Diagnosing Delays in Multi-Agent Plans Execution

Diagnosing Delays in Multi-Agent Plans Execution

Roberto Micalizio and **Gianluca Torta**[1]

Abstract. The paper introduces the notion of Temporal Multi-Agent Plan (TMAP) and proposes a methodology, based on Simple Temporal Problems (STP), for detecting and diagnosing action execution delays. Actions are characterized by a finite set of behavioral modes, and each behavioral mode is a continuous interval of possible durations of the action. Nominal modes represent the expected durations, whereas faulty modes represent delays.
Solving such diagnostic problems requires to find an assignment of modes to the actions that is consistent with the received observations and maximizes the likelihood of the delayed durations. An implementation of the approach and some preliminary experimental results are also discussed.

1 INTRODUCTION

The diagnosis of the execution of a multi-agent plan (MAP) - i.e., a plan assigned to a team of (cooperating) agents - has recently been addressed in a number of works (see e.g., [6, 2, 5]), proposing different notions of plan diagnosis and different diagnostic methodologies. These works assume that the MAP is correct, but during its execution action failures can happen as a consequence of unexpected events such as faults in the functionalities of the agents, or unpredictable changes in the environment.

All these works, however, do not consider the temporal dimension; that is, action delays are never taken into account as possible sources of plan execution anomalies. In many practical situations this appears to be a strong limitation since the MAP is often enriched with a schedule, and hence intermediate deadlines and the temporal constraints between actions have to be satisfied at execution time.

To the best of our knowledge, only the work by Roos and Witteveen [7] has addressed the diagnosis of delayed actions. In their approach, each fault mode is a real number δ representing a specific delay from the nominal duration interval, and observations are given as uncertain time intervals. They propose a notion of preferred diagnosis (*maximum confirmation*) which is based on the uncertainty of observations and, instead of trying to maximize the probability of action delays, tries to maximize the coverage of the intervals associated with observations.

In this paper we propose a different approach to the problem; our main objective is to provide the user (i.e., a human supervising the plan execution) with high-level explanations that directly mention the delayed actions in the plan, and that maximize the probability of such action delays.

To this end, following [6, 2], we map our multi-agent setting into the Model-Based Diagnosis (MBD) framework: the MAP becomes the model of the system to be diagnosed, the actions in the plan are the system components, and constraints between actions represent the connections between the components.

As usual in MBD, we associate each action with a finite set of behavioral modes. Since our interest is in diagnosing action delays, behavioral modes are duration intervals: the nominal mode is a range of acceptable durations for a given action; faulty modes, on the other hand, are intervals of delayed durations of increasing order (and decreasing probability) that can affect the action.

In this paper we adopt the consistency-based notion of diagnosis: a MAP diagnosis is a subset of actions whose non-nominal behavior is consistent with the observations received so far; and we propose a methodology to infer the set of all the diagnoses with *minimal rank* [4], i.e., with the highest (order-of-magnitude) likelihood.

The general framework within which we situate the diagnostic task is a loop of control including plan monitoring and diagnosis. The objective is to keep the system diagnosis updated along time; as soon as new observations are available, the monitoring is in charge of verifying whether the current diagnostic hypothesis is still valid (detection). If not, new diagnostic hypotheses must be inferred.

The paper is organized as follows. In the next section we briefly review some basic notions about Simple Temporal Problems and Multi-Agent Plans. In section 3 we introduce our Temporal Multi-Agent Plan (TMAP) framework and the notion of Delayed Action Execution (DAX) diagnostic problem. In sections 4 and 5 we discuss how a DAX problem can be detected and solved, respectively. In section 6 we discuss a possible implementation of the approach supported by some preliminary results. Finally, the conclusions.

2 BACKGROUND

Simple Temporal Problems. A Simple Temporal Problem (STP) [3] is a special case of TCSP (Temporal Constraint Satisfaction Problem), consisting of a set V of variables (which represent real-valued time points), and a set of constraints C of the form:

$$a_{ij} \leq X_j - X_i \leq b_{ij} \tag{1}$$

which can be expressed as the pair of inequalities:

$$X_j - X_i \leq b_{ij}; \quad X_i - X_j \leq -a_{ij} \tag{2}$$

where $X_i, X_j \in V$ and a_{ij}, b_{ij} denote the lower and the upper bound, respectively, of the time interval between the events X_i, X_j.

A solution of an STP is an assignment to the variables in V that does not violate any constraint in C. A *consistent* STP is an STP with at least one solution. The consistency of an STP can be checked by translating it into a *distance graph* $\mathcal{G} = \langle V, E \rangle$, where V is the set of time point variables as before, and E is a set of directed edges between the variables in V. For each constraint in (2), a weighted edge from X_i to X_j (resp. from X_j to X_i) with weight b_{ij} (resp.

[1] Dipartimento di Informatica, Università di Torino, Italy, email: {micalizio,torta}@di.unito.it

$-a_{ij}$) is added to E. A special node z is added to V and represents the event from which time starts. The STP is consistent iff its associated distance graph has no negative cycles [3]. This condition can be checked by exploiting well-known algorithms, such as the Floyd-Warshall algorithm which has time complexity $O(|V|^3)$, or (for sparse graphs) the Johnson algorithm which has time complexity $O(|V|^2 \cdot log(|V|) + |V| \cdot |E|)$.

Multi-Agent Plans. In this paper we consider a simplified notion of Multi-Agent Plan (MAP) in order to better focus on the time extensions; a MAP P is a tuple $\langle \mathcal{T}, A, R \rangle$ where:

- $\mathcal{T}$ is the team of agents;
- A is a set of action instances; each action is assigned to a specific agent in $\mathcal{T}$;
- R is a partial order relation over A; each $p \in R$ is a pair $\langle a, a' \rangle$, $a, a' \in A$, meaning that the execution of a must precede the execution of a'; we say that a is a *predecessor* of a', and a' is a *successor* of a.

The synthesis of a MAP is out of the scope of the paper; see e.g., [1].

3 DIAGNOSING DELAYED ACTIONS: FRAMEWORK

In this paper we are interested in detecting and explaining delays in the execution of the actions in a given MAP. Thus we extend the MAP formalization with temporal information. Note that we are not just interested in modeling durative actions; in our diagnostic process, in fact, we aim at identifying the (possibly anomalous) *behavioral modes* of the actions performed so far. For this reason, we first formalize the notion of Temporal MAPs (TMAP), and then we formalize the Delayed Action Execution (DAX) diagnostic problem.

3.1 Temporal MAPs

A Temporal MAP (TMAP) is a tuple $\langle \mathcal{T}, A, R, \mathcal{M}, modes \rangle$ where $\langle \mathcal{T}, A, R \rangle$ is the embedded MAP, and:

- $\mathcal{M}$ is the set of all the possible behavioral modes that can be associated with the action instances in A. Each mode $m \in \mathcal{M}$ is a triple of the form $\langle label, range, rank \rangle$:
  - $label$ is the mode name;
  - $range$ is an interval of time corresponding to the possible durations of the action when it behaves like this mode. The lower and upper bounds of the interval are denoted as $m.range.l$ and $m.range.u$, respectively; for the sake of readability, we will use $m.l$ and $m.u$ as shortcuts. The interval can be closed, open, or half-open according to the modeler's needs; possibly, $m.u$ can be $+\infty$, but $m.l$ is always a finite value in $\Re$;
  - $rank$ is a non-negative integer value representing the order-of-magnitude probability of the mode [4]: lower ranks correspond to higher probabilities. Rank zero is associated with all and only the nominal modalities.

  Given a modality $m \in \mathcal{M}$, we use the dot notation to retrieve its fields (e.g., $m.rank$).
- $modes \subseteq A \times \mathcal{M}$ is a relation that associates each action in A with a set of modes. In particular we assume that such a relation satisfies the following conditions:
  1. $\forall a \in A \; modes(a) \neq \emptyset$
  2. $\forall a \in A \; \exists! m \in modes(a) | m.rank = 0$

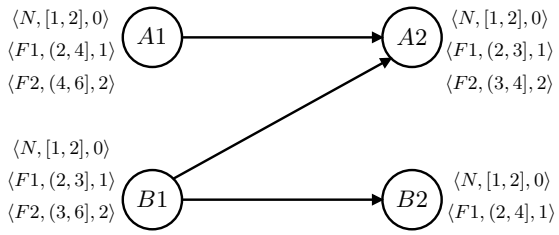

Figure 1. An example of TMAP.

3. $\forall a \in A, \forall m_l, m_k \in modes(a), m_l \neq m_k$, $m_l.range \cap m_k.range = \emptyset$
4. $\forall a \in A$, let $a.L$ be the lowest bound of the modes in $modes(a)$, and let $a.U$ be the highest bound of the modes in $modes(a)$; then, $[a.L, a.U] = \bigcup_{m \in modes(a)} m.range$

Condition 1 simply states that each action must have at least one behavioral mode, and condition 2 specifies that each action must always be associated with at least one, and no more than one, nominal mode (i.e., the interval of expected possible durations of the action). Condition 3 requires that the modes of an action a do not overlap with one another, and condition 4 imposes the continuity of the intervals of the modes. In short, the modalities associated with action a represent a partition of the interval $[a.L, a.U]$. In the next sections we will refer to such an interval as the *relaxed interval* of action a.

Finally, given an action a, we denote as $m\langle a, lab \rangle$ the mode $m \in modes(a)$ such that $m.label$ equals lab.

Example 1 *Let us consider the simple TMAP depicted as a graph in Figure 1; it involves two agents, A and B; the first is in charge of actions $A1$ and $A2$, the second of actions $B1$ and $B2$. Actions are nodes, whereas edges between nodes are precedence links between the actions; e.g., action $A2$ can be executed only after the completion of actions $A1$ and $B1$. The triplets near each node represent the modalities associated with the corresponding action. For example, $A1$ is considered to have fault* F1 *(with rank 1) if its duration is more than 2 and up to 4.*

3.2 Delayed Action Execution (DAX) Problems

Timed observations. We define a timed observation as a pair $\langle e, t \rangle$, where e is the observed event, and t is the time when e occurred. In our TMAP framework, observable events are the start a_s and the completion a_e of any action a in A; for instance, $\langle a_s, t \rangle$ means that action a started its execution at time t. Of course, during the execution of P, only a few of these events will be observed.

It is important to note that, since the agents share the same environment and resources, it is possible that the misbehavior of an agent causes cascade delays in other agents' activities. As a consequence, in solving a diagnostic problem, one has to consider that the delay observed in the completion of an action can be a consequence of a delay occurred in a previous action.

Delayed Action Execution problem. A DAX problem is a tuple $\langle P, \Delta_{cur}, O \rangle$ and arises when, given a TMAP P, a sequence of timed observations $O = \langle o_1, \ldots, o_n \rangle$, and a hypothesis Δ_{cur} (which maps each action a in P with a behavioral mode in $modes(a)$), we get an inconsistency. Intuitively, this happens when at least one timed observation $\langle e, t \rangle \in O$ is in conflict with the estimations about e that can be inferred by using P and Δ_{cur}. In section 4 we address this

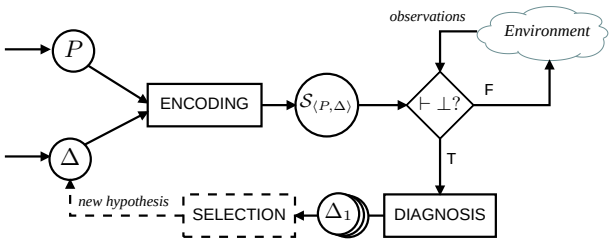

Figure 2. An outline of the proposed methodology.

problem more formally, and we discuss how inconsistencies can be detected by means of STPs.

A solution of a DAX problem, i.e., a DAX *diagnosis*, is a new mapping $\Delta : a \rightarrow m(a), a \in A$ such that:

1. $P \cup \Delta \cup O \nvdash \perp$
2. the rank $\Delta.rank = \sum_{a \in A} \Delta(a).rank$ is minimal, i.e. for each Δ' satisfying the previous conditions, $\Delta.rank \leq \Delta'.rank$

Condition 1 states that the diagnosis is correct, i.e. consistent with the timed-observations that have been received. Condition 2 is a preference criterion: we are not interested in any possible diagnosis, but in a diagnosis whose rank is minimal. Note that, since we are assuming that failures (i.e., non-nominal modes) are independent of one another, the rank of a diagnosis is the summation of the ranks associated with the assumed modes [4]. Thus we look for the diagnosis, in fact for all the diagnoses, with the highest (qualitative) probability.

4 SOLVING A DAX PROBLEM: DETECTION

The basic idea of the methodology we propose, outlined in Figure 2, is to solve a DAX problem by combining the techniques developed to solve STPs with those developed within Model-Based Diagnosis. More precisely, our methodology consists of four main steps:
1) *encoding*: given a TMAP P and a hypothesis Δ, representing the current (assumed) diagnosis of P, the encoding phase produces an STP $\mathcal{S}_{\langle P,\Delta\rangle}$. Of course, at the beginning of the plan execution phase, Δ maps each action to its nominal behavior; that is, all the actions are expected to be on-time.
2) *detection*: verifies the validity of hypothesis Δ at plan execution time: whenever new observations become available these observations are asserted within $\mathcal{S}_{\langle P,\Delta\rangle}$. If $\mathcal{S}_{\langle P,\Delta\rangle}$ results to be consistent, then Δ is still valid. Otherwise, a new diagnosis has to be inferred.
3) *diagnosis*: infers a set *MinDiag* of minimal rank diagnoses; as we will discuss, also this step relies on a translation of the TMAP into an STP.
4) *selection:* the final step consists in the selection of a new current diagnostic hypothesis out of *MinDiag*; then the process starts again with the encoding until the plan execution phase terminates.

In the rest of this section we discuss the encoding and the detection phases; the diagnosis is presented in section 5, while we leave the selection phase to a future work.

4.1 Encoding

Given a TMAP $P = \langle \mathcal{T}, A, R, \mathcal{M}, modes \rangle$ and the current diagnostic hypothesis Δ, the associated STP $\mathcal{S}_{\langle P,\Delta\rangle}$ includes the following set of constraints:

$$\bigcup_{a \in A} \Delta(a).l \leq a_e - a_s \leq \Delta(a).u \tag{3}$$

These constraints state that each action in the plan has an expected duration which depends on its mode assumed in Δ. Further constraints, however, need to be added in order to model the precedence relations existing among the actions in P. To add these constraints one has to reason about the topology of the precedence links in R. Three configurations of precedence links have to be considered:

- **split**: let us consider the precedence links $\langle a, a' \rangle$, $\langle a, a'' \rangle \in R$; action a is a direct predecessor of both a' and a''; we impose that as soon as a terminates a' and a'' start their execution by adding to $\mathcal{S}_{\langle P,\Delta\rangle}$ the constraints:

$$a'_s - a_e = 0; \quad a''_s - a_e = 0 \tag{4}$$

- **pipe**: is a special case of split where an action a' has only one direct predecessor a; thus $a'_s - a_e = 0$ is added to $\mathcal{S}_{\langle P,\Delta\rangle}$;
- **join**: for any pair of precedence links $\langle a', a \rangle$, $\langle a'', a \rangle \in R$, we would like to say that a can start only after the completion of both a' and a''; that is, as soon as the latest of them terminates. At this stage, however, we cannot say which of them will terminate last, so we add to $\mathcal{S}_{\langle P,\Delta\rangle}$ two very relaxed constraints:

$$0 \leq a_s - a'_e \leq +\infty; \quad 0 \leq a_s - a''_e \leq +\infty \tag{5}$$

These two constraints only impose that a starts after the completion of both a' and a'', but the amount of "waiting time" elapsing between the completion of the two actions and the start of a varies from 0 to $+\infty$. In the next subsection we describe how these relaxed intervals are restricted during the detection phase. Note that a join configuration represents a *synchronization point*: the agent in charge of performing a cannot go further unless the actions preceding a (even when they are assigned to different agents), have been completed. In the rest of the paper we denote with $SYNC$ the sequence of all the synchronization points in A chronologically ordered. When there is not an order relation between two synchronization points, they are non-deterministically serialized in $SYNC$ by putting one before the other.

4.2 Validating the current hypothesis

To validate the current diagnostic hypothesis Δ against a given set O of timed observations, we consider the STP $\mathcal{S}_{\langle P,\Delta\rangle}$ resulting from the encoding phase, and translate it into the associated distance graph $\mathcal{G}_{\langle P,\Delta\rangle}$=$\langle V, E \rangle$, as usual. The validation process we propose involves two basic steps:

1. Updating $\mathcal{G}_{\langle P,\Delta\rangle}$ by asserting the observations in O; the result is a new graph $\mathcal{G}_{\langle P,\Delta,O\rangle}$;
2. Reducing in $\mathcal{G}_{\langle P,\Delta,O\rangle}$ the expected waiting times for the synchronization points to validate Δ.

Step 1. The assertion of the observations is a simple task. For each observation $\langle e, t \rangle \in O$, we add in $\mathcal{G}_{\langle P,\Delta,O\rangle}$ the two following weighted edges:

$$\langle z, t, e \rangle; \langle e, -t, z \rangle \tag{6}$$

meaning that the event e (either the starting or ending of an action) must necessarily happen at time t: the distance between z and e is t.
Step 2. The second step takes the distance graph $\mathcal{G}_{\langle P,\Delta,O\rangle}$, and validates the current hypothesis Δ against O. To reach this result, it is necessary to restrict the waiting times before the synchronization points. Without this step, in fact, any observation coming after a synchronization point would always be consistent due to the relaxed constraints (5). This step is outlined in Algorithm 1. First of all, we

Algorithm 1 *ValidateHypothesis*

Require: $\mathcal{G}_{\langle P,\Delta,O\rangle}$ distance graph;
$SYNC$ sequence of synchronization points
Returns: *true* when Δ is consistent with O, *false* otherwise;
1: $\mathcal{G}_{\langle P,\Delta,O\rangle} \leftarrow$*Johnson*$(\mathcal{G}_{\langle P,\Delta,O\rangle})$
2: **if** $\mathcal{G}_{\langle P,\Delta,O\rangle}$ has a negative cycle **return** *false*
3: **for each** synchronization point $a \in SYNC$ **do**
4: let $MWT(a)$ be the maximal waiting time for a
5: let $mwt(a)$ be the minimal waiting time for a
6: $\mathcal{G}_{\langle P,\Delta,O\rangle} \leftarrow \mathcal{G}_{\langle P,\Delta,O\rangle} \cup \{\langle z, MWT(a), a_s\rangle\}$
7: $\mathcal{G}_{\langle P,\Delta,O\rangle} \leftarrow \mathcal{G}_{\langle P,\Delta,O\rangle} \cup \{\langle a_s, -mwt(a), z\rangle\}$
8: $\mathcal{G}_{\langle P,\Delta,O\rangle} \leftarrow$*Johnson*$(\mathcal{G}_{\langle P,\Delta,O\rangle})$
9: **if** $\mathcal{G}_{\langle P,\Delta,O\rangle}$ has a negative cycle **return** *false*
10: **end for**
11: **return** *true*

invoke Johnson algorithm [2] to minimize the distances between any pairs of vertices in $\mathcal{G}_{\langle P,\Delta,O\rangle}$; if a negative cycle exists, we have discovered that the predictions made by means of Δ are inconsistent with the observations, and the algorithm terminates returning *false*.

Otherwise, for each synchronization point in $SYNC$, taken in the order, we compute two measures: the *Maximal Waiting Time* (MWT), and the *minimal waiting time* (mwt).

The maximal waiting time of a synchronization point a is the maximal time span between z and a_s; i.e., it is the longest absolute time before the start of action a. Similarly, the minimal waiting time of a is the shortest absolute time after which a can start. These two measures can be easily computed as:

- $MWT(a)$= $max_{a' \in dPred(a)}$ *longest_makespan*(a')
- $mwt(a)$= $max_{a' \in dPred(a)}$ *shortest_makespan*(a')

where $dPred(a)$ is the set of direct predecessors of a, while *longest_makespan*(a') is the longest distance between the nodes z and a'_e in $\mathcal{G}_{\langle P,\Delta,O\rangle}$ and *shortest_makespan*(a') is the shortest distance between z and a'_e in $\mathcal{G}_{\langle P,\Delta,O\rangle}$. It is worth noting that, thanks to the distances computed by Johnson algorithm, such makespans can be obtained in linear time.

Thus we can predict that, according to hypothesis Δ, the starting time of action a falls within the interval $[mwt(a), MWT(a)]$. To verify whether this prediction is correct, we update the distance graph by imposing that $a_s \in [mwt(a), MWT(a)]$ (see lines 6 and 7).

Then, we invoke again Johnson algorithm on $\mathcal{G}_{\langle P,\Delta,O\rangle}$ to propagate the new constraints and check whether the distance graph contains negative cycles. If we find an inconsistency we return *false*; otherwise, the current hypothesis is still consistent and the subsequent synchronization point is considered. After having considered all the synchronization points without discovering an inconsistency, the algorithm terminates returning *true* meaning that Δ is still valid.

Example 2 *Let us consider the TMAP in Figure 1; it is easy to see that nodes $B1$ and $B2$ are in pipe, while nodes $A1$, $B1$, and $A2$ form a join with synchronization point $A2$. Figure 3 shows the corresponding distance graph when all the actions behave nominally.*

To exemplify the detection, let us suppose that after a while we receive the observation O=$\{\langle A2_e, 6\rangle\}$. To validate the nominal hypothesis, we first assert this observation into the distance graph by adding the following edges: $\langle z, 6, A2_e\rangle$ and $\langle A2_e, -6, z\rangle$. After applying Johnson to this graph, it is possible to infer that event $A2_s$ must be in the interval $[4, 5]$ (because the nominal duration of $A2$ is $[1, 2]$ and $A2_e = 6$). However, when we minimize the waiting time of synchronization point $A2$ (see Algorithm 1), we predict that $A2_s$ must be in the interval $[2, 3]$ (indeed, $MWT(A2) = 3$ and $mwt(A2) = 2$); by asserting the edges $\langle z, 3, A2_s\rangle$ and $\langle A2_s, -2, z\rangle$ in the graph and invoking Johnson algorithm, we detect an inconsistency since the intersection between the predicted and inferred intervals is empty.

[2] We use Johnson algorithm instead of Floyd-Warshall because, for typical TMAPs, $\mathcal{G}_{\langle P,\Delta,O\rangle}$ is relatively sparse; the algorithm operates just on the portion of the distance graph relevant for O; namely, those nodes preceding the observations in O.

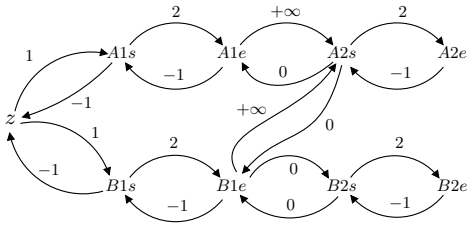

Figure 3. Distance graph of the sample TMAP with all nominal actions.

5 SOLVING A DAX PROBLEM: DIAGNOSIS

In the previous section we have discussed how a DAX problem is detected; now we discuss how it is actually solved. Indeed, solving a DAX problem means finding all the (preferred) possible explanations which are consistent with the observations received so far.

To solve the problem we propose a variant of the *conflict-directed A** algorithm (cd-A*) discussed in [8]. In cd-A* there is a distinction between *decisional variables*, among which soft constraints are defined, and *non-decisional variables*, among which hard constraints are defined; the algorithm efficiently solves a CSP defined on both decisional and non-decisional variables in an optimized way. In fact, cd-A* is proven to find all and only the *optimal* solutions; namely, solutions that minimize the cost associated with the violation of the soft constraints. In our problem, non-decisional variables correspond to the real-valued start/end events of actions (a_s, a_e), with hard constraints encoded in the distance graph. Decisional variables, instead, model the behavior of the actions in A; more precisely, for each action a in A, we have a decisional variable a_{dec} whose domain $dom(a_{dec})$ is given by $\{m.label : m \in modes(a)\}$; that is, the decisional variable assumes values in the set of mode labels associated with a. We denote with $\mathcal{D}$ the set of all the decisional variables.

The idea is that the assignment of a non-nominal mode to a decisional variable corresponds to the violation of a soft constraint; the rank of such a mode represents the violation cost. Algorithm cd-A* can therefore be used to infer minimal rank diagnoses by finding assignments of modes to the decisional variables that minimize the global cost. The diagnoses we look for are therefore expressed in terms of the decisional variables in $\mathcal{D}$.

Search Space Due to lack of space we cannot describe the search process in detail, and we focus on the most important aspects only. First of all, we specify which pieces of information are maintained within a node of the search graph; each node contains:

- $\mathcal{H}$, a (partial) assignment of label modes to decisional variables; it is a set of pairs $\langle a_{dec}, label\rangle$ where $a_{dec} \in \mathcal{D}$ and $label \in dom(a_{dec})$ is the value assigned to a_{dec} in this node;
- $\mathcal{D}oms$ is a set of pairs of the form $\langle a_{dec}, dom(a_{dec})\rangle$ where $a_{dec} \in \mathcal{D}$ and $dom(a_{dec})$ is the domain of a_{dec} in the current node; note that as the search for a solution proceeds the domain of a decisional variable is progressively reduced to just one specific value;

- $\mathcal{G}_{\langle P,\mathcal{H},O\rangle}$ is the distance graph encoding the current hypothesis $\mathcal{H}$ and observations O; it is analogous to graph $\mathcal{G}_{\langle P,\Delta,O\rangle}$ described in section 4 for encoding a complete diagnosis Δ;
- $f_{node} = g(\mathcal{H}) + h(\mathcal{D}oms)$ is the heuristic evaluation of the node; more precisely, g is the sum of the ranks associated with the label assignments already made in $\mathcal{H}$; while h sums the minimal ranks that can still be assigned to the variables mentioned in $\mathcal{D}oms$; in this way, heuristic h is *admissible* and cd-A* is guaranteed to find optimal solutions. Of course, a decisional variable can either be mentioned in $\mathcal{H}$ or in $\mathcal{D}oms$, but not in both sets.

Search Strategy At the beginning of the search process, the $Open$ list (used to store the nodes on the frontier of the search graph in increasing value of f_{node}) is initialized with the node $\langle \emptyset, \mathcal{D}oms, \mathcal{G}_{\langle P,\emptyset,O\rangle}, 0\rangle$. Initially $\mathcal{H}$ is empty, and therefore $\mathcal{G}_{\langle P,\emptyset,O\rangle}$ contains constraints of the form:

$$a.L \leq a_e - a_s \leq a.U \tag{7}$$

for the duration of each action a, i.e. the *relaxed intervals*, according to which each mode is possible for a. At this stage, all the decisional variables are mentioned in $\mathcal{D}oms$; in particular, for each decisional variable a_{dec}, $dom(a_{dec})$ contains all the possible mode labels associated with a. Finally, f_{node} equals zero, in fact $g(\emptyset)$ is zero because no assignment has already been made, and $h(\mathcal{D}oms)$ is zero because the nominal mode (whose rank is zero) is possible in each decisional variable domain.

The search continues by extracting the top element from the $Open$ list and by expanding that node; this step is outlined in Algorithm 2 and discussed in detail afterwards; for the time being we just say that the expansion of a node returns the node itself, possibly modified, and tagged with a label which can either be *solution* or *partial*.

If the label is *solution*, the node contains a solution, that is, a complete assignment of values to the decisional variables in $\mathcal{D}$, which represents a minimal rank diagnosis consistent with O; in this case, the solution $\mathcal{H}$ is extracted from the current node and stored in the set of all minimal diagnoses *MinDiag*. If this is the first solution that has been found, the value of the node f_{node} is the rank of the solution *solutionRank*; any other minimal solution will have the same rank.

If the label is *partial*, the node does not contain a complete solution; in this case the search strategy iterates by picking up the new node at the top of the $Open$ list. The search terminates either when the list becomes empty, or when the value f_{node} of the top node in $Open$ is greater than $solutionRank$, that means that all the minimal rank diagnoses have been found.

At the end of the search process, the set *MinDiag* contains all the minimal rank diagnoses with rank *solutionRank*.

Node Expansion This step generates the children of the node extracted from the top of the $Open$ list (see Algorithm 2). Before that, however, it is necessary to minimize the node in order to be sure that it is the best one in $Open$. Thus, from line 1 to line 12, the current node is updated with minimization (line 2, see later). Note that as an effect of this step, the algorithm can discover a solution and hence terminate by returning the current node labeled as *solution* (line 5). Otherwise, the node is not a solution and it is heuristically evaluated (line 7); in case the new value f_{new} is worse than the previous estimated value f_{node}, the node is put again in the $Open$ list, and the expansion terminates returning the updated node and the label *partial*, meaning that the node does not contain a solution.

In case the current node is still the best node in the $Open$ list, it can be expanded. Among the decisional variables still to be assigned in $\mathcal{D}oms$, we select the one with the smallest domain size; this heuristic is known as *Minimum Remaining Values* (MRV) and aims at reducing the number of children to be generated (line 13). Having selected the decisional variable a_{dec}, the algorithm generates as many children as there are labels in its domain $dom(a_{dec})$. Each child is a new node generated by updating the information of the parent node; the new node is therefore enqueued in the $Open$ list (see lines from 14 thru 19). Finally, the algorithm terminates returning the current node with the status label *partial*.

Node Minimization Another important step of our solution is the minimization of a node, outlined in Algorithm 3. The purpose of this step is to reduce as much as possible the domains of the decisional variables still to be assigned by propagating the implications of the variables already assigned. First of all, Johnson algorithm is invoked so that the assignment made during the expansion of the parent node can be propagated into the distance graph of this node.

After that, it is possible to minimize the domains of all the decisional variables in $\mathcal{D}oms$. In particular, for each $a_{dec} \in \mathcal{D}oms$, function *restrictDomain* prunes out from $dom(a_{dec})$ the mode labels referring to intervals which are no longer possible in the updated distance graph. It is possible that after this pruning, the domain of variable a_{dec} contains just one mode label; thus a_{dec} must be moved from $\mathcal{D}oms$ to $\mathcal{H}$ (i.e., it becomes part of the current assignment). If at the end of the iteration over the decisional variables in $\mathcal{D}oms$, $\mathcal{D}oms$ becomes empty, it means that a complete assignment has been found and the algorithm terminates by returning the updated node tagged as *solution*; otherwise, the algorithm returns the node tagged as *partial*.

It is important to note that after the minimization of a node, each decisional variable still in $\mathcal{D}oms$ is associated with a set of mode labels that are all consistent with the distance graph kept in the node itself. In other words, given a decisional variable $a_{dec} \in \mathcal{D}oms$ there exists at least one consistent diagnosis for each mode label in $dom(a_{dec})$. As a consequence, whenever we create a new child node in Algorithm 2, we have the guarantee that the new node has a consistent distance graph; for this reason, we never have to check its consistency.

Example 3 *Let us consider a very simple DAX problem where the plan is the one in Figure 1 and the observation is* O=$\{\langle A2_e, 6\rangle;\langle B2_e, 6\rangle\}$. *From example 2, where we only considered* $\langle A2_e, 6\rangle$, *we know that the nominal hypothesis is inconsistent. After a few steps of cd-A*, the search node at the top of the* $Open$ *list is:*

$\langle \mathcal{H} : \{\langle A1, N\rangle; \langle B2, N\rangle\};$
$\mathcal{D}oms : \{(B1, \langle N, F1, F2\rangle); (A2, \langle N, F1, F2\rangle)\};\ \mathcal{G}_{\langle P,\mathcal{H},O\rangle}; 0\rangle;$

According to Algorithm 2 the node needs to be minimized before creating its children. The minimization (Algorithm 3) first invokes Johnson and then refines the domains of the variables in $\mathcal{D}oms$*; in particular, the observation on* $B2_e$ *prunes the nominal behavior from* $dom(B1)$*; this propagates through the join and since* $B1$ *cannot be nominal and* $A2_e$ *is observed at time 6, the domain of* $A2$ *is restricted to the nominal behavior only. Thus the node is updated by the minimization as follows:* $\langle \mathcal{H} : \{\langle A1, N\rangle; \langle B2, N\rangle; \langle A2, N\rangle\};$ $\mathcal{D}oms : \{(B1, \langle F1, F2\rangle)\};\ \mathcal{G}_{\langle P,\mathcal{H},O\rangle}; 1\rangle$. *Note that the heuristic evaluation of the node has now changed since action* $B1$ *can be qualified, in the best situation, with a rank 1 fault. The node is not a solution yet, so its two children (*$B1$ *is either* $F1$ *or* $F2$*) are created and enqueued into the* $Open$ *list. The process iterates and the new node on the top of* $Open$ *has a complete hypothesis* $\mathcal{H} : \{\langle A1, N\rangle; \langle B2, N\rangle; \langle A2, N\rangle; \langle B1, F1\rangle\}$, *which is consistent with the observations, and hence is a solution. In this particular case, this is the unique minimal rank diagnosis.*

Algorithm 2 *NodeExpansion*

Require: $node$: $\langle \mathcal{H}, \mathcal{D}oms, \mathcal{G}_{\langle P,\mathcal{H},O\rangle}, f_{node}\rangle$
Returns: $\langle node$, *solution*$\rangle$ when the current node is a solution; $\langle node$, *partial*$\rangle$ when current node is still a partial solution

if $node$ is not minimized **then**
  $node \leftarrow$ *NodeMinimize*($node$)
  mark $node$ as minimized
  if $\mathcal{D}oms$ is empty **then**
    return $\langle node$, *solution*$\rangle$
  end if
  $f_{new} \leftarrow g(\mathcal{H}) + h(\mathcal{D}oms)$
  if $f_{new} > f_{node}$ **then**
    $Open$.enque($node : \langle \mathcal{H}, \mathcal{D}oms, \mathcal{G}_{\langle P,\mathcal{H},O\rangle}, f_{new}\rangle$)
    return $\langle node$, *partial*$\rangle$
  end if
end if
$\langle a_{dec}, dom(a_{dec})\rangle \leftarrow MRV(\mathcal{D}oms)$
for each mode label $l \in dom(a_{dec})$ **do**
  $\mathcal{H}' \leftarrow \mathcal{H} \cup \{\langle a_{dec}, l\rangle\}$
  $\mathcal{D}oms' \leftarrow \mathcal{D}oms \setminus \{\langle a_{dec}, dom(a_{dec})\rangle\}$
  $\mathcal{G}'_{\langle P,\mathcal{H}',O\rangle} \leftarrow$ update($\mathcal{G}_{\langle P,\mathcal{H},O\rangle}$, $m\langle a, l\rangle.range$)
  $f'_{node} \leftarrow g(\mathcal{H}') + h(\mathcal{D}oms')$
  $Open$.enque($newNode : \langle \mathcal{H}', \mathcal{D}oms', \mathcal{G}'_{\langle P,\mathcal{H}',O\rangle}, f'_{node}\rangle$)
end for
return $\langle node$, *partial*$\rangle$

Algorithm 3 *NodeMinimize*

Require: $node$: $\langle \mathcal{H}, \mathcal{D}oms, \mathcal{G}_{\langle P,\mathcal{H},O\rangle}, f_{node}\rangle$
$\mathcal{G}_{\langle P,\mathcal{H},O\rangle} \leftarrow$ *Johnson*($\mathcal{G}_{\langle P,\mathcal{H},O\rangle}$)
for each decision variable $a_{dec} \in \mathcal{D}oms$ **do**
  $dom(a_{dec}) \leftarrow$ *restrictDomain*($\mathcal{G}_{\langle P,\mathcal{H},O\rangle}, dom(a_{dec})$)
  if $dom(a_{dec})$ has just one label l **then**
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{\langle a_{dec}, l\rangle\}$
    $\mathcal{D}oms \leftarrow \mathcal{D}oms \setminus \{\langle a_{dec}, dom(a_{dec})\rangle\}$
  end if
end for
return $node$

6 IMPLEMENTATION AND TEST

We implemented the algorithms proposed in section 5 as a Perl program; the C++ Boost Graph Library has been used for handling STNs. We conducted a first set of tests to check the feasibility of the approach; tests have run on a Intel i7 M640 processor at 2.80GHz with 8GB of RAM.

We defined two test sets, *2Ag* and *3Ag*, each containing 25 DAX problems with 2 and 3 agents, respectively. Each DAX problem consists of a TMAP and a sequence of timed observations. Note that TMAPs are not trivial plans: each of them contains 20 actions per agent and 3 joins involving up to 5 actions.

We perturbed the simulated execution of each TMAP with up to three randomly generated action delays. We assessed the approach by using a very scarce observability rate: only three timed observations per injected delay to infer *all* the minimal diagnoses.

	#sols	#nodes	time all sols	time/sol	time first sol
2Ag	9	655	39.3	4.4	14.9
3Ag	12.6	700	60.2	4.8	33.2

Table 1. Preliminary results. Columns show: avg # of solutions, avg # of visited nodes, avg time (sec), avg time per sol, and avg time for first sol.

Table 1 shows the main results of the experiments. We first note that the cases with 3 agents expand (on average) more nodes than the ones with 2 agents, and take more time to find all the minimal rank diagnoses. The average time for finding a solution is also (slightly) higher for 3 agents. Moreover, the average time for expanding each node (not reported in the table) is 0.06sec for 2 agents and 0.09sec for 3 agents.

We also note that for both test sets the computation of the first solution takes significantly more time than the average. This can be associated with the *slow start* behavior of cd-A*, which spends relatively more search time at the beginning of the process [8].

These initial results show that the time spent in doing temporal reasoning and propagation within each node is quite reasonable even without optimized STN-manipulation techniques. On the other hand, the results suggest that our version of cd-A* would benefit from a stronger pruning of the search tree to reduce the number of expanded nodes and consequently the computation time.

7 CONCLUSION

In this paper we have presented a novel approach to the diagnosis of actions delays in the execution of Multi-Agent Plans. We have modeled possible delays as duration intervals with an associated order-of-magnitude likelihood expressed by a rank, and formally defined diagnoses accordingly. To the best of our knowledge, the closest work in the literature is [7] where, however, the authors develop their approach specifically for uncertain observations and prefer diagnoses that maximize the coverage of such uncertainty intervals. In our framework we assume that observations are (reasonably) precise, and therefore we adopt a more classic criterion which prefers diagnoses that maximize the probability of action delays.

We have developed our proposal in the context of a monitoring process, where the system is in charge of detecting the inconsistency of the current hypothesis, perform diagnosis and continue the process with one or more updated hypotheses. In particular, we have focused on detection and diagnosis, leaving the study of the iterative updates of hypotheses to future work.

An implementation of the proposed algorithms has showed the feasibility of the approach; a possible improvement may be the study of a more sophisticated heuristic function in order to prune the cd-A* search tree more effectively.

REFERENCES

[1] J. S. Cox, E. H. Durfee, and T. Bartold, 'A distributed framework for solving the multiagent plan coordination problem', in *Proc. AAMAS05*, pp. 821–827, (2005).
[2] F. de Jonge, N. Roos, and C. Witteveen, 'Primary and secondary diagnosis of multi-agent plan execution', *Journal of Autonomous Agent and MAS*, **18**, 267–294, (2009).
[3] R. Dechter, I. Meiri, and J. Pearl, 'Temporal constraint networks', *Artificial Intelligence*, **49**, 61–95, (1991).
[4] M. Goldszmidt and J. Pearl, 'Rank-based systems: a simple approach to belief revision, belief update, and reasoning about evidence and actions', in *Proc. KR92*, pp. 661–672, (1992).
[5] M. Kalech and G. A. Kaminka, 'On the design of coordination diagnosis algorithms for teams of situated agents', *Artificial Intelligence*, **171**(8-9), 491–513, (2007).
[6] R. Micalizio and P. Torasso, 'Monitoring the execution of a multi-agent plan:dealing with partial observability', in *Proc. of ECAI'08*, pp. 408–412, (2008).
[7] N. Roos and C. Witteveen, 'Diagnosis of simple temporal networks', in *Proc. of ECAI'08*, pp. 593–597, (2008).
[8] M. Sachenbacher and B. C. Williams, 'Conflict-directed a* search for soft constraints', in *CPAIOR*, pp. 182–196, (2006).