DISTRIBUTED DIAGNOSIS FOR EMBEDDED SYSTEMS IN AUTOMOTIVE VEHICLES

Jonas Biteus[*], **Mathias Jensen**[**], **Mattias Nyberg**[**]

[*] *Dept. of Electrical Engineering, Linköpings universitet SE-581 83 Linköping, Sweden,* biteus@isy.liu.se
[**] *Engine Development, SCANIA AB, SE-151 87 Södertälje, Sweden* {mathias.jensen, mattias.nyberg}@scania.com

Abstract: Fault diagnosis is important for automotive vehicles, due to economic reasons such as efficient repair and fault prevention, and legislations which mainly deals with safety and pollution. In embedded systems with dozens of electronic control units that states local diagnoses, it can be arbitrary difficult to find which combination of local diagnoses that points at the correct faulty components. An algorithm is presented that both finds the diagnoses that in themselves are complete and chooses only those diagnoses that are more likely to be correct, this restriction is wanted due to the limitations in processing power, memory, and network capacity. The embedded system in a Scania heavy duty vehicle has been used as a case study to find realistic requirements on the algorithm.

Keywords: Distributed diagnosis; Diagnosis; Fault isolation; Embedded systems.

1. INTRODUCTION

Most modern automotive vehicles include several *electronic control units* (ECU) which communicate over an electronic network. Each ECU is usually connected to one or several *components*, e.g. sensors and actuators, and to make sure that the components are operating correctly, they are *monitored* by the ECUs. Often *diagnostic tests*, which can be simple or complex, are used to perform the monitoring.

The results of the tests can be used in different ways. The most direct approach is to collect all results in a central unit and then use the results to calculate the *global diagnoses*, where each global diagnosis states a diagnosis for the complete system. Since each of the global diagnoses points at a set of possibly faulty components, they are easy to understand. Unfortunately, the number of global diagnoses grows exponentially both with the number and the size of the results from the tests.

An alternative is to directly calculate the *local diagnoses* from the results in each ECU. Since several of these sets of local diagnoses might include the same components, a combination of local diagnoses must be used to correctly point at the faulty components. Since the correct combination might be difficult to find, the local diagnoses in the different ECUs can be merged to form the same global diagnoses as in the first approach, however, this leads also to an exponential growth of global diagnoses.

To reduce this combinatorial explosion it is sometimes useful to only consider the diagnoses that are more likely to be correct. Which diagnoses that are more likely could be decided in several different ways. In this paper, the diagnoses with the smallest *cardinality*, i.e. the smallest number of faulty components, will be considered to be the most likely.

An *algorithm* is presented that calculates the *global diagnoses with minimal cardinality from the local diagnoses*. To reduce the complexity, the global diagnoses are divided into smaller sets of diagnoses, where each of these sets are guaranteed to be free of complex relations to the others.

The algorithm can be run in a central diagnostic computer, or as presented here, it can *distribute the computation intense tasks to the local ECUs*. Often, there are limitations in both processing power, memory, and network capacity, therefore, this possibility makes the algorithm more adaptable.

1.1 *A Typical Embedded System*

Many vehicles have a *controller area network* (CAN) which connects several ECUs to each other. For ex-

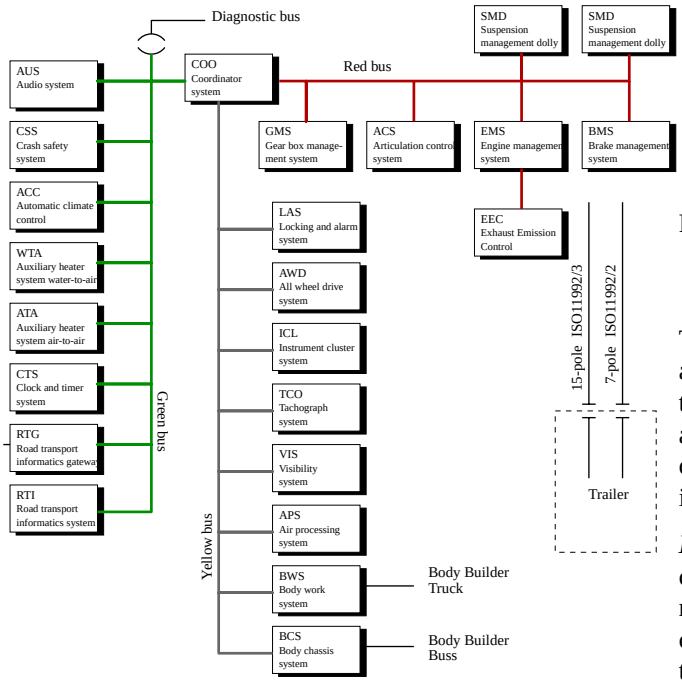


Fig. 1. The embedded system in a Scania vehicle.

ample, Fig. 1 shows a configuration of the embedded system used in the current Scania heavy-duty vehicles. This system includes three separate CAN buses and there are between 20 and 30 ECUs in the system, depending on the truck's type and outfit, and between 4 and 110 components are connected to each ECU. The ECUs' CPUs have typically a clocking speed of 8 to 64 MHz, and a memory capacity of 4 to 150 kB. The current Scania diagnostic system includes between 10 and 1000 diagnostic tests in each ECU.

1.2 *Related Work*

Diagnosis for embedded systems can be centralized or distributed. Most research has been aimed at the centralized problem, where a single process collects relevant data from the system and states global diagnoses, see for example (Hamscher *et al.*, 1992).

Distributed diagnosis has mostly been discussed for discrete event dynamic systems, see for example (Lunze and Schröder, 2001). One paper that discusses distributed consistency based diagnosis is (Roos *et al.*, 2003) where an algorithm for distributed diagnosis that uses both consistency and abduction based diagnosis (which is another type of diagnosis) is presented. The algorithm stipulates how conflicts should be exchanged such that each agent can state global diagnoses.

Distributed diagnosis requires a overall network architecture for communication, storage, etc. Such an architecture useful for diagnosis and control is being developed within the MAGIC project and is presented in (Köppen-Seliger, 2003). The aim for their project is to develop a general purpose architecture for diagnosis in complex systems.

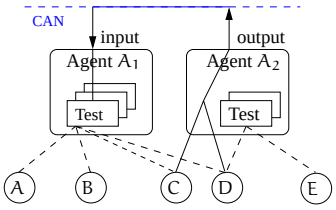


Fig. 2. A typical ECU, component and test layout.

2. SYSTEM DESCRIPTION

The diagnostic system involves a set of agents $\mathcal{A}$ and a set of components $\mathcal{C}$ which are the set of objects that can be diagnosed, by one or several agents, for abnormal behavior. The agents are connected to each other via a network, where an output signal in an agent is linked to input signals in one or several other agents.

Example 1: In Fig. 2, a typical layout of agents and components is shown. The tests involve the components A to E connected with dashed lines. Some calculated value involving components C and D can be transmitted over the network from agent A_2 to A_1. $\diamond$

In a vehicle, the agents are the software programs that are implemented in the ECUs. The components are the sensors, actuators, pipes, etc., which are monitored by the agents. The output signals, which are values from sensors, to actuators, or from calculations, are made available to the other agents over the network.

Each component has a general fault mode and a no-fault mode. In this paper it is assumed that the general fault mode does not have a model, and therefore the notation used in for example GDE can be employed (Hamscher *et al.*, 1992).

3. RELATION BETWEEN LOCAL AND GLOBAL DIAGNOSES

A diagnosis is represented by a set $\mathrm{D} \subseteq \mathcal{C}$ and states that the components $\mathrm{c} \in \mathrm{D}$ are in the faulty mode, while the remaining components are not in the faulty mode. A minimal diagnosis is a diagnosis where no proper subset is a diagnosis. The minimal diagnoses characterize all diagnoses, therefore unless explicitly stated, a *diagnosis* is here used to denote a minimal diagnosis. Considering minimal diagnoses, the similarity-equivalence $\simeq$ is used to show equivalence.

A *local diagnosis* is a diagnosis stated by an agent. It states that some components, which are somehow involved in the operation of the agent, are in the faulty mode. A *global diagnosis* is a diagnosis that states the mode of all components in the complete system.

Due to the combinatorial explosion when searching for global diagnoses, it is preferable to only consider some more likely sub-set of global diagnoses. One such set is the *minimal-cardinality* (mc) *diagnoses*

$$\mathcal{D}^{\mathrm{mc}} \triangleq \{\mathrm{D} \,\big|\, |\mathrm{D}| = \min_{\mathrm{D} \in \mathcal{D}} |\mathrm{D}|, \mathrm{D} \in \mathcal{D}\}$$

for a set $\mathcal{D}$, where $|D|$ is the number of components included in D. For a given set of diagnoses, the number of minimal cardinality diagnoses are often much less than the number of diagnoses. These diagnoses can therefore be used to reduce the combinatorial explosion that arises when several sets of diagnoses are merged together. Notice that if all components fail independently with the same probability, then the most probable global diagnoses will simply be the global minimal cardinality diagnoses.

In Section 1, the merge of different sets of local diagnoses was discussed. The *merging* of two sets of diagnoses $\mathbb{D}_1$ and $\mathbb{D}_2$ is defined as $\mathbb{D}_1 \bowtie \mathbb{D}_2 \triangleq \{D_1 \cup D_2 \,|\, D_1 \in \mathbb{D}_1, D_2 \in \mathbb{D}_2\}$, i.e. an addition of a union compared to the definition of Cartesian product. It can be proved that the following relation holds

$$\mathcal{D} \simeq \underset{A}{\bowtie} \mathbb{D}_A$$

where the set $\mathcal{D}$ is the global diagnoses and $\mathbb{D}_A$ is the local diagnoses in agent A.

Example 2: [Merge of local diagnoses] Consider the local diagnoses $\mathbb{D}_1 = \{\{A, B\}, \{C\}\}$ and $\mathbb{D}_2 = \{\{B\}\}$. The global diagnoses are $\mathcal{D} \simeq \mathbb{D}_1 \bowtie \mathbb{D}_2 = \{\{A, B\}, \{C, B\}\}$. ◇

Unfortunately, it is not the case that a merge of the local minimal cardinality diagnoses results in the global minimal cardinality diagnoses, i.e.

$$\mathcal{D}^{mc} \not\simeq \underset{A}{\bowtie} \mathbb{D}_A^{mc}$$

where $\mathbb{D}_A^{mc}$ is the mc diagnoses in agent A.

Example 3: [Merge of $\mathbb{D}_A^{mc}$] Consider the example above, where $\mathbb{D}_1^{mc} = \{\{C\}\}$ and $\mathbb{D}_2^{mc} = \{\{B\}\}$. The merge gives $\mathbb{D}_1^{mc} \bowtie \mathbb{D}_2^{mc} = \{\{C, B\}\}$, while $\mathcal{D}^{mc} = \{\{A, B\}, \{C, B\}\}$, i.e. the global mc diagnosis $\{A, B\}$ was not included in the merge of the local minimal cardinality diagnoses. ◇

3.1 *Representation of Global Diagnoses*

For the diagnoses to be easily understandable they should be as small as possible while being free of complex relations to other diagnoses. One such type of diagnoses that are free of complex relations, are the global diagnoses. These might however become quite large due to the inclusion of all local diagnoses.

One way to reduce the size of the global diagnoses is to represent them as a *conjunction of smaller disjoint parts of diagnoses*. Since these smaller parts are disjoint, the global diagnoses will simply be a merge of all these smaller parts, i.e. a simple relation would exist between the parts of diagnoses. From a technicians point of view they are more easily understandable than the complete set of global diagnoses.

One approach to achieve this is to merge the local diagnoses from a sub-set of agents, into a sub-set of global diagnoses, so that each such sub-set is disjoint from the other. Such a set of agents is here denoted a *module*. A set of agents $\bar{\mathcal{A}}$ is a module if for all other modules $\bar{\mathcal{A}}_i$, $\bar{\mathcal{A}} \cap \bar{\mathcal{A}}_i = \emptyset$ and $(\bigcup_{A \in \bar{\mathcal{A}}} \bigcup_{D \in \mathbb{D}^A} D) \cap (\bigcup_{A \in \bar{\mathcal{A}}_i} \bigcup_{D \in \mathbb{D}^A} D) = \emptyset$. Given a module $\bar{\mathcal{A}}$, the *module diagnoses* is $\mathcal{D}^{mod} \triangleq \bowtie_{A \in \bar{\mathcal{A}}} \mathbb{D}_A$.

Example 4: [Module diagnoses] If $\mathbb{D}_1 = \{\{A, B\}\}$, $\mathbb{D}_2 = \{\{B, C\}\}$, and $\mathbb{D}_3 = \{\{E\}\}$, then for the modules $\bar{\mathcal{A}}_1 = \{A_1, A_2\}$ and $\bar{\mathcal{A}}_2 = \{A_3\}$, it follows that $\mathcal{D}_1^{mod} = \{\{A, B, C\}\}$ and $\mathcal{D}_2^{mod} = \{\{E\}\}$. ◇

Given the m:th set of module diagnoses, the *module minimal cardinality diagnoses* (MMCD) is denoted $\mathcal{D}_m^{mod,mc}$. In contrast to the case with merged local minimal cardinality diagnoses, it can be proven that

$$\mathcal{D}^{mc} \simeq \bowtie \mathcal{D}_m^{mod,mc}$$

This strongly motivates the use of module diagnoses.

4. FINDING ALL MODULE MINIMAL CARDINALITY DIAGNOSES

After each agent has created *local diagnoses*, these could be merged to *MMCDs*. To calculate the MMCDs the following could be done: Divide the agents into modules so that the merge of the corresponding local diagnoses will become MMCDs; In each module, sort the agents into a sort order such that the complexity of the upcoming merge is reduced and then merge the local diagnoses according to this order.

A direct and simple approach to divide the agents into modules is to first calculate $\mathbb{D}_A = \bigcup_{D \in \mathbb{D}_A} D$ for each agent, and then choose the minimal module $\bar{\mathcal{A}}$ such that $\bigcup_{A \in \bar{\mathcal{A}}} \mathbb{D}_A$ is disjoint from all other such sets.

A simple ordering is to choose the agent with largest minimal cardinality diagnosis to be first and then the rest follows in decreasing size.

A merge can be done by first finding a low lower limit on the size of the MMCDs and then merge those of the local diagnoses that are smaller or equal to this limit. If no MMCDs was found, then the limit is increased and the merge is started again from the first agent.

However, the computation time of this approach can be greatly improved by further reducing the size of the modules, sorting them into a better order, and finally merging the diagnoses such that the total number of merges is minimized. These improvements have been implemented in the algorithm described in this section.

4.1 *Main – Algorithm 1*

The improved algorithm consists of the same three main parts as the approach described above. Firstly, algorithm FindSubGraph is used to find a graph $\mathbb{G}$ whose *sub-graphs represent the modules*. Algorithm FindR is then used to sort the agents in a sub-graph into a *merge order* R. The agents in the module are

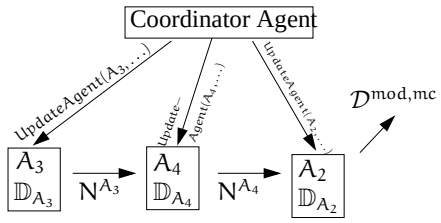


Fig. 3. The information flow in Example 5.

ordered in such a way that the complexity of the remaining algorithm is reduced. Finally, the local diagnoses are with algorithm UpdateAgent iteratively merged into sets of MMCDs.

The improvements over the first approach are that the modules are divided into smaller sets, the merge orders are chosen in a better way, and that the merge algorithm is much more efficient because it both keeps track and uses what have been done in previous runs.

The algorithm can be *distributed* in such a way that Algorithm 1, FindSubGraph, and FindR are evaluated in some coordinating agent, while the computation and memory intensive UpdateAgent is *evaluated in the corresponding agent*.

Now consider the main Algorithm 1 where an ordered set is represented by $(\cdot)$ and an unordered set by $\{\cdot\}$. Firstly, the graph is found and a merge order R is calculated from each sub-graph.

The algorithm starts with a low lower limit $\mathcal{L}$ on the cardinality of the MMCDs. The first approximation of $\mathcal{L}$ is that the MMCDs must be at least as large as the *largest* local minimal cardinality diagnosis, considering all agents in R. After the limit has been found, an evaluation of UpdateAgent is performed in each agent or until an agent returns a new value on $\mathcal{L}$.

Algorithm UpdateAgent calculates those diagnoses with cardinality less than or equal to $\mathcal{L}$ that would be formed if the local diagnoses in agents A_1 to A_k, which also have a cardinality less than or equal to $\mathcal{L}$, was merged, the set is denoted N in the algorithm. If no diagnoses could be found with cardinality less than or equal to $\mathcal{L}$, a new larger $\mathcal{L} = \mathcal{L}^{new}$ is calculated and a new search begins from A_1. This new $\mathcal{L}$ is chosen such that new merged diagnoses will be found at the next iteration.

When the last agent calculates a non-empty N, this set of diagnoses are the MMCDs for this module.

Example 5: [Algorithm 1] Consider Fig. 3 where the MMCDs should be found for the merge order (A_3, A_4, A_2). Algorithm 1 sends the UpdateAgent command with input $(A_3, \emptyset, \mathcal{L})$ to agent A_3, which extracts the diagnoses with cardinality lower than or equal to $\mathcal{L}$, denoted N^{A_3}. Thereafter, the UpdateAgent command is sent to A_4, which collects N^{A_3} over the network and merge it with its own local diagnoses with cardinality less than or equal to $\mathcal{L}$; those of the merged diagnoses with cardinality less than or equal to $\mathcal{L}$ is stored in N^{A_4}. After some iterations, the end result is the MMCDs $\mathcal{D}^{mod,mc}$. ◇

Algorithm 1 Module minimal cardinality diagnoses

Require: All local diagnoses $\mathbb{D}_A$ for all agents $\mathcal{A}$.
Ensure: All MMCDs $\mathcal{D}_m^{mod,mc}$.
1: $\mathbb{G} :=$ FindSubGraph$(\mathcal{A})$ [*Graph of sub-graphs.*]
2: **for all** $G \in \mathbb{G}$ **do**
3: $R :=$ FindR(G) [*Merge order* $R = (A_1, \ldots, A_n)$.]
4: $\mathcal{L} :=$ a lower bound on the cardinality for the MMCDs in R.
5: $k := 0$, $A_0 := \emptyset$
6: **repeat**
7: $k := k + 1$
8: $\mathcal{L}^{new} :=$ UpdateAgent$(A_k, A_{k-1}, \mathcal{L})$
[N *stored in* A_k *includes the new diagnoses.*]
9: **if** $\mathcal{L}^{new} > \mathcal{L}$ **then**
10: $k := 0$, $\mathcal{L} := \mathcal{L}^{new}$ [*Increase and restart*]
11: **end if**
12: **until** $k = n$ [A_n *is the last in* R.]
[$\mathcal{D}_m^{mod,mc} =$ N *stored in* A_n*, for the m:th* R.]
13: **end for**

4.2 *Find Sub Graph* $\mathbb{G}$ *– Algorithm 2*

In the direct approach the agents was divided with respect to all components in all diagnoses. This is however a pessimistic approach since some of the diagnoses might have a cardinality higher than the cardinality of the MMCDs, which means that it is unnecessary to consider them when deciding the modules.

The main idea in Algorithm 2 is to find an upper limit $\mathcal{U}$ of the cardinality of the MMCDs, and use this to reduce the size of the modules.

A first value of the upper limit can be chosen as $\mathcal{U} := \sum_{A \in \bar{\mathcal{A}}} \min_{D \in \mathbb{D}_A} |D|$. A lower upper limit is found by making a pre-merge of all minimal cardinality diagnoses. To reduce the complexity it is possible to only consider the minimal cardinality diagnoses after each merge, i.e. $\mathcal{U} := \min_{D \in \bar{\mathbb{D}}} |D|$ where

$$\bar{\mathbb{D}} := (\ldots((\mathbb{D}_{A_1}^{mc} \bowtie \mathbb{D}_{A_2}^{mc})^{mc} \bowtie \mathbb{D}_{A_3}^{mc})^{mc} \ldots).$$

The later limit will be used in this paper.

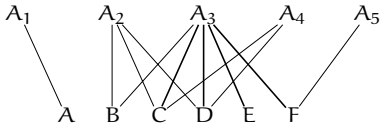
For each sub-graph found, Algorithm 2 is called recursively so that each sub-graph is, if it is possible, divided into even smaller sub-graphs.

Example 6: Consider a system with five agents including the local diagnoses

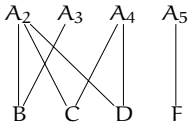
$$\mathbb{D}_1 = \{\{A\}\} \qquad \mathbb{D}_2 = \{\{B, C\}, \{D\}\}$$
$$\mathbb{D}_3 = \{\{B\}, \{C, D, E, F\}\} \qquad \mathbb{D}_4 = \{\{C\}, \{D\}\}$$
$$\mathbb{D}_5 = \{\{F\}\}$$

When using Algorithm 2, $\bar{\mathbb{D}} = \{\{A, D, B, F\}\}$ and $\mathcal{U} = 4$. The graph shown in Fig. 4(a) is created. It is divided into two sub-graphs which corresponding sets of agents are used in the next iteration. The first sub-graph only includes agent A_1.

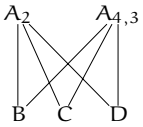
For the second sub-graph, the limit $\mathcal{U} = 3$ and the diagnosis $\{C, D, E, F\}$ can therefore be ignored. The result is the graph in Fig. 4(b) which is further divided into two sub-graphs. The final result is that the agents are divided into three sub-graphs including $\{A_1\}$, $\{A_2, A_3, A_4\}$, and $\{A_5\}$. ◇



(a) Original graph.



(b) Updated graph.



(c) Graph for Merge order.

Fig. 4. The graph representing a system with five agents and six components.

Algorithm 2 FindSubGraph

Input: Set of agents $V^{\mathcal{A}} \subseteq \mathcal{A}$. Require the local diagnoses $\mathbb{D}_A$ stored in the local agents.
Output: The graph $\mathbb{G}$ consisting of sub-graphs G, where a sub-graph $G = (\bar{V}^{\mathcal{A}}, \bar{V}^{\mathcal{C}}, \bar{E})$.

$\mathcal{U} :=$ a upper bound on the cardinality for the MMCDs.
$\mathbb{D}_A := \{D \mid |D| \leq \mathcal{U}, D \in \mathbb{D}_A\}$ for each agent
$V^{\mathcal{A}} := \{V^{\mathcal{A}} \mid \mathbb{D}_A \neq \emptyset\}$
$V^{\mathcal{C}} := \bigcup_{A \in V^{\mathcal{A}}} \bigcup_{D \in \bar{\mathbb{D}}_A} D$
$E := \{(A, c) \mid A \in V^{\mathcal{A}}, c \in \bigcup_{D \in \bar{\mathbb{D}}_A} D\}$
$\mathbb{G} := (V^{\mathcal{A}}, V^{\mathcal{C}}, E)$ [*Graph.*]
Find subgraphs $G \in \mathbb{G}$.
if $|\mathbb{G}| > 1$ **then** [*More than 1 sub-graph.*]
  $\mathbb{G} := \{\text{FindSubGraph}(\bar{V}^{\mathcal{A}}) \mid \bar{V}^{\mathcal{A}} \in G, G \in \mathbb{G}\}$ [*Recursive.*]
end if

4.3 *Finding Merge Order* $\mathbb{R}$ *– Algorithm 3*

The calculation of a merge order R is an interesting problem. By changing the merge order, the complexity of Algorithm 4 might change dramatically. The input is one of the sub-graphs that was found with Algorithm 2. The output is an ordered set $R \subseteq \mathcal{A}$, where each R represents a module.

When calculating R, the main idea is that the lower bound $\mathcal{L}$ in Algorithm 1 should be raised as soon as possibly towards its final value. This to reduce the complexity when the local diagnoses finally are merged. To raise $\mathcal{L}$ as fast as possible, the sets of local diagnoses should have as *few components in common* and the minimal cardinality local diagnoses should be as large *as possible*. To compromise between the number of common components c and the cardinality r of the diagnoses, $\alpha(c, r) = (c + 1)/r^2$ is used.

The number of common components is decided by looking at the number of connections between the agents and the components in the graph. For this is function Γ used, it gives the number of components that are shared between two agents

$$\Gamma(A_1, A_2) \triangleq |\{c \mid (A_1, c) \in E, (A_2, c) \in E\}|.$$

After two sets of local diagnoses have been merged, the merged set might include components from both sets of local diagnoses. Therefore, the agents and the corresponding components should be joined in the graph. A join of A_2 to A_1 in graph $G = (V^{\mathcal{A}}, V^{\mathcal{C}}, E)$

$$\text{join}(G, A_1, A_2) \triangleq (V^{\mathcal{A}} \backslash A_2, V^{C}, \bar{E})$$

where $\bar{E} \triangleq E \backslash \{(A_2, c) \mid \forall c\} \cup \{(A_1, c) \mid (A_2, c) \in E\}$.

Algorithm 3 FindR

Input: Sub-graph $G(V^{\mathcal{A}}, V^{\mathcal{C}}, E)$.
Output: A merge order R which is an ordering of $V^{\mathcal{A}}$.

if $|V^{\mathcal{A}}| = 1$ **then**
  $R := (V^{\mathcal{A}})$ [*The vertex* $V^{\mathcal{A}} \in \mathcal{A}$.]
else
  $(A_i, A_j) \in \arg\min_{A_i, A_j} \alpha(\Gamma(A_i, A_j), \min_{D \in \mathbb{D}_{A_i}} |D|)$
  $R := (A_i, A_j)$ [*First ordered set.*]
  $G := \text{join}(G, A_j, A_i)$ [A_i *merged* A_j.]
  while $|R| < |V^{\mathcal{A}}|$ where $R = \{A_1, \ldots, A_{end}\}$ **do**
    $A \in \arg\min_A \alpha(\Gamma(A, A_{end}), \min_{D \in \mathbb{D}_A} |D|)$
    $G := \text{join}(G, A, A_{end})$ [A_{end} *joined* A.]
    $R := R \cup A$ [*Ordered set. New* $A_n = A$.]
  end while [*All* $A \in V^{\mathcal{A}}$ *is included in* R.]
end if

Example 7: [Algorithm 3] Consider the first sub-graph in Example 6 and Fig. 4(b). Using the algorithm with the sub-graph as input, finds that A_3 and A_4 have the least number of components in common, i.e. zero components which give $\alpha(A_3, A_4) = 1$; since this is the lowest weight, $R = (A_3, A_4)$. After this A_3 is merged to A_4 which gives the graph in Fig. 4(c). Agent A_2 has three components in common with $A_{4,3}$ and is added to R, which give $R = (A_3, A_4, A_2)$. ◇

4.4 *Update Agent – Algorithm 4*

When the main algorithm should evaluate UpdateAgent with input $(A_k, A_{k-1}, \mathcal{L})$, it sends this command to agent A_k. The agent that receives this command should find the diagnoses with cardinality less than or equal to $\mathcal{L}$ in the set $\bowtie_{A=A_1}^{A_k} \mathbb{D}_A$. Since agent A_{k-1} has already calculated the diagnoses with cardinality less than or equal to $\mathcal{L}$ from $\bowtie_{A=A_1}^{A_{k-1}} \mathbb{D}_A$, the desired diagnoses can be calculated from the result in A_{k-1} merged with $\mathbb{D}_{A_k}$.

In the direct approach, this is repeated each time that $\mathcal{L}$ is raised. Algorithm 4 is more efficient, since it only calculates those diagnoses that have not been considered in previous runs, i.e. those with cardinality between the old and the new $\mathcal{L}$.

The main parts of the algorithm are the merge of the *old and new global diagnoses* for $\{A_1, \ldots, A_{k-1}\}$ with the *new local diagnoses* (TE), and the merge of the *new global diagnoses* for $\{A_1, \ldots, A_{k-1}\}$ with the *old local diagnoses* (TF). The merged diagnoses with cardinality greater than $\mathcal{L}$ are stored in M for later consideration, i.e. if $\mathcal{L}$ is later raised then parts of M might be included in the new N. Variable l is the number of times that the current agent has been called with command UpdateAgent(·).

The new global diagnoses N is saved for use by agent A_{k+1}. If A_k is the last agent and $N \neq \emptyset$ then N is the set of MMCDs, i.e. the wanted result.

Example 8: [Algorithm 4 and part of Algorithm 1] Consider the second set of agents found in Example 7, with the sort order $R = (A_3, A_4, A_2)$. The local diagnoses for these agents are

$\mathbb{D}_2 = \{\{B, C\}, \{D\}\}$ $\mathbb{D}_3 = \{\{B\}, \{C, D, E, F\}\}$ $\mathbb{D}_4 = \{\{C\}, \{D\}\}$

Algorithm 4 UpdateAgent

Input: A_k, A_{k-1} and $\mathcal{L}$. Require $\mathbb{D}_{A_k}$ stored in agent A_k and N stored in agent A_{k-1}.

Output: New sub-set of $\mathcal{D}^{mod}$ for agents $\{A_1, \ldots, A_k\}$ with cardinality $\leq \mathcal{L}$ stored as N^{A_k} in agent A_k. $\mathcal{L}$ as output.

1: $E := \{D \,|\, |D| \leq \mathcal{L}, D \in \mathbb{D}\}$
2: $\mathbb{D} := \mathbb{D} \backslash E$
3: **if** $i = 1$ **then** [*The first A in* R.]
4: $N := E$ [*New diagnoses from* A_k.]
5: **else if** $i > 1$ **then**
6: $T_i := N$ in A_{k-1}. [*New diagnoses from* A_{k-1}.]
7: $ET := \bigcup_{j=1}^{i} E \bowtie T_j$ [*Old and new merged with new.*]
8: $FT := F \bowtie T_i$ [*New merged with old.*]
9: $\mathbb{D}_{new} := ET \cup FT \cup M$
10: $N := \{D \,|\, |D| \leq \mathcal{L}, D \in \mathbb{D}_{new}\}$ [*New from* A_k.]
11: $M := \mathbb{D}_{new} \backslash N$ [*Store for later consideration.*]
12: **if** $N = \emptyset$ and N has not been non-empty **then**
13: $\mathcal{L} := \min(\min_{m \in M} |m|, \min_{D \in \mathbb{D}} |D|)$ [*New lower limit.*]
14: **end if**
15: $F := F \cup E$
16: **end if**

With $\mathcal{L} = 1$, the first agent A_3 finds the new diagnoses with cardinality less than or equal to $\mathcal{L}$ which give $N^{A_3} = \{\{B\}\}$. Agent A_4 collects N^{A_3} over the network and calculates $W^{A_4} := \{\{B, C\}, \{B, D\}\}$, but since $\mathcal{L} = 1$, the new set of diagnoses $N^{A_4} := \emptyset$. Since $N^{A_4} = \emptyset$ and N^{A_4} has not been non-empty, a new $\mathcal{L}$ is $\mathcal{L}^{new} = 2$. Since a new $\mathcal{L}$ was returned, the algorithm starts over from the first agent. A_3 has already stated diagnosis $\{B\}$ and since there are no other diagnosis with low cardinality, the new diagnoses are $N^{A_3} := \{\{\}\}$. Agent A_4 collects the new diagnoses and finds the new diagnoses $N^{A_4} := \{\{B, C\}, \{B, D\}\}$. A_2 takes over and calculates the new diagnoses $N^{A_2} := \{\{B, C\}, \{B, D\}\}$. Since $N^{A_n} = N^{A_2} \neq \emptyset$, the iterations ends and the set of MMCDs is $\mathcal{D}^{mod,mc} = N^{A_2}$, which is stored in agent A_2. ◇

4.5 *Simulations*

To test the algorithms, a model of an embedded system has been constructed. It is inspired by the system described in Section 1.1.

The model consists of components divided over three busses and 30 ECUs. Components, connections, and tests are picked by random. The connections are divided so that there are more connections between the agents within a single bus than between the agents in two different buses. The number of tests are between 5 % to 10 % of the number of components, i.e. for 500 components about 25 to 50 tests per agent. Whenever a diagnosis is transfered, a delay proportional to the cardinality of the diagnosis has been introduced, this models the transfer-time in the network. Into the model between 1 and 5 random faults have been inserted and then four variations of the algorithm have been simulated. The first is the complete Algorithm 1.

The second has replaced FindSubGraph with the more crude differentiation of the agents which was described in the beginning of Section 4. The third has replaced FindR with an ordering such that the agent with largest smallest diagnosis is first, then the rest follows in descending order. The last uses the direct method described in the beginning of Section 4.

Mean values of the computation times can be seen in Fig. 5. With this model, the number of modules found by FindSubGraph and the more simple differentiation are mostly the same, and therefore only a small difference can be seen in the mean times, most notably at 400 component. The complete algorithm is somewhat slower for few components, where the merge order is quite irrelevant, but *faster* for many components.

5. CONCLUSIONS

It has been shown how an algorithm that uses the agents local diagnoses to derive global diagnoses could be designed. To reduce the complexity, only the global diagnoses with minimal cardinality was considered. The algorithm could use the agents own processing power, which reduce the need for a central diagnostic agent.

6. REFERENCES

Hamscher, W., Console, L. and Kleer, de, John, Eds.) (1992). *Readings in Model-Based Diagnosis*. Morgan Kaufmann Publishers.

Köppen-Seliger, et al. (2003). Magic: An integrated approach for diagnostic data management and operator support. In: *Proceedings of IFAC Safeprocess'03*. Washington, USA.

Lunze, J. and J. Schröder (2001). State observation and diagnosis of discrete-event systems described by stochastic automata. *Discrete Event Dynamic Systems* **11**(4), 319–369.

Roos, N., ten Teije, A. and C. Witteveen (2003). A protocol for multi-agent diagnosis with spatially distributed knowledge. In: *2nd Conference on Autonomous Agents and Mult-Agent Systems*.

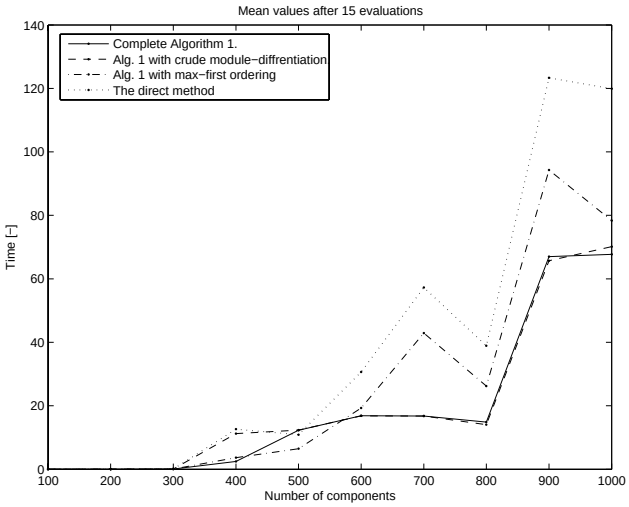


Fig. 5. Computation times for the algorithms with increasing number of components.