Résolution d'un problème de diagnostic de systèmes à événements discrets par SAT

Alban Grastien **Anbulagan**
NICTA et Australian National University
Canberra, Australia
{alban.grastien|anbulagan}@nicta.com.au

Résumé

Diagnostiquer un système consiste à déterminer si le fonctionnement du système est normal ou défectueux grâce à des observations sur ce fonctionnement. Dans le cadre des systèmes à événements discrets (SED), lorsqu'un modèle complet est fourni, cela revient à calculer sur le modèle les comportements compatibles avec les observations et tester la normalité de ces comportements. Nous montrons que les problèmes de diagnostic de SED peuvent être traduits en problèmes de satisfiabilité propositionnelle (SAT) de formules booléennes sous forme normale conjonctive (CNF). Nous résolvons ensuite le problème SAT à l'aide des meilleurs solveurs actuels. Les résultats montrent que les algorithmes SAT permettent de résoudre des problèmes que les algorithmes traditionnels de diagnostic ne peuvent traiter.

1 Introduction

Diagnostiquer un système consiste à déterminer si le fonctionnement du système est normal ou défectueux grâce à des observations sur ce comportement. Nous nous plaçons ici dans le cadre du diagnostic à base de modèle (MBD) de systèmes à événements discrets (SED). Dans ce cadre, un modèle complet décrit sous forme de *trajectoires* tous les comportements possibles du système, y compris mais éventuellement avec un certain niveau d'abstraction les comportements fautifs. Le diagnostic consiste alors à déterminer s'il existe des trajectoires normales/fautives sur le système compatibles avec les observations. La difficulté de la tâche réside principalement dans le fait que le comportement est partiellement observable.

On peut diviser les approches actuelles en deux catégories. Dans la première, le modèle est compilé en une structure permettant d'associer les observations au diagnostic. C'est principalement le cas du *diagnostiqueur* de Sampath et coll. [14]. Cette méthode souffre cependant de certaines limitations. Ainsi, la taille du diagnostiqueur est dans le pire des cas doublement exponentielle par rapport au nombre de variables du système [12], ce qui rend le diagnostiqueur impossible à calculer dans le cas général (même si des approches décentralisées ou symboliques [15, 19] permettent dans certains cas de contourner cette difficulté). De plus, le diagnostiqueur n'accepte que des observations totalement ordonnées, et l'extension à des observations incertaines conduirait à une complexité supplémentaire.

La seconde catégorie consiste à calculer l'ensemble des trajectoires du modèle compatibles avec les observations puis à tester la normalité de ces trajectoires [9, 10, 16, 7, 3]. Le calcul de toutes les trajectoires est cependant problématique pour les systèmes réalistes.

Le problème qui nous intéresse peut se voir comme trouver un chemin sur un modèle de système. Une approche efficace utilisée pour résoudre des problèmes similaires en planification classique [8] ou en vérification de modèle [2] a été de les traduire en problème de satisfiabilité propositionnelle (SAT) et ensuite de résoudre ce problème à l'aide des solveur SAT actuels. Rintanen et Grastien ont déjà utilisé une telle approche pour démontrer qu'un système n'était pas *diagnosticable* [13]. Le diagnostic de système statique a déjà été traduit en problème de satisfiabilité logique et en particulier en problème SAT [11, 18]. Williams et Nayak [20] ont proposé l'utilisation de la logique propositionnelle pour déterminer l'état courant d'un système dynamique. Dans ce papier, nous proposons de transformer le problème de diagnostic de SED en problème SAT. Nos expérimentations avec les meilleurs solveurs SAT actuels ont montré que cette traduction permet de résoudre des problèmes que les méthodes

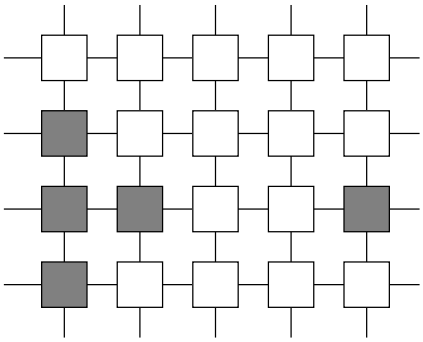


FIG. 1 – Topologie du système

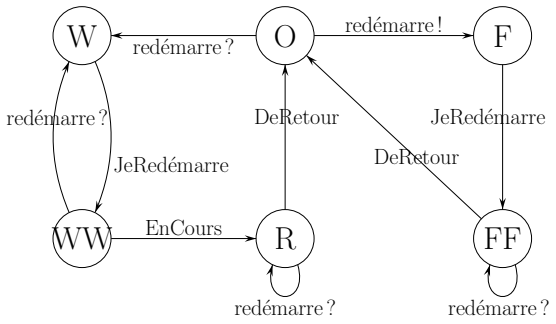


FIG. 2 – Le comportement d'un composant

classiques ne peuvent pas traiter.

La suite de l'article est divisée ainsi. Nous donnons d'abord un exemple. Puis, nous donnons la définition formelle d'un problème simple de diagnostic de SED et sa traduction en problème SAT. Ensuite, nous étendons les définitions pour considérer des cas plus complexes en terme d'observabilité et de comportement anormal. Nous présentons finalement des expérimentations sur le problème présenté en section 2.

2 Exemple

Considérons le système constitué de 20 composants identiques disposant dans une grille 5 par 4 présenté dans la figure 1. Chaque composant communique avec ses quatre voisins ; si un composant est sur un bord de la grille, il communique avec le composant à l'opposé de la grille. Ainsi, le composant $(0, 2)$ est connecté aux composants $(0, 1)$, $(0, 3)$, $(1, 2)$ et $(4, 2)$ – ces composants sont représentés en gris dans la figure 1. Le comportement de chacun des composants est représenté sous forme d'automate sur la figure 2. Les composants commencent dans l'état O. Lorsqu'une faute a lieu sur un composant, il passe dans l'état F et envoie le message *redémarre* à ses 4 voisins qui passent dans l'état W, R ou FF selon leur état courant.

Les événements *JeRedémarre* et *DeRetour* correspondent à l'émission d'une alarme par le composant en direction d'un centre de supervision. Grâce à ces alarmes, le centre de supervision doit surveiller le système et donner un diagnostic, ici déterminer quel composant a emprunté la transition entre O et F. Ainsi, si chacun des composants $(0, 2)$, $(0, 1)$, $(0, 3)$, $(1, 2)$ et $(4, 2)$ a émis les alarmes *JeRedémarre* puis *DeRetour*, alors il est très vraisemblable que le composant $(0, 2)$ est responsable de l'émission de ces alarmes.

Ce système ne peut pas être diagnostiqué avec les méthodes traditionnelles : le diagnostiqueur est une machine à états finis déterministe dont les transitions sont étiquetées par les observations. Il suffit de suivre les transitions étiquetées par les observations reçues pour avoir le diagnostic. Le nombre exact d'états du diagnostiqueur de ce système est inconnu mais borné seulement par $2^{6^{20}}$. La construction du diagnostiqueur échoue donc rapidement. La seconde méthode consiste à déplier le modèle conformément aux observations, mais à cause de la taille du système et des problèmes de concurrence, le dépliage échoue également.

3 Définitions

3.1 Système

Nous considérons des SED, c'est-à-dire des systèmes dont l'état peut être décrit par l'affectation d'un ensemble fini A de variables dans un domaine fini. Ici, nous nous restreignons à des variables booléennes, mais les résultats peuvent être étendus pour des domaines non binaires. Ainsi, un état est une fonction $s : A \to \{0, 1\}$ qui associe à chaque variable la constante 1 (*vrai*) ou 0 (*faux*). Un litéral est une variable ou sa négation, et l'ensemble des litéraux est noté $L = A \cup \{\neg a \mid a \in A\}$. Le langage $\mathcal{L}$ sur A est l'ensemble des formules qui peuvent être construites à partir de A et des connecteurs $\neg$ et $\vee$. Les connecteurs logiques suivants sont définis de manière classique : $\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi)$, $\phi \to \psi \equiv \neg\phi \vee \psi$ et $\phi \leftrightarrow \psi \equiv (\phi \to \psi) \wedge (\psi \to \phi)$.

Définition 1 (Modèle du système)
Le modèle du système *est un tuple* $SD = \langle A, \Sigma_u, \Sigma_o, \delta, s_0 \rangle$ *où*

- A *est un ensemble fini de* variables d'état *booléennes,*
- Σ_u *est un ensemble fini d'*événements non observables,

– Σ_o *est un ensemble fini d'*événements observables,
– $\delta \subseteq \Sigma_o \cup \Sigma_u \rightarrow 2^{\mathcal{L} \times 2^L}$ *associe à chaque événement un ensemble d'*instances d'événement $\langle \phi, c \rangle$, *et*
– $s_0 : A \rightarrow \{0, 1\}$ *est* l'état initial.

Une *instance de l'événement* e est une paire $\langle \phi, c \rangle \in \delta(e)$ qui indique que les effets de l'événement e peuvent être c si l'état vérifie ϕ. Plus formellement, pour que e ait lieu dans l'état s, une instance $\langle \phi, c \rangle \in \delta(e)$ doit être *tirée* telle que $s \models \phi$ (si aucune instance ne vérifie cette propriété, l'événement ne peut pas avoir lieu) et le nouvel état vaut alors $s' = succ(s, c)$ défini par :
– $s'(a) = 1$ si $a \in c$,
– $s'(a) = 0$ si $\neg a \in c$, et
– $s'(a) = s(a)$ sinon.

Le système est initialement dans l'état s_0 et traverse les états $s_1, \ldots, s_n$ grâce à la séquence d'événements $e_0, \ldots, e_{n-1}$ telle que $\forall i \in \{0, \ldots, n-1\}$, $\exists \langle \phi, c \rangle \in \delta(e_i)$ avec $s_i \models \phi$ et $s_{i+1} = succ(s_i, c)$. La séquence d'états et d'événements est appelée *trajectoire*. Pour simplifier, on confond parfois la trajectoire et la séquence d'événements par la suite. La trajectoire définit un ensemble de *dates* : l'événement e_i a lieu à la date i.

Le modèle du système est très semblable aux modèles utilisés dans la planification classique. Or, un facteur d'efficacité de la planification classique par SAT [8] est la notion de plan partiellement ordonné ou en parallèle qui permet de considérer que m événements peuvent se produire en même temps, et ainsi de ne pas prendre en compte tous les $m!$ ordonnancements de ces événements. Deux instances d'événement $\langle \phi_1, c_1 \rangle$ et $\langle \phi_2, c_2 \rangle$ *interfèrent* si une variable intervient positivement/négativement dans c_1 ou ϕ_1 et négativement/positivement dans c_2 ou ϕ_2. Deux instances interfèrent également si elles correspondent au même événement : pour un événement donné, seule une instance est tirée. Ces interférences peuvent être facilement calculées à partir du modèle.

Plutôt qu'une séquence d'instances d'événement, nous considérons donc maintenant une séquence d'ensembles (éventuellement vides) d'instances d'événements non-interférants $O_0, \ldots, O_{n-1}$ correspondant aux ensembles d'événements $E_0, \ldots, E_{n-1}$. Le successeur s' de s par l'occurrence de O est alors défini par $succ(s, \bigcup_{\langle \phi, c \rangle \in O} c)$.

3.2 Observations

La séquence d'événements qui a lieu sur le système est partiellement observable de l'extérieur du système. En effet, l'occurrence d'un événement observable génère une observation sur le système. Selon la nature des canaux de communication qui transmettent cette observation au centre de supervision, les observations peuvent être reçue plus ou moins bruitées.

Dans cette section et la suivante, nous considérons des hypothèses simplistes et des cas plus généraux sont proposés dans la section 5. Ici, les observations émises sont identiques aux observations reçues, l'événement observable ayant généré une observation est connu et la date d'occurrence de cet événement est connue.

Définition 2 (Observations)
Les observations *sont un ensemble OBS de paires $\langle e, i \rangle$ où $e \in \Sigma_o$ est un événement observable et $i \in \mathbf{N}$ est une date.*

La sémantique de OBS est la suivante : étant donné un événement observable e, si $\langle e, i \rangle \in OBS$, alors e a eu lieu à la date i ; sinon e n'a pas eu lieu à la date i.

Soit $E_0, \ldots, E_{n-1}$ une séquence d'ensembles d'événements. Cette séquence est compatible avec les observations ssi :

$$\langle e, i \rangle \in OBS \Leftrightarrow (i < n \wedge e \in E_i) \quad \forall i \in \mathbf{N}, \forall e \in \Sigma_o.$$

3.3 Diagnostic de SED

Une trajectoire peut être *normale* ou *défectueuse*. Dans cette section et la suivante, nous prenons une définition simple d'un comportement défectueux. Des extensions de cette définition sont données en section 6. Une trajectoire est normale si elle ne comporte aucun événement parmi le sous-ensemble connu de fautes $\Sigma_f \subseteq \Sigma_u$. On dit que deux trajectoires sont f-équivalentes si leur diagnostic est le même.

Les fautes ne sont pas directement observables. Il y a généralement un délai entre l'occurrence d'une faute et l'émission d'observations permettant de diagnostiquer cette faute. Dans certains cas, ce délai ne peut pas être borné et le système est non diagnosticable [14, 13]. À cause de cette incertitude, on considère souvent que le comportement du système est normal si on peut trouver un comportement normal et compatible avec les observations.

Définition 3 (Diagnostic)
Soit SD le modèle du système et soit OBS les observations sur le comportement du système. Le comportement du système est normal *s'il existe une séquence $\mathcal{E} = (E_0, \ldots, E_{n-1})$ d'ensembles d'événements normale et compatible avec les observations.*

4 Diagnostic par SAT

Comme le montre la définition 3, le problème du diagnostic de SED se ramène à trouver un chemin particulier sur le modèle du système. Ce test peut être

$$\omega^t \rightarrow \phi^t \quad (1) \qquad \omega^t \rightarrow \bigwedge_{l \in c} l^{t+1} \quad (2)$$

$$(v^t \wedge \neg v^{t+1}) \rightarrow (\omega_1^t \vee \cdots \vee \omega_k^t) \quad (3)$$

$$\neg(\omega_1^t \wedge \omega_2^t) \quad (4) \qquad (\bigvee_{\omega \in \delta(e)} \omega^t) \leftrightarrow e^t \quad (5)$$

TAB. 1 – Règles de génération de $\mathcal{T}(t, t+1)$

traduit en problème SAT, comme d'autres problèmes semblables en planification [8] ou en vérification de modèle [2].

Dans cette section, nous construisons une formule dont les valuations satisfiables correspondent aux séquences $s_0, \ldots, s_n$ d'états et $E_0, \ldots, E_{n-1}$ de transitions compatibles avec les observations. Le nombre n est la valeur maximale telle qu'il existe $\langle e, n-1 \rangle \in OBS$: on ne cherche pas à diagnostiquer ce qui s'est passé après la dernière observation. La traduction proposée n'a pas prétention d'être optimale.

Les variables propositionelles dont l'exposant t correspond à la date t sont définies comme suit :

- a^t pour toute variable d'état $a \in A$ et $t \in \{0, \ldots, n\}$,
- e^t pour tout événement $e \in \Sigma_o \cup \Sigma_u$ et $t \in \{0, \ldots, n-1\}$, et
- ω^t pour tout $\omega \in \delta(e)$, $e \in \Sigma_o \cup \Sigma_u$ et $t \in \{0, \ldots, n-1\}$.

Nous définissons à présent $\mathcal{T}(t, t+1)$ pour une date t qui modélise les transitions entre la date t et la date $t+1$. Les formules sont données dans la table 1.

Une instance d'événement $\omega = \langle \phi, c \rangle$ ne peut être tirée à la date t que si la précondition pour cette instance est satisfaite à cette date (1). Les effets c doivent s'appliquer à la date $t+1$ (2). Soit $\{\omega_1, \ldots, \omega_k\}$ l'ensemble des instances d'événement qui ont pour effet le passage de v à *faux*, alors la variable v ne peut passer de *vrai* à *faux* que si une de ces instances est tirée (3). Le passage de *faux* à *vrai* se modélise pareillement. En outre, deux événements ne peuvent avoir lieu en même temps s'ils interfèrent (4). L'instanciation des événements est représentée par la formule (5).

La conjonction de toutes ces formules pour une date donnée t est notée $\mathcal{T}(t, t+1)$. Une formule représentant l'état initial est :

$$\mathcal{I}_0 = \bigwedge_{a \in A | s_0(a)=1} a \wedge \bigwedge_{a \in A | s_0(a)=0} \neg a \quad (6)$$

La formule qui représente tous les comportements du système pour n transitions est :

$$\Phi_{SD} = \mathcal{T}(0,1) \wedge \cdots \wedge \mathcal{T}(n-1, n) \wedge \mathcal{I}_0. \quad (7)$$

Les observations sont un ensemble d'événements observables datés : l'événement observable e a eu lieu à la date t ssi $\langle e, t \rangle \in OBS$. La traduction des observations en CNF consiste à mettre à *vrai* e^t si l'événement observable e a eu lieu à la date t et à *faux* sinon.

$$\Phi_{OBS} = \bigwedge_{\langle e,t \rangle \in OBS} e^t \wedge \bigwedge_{\langle e,t \rangle \notin OBS} \neg e^t \quad (8)$$

Théorème 1
Les solutions au problème SAT $\Phi_{SD} \wedge \Phi_{OBS}$ représentent l'ensemble des trajectoires sur le modèle compatibles avec les observations.

Considérons d'une part les contraintes du problème SAT et d'autre part les contraintes du problème consistant à trouver une trajectoire sur le modèle compatible avec les observations. On montre facilement que toute contrainte du premier problème est présente dans une ou plusieurs contraintes du second, et réciproquement. Ainsi, les solutions du premier et du second problèmes sont les mêmes. □

Le problème du diagnostic est de déterminer s'il existe une trajectoire normale et compatible avec les observations, c'est-à-dire qui ne comprend pas d'événement de faute. Nous proposons donc de filtrer les trajectoires retournées par le problème SAT, comme proposé par Torta et Torasso [17] pour les systèmes statiques, avec la formule suivante :

$$\Phi_{Que} = \bigwedge_{e \in \Sigma_f, t \in \{0,\ldots,n-1\}} \neg e^t \quad (9)$$

La formule qui n'est satisfiable que s'il existe une trajectoire normale et compatible avec les observations est définie par $\Phi_\Delta = \Phi_{SD} \wedge \Phi_{OBS} \wedge \Phi_{Que}$. Si cette formule est satisfiable, alors le diagnostic du système est normal. Sinon, le diagnostic est défectueux.

5 Incertitudes sur les observations

Les observations reçues par l'observateur sont généralement différentes des observations émises : elles peuvent être bruitées ; leur ordre d'émission peut être inconnu ; certaines observations peuvent être perdues, etc. Les travaux récents dans le domaine du diagnostic [3, 9, 10] considèrent de plus en plus les incertitudes sur les observations. Nous discutons d'abord du problème du nombre de transitions empruntées sur le modèle. Puis nous considérons successivement les observations totalement ordonnées, partiellement ordonnées et enfin représentées sous forme d'automate.

5.1 Nombre de transitions

Dans l'hypothèse où toutes les observations sont datées, on connait la date de la dernière observation et il est suffisant de considérer le comportement jusqu'à

cette date puisqu'aucune information ne nous est fournie sur ce qui a eu lieu après. Cependant, si les observations ne sont plus datées, la date finale n est moins triviale à déterminer. En planification par SAT, la valeur de n est incrémentée jusqu'à trouver *une* trajectoire. Au contraire, en diagnostic, il est nécessaire de considérer les trajectoires pour *toutes* les valeurs n.

Il est possible de fixer une valeur pour n si pour toute trajectoire de longueur $k \neq n$ compatible avec les observations, il existe une trajectoire de longueur n telle que cette trajectoire est également compatible avec les observations et les deux trajectoires sont f-équivalentes. En effet, la première trajectoire est prise en compte grâce à la seconde.

On peut utiliser cette propriété de plusieurs manières. Tout d'abord, soit $E_0, \ldots, E_{k-1}$ une séquence d'ensembles d'événements de longueur k. La séquence $E_0, \ldots, E_i, \emptyset, E_{i+1}, \ldots, E_{k-1}$ de longueur $k+1$ est vraisemblablement compatible avec les observations et f-équivalente avec la première.[1] On peut alors utiliser une sur-estimation du nombre de transitions n. Ainsi, si au plus x événements non observables peuvent avoir lieu entre deux observations, une sur-estimation de n est $(x+1) \times p$ où p est le nombre maximum d'observations émises.

Cette sur-estimation peut ne pas exister, lorsque le modèle a des boucles non observables, ou générer un trop grand nombre de variables. Cependant, il est possible de réduire la valeur de x dans de nombreux cas. En effet, on peut transformer des trajectoires compatibles avec les observations en de nouvelles trajectoires plus petites également compatibles avec les observations et f-équivalentes aux premières en fusionnant certains ensembles de transitions ou en déplaçant ces ensembles. Par exemple, soit une séquence E_0, E_1, E_2, E_3 telle que les événements de E_1 et E_2 n'interfèrent pas. Alors, la séquence $E_0, E_1 \cup E_2, E_3$ est vraisemblablement équivalente. Ceci permet de considérer une valeur plus faible pour x. Ainsi, dans l'exemple de la section 2, il est suffisant de considérer $x = 1$.

5.2 Ordre total sur les observations

Nous considérons que les événements observables qui ont eu lieu sont connus, ainsi que l'ordre de leur occurrence, mais non pas la date d'occurrence. Nous donnons une estimation de la date d'occurrence de l'événement observable numéro i de la liste et notons $d(i)$ cette estimation. Comme vu dans le paragraphe précédent, une valuation de $d(i)$ peut être $((x+1) \times i) - 1$. Il est ensuite possible d'utiliser la traduction utilisée dans la section 4. La seule difficulté est donc d'estimer x et de réduire cette valeur autant que possible sans perdre de diagnostic.

[1] Cette assertion dépend cependant de la définition des observations et de celle d'un comportement fautif.

5.3 Ordre partiel sur les observations

Nous considérons maintenant que l'ordre d'émission des observations n'est que partiellement connu. Les observations sont alors représentées par un ensemble de paires $o = \langle e, j \rangle$ et un ordre partiel $\prec$ sur cet ensemble. La sémantique de cet ordre est la suivante : si $\langle e_1, j_1 \rangle \prec \langle e_2, j_2 \rangle$, alors l'événement observable e_1 ayant généré la première observation a eu lieu avant l'événement e_2 ayant généré la seconde observation. On considère que deux observations correspondant au même événement sont ordonnées.

Comme dans le paragraphe précédent, nous considérons que l'événement observable numéro i a eu lieu à la date $d(i)$. Nous créons une variable propositionnelle $o^{d(i)}$ pour chaque observation o et chaque date $d(i)$ dont la sémantique est que l'observation o est émise à la date $d(i)$ si cette variable est *vraie*. De plus, nous créons la variable $\hat{o}^{d(i)}$, *vraie* si l'observation o a été émise à la date $d(i)$ ou avant.

$$\hat{o}^{d(i)} \leftrightarrow \hat{o}^{d(i-1)} \vee o^{d(i)} \tag{10}$$

$$\hat{o}^{d(i-1)} \rightarrow \neg o^{d(i)} \tag{11}$$

$$\hat{o}^{d(p)} \qquad \forall o \in OBS \tag{12}$$

$$o_2^{d(i)} \rightarrow \hat{o}_1^{d(i-1)} \qquad \forall o_1, o_2,\ o_1 \prec o_2 \tag{13}$$

$$e^t \leftrightarrow o_1^t \vee \cdots \vee o_k^t \qquad \forall e \in \Sigma_o \tag{14}$$

La formule Φ_{OBS} est la conjonction des formules suivantes. Les variables qui n'existent pas (en particulier $\hat{o}^{d(0)}$) sont *fausses*. Tout d'abord, l'observation est émise à la date $d(i)$ ou avant ssi elle est effectivement émise à la date $d(i)$ ou si elle est émise à la date $d(i-1)$ ou avant (10). Une observation n'est émise qu'une seule fois (11). Toutes les observations sont émises par le système (12) où p est le nombre d'observations. L'ordre partiel est défini en (13). Finalement, un événement observable n'a lieu que si une observation correspondante a été émise (14), où $o_1, \ldots, o_k$ représente les observations associées à l'événement e. Remarquez que si $\nexists i : t = d(i)$, alors cette dernière clause est équivalente à $\neg e^t$.

Le nombre de variables peut être fortement réduit. Ainsi, si $o_1 \prec o_2$, alors o_2 n'a pas eu lieu à la date $d(1)$ puisque c'est au moins le second événement. Ainsi, s'il y a k_1 observations précédant certainement o et k_2 observations suivant certainement o, alors il n'est nécessaire de créer que les variables $o^{d(k_1+1)}, \ldots, o^{d(p-k_2)}$ et $\hat{o}^{d(k_1+1)}, \ldots, \hat{o}^{d(p-k_2)}$. De même, le nombre de clauses

est réduit : si $o_1 \prec o_2 \prec o_3$, l'équation (13) ne nécessite pas d'être utilisée pour la relation $o_1 \prec o_3$. Il est raisonnable de penser que dans le cadre du diagnostic, le nombre de variables et de clauses créées pour représenter les observations est ainsi linéaire.

5.4 Automate des observations

Les récents travaux considèrent des représentations plus générales des observations : perte d'observations, bruit sur les observations, etc. Les observations sont alors représentées sous la forme d'un automate appelé *index space* [9] ou *automate des observations* [5] où les chemins entre un état initial et un état final représentent l'ensemble des séquences d'événements observables compatibles avec les observations.

Il est possible de traduire cet automate en une formule SAT d'une manière comparable à celle utilisée pour traduire le modèle du système en formule SAT. La principale difficulté est de déterminer le nombre maximum de transitions n, en particulier si la taille des chemins sur l'automate n'est pas bornée. Il convient alors de définir une borne maximale. D'autre part, il convient de définir des clauses représentant l'ensemble des états finaux à la date $t = n$.

Cette approche peut être encore raffinée en intégrant dans les observations des informations concernant non seulement les événements observables mais également des variables d'états (bien que les variables d'états soient généralement considérées non observables).

6 Dysfonctionnements non triviaux

Dans les sections précédentes, nous considérions que le comportement du système était fautif si un événement de panne avait lieu. Dans le cadre du diagnostic, une information plus complète est généralement attendue. Ainsi, on souhaite connaître le nombre de fautes et leur localisation. Plus récemment, des définitions plus complexes d'un comportement fautif ont été proposées.

6.1 Nombre de fautes

Nous cherchons à connaître le nombre minimal de fautes qui ont eu lieu sur le système. Pour cela, nous définissons les formules SAT Φ_{Que_i} qui sont satisfiables si exactement i variables e^t sont évaluées à *vrai* en utilisant la traduction proposées par Bailleux et Boufkhad [1] avec $e \in \Sigma_f$. Remarquons que la formule SAT Φ_{Que} défini jusque là est équivalent à Φ_{Que_0}. Si la formule $\Phi_{\Delta_i} = \Phi_{SD} \wedge \Phi_{OBS} \wedge \Phi_{Que_i}$ est satisfiable, alors il existe une trajectoire sur le modèle comprenant i événements de faute et compatible avec les observations.

Nous testons la satisfiabilité de Φ_{Δ_i} en commençant par $i = 0$ et en incrémentant cette valeur jusqu'à obtenir une formule satisfiable. La valeur de i représente alors le nombre minimal de fautes ayant eu lieu sur le système. Le nombre d'appel à un solveur SAT est $i+1$.

Si i est très grand, on peut réduire le nombre d'appels au solveur. Soit $\Phi_{Que_{i \to j}}$ la formule SAT satisfiable si entre i et j événements fautifs ont eu lieu. Remarquons que cette formule se code très bien avec [1]. À chaque appel au solveur, on double la valeur de i dès que $i \neq 0$ en testant la satisfiabilité de $\Phi_{SD} \wedge \Phi_{OBS} \wedge \Phi_{Que_{i \to 2 \times i - 1}}$. Puis, une fois qu'il est identifié que le nombre minimal de fautes k est entre i et $2 \times i - 1$, on peut retrouver la valeur exacte de k par dichotomie. Le nombre d'appels au solveur est alors $2 \times \log_2 k$.

6.2 Localisation des fautes

L'objectif principal du diagnostic est d'indiquer quels sont les composants défectueux dans un système pour pouvoir utiliser les capacités réelles du système ou pour pouvoir le réparer. On appelle maintenant *diagnostic* la liste des événements de faute ainsi que leur date d'occurrence. Étant donné la formule SAT satisfiable Φ_{Δ_i}, le diagnostic S_1 peut être aisément extrait de la solution fournie par le solveur. Ce diagnostic n'est cependant pas nécessairement le seul diagnostic de taille i pour ce problème. Si le solveur renvoie toutes les solutions au problème, il est possible d'extraire tous les diagnostics. Sinon, il est nécessaire de rappeler le solveur en prenant soin d'interdire les diagnostics $S_1, \ldots, S_k$ déjà calculés avec la formule suivante :

$$\Phi_{\Delta_i} \wedge \bigwedge_{j \in \{1,\ldots,k\}} \Big(\bigvee_{e^t \in S_j} \neg e^t \Big)$$

Lorsque cette formule n'est plus satisfiable, tous les diagnostics de taille i ont été calculés. Il est donc nécessaire d'appeler encore k fois le solveur s'il y a k diagnostics.

6.3 Motifs de surveillance

Jéron et coll. ont récemment proposé [6] de considérer des motifs de surveillance plutôt que de simples fautes. Un motif de surveillance Que est un automate dont certains états sont étiquetés comme étant fautifs. Si une séquence d'événements conduit à un état fautif sur cet automate, alors le comportement est fautif.

Il est possible de traduire cet automate en formule SAT Φ_{Que} comme nous l'avons montré pour le modèle du système Φ_{SD} ou lorsque les observations sont représentées sous forme d'automate. Il suffit de forcer les états finaux à être non fautifs/fautifs pour ne conserver que les trajectoires normales/fautives. Le motif Que

est reconnu si $\Phi_{SD} \wedge \Phi_{OBS} \wedge \Phi_{Que}$ est satisfiable. On peut étendre cette approche en intégrant dans Φ_{Que} des informations sur la valuation des variables d'états.

D'une manière générale, on peut considérer qu'il existe un ensemble de motifs QUE partitionné en $\{QUE_1, \ldots, QUE_k\}$ tel que si deux motifs $Que_i \in QUE_i$ et $Que_j \in QUE_j$ peuvent être diagnostiqués, on diagnostique en priorité Que_i si $i < j$. QUE_0 contient le motif de comportement normal.

Comme présenté précédemment, on cherche des trajectoires sur le modèle compatibles avec les observations et reconnues par un motif de QUE_i en partant de $i = 0$ et en incrémentant cette valeur jusqu'à ce qu'un diagnostic soit trouvé. Ceci est fait en testant la satisfiabilité de la formule suivante :

$$\Phi_{\Delta_i} = \Phi_{SD} \wedge \Phi_{OBS} \wedge \Big(\bigvee_{Que \in QUE_i} \Phi_{Que} \Big).$$

Lorsque le solveur a trouvé une solution, il est facile d'extraire le motif $Que \in QUE_i$ reconnu et le solveur peut être rappelé en interdisant la reconnaissance de ce motif avec la formule suivante : $\Phi_{\Delta_i} \wedge \neg\Phi_{Que}$.

7 Validation expérimentale

7.1 Le problème de diagnostic

Nous reprenons le modèle présenté en section 2. Nous avons voulu tester l'efficacité de l'approche par SAT pour différents cas de figure. Aussi, nous avons construit 360 problèmes de diagnostic sur ce système avec les paramètres suivants :

longueur du problème nous avons considéré des problèmes comprenant de 1 à 20 fautes, avec environ 8 observations par faute. Le but est ici de déterminer comment la complexité évolue quand le nombre d'observations augmente.

difficulté du problème certains scénarios sont plus difficiles à reconstruire que d'autres. En particulier, lorsque peu de composants retournent à l'état O, il est plus facile de déterminer quel composant tombe en panne. Nous avons donc généré aléatoirement des scénarios où les composants retournent rapidement en O (scénarios dits *difficiles*), des scénarios où la moitié des composants retourne en O (*moyens*) et des scénarios où les composants retournent rarement en O (*faciles*).

observabilité nous avons considéré des observations datées, des observations totalement ordonnées et des observations partiellement ordonnées ; dans ce dernier cas, l'observation numéro i précède sûrement l'observation j si $i + 5 < j$.

solution étant donné le nombre k de fautes dans un scénario, nous avons cherché des trajectoires comprenant k fautes (**s**atisfiable) et des trajectoires comprenant $k - 1$ fautes (**n**on satisfiable). Pour effectuer un diagnostic par SAT, il est nécessaire de pouvoir résoudre ces deux types de problèmes.

Le plus petit scénario comporte 7860 variables, 32560 clauses et 71840 litéraux, tandis que la plus grande instance comporte 366284 variables, 1770145 clauses et 4124906 litéraux.

7.2 Les solveurs SAT

Nous utilisons la puissance des algorithmes SAT actuels, basés soit sur la recherche systématique, soit sur la recherche locale stochastique.[2] Les meilleurs solveurs systématiques sont basés sur la procédure Davis-Putman-Logemann-Loveland (DPLL) [4]. Nous avons utilisé les solveurs suivants dans notre étude :

- **March_dl** est l'un des meilleurs algorithmes des approches prospectives qui utilise l'algorithme *double look-ahead* (LA) basé sur la procédure DPLL pour résoudre les problèmes industriels et réalistes ;
- **MiniSat v1.14, zChaff v151104 et Siege v4**[3] sont les meilleurs algorithmes des approches rétrospectives qui utilisent l'algorithme *conflict-driven clause learning* (CDCL) basé sur la procédure DPLL pour résoudre les problèmes industriels de grande taille ;
- **R+AdaptNovelty$^+$** est l'un des meilleurs solveurs de la famille WalkSAT avec un prétraitement par calcul de résolvantes dans la formule SAT en entrée ;
- **R+RSAPS** est l'un des meilleurs algorithmes de recherche locale stochastique (SLS) basé sur la méthode de pondération des clauses avec un pré-traitement par calcul de résolvantes dans la formule SAT en entrée.

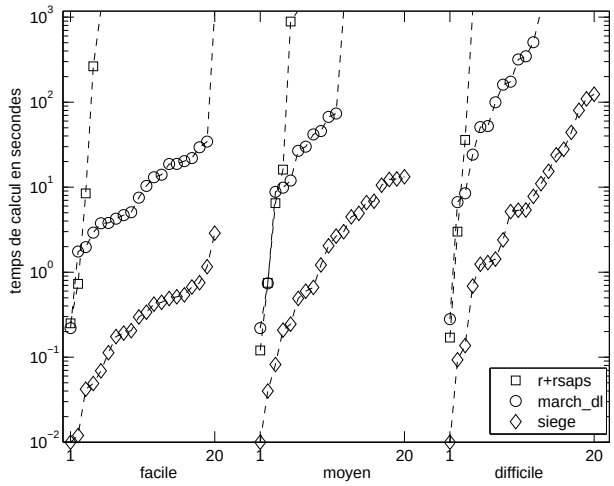
7.3 Résultats

Les tests ont été effectués sur un cluster de 16 processeurs AMD Athlon 64 fonctionnant à 2 GHz avec 2Go de RAM. La figure 3 présente les performances de quelques solveurs pour certaines classes de problèmes. Nous ne donnons pas les résultats de R+AdaptNovelty$^+$ qui n'est pas parvenu à résoudre la moindre instance en 20 minutes.

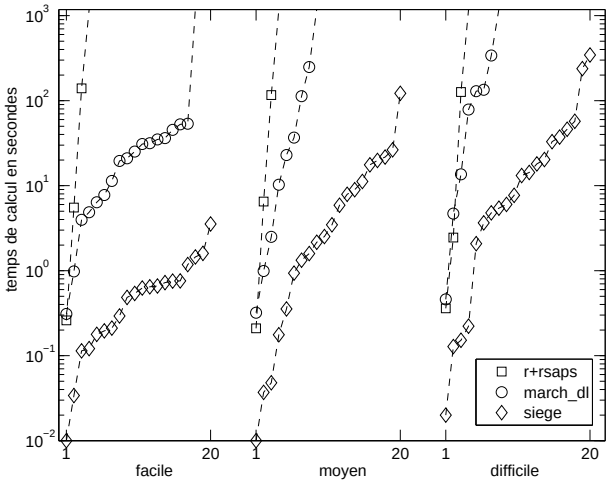
Les figures 3a à 3c donnent les résultats d'un solveur SLS, un LA et un CDCL pour différentes classes de problèmes satisfiables. Les résultats sont triés par

[2] Tous ces solveurs peuvent être téléchargés depuis le site www.satcompetition.org/

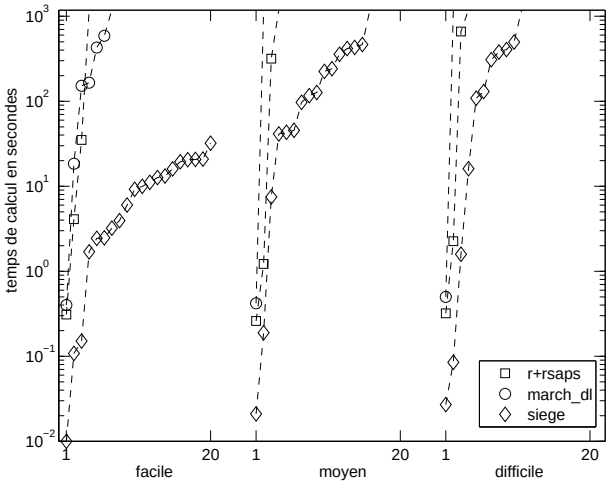
[3] www.cs.sfu.ca/∼cl/software/siege/



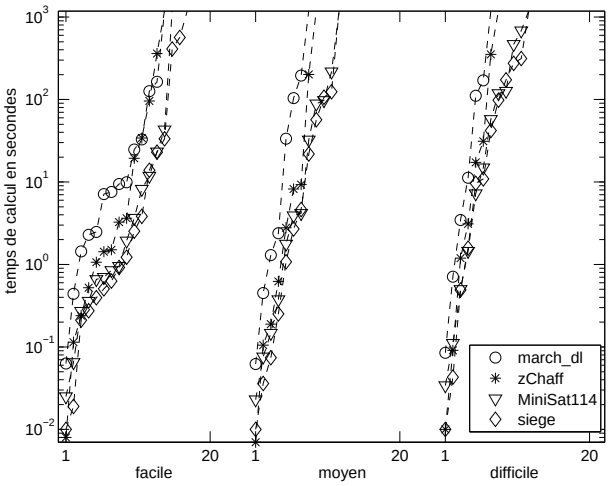
a. Problèmes datés triés (sat)



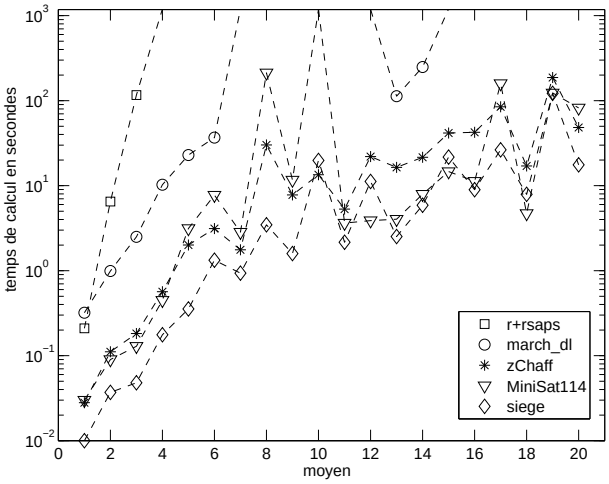
b. Problèmes totalement ordonnés triés (sat)



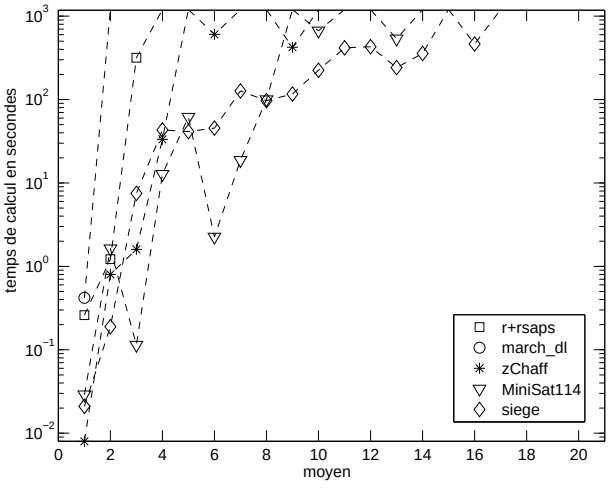
c. Problèmes partiellement ordonnés triés (sat)



d. Problèmes totalement ordonnés triés (non sat)



e. Problèmes totalement ordonnés (sat)



f. Problèmes partiellement ordonnés (sat)

FIG. 3 – Performances des solveurs SAT pour les certaines problèmes de diagnostic

Problème	R+RSAPS			March_dl			MiniSat			Siege			zChaff		
	#Sol	Tmed	Tmoy	#Sol	Tmed	Tmoy	#Sol	Tmed	Tmoy	#Sol	Tmed	Tmoy	#Sol	Tmed	Tmoy
date-facile-*-s	4	1200	973.68	19	8.93	70.85	20	0.37	**0.44**	20	**0.32**	0.47	20	1.16	1.18
date-moyen-*-s	5	1200	945.60	11	70.35	555.83	20	2.88	6.82	20	**2.35**	**4.14**	20	3.39	8.38
date-difficile-*-s	3	1200	1021.96	12	332.29	567.48	16	23.28	256.94	20	**5.37**	**23.35**	20	20.17	43.01
date-facile-*-n	–	–	–	15	7.11	305.20	16	**3.65**	242.97	17	3.67	**198.35**	15	14.64	309.93
date-moyen-*-n	–	–	–	8	1200	735.09	13	130.15	462.98	13	**79.26**	**443.75**	8	1200	721.77
date-difficile-*-n	–	–	–	4	1200	960.45	11	167.64	566.95	11	**150.55**	**564.77**	9	1200	727.74
total-facile-*-s	3	1200	1027.28	17	28.02	199.32	20	0.70	0.75	20	**0.58**	**0.71**	20	1.67	1.68
total-moyen-*-s	3	1200	1026.15	8	1200	741.79	19	6.22	92.66	20	**2.99**	**12.70**	20	14.95	27.28
total-difficile-*-s	3	1200	1026.47	7	1200	815.01	14	134.10	413.47	20	**10.42**	**42.59**	20	27.84	53.19
total-facile-*-n	–	–	–	13	28.74	439.41	14	5.93	364.78	16	**3.18**	**292.77**	13	26.74	446.08
total-moyen-*-n	–	–	–	7	1200	796.83	11	157.21	562.30	11	**115.18**	**555.89**	8	1200	731.13
total-difficile-*-n	–	–	–	6	1200	854.84	11	574.88	613.82	11	**293.93**	**586.18**	7	1200	800.26
partial-facile-*-s	3	1200	1021.97	6	1200	907.68	20	**4.35**	**7.43**	20	9.59	10.29	20	53.45	70.82
partiel-moyen-*-s	3	1200	1035.92	1	1200	1140.02	10	933.31	669.98	15	**232.96**	**430.59**	6	1200	893.23
partiel-difficile-*-s	3	1200	1053.20	1	1200	1140.03	6	1200	842.21	10	**848.89**	**692.46**	4	1200	971.05
partiel-facile-*-n	–	–	–	2	1200	1080.03	4	1200	973.01	4	1200	**968.90**	3	1200	1020.84
partiel-moyen-*-n	–	–	–	2	1200	1080.03	4	1200	994.38	4	1200	**973.41**	3	1200	1021.36
partiel-difficile-*-n	–	–	–	2	1200	1080.04	3	1200	1020.74	4	1200	**987.59**	3	1200	1021.15
Total #Sol	30/180			141/360			232/360			**256/360**			219/360		

Tab. 2 – Performances des solveurs SAT pour les problèmes de diagnostic

temps de calcul. Comme prévu, le temps augmente avec la complexité des problèmes de diagnostic et les incertitudes sur les observations. Les résultats pour les observations datées et les observations totalement ordonnées sont proches puisque le nombre x de transitions entre deux événements observables est très faible. Siege résoud la plupart des problèmes en moins de 10 minutes alors que les solveurs SLS ou LA sont bien moins performants. Remarquez que ces problèmes ne peuvent pas être résolus avec les méthodes classiques.

La figure 3d présente les temps de calcul pour les problèmes non satisfiables dont les observations sont totalement ordonnées (les autres types d'observations donnent des tendances équivalentes). Logiquement, les temps de calcul sont beaucoup plus longs. Pour les problèmes les plus difficiles, Siege ne parvient à prouver la non satisfiabilité que jusqu'à $k = 11$. Nous pensons que la difficulté provient principalement de la grande valeur de k, et que d'autres problèmes avec autant d'observations mais une faible valeur k seraient résolus beaucoup plus facilement.

Les figures 3e et 3f montrent comment le temps de calcul évolue avec le nombre de fautes à diagnostiquer. Nous nous intéressons ici uniquement aux problèmes satisfiables de difficulté moyenne et pour des observations totalement ou partiellement ordonnées. Il est intéressant de remarquer que la difficulté n'augmente pas nécessairement avec le nombre de fautes à diagnostiquer. Ainsi, figure 3e, 20 secondes sont nécessaires à Siege pour trouver une solution au problème 10, alors que 2 secondes sont suffisantes pour le problème 11. Ceci est vraisemblablement dû au fait que les observations supplémentaires dans le problème 11 permettent de retrouver plus facilement le comportement que dans le problème 10.

Un autre résultat intéressant est qu'aucun solveur SAT n'est plus efficace que les autres. Ainsi, si Siege est souvent le meilleur solveur, MiniSat et zChaff sont parfois plus efficaces. D'autre part, la difficulté du problème dépend grandement de l'algorithme utilisé. Nous pensons que déterminer des heuristiques spécialement efficaces pour les problèmes de diagnostic est un problème ouvert intéressant.

La table 2 résume le nombre de problèmes résolus par chaque solveur. Tmed et Tmoy représentent les temps médian et moyen, #Sol le nombre d'instances résolues par un solveur en 1 200 secondes par instance. Aucun résultat n'est donné pour les problèmes non satisfiables dans la colonne de R+RSAPS puisque les solveurs SLS ne peuvent prouver la non satisfiabilité. Pour chaque ligne, 20 problèmes doivent être résolus comprenant de 1 à 20 fautes. La plupart des problèmes sont résolus par les solveurs CDCL, 71% pour Siege, 64% pour MiniSat et 60% pour zChaff tandis que les autres solveurs sont moins efficaces.

8 Conclusion et travaux futurs

Nous avons présenté une traduction d'un problème de diagnostic de système dynamique en problème SAT. Nos expérimentations avec les meilleurs solveurs SAT actuels ont montré que cette traduction permet de résoudre des problèmes que les méthodes classiques ne peuvent pas traiter. Nous pensons que les algorithmes de diagnostic actuels peuvent s'inspirer des algorithmes utilisés en SAT pour améliorer leurs performances.

Du point de vue SAT, les résultats montrent de meilleures performances pour les algorithmes basés sur l'apprentissage de clauses. Des heuristiques spécialement dédiées au diagnostic pourraient être développées pour améliorer encore ces résultats.

Le problème de diagnostic incrémental consiste à mettre à jour le diagnostic étant données de nouvelles

observations sur le comportement du système. Ce problème est très important dans le domaine du diagnostic puisqu'il permet la surveillance *en ligne*, c'est-à-dire pendant que le système fonctionne. la méthode présentée dans cet article doit être étendue pour permettre le diagnostic incrémental. La difficulté est de s'assurer que le diagnostic d'une fenêtre temporelle va être compatible avec le diagnostic de la fenêtre temporelle suivante. Une possibilité est de diagnostiquer la première fenêtre en prenant en compte les observations (ou une partie d'entre elles) de la fenêtre suivante.

Remerciements

Ce travail est supporté par NICTA au sein des projets SuperCom et G12. NICTA est subventionné au travers de l'initiative du Gouvernement Australien *Backing Australia's Ability*, en partie au travers de l'*Australian National Research Council*. Nous remercions les relecteurs pour la pertinence de leurs commentaires, ainsi que Jussi Rintanen et Elena Kelareva pour le travail préliminaire sur ce sujet.

Références

[1] O. Bailleux and Y. Boufkhad. Efficient CNF encoding of Boolean cardinality constraints. In *Proc. of 9th CP*, pages 108–122, 2003.

[2] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Proc. of 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS-99)*, pages 193–207, 1999.

[3] M.-O. Cordier and A. Grastien. Exploiting independence in a decentralised and incremental approach of diagnosis. In *Proc. of 20th IJCAI*, pages 292–297, 2007.

[4] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communication of ACM*, 5 :394–397, 1962.

[5] A. Grastien, M.-O. Cordier, and Ch. Largouët. Incremental diagnosis of discrete-event systems. In *Proc. of 16th International Workshop on Principles of Diagnosis (DX-05)*, pages 119–124, 2005.

[6] T. Jéron, H. Marchand, and M.-O. Cordier. Motifs de surveillance pour le diagnostic de systèmes à événements discrets finis. In *Quinzième congrès francophone de Reconnaissance de formes et intelligence artificielle (RFIA-06)*, 2006.

[7] G. Jiroveanu and R. Boël. Petri net model-based distributed diagnosis for large interacting systems. In *Proc. of 16th International Workshop on Principles of Diagnosis (DX-05)*, pages 25–30, 2005.

[8] H. Kautz and B. Selman. Pushing the envelope : planning, propositional logic, and stochastic search. In *Proc. of 13th AAAI*, pages 1194–1201, 1996.

[9] G. Lamperti and M. Zanella. *Diagnosis of Active Systems*. Kluwer Academic Publishers, 2003.

[10] Y. Pencolé and M.-O. Cordier. A formal framework for the decentralised diagnosis of large scale discrete-event systems and its application to telecommunication networks. *Artificial Intelligence (AIJ)*, 164 :121–170, 2005.

[11] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence (AIJ)*, 32 :57–95, 1987.

[12] J. Rintanen. Diagnosers and diagnosability of succinct transition systems. In *Proc. of 20th IJCAI*, pages 538–544, 2007.

[13] J. Rintanen and A. Grastien. Diagnosability testing with satisfiability algorithms. In *Proc. of 20th IJCAI*, pages 532–537, 2007.

[14] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9) :1555–1575, 1995.

[15] A. Schumann, Y. Pencolé, and S. Thiébaux. Symbolic models for diagnosing discrete-event systems. In *Proc. of 16th ECAI*, pages 1085–1086, 2004.

[16] R. Su and W. M. Wonham. Global and local consistencies in distributed fault diagnosis for discrete-event systems. *Transactions on Automatic Control*, 50(12) :1923–1935, 2005.

[17] G. Torta and P. Torasso. The role of OBDDs in controlling the complexity of model based diagnosis. In *Proc. of 15th International Workshop on Principles of Diagnosis (DX-04)*, 2004.

[18] A. Veneris. Fault diagnosis and logic debugging using Boolean satisfiability. In *4th International Workshop on Microprocessor Test and Verification : Common Challenges and Solutions*, pages 60–65, 2003.

[19] Y. Wang, T.-S. Yoo, and S. Lafortune. New results on decentralized diagnosis of discrete-event systems. In *42nd Annual Allerton Conference on Communication, Control, and Computing*, 2004.

[20] B. Williams and P. Nayak. A model-based approach to reactive self-configuring systems. In *Proc. of 13th AAAI*, pages 971–978, 1996.