

---

# Une méthode complète de recherche locale pour des bases de connaissance propositionnelles non monotones

---

**Éric GRÉGOIRE**

CRIL – Université d’Artois  
rue de l’Université SP 16  
F-62307 Lens Cedex, France  
gregoire@cril.univ-artois.fr

**Bertrand MAZURE**

CRIL – Université d’Artois  
rue de l’Université SP 16  
F-62307 Lens Cedex, France  
mazure@cril.univ-artois.fr

## Résumé

Dans ce papier, nous présentons une nouvelle approche pour calculer de façon effective des inférences au sein d’une logique non monotone propositionnelle simple mais utile, fondée sur un concept de modèles préférés. L’approche proposée est originale à deux égards au moins. Premièrement, elle se base sur des techniques de recherche locale tout en préservant la complétude logique. En second lieu, elle s’avère efficace d’un point de vue expérimental pour une classe importante de grandes bases de connaissance non monotones. Plus précisément, nous étendons à un cadre non monotone des résultats heuristiques récents liés à la résolution pratique du problème de satisfaisabilité d’un ensemble de clauses (SAT). De fait, la procédure de preuve employée se base sur l’utilisation d’une méthode de recherche locale pour SAT et sur une heuristique efficace lorsque cette recherche locale n’a pu exhiber un modèle. Nous l’employons au sein d’un formalisme de représentation permettant des règles de raisonnement par défaut avec priorités exprimées à l’aide de propositions d’anormalités à la McCarthy. Un domaine d’applications typique concerne les formes de raisonnement révisable qui peuvent être tenues au sujet d’un modèle « profond » d’un système ou d’un appareillage complexe, où les propositions d’anormalité sont utilisées pour représenter les défaillances possibles de composants et où un nombre limité de pannes peuvent survenir simultanément.

## 1 Introduction

La conception et l’étude de logiques non monotones sont des questions primordiales au centre de nombreuses recher-

ches en représentation des connaissances et de la formalisation de raisonnements. Beaucoup d’efforts ont été consacrés à la définition de formalismes au pouvoir expressif suffisant et adéquat et possédant des propriétés logiques souhaitables. Cependant, même dans le cadre élémentaire de la logique propositionnelle, les facettes computationnelles de ces logiques ont été souvent négligées par la plupart des chercheurs, sans doute à cause des propriétés de complexité algorithmique rendant apparemment ces logiques intraitables. Par exemple, le schéma habituel de raisonnement par défaut « s’il est consistant de supposer  $X$ , alors inférer de manière révisable  $Y$  », nécessite un test de consistance, lequel est exponentiel dans le pire cas dans la logique propositionnelle standard (en supposant que  $P \neq NP$ ). Il n’est donc pas étonnant que des travaux théoriques récents ont montré l’intractabilité de la plupart des formalismes non monotones (voir e.g. (Cadoli et Schaerf, 1993) pour une bonne synthèse de ces résultats).

Dans un même temps, un progrès impressionnant a été obtenu dans la résolution pratique d’instances du problème SAT. SAT est en rapport direct avec le test précédent de consistance ; il consiste à vérifier si une formule propositionnelle mise sous forme normale conjonctive (CNF) admet ou n’admet pas de modèle. Il constitue un problème  $NP - complet$  de référence, ce qui signifie que n’importe quel algorithme de résolution pour SAT sera exponentiel dans le pire des cas (à moins que  $P = NP$ ). Utilisant des algorithmes de recherche locale étonnamment efficaces, ainsi qu’initialement proposé par Selman *et al.* (Selman *et al.*, 1992), de grandes et difficiles instances de SAT peuvent être démontrées consistantes. Cependant, les techniques de recherche locale et stochastique sont par nature logiquement incomplètes. Elles peuvent être utilisées pour montrer qu’une formule CNF admet un modèle. Cependant, lorsque la méthode échoue après un temps de calcul fixé à l’avance, nous ne pouvons pas conclure que la formule est inconsistante. D’un autre côté, les algorithmes logiquement complets ne peuvent actuellement souvent traiter que des problèmes de plus petite taille (Selman *et al.*, 1997). Récem-

ment, nous avons montré que les techniques de recherche locale peuvent également s'avérer efficaces pour aussi établir qu'une formule est inconsistante (Mazure et al., 1996). A cette fin, le travail effectué par la recherche locale lorsqu'elle n'a pas réussi à exhiber un modèle est étudié. Le plus souvent, une heuristique efficace permet de détecter un noyau inconsistant directement. Plus généralement, cette heuristique fournit une stratégie de branchement qui accentue souvent l'efficacité des méthodes complètes de type Davis & Putnam (Davis et Putnam, 1960). Une telle synergie entre méthodes de recherche locale et techniques complètes est reconnue comme une question essentielle pour l'obtention de progrès supplémentaires en raisonnement propositionnel (Selman et al., 1997). De plus, elle peut constituer une clé pour obtenir de meilleurs résultats pratiques dans le domaine de l'inférence non monotone ; c'est ce que nous illustrerons par ce travail.

Dans ce papier, nous présentons une nouvelle approche pour calculer de façon effective des inférences au sein d'une logique non monotone propositionnelle simple mais utile, fondée sur un concept de modèles préférés. L'approche proposée est originale à deux égards au moins. Premièrement, elle se base sur des techniques de recherche locale tout en préservant la complétude logique. En second lieu, elle s'avère efficace d'un point de vue expérimental pour une classe importante de grandes bases de connaissance non monotones. Plus précisément, nous étendons à un cadre non monotone des résultats heuristiques récents liés à la résolution pratique du problème de satisfaisabilité d'un ensemble de clauses (SAT). De fait, la procédure de preuve employée se base sur l'utilisation d'une méthode de recherche locale pour SAT et sur une heuristique efficace lorsque cette recherche locale n'a pu exhiber un modèle. Nous l'employons au sein d'un formalisme de représentation permettant des règles de raisonnement par défaut avec priorités exprimées à l'aide de propositions d'anormalités à la McCarthy (McCarthy, 1986). La simplicité du langage de représentation et la façon par laquelle il doit être utilisé par l'ingénieur de la connaissance constitue un bon compromis entre l'expressivité et la complexité expérimentale au regard d'une classe spécifique d'applications. En effet, bien qu'aucun nouveau résultat de complexité ne soit obtenu dans le pire des cas, l'approche apparaît efficace d'un point de vue expérimental pour une classe importante de grandes bases de connaissance non monotones.

Les bonnes performances expérimentales obtenues sont dues principalement aux caractéristiques suivantes souvent interdépendantes :

1. aux propriétés naturelles des applications spécifiques concernées ;
2. à une utilisation disciplinée du langage de représentation ;

3. à une définition soignée de la structure de la procédure de preuve ;
4. à la faible probabilité de rencontrer un pire cas ;
5. de nouvelles découvertes heuristiques puissantes ;
6. à l'attention qui est portée successivement aux différentes situations possibles, selon leur probabilité décroissante ;
7. au fait que les techniques les plus efficaces sont utilisées pour chaque type de situations.

Ce papier est organisé de la manière suivante. Premièrement, le formalisme non monotone propositionnel choisi est décrit. Puis, nous rappelons brièvement les différentes techniques de recherche locale pour SAT, ainsi que leur utilisation en tant qu'heuristique pour prouver l'inconsistance. Dans une troisième partie, nous présenterons une procédure de preuve originale pour ce formalisme non monotone, étendant ainsi ces résultats obtenus pour SAT. L'attention est portée sur les raisons du bon comportement computationnel de notre procédure. La procédure de preuve, très technique, est donnée en annexe. Nous décrirons des résultats expérimentaux obtenus sur de très grandes bases de connaissance propositionnelles (*BCs*). Enfin, nous discuterons les limites de notre approche et ses possibles extensions.

## 2 Un formalisme non monotone et son domaine cible

Sans de nouveaux résultats de complexité hautement improbables, nous ne pouvons pas espérer trouver une approche tractable pour la plupart des logiques non monotones sans une forme d'approximation ou de restriction du pouvoir expressif du formalisme de représentation des connaissances et/ou du mécanisme de raisonnement, ou encore, sans imposer certaines contraintes sur les applications traitées. Dans ce papier, nous conservons l'intégralité du langage de la logique propositionnelle et intégrons un mécanisme simple de raisonnement à partir de modèles préférés. En conséquence, comme nous ne faisons aucune restriction quant au pouvoir expressif de la logique, nous limitons son utilisation à certaines classes d'applications, lesquelles correspondent cependant à des problèmes réels, que nous pourrions souvent résoudre informatiquement.

Le domaine d'applications que nous visons est le suivant. Nous voulons modéliser des formes d'inférences révisables au départ de *très grandes* bases de connaissance propositionnelles représentant des systèmes ou des appareillages complexes. Nous voulons pouvoir exprimer des règles de

raisonnement par défaut permettant à l'ingénieur de la connaissance de représenter les conditions normales de fonctionnement du système avec des possibilités de pannes (lesquelles conduiraient à l'inconsistance si la représentation était opérée à l'aide de logique standard). L'ingénieur de la connaissance est supposé être capable d'ordonner la probabilité et l'importance des pannes possibles, du moins dans une certaine mesure. Nous supposons également que nous ne rencontrerons pas la loi de Murphy : peu de pannes apparaissent au même moment. Quand plusieurs composants apparaissent défectueux, nous sommes surtout intéressés par les pannes les plus probables et les plus importantes, selon la spécification de l'ingénieur de la connaissance. En effet dans les systèmes réels, lorsque plusieurs pannes apparaissent, de nombreux symptômes de mauvais fonctionnement ne sont souvent que des conséquences d'un simple problème, source de tous les autres. Dans une première analyse il est préférable de se concentrer sur la panne la plus probable et la plus importante et éviter d'être perturbé par ces autres symptômes. Nous pensons que ceci représente une classe réellement importante d'applications pour le raisonnement non monotone. À cet égard, notre approche étend le simple travail de diagnostic dans la mesure où nous voulons être capables de détecter les pannes et de raisonner de manière révisable en leur présence.

Le langage de représentation considéré est la logique propositionnelle standard où la connaissance est supposée être codée en CNF. Le langage est enrichi d'un ensemble fini de propositions (avec priorités) d'anormalité à la McCarthy, noté  $Ab_i$  (McCarthy, 1986), permettant de représenter des règles de raisonnement par défaut avec exceptions. Les propositions d'anormalité sont utilisées pour décrire des défaillances possibles de composants (mais dont les pannes sont inattendues). En accord avec le domaine d'applications cible, on s'attend à ce que seul un petit nombre de propositions d'anormalité doivent être interprétées à *vrai* lorsque la  $BC$  est interrogée, sous peine d'inconsistance de  $BC$ . Par exemple, la règle exprimant que, sous des circonstances normales, quand l'interrupteur est sur la position marche, alors la lampe est allumée, est représentée par la formule  $switch\_on \wedge \neg Ab_1 \Rightarrow light\_on$ , et en forme clausale par  $\neg switch\_on \vee Ab_1 \vee light\_on$ . Soulignons que dans ce cadre très standard, on s'attend à ce que les propositions  $Ab_i$  soient *faux* sous les circonstances opératoires normales. Ceci constitue donc une utilisation très spécifique des variables  $Ab_i$ . Nous ne considérons pas les  $BCs$  incluant beaucoup de règles par défaut où des exceptions possibles existent sous des circonstances normales, comme : « Typiquement les humains sont droitiers ».

Interprétons ce formalisme de représentation des connaissances à l'aide d'un cadre particulier de modèles préférés. D'abord rappelons qu'une *interprétation* assigne des valeurs de  $\{vrai, faux\}$  aux variables propositionnelles du

langage (enrichi). Un *modèle* d'une formule  $f$  est une interprétation sous laquelle la formule  $f$  obtient la valeur de vérité *vrai*. Un modèle sera représenté par l'ensemble des variables propositionnelles satisfaites. Les propositions d'anormalités  $Ab_i$  satisfont les règles de priorités suivantes.  $Ab_i$  est dit *plus petit* que  $Ab_j$  (noté  $Ab_i < Ab_j$ ) lorsque  $i < j$ . L'ingénieur de la connaissance est supposé coder les règles par défaut de telle manière que les pannes de composant les plus probables et les plus importantes soient en rapport avec de petits  $Ab_i$ . En conséquence, nous préférons les modèles qui contiennent le plus petit  $Ab_i$  possible. Soulignons que cette construction peut facilement être étendue à un pré-ordre sur les propositions  $Ab_i$ . Plus formellement, soit  $M_1, M_2, M_3$  des modèles d'une  $BC$ .

- $M_1$  est un modèle préféré de  $BC$  si et seulement si :
  - soit  $M_1$  ne contient aucun  $Ab_i$  ;
  - soit  $M_1$  contient au moins un  $Ab_i$  et :
    1. il n'existe pas de modèle  $M_2$  de  $BC$  ne contenant pas d' $Ab_i$  ;
    2. il n'existe pas de modèle  $M_3$  de  $BC$  contenant un  $Ab_j$  tel que  $Ab_j < Ab_i$  pour tous les  $Ab_i$  de  $M_1$ .

En conséquence, l'inférence non monotone au départ de  $BC$  est définie comme suit. Soit  $f$  un littéral, c'est-à-dire une variable propositionnelle ou sa négation et soit  $BC$  une base de connaissance supposée consistante (cette définition peut facilement être étendue à l'inférence non monotone de clauses ou de formules):

- $BC \models f$  si et seulement si  $f$  est *vrai* dans tous les modèles préférés de  $BC$ .

Introduisons la définition pratique suivante :

- $M_1$  est au moins aussi préférable que  $M_2$  ssi :
  - $M_1$  ne contient pas d' $Ab_i$  ;
  - ou si  $M_2$  ne contient pas d' $Ab_i$  alors  $M_1$  ne contient aucun  $Ab_j$  ;
  - ou si  $M_1$  et  $M_2$  contiennent au moins une proposition d'anormalité alors  $\forall Ab_i \in M_2, \exists Ab_j \in M_1$  tel que  $Ab_j \leq Ab_i$ .

Soulignons que cette définition diffère de l'interprétation sémantique de la plupart des formes de circonscription avec priorités, dans le sens où l'ensemble des modèles est ici partitionné en  $(n + 1)$  classes d'équivalence, où  $n$  est le nombre total de propositions d'anormalité (sommairement, seule la plus petite proposition  $Ab_j$  qui est à *vrai* dans un modèle influe la relation d'ordre ; aucun raffinement n'est fait sur la valeur de vérité des propositions  $Ab_i$  qui ne sont pas les plus petites). Cette spécificité est motivée par la

nature des applications cibles : les pannes les plus importantes et les plus probables doivent être exhibées dans une première analyse ; les autres pannes étant souvent de simples conséquences des premières, elles doivent être considérées dans un deuxième temps et traitées par la suite (sinon elles risqueraient de cacher la panne *réelle*). Par coïncidence, ceci permettra d'obtenir une meilleure efficacité expérimentale.

En conséquence, nous pouvons réécrire la définition de l'inférence non monotone de la façon suivante, laquelle s'avère être plus adéquate à notre dessein.

Soit une bases de connaissance  $BC$  consistante et soit  $f$  un littéral :

- $BC \mid \sim f$  si et seulement s'il existe un modèle  $M_1$  de  $BC$  satisfaisant  $f$  et il n'existe pas de modèle  $M_2$  de  $BC$  qui soit au moins aussi préférable que  $M_1$  et qui ne satisfait pas  $f$ .

### 3 Méthode de recherche locale pour SAT

Nous décrivons ici brièvement le fondement des méthodes de recherche locale, lesquelles ont permis un progrès significatif dans la preuve de la consistance de très grandes et très difficiles instances de SAT. La méthode la plus représentative est certainement GSAT de Selman *et al.* (Selman et al., 1992; Selman et al., 1993) quoiqu'il existe des variantes beaucoup plus efficaces. Cet algorithme est basé sur une recherche locale gloutonne afin de trouver une affectation des variables propositionnelles satisfaisant l'ensemble de clauses. Il commence par générer aléatoirement une affectation des variables et change (« flippe ») l'affectation de la variable qui donne la plus grande augmentation de clauses satisfaites. Ce procédé est répété tant qu'un modèle n'est pas trouvé ou qu'un nombre de changements maximum (Max-Flips), fixé au départ de la procédure, n'est pas atteint. En cas d'échec, la procédure pourra être répétée un nombre de fois fixé à l'avance (Max-Tries) (cf. FIG. 1).

Il faut préciser que des mouvements additionnels (comme par exemple des mouvements aléatoires) sont nécessaires pour s'extraire des extréma locaux. Dans la suite du papier, nous utiliserons une variante appelée TSAT (Mazure et al., 1997b). Celle-ci maintient une liste de « flips » interdits (liste tabou), afin d'éviter des « flips » récurrents et ainsi de s'échapper des extrema locaux ; cet algorithme est extrêmement compétitif. De plus, il permet de supprimer certains caractères stochastiques de GSAT et peut même être rendu complètement déterministe pour autant que les interprétations initiales ne soient pas choisies aléatoirement. Tous les résultats présentés ci-après ont été obtenus avec TSAT cependant ils peuvent être reproduits avec la plupart des algorithmes de recherche locale.

```

Procédure GSAT ;
Entrée : une CNF  $S$ , Max-Flips et Max-Tries ;
Sortie : un modèle de  $S$ , si trouvé.
Begin
  for i := 1 to Max-Tries
    I := affectation aléatoire des
      variables propositionnelles de  $S$  ;
    for j := 1 to Max-Flips
      if I satisfait  $S$  then return I ;
      x := la variable qui, si elle est
        « flippée », augmentera le plus (ou
        diminuera le moins) le nombre de
        clauses satisfaites ;
      I := I avec la valeur de x inversée ;
    end for ;
  end for ;
  return « aucun modèle trouvé » ;
End.

```

FIG. 1: GSAT

### 4 Une heuristique pour localiser des noyaux inconsistants

Dans cette section, nous rappelons un résultat surprenant : malgré leur incomplétude logique qui restreint leur champ d'applications à la seule preuve de la consistance, les algorithmes à la GSAT peuvent être utilisés pour localiser des noyaux inconsistants dans des bases de connaissance propositionnelles.

Le test suivant a été répété intensivement, donnant le même résultat extrêmement souvent (Mazure et al., 1996). TSAT (ou n'importe quelle autre méthode de type GSAT) est exécuté sur une instance de SAT. Le phénomène suivant est rencontré lorsque la méthode échoue dans sa quête d'un modèle. A chaque étape de l'algorithme, c'est-à-dire à chaque « flip », les clauses falsifiées voient leur compteur du nombre de fois où elles ont été contredites mis à jour. Un compteur similaire est maintenu pour chaque littéral, c'est-à-dire, qu'un compteur maintient le nombre de fois où le littéral est apparu dans une clause falsifiée. Intuitivement, il nous semble que si le problème est inconsistant alors les clauses les plus souvent falsifiées doivent constituer un noyau inconsistant du problème. De la même manière, il nous semble que les littéraux possédant les plus forts scores doivent appartenir à ce noyau.

En fait, cette hypothèse s'est avérée expérimentalement très souvent correcte. Le phénomène rencontré peut être résumé comme suit. Informellement, un problème inconsistant localement contient un *petit* noyau inconsistant qu'il suffit de retirer pour rendre l'instance satisfaisable. Quand un algorithme de recherche locale, du type GSAT, opère sur

un problème localement inconsistant, les compteurs précisés permettent de scinder le problème en deux parties : une consistante et une inconsistante. Un écart significatif entre les scores des clauses des deux parties permet de clairement différencier le probable noyau inconsistant du reste du problème. Ainsi, il apparaît que la trace des algorithmes de recherche locale permet de délivrer un probable noyau inconsistant d'une base de connaissance propositionnelle inconsistante localement.

Il faut préciser que la validité de ces résultats dépend de la taille du probable noyau détecté par rapport à la taille initiale du problème. Quand l'instance tend à être globalement inconsistante, c'est-à-dire lorsque la taille du plus petit noyau inconsistant converge vers la taille de l'instance, l'écart de scores n'est pas suffisant pour pouvoir conclure. Cependant beaucoup de bases de connaissance réelles sont localement inconsistantes, puisque l'inconsistante résulte souvent de quelques informations conflictuelles. En particulier, les bases de connaissance que nous considérons ici et qui traduisent le modèle « profond » d'un système complexe satisfont cette propriété la plupart du temps.

Nous avons proposé et expérimenté plusieurs possibilités d'utilisation de l'heuristique ci-dessus (Mazure et al., 1996). La plus directe consiste à utiliser un algorithme à la GSAT pour détecter un noyau probablement inconsistant. Puis, d'utiliser une technique complète comme la procédure de Davis & Putnam (DP) (Davis et Putnam, 1960) pour prouver que ce noyau est inconsistant et par conséquent que  $BC$  est inconsistante. Une autre manière consiste à trier par ordre décroissant de scores les clauses et d'utiliser une méthode complète incrémentalement sur les clauses jusqu'à obtenir un ensemble de clauses inconsistant. À cet égard, les clauses n'intervenant pas réellement dans l'inconsistance pourraient être prises en compte. Plus généralement, les scores délivrés par l'algorithme de recherche locale peuvent être utilisés pour guider la recherche d'une technique complète. Dans (Mazure et al., 1996), nous avons proposé une procédure élémentaire basée sur la combinaison de DP et de cette heuristique ; cette combinaison utilise l'heuristique présenté ci-dessus comme critère de branchement de DP et s'avère extrêmement compétitive.

## 5 Comment l'inférence peut (très souvent) être faite efficacement

Revenons maintenant à notre problème initial. Comment est-il possible d'utiliser les techniques ci-dessus pour faire des inférences dans notre formalisme non monotone et ce d'une manière tractable dans la réalité ?

Pour inférer  $f$  de  $BC$ , nous allons essayer de trouver un modèle  $M_1$  de  $BC$  satisfaisant  $f$  et tel qu'il n'existe pas de modèle de  $BC$  aussi préférable que  $M_1$  et ne satisfaisant

pas  $f$ .

Il est évident, que dans le *pire des cas*, le schéma de modèle préféré a été soigneusement restreint pour requérir au plus  $\mathcal{O}(n)$  tests de satisfaisabilité, où  $n$  représente le nombre total de propositions d'anormalité et *non pas* la taille de la base de connaissance, c'est-à-dire que ce schéma appartient à  $P^{NP[\mathcal{O}(n)]}$ . Cette forme d'inférence non monotone reste au premier niveau de la hiérarchie polynomiale, ce qui n'est pas habituel en logique non monotone. Sommairement, ceci est dû au fait que seule la plus petite proposition  $Ab_j$  qui est *vrai* dans le modèle importe dans la relation entre les modèles ; aucun raffinement n'est fait sur les valeurs des propositions  $Ab_i$  qui ne sont pas les plus petites. Ceci est l'une des clés des bonnes propriétés computationnelles. Mais, dans la plupart des situations *réelles* bien que  $n$  puisse être très grand, nous pouvons observer que, le plus souvent, un petit nombre d'appels (environ 5) à la procédure de test de satisfaisabilité (lesquels s'avèrent souvent extrêmement efficaces) sont suffisants pour décider si oui ou non  $f$  peut être inféré.

Une fois encore, il faut préciser que la restriction sur l'ordre entre les modèles n'est pas artificielle mais correspond au contraire parfaitement à la nature des applications ciblées, certes spécifiques mais importantes. Répétons ici que dans les systèmes ou les composants complexes, seules les sources de mauvais fonctionnement les plus importantes et les plus probables ont effectivement de l'importance pour une première analyse. La connaissance est exprimée de telle manière que les pannes sont traduites en utilisant les plus petites propositions d'anormalité qui doivent être à *vrai*, sans quoi l'inconsistance survient. Les autres symptômes de mauvais fonctionnement utilisant d'autres propositions d'anormalités sont souvent des conséquences et ne devraient pas être considérés directement, ceci afin qu'ils ne cachent pas le problème *réel*.

Par ailleurs, dans les applications de bases de connaissance réelles, quand une inconsistance apparaît, elle est souvent en rapport avec une petite sous-partie de la  $BC$ . Les problèmes réellement difficiles pour vérifier l'inconsistance propositionnelle, semblent être souvent apparentés aux problèmes inconsistants globalement. Ceci se traduit par de bonnes propriétés (expérimentales) computationnelles pour le test de (in)consistance, du moins en ce qui concerne le cadre spécifique d'applications considérées. A savoir, lorsque la  $BC$  est consistante, les méthodes de recherche locale sont souvent très efficaces ; et lorsque la  $BC$  est inconsistante, des techniques complètes à la DP, lorsqu'elles se concentrent sur la trace fournie par la méthode de recherche locale, sont également très souvent efficaces. En conséquence, ces techniques de calcul se comportent (expérimentalement) la plupart du temps de façon polynomiale en la taille de la  $BC$ . En fait, elles sont exponentielles par rapport à la

taille du noyau inconsistant, lequel est très souvent extrêmement petit par rapport la taille globale de la  $BC$ . Bien évidemment, en supposant  $P \neq NP$ , il est possible d'imaginer des pires cas qui seront impossibles à traiter en pratique. Cependant, nous sommes convaincus que la probabilité qu'une telle situation apparaisse dans notre domaine d'applications, est extrêmement faible.

Une troisième clé de bonne efficacité computationnelle pratique est la manière selon laquelle la procédure de preuve est construite. À chaque fois qu'un test de consistance ou d'inconsistance est requis, nous utilisons une méthode de recherche locale et si nécessaire une technique complète à la DP qui se concentrera en premier lieu sur le probable noyau inconsistant mis en lumière par la méthode de recherche locale. En conséquence, les techniques de recherche locale précéderont toujours un appel à une méthode logiquement complète, laquelle bénéficiera du travail effectuée auparavant. Nous traitons en premier lieu les situations les plus probables, c'est-à-dire les situations où il existe des modèles de circonstances normales (Mazure et al., 1997a) (c'est-à-dire des modèles ne contenant aucun  $Ab_i$ ). Ce cas correspond à des conditions normales de fonctionnement du composant. Deux appels à la combinaison de la méthode de recherche locale et de la technique complète doivent être effectués pour tester l'existence de tels modèles pour  $f$  et pour  $\neg f$ , ceci en considérant que tous les  $Ab_i$  sont à *faux*. Les cas de pannes de composants sont adressés dans une seconde étape (normalement, le composant doit fonctionner correctement !). Dans ces cas de pannes, il doit y avoir au moins un  $Ab_i$  à *vrai* pour que la  $BC$  soit consistante. Aussi, dans la mesure où il suffit qu'un nombre limité de propositions d'anormalité soient *vrai* en même temps pour rétablir la consistance, la trace de la recherche locale qui a été obtenue à la première étape nous délivre très souvent l'ensemble correct de propositions d'anormalité qui « restaureront » la consistance lorsqu'elles seront à *vrai* (ces propositions d'anormalité sont en fait celles qui apparaissent le plus souvent dans les clauses falsifiées, lorsque tous les  $Ab_i$  sont supposées être à *faux*). Ceci correspond également à la nature des applications cibles : mise à part la loi de Murphy, un petit nombre de pannes peuvent survenir en même temps. Ce noyau inconsistant nous fournit une liste des propositions d'anormalités  $Ab_j$  qui seront les premières à être testées comme jouant le rôle du plus petit  $Ab_j$  à *vrai* dans l'essai de modèles préférés. Il apparaît expérimentalement que ces  $Ab_i$  sont souvent à *vrai* dans n'importe quel modèle de  $BC$ . Le rôle de l'étape 2 est de se concentrer sur cette liste pour trouver un modèle de  $f$  contenant au moins un des membres de la liste, qui est tel qu'il n'existe aucun modèle préférable ne satisfaisant pas  $f$ .

Un très important résultat heuristique est le suivant. Supposons que nous ne trouvions pas de modèle pour  $BC \wedge f$

(respectivement pour  $BC \wedge \neg f$ ) avec des membres de la liste à *vrai*. Dans ce cas, extrêmement souvent, nous ne devrions pas trouver de modèle avec un autre  $Ab_j$  fixé à *vrai*. Ceci dépend à l'exactitude de la première heuristique délimitant la liste ; si la trace de l'étape 1 nous délivre un sous-ensemble correct de clauses inconsistantes de  $BC$ , alors nous devons nécessairement choisir comme *vrai* au moins une proposition d'anormalité de la liste. Ainsi, lorsqu'un modèle de  $BC \wedge f \wedge Ab_i$  (respectivement  $BC \wedge \neg f \wedge Ab_i$ ) est trouvé, au lieu de  $i$  appels successifs à TSAT et DP pour prouver l'inconsistance de  $BC \wedge f$  (resp.  $\neg f$ )  $\wedge Ab_i$  ( $j \leq i$ ), nous faisons juste un appel à TSAT et à DP sur :  $BC \wedge f$  (resp.  $\neg f$ )  $\wedge (Ab_1 \vee Ab_2 \vee \dots \vee Ab_i)$ . Plus généralement, un tel modèle ne pourra être trouvé et la procédure sera alors terminée. Par ailleurs, choisir le plus petit  $Ab_j$  dans la liste est souvent suffisant, étant donné qu'il représente la panne la plus probable. En conséquence, dans la plupart des cas, la procédure ne nécessitera qu'un très petit nombre constant d'appels aux procédures de vérification de (in)satisfaisabilité. Cependant, la procédure de décision est logiquement complète et doit donc explorer toutes les possibilités si cela est nécessaire. À cet égard, il faut noter qu'à chaque fois qu'un modèle de  $\neg f$  est trouvé avec  $Ab_j$  comme plus petite proposition d'anormalité à *vrai*, nous savons qu'il faut essayer de trouver des modèles de  $f$  qui soient tels que leur plus petit  $Ab_i$  à *vrai* est tel que  $i < j$ .

Notons que la procédure peut être optimisée de différentes manières. Par exemple, des appels successifs à DP peuvent être évités en considérant les sous-arbres des appels précédents. Plus important, une amélioration évidente consiste à délimiter la composante connexe du graphe d'occurrences des variables de  $BC \cup \{f\}$ . En conséquence, seule la sous-partie correspondante de  $BC$  sera considérée dans la procédure de preuve. Dans le papier complet, plusieurs variantes de cette procédure de preuve sont discutées et comparées.

Enfin, notons que cette manière de calculer les inférences n'est pas sensible au fait qu'elle soit définie pour des littéraux. Elle s'étend directement à l'inférence non monotone de clauses.

La procédure de preuve complète, très technique, est donnée en annexe.

## 6 Résultats expérimentaux

Illustrons brièvement l'efficacité expérimentale de la procédure. Premièrement, soulignons qu'expérimenter des procédures de preuve pour les logiques non monotones sur de très grandes  $BC$ s est inhabituel. S'il existe des benchmarks issus du monde réel pour le problème SAT, il n'en existe pas réellement pour les logiques non monotones (nous parlons de plus grands benchmarks que ceux qui existent, lesquels ont été créés pour simplement vérifier si les logiques

offraient un pouvoir expressif adéquat).

En conséquence, nous avons recueilli et généré des benchmarks par nous-mêmes. Ils sont décrits dans le papier complet. En particulier, nous avons généré une classe de benchmarks de telle manière qu'ils « collent » le plus possible aux problèmes ciblés du monde réel. Par exemple, nous avons transformé des benchmarks du monde réel pour SAT pour inclure les propositions d'anormalité, permettant aux règles précédemment encodées de supporter des exceptions.

Dans ce papier, nous relatons un test (obtenu sur un Pentium 166Mhz) qui est une bonne illustration de la taille des benchmarks testés et des résultats computationnels obtenus.

Par exemple, nous avons mélangé 5 grandes  $BC$ s consistantes représentant des problèmes de VLSI ( $ssa-7522-\{156, 157, 157, 159, 160\}$ ) avec une petite instance de SAT inconsistante de la série des Dubois, toutes ces instances provenant de (DIMACS, 1993). La  $BC$  inconsistante résultante contient 1529 clauses (pas nécessairement Horn), construites sur 7041 variables propositionnelles et est intraitable par des techniques standards basées sur DP. Nous avons rétabli la consistance de la  $BC$  en introduisant 1593 propositions  $Ab_i$  différentes de manière aléatoire dans la  $BC$  (cependant, en s'assurant que la  $BC$  redevienne consistante et en utilisant au plus une proposition d'anormalité par clause). Cette nouvelle  $BC$  reste intraitable en utilisant les procédures à base de DP standard; en utilisant TSAT, nous avons prouvé en 4,6 secondes (temps CPU) qu'elle était consistante.

Nous avons choisi  $f$  de façon aléatoire, parmi les littéraux apparaissant dans la  $BC$  initiale et utilisé notre procédure de preuve pour déterminer si oui ou non  $BC \models f$ .

La première étape consiste à vérifier s'il existe des modèles de circonstances normales pour  $BC$  qui satisfont  $f$  ou qui le falsifient. TSAT (avec l'option de « poids », une longueur tabou de 10, 10 « tries » et 10000 « flips » par « try ») fut exécuté sur la  $BC$ , avec tous les  $Ab_i$  mis à *faux* et  $f$  mis à *vrai*, pendant 1m01s30. Il échoue à délivrer un modèle (en effet, il n'existe aucun modèle). La trace fournie par cette étape est donnée en FIG. 2. Un DP sur cette trace permet de prouver, en 0,94 secondes de temps CPU qu'il n'existe aucun modèle de circonstances normales pour  $f$ .

Un travail similaire avec  $f$  mis à *faux*, prenant 59s26 pour TSAT et 0s93 pour DP utilisant sa trace, a été effectué pour vérifier l'absence de modèles de circonstance normales pour  $\neg f$ .

La trace de la FIG. 2 permet d'affirmer que très probablement la proposition d'anormalité  $Ab_4$  doit être mise à *vrai* pour que la  $BC$  redevienne consistante. En effet  $Ab_4$  apparaît dans les clauses les plus souvent falsifiées (pré-

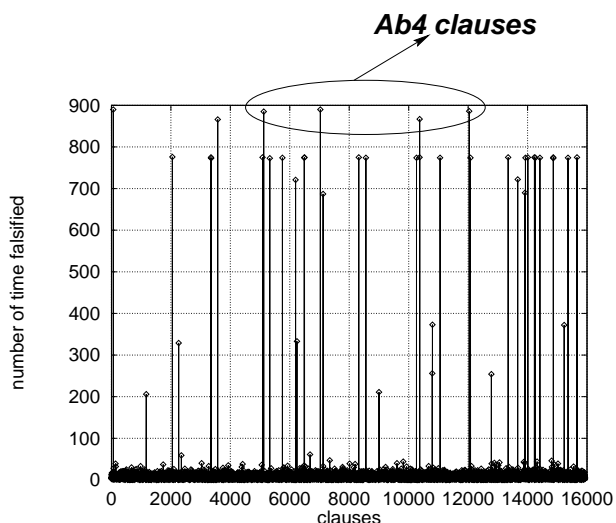


FIG. 2: Trace de TSAT pour  $BC \wedge f \wedge (\neg Ab_1 \wedge \neg Ab_2 \wedge \dots \wedge \neg Ab_{1593})$

cisons que le fait que ce soit un  $Ab_i$  très petit est purement accidentel, et comme discuté ci-dessus, le plus souvent cela n'influencera pas l'efficacité des résultats. Il correspond simplement à une défaillance qui présente une des plus grandes probabilités de survenir dans le système modélisé).

En 39,35 secondes, TSAT permet de trouver un modèle de  $BC \wedge f \wedge Ab_4$ . Puis, nous avons vérifié qu'il n'existe pas de modèle pour  $BC \wedge \neg f \wedge (Ab_1 \vee Ab_2 \vee Ab_3 \vee Ab_4)$ . Ceci est fait en une étape, en ajoutant la clause  $\{Ab_1 \vee Ab_2 \vee Ab_3 \vee Ab_4\}$ , utilisant premièrement TSAT, puis DP pour prouver qu'un tel modèle n'existe pas. TSAT a nécessité 48 secondes et l'appel unique à DP à partir de la trace délivrée par TSAT, a été calculé en moins de 1,8 secondes.

## 7 Conclusions

L'analyse de complexité dans le pire des cas est un outil important pour exhiber les limites computationnelles de formalismes de représentation de connaissance et de formalisation du raisonnement. Cependant l'étendue selon laquelle les situations réelles font figurent de pires cas doit toujours être étudiée. À cet égard, les mauvais résultats de complexité dans le pire des cas pour les formalismes non monotones ne devraient pas être mal interprétés et ne devraient pas *toujours* nous rendre trop pessimistes quant à la possibilité réelle d'implanter ces formalismes, même pour des applications de grandes tailles.

Dans ce contexte, la contribution de ce papier est double. Tout d'abord, on y a décrit une procédure de preuve qui s'avère efficace pour de très grandes bases de connaissance non monotones utilisant le langage complet de la logique

propositionnelle. A notre connaissance, ceci est très inhabituel. En deuxième lieu les ingrédients de cette procédure de preuve sont eux-mêmes inhabituels, puisqu'ils consistent en des mécanismes de recherche locale utilisés de manière à préserver la complétude logique.

Cependant pas de miracle, comme nous ne restreignons pas le pouvoir expressif de la logique, nous limitons son utilisation à une classe spécifique d'applications, laquelle est formée de problèmes réels qui pourront se montrer traitables très souvent. La procédure de preuve est soigneusement construite pour ces applications et pour résoudre les situations les plus courantes d'abord. Les applications considérées nécessitent une utilisation spécifique des propositions d'anormalité. Cependant nous pensons que si les logiques non monotones doivent être utilisées dans les applications à grande échelle, il est acceptable d'exiger de l'ingénieur de la connaissance une utilisation disciplinée d'ingrédients non monotones au pouvoir expressif limité, lors de la construction de la base de connaissance. Également comme le papier l'illustre nous pensons que définir des procédures de preuve orientées applications pour les logiques non monotones peut apparaître comme une alternative viable aux procédures de preuve générales.

Dans le papier complet, nous décrivons d'un point de vue technique les deux principales autres conditions nécessaires pour que notre technique demeure tractable dans la réalité. Principalement chaque fois qu'une inconsistance apparaît dans une *BC*, elle doit être causée par un petit sous-ensemble de celle-ci. Nous pensons que cette propriété est souvent rencontrée dans les bases d'applications réelles que nous considérons, lesquelles décrivent le modèle « profond » de composants physiques complexes. C'est aussi le cas pour notre dernière condition, sous réserve de la loi de Murphy, peu de composants défont en même temps. Dans le papier, nous décrivons comment ceci peut obliger l'ingénieur de la connaissance à respecter des règles supplémentaires d'encodage, ceci sans perte sérieuse d'expressivité.

Finalement, ce papier montre une façon particulière d'importer dans le monde la non-monotonie des progrès récents et importants au sujet de la résolution pratique d'instances de SAT. Beaucoup de formes de raisonnement non monotone reposent sur des tests de consistance, de façon explicite ou implicite. À cet égard, nous sommes convaincus que beaucoup d'autres formalismes non monotones peuvent bénéficier de ces progrès récents au sujet de SAT. Ce papier constitue juste un premier pas dans cette nouvelle direction prometteuse pour l'utilisation de logiques non monotones dans de grandes applications.

Nous croyons aussi qu'après vingt ans de recherche théorique au sujet de la non-monotonie les algorithmes qui apparaissent efficaces pour des très grandes applications sont essentiels au sens où ils peuvent aider à démontrer l'intérêt

pratique de cet important domaine de recherche.

## Remerciements

Ce travail est financé en partie par le projet Ganymède II du plan État/Région Nord-Pas-de-Calais. Nous remercions Pierre Marquis et Lakhdar Saïs pour les discussions stimulantes à propos du contenu de ce papier.

## Références

- CADOLI, M. et SCHAERF, M. (1993). « A survey on complexity results for non-monotonic logics ». *Journal of Logic Programming*, 17:127–160.
- DAVIS, M. et PUTNAM, H. (1960). « A Computing Procedure for Quantification Theory ». *Journal of the Association for Computing Machinery*, 7:201–215.
- DIMACS (1993). « Second Challenge on Satisfiability Testing organized by the Center for Discrete Mathematics and Computer Science of Rutgers University ».
- MAZURE, B., SAÏS, L., et GRÉGOIRE, E. (1996). « Detecting Logical Inconsistencies (Extended Abstract) ». Dans *Proceedings of AI and Maths Symposium*, pages 116–121, . (version étendue à paraître dans "Annals of AI and Maths").
- MAZURE, B., SAÏS, L., et GRÉGOIRE, E. (1997a). « Checking Several Forms of Consistency in Non-Monotonic Knowledge-Bases ». Dans *Proceedings of ECSQARU-FAPR'97*, volume 1244 de *Lecture Notes in Artificial Intelligence*, pages 122–130. Springer.
- MAZURE, B., SAÏS, L., et GRÉGOIRE, E. (1997b). « Tabu Search for SAT ». Dans *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'97)*. 281–285.
- MCCARTHY, J. (1986). « Applications of circumscription to formalizing common-sense knowledge ». *Journal of Artificial Intelligence*, 28:89–116.
- SELMAN, B., KAUTZ, H. A., et COHEN, B. (1993). « Local Search Strategies for Satisfiability Testing ». Dans *Proceedings of the DIMACS Workshop on Maximum Clique, Graph Coloring and Satisfiability*.
- SELMAN, B., KAUTZ, H. A., et MCALLESTER, D. (1997). « Computational Challenges in Propositional Reasoning and Search ». Dans *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, volume 1, pages 50–54.
- SELMAN, B., LEVESQUE, H., et MITCHELL, D. (1992). « A New Method for Solving Hard Satisfiability Problems ». Dans *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI'92)*, pages 440–446.

## Annexe : procédure de preuve

Pour des raisons de place, la procédure TSAT n'est pas décrite dans cette annexe, cependant une description détaillée peut être trouvée dans (Mazure et al., 1997b).

Procédure Propagation\_Unitaire;  
Input : Une  $BC$  exprimée sous CNF ;  
Output : Une base de connaissance tel qu'il n'existe plus de clause unitaire ou qu'il existe une clause vide dans cette base ;  
Begin  
  while (( $\exists$  une clause unitaire dans  $BC$ )  
    and ( $\nexists$  de clause vide dans  $BC$ ))  
    do  
       $l$  := un littéral composant une clause unitaire ;  
       $BC := BC \wedge l$  ;  
    done  
    return  $BC$  ;  
End ;

Procédure DP\_sur\_trace;  
Input : Une  $BC$  exprimée sous CNF, une liste  $T$  de couples (*littéral, score*), correspondant à la trace d'une exécution de TSAT ;  
Output : *Vrai* si  $BC$  est satisfiable, *Faux* sinon ;  
Begin  
   $BC :=$  Propagation\_Unitaire( $BC$ ) ;  
  if ( $\exists$  clause vide  $\in BC$ ) return *Faux* ;  
  elif ( $BC = \emptyset$ ) return *Vrai* ;  
  else  
     $l$  := le littéral ayant le plus fort score dans  $T$ , tel que  $l$  apparaisse dans  $BC$  ;  
    return ( DP\_sur\_trace( $BC \wedge l, T$ )  $\vee$  DP\_sur\_trace( $BC \wedge \neg l, T$ ) ) ;  
  endif  
End ;

Procédure Test\_Satisfaisabilité;  
Input : Une  $BC$  exprimée sous CNF ;  
Output : *Vrai* si  $BC$  est satisfiable, *Faux* sinon ;  
Begin  
  if (TSAT( $BC$ )) return *Vrai* ;  
  else  
    trace := la trace de TSAT ;  
    return (DP\_sur\_trace( $BC$ , trace)) ;  
  end if ;  
End ;

1. A ce niveau, nous sommes assurés qu'il n'existe pas de modèle de circonstances normales

2. Nous savons que cet appel est voué à l'échec, puisque déjà réalisé dans l'étape 1, mais nous avons besoin de sa trace, pour poursuivre notre analyse. Dans la pratique, cet appel n'est pas réalisé, car les scores des propositions d'anormalité sont directement sauvegardés lors du premier appel à TSAT.

Procédure Inférence\_non\_monotone ;  
Input : Une  $BC$  supposée consistante et  $f$  un littéral ;  
Output : *Vrai* si  $BC \models f$ , *Faux* sinon ;  
Begin  
  { **Étape 1** : Recherche de modèles de circonstances normales }  
   $BC' := BC \wedge \{\neg Ab_i\}_{(\forall Ab_i \in BC)} \wedge f$  ;  
  if (Test\_Satisfaisabilité( $BC'$ ))  
    {Il existe un modèle de circonstances normales, qui évidemment est le modèle préféré}  
   $BC' := BC \wedge \{\neg Ab_i\}_{(\forall Ab_i \in BC)} \wedge \neg f$  ;  
  {Pour que  $BC \models f$ , il ne faut pas qu'il existe de modèle de circonstances normales pour  $BC \wedge \neg f$ }  
  return (not(Test\_Satisfaisabilité( $BC'$ ))) ;  
  else  
   $BC' := BC \wedge \{\neg Ab_i\}_{(\forall Ab_i \in BC)} \wedge \neg f$  ;  
  if (Test\_Satisfaisabilité( $BC'$ ))  
    {Il existe un modèle de circonstances normales pour  $BC \wedge \neg f$ , mais il n'existe pas pour  $BC \wedge f$ , par conséquent :  $BC \not\models f$ }  
    return *Faux* ;  
  else  
  { **Étape 2** : Recherche de modèles préférés qui ne sont pas de circonstances normales }<sup>1</sup>  
   $BC' := BC \wedge \{\neg Ab_i\}_{(\forall Ab_i \in BC)} \wedge f$  ;  
  TSAT( $BC$ )<sup>2</sup> ;  
   $L$  := l'ensemble des propositions d'anormalité  $\{Ab_m, \dots, Ab_t\}$  trié en fonction de leur score obtenu à partir de la trace de TSAT ;  
  prouvé := *Faux* ; {*Vrai*, lorsque  $BC \models f$ }  
  while (not prouvé) and ( $L \neq \emptyset$ )  
    do  
      lowest\_Ab <sub>$j$</sub>  := le premier élément de la liste  $L$  ;  
       $L := L - \{\text{lowest\_Ab}_j\}$  ;  
       $BC' := BC \wedge \{\neg Ab_i\}_{(\forall Ab_i < \text{lowest\_Ab}_j)}$  ;  
       $BC' := BC' \wedge \text{lowest\_Ab}_j \wedge f$  ;  
      if (Test\_Satisfaisabilité( $BC'$ ))  
         $c := Ab_1 \vee Ab_2 \vee \dots \vee \text{lowest\_Ab}_j$   
         $BC' := BC \wedge c \wedge \neg f$  ;  
        if (Test\_Satisfaisabilité( $BC'$ ))  
          {Il existe des modèles préférés pour  $f$  et pour  $\neg f$ , incluant lowest\_Ab <sub>$j$</sub> }  
           $Ab_r :=$  le plus petit  $Ab_i$  dans le modèle découvert pour  $\neg f$  ;  
           $L := \{Ab_j \text{ tel que } Ab_j < Ab_r\}$  ;  
        else  
          {Pas de modèle préféré pour  $\neg f$  :  $BC \models f$ .}  
          prouvé := *Vrai* ;  
        endif  
      endif  
    done  
    return prouvé ;  
  end if  
end if  
End ;