Variantes de l'AO* pour générer automatiquement des arbres de diagnostic presque optimaux

AO* variants to generate automatically near optimal diagnosis trees

Xavier Olive[1,2] Louise Travé-Massuyès[1] Hervé Poulard[2]

[1] LAAS-CNRS
[2] Actia

7, Avenue du colonel Roche - 31077 Toulouse cedex
xolive@laas.fr

Résumé

Cet article propose 2 méthodes qui permettent de générer des arbres de diagnostic presque optimaux de manière performante en terme de temps de calcul et en limitant la perte d'optimalité. Ce travail est basé sur une approche à base de modèles (AGENDA) pour la génération d'arbres de diagnostic optimaux en utilisant un algorithme AO. Les entrées de cet algorithme sont les fautes qui peuvent survenir sur le système avec leur probabilité d'occurrence a priori, les tests disponibles et une matrice de signatures des fautes qui associe à chaque couple (faute / test) un ensemble de modalités qui sont les résultats des tests lorsque une faute donnée est présente. Le principal inconvénient d'AGENDA réside dans un temps de calcul trop important pour obtenir l'arbre de diagnostic pour des systèmes de grande taille.*

Mots Clef

algorithme AO*, heuristique, méthode presque optimale, arbre de diagnostic.

Abstract

This paper presents two methods which allow us to generate near optimal diagnosis trees in an efficient way while reducing the optimality loss. These methods use a model-based Automatic GENeration of DiAgnosis trees method (AGENDA) using the AO algorithm. The inputs of this algorithm are the faults which may occur on the system to diagnose with their respective occurrence probability, the tests that can be performed and the crosstable which assigns to each (fault/test) pair the set of modalities which are expected as outcome of the test when the fault occurs. The main drawback of AGENDA is the required high computation time to obtain an optimal diagnosis tree for large complex systems.*

Keywords

AO* algorithm, heuristic, near-optimal method, diagnosis tree.

1 Introduction

Dans le domaine de l'automobile, l'utilisation des systèmes électroniques afin de contrôler plusieurs fonctions (injection, ABS) a été largement développée pendant ces dernières années. Ces systèmes électroniques se composent d'alimentations électriques, de capteurs et d'actionneurs reliés à un calculateur par un faisceau de fils ou un bus (Figure 1).

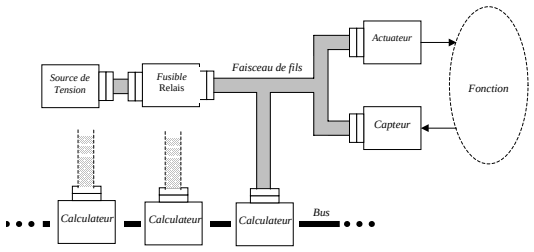

Figure 1: Système électronique global

Afin de diagnostiquer de tels circuits électroniques, des arbres de diagnostic sont construits. Ces arbres permettent aux garagistes de trouver les composants défectueux en réalisant une séquence de tests. Afin de construire automatiquement ces arbres de diagnostic à partir des données de conception fournies par les constructeurs automobiles, AGENDA (génération automatique d'arbres de diagnostic) [5], une méthode raisonnant sur les intervalles a été développée. AGENDA est une méthode abductive non interactive qui utilise un algorithme de prédiction, qui, à partir d'ensembles de pannes et de tests anticipés, permet de construire une matrice de signature

des fautes. L'algorithme AO* est utilisé pour obtenir un arbre de diagnostic optimal en terme de coût de mesure.
L'algorithme AO* est un algorithme de recherche heuristique dont la durée d'exécution est difficilement maîtrisable. C'est pourquoi on trouve dans la littérature différentes méthodes permettant la génération des arbres de diagnostic presque optimaux. Différents sous-ensembles de tests de cardinalité réduite sont proposés par Faure [4]. La notion d'*ϵ-admissibilité* qui permet de garantir un maximum de l'écart à la valeur optimale en pourcentage et une approche basée sur un algorithme utilisant une heuristique d'information à plusieurs pas en avant (*Multistep information Heuristic*) sont présentées dans [11]. Des stratégies nommées "Rollout" (Définition d'une politique de recherche dans l'arbre) sont décrites dans [12] pour prendre en compte des systèmes de grande taille. D'autres méthodes comme MAO* (Memory AO*) et WAO* (Weighted AO*) sont proposées dans [2] et [3], respectivement. Cet article propose deux nouvelles méthodes dans le but de réduire les temps de calcul liés à la génération d'arbres de diagnostic.
L'article est décomposé comme suit. La section 2 définit le Test Sequencing Problem. La section 3 détaille les solutions pour résoudre ce problème au moyen de l'algorithme AO*. La section 4 présente les méthodes utilisées dans AGENDA pour trouver des solutions presque optimales ; les 2 nouvelles méthodes sont proposées dans les sections 5 et 6. La section 7 compare les performances de ces méthodes par rapport à celle donnant la solution optimale sur 3 exemples réels de notre domaine d'application.

2 Test Sequencing Problem

Le Test Sequencing Problem (TSP) est défini comme suit :

- Un ensemble F de n_F fautes f_i avec $i \in \{1, ..., n_F\}$.
- Un ensemble p de n_F probabilités d'occurrence a priori p_i avec $i \in \{1, ..., n_F\}$ tel que $\sum_{i=1}^{n_F} p_i = 1$.
- Un ensemble S de n_S tests s_j avec $j \in \{1, ..., n_S\}$. Les tests sont supposés multi-valués : chaque test s_j possède au moins 2 résultats, appelés modalités. Le nombre de modalités du test s_j est noté n_M^j. Les tests sont aussi supposés asymétriques : en présence d'une faute donnée f_i, un test s_j peut avoir plusieurs modalités possibles (i.e. la faute f_i apparaît dans plusieurs feuilles sous le test s_j).
- Un ensemble C de n_S coûts de test $c_j \geq 0$ avec $j \in \{1, ..., n_S\}$ qui représente le coût de réalisation de la mesure du test s_j en terme de temps, ressources (humaine et matériel), accessibilité du point de mesure ...
- Une matrice de signature de fautes $A = [a_{ij}]$ de dimension $n_F \times n_S$ où a_{ij} représente les modalités du test s_j en présence de la faute f_i.

Le problème consiste alors à construire une séquence de tests (i.e. un arbre de diagnostic) qui est capable d'isoler sans ambiguïté chacune des fautes présentes dans F en utilisant les tests de S, et ceci en minimisant le coût de l'arbre de diagnostic J donné par l'équation (1) où $D = [d_{ij}]$ est une matrice binaire de dimension $n_F \times n_S$ telle que $d_{ij} = 1$ si le test s_j est utilisé dans la séquence de tests menant à l'identification de la faute f_i et $d_{ij} = 0$ sinon.

$$J = \sum_{i=1}^{n_F} \left(p_i \times \left(\sum_{j=1}^{n_S} d_{ij} \times c_j \right) \right) \quad (1)$$

Il est communément établi que construire un arbre de diagnostic optimal est un problème NP-complet [6] [7] [8].

3 Solution au TSP

3.1 Arbre de diagnostic

Un arbre de diagnostic peut être vu comme une succession alternée de nœuds OU et de nœuds ET [9] lors de sa construction. Un nœud OU représente l'ensemble courant des fautes (ensemble d'ambiguïté). Un nœud ET représente un test et a autant de nœuds OU fils que le test a de modalités. Dans l'arbre de recherche ET/OU, l'ensemble courant des tests est considéré sous chaque nœud OU (le nœud OU représente alors un choix). Par contre, un nœud ET a comme fils les différentes valeurs d'un test, qui seront toutes conservées si le nœud ET est choisi.

A partir de l'arbre de recherche ET/OU, un arbre de diagnostic est un sous-arbre qui représente les séquences de tests permettant de discriminer l'ensemble des fautes : à chaque nœud OU, un des fils (i.e. un test) est choisi, tandis que chaque nœud ET conserve l'ensemble de ses fils.

La figure 2 montre un exemple d'arbre de diagnostic contenant 4 fautes $\{f_1, ..., f_4\}$ et utilisant 5 tests asymétriques multi-valués $\{s_1, ..., s_5\}$.

Nous adoptons la notation suivante pour la suite de l'article. Soit T un arbre de diagnostic qui discrimine l'ensemble des fautes F en utilisant l'ensemble des tests S avec la matrice de signature des fautes A. Soit n_L le nombre de feuilles $\{l_1, ..., l_{n_L}\}$ et $P(l_i)$ la probabilité d'occurrence de chaque feuille l_i telle que $\sum_{i=1}^{n_L} P(l_i) = 1$. Soit d_{ij} une variable booléenne égale

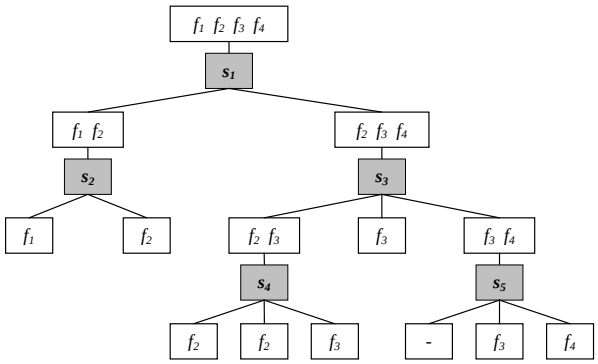

Figure 2: Arbre de diagnostic

à 1 si le test s_j appartient au chemin de la racine à la feuille l_i et 0 sinon.
La fonction objectif K, définie par l'équation (2), est utilisée pour évaluer les différents arbres de diagnostic T.

$$K(T) = \sum_{i=1}^{n_L} \left(P(l_i) \times \left(\sum_{j=1}^{n_S} d_{ij} \times c_j \right) \right) \qquad (2)$$

Sous hypothèse de coût des tests unitaires, cette fonction objectif $K(T)$ est équivalente à la profondeur moyenne de l'arbre T.

3.2 Algorithme AO*

L'algorithme AO* est basé sur un arbre de recherche ET/OU. Un arbre de recherche ET/OU contient des nœuds ET qui ont la propriété d'être vrai si tous les fils sont vrais et des nœuds OU qui ont la propriété d'être vrai si au moins un de leurs fils est vrai.
L'arbre de recherche ET/OU implicite représente toutes les solutions possibles d'un problème partant des éléments de base du problème. Dans le problème de l'arbre de diagnostic optimal, les éléments de base sont l'ensemble des fautes F, l'ensemble des tests S et la matrice de signatures des fautes A. Les solutions possibles sont tous les arbres de diagnostic T qui permettent de discriminer l'ensemble des fautes F en utilisant un sous-ensemble des tests de S.
En raison d'une complexité généralement trop élévée, l'arbre de recherche ET/OU implicite est rarement entièrement exprimé. L'idée de l'algorithme AO* est de développer seulement une partie de l'arbre de recherche implicite, correspondant à la solution la plus intéressante du problème, en accord avec la fonction objectif à optimiser. Ce sous-arbre de l'arbre de recherche ET/OU implicite est sélectionné selon l'évaluation heuristique la plus pertinente et est appelé l'arbre explicite. Plus l'évaluation heuristique est pertinente, moins l'arbre explicite s'accroît et seule la solution la plus intéressante est conservée. La solution optimale du problème peut être trouvée à l'intérieur de l'arbre de recherche ET/OU explicite. Par conséquent, l'arbre de diagnostic optimal T^* est un sous-arbre de l'arbre de recherche ET/OU explicite développé par l'algorithme AO*.
Etant donné un nœud OU N^O de l'arbre explicite, $H(N^O)$ représente la valeur de l'heuristique de $K(T^*_{N^O})$ où $T^*_{N^O}$ est le sous-arbre optimal ayant N^O comme racine alors que $F(N^O)$ représente la valeur estimée courante de $K(T^*_{N^O})$. Si N^O est une feuille de l'arbre explicite alors $F(N^O) = H(N^O)$.

1. *Initialisation*
   Au début de l'algorithme AO*, l'arbre de recherche ET/OU explicite est seulement composé de sa racine, appelé N_r, qui est un nœud OU représentant l'ensemble complet F des fautes. L'arbre de diagnostic optimal courant marqué T^* est seulement composé de ce nœud OU. L^* est l'ensemble des nœuds OU feuille de l'arbre courant développable T^*,i.e. les feuilles qui ne correspondent pas à des fautes isolées (i.e. les feuilles réelles de l'arbre de diagnostic définitif).

   La valeur $F(N_r)$ est initialisée avec la valeur heuristique $H(N_r)$ calculée pour ce nœud OU. La probabilité d'un nœud OU est calculée comme la somme des probabilités d'occurrence a priori des fautes appartenant à ce nœud. Pour le nœud OU N_r, la probabilité $P(N_r)$ est la somme des probabilités d'occurrence de toutes les fautes à discriminer, soit 1.

2. *Traitement itératif*
   A chaque étape de l'algorithme AO*, un des nœuds OU feuille de l'arbre courant sélectionné T^* de diagnostic, appelé N^O_c, est développé. Tous ses n_A nœuds ET fils, appelé $N^A_{c,j}$ avec $j \in \{1, ..., n_A\}$, correspondant à tous les tests disponibles (i.e. pas encore utilisés) sont créés. Chacun de ces nœuds ET $N^A_{c,j}$ est développé en créant leurs $n^{c,j}_O$ nœuds OU fils, appelés $N^O_{c,j,k}$ avec $k \in \{1, ..., n^{c,j}_O\}$, correspondant aux $n^{c,j}_O$ différentes modalités du test s_j.

   Pour chaque nœud OU feuille créé $N^O_{c,j,k}$, $F(N^O_{c,j,k})$ est calculé comme $H(N^O_{c,j,k})$ (évaluation heuristique du nœud $N^O_{c,j,k}$). Ensuite, pour chaque nœud ET feuille créé $N^A_{c,j}$, $F(N^A_{c,j})$ est calculé comme $c_j + \sum_{k=1}^{n^{c,j}_O} F(N^O_{c,j,k})$.

   Enfin, un traitement récursif est appliqué sur les nœuds successifs rencontrés sur le chemin à partir du nœud OU courant N_c jusqu'à la racine N_r. Ce traitement consiste à mettre à jour successivement les valeurs F de ces nœuds et le marquage de sélection des nœuds ET qui constituent l'arbre de diagnostic optimal courant T^*.

Si le nœud courant à mettre à jour N_c^O est un nœud OU, le marquage de sélection de ses nœuds ET fils, si il existe, est supprimé. $F(N_c^O)$ est alors calculé comme $\min_{j=1}^{N_{c,j}^A} F(N_{c,j}^A)$ et le nœud ET fils pour lequel ce minimum est atteint est marqué.

Si le nœud courant à mettre à jour $N_{c,j}^A$ est un nœud ET, $F(N_{c,j}^A)$ est alors calculé comme $c_j + \sum_{k=1}^{N_{c,k}^O} P(N_{c,k}^O) \times F(N_{c,k}^O)$.

Tant que le nœud courant à mettre à jour N_c est différent de la racine N_r, le nœud courant à mettre à jour devient le père de N_c.

3. *Condition d'arrêt*
   L'algorithme AO* s'arrête quand l'ensemble L^* des nœuds OU feuilles développables de l'arbre de diagnostic optimal courant T^* est vide.

   Alors, l'arbre de diagnostic optimal courant marqué T^* est l'arbre de diagnostic optimal définitif et $K(T^*) = F(N_r)$.

3.3 Heuristiques utilisées

Les heuristiques utilisées ne sont pas détaillées dans cet article. Nous précisons que nous utilisons des heuristiques admissibles qui garantissent l'obtention d'un arbre de diagnostic optimal par rapport au critère défini précédemment (équation 2) [1]. Les deux heuristiques dont nous nous servons sont issues de la théorie de l'information : l'une est basée sur la codage de Huffman [10] et l'autre sur une approche entropique prenant en compte le nombre de modalités et le coût d'un test [13].

4 Méthodes issues d'AGENDA

4.1 Sous-ensemble de tests discriminant

Un sous-ensemble de tests est dit discriminant pour un ensemble de fautes si et seulement si il est capable d'isoler toutes les fautes données.

Définition 1 (Discriminant premier)
Considérons un ensemble S de n_S tests s_j ordonnés par ordre croissant de coût. Un sous-ensemble de tests discriminant premier S_{First} (initialement vide) est défini en ajoutant successivement à S_{First} les tests de S un par un, en respectant l'ordre croissant sur les coûts des tests, jusqu'à obtenir un sous-ensemble de tests discriminant.

Définition 2 (Discriminant minimal) *Un sous-ensemble S_{Min} de tests discriminant est dit minimal si et seulement si, pour chacun de ses n_S^{Min} tests s_j^{Min} avec $j \in \{1, ..., n_S^{Min}\}$, $S^{Min} - \{s_j^{Min}\}$ n'est plus un ensemble de tests discriminant pour l'ensemble des fautes donnés.*

Il existe plusieurs sous-ensemble de tests discriminant minimal de cardinalités différentes. Dans la suite, nous le sélectionnons de la manière suivante et S_{Min} représente alors ce sous-ensemble de tests discriminant minimal particulier. S_{Min} est construit à partir d'un sous-ensemble de tests discriminant premier S_{First}[1], en testant la discriminabilité du sous-ensemble en retirant successivement chacun de ses tests.

Définition 3 (Discriminant optimal) *Un sous-ensemble de tests discriminant optimal S^* est le sous ensemble de tests discriminant associé à un arbre de diagnostic optimal donné, pour l'ensemble des fautes considérées F.*
De plus, il est important de remarquer que pour un même ensemble de tests initial, il existe autant de sous-ensembles de tests discriminants optimaux que d'arbres de diagnostic optimaux.

Remarque 1 *Un sous-ensemble de tests discriminant optimal n'est pas nécessairement un sous-ensemble de tests discriminant minimal. La figure 3 montre un exemple.*

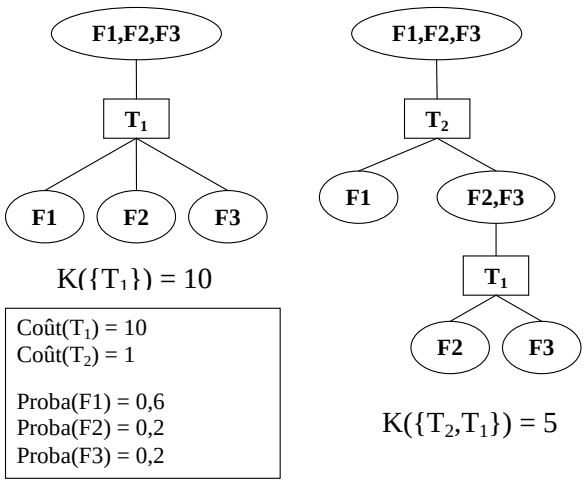

Figure 3: Sous-ensembles de tests minimal (gauche) et optimal (droite)

Remarque 2 *Un sous-ensemble de tests discriminant optimal inclut nécessairement au moins un sous-ensemble de tests discriminant minimal.*

D'après les définitions 1 et 2, deux sous-ensembles de tests discriminants pertinents sont envisageables pour générer des arbres de diagnostic sous optimaux : sous-ensemble de tests discriminant premier S_{First} et sous-ensemble de tests discriminant minimal S_{Min}. Ces 2 sous-ensembles sont en relation étroite avec l'ensemble

[1] Si le sous-ensemble de tests discriminant premier n'est pas unique, nous en choisissons un arbitrairement

des tests initial et le sous-ensemble de tests discriminant optimal associé à un arbre de diagnostic. La figure 4 montre les différentes possibilités d'inclusion entre ces différents ensembles.

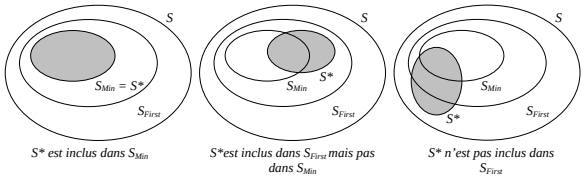

Figure 4: Différents cas d'inclusion des ensembles S, S^*, S_{First} et S_{Min}

4.2 Utilisation des sous-ensembles de tests discriminants

Faure [4] utilise les deux sous-ensembles de tests discriminants définis précédemment (définitions 1 et 2) pour construire des arbres de diagnostic sous optimaux à partir de la méthode AGENDA. Au lieu de considérer l'ensemble initial des tests (i.e. complet), il substitue à celui-ci un des 2 sous-ensembles précédents S_{Min} ou S_{First}.

Dans la suite de cet article, nous ferons référence à la méthode AO^*_{Min} pour une initialisation de l'ensemble de tests du Test Sequencing Problem avec le sous-ensemble S_{Min} et à la méthode AO^*_{First} pour une initialisation avec le sous-ensemble S_{first}.

Les résultats fournis dans [4] permettent d'affirmer que la méthode AO^*_{First} ne s'écarte pas de la valeur optimale de plus de 5 %. Ce résultat est obtenu de manière empirique. Par contre, pour la méthode AO^*_{Min}, l'écart est, dans certains cas, très important de l'ordre de 30 % mais en contre partie, l'arbre est généré très rapidement puisque la cardinalité de l'ensemble de tests utilisé est la plus petite qu'il soit possible d'avoir.

5 Réduction dynamique du nombre de tests

Dans notre domaine d'application, la cardinalité de l'ensemble des tests peut être un ordre de grandeur supérieur à celle de l'ensemble des fautes.

Considérons le cas de tests binaires (i.e. cas défavorable par rapport aux tests multi-valués) et de n_F fautes, il est nécessaire d'utiliser, au plus, $n_F - 1$ tests pour discriminer l'ensemble de fautes. Il est évident sur cet exemple que si l'ensemble des tests (ensemble discriminant) et l'ensemble des fautes ont une cardinalité de même ordre de grandeur, alors la discriminabilité n'est pas affectée (i.e. le diagnostic est toujours possible).

Dans le cas d'ordre de grandeur différent pour les 2 cardinalités, comme l'algorithme AO* évalue à chaque itération pour le nœud OU choisi, tous les tests restants (i.e. ceux qui n'ont jamais été choisis préalablement), l'étape de calcul des évaluations heuristiques devient très coûteuse en temps de calcul.

Nous proposons, par conséquent, de considérer un sous-ensemble de l'ensemble des tests restants selon un critère donné, avant de réaliser l'étape d'évaluation heuristique. Il est important de remarquer que ce critère est précalculé une seule fois pour chaque test, sinon il est évident que remplacer un calcul d'heuristique par un autre n'a aucun effet sur les temps de calcul. Ainsi définie, nous notons AO^*_{Dyn} cette méthode.

Une méthode similaire appelée *Limited Search AO** restreinte à l'utilisation de tests binaires est proposée dans [11]. Un paramètre est utilisé pour limiter le nombre de tests considérés après avoir ordonné les tests par une évaluation heuristique basée sur le gain d'information.

5.1 Méthode AO^*_{Dyn}

La méthode AO^*_{Dyn} modifie l'étape itérative de l'algorithme AO^*. En effet, au lieu de considérer tous les tests restants pour le nœud OU sélectionné, seulement un sous ensemble de tests est sélectionné. Plusieurs critères de sélection peuvent être utilisés.

Un test s_j est caractérisé par trois élément :

1. son coût c_j
2. son nombre de modalités n^j_M
3. son efficacité $eff_j = Log(n^j_M)/c_j$ [13]

Ces éléments permettent d'ordonner les tests a priori. Le plus intéressant est le troisième parce qu'il combine les deux premiers. Une fois, les tests ordonnés, un nombre constant n_{max} définissant la cardinalité du sous-ensemble de tests choisi, est défini. Il est préférable qu'il ait le même ordre de grandeur que n_F pour les raisons évoquées précédemment.

Cette méthode consiste donc à sélectionner les n_{max} premiers tests ordonnés et à développer les nœuds ET correspondants. Par ce fait, nous avons simplement réduit le nombre de tests devant être évalués par l'estimation heuristique sans évaluer aucune fonction heuristique, mais en utilisant un critère précalculé.

A l'itération suivante, pour chaque nœud OU, une nouvelle réduction est réalisée à partir de l'ensemble des tests complets (i.e. dans la suite de l'algorithme, on ne considère pas uniquement l'ensemble réduit mais tous les tests restants disponibles) (Figure 5). Alors, la réduction de l'ensemble des tests n'a pas de conséquence sur les itérations suivantes de l'algorithme.

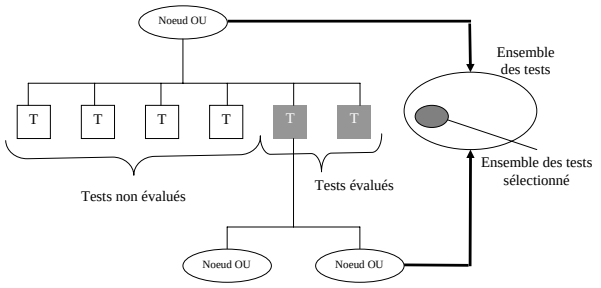

Figure 5: Conservation de l'ensemble complet de tests

6 Génération itérative d'arbres de diagnostic

Dans le domaine scientifique, certaines techniques de résolution de problème utilisent la convergence vers une solution par des itérations ou des approximations successives. Dans notre cas, nous ne sommes pas capable de fournir une solution immédiate à notre problème : résoudre le Test Sequencing Problem. C'est pourquoi, nous proposons de rechercher une solution à partir d'un ensemble de tests discriminant de cardinalité réduite que nous allons faire évoluer au cours de la recherche de solution.
Comme nous venons de le voir dans la section précédente, les deux sous-ensembles de tests décrits par les définitions 1 et 2, permettent d'obtenir une première approximation de la solution de manière acceptable en temps de calcul et en ressources matérielles.
Cette méthode est qualifiée de "*Anytime*", c'est-à-dire que l'on peut fixer un temps d'exécution maximum et se contenter de la solution alors obtenue. La maîtrise du temps de création de l'arbre est totale. Nous appelons $Anytime_{First}$ la méthode initialisée avec S_{First} et $Anytime_{Min}$ celle initialisée avec S_{Min}

6.1 Evolution de l'ensemble des tests

Comme nous venons de le préciser, le sous-ensemble de tests discriminant initial est soit S_{first}, soit S_{min}. Ce sous ensemble permet de générer un arbre de diagnostic dont le coût sert de référence au cours des différentes itérations (i.e. à chaque évolution de l'ensemble des tests).
Considérons un sous-ensemble $S_{courant}$ de tests discriminant utilisé, pour construire un arbre de diagnostic optimal.
Comme nous l'avons décrit dans l'introduction de cette méthode, $S_{courant}$ doit évoluer au fil des itérations pour nous permettre d'obtenir l'arbre de diagnostic optimal par rapport à l'ensemble des tests complet, ou du moins le plus proche possible de ce coût. Pour se faire, nous devons ajouter des tests à $S_{courant}$.
Un test s_j est caractérisé par son coût c_j, son nombre de modalités n_M^j et son efficacité $eff_j = Log(n_M^j)/c_j$ telle qu'elle est définie dans [13].
Considérons $S_{possible}$ (=$\{s_{P_0}, \ldots, s_{P_p}\}$) l'ensemble des p tests qui peuvent être ajoutés à $S_{courant}$: $S_{possible} = S \setminus S_{courant}$. Nous supposons, sans perte de généralité, que les tests sont ordonnés par ordre d'efficacité croissante dans cet ensemble. Alors le test s_{add} à ajouter à $S_{courant}$ est celui d'indice P_0.

6.2 Impact sur l'AO*

Considérons K_0 le coût de l'arbre de diagnostic créé en utilisant un des deux sous-ensembles de tests discriminants servant à initialiser $S_{courant}$, et $K_{courant}$ le coût de l'arbre de diagnostic en cours de création avec $S_{courant}$ (i.e. le coût de la racine de l'arbre). Initialement, $K_0 = K_{courant}$.
Cette approche ne modifie pas l'algorithme AO*. $K_{courant}$ ne peut pas croître au cours des itérations car l'heuristique est admissible et le sous-ensemble de tests $S_{courant}$ contient toujours le sous-ensemble optimal ayant servi à la construction de l'arbre de diagnostic de l'itération précédente. Le coût décroît donc uniquement si le test ajouté est utilisé.
De manière générale, la mise à jour de ce coût de référence est effectuée de la manière suivante lorsque l'algorithme AO* est terminé :

- Si $K_{courant} \leq K_0$, alors $K_0 = K_{courant}$.
- Si $K_{courant} = K_0$, alors K_0 conserve sa valeur.

6.3 Itération

L'addition d'un test à l'ensemble $S_{courant}$ et l'exécution de l'algorithme AO* sont répétées autant de fois que nécessaire pour se rapprocher le plus possible de la valeur optimale (a priori inconnue) du coût de l'arbre de diagnostic créé avec l'ensemble des tests complet.
Un inconvénient de la méthode est la croissance, nécessaire, de l'ensemble des tests $S_{courant}$ qui augmente le temps d'exécution de l'algorithme AO* petit à petit. Cependant, dans le cas de solutions trop éloignées, la condition d'arrêt modifiée permet de sortir rapidement de l'algorithme et de passer à l'itération suivante. Dans le cas de solutions proches de l'optimale, la valeur courante de l'arbre de diagnostic peut être améliorée, moyennant un temps de calcul un peu plus important.
Pour éviter des temps de calcul trop long, la rapidité d'exécution de l'AO* peut être privilégiée au détriment d'une recherche plus complète des sous-ensembles de tests. En effet, une cardinalité limite peut être fixée pour l'ensemble $S_{courant}$ permettant ainsi de maîtriser en partie la durée de l'itération. Lorsque cette cardinalité est atteinte, l'ensemble $S_{courant}$ est réduit au dernier sous ensemble S^* de tests discriminant optimal trouvé.

L'avantage majeur de cette approche est la maîtrise du temps de calcul de l'algorithme AO*. Ce temps peut être borné, la méthode fournissant alors la meilleure solution par rapport au temps imparti.

7 Evaluation des performances

Les méthodes proposées *Anytime* et $AO*_{Dyn}$ sont comparées avec les 3 autres méthodes suivantes : AO* avec ensemble des tests complets (donnant la solution optimale), AO^*_{Min} et AO^*_{First}, sur 3 systèmes différents Σ_i avec $i \in \{1, 2, 3\}$ triés par complexité croissante (Figure 8, 9 et 10). Ces systèmes sont des modèles réels du domaine de l'automobile.
La figure 6 montre les temps de calcul pour chaque système et chaque méthode, entre parenthèses, l'écart en pourcentage par rapport au coût optimal et, entre crochets, le nombre d'itérations pour obtenir la solution optimale pour les algorithmes *Anytime*.
La figure 7 montre l'écart en pourcentage par rapport à la valeur optimale en fonction du nombre d'itérations pour les 3 systèmes $\{\Sigma_1, \Sigma_2, \Sigma_3\}$ quand la méthode $Anytime_{Min}$ est utilisée. Σ_2 converge lentement par valeur supérieure vers la valeur du coût optimal à cause de sa structure qui requiert un test particulier. En général, au moins 10 itérations sont nécessaires pour être à moins de 5% de la valeur optimale du coût.

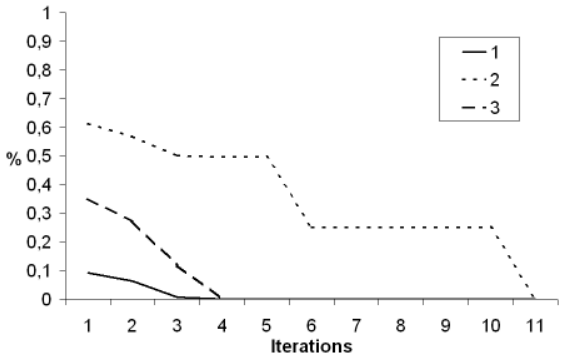

Figure 7: Ecart par rapport à la valeur optimale en fonction du nombre d'itérations pour les 3 exemples pour la méthode $Anytime_{Min}$

Les méthodes AO^*_{First} et *Anytime* sont celles qui donnent les meilleurs résultats pour notre domaine d'application. $Anytime_{First}$ est la méthode préférée parce qu'elle donne dès la première itération le même résultat que AO^*_{First} avec le même temps de calcul. Par contre, les itérations suivantes peuvent donner un résultat plus proche de l'optimal si plus de temps est disponible pour la recherche de la solution.

8 Conclusion

Cet article donne un état de l'art et décrit 2 nouvelles méthodes pour générer des arbres de diagnostic

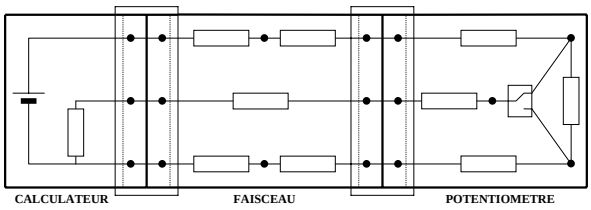

Figure 8: Système Σ_1

presque optimaux, à partir d'une approche basée sur l'algorithme AO* et en maîtrisant les temps de calcul. Ce temps de calcul est proportionnel à la cardinalité n_S de l'ensemble de tests S. C'est pourquoi une des méthodes que nous proposons consiste à réduire dynamiquement (i.e. pendant l'exécution de l'algorithme AO*) S à un sous-ensemble S' tel que $S' \subseteq S$ dans le but de réduire les temps de calcul. La seconde utilise le fait que l'algorithme AO* crée rapidement des arbres de diagnostic lorsque la cardinalité de l'ensemble des tests est faible. Ces 2 méthodes permettent d'obtenir des arbres de diagnostic presque optimaux.
Cependant, cette approche est validée de manière empirique en comparant les résultats sur un panel d'exemples représentatifs de notre domaine d'application : l'automobile. Les justifications théoriques concernant la supériorité d'une de ces méthodes sur les autres ou la quantification de l'écart à l'optimal en terme de coût sont très difficiles à caractériser. Nous avons toujours obtenu des arbres dont l'écart à l'optimal est inférieur à 5 %, pour les exemples traités.
Le problème de bornage de l'écart par rapport à la valeur du coût optimal en utilisant ces méthodes reste posé.

References

[1] A. Bagchi and A. Mahanti, 'Admissible Heuristic Search in AND/OR Graphs', *Theoritical Computer Science*, **24**, 207–219, (1983).

[2] P.P. Chakrabarti, S. Ghose, A. Acharya, and S.C. de Sarkar, 'Heuristic Search in Restricted Memory', *Artificial Intelligence*, **41**, 197–221, (1989).

[3] P.P. Chakrabarti, S. Ghose, and S.C. de Sarkar, 'Admissibility of AO* When Heuristics Overestimate', *Artificial Intelligence*, **34**, 97–113, (1988).

[4] P. P. Faure, *An Interval Model-Based Approach for Optimal Diagnosis Tree Generation : Application to the Automotive Domain*, Ph.D. dissertation, LAAS, 2001.

[5] P. P. Faure, X. Olive, L. Travé-Massuyès, and H. Poulard, 'Agenda : Automatic GENeration of Diagnosis trees', in *JDA'01*, Toulouse (France), (2001).

	Σ_1	Σ_2	Σ_3
Optimal	$< 1''$ (-)	$1'32''$ (-)	$3'55''$ (-)
AO^*_{Min}	$< 1''$ (9 %)	$< 1''$ (61 %)	$< 1''$ (35 %)
AO^*_{First}	$< 1''$ (0 %)	$1'05''$ (0 %)	$7''$ (0 %)
AO^*_{Dyn}	$< 1''(0\%)$	$< 1''(1\%)$	$3''(1\%)$
$Anytime_{Min}$	$2''$ [4 *iter.*] (0 %)	$47''$ [12 *iter.*] (0 %)	$8''$ [4 *iter.*] (0 %)
$Anytime_{First}$	$< 1''$ [1 *iter.*] (0 %)	$1'05''$ [1 *iter.*] (0 %)	$7''$ [1 *iter.*] (0 %)

Figure 6: Evaluation pour 3 systèmes différents

[6] M. R. Garey, 'Optimal binary identification procedures', *SIAM Journal of Applied Mathematics*, **23**(2), 173–186, (1972).

[7] L. Hyafil and R. L. Rivest, 'Constructing optimal binary decision trees is np-complete', *Information Processing Letters*, **5**(1), 15–17, (1976).

[8] B. M. E. Moret, 'Decision trees and diagrams', *Computer Surveys*, **14**(4), 593–623, (1982).

[9] K. R. Pattipati and M. G. Alexandridis, 'Application of Heuristic Search and Information Theory to Sequential Fault Diagnosis', *IEEE System, Man and Cybernetic*, **20**(4), 872–887, (July 1990).

[10] K. R. Pattipati and M. Dontamsetty, 'On a Generalized Test Sequencing Problem', *IEEE Transactions on Systems, Man and Cybernetics*, **22**(2), 392–396, (March 1992).

[11] V. Raghavan, M. Shakeri, and K. R. Pattipati, 'Optimal and Near-Optimal Test Sequencing Algorithms With Realistic Test Models', *IEEE System, Man and Cybernetic - Part A*, **29**(1), 11–26, (January 1999).

[12] F. Tu and K. R. Pattipati, 'Rollout Strategies for Sequential Fault Diagnosis', *IEEE System, Man and Cybernetic - Part A*, (November 2002).

[13] R. W. Yeung, 'On Noiseless Diagnosis', *IEEE System, Man and Cybernetic*, **24**(7), 1074–1082, (1994).

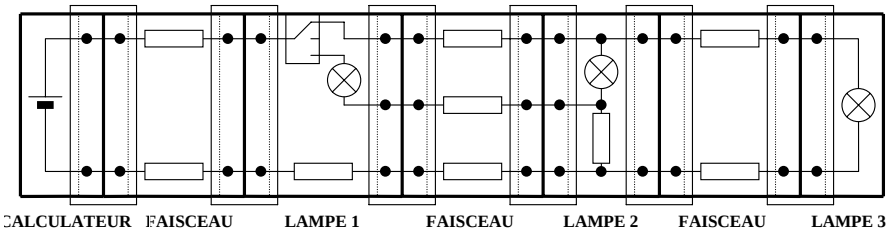

Figure 9: Système Σ_2

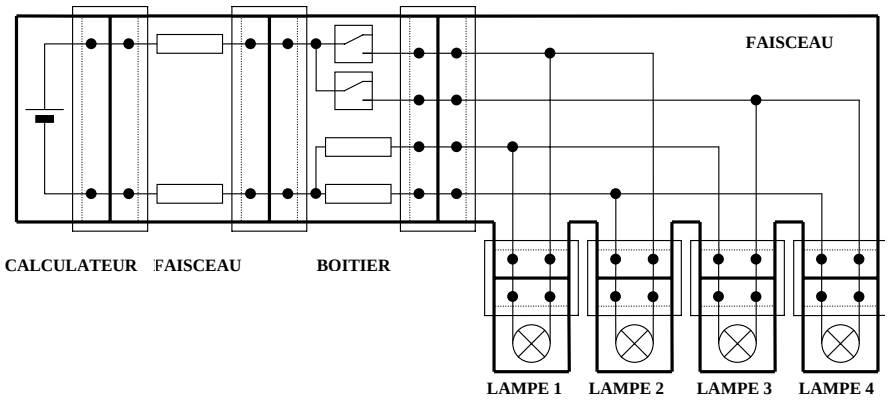

Figure 10: Système Σ_3