Generating Temporal Decision Trees for Diagnosing Dynamic Systems

Luca Console[(1)], Claudia Picardi[(1)], Daniele Theseider Dupré[(2)]
(1) Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, 10149 Torino, Italy
e-mail:{lconsole,picardi}@di.unito.it
(2) DSTA, Università del Piemonte Orientale
Corso Borsalino 54, 15100 Alessandria, Italy
e-mail: dtd@mfn.unipmn.it

Abstract

In previous work we showed that the automatic generation of diagnostic decision trees from qualitative models is a reasonable compromise between the advantages of using a model-based approach in automotive domains and the constraints imposed by on-board applications. In this paper we extend the approach to deal with temporal information. We introduce a notion of temporal diagnostic decision tree, in which nodes have a temporal label providing temporal constraints on the observations, and we present an algorithm for compiling such trees starting from a model-based diagnostic system.

Introduction

The adoption of on-board diagnostic software in automotive systems is becoming more and more important. This is due to the increasing complexity of the electronic control systems in cars and the consequent need of guaranteeing the availability of the vehicle, safety and easy maintenance. Moreover, regulations impose the presence of diagnostic software. However, in order to keep costs acceptable, limited hardware resources are available on board. Thus, although the model-based approach is very interesting and promising for the automotive domain (Console & Dressler 1999; Sachenbacher, Malik, & Struss 1998), it is still questionable if diagnostic systems can be designed to reason on first-principle models on board. For this reason within the VMBD project we experimented a *compilation-based* scheme to the design of on-board diagnostic systems for vehicles (Cascio *et al.* 1999).

The basic idea behind this approach can be summarized as follows:

- We start from a model-based diagnostic system which can be used to solve problems off-line (and which can also be used for off-board diagnosis in the garage).
- We collect a set of significant cases (that is, at least a good sample of the cases that have to be faced on-board) and we determine, using the model-based system, a solution for each one of the cases. The solution includes the set of candidate diagnoses and, more important, the recovery action that should be performed on-board in that case.
- Starting from a table that collects the cases and the corresponding recovery actions, we generate a set of diagnostic rules which are then implemented in the on-board system. Specifically, in (Cascio *et al.* 1999) we generated decision trees using an algorithm based on ID3 (Quinlan 1986).

The approach was successfully experimented for the Common-rail fuel injection system of a Lancia K (Cascio *et al.* 1999).

However, the system we implemented has a major limitation: it either relies on a model where the dynamics in the system has been abstracted away, or, even if reasoning on the dynamics is used to achieve better diagnostic results wrt reasoning in a single state, it assumes that the dynamics of the fault is such that all abnormal manifestations arise and are detected in the same qualitative state (Theseider Dupré & Panati 1998). This means that all abnormal data are considered relative to a single snapshot, and therefore no temporal information is associated with the data in the table used by the compiler and then in the decision tree. Such a restrictive assumption clearly limits the generality and efficacy of the approach.

In this paper we propose an extension in which we take into account temporal information associated with the model, with the manifestations of faults and with the decisions to be made. This, as we shall see in the following, leads to new interesting problems and to a completely different way of organizing and compiling the decision trees to be used as the on-board diagnostic system. Moreover, the approach allows us to generate more discriminatory decision trees that exploit the additional power provided by temporal information.

Temporal diagnostic decision trees

Defining and compiling decision trees is very simple in the atemporal case. Each node of the tree corresponds to an observable (e.g., a sensor reading) and has several descendants depending on the qualitative values that we associate with the sensor. The leaves of the tree correspond to actions that can be performed on board. Thus, given the available sensor readings, the tree can be very easily used to make a decision on the recovery action to be performed.

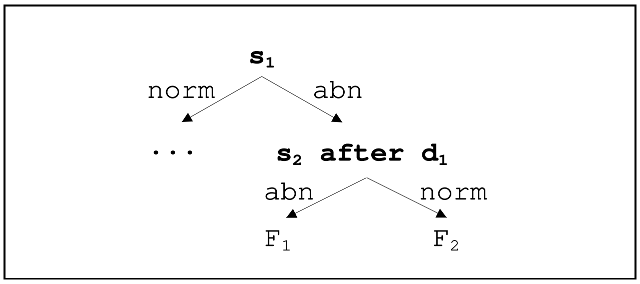

Figure 1: A simple example of temporal decision tree

New opportunities and problems have to be faced in the temporal case. The underlying device evolves across time and each fault leads to a different evolution, i.e., to different manifestations (observable through sensor readings), possibly at different times. For each fault we also have temporal constraints on the recovery action to be performed. Usually these constraints are imposed by safety or by the need of avoiding dramatic consequences to the device. For example, in case of fuel leakage in the engine, the action of stopping the engine must be performed immediately in order to avoid fire.

In the definition of the decision tree we can exploit the possibility of distinguishing between faults depending on temporal information. For example, if we can represent time in the tree, we can discriminate faults that lead to the same manifestations with different timing. Let us consider the case of two faults F_1 and F_2 that both produce an abnormal value for sensor s_1 but such that F_1 produces an abnormal value for sensor s_2 after time d_1 while F_2 produces the same abnormal value for s_2 after time $d_2 > d_1$, or F_2 does not give rise to the abnormal value. In the atemporal case a decision tree cannot distinguish between these two faults. On the contrary, in the temporal case we can build a tree like the one in figure 1. This means that after observing an abnormal value for s_1, we can wait and make a decision after d_1 units of time, depending on the fact that at that time we observe an abnormal reading for s_2 as well. However, there is a further constraint to be taken into account: the tree above is reasonable only in case the recovery actions associated with faults F_1 and F_2 allow the system to wait for d_1 units of time, i.e., in case they do not need to be performed before d_1 units of time after fault detection (that in this example corresponds to observing an abnormal value for s_1).

Thus, timing information leads to the definition of a more sophisticated type of diagnostic decision tree that has a greater discriminatory power and allows the design of more accurate on-board diagnostic systems. In particular, we introduce the notion of *temporal diagnostic decision tree*. This is a decision tree such that:

- Each node is labelled with a pair $\langle s, t\rangle$, where s is the name of an observable (sensor) and t is a time delay.
- The root corresponds to the sensor used for fault detection; in case more than one sensor may be used for such a purpose we generate multiple decision trees, one for each sensor.
- Each edge is labelled with one of the qualitative values of the sensor associated with its starting node.
- Leaves correspond to recovery actions.

During the diagnostic process the tree is used as follows:

1. Start from the root when a fault is first detected.
2. When moving to a node $\langle s, t\rangle$, wait for t units of time and then (i) read sensor s, (ii) immediately move to the descendant of s corresponding to the observed value for the sensor.
3. When moving to a leaf, perform immediately the corresponding recovery action.

In the following sections we present how a temporal diagnostic decision tree can be generated automatically. Before describing that, however, we must first briefly discuss the assumptions we make as regards the underlying model-based diagnostic approach.

The underlying model-based approach

The architecture in (Cascio *et al.* 1999) relies on a model-based diagnostic approach that makes some assumptions on the way temporal and dynamic behavior is modeled. As regards the temporal dimension in diagnosis, such an approach is either static diagnosis (i.e. diagnosis based on a static model, or based on reasoning in a single state of the system) or a limited form of temporal diagnosis, in the general sense used in (Brusoni *et al.* 1998), where *temporal* models include dynamic models, but also models where the dynamics has been abstracted away and temporal relations (e.g. delays) are given between faults and observations. In particular, in order to generate decision trees as in (Cascio *et al.* 1999), the notion of dynamic diagnosis in (Theseider Dupré & Panati 1998) can be used: it relies on a qualitative differential equations model and on a definition of diagnosis where abnormal manifestations are assumed to be detected in the same qualitative state, i.e.:

- The model of time is purely qualitative (a sequence of qualitative states).
- Abnormal manifestations are assumed to be acquired in a single snapshot t and therefore matched with a single qualitative state.
- Normality of observations in previous times wrt t is used to rule out diagnoses: i.e. if we observed abnormal manifestations relative to a snapshot t, and fault F predicts a sequence of states including one matching t, but also including a previous state t' with abnormal manifestations, F is ruled out as a possible diagnosis since the previous state was not detected (this involves assumptions about fault detection, in particular that detection at t' can be as good as detection at t, see (Theseider Dupré & Panati 1998) for further details).

What is *not* exploited is the ability of discriminating among explanations for the observations at snapshot t using information acquired at *subsequent* snapshots, which is what we intend to exploit in the present paper. In other words, we want to extend the approach to other forms of temporal models and other notions of temporal diagnosis.

In particular, we consider models of temporal behavior in which temporal information is associated with the relations describing the behavior of a device (of its components). We do not make specific assumptions on the model of time, even though, as we shall see in the following, this has an impact on the way the set of cases for the learning algorithm has to be generated. Similarly, we do not make specific assumptions on the notion of diagnosis being adopted (in the sense of (Brusoni *et al.* 1998)). This means that the starting point of our approach is the table containing the results of running the model-based diagnostic system on a set of cases, (almost) independently of the model-based diagnostic system that has been used for generating the table.

However, some discussion is necessary on the type of information that must be available in that table, which must relate possible faults to observations across time. In the static case, with finite qualitative domains, the number of possible combinations of observations is finite, and usually small, therefore there are two equivalent ways of building the table:

1. *Simulation approach:* for each fault F, we run the model-based system to predict the observations corresponding to F.
2. *Diagnostic approach:* we run a diagnosis engine on combinations (all relevant combinations) of observations, to compute, the candidate diagnoses for each one of these cases.

In either case, the resulting decision tree contains the same information as the table; if, once sensors are placed in the system, observations have no further cost, the decision tree is just a way to save space and speed up table lookup.

In the temporal case, if predictions are purely qualitative, we cannot build a table with temporal information on observations by prediction, while we can do it running the diagnosis engine on a set of cases with quantitative information: diagnoses which make qualitative predictions that are inconsistent with the quantitative information can be ruled out. Of course, this cannot in general be done exhaustively, even if the observations are assumed to be acquired at discrete times; if it is not, the decision tree generation will be really *learning* from examples.

Thus a prediction approach can be used only in case the temporal constraints in the model are precise enough to generate predictions on the temporal location of the observations (e.g., in case the model included quantitative temporal constraints), while a diagnostic approach can be generate also in case of weaker (qualitative) temporal constraints in the model.

As regards the observations, we consider the general case where a set of snapshots is available; each snapshot is labelled with the time of observation and reports the value of the sensors (observables) at that time. This makes the approach suitable for different notions of time in the underlying model and on the observations (see again the discussion in (Brusoni *et al.* 1998)).

In the approach we require that some knowledge is available on the recovery actions that can be performed on-board. In particular:

- A recovery action a_i is associated with each fault F_i.
- A cost $c(a_i) \in \mathbb{R}$ is associated with each recovery action a_i.
- A partial order $\prec$ is defined on recovery actions: $a_i \prec a_j$ means that action a_j is stronger than a_i, in the sense that all recovery effects produced by a_i are produced also by a_j. This means that a_j can be performed to recover from a fault F_i (whose associated recovery action is a_i) although in this case the action is stronger than needed. For example, two recovery actions that we considered in the test case in (Cascio *et al.* 1999) are limp home and reduce performance: in this case we have that reduce performance $\prec$ limp home as the latter correspond to a very strong reduction of car performances which simply allows the driver to reach the first workshop. If two actions are not ordered wrt the $\prec$ relation then they are independent. Thus actions define a lattice, whose bottom corresponds to performing no action and whose top corresponds to the strongest action, e.g., stop engine in the automotive domain.
- Costs are increasing wrt $\prec$, that is, if $a_i \prec a_j$ then $c(a_i) < c(a_j)$.
- Actions can be combined, i.e., multiple recovery actions can be performed on the system under examination. In particular, given two actions a_1 and a_2 we define the compound action $A = merge(a_1, a_2)$ as follows:
$$A = \begin{cases} \{a_1\} & \text{if } a_2 \prec a_1 \\ \{a_2\} & \text{if } a_1 \prec a_2 \\ \{a_1, a_2\} & \text{otherwise} \end{cases}$$
The merge can be extended to compound actions: $merge(A_1, A_2) = (A_1 \cup A_2) \setminus \{a_i \in (A_1 \cup A_2) \mid \exists a_j \in (A_1 \cup A_2) \text{ s.t. } a_i \prec a_j\}$
Notice that a simple action can be seen as a special case of compound action identifying a_i with $\{a_i\}$. In the following we will use the term "action" for both simple and compound ones. In general we have that $A_1, A_2 \preceq merge(A_1, A_2)$.
- For a compound action A the cost is $c(A) = \sum_{a_i \in A} c(a_i)$.

In this way we can not only associate a recovery action with single faults but also with multiple faults: given a diagnosis D corresponding to multiple faults, i.e., $D = \{F_1, F_2, \ldots, F_m\}$ such that a_i is the recovery action associated with each fault F_i, then the recovery action A associated with D is the merge of the actions a_i $(i = 1, \ldots, m)$.

The possibility of merging actions is important also in those cases where we cannot discriminate a set of candidate diagnoses. Suppose two diagnoses D_1 and D_2 (possibly involving multiple faults) cannot be discriminated, either because no further sensors are available or there is no time to get further sensor readings. Some action has to be performed, taking into account that we do not know the actual fault. A cautious solution is to consider the recovery actions associated with the two diagnoses and merge them.

We now have all the elements to describe the structure of the table in which we collect the results of applying the model-based system on the set of simulated cases:

- The table has one row for each case.
- Each row contains:
  - the set of observations corresponding to the case. In particular, for each observable S_j we have the value in each snapshot (this will be denoted as $S_j^{(i,t)}$ where t identifies the snapshot and i the row of the table);
  - the recovery action to be performed and its cost. This is determined by considering the set of candidate diagnoses for the case under examination and, in case of multiple faults or multiple candidate diagnoses, merging the recovery actions associated with the individual faults occurring in the diagnoses;
  - a probability which corresponds to the prior probability of the candidate diagnoses associated with the row;
  - the maximum number of snapshots that can be used for completing the diagnosis.

This table is the input to the algorithm for generating the temporal diagnostic decision tree.

Generating the tree

In this section we present the algorithm for generating temporal diagnostic decision trees. The algorithm we present is a variant of the usual ID3 one since it exploits entropy to obtain a tree with a small depth, but it has a significant improvement for the temporal case, as it is able to find a tree with the minimal expected cost.

An important issue to consider at this point is that of optimality. A standard decision tree is said to be optimal (with respect to the set of decision trees solving a particular decision problem) if its depth is minimal. The ID3 algorithm exploits the quantity of information (or its inverse, entropy) carried by an observation as a heuristic to build a decision tree that is, if not optimal, at least good enough.

For *temporal diagnostic decision trees* depth is only a secondary issue, since the ability to select a suitable recovery action in the available time becomes more important. A *suitable* action is one that has the proper recovery effects, but that is not more expensive than necessary. Each time we perform a merge operation between actions (and that happens each time we cannot discriminate further), we end up with performing an action that could have a higher cost than needed.

For each tree we can compute the expected cost of its leaves in the following way:

- the expected cost of a leaf $\langle A \rangle$, where A is a recovery action, is $c(A)$;
- if a node partitions a set of candidates $\mathcal{R}$ in $\mathcal{R}_1, \ldots, \mathcal{R}_n$ then the expected cost for $\mathcal{R}$, denoted by $\mathbb{E}(\mathcal{R})$, is:

$$\mathbb{E}(\mathcal{R}) = \sum_{i=1}^{n} \mathbb{E}(\mathcal{R}_i) P(\mathcal{R}_i \mid \mathcal{R})$$

  where $P(\mathcal{R}_i \mid \mathcal{R})$ can be computed from the prior probabilities P_j of the single cases as $\frac{\sum_{R_j \in \mathcal{R}_i} P_j}{\sum_{R_h \in \mathcal{R}} P_h}$.

Therefore, we say that a *temporal decision tree* is optimal if:

1. it has the minimum expected cost and
2. it has the minimum depth between trees with the same expected cost.

The algorithm is essentially recursive. Each recursive call is composed of the following steps: *(i)* it takes as input a set of candidate situations, represented by table rows; *(ii)* it selects a pair $\langle observable, time \rangle$ for the current tree node; *(iii)* it partitions the set of rows accordingly to the possible values of the selected attribute; *(iv)* it generates a child of the current node for each block of the partition; *(v)* it performs a recursive call on each child.

The difficult step is *(ii)*; as we said before there are two parameters to take into account: the expected cost of the tree we will obtain, and its depth. The second requirement is no problem, since the tree depth can be limited selecting the observation with minimum entropy. But while entropy can be computed locally, finding the expected cost requires building the entire subtree and is therefore computationally more expensive.

The idea is that we can limit the number of pairs $\langle observable, time \rangle$ we consider, and moreover we can limit the number of pairs for which we have to build the entire subtree. We have the following property:

Proposition. Let $\mathcal{R}$ be a set of cases for which we have to build a decision tree ($\mathcal{R}$ could be the initial table of cases, or a subset in a node of a partially built tree). Moreover, let $\langle S_i, t_1 \rangle, \langle S_j, t_2 \rangle$ be two eligible pairs for partitioning $\mathcal{R}$, and C_1, C_2 denote the minimum cost that can be achieved selecting the former or the latter respectively. Then

$$t_1 \leq t_2 \Longrightarrow C_1 \leq C_2.$$

Proof. Let $v_1, \ldots, v_n$ denote the possible values for observable S_i. Then $\langle S_i, t_1 \rangle$ will partition $\mathcal{R}$ in $\mathcal{R}_1, \ldots, \mathcal{R}_n$. We denote by P_h the probability for a case in $\mathcal{R}$ of falling into R_h. For each $\mathcal{R}_h$ let us consider all the subtrees obtained selecting as next observation $\langle S_j, t_2 - t_1\rangle$, and let us denote by C'_h the cost of the least expensive one. Since for any h, $\mathcal{R}_h \subseteq R$ we have that $C'_h \leq C_2$. Now let us consider the expected cost C' of the tree for $\mathcal{R}$. The following holds:

$$C' = \sum_{h=1}^{n} P_h C'_h \leq \sum_{h=1}^{n} P_h C_2 = C_2 \sum_{h=1}^{n} P_h = C_2.$$

But since C_1 is the minimum cost of a tree for $\mathcal{R}$ starting with $\langle S_i, t_1\rangle$, we have also

$$C_1 \leq C' \leq C_2.$$

This leads to two interesting consequences:

1) all the subtrees spreading out from pairs in the same snapshot have the same expected cost. This means that if we have to choose between $\langle S_i, t\rangle$ and $\langle S_j, t\rangle$ we can simply select the one with minimum entropy.

   This allows us to consider, for each snapshot, only the observation with minimum entropy.

2) if we order pairs temporally, we also have them ordered by expected cost. Thus it suffices to find the latest observation $\langle S_L, t_L\rangle$ that has the same cost as the first (let us denote it by $\langle S_F, t_F\rangle$), and then select the pair with minimum entropy among $\langle S_i, t\rangle$ with $t_F \leq t \leq t_L$.

   To find $\langle S_L, t_L\rangle$ we can exploit a binary search, significantly reducing the number of pairs for which we compute expected cost.

The algorithm

The algorithm for compiling the tree is reported in figures 2 and 3.

To start the algorithm up, the sensors used for fault detection must be determined, since a separate tree will be built for each of them. Notice that each tree covers a subset of cases, since in different cases the abnormal measurement that activates diagnosis can be on a different sensor.

Then, given the sensor used as the root of the tree, the values that the sensor can assume induce a partition on the considered set of cases, and a subtree must be generated for each block of the partition. This is precisely the purpose of the TEMPORALID3 function (see figure 2), that takes in input a set of cases $\mathcal{R} = \{R_1, \ldots, R_H\}$ and returns a temporal decision tree along with its expected cost.

It is worth noting that, in order to express delays instead of absolute times, TEMPORALID3 needs a notion of *current snapshot* that varies from one call to another, and that corresponds to the last snapshot considered. More precisely, since each call to TEMPORALID3 builds a subtree, the current snapshot is the one in which the parent node's observation takes place. However, the way in which the tree root is built gives rise to a problem. We have previously said that the tree root is activated when the fault is detected, but for each case this can happen with a different delay wrt fault occurrence. Therefore each candidate R_i in the input set of cases $\mathcal{R}$ has to maintain a pointer to its own *current snapshot*; we denote this pointer by t_i. At the beginning, for each row R_i of the table, t_i is set to the snapshot at which the corresponding fault (or combination of them) is detected.

Moreover, we denote by P_i the prior probability of R_i, by T_i the maximum time by which a recovery action has to be performed for case R_i, and by A_i the proper recovery action (either simple or compound).

Let us briefly describe how TEMPORALID3 behaves. First, it considers all those situations in which it should simply return a tree leaf. This happens when: *(i)* the input set does not need further discrimination, that is, all the cases in $\mathcal{R}$ have the same recovery action; or *(ii)* in the time left for selecting an action there are no more discriminating observables (a particular case is when there is no time left at all). This means that it is not possible to distinguish between $R_1, \ldots, R_H$ and therefore the selected recovery action is the merge of $A_1, \ldots, A_H$.

If none of the previous cases holds, TEMPORALID3 builds for each available snapshot w a pair $\langle S_j, w\rangle$ where o_j is an observable with minimum entropy over w. These pairs are then enumerated in temporal order obtaining a sequence $e_1, \ldots, e_W$; as we previously said this means that they are also in order of expected cost. At this point a binary search is used to find the latest pair e_{L} that generates subtrees with minimal expected cost. The selected pair $\langle S_{\mathsf{ok}}, w_{\mathsf{ok}}\rangle$ is the one with minimum entropy between $e_1, \ldots, e_{\mathsf{L}}$, and consequently between all pairs leading to the minimum expected cost.

Finally, TEMPORALID3 updates the current snapshot pointers and calls the auxiliary function BUILDTREE (see figure 3) that partitions $\mathcal{R}$ accordingly to the selected observation and calls recursively TEMPORALID3 on each block of the partition.

An example

Let us consider a simple example. There are three sensors, S_1, S_2 and S_3; each one of them can have five possible values: normal (n), high (h), low (l), very low (v) and zero (z). Possible recovery actions are denoted by a_1, a_2, a_3, a_4; we assume the following partial order: $a_4 \prec a_2 \prec a_1$, $a_4 \prec a_3 \prec a_1$. We assume that the faults associated with the four rows have the same prior probability. Initial data are shown in table 1.

We will build the tree for the sensor S_2; indeed each fault (or combination of faults) represented in the table is first detected on S_2, therefore this is the only tree that has to be built. If, for example, cases R_1 and R_2 were first detected on S_2 while R_3 and R_4 on S_1, we would have had to build two separate trees, the first with root S_2 dealing with R_1, R_2 and the second with root S_1 dealing with R_3, R_4.

function TEMPORALID3($\mathcal{R}$) **returns** a pair (*Tree*, *Cost*)
begin
  {*First let us analyze cases in which the current node becomes a leaf of the tree*}
  if $A_1 = \ldots = A_H$ **then**   {*all cases in* $\mathcal{R}$ *have the same recovery action*}
    return ($Tree = \langle A \rangle$, $Cost = c(A)$);
  $T \longleftarrow \mathsf{min}\{T_i - t_i \mid i = 1, \ldots, H\}$;   {*remaining time*}
  $A \longleftarrow merge(A_1, \ldots, A_H)$;
  if $T < 0$ **then**   {*no time left*}
    return ($Tree = \langle A \rangle$, $Cost = c(A)$);
  {*For each remaining snapshot collect those sensors that provide some useful discrimination*}
  for each w in $\{0, \ldots, T\}$ **do**
    $Sens_w \longleftarrow \{S_j \mid \text{there are } R_i, R_{i'} \text{ such that } S_j^{(i, t_i + w)} \neq S_j^{(i', t_{i'} + w)} \text{ and } A_i \neq A_{i'}\}$;
  if $\forall w\ Sens_w = \emptyset$ **then**   {*there are no more useful observations*}
    return ($Tree = \langle A \rangle$, $Cost = c(A)$);
  {*For each snapshot find an observation with minimal entropy, and enumerate the sequence in temporal order*}
  $i \longleftarrow 0$;
  for each w in $\{0, \ldots, T\}$ **do**
    if $Sens_w \neq \emptyset$ **then begin**
      $i \longleftarrow i + 1$;
      $e_i \longleftarrow$ a $\langle S_j, w \rangle$ such that for all $S_{j'} \in Sens_w$, $\langle S_{j'}, w \rangle$ has an entropy $\geq$ that of $\langle S_j, w \rangle$;
    end
  $W \longleftarrow i$;   {*the sequence is thus* $e_1, \ldots, e_W$}
  {*Find through binary search the latest snapshot that leads to the minimum expected cost*}
  $(Tree_{\mathsf{min}}, Cost_{\mathsf{min}}) \longleftarrow$ BUILDTREE($\mathcal{R}, e_1$);
  $(Tree_{\mathsf{max}}, Cost_{\mathsf{max}}) \longleftarrow$ BUILDTREE($\mathcal{R}, e_W$);
  if ($Cost_{\mathsf{min}} = Cost_{\mathsf{max}}$) **then**   {*All* e_i *have the same expected cost*}
    $\mathsf{L} \longleftarrow W$;
  else
    $\mathsf{L} \longleftarrow$ BINARYSEARCH($\mathcal{E}, (e_1, \ldots, e_W), Cost_{\mathsf{min}}$);
  $\langle S_{\mathsf{ok}}, w_{\mathsf{ok}} \rangle \longleftarrow$ the earliest element of $e_1, \ldots, e_{\mathsf{L}}$ with minimal entropy;
  {*Update the value of the* current *snapshot for each candidate*}
  for each $R_i \in \mathcal{R}$ **do**
    $t_i \longleftarrow t_i + w_{\mathsf{ok}}$;
  {*Build the subtree and return it*}
  return BUILDTREE($\mathcal{R}, \langle S_{\mathsf{ok}}, w_{\mathsf{ok}} \rangle$);
end.

Figure 2: The function TEMPORALID3

The root of the tree is thus $\langle S_2 \rangle$ but it does not partition the set of candidates, so we still have $\mathcal{R} = \{R_1, R_2, R_3, R_4\}$, and $t_1 = t_2 = t_3 = t_4 = 1$. In this example all faults show themselves with the same delay wrt fault occurrence; this is the reason why all t_is have the same value. In a different situation each t_i would point to the first snapshot for which sensor S_2 has a non-normal reading. It turns out that diagnosis must be completed in three snapshots, and we get $Sens_0 = \emptyset, Sens_1 = Sens_2 = \{S_2\}, Sens_3 = \{S_1, S_2\}$. The computation of entropy leads then to the sequence $e_1 = \langle S_2, 1 \rangle, e_2 = \langle S_2, 2 \rangle, e_3 = \langle S_2, 3 \rangle$.

Subtree for e_1: e_1 partitions the set of candidate diagnoses in $\mathcal{R}_1 = \{R_1, R_4\}$ and $\mathcal{R}_2 = \{R_2, R_3\}$, with $t_1 = t_2 = t_3 = t_4 = 2$. Let us consider first $\mathcal{R}_1$: there are two snapshots left so $Sens_0 = \emptyset, Sens_1 = Sens_2 = \{S_2\}$ and $e_1 = \langle S_2, 1 \rangle, e_2 = \langle S_2, 2 \rangle$. Since both conclude the diagnosis, we can safely choose the first one. As regards $\mathcal{R}_2$, there are three snapshots left and $Sens_0 = Sens_1 = \emptyset, Sens_2 = \{S_1, S_2\}, Sens_3 = \{S_1, S_3\}$; all available pairs conclude the diagnosis and a reasonable choice could be $\langle S_2, 2 \rangle$. In the description of the algorithm we did not specify how to choose a pair when there are many that have the same entropy and lead to the same expected cost. This can depend on the particular implementation; for example it could be useful to choose the earliest pair, or to limit the number of sensor used.

Subtree for e_3: The partitions induced by e_3 is $\mathcal{R}_1 = \{R_1, R_2\}, \mathcal{R}_2 = \{R_3\}, \mathcal{R}_3 = \{R_4\}$. For both $\mathcal{R}_2$ and $\mathcal{R}_3$ the algorithm generates tree leaves, respectively $\langle a_3 \rangle$ and $\langle a_4 \rangle$. As to $\mathcal{R}_1$, we find out that diagnosis has to be completed in the current snapshot, and this cannot be done. Therefore the algorithm generates a tree leaf with the action $merge(a_1, a_2) = a_1$. It is obvious that the expected cost of this subtree is higher than that

function BUILDTREE($\mathcal{R}$, $\langle S_i, w\rangle$) **returns** a pair (*Tree*, *Cost*)
begin
  $\Pi = \{\mathcal{R}_1, \ldots, \mathcal{R}_S\} \longleftarrow$ partition of $\mathcal{R}$ induced by (S_i, w);
  {*Build a subtree for each block in the partition*}
  for each $\mathcal{R}_j \in \Pi$ **do**
    $(Tree_j, Cost_j) = \text{ID3TEMP}(\mathcal{R}_j)$;
  return $(Tree = \langle\langle S_j, w\rangle, Tree_1, \ldots, Tree_S\rangle, Cost = \sum_{j=1,\ldots,S} \frac{\sum_{R_i \in \mathcal{R}_j} P_i}{\sum_{R_i \in \mathcal{R}} P_i} \cdot Cost_j)$;
end.

Figure 3: The function BUILDTREE

Row	S_1						S_2						S_3						Action	T_i	P_i	Cost
	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5				
R_1	n	n	n	n	v	v	n	l	v	v	v	v	n	n	n	n	n	l	a_1	4	$\frac{1}{8}$	100
R_2	n	n	n	n	v	v	n	l	l	l	v	v	n	n	n	n	n	v	a_2	5	$\frac{1}{8}$	10
R_3	n	n	n	n	h	h	n	l	l	l	l	v	n	n	n	n	n	h	a_3	5	$\frac{1}{8}$	10
R_4	n	n	n	n	v	v	n	l	v	z	z	z	n	n	n	n	n	z	a_4	4	$\frac{1}{8}$	5

Table 1: A sample table of data

of the subtree for e_1, thus the algorithm moves to the subtree for e_2.

Subtree for e_2: e_2 partitions the initial set in $\mathcal{R}_1 = \{R_1\}, \mathcal{R}_2 = \{R_2, R_3\}, \mathcal{R}_3 = \{R_4\}$. Like in the previous case, for $\mathcal{R}_1$ and $\mathcal{R}_3$ the algorithm generates the tree leaves $\langle a_1\rangle$ and $\langle a_4\rangle$. For $\mathcal{R}_2$ we have $Sens_0 = \emptyset, Sens_1 = \{S_1, S_2\}, Sens_2 = \{S_1, S_3\}$. All available pairs have the same entropy and conclude the diagnosis, so we can choose for example $\langle S_2, 1\rangle$.

The subtree for e_2 has the same expected cost as the subtree for e_1; it turns out that the entropy of e_2 is less than that of e_1 and therefore the algorithm chooses the former. The resulting decision tree is shown in figure 4.

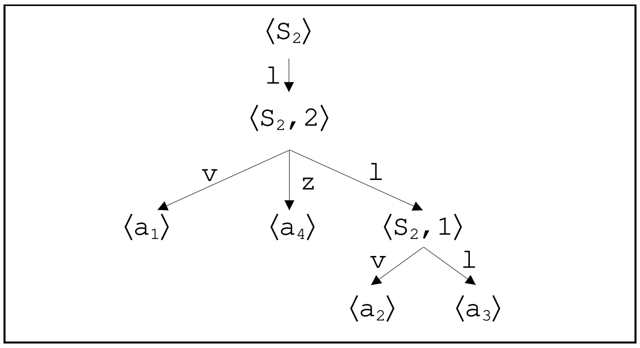

Figure 4: Resulting decision tree

Conclusions

In this paper we introduced a new notion of diagnostic decision tree that takes into account temporal information on the observations and temporal constraints on the recovery actions to be performed. In this way we can take advantage of the discriminatory power that is available in the model of a dynamic system. We presented an algorithm that generates temporal diagnostic decision trees from a set of examples, discussing also how an optimal tree can be generated.

The automatic compilation of decision trees seems to be a promising approach for reconciling the advantages of model-based reasoning and the constraints imposed by the on-board hardware and software environment. It is worth noting that this is not only true in the automotive domain and indeed the idea of compiling diagnostic rules from a model has been investigated also in other approaches (see e.g., (Darwiche 1999; Dvorak & Kuipers 1989)). (Darwiche 1999), in particular, discusses how rules can be generated for those platforms where constrained resources do not allow a direct use of a model-based diagnostic system.

We are currently investigating the possibility of generalizing our approach to integrate actions and candidate discrimination, e.g. including probing actions (which modify the system state) and sequences of actions interleaved with measurements.

References

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

Cascio, F.; Console, L.; Guagliumi, M.; Osella, M.; Panati, A.; Sottano, S.; and Theseider Dupré, D. 1999. Generating on-board diagnostics of dynamic automotive systems based on qualitative deviations. *AI Communications* 12(1):33–44. Also in Proc. 10th Int. Workshop on Principles of Diagnosis, Loch Awe, Scotland.

Console, L., and Dressler, O. 1999. Model-based diagnosis in the real world: lessons learned and challenges remaining. In *Proc. 16th IJCAI*, 1393–1400.

Darwiche, A. 1999. On compiling system descriptions into diagnostic rules. In *Proc. 10th Int. Work. on Principles of Diagnosis*, 59–67.

Dvorak, D., and Kuipers, B. 1989. Model-based monitoring of dynamic systems. In *Proc. 11th IJCAI*, 1238–1243.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning* 1:81–106.

Sachenbacher, M.; Malik, A.; and Struss, P. 1998. From electrics to emissions: experiences in applying model-based diagnosis to real problems in real cars. In *Proc. 9th Int. Work. on Principles of Diagnosis*, 246–253.

Theseider Dupré, D., and Panati, A. 1998. State-based vs simulation-based diagnosis of dynamic systems. In *Proc. DX 98, 9th Int. Workshop on Principles of Diagnosis*.