An AO*-like algorithm implementation for active diagnosis

E. Chanthery, Y. Pencolé, N. Bussac

CNRS ; LAAS ; 7 avenue du colonel Roche, Toulouse, France
Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ;Toulouse, France
e-mail: [elodie.chanthery; yannick.pencole; nicolas.bussac]@laas.fr

Abstract

This paper details an algorithm that solves the active diagnosis problem. This algorithm relies on an AO*-search and computes a conditional plan to apply in order to refine the diagnosis. The input of the AO* is a set of ambiguous diagnosis candidates (represented as belief states) processed by the diagnoser at the time when the active diagnosis session starts. The search-space of AO* is defined by the behavioral model of the system that is common to the diagnoser and the diagnosis planner. Several problems have to be addressed. The first one is to adapt AO* algorithm to an optimization problem with various costs and rewards. The challenge is to find a relevant criterion that may be used by all kinds of autonomous systems (like satellites or rovers). The second challenge is to define an efficient heuristic in order to prune the search-space and to guide the search. This heuristic has to take into account the goal of active diagnosis and the mission goal of the autonomous system (mission model). Finally, the computational time required by this planner must be compatible with the time constraints inherent to autonomy in spatial systems.

1 Introduction

Autonomy is the ability to independently perform decisions and act in a changing environment. One of the most important characteristic of an autonomous system is its ability to take care of itself when performing its mission. In this work, we are particularly interested in increasing the autonomy of satellites (AGATA project, see Figure 1).[1]

Autonomous satellites are able to give relevant information to the ground, detect if one of their components is out of order and eventually decide to reboot it or to reconfigure itself and use another component. This ability requires *on-line diagnosis* (fault detection and isolation) and *on-line replanning*. On-line diagnosis is the problem of determining the faults that have occurred on the system, relying on the available observations. However, available observations may not be sufficient to determine the faults with certainty (ambiguous diagnoses). We thus propose to improve the performance of the diagnostic process by the use of the action capabilities of the system in order to refine the diagnosis in case of ambiguity: this is the *active diagnosis* problem. Active diagnosis problems are planning problems that aim at proposing an admissible sequence of actions (or *plan*) whose goal is to refine the diagnosis without radically changing or degrading the initial mission plan.

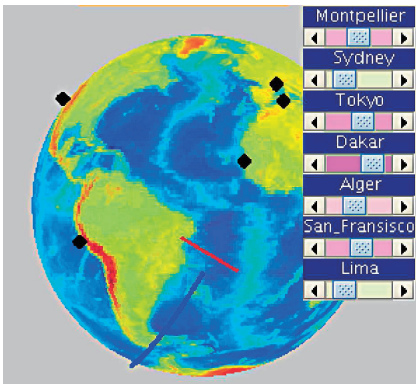

Figure 1. Observation Satellite (AGATA project)

In previous works [4], we have proposed a formal definition of the active diagnosis problem on discrete-event systems (DES) and an on-board architecture that integrates a diagnosis planner that is in charge of solving the active diagnosis problem (Figure 2). A first algorithm was sketched and we have raised the difficulties regarding the integration of active diagnosis into an on-board architecture including the mission planning process. The objective of this paper is to detail an algorithm that solves the active diagnosis problem. This algorithm relies on an AO*-search [3] and computes a conditional plan to apply in order to refine the diagnosis. As illustrated in Figure 2, the inputs of the AO* is not only a set of ambiguous diagnosis candidates (represented as belief states) processed by the diagnoser at the time when the active diagnosis session starts, but also objectives and constraints provided by the mission model and some constraints from the be-

[1]The AGATA project (Autonomy Generic Architecture - Tests and Application) is a French collaborative project funded by the French space agency (CNES) involving researchers and engineers from CNES, ONERA, and LAAS-CNRS.

haviour model of the system. The search-space of AO* is then constrained by the behavioral model of the system that is common to the diagnoser and the diagnosis planner. Several problems have to be addressed. The first one is to adapt AO* algorithm to an optimization problem with various costs and rewards. The challenge is to find a relevant criterion that may be used by all kinds of autonomous systems (like satellites or rovers). The second challenge is to define an efficient heuristic in order to prune the search-space and to guide the search. This heuristic has to take into account the goal of active diagnosis and the mission goal of the autonomous system (mission model). Finally, the computational time required by this planner must be compatible with the time constraints inherent to autonomy in spatial systems.

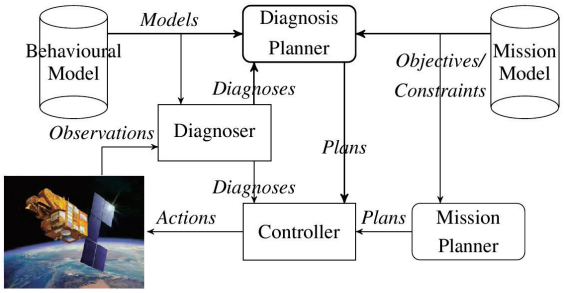

Figure 2. Integration of active diagnosis capabilities in an on-board architecture.

This paper is organized as follows. Section 2 describes related works on active diagnosis. Section 3 recalls how the diagnoser module works in the proposed architecture. Section 4 presents the AO*-like algorithm. Section 5 details the heuristics used in the AO*-like algorithm. Section 6 presents the spatial case study and our results. Section 7 concludes the paper.

2 Related Works

The choice of the set of actions performed on-line for refining the diagnosis may be compared with an on-line test sequencing problem. However, for dynamical systems that are currently performing a mission, the challenge of an active diagnosis process is to propose an admissible sequence of actions (or *plan*) that refines the diagnosis without radically changing the mission plan. The conflicts between the plan for diagnosis and the mission plan have to be taken into account.

To our best knowledge, there are very few works about active diagnosis in dynamical systems. The main contribution on active diagnosis of discrete-event systems is the work of [12]. Active diagnosis is formulated as a supervisory control problem [11] where the legal language is an "appropriate" regular sublanguage of the regular language of the system. The proposed solution is to design a system controller in such a way that it satisfies specified control objectives and results in a diagnosable controlled system. In other words, the action domain is restricted so that the system always remains diagnosable. This approach seems to be too restrictive for autonomous systems that realize a mission. Indeed, they need to keep all their action capability, even if they intermittently loose their diagnosis capability. In this article, we reuse the idea that consists in combining monitoring, diagnosis capability and controller, without restricting the controller's action capabilities.

In the field of hybrid systems, [2] propose to achieve active diagnosis in a hybrid framework. Starting in a non diagnosable region, the authors use the diagnosability analysis method to determine the sequence of controllable actions to be applied to the system in order to bring it into a diagnosable one. The problem of the search of an active diagnosis plan is formulated as follows: given an uncertain state of the active diagnoser, the active diagnosis problem is to find a controllable path leading to a certain state. The authors propose to solve this problem as a conditional planning problem using an AND-OR graph. The authors can take advantage of the fact that the discrete-event system resulting of the abstraction of an hybrid system has a limited a number of states. So the global exploration of the AND-OR tree is still possible. For systems completely described by a discrete-event system, the number of states is huge and the entire exploration of the AND-OR tree is intractable on-line.

The integration of active diagnosis into production plans is described in [7] and experimented on a model of an industrial digital printing press. The method is based on an active diagnoser that updates the current system state and stimulates the planner to generate informative plans that may determine faulty actions and achieve production goals as well. The advantage of interleaving diagnosis and planning is that diagnosis is quickly confirmed or invalidated. The objective is to determine faulty actions without explaining the cause of the failure. The repair procedure then consists in exchanging the printer modules according to their failure probability until the system is working properly. Our study focuses on autonomous systems that face hardware/software faults without any possibility of external repairing. The mission planner requires explicit diagnosis that determines the current degraded mode in order to optimally update the plan. Our approach consists in separating diagnosis and mission planning modules, because this is today the most common way to construct an on-board architecture for autonomous systems [1].

The problem of diagnosis refinement by action planning has also been raised by [8]. The author proposes a formal situation calculus framework for diagnostic problem

solving which incorporates a theory of action and change.

3 Background about Active Diagnosis on DES

Before going into the details of the AO*-like algorithm that solves the active diagnosis problem, it is necessary to recall how the diagnoser module works in the proposed on-board architecture (see Figure 2). As opposed to classical diagnosers like the one defined in [13] or more recently the ones defined in [10], the proposed diagnoser takes into account the fact that at a given time, it may be possible to act on the system and get a better diagnosis if such an action is performed. As shown in Figure 2, the purpose of the diagnoser is firstly to provide a diagnosis for any observable situation (like any other diagnoser) and secondly to provide information about how useful the trigger of an active diagnostic session could be.

3.1 Model extension for active diagnosis

The diagnoser proposed in this architecture relies on a discrete-event model of the system as proposed for instance in [13]: such a model represents the dynamical behaviour of the system reacting to actions from the controller and to the occurrence of fault events. The model is then represented as an automaton:

$$G = (X, \Sigma, T, x_0)$$

where X is the finite set of system states, Σ is the finite set of events, T is the transition function that represents the consequence of the occurrence of an event to the system's state and x_0 is the initial state of the system.

Basically, the general task of the diagnoser is to monitor the system (i.e. recording the available flow of *observed events*) and to determine whether some faults $f \in \Sigma_f \subseteq \Sigma$ have occurred or not. An observed event is defined as the output of an observation mask $Obs : \Sigma \rightarrow \Sigma_o \cup \{\epsilon\}$ that assigns to any event of Σ an event of $\Sigma_o \cup \{\epsilon\}$. For instance, a fully observable event $e \in \Sigma$ is such that $Obs(e) = e$ whereas a non-observable event e (like a fault for instance) is such that $Obs(e) = \epsilon$.

In order to deal with the active diagnosis problem, the diagnoser must also consider that some events $a \in \Sigma_a$ of Σ are actually *actions* performed by the controller. From the system model point of view, an *action* performed by the controller is represented as a controllable event [11].

In the following, we suppose that the controller notifies the diagnoser about the performed action which means that any action is actually fully observed by the diagnoser (see Figure 2):

$$\forall a \in \Sigma_a, (a \in \Sigma) \wedge (Obs(a) = a).$$

Finally, in order to ensure that the controller is always able to perform an action on the system, the study of the active diagnosis problem is restricted to the subclass of discrete-event systems in which the following hypothesis holds.

Hypothesis 1 *From any state $x \in X$, it is always possible to perform an action $a \in \Sigma_a$ after the occurrence of a finite sequence of reactive events $e \in \Sigma \setminus \Sigma_a$.*

In the case that this hypothesis does not hold, it means that the system can reach a state from which the controller cannot perform any more actions. In this case, the active diagnosis problem has trivially no solution as well as the mission control problem.

3.2 Characterisation of the diagnoser module

Here we recall how the diagnoser module is characterised. For a fully detailed description of the diagnoser, see [4]. The diagnoser module is a process Δ that continuously returns the set of possible *health states* (i.-e. a belief state in X) of the system that is consistent with the current sequence of observations $\sigma \in \Sigma_o^\star$. Like a classical diagnoser, the heath states returned by the diagnoser Δ also exhibit the status of the faults of Σ_f. The classical status of a fault are [10]:

1. *F-sure*: every event sequence *seq* of G from x_0 such that $Obs(seq) = \sigma$ contains the event F;

2. *F-safe*: no event sequence *seq* of G from x_0 such that $Obs(seq) = \sigma$ contains the event F;

3. *F-ambiguous*: there exists at least two sequences *seq* and *seq′* in G from x_0 such that $Obs(seq) = Obs(seq') = \sigma$ and only one of them contains the fault F.

In the following, $\Delta(\sigma)$ denotes the belief state corresponding to the diagnosis of the system with regards to the observations σ. The active diagnosis problem occurs only if the status of a fault F is ambiguous and thus requires complementary observations to discriminate the fault. To speed up the planning process (see next section), the diagnoser Δ that we propose actually returns two different types of status in case of ambiguity:

1. *F-discriminable*: the occurrence of F is ambiguous but there exists at least one observable sequence $\sigma.\sigma' \in \Sigma_o^\star$ of G such that the status of F in $\Delta(\sigma.\sigma')$ is either *F-sure* or *F-safe*;

2. *F-undiscriminable*: the occurrence of F is ambiguous and for any observable sequence $\sigma.\sigma' \in \Sigma_o^\star$ of G, the status of F in $\Delta(\sigma.\sigma')$ is always *F-ambiguous*.

The last two tags are defined in order to help the planner to trigger an active diagnosis session. In fact, if the current heath state is such that *F-undiscriminable* holds, then the planner already knows that it is useless to open an active diagnostic session to determine that F has effectively occurred. The planner may open the active diagnosis session only when *F-discriminable* holds.

To summarize, the diagnoser is a process that returns diagnosis candidates(as a belief states) for a given flow of observations and therefore it can be represented as finite state machine $\Delta = (X_\Delta, \Sigma_o, T_\Delta, x_{0\Delta},)$ like in [13] where X_Δ is the set of possible belief states. However, within the architecture, the diagnoser module can implement the FSM Δ with different techniques that depend on the nature of the modelled systems. The diagnoser module can, for instance use, distributed techniques [6], specialised techniques [10] or symbolic techniques [14, 5].

4 An AO*-like algorithm for Active Diagnosis

The input of the AO*-like algorithm we proposed is the ambiguous state on which the active diagnosis session has been triggered. On the fly, the algorithm receives the next possible states and the next possible events from the diagnoser structure. It implicitly constructs an AND-OR graph on which it applies an ordered depth-first search strategy. The output of the algorithm is a conditional plan. This section describes how is constructed the underlying AND-OR graph during the search, then the core of the AO*-like algorithm, the output of the algorithm and the criteria taken into account.

4.1 The implicit AND-OR graph

When the controller performs an action, it is assumed that the system reacts and that the diagnoser receives some other observation events resulting from this action in finite time.

Hypothesis 2 *In the diagnoser, each action is followed by an observable event that is not an action.*

The structure constructed thanks to the diagnoser is an AND-OR graph represented by a graph G = (V,E) where V is the set of vertices and E is the set of edges. The set V contains three types of nodes, "OR", "AND" and "LEAF" nodes that are described in Figure 4 by circles, rectangles and triangles respectively. These nodes have attributes including an event (an action for AND node, an observable event or a sequence of observable events that not included action for OR node) and a tag that is derived from the diagnoser.

The construction of the AND-OR graph is done incrementally as follows:

1. v_i is initial vertice of the graph. It corresponds to the state for which it is useful to trigger an active diagnosis session and for which it exists one fault F tagged *F-discriminable*. This vertice is an OR node.

2. Consider all the actions that can be applied considering the behavioural model of the system. Each applicable action is an AND node. If there is no applicable action this OR node is retagged *F-undiscriminable*.

3. Following the transition function of Δ, we found all the minimal sequences of observable events that represent a belief state for the system. These belief states are tagged by the tag of the state of Δ after each sequence. Every belief states tagged *F-undiscriminable*, *F-sure* or *F-safe* are LEAF nodes. Every belief states tagged *F-discriminable* are OR nodes.

4. For each LEAF node, stop. For each OR node, consider all the actions that can be applied considering the behavioural model of the system. Each applicable action is an AND node. Each OR node for which there is no applicable action is retagged *F-undiscriminable*.

5. Iterate step (3) and (4) for each AND node.

Figure 3 illustrates what could be a part of the diagnoser. Figure 4 shows the AND-OR graph that will be constructed.

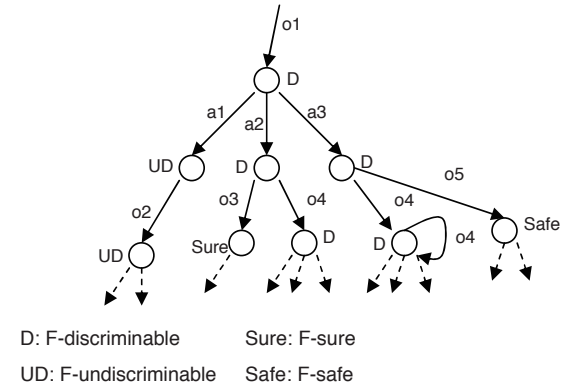

Figure 3. A part of the diagnoser

4.2 The AO*-like algorithm

One contribution of this paper is to adapt AO* algorithm to an optimization problem with various costs and rewards. The challenge is to find a relevant criterion that may be used by all kinds of autonomous systems (like satellites or rovers). The pseudo-code of the AO*-like algorithm we developed is given in Figure 5.

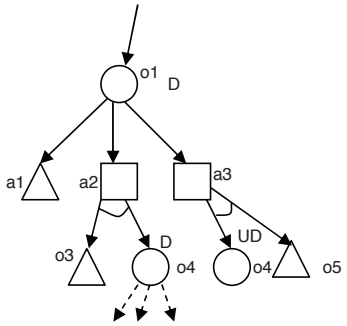

Figure 4. The AND-OR graph that could be deduced from the diagnoser

The RootNode (line 1) is an OR node. The list ORnodeToBedeveloped (line 2) keeps in memory all the OR nodes that have to be developed for constructing the entire conditional plan. FirstNode (line 3) removes the first node of the list and returns it. ComputesHeuristic (line 9) computes the heuristic of an AND node. The heuristic is explained in the next section. BestSuccessor (line 8) returns the AND node that has the minimum heuristic. CreateSuccessorNodes (line 11) creates the list of the successor nodes. If a node already exists, it is not created. CycleSearch (line 13) is a function that detects a cycle i-e that checks if some nodes of NextOrNodes have already been developed in the past. If it is the case, the CycleSearch function removes the incriminate nodes from NextOrNodes.

4.3 Output of the algorithm

The output of the AO*-like algorithm is a conditional plan. It starts from the RootNode, says what is the best action to apply, and for all possible observations, indicates what is the best action to preform, and so on. Figure 6 illustrates a conditional plan. The conditional plan is then an AND-OR tree where there is only one AND-node after an OR-node. Leaves are nodes that are tagged *F-sure*, *F-safe*, or *F-undiscriminable*, or also nodes that have been detected as redundant in the CycleSearch function. Terminal OR-nodes tagged *F-undiscriminable* correspond to belief states where no action is applicable because of the behavioural model.

On-line when the conditional plan is applied, only one branch of the tree is really explored, depending of the observable events. Thus it is clear that a conditional plan do not ensure to guide the system toward a totally discriminated belief state.

4.4 Criterion

The system has to apply the "best" conditional plan in order to refine the diagnosis. This supposes that there exists a criterion that indicates the best conditional plan. In the active diagnosis for autonomous systems, a conditional plan should particularly:

- minimize costs for discriminating faults;
- allow to disambiguate faults that are the most critical i-e faults that could cost a lot if they are sure and not taken into account;
- maximize rewards on objectives.

Let T be the tree plan. Let $\mathbb{P}$ the set of paths p such that the first node is the initial node of the tree and the last node is a leaf of the tree. One defines three terms for calculating the criterion. The first one is the cost of a path p. It is the sum of all the actions that are chosen in p. It is denoted $Cost(p)$. The second one focuses on the faults that are disambiguated along the path. Let FD be the set of faults such F was tagged *F-discriminable* in the first node of p and that get tagged *F-sure* for the last node of the path. Let O_{na} be the set of objectives

```
1  Initialisation;
2  Build RootNode ;
3  ORnodeToBeDeveloped ← RootNode;
4  CurrentNode =
   FirstNode(ORnodeToBeDeveloped);
5  while ORnodeToBeDeveloped is not empty do
6      while (CurrentNode.status = unsolved) and
       (CurrentNode.status = not unsolvable) do
7          NextAndNodes ←
           CreateSuccessorNodes(CurrentNode) ;
8          forall ANDnode in NextAndNodes do
9              ANDnode.tag←
               GetTag(ANDnode.Successor) ;
10             ANDnode.Heuristic =
               ComputesHeuristic(ANDnode);
11         ANDnodeToBeDeveloped =
           BestSuccessor(NextAndNodes);
12         NextOrNodes = CreateSucce-
           sorsNodes(ANDnodeToBeDeveloped);
13         Order NextOrNodes in best first criterion
           order;
14         Cycle search;
15         Put NextOrNodes on the front of
           ORnodeToBeDeveloped;
16         CurrentNode =
           FirstNode(ORnodeToBeDeveloped) ;
17     CurrentNode =
       FirstNode(ORnodeToBeDeveloped) ;
18 Reduce plan;
19 Edit conditional plan solution ;
```

Figure 5. AO*-like pseudo-code

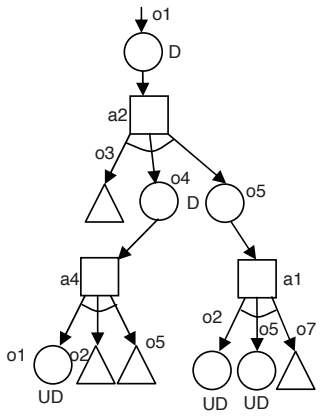

Figure 6. Illustration of a conditional plan

that are not achievable if all F in FD are finally tagged *F-sure* and R_o the reward associated to the objective o. The $AvoidLossOfIncome(p)$ is defined as the sum of

$$AvoidLossOfIncome(p) = \sum_{o \in O_{na}} R_o \quad (1)$$

The rewards accumulated on the objectives are embodied by the third term of the criterion. This term represents the sum of the rewards for all the objectives that have been achieved along the path. It is denoted $Reward(p)$.

Finally, in order to calculate the criterion of a conditional plan, it is necessary to evaluate all the branches of the tree. For example, in Figure 6, there are 7 possible branches to evaluate. There are three possibilities to evaluate the criterion. The first one is to evaluate the worst branch of the tree:

$$C_{MAX} = \max_{p \in \mathbb{P}} \{Cost(p) + AvoidLossOfIncome(p) - Reward(p)\} \quad (2)$$

Another solution is to evaluate the best case and take the best branch of the tree:

$$C_{MIN} = \min_{p \in \mathbb{P}} \{Cost(p) + AvoidLossOfIncome(p) - Reward(p)\} \quad (3)$$

The last solution consists in taking an average of the criterion:

$$C_{AVE} = \frac{1}{|\mathbb{P}|} . \sum_{p \in \mathbb{P}} \{Cost(p) + AvoidLossOfIncome(p) - Reward(p)\} \quad (4)$$

To conclude, a performance criterion has been defined in order to evaluate the solutions that are proposed by the algorithm.

5 Heuristic

5.1 Existing heuristics for AO*

The heuristics used in AO* are mainly those of Yeung [15], Patippati [9]. Yeung's heuristic uses the notion of cost-entropy function that is an information-theoretic lower bound of the expected cost of an optimal testing tree. It takes into account the number of possible responses for each test and the cost of the test. In our approach, it is impossible to apply this solution because a test is replaced by an action on the dynamic system: the number of possible responses for each action is unknown and depends on the current state of the state when the action is applied. Patippati's heuristic is used for test-sequencing problems. It supposes that there are known probabilities on the faults, and works with exclusive binary tests. Our context is different and even if some works extend this heuristic for multivalued tests and dynamic costs, it remains hard to apply in our case.

5.2 Guidance Heuristic on AO* for active Diagnosis

It is well-known that the task of finding an optimal testing tree is an NP-complete problem. Therefore the use of heuristics to search for an optimal tree is inevitable when the problem is large. As defined in [15], the heuristic search process relies on a heuristic evaluation function (HEF) as a guide to avoid evaluating all the possible testing trees before finding the optimal one. An HEF is an easily computable heuristic estimate (usually in a form of a lower or an upper bound) of the optimal cost-to-go at a non terminal node of a testing tree.

Another contribution of this work is to propose an original heuristic adapted with active diagnosis. The objective of active diagnosis is to find the sequence of actions that:

- minimizes costs for discriminating faults;
- allows to disambiguate faults that are the most critical i-e faults that could cost a lot if they are sure and not taken into account;
- maximizes rewards on objectives;
- penalizes undiscriminable nodes.

As it is crucial to take into account the impact of the the active diagnosis session on the all mission, we choose to evaluate the criterion until the end of the mission. In the AND-OR graph exploration, the decision is on the AND-node: only one AND-node is chosen at each step whereas all the OR nodes that follow the chosen AND-nodes are developed. Then the heuristic has to give information for guiding the choice of the AND-node to develop.

Intuitively, the goal is to minimize a criterion C that is defined as: C = *Costs to discriminate faults + loss of*

income of "sure" faults - sum of rewards on the past . In other words, one prefers discriminating a fault that could cost a lot in the future if it appears to be *F-sure* instead of a fault that could give a lot of rewards if it appears to be *F-safe*. That means that the most important is the criticity of a fault. One recalls that the heuristic has to give a maximum bound of C.

5.2.1 AND node heuristic

The heuristic H_{AND} of an AND node n is equal to the action cost associated to n plus the value of the heuristic of all the OR nodes that follow n.

$$H_{AND}(n) = C_n + \sum_{n_s \in Successors(n)} H_{OR}(n_s) \quad (5)$$

where C_n is the action cost associated to n; $Successors(n)$ is the set of the successors of n; H_{OR} is the heuristic for an OR node.

5.2.2 OR node heuristic

Let n_s be the OR node for which we want to calculate the heuristic value. One distinguishes 3 different heuristics depending of the tag of each fault for n_s.

Heuristic for undiscriminable faults: belief states with a lot of faults tagged *F-undiscriminable* have to be avoided in the plan. Thus undiscriminable faults are penalized. Let *Penality* be a number 100 times greater that the order of magnitude of the costs and reward used in the behaviour model of the system, and FuD the set of faults tagged *F-undiscriminable* in n_s.

$$H_1(n_s) = \sum_{F \in FuD} Penality \quad (6)$$

Heuristic for repairable faults: the simplest action to discriminate the fault is to repair it. A fault tagged *F-discriminable* becomes *F-safe* in any cases. As the fault is repaired, there is no consequence on the future rewards. Let $C_{repair}(F)$ be the cost to repair the fault F and FD_r the set of faults tagged *F-discriminable* in n_s and that are reparable.

$$H_2(n_s) = \sum_{F \in FD_r} C_{repair}(F) \quad (7)$$

Heuristic for non reparable faults: for this type of faults, the number of steps to a goal node has to be evaluated. When the diagnoser is developed, an evaluation of this distance is given. Let n_{eMAX} be the maximum number of steps for discriminating a node tagged *F-discriminable* for the set of not repairable faults. Let C_{aMAX} be the maximum cost of an action. Let FD_{nr} be the set of faults that are tagged *F-discriminable* and that are not repairable, and $|FD_{nr}|$ its cardinal. So the maximum cost to disambiguate a state is:

$$C_{MAX} = n_{eMAX}.C_{aMAX}.|FD_{nr}| \quad (8)$$

For a node n_s with some faults tagged *F-discriminable*, the loss of income is defined as follows. Let O_{na} be the set of objectives will not be achievable if all F in FD_{nr} are finally tagged *F-sure* and R_o the reward associated to the objective o. The loss of income is:

$$LossOfIncome = \sum_{o \in O_{na}} R_o \quad (9)$$

So, for non repairable faults, we define the heuristic H_3 as:

$$H_3(n_s) = n_{eMAX}.C_{aMAX}.|FD_n| + \sum_{o \in O_{na}} R_o \quad (10)$$

The heuristic $H_{OR}(n)$ for an OR node n_s is the following:

$$H_{OR}(n_s) = H_1(n_s) + H_2(n_s) + H_3(n_s) \quad (11)$$

6 Spatial case study and results

The AO*-like algorithm has been implemented in C++ and tested on a module (a hotspotter) of a fictive satellite developed in the AGATA project (see Figure 1). This module has four possible actions: *on*, *reset*, *close* and *off* and 11 types of observable events are available on this module. 13 types of fault may occur in this module. The behavioural DES model associated to the module has 13538 states and 39366 transitions. Table 1 presents minimal/average/maximal computation time to solve different active diagnosis problems. Each column represents 1000 runs for one particular fault F. To initiate the run, a generator of random scenarios has been used. A first result is that the number of interesting scenarios (IS) , i-e generated scenarios such that the diagnosis returned by the diagnoser is ambiguous, is limited. It is indicated in the first line of Table 1 in percent. Another result is that each interesting run stops in less than 407 ms which is satisfactory thanks to the heuristics. The average time for getting a conditional plan is about 32 ms.

Hotspotter Fault Type	F_1	F_2	F_3
Nb of IS (%)	100	3,8	6,1
Min (ms)	< 1	31	16
Max (ms)	94	109	407
Average (ms)	19,6	53,8	50,3
Hotspotter Fault Type	F_4	F_5	F_6
Nb of IS (%)	2,6	4,6	1,3
Min (ms)	< 1	< 1	31
Max (ms)	16	31	94
Average (ms)	6,9	11,7	50,3

Table 1. Results: AO* computation time

7 Conclusion and future works

To conclude, this paper shows the feasibility of active diagnosis for space systems like satellites. An adapted AO*-like algorithm has been implemented with dedicated heuristics. In the future, it would be interesting to develop some variant of this algorithm. One interesting case is to develop the tree plan with a limited depth so as to be faster online and more efficient. After the application of this partial plan, it will be decided to replan with uncertainties or to recompute a partial plan for diagnosis with the AO*-like algorithm. Thanks to that, we will avoid the danger of a too large diagnoser and the development of some branches of the tree for nothing. Furthermore, we are interested in studying how active diagnosis, planning and diagnosis may be integrated in a distributed architecture, when decision is distributed between the ground and the satellite for example, or between several satellites.

References

[1] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An architecture for autonomy. *Intern. Journal of Robotics Research*, 17:315–337, 1998.

[2] M. Bayoudh, L. Travé-Massuyès, and X. Olive. Towards active diagnosis of hybrid systems. In *Proc of DX'08*, 2008.

[3] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Planning with incomplete information as heuristic search in belief space*, pages 52–61, 2000.

[4] E. Chanthery and Y. Pencolé. Monitoring and active diagnosis for discrete-event systems. In *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, 2009.

[5] A. Grastien, Anbulagan, J. Rintanen, and Elena Kelareva. Diagnosis of discrete-event systems using satisfiability algorithms. In R. Holte, editor, *Nineteenth National Conference on Artificial Intelligence (AAAI-07)*. AAAI Press, 2007.

[6] P. Kan John and A. Grastien. Local consistency and junction tree for diagnosis of discrete-event systems. In *European Conference on Artificial Intelligence (ECAI-08)*, 2008.

[7] L. Kuhn, B. Price, J. de Kleer, M. B. Do, and R. Zhou. Pervasive diagnosis: The integration of diagnostic goals into production plans. In *AAAI 2008*, pages 1306–1312, 2008.

[8] S. McIlraith. Incorporating action into diagnostic problem solving (an abridged report). In *Working Notes of the 1995 AAAI Spring Symposium on Extending Theories of Action: Formal Theory and Practical Applications*, pages 139–144, 1995.

[9] K. R Pattipati and M. Dontamsetty. On a generalized test sequencing problem. *IEEE transactions on systems, man, and cybernetics*, 22(2):392–396, 1992.

[10] Y. Pencolé, D. Kamenetsky, and A. Schumann. Towards low-cost diagnosis of component-based systems. In *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Process (SAFEPROCESS)*, 2006.

[11] P.J.G. Ramadge and W.M. Wonham. The control of discrete event processes. In *IEEE Proc.: Special issue on Discrete Event Systems*, 1989.

[12] M. Sampath, S. Lafortune, and D. Teneketzis. Active diagnosis of discrete-event systems. *IEEE Transactions on Automatic Control*, 43, 1998.

[13] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Failure diagnosis using discrete event models. *IEEE Transactions on Control Systems Technology*, 4(2):105–124, March 1996.

[14] A. Schumann, Y. Pencolé, and S. Thiébaux. A spectrum of symbolic on-line diagnosis approaches. In *Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, Vancouver, Canada, July 2007.

[15] R.W Yeung. On noiseless diagnosis. *IEEE Transactions on systems, man, and cybernetics*, 24:1074–1082, 1994.