* where all variables have a finite, discrete domain, thus representing either a discretization of the corresponding physical quantity, or an abstraction over some of its properties. This implies that the model of a component C is a finite relation including a *mode* variable $C.m$, that ranges over the set of behavior modes $\{ok, ab_1, \ldots, ab_n\}$ of the component itself.

In the example in this paper we will use *deviation models* [Malik and Struss, 1996] where each physical quantity q represented in the model has a corresponding variable Δq representing a qualitative abstraction of the *deviation* of q with respect to its expected value; in particular, the sign abstraction $\{-, 0, +\}$, expressing whether q is lower than, equal to, or higher than its expected value.

In this paper we only deal with atemporal diagnosis, assuming that the models abstract away the temporal information, which is of course restrictive [Brusoni *et al.*, 1998].

As a running example we use a simplified part of an Aircraft System, namely the part concerned with the Landing Gear. Figure 1 represents a high-level view of the system in terms of subsystems. The ones pictured with a dashed line do not directly take part in the example, but are shown for the sake of completeness.

The Hydraulic Extraction System (**HES**) creates a pressure that mechanically pushes the Landing Gear, thereby extracting it. The **HES** is also connected to some leds on the cockpit, that show whether the subsystem is powered up or not. In order to create the pressure, the **HES** takes power from two sources. The main source is the Power Transmission from the aircraft engine, which actually powers up the main pump of the **HES**. A secondary source (used to transmit the pilot command from the cockpit, and to light up leds) is the Independent Power Supply, which produces a low-amperage DC current.

4 Decentralized Diagnostic Process

In this section we will describe how the diagnostic process is carried out. As we have already mentioned, a Supervisor coordinates the activities of several Local Diagnosers

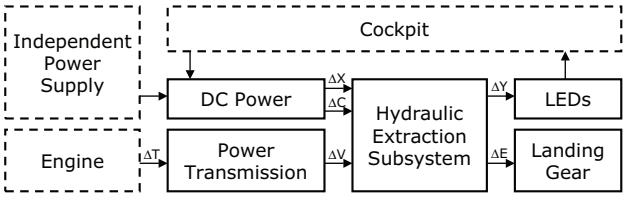

Figure 1: The Landing Gear section of an Aircraft System

$\{LD_1, \ldots, LD_n\}$. In order to obtain a loosely coupled integration, the Supervisor assumes that each LD_i is stateless, that is, every invocation of a Local Diagnoser is independent of the others. The interaction is thus defined by:

- An interface that each Local Diagnoser must implement, discussed in section 4.1. The role of each Local Diagnosers consists in generating hypotheses consistent with the model of its subsystem and the corresponding observations.
- An algorithm for the Supervisor that computes diagnoses by coordinating the Local Diagnosers through the above interface, as explained in section 4.2. The role of the Supervisor is to propagate hypotheses to the neighbors of a Local Diagnoser.

4.1 The Extend interface

We assume that each Local Diagnoser LD_i reasons on a model M_i of its subsystem S_i. From the point of view of the Supervisor and its interactions with Local Diagnosers, each model M_i is a relation over a set of variables where:

- mode variables, denoted by $S_i.m$, express the behavior mode of components in S_i;
- input/output variables express values of quantities that S_i exchanges with other subsystems.
- there may be additional internal (private) variables.

The interface between the Supervisor and the Local Diagnosers is made of a single operation called EXTEND that the Supervisor invokes on the Local Diagnosers. Moreover, upon an abnormal observation, Local Diagnosers autonomously execute EXTEND and send the results to the Supervisor.

The goal of EXTEND is to explain and/or verify the consistency of observations and/or hypotheses made by other Local Diagnosers. Thus, the input to EXTEND when invoked on LD_i is a set of hypotheses on the values of input/output variables of M_i. Such hypotheses are represented as a set of *partial* assignments over the variables of interest.

In the particular case where EXTEND is autonomously executed by a Local Diagnoser upon receiving an abnormal observation from its subsystem, the set of hypotheses contains only the empty assignment (i.e. the one with empty domain).

For each hypothesis α received in input, LD_i must first check whether α is consistent with the local diagnostic model M_i and the local observations ω_i. The extension that LD_i should perform may be interpreted as follows (in causal terms): backwards, to verify whether the hypothesis needs further assumptions to be supported within M_i and ω_i; forwards, to see whether the hypothesis has consequences on other subsystems, which can be used to discard or confirm it.

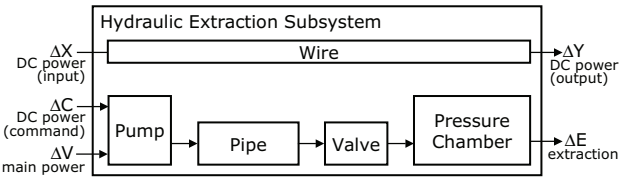

Figure 2: A closer view to the Hydraulic Extraction System

Let us consider as an example the Hydraulic Extension System (Figure 2) and suppose the Local Diagnoser LD_{HES}, associated to the **HES**, receives in input an assignment that assigns the value "$-$" to its output data variable **HES**.ΔE, representing the mechanical extraction of the Landing Gear. This value means that the Landing Gear did not extract when expected. LD_{HES} may then see that, with respect to the local model, one of the following must have happened:

- An internal component has failed, for example the pump is stuck or the pipe is leaking.
- One of the two inputs of the pump is wrong, for example the pump is not powered or it has not received the command.

This means that the partial assignment **HES**.$\Delta E = -$ can be extended in four ways: by assigning **HES**.$m =$ **stuck**, or **HES**.$m =$ **leaking**, or **HES**.$\Delta V = -$, or **HES**.$\Delta C = -$.

What is important regarding extensions is that they should include everything that can be said under a given hypothesis, but nothing that could be said also without it. In other words, we are interested in knowing whether a given assignment constrains other variables *more* than the model alone does. The following definition captures and formalizes this notion.

Definition 1 Let M_i be a local model with local observations ω_i, and let γ be an assignment. We say that γ is *admissible* with respect to M_i and ω_i if:

i. γ is an extension of ω_i;

ii. γ is consistent with M_i;

iii. if $M_{i,\gamma}$ is the model obtained by constraining M_i with γ, its restriction to variables *not* assigned in γ is equivalent to the restriction of M_i itself to the same variables:

$$M_{i,\gamma}|_{\text{VAR}(M_i)\setminus\textbf{Dom}(\gamma)} \equiv M_i|_{\text{VAR}(M_i)\setminus\textbf{Dom}(\gamma)}.$$

Thus, for each assignment α received in input, the EXTEND operation of a Local Diagnoser LD_i returns a set of assignments E_α that contains **all minimal admissible extensions** of α with respect to M_i and ω_i, restricted to input, output and mode variables of M_i. Notice that, whenever α is inconsistent with the observations or the model, E_α is empty.

Minimal admissibility avoids unnecessary commitments on values of variables that are not concretely constrained by the current hypothesis.

In order to illustrate these definitions, let us consider a (simplified) model M_P of the pump P alone. The model includes four variables: $P.m$ represents the behavior mode, $P.\Delta V$ the power supply to the pump, $P.\Delta C$ the command that turns on the pump, and $P.\Delta F$ the flow coming out from the pump. The extensional representation of M_P is:

$P.m$	$P.\Delta C$	$P.\Delta V$	$P.\Delta F$	$P.m$	$P.\Delta C$	$P.\Delta V$	$P.\Delta F$
ok	0	0	0	stuck	0	0	$-$
ok	0	$+$	$+$	stuck	0	$+$	$0,+,-$
ok	0	$-$	$-$	stuck	0	$-$	$-$
ok	$+$	0	$+$	stuck	$+$	0	$0,+,-$
ok	$+$	$+$	$+$	stuck	$+$	$+$	$0,+,-$
ok	$+$	$-$	$0,+,-$	stuck	$+$	$-$	$0,+,-$
ok	$-$	0	$-$	stuck	$-$	0	$-$
ok	$-$	$+$	$0,+,-$	stuck	$-$	$+$	$0,+,-$
ok	$-$	$-$	$-$	stuck	$-$	$-$	$-$

Suppose EXTEND is called on M_P alone, having in input an assignment α such that **Dom**$(\alpha) = \{P.\Delta F\}$ and $\alpha(P.\Delta F) = -$. It can be verified that α itself is not admissible, while the minimal admissible extensions of α wrt M_P are:

	$P.m$	$P.\Delta C$	$P.\Delta V$	$P.\Delta F$
γ_1	stuck			$-$
γ_2		$-$		$-$
γ_3			$-$	$-$

γ_1, γ_2 and γ_3 express the three possible explanations for $\Delta F = -$: either the pump is stuck, or it is not powered, or it did not receive the command. For each of the three assignments, unassigned variables are those whose value is irrelevant to the explanation.

4.2 The Supervisor algorithm

The Supervisor is started by a Local Diagnoser LD_s that sends to it the results of an EXTEND operation executed as a consequence of an abnormal observation.

During its computation, the Supervisor records the following information:

- a set $\mathcal{H}$ of assignments, representing current hypotheses;
- for each assignment α and for each variable $x \in$ **Dom**(α), a *modified bit* **mdf**$(\alpha(x))$.

Moreover, we assume that given an input or output variable x that has been mentioned by one of the Local Diagnosers, the Supervisor is able to determine the variable(s) $\textbf{conn}_1(x), \ldots, \textbf{conn}_k(x)$ connected to it. We do not make any assumption on whether this information is known *a priori* by the Supervisor, or it is retrieved dynamically, or it is provided by the Local Diagnoser itself.

The Supervisor initializes its data structures with the results of the initial EXTEND operation that has awakened it:

- $\mathcal{H}$ contains all the assignments that were sent by LD_s as the result of EXTEND;
- for each $\alpha \in \mathcal{H}$, and for each $x \in$ **Dom**(α), the Supervisor extends α to $\textbf{conn}_1(x), \ldots, \textbf{conn}_k(x)$ by assigning $\alpha(\textbf{conn}_j(x)) = \alpha(x)$ for $j = 1, \ldots, k$. Then the Supervisor sets $\textbf{mdf}(\alpha(\textbf{conn}_j(x))) = 1$.

Modified bits are used by the Supervisor to understand whether it should invoke a Local Diagnoser or not, according to the following:

Rule R. If a subsystem S_i has a variable x with **mdf**$(\alpha(x)) = 1$ for some α, then LD_i is a candidate for invocation and α should be passed on to EXTEND.

After initializing data structures, the Supervisor loops over the following four steps:

Step 1: select the next LD_i to invoke. The Supervisor selects one of the Local Diagnosers LD_i for which there is at least one assignment α meeting **Rule R**. *If there is none, the loop terminates.*

Step 2: invoke EXTEND**.** If LD_i has never been invoked before in this diagnostic process, then the input to EXTEND is the set of all assignments in H, restricted to variables of M_i. Otherwise the input consists only of those assignments α that meet **Rule R**.

Step 3: update $\mathcal{H}$. The Supervisor receives the output of EXTEND from LD_i. For each α in input, EXTEND has returned a (possibly empty) set of extensions $E_\alpha = \{\gamma_1, \ldots, \gamma_k\}$. Then α is *replaced* in $\mathcal{H}$ by the set of assignments $\{\beta_1, \ldots, \beta_k\}$ where β_j is obtained by:

- combining α with each $\gamma_j \in E_\alpha$;
- extending the result of this combination to connected variables, so that for each $x \in \mathbf{Dom}(\gamma)$ representing an input/output variable, $\beta_j(\mathbf{conn}(x)) = \beta_j(x)$.

This implies that rejected assignments, having no extensions, are removed from $\mathcal{H}$. This step and the following take place in exactly the same way if the EXTEND operation was not invoked by the Supervisor, but rather autonomously executed by a Local Diagnoser upon receiving an alarm. This is treated as an extension of the empty assignment, thus applicable to all α in $\mathcal{H}$.

Step 4: update the mdf bits. For each assignment β_j added in Step 3 **mdf** bits are set as follows:

(i) For each variable x *not* belonging to M_i such that $x \in \mathbf{Dom}(\beta_j)$ and $x \notin \mathbf{Dom}(\alpha)$, $\mathbf{mdf}(\beta_j(x))$ is set to 1.

(ii) For each variable x belonging to M_i such that $x \in \mathbf{Dom}(\beta_j)$, $\mathbf{mdf}(\beta_j(x))$ is set to 0.

(iii) For any other variable $x \in \mathbf{Dom}(\beta_j)$, $x \in \mathbf{Dom}(\alpha)$ and $\mathbf{mdf}(\beta_j(x)) = \mathbf{mdf}(\alpha(x))$.

Notice that the diagnostic process terminates: new requests for EXTEND are generated only if assignments are properly extended, but assignments cannot be extended indefinitely.

We can prove the following:

Theorem 1 *Let M denote the global model, and let ω denote the set of observations obtained during the diagnostic process. At the end of the Supervisor loop, the set $\mathcal{H}$ of partial assignments has the following properties:*

i. each partial assignment $\alpha \in \mathcal{H}$ is admissible with respect to M and ω;

ii. the set of extensions is complete *wrt M and ω, that is, for each* total *assignment γ consistent with M and ω, there exists $\alpha \in \mathcal{H}$ such that γ is an extension of α.*

The proof of this theorem relies on the following property of admissible extensions:

Proposition 2 *Let M_1 and M_2 denote two interacting models, with observables ω_1 and ω_2, and let α be a partial assignment over $M_1 \bowtie M_2$ (where $\bowtie$ denotes the composition of the two models). If α is admissible both wrt M_1, ω_1 and wrt M_2, ω_2 then it is admissible wrt $M_1 \bowtie M_2, \omega_1 \bowtie \omega_2$.*

We need now to establish a relation between $\mathcal{H}$ and consistency-based diagnoses. Let us first extract diagnostic information from assignments in $\mathcal{H}$:

Definition 2 Let α be an assignment in $\mathcal{H}$ at the end of the diagnostic process. The *diagnostic information* $\Delta(\alpha)$ associated with α is its restriction to mode variables.

As a consequence of Theorem 1 we have:

Proposition 3 *The set $\{\Delta(\alpha) \mid \alpha \in \mathcal{H}\}$ is a* complete set of partial diagnoses [de Kleer *et al.*, 1992] *wrt the model M and observations ω, meaning that every total mode assignment completing a $\Delta(\alpha)$ is a consistency-based diagnosis for M and ω, and every consistency-based diagnosis is a completion of some $\Delta(\alpha)$.*

Corollary 4 *Let $\mathcal{D}$ be the set of diagnoses obtained by completing all $\Delta(\alpha)$ with ok assignments. The set of minimal diagnoses in $\mathcal{D}$ coincides with the set of minimal consistency-based diagnoses for M and ω.*

Notice that in the Supervisor algorithm the role of mode variables is rather peculiar. Since these variables are local to a given system, the Supervisor never communicates their value to a Local Diagnoser different than the one originating them. Thus, they are not needed for cross-consistency checks. There are two main reasons why we need the Supervisor to be aware of them:

- They provide the connection between two different invocations on the same Local Diagnoser; by having the Supervisor record them and send them back on subsequent calls, we allow the Local Diagnosers to be stateless.
- The Supervisor needs them to compute globally minimal diagnoses.

Notice however that, for both of these goals, the values of mode variables need not be explicit: Local Diagnosers may associate to each fault mode a coded Id and use it as a value for mode variables. In this way, information on what has happened inside a subsystem can remain private.

Let us sketch a scenario of execution of the Supervisor algorithm for the system in Figure 1. Suppose that LD_{LG} notices that the landing gear did not extract and blames it on the Hydraulic Extraction System. Then the Supervisor asks LD_{HES} for an explanation, and receives back four extensions. In two of them, the **HES** takes responsibility for the problem (either the pump is stuck or the pipe is leaking). In the other two, it blames respectively the Power Transmission (input ΔV) and the DC Power (input ΔC).

The Local Diagnoser of the Power Transmission is then asked to explain the blame; it has an observation confirming that its input ΔT is ok, thus it can explain it only as an internal failure. Then the Local Diagnoser of the DC Power is asked to explain its blame. Also the DC Power can explain it only as an internal failure, but it also states that this type of error would produce a problem also on its other output ΔX. At this point LD_{HES} is invoked again to verify the consequences of this prediction, and it states that a problem on ΔX would

produce a similar problem on ΔY. The Local Diagnoser of the LEDs subsystem discards this possibility thanks to its local observations; thus the causes of the failure can only be in the Hydraulic Extraction System or in the Power Transmission.

4.3 Hierarchies of Supervisors

In this section we discuss the problem of having a multi-layered hierarchical structure of diagnosers. As we will see in section 5, this has also a relevant impact on complexity issues.

Let us consider a Supervisor S' at an intermediate level of the hierarchy, in charge of a portion M' of the global model M, with observations ω'. When S' is invoked from a higher-level Supervisor S, its loop starts with data structures initialized as follows:

- $\mathcal{H}$ contains all the assignments that S asked to extend;
- for each $\alpha \in \mathcal{H}$, and for each $x \in \mathbf{Dom}(\alpha)$, S' sets $\mathbf{mdf}(\alpha(x)) = 1$.

In general, at the end of its loop, S' does not compute all minimal admissible extensions of the initial assignments in $\mathcal{H}$ wrt M' and ω'. In fact, Theorem 1 guarantees that all the returned extensions are admissible, and that for each input assignment a complete set of extensions is returned, but not that such extensions are also *minimal*.

It is however possible to give a sufficient condition which guarantees minimality. Moreover, this provides a way to estimate the distance that returned extensions have from minimality. Intuitively, minimality cannot be guaranteed when the context where a subsystem is operating (i.e., the other subsystems it is connected to) limits its possible behaviors wrt those described in its model.

More formally, let us consider a Supervisor S with n Local Diagnosers: the model M is thus partitioned in $M_1, \ldots, M_n$ each owned by a different LD_i. Let $M|_{\text{VAR}(M_i)}$ denote the projection of M over variables in M_i. If, for every $i = 1, \ldots, n$, $M_i \equiv M|_{\text{VAR}(M_i)}$, then the Supervisor actually computes minimal admissible extensions. The more M_i is larger than $M|_{\text{VAR}(M_i)}$, the farther the returned extensions may be from minimality.

The operation computed by a Supervisor S with model M and observations ω is then, in the general case, a RELAXEDEXTEND that, for each input assignment α, returns a *complete set* of admissible extensions of α wrt M and ω.

Replacing EXTEND with RELAXEDEXTEND has no effect on the correctness of the algorithm, since Theorem 1 does not exploit the assumption that the extensions returned by Local Diagnosers are minimal:

Proposition 5 *If all Local Diagnosers involved in a diagnostic process implement the* RELAXEDEXTEND *operation, then the Supervisor coordinating them implements* RELAXEDEXTEND *as well.*

Thus, a multi-layered hierarchy of Supervisors is possible, although choosing to treat a subsystem with multiple Local Diagnosers, rather than as a whole, can result in non-minimal admissible extensions.

Not having minimal extensions has impact on at least three aspects of the diagnostic process. First, the contents of the exchanged messages and of the hypotheses set $\mathcal{H}$ may be larger. Second, some local diagnoser could be invoked even if the information it can provide is irrelevant to the diagnostic process. Third, as we will see in the next section, RELAXEDEXTEND has a lower complexity than EXTEND. This trade-off must be taken into account when selecting a diagnostic architecture.

5 Complexity

In this section we discuss the complexity of the proposed algorithm. Diagnosis is NP-complete, therefore we cannot expect anything better.

In our approach, the core of the diagnostic process is the Supervisor loop. This task can is composed by two activities: invocations of Local Diagnosers and merging of the results. The latter depends on the size of the results themselves; we need however to recall that the Supervisor builds a set of partial assignments, which in the worst case is exponential in the number of variables k. Thus the merging activity is in the worst case exponential in k. As to Local Diagnoser invocations, each of them is invoked at most once for each interface variable in its model. If we denote by k_i the number of variables in model M_i, and by $\tilde{k}_i \leq k_i$ the number of interface variables in M_i, then we can express the complexity of the Supervisor as:

$$O(e^k) + O(\tilde{k}_i \mathcal{C}_{LD}(k_i))$$

where $\mathcal{C}_{LD}(k_i)$ denotes the complexity of a Local Diagnoser invocation over a model with k_i variables.

Let us now consider the EXTEND operation performed by Local Diagnosers. Given a model M, and an input assignment α, the task of finding an admissible extension for α in M is essentially a satisfiability problem: it suffices to find a *total* assignment satisfying M_α. Thus, this task is exponential in the number of involved variables.

Unfortunately, finding minimal admissible extensions is harder: it is NP-hard, and belongs to NP^{NP} (although it does not appear to be complete for this class).

However, analyzing the complexity expression above, we see that, if the model can be split in subsystems with an upper bound on the number of variables in each of them, then k_i can be treated as a constant, and the complexity of the overall algorithm goes back to $O(e^k)$.

In component-oriented systems and models this decomposition is straightforward: components are usually reasonably sized, and the complexity of systems depends on the number of components rather than on the complexity of individual components. This suggests that the best way to tackle the complexity of EXTEND is to increase the number of local diagnosers and decrease the size of models in their care. In this case, the multi-layered architecture in section 4.3 can be profitable. In particular, if the split models can be obtained by projecting the global model (as it is the case when the split is due to efficiency reasons and not to subsystem privacy), there is no loss in the quality of the results, while the complexity of the algorithm is lower.

6 Conclusions

In this paper we discuss a supervised decentralized approach to diagnosis that allows to loosely couple multiple diagnosers, in order to deal with systems consisting of several subsystems, whose models cannot be composed.

The goal of the paper is to show that we can effectively perform the diagnostic task and define intelligent strategies for the Supervisor, even if it is not aware of the internal aspects of Local Diagnosers, the Local Diagnosers do not know each other, and their paths of interaction dynamically change.

The approach proposed in this paper builds on [Ardissono *et al.*, 2005], which deals with diagnosis of Web Services. In particular, the definition of the EXTEND interface is borrowed from [Ardissono *et al.*, 2005], while the Supervisor algorithm is generalized in order to loosen the assumptions while not losing diagnostic coverage. Moreover, the properties of this algorithm are proved and its complexity analyzed.

The proposed framework was presented for atemporal models, where dynamic features are either absent, or have been abstracted away. The extension to temporal issues in diagnosis [Brusoni *et al.*, 1998] is subject for future work. Dynamic models (possibly including feedback loops) should be dealt with, as well as using temporal information on observations for discrimination (with the additional problem of possible communication delays among diagnosers).

The approach in [Pencolé and Cordier, 2005] has some relevant similarity with that of this paper, from the point of view of diagnostic architecture: a Supervisor is in charge of computing global diagnoses exploiting the work of several local diagnosers. However, this work focuses on quite different theoretical and practical problems:

- the system to diagnose is modeled as a discrete-event system;
- the main problem is to avoid composing the whole model, because this would produce a state-space explosion;
- as a consequence, the supervisor is aware of the subsystem models, but cannot compose them: it can only compose local diagnoses to obtain a global one;
- due to the nature of the considered systems, reconstructing global diagnoses is a difficult task, and as such it is one of the main focuses of the paper.

Other papers in the literature deal with completely distributed approaches to diagnosis, where diagnosers communicate with each other and try to reach an agreement, without a supervisor coordinating them. Our choice of a supervised approach was motivated by the need of having loosely coupled diagnosers, while a purely distributed approach requires diagnosers to establish diagnostic sessions and exchange a greater amount of information in order to compute diagnoses that are consistent with each other.

Nevertheless, the distributed approach proposed in [Roos *et al.*, 2003] has some similarity with ours because it is based on the idea of diagnosers explaining blames received from others, or verifying hypotheses made by others. Needless to say, the proposed algorithms are completely different, having to deal with a distributed approach. Moreover, in order to reduce computational complexity, the authors introduce some fairly restrictive assumptions on the models (e.g. requiring that explanations imply normal observations, or that output values are either completely undetermined or completely determined).

Finally, [Provan, 2002] deals with a scenario similar to ours: each diagnoser has a local model and observations, which are not shared with the others. Each local diagnoser computes local minimal diagnoses, and then engages in a conversation to reach a globally sound diagnosis. However, being the approach purely distributed, solutions are different. In particular, in order to propose a solution with reduced complexity, the author focuses on systems whose connection graph has a tree-like structure. On the contrary, our approach (thanks to the presence of the supervisor) does not need to make any assumption on system structure.

References

[Ardissono *et al.*, 2005] L. Ardissono, L. Console, A. Goy, G. Petrone, C. Picardi, M. Segnan, and D. Theseider Dupré. Enhancing Web Services with diagnostic capabilities. In *Proceedings of the 3rd IEEE European Conference on Web Services (ECOWS 2005)*, Vaxjo, Sweden, 2005.

[Brusoni *et al.*, 1998] V. Brusoni, L. Console, P. Terenziani, and D. Theseider Dupré. A spectrum of definitions for temporal model-based diagnosis. *Artificial Intelligence*, 102(1):39–79, 1998.

[de Kleer *et al.*, 1992] J. de Kleer, A. Mackworth, and R. Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2–3):197–222, 1992.

[Genesereth, 1984] M.R. Genesereth. The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24(1-3):411–436, 1984.

[Malik and Struss, 1996] A. Malik and P. Struss. Diagnosis of dynamic systems does not necessarily require simulation. In *Proc. 7th Int. Work. on Principles of Diagnosis (DX-96)*, pages 147–156, 1996.

[Mozetic, 1991] I. Mozetic. Hierarchical model-based diagnosis. *Int. J. of Man-Machine Studies*, 35(3):329–362, 1991.

[Pencolé and Cordier, 2005] Y. Pencolé and M.-O. Cordier. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence*, 164(1-2), 2005.

[Provan, 2002] G. Provan. A model-based diagnostic framework for distributed systems. In *Proc. 13th Int. Work. on Principles of Diagnosis (DX-02)*, pages 16–22, 2002.

[Roos *et al.*, 2003] N. Roos, A. ten Teije, and C. Witteveen. A protocol for multi-agent diagnosis with spatially distributed knowledge. In *2nd Int. Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2003)*, Melbourne, Australia, July 2003.