Localiser des sources d'incohérence spécifiques sans les calculer toutes

Eric Gregoire, Bertrand Mazure, Cédric Piette

Localiser des sources d'incohérence spécifiques sans les calculer toutes

Éric Grégoire **Bertrand Mazure** **Cédric Piette**

Université Lille-Nord de France, Artois, F-62307 Lens
CRIL, F-62307 Lens
CNRS UMR 8188, F-62307 Lens

{gregoire,mazure,piette}@cril.fr

Résumé

Cet article pose le problème suivant : étant donné un sous-ensemble Γ d'une formule CNF Σ, Γ prend-il part aux sources de l'incohérence de Σ ? Pour répondre à cette question, une approche originale permettant de vérifier si Γ chevauche au moins une sous-formule minimale incohérente (MUS) de Σ est présentée. De plus, si la réponse est positive, un tel MUS est calculé et retourné. Cette technique ré-exprime le problème dans un cadre où les clauses de Σ sont regroupées en *clusters* qui sont manipulés comme les entités atomiques de la formule. Les clusters sont ensuite progressivement affinés selon leur niveau de conflits mutuels, et les plus prometteurs sont sélectionnés et divisés, permettant d'élaguer les clauses inutiles jusqu'à ce que la quantité de ressources allouées soit épuisée ou jusqu'à ce qu'une solution soit retournée. La viabilité et l'utilité de cette approche sont évaluées empiriquement.

1 Introduction

Ces dernières années, de nombreuses études se sont concentrées sur l'explication de l'incohérence d'une instance SAT, fournie à travers des ensembles minimaux de clauses conflictuelles. En effet, bien que certains solveurs (e.g. [5, 1]) puissent fournir une trace de la preuve d'incohérence calculée, celle-ci n'est pas toujours suffisante car elle ne garantit pas que l'ensemble de clauses retourné devienne cohérent avec la suppression de n'importe lequel de ses membres. Par conséquent, plusieurs contributions récentes ont porté sur la localisation de sous-ensembles minimalement incohérents de clauses (MUS pour *Minimal Unsatisfiable Subformula*) d'instances SAT incohérentes ([11, 21], consulter [10] pour une vue d'ensemble récente des différentes techniques). En terme de complexité dans le pire des cas, le calcul des MUS posent plusieurs problèmes majeurs. Tout d'abord, une formule CNF contenant n clauses peut posséder jusqu'à $C_n^{n/2}$ MUS dans le pire cas. Ensuite, vérifier si une formule donnée appartient ou non à l'ensemble des MUS d'une CNF Σ est un problème Σ_2^p-difficile [4]. Toutefois, des approches pour localiser un MUS viables dans de nombreuses circonstances ont été proposées [22, 9]. En outre, dans le but de circonvenir à la possible explosion combinatoire du nombre de MUS d'une formule CNF, des variantes au problème de l'extraction exhaustive des MUS ont été proposée, comme celui de concept de couverture inconsistante, représentant un ensemble de causes minimales et non corrélées de l'incohérence [9]. Enfin, des algorithmes pour calculer l'ensemble complet des MUS de Σ ont également été proposés et se montrent efficaces, au moins pour certains problèmes ne possédant pas un trop grand nombre de MUS.

Quand un utilisateur est face à une instance SAT incohérente Σ, il peut avoir certaines croyances à propos d'un sous-ensemble de clauses Γ potentiellement en cause dans l'un des conflits de Σ. Dans ce cadre, il peut désirer vérifier si ses croyances sont fondées et si Γ participe effectivement à l'incohérence de Σ. En dehors de la situation où $\Sigma \setminus \Gamma$ est satisfiable pour laquelle la réponse est évidente, c'est une requête qui est calculatoirement lourde, tout particulièrement si l'utilisateur désire obtenir une explication sous forme d'un MUS contenant des informations de Γ. En fait, les techniques actuelles ne permettent pas de répondre à une requête comme celle-ci sans calculer l'ensemble des MUS de Σ.

Nous proposons dans ce papier une approche origi-

nale pour répondre à ce problème, sans calculer *tous* les MUS de Σ. Pour circonvenir à la très haute complexité dans le pire cas (du moins, dans une certaine mesure) de cette requête, le problème est exprimé dans un cadre à gros grains où des clusters de clauses de Σ sont formés, puis considérés comme les éléments de base de la formule et examinés selon leur niveau de conflits mutuels. Le cadre est ensuite progressivement raffiné en divisant les clusters les plus prometteurs et en élaguant ceux qui ont été prouvés inutiles jusqu'à ce que la quantité de ressources allouées soit épuisée ou jusqu'à ce qu'une solution soit retournée. Notons que le niveau de conflits mutuels entre clusters est mesuré grâce à une forme de valeur de Shapley [20], célèbre dans la communauté de la théorie des jeux.

Le papier est organisé comme suit. Dans la section suivante, les préliminaires formels sont donnés, ainsi que les définitions de base, certaines propriétés sur les MUS et la définition de la mesure d'incohérence de Shapley. Les principes généraux de l'approche sont décrits en section 3. Le schéma de notre algorithme est détaillé en section 4, tandis que des résultats expérimentaux sont fournis et discutés en section 5. Certains travaux connexes sont ensuite décrits avant de conclure par des pistes de recherche que nous pensons prometteuses.

2 Préliminaires logiques, MUS et mesure de Shapley

2.1 Préliminaires logiques et SAT

Soit $\mathcal{L}$ le langage propositionnel standard, défini inductivement de manière usuelle à partir d'un ensemble P de symboles propositionnels (représentés par des lettres minuscules a, b, c, etc.), les constantes booléennes $\top$ et $\bot$, et les connecteurs standard $\neg$, $\wedge$, $\vee$, $\Rightarrow$ et $\Leftrightarrow$. Une instance SAT est une formule propositionelle sous forme CNF, c'est-à-dire une conjonction (représentée par un ensemble) de clauses, où une clause est une disjonction (représentée par un ensemble) de littéraux, un littéral étant une variable propositionelle ou sa négation. Dans la suite, les lettres grecques majuscules comme Δ, Γ, etc. seront utilisées pour représenter des ensembles de clauses, les lettres grecques minuscules telles que α, β, γ etc. pour représenter des clauses.

SAT est le problème de décision NP-complet qui consiste à vérifier si une CNF est satisfiable, c'est-à-dire si elle admet au moins un modèle. Par la suite, nous utiliserons indifféremment les termes satisfiable (resp. insatisfiable) et cohérent (resp. incohérent). En outre, tout au long de ce papier, nous supposons que Σ est une instance SAT incohérente et Γ est un ensemble de clauses tel que $\Gamma \subset \Sigma$.

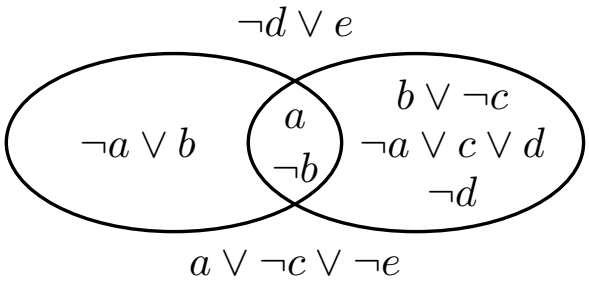

FIG. 1 – MUS de l'Exemple 1.

2.2 MUS d'une instance SAT

Un MUS d'une instance SAT insatisfiable Σ est un sous-ensemble de clauses de Σ qui ne peut être rendu plus petit sans restaurer sa cohérence. Un MUS représente par conséquent une cause minimale de l'incohérence, exprimée sous forme de clauses conflictuelles.

Définition 1 *Un ensemble de clauses Γ est un sous-ensemble minimalement incohérent (MUS pour* Minimally Unsatisfiable Subformula*) de Σ ssi :*

1. $\Gamma \subseteq \Sigma$
2. Γ *est insatisfiable*
3. $\forall \Delta \subset \Gamma$, Δ *est satisfiable.*

L'exemple suivant illustre le fait que deux MUS peuvent se chevaucher l'un l'autre.

Exemple 1 *Soit* $\Sigma = \{\neg d \vee e,\ b \vee \neg c,\ \neg d,\ \neg a \vee b,\ a,\ a \vee \neg c \vee \neg e,\ \neg a \vee c \vee d,\ \neg b\}$. Σ *est insatisfiable et contient 2 MUS* $\{a,\ \neg a \vee b,\ \neg b\}$ *et* $\{b \vee \neg c,\ \neg d,\ a,\ \neg a \vee c \vee d,\ \neg b\}$ *représentés en Figure 1.*

Cet exemple illustre également les différents rôles que peut jouer une clause par rapport à l'incohérence d'une instance SAT Σ. Suivant la catégorisation proposée dans [16], une clause est dite *nécessaire* si celle-ci apparaît dans tous les MUS de Σ. Supprimer une telle clause de Σ en restaure la satisfiabilité. Les clauses *potentiellement nécessaires* de Σ appartiennent à au moins l'un de ses MUS, mais pas à tous. Retirer l'une de ces clauses ne suffit pas à restaurer la cohérence de l'instance SAT, mais elles peuvent devenir nécessaires avec la suppression de clauses appropriées. Les clauses n'appartenant à aucune de ces 2 catégories (c'est-à-dire n'appartenant à aucun MUS) sont appelées *non nécessaires*. Retirer n'importe quel combinaison de ces clauses ne permet en aucun cas de restaurer la satisfiabilité de Σ. Assez

clairement, une instance SAT incohérente ne contient pas toujours de clauses nécessaires, puisque celle-ci peut contenir 2 MUS ne se chevauchant pas.

Exemple 1 (suite) *Les 2 clauses « a » et « $\neg b$ » sont nécessaires par rapport Σ, « $\neg d \vee e$ » et « $a \vee \neg c \vee \neg e$ » sont non nécessaires, tandis que les autres clauses de Σ sont potentiellement nécessaires.*

De nombreuses techniques ont été proposées pour la localisation d'un MUS au sein d'une CNF incohérente [10]. Cependant, ces approches ne peuvent *a priori* pas tenir compte de conditions stipulant que le MUS calculé doit contenir certaines clauses. En conséquence, dans le but de calculer une source de conflits spécifique, la seule solution consiste en l'utilisation d'outils tels que [17, 8] qui retourne l'ensemble complet des MUS de Σ. Notons que les plus efficaces de ces approches extraient les sous-formules maximales satisfiables (MSS pour *Maximal Satisfiable Subsets*) de Σ [17] de la formule comme étape préliminaire, et en dérivent par ensemble intersectant celui des MUS. Ainsi, ces techniques sont difficilement viables en pratique quand des formules de grande taille sont considérées, même si le nombre de MUS de celles-ci est relativement petit par rapport au nombre de clauses de la CNF.

2.3 Mesure d'incohérence de Shapley

Il existe de nombreuses études visant à mesurer les différents niveaux d'incohérence d'un ensemble de formules. Dans la suite de ce papier, nous utiliserons une forme de mesure de Shapley [20] présentée dans [12]. La mesure d'incohérence de Shapley d'une clause α dans une instance SAT Σ fournit un score basé sur le nombre de MUS de Σ contenant α, ainsi que sur la taille de chacun de ces MUS, ce qui permet de tenir également compte de « l'importance » de chaque clause dans chaque source d'incohérence.

Définition 2 *[12] Soient Σ une instance SAT et $\alpha \in \Sigma$. Soit $\cup_{MUS_\Sigma}$ l'ensemble des MUS de Σ. La mesure d'incohérence MI de α dans Σ, notée $MI(\Sigma, \alpha)$, est définie comme :*

$$MI(\Sigma, \alpha) = \sum_{\{\Delta t.q. \Delta \in \cup_{MUS_\Sigma} \ et \ \alpha \in \Delta\}} \frac{1}{|\Delta|}$$

Les propriétés d'une telle mesure sont étudiées dans [12]. L'idée générale est qu'une clause apparaissant dans de nombreux conflits va obtenir un score important, alors qu'une clause contenue dans un « grand » MUS aura un plus faible score qu'une autre clause contenue dans un MUS plus petit.

Exemple 2 *En considérant la CNF Σ de l'Exemple 1, on a :*
- $MI(\Sigma, \neg a \vee b) = \frac{1}{3}$
- $MI(\Sigma, b \vee \neg c) = \frac{1}{5}$
- $MI(\Sigma, \neg d \vee e) = 0$
- $MI(\Sigma, a) = MI(\Sigma, \neg b) = \frac{1}{3} + \frac{1}{5} = \frac{8}{15}$.

Dans la suite de ce papier, cette mesure sera étendue pour caractériser les niveaux de conflits entre des formules conjonctives.

3 Principes généraux de l'approche

3.1 Définition du problème

On dit qu'un ensemble de clauses booléennes Γ *participe effectivement* à l'incohérence d'une instance SAT Σ si Γ contient au moins une clause qui appartient à au moins un MUS de Σ, c'est-à-dire une clause qui est nécessaire ou potentiellement nécessaire par rapport à l'incohérence de Σ. Dans ce cas, notre but est de fournir l'un des MUS de Σ chevauchant Γ. Plus formellement :

Définition 3 *Soient $\cup_{MUS_\Sigma}$ l'ensemble des MUS d'un ensemble de clauses booléennes Σ et Γ un sous-ensemble de Σ. Γ participe à l'incohérence de Σ ssi $\exists \alpha \in \Gamma, \exists \Delta \in \cup_{MUS_\Sigma}$ t.q. $\alpha \in \Delta$.*

De manière évidente, si Γ contient au moins une clause nécessaire (par rapport à Σ), alors elle participe effectivement à son incohérence. Dans ce cas spécifique, vérifier sa participation ne nécessite au plus que k appels à un solveur SAT, où k est le nombre de clauses de Γ, puisqu'il existe $\alpha \in \Sigma \cap \Gamma$ t.q. $\Sigma \setminus \{\alpha\}$ soit satisfiable. De plus, dans ce cas tout MUS de Σ chevauche Γ et fournit une solution à notre problème. Cependant, dans le cas général, Γ ne contient pas nécessairement une telle clause, rendant le problème du calcul d'un MUS contenant l'une de ces clauses particulières plus difficile.

3.2 Utilisation des clusters pour simplifier le problème

Pour circonvenir à la forte complexité du problème de l'extraction d'un MUS spécifique, nous avons recours à une stratégie consistant à partitioner Σ en un nombre prédéfini de m sous-ensembles, appelés clusters, comme étape préliminaire.

Définition 4 *Soit Σ une formule CNF. Un m-*clustering de clauses Π *de Σ est une partition de Σ en m sous-ensembles $\Pi_j (j \in [1..m])$. Π_j est appelé le $j^{ème}$ cluster de Σ (par rapport au clustering Π de Σ).*

Pour des raisons pratiques, nous utiliserons la notation Π_j pour représenter à la fois l'ensemble des clauses formant le $j^{ème}$ cluster de Σ et la formule faite de la conjonction de ces clauses. De plus, l'union ensembliste de clusters représentera la conjonction des clauses qu'ils contiennent.

Les clauses contenues au sein d'un même cluster sont considérées conjonctivement pour devenir une formule vue comme une entité indivisible au sein de Σ. Les clauses de Γ sont traitées de la même manière. Le problème initial est donc élevé à un niveau d'abstraction moins fin, consistant à tester la façon dont la formule Γ prend place dans l'incohérence de Σ, maintenant composée de m sous-formules atomiques.

3.3 Des MUS aux MUSC

Le problème que nous traitons ici est donc relégué à un cadre où l'incohérence est analysée à travers des clusters de clauses. Le concept de MUS doit donc être adapté en conséquence, donnant naissance au concept d'ensemble minimalement incohérent de clusters pour un clustering Π (noté $MUSC_\Pi$ pour *Minimally Unsatisfiable Set of Clusters*) de Σ.

Définition 5 *Soient Σ une formule CNF et Π un m-clustering de clauses de Σ. Un* ensemble minimalement incohérent de clusters *pour Π (noté $MUSC_\Pi$) M_Π de Σ est un ensemble de clusters t.q. :*

1. $M_\Pi \subseteq \Pi$
2. M_Π *est insatisfiable*
3. $\forall \Phi \in M_\Pi$, $M_\Pi \setminus \{\Phi\}$ *est satisfiable*

L'ensemble des $MUSC_\Pi$ de Σ est noté $\cup_{MUSC_\Pi}$.

Le concept « classique » de MUS est une instanciation particulière de celui de MUSC, où chaque cluster est en fait une simple clause. Par conséquent, les MUS représentent le niveau d'abstraction le plus fin pour expliquer l'incohérence de Σ, tandis que celui de MUSC permet la découverte d'explication de l'incohérence à gros grains.

Considérons les clauses dont la participation à l'incohérence d'une instance SAT Σ veut être établie. Quand de telles clauses apparaissent dans un $MUSC_\Pi$ de Σ (pour n'importe quel clustering Π), elles participent également à l'incohérence de Σ quand le problème est traité à des niveaux d'abstraction inférieurs, c'est-à-dire au niveau des clauses. Plus formellement :

Proposition 1 *Soient Σ une instance SAT, Π un m-clustering de Σ et M_Π un $MUSC_\Pi$ de Σ. Si $\Pi_i \in M_\Pi$, alors $\exists \alpha \in \Pi_i, \exists \Psi \in \cup_{MUS_\Sigma}$ t.q. $\alpha \in \Psi$.*

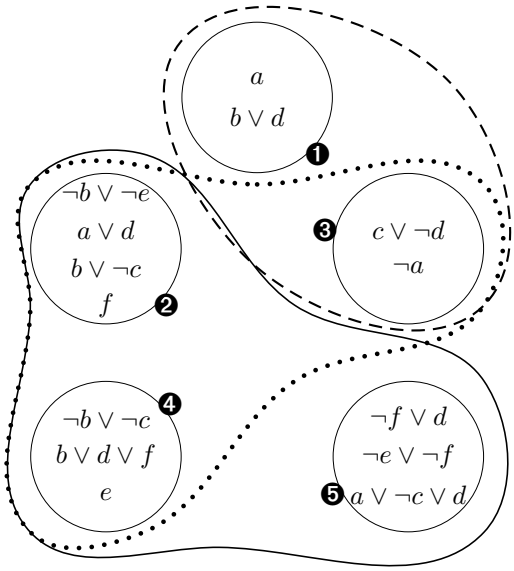

FIG. 2 – MUS de la CNF Σ de l'Exemple 3.

Une telle transformation de ce problème possède des caractéristiques intéressantes, d'un point de vue calculatoire. Celles-ci seront exploitées dans notre approche. Cependant il n'y a pas de miracle : comme le problème est une simplification du problème initial, certaines sources d'incohérence peuvent disparaître. Toutefois, cette simplification reste tout à fait compatible avec notre but initial et un algorithme correct et complet (dans le sens où il retourne un MUS de Σ chevauchant Γ s'il en existe un) est proposé.

Exemple 3 *Soit Σ une instance SAT composée de 14 clauses, telle qu'illustrée en Figure 2. Ces 14 clauses forment 9 MUS. Par exemple, prouver que les clauses « $\neg a$ » et « $c \vee \neg d$ » participent effectivement à l'incohérence de Σ nécessite dans le pire cas l'extraction de l'ensemble exhaustif des MUS de cette CNF. Regroupons maintenant les clauses en 5 clusters, de la façon dépeinte en Figure 2. À ce niveau d'abstraction, on ne compte que 3 $MUSC_\Pi$, qui sont $M_1 = \{❶,❸\}$, $M_2 = \{❷,❸,❹\}$ et $M_3 = \{❷,❹,❺\}$, en utilisant les étiquettes des clusters de la Figure. Ce clustering limite donc le nombre d'entités en contradiction tout en prouvant qu'au moins l'une des clauses « $\neg a$ » ou « $c \vee \neg d$ » appartient à un MUS de Σ.*

Cet exemple illustre l'intérêt de grouper les clauses en clusters, permettant d'éviter le calcul de *tous* les MUS pour répondre à la requête de l'existence d'un MUS spécifique. Cependant, si certaines clauses groupées en un certain clustering n'apparaissent dans aucun MUSC, ceci ne signifie pas nécessairement qu'elles ne participent pas à l'incohérence de la CNF au niveau des clauses.

Proposition 2 *Soient Σ une instance SAT et Π un n-clustering de Σ. Si $\exists \Pi_i \in \Pi, \exists$ une clause $\alpha \in \Pi_i, \exists$ un MUS $\Psi \in MUS_\Sigma$ t.q. $\alpha \in \Psi$, alors Π_i n'apparaît pas nécessairement dans un $MUSC_\Pi$ de Σ.*

Exemple 4 *Soit $\Sigma = \{\neg c, \neg a, b \vee c, a, \neg b\}$. Σ possède 2 MUS : $MUS^1_\Sigma = \{\neg a, a\}$ et $MUS^2_\Sigma = \{\neg b, b \vee c, \neg c\}$. Considérons maintenant le 3-clustering Π de Σ t.q. $\Pi_1 = \{\neg b, a\}$, $\Pi_2 = \{\neg a, b \vee c\}$ et $\Pi_3 = \{\neg c\}$. Nous obtenons :*

	Π_1	Π_2	Π_3
MUS^1_Σ	a	$\neg a$	
MUS^2_Σ	$\neg b$	$b \vee c$	$\neg c$

À ce niveau d'abstraction, le seul $MUSC_\Pi$ est $\{\Pi_1, \Pi_2\}$, capturant MUS^1_Σ. Cependant, MUS^2_Σ n'est pas représenté dans cet unique $MUSC_\Pi$, puisque ses clauses sont divisées dans chaque cluster de Π. En particulier, ce clustering ne permet pas de conclure quant à la participation effective de « $\neg c$ » dans l'incohérence de Σ.

Il existe donc un phénomène de « subsumption » entre les causes de l'incohérence quand les clauses sont regroupées en clusters. D'un côté, le nombre de sources d'incohérence de Σ décroît en conséquence (ce qui est notre but) et une approche adaptée peut tirer parti des MUSC pour se focaliser sur les plus prometteurs. D'un autre côté, il faut s'assurer que l'information qui nous intéresse sur l'incohérence de Σ ne soit pas cachée par le clustering, rendant notre but de calculer un MUS spécifique impossible à atteindre, bien qu'un tel MUS existe. Dans une certaine mesure, cet exemple illustre également que la façon dont les clusters sont formés (et les MUS disséminés en leur sein) influe de manière cruciale sur l'efficacité d'une telle approche.

3.4 Mesurer les conflits grâce aux MUSC

La mesure d'incohérence de Shapley décrite en section 2.3 nécessite d'être adaptée aux clusters de clauses, maintenant considérés comme éléments atomiques d'une instance SAT.

Définition 6 *Soient Σ une instance SAT et Π un m-clustering de Σ. La mesure d'incohérence MI d'un cluster Π_i de Π est définie comme suit :*

$$MI(\Pi_i) = \sum_{\{M | M \in \cup_{MUSC_\Pi} \ et \ \Pi_i \in M\}} \frac{1}{|M|}$$

Une fois l'ensemble des MUSC extrait, cette mesure peut être obtenue en temps polynomial.

Exemple 5 *Soient Σ l'instance SAT et Π le clustering de l'Exemple 3. On a :*

- $MI(\Pi_1) = 1/2$
- $MI(\Pi_2) = MI(\Pi_4) = 2 \times 1/3 = 2/3$
- $MI(\Pi_3) = 1/2 + 1/3 = 5/6$
- $MI(\Pi_5) = 1/3$

Le cluster possédant le score le plus important est Π_3. Ceci est en partie dû au fait que Π_3 appartient à 2 $MUSC_\Pi$. Π_2 et Π_4 appartiennent également à 2 $MUSC_\Pi$, mais chacun d'eux ne représente que le tiers des 2 sources d'incohérence, alors que Π_3 est impliqué dans un $MUSC_\Pi$ composé de seulement 2 clusters. Le rôle de Π_3 est donc important dans ce conflit, et son score est pondéré en conséquence.

3.5 Initialiser le processus

Dans le but de former un clustering initial, une heuristique peu coûteuse en ressources est utilisée. Celle-ci fournit un score à chaque clause de Σ en fonction de sa probabilité d'appartenance à au moins un MUS de la CNF. Cette heuristique est basée sur une recherche locale et le concept de *clause critique* qui permet de prendre en compte un voisinage partiel pertinent de chaque interprétation parcourue [9].

Σ est divisé en m clusters de la manière suivante. Les $\frac{|\Sigma \setminus \Gamma|}{m}$ clauses de $\Sigma \setminus \Gamma$ qui ont les scores les plus importants prennent (probablement) part à un grand nombre de conflits et leur présence dans Σ peut être la cause de nombreux MUS. Celles-ci sont donc rassemblées pour former un cluster Π_1. Ensuite, les autres clauses de $\Sigma \setminus \Gamma$ sont également triées par rapport à leur score et groupées de la même manière, pour obtenir les $m-1$ clusters de Π et fournir un m-clustering de $\Sigma \setminus \Gamma$. Les clauses de Γ sont alors ajoutées pour avoir un $(m+1)^{ème}$ cluster, noté Π_Γ. En conséquence, un $m+1$-clustering de Σ est obtenu.

3.6 Analyser les conflits au niveau des clusters

Chaque cluster est donc interprété comme une formule conjonctive et l'intéraction entre ces entités est analysée en terme (d'ensembles) de clusters mutuellement contradictoires. Cette analyse de conflits a pour but d'élaguer le problème en rejetant les clusters qui ne sont pas contradictoires avec Γ, et en se concentrant sur ceux qui sont au contraire les plus en conflit avec Γ. L'ensemble exhaustif des MUSC de ce clustering (i.e. $\cup_{MUSC_\Pi}$) est donc calculé et on est alors face à l'une des trois situations suivantes :

1. *Incohérence globale :* une forme d'incohérence globale survient au niveau des clusters quand pour chaque Π_i ($i \in [1..m]$) du m-clustering Π, $\Pi \setminus \Pi_i$ est cohérent. Une telle situation se produit quand au moins une clause de chaque cluster est nécessaire pour former un conflit. Le clustering ne

fournit alors aucune information réellement pertinente. Le paramètre m (représentant le nombre de clusters) est alors incrémenté pour obtenir un clustering plus fin. Notons que cette situation ne requiert pas énormémement de ressources pour être vérifiée, puisque que ce clustering ne contient qu'un seul MUSC et un nombre linéraire d'ensembles satisfiables de clusters.

2. Γ *apparaît dans au moins conflit :* certains conflits locaux ont été détectés entre les clusters, et au moins l'un d'entre eux contient Π_Γ. Plus précisément, $\exists M \in \cup_{MUSC_\Pi}$ t.q. $\Pi_\Gamma \in M$. Dans ce cas, l'idée est de se concentrer sur un seul MUSC puisqu'il implique que certaines clauses de Γ sont nécessairement en conflit avec d'autres, contenues dans les clusters de ce MUSC. En conséquence, l'ensemble courant de clauses peut être élagué en supprimant toutes les clauses n'apparaissant pas dans ce MUSC particulier contenant Π_Γ.

3. Γ *n'apparaît dans aucun conflit :* certains conflits entre clusters ont été identifiés mais aucun d'entre eux ne comprend le cluster composé des clauses de Γ. Plus précisément, $\nexists M \in \cup_{MUSC_\Pi}$ t.q. $\Pi_\Gamma \in M$. Dans une telle situation, un raffinement apparaît nécessaire. Plutôt qu'un redécoupage général (cf. situation d'incohérence globale), nous pouvons ici bénéficier d'informations importantes sur les sources de conflits. En effet, les sources de conflits étant ici locales, il nous est possible de sélectionner les clusters les *plus conflictuels* en utilisant la mesure de Shapley décrite précédemment. Ainsi, un nombre prédéterminé de clusters ayant les scores de Shapley les plus importants sont divisés, selon l'intuition que ceux-ci cachent de nombreuses autres sources d'incohérence, dans l'espoir d'en découvrir une impliquant Γ.

3.7 Itération et fin du processus

Le clustering est donc modifié selon l'une des 3 situations présentées ci-dessus, permettant d'augmenter le nombre de clusters, les diviser et se concentrer sur les sous-ensembles de Σ les plus prometteurs. Comme ce processus est itéré, nous obtenons en temps fini un $|\Sigma|$-clustering (chaque cluster est en fait une seule clause), correspondant au calcul « classique » de MUS, sur un nombre réduit de clauses de Σ. En pratique, quand la taille du MUSC comprenant Γ devient de taille raisonnable pour envisager un calcul exhaustif, celui-ci est effectué. L'un des attraits de cette approche est son caractère *any-time* : après qu'une certaine quantité de ressources soit épuisée, celle-ci peut fournir la solution courante, c'est-à-dire la sous-formule de Σ courante, en conflict avec Γ.

La technique que nous proposons, basée sur ces idées, est décrite dans la section suivante et est illustrée dans l'algorithme 1.

4 Algorithme principal

Dans cette section, nous décrivons l'algorithme principal de l'approche proposée. Pour des raisons pratiques, certains cas spécifiques gérés dans notre implémentation (par exemple les situations où Γ est incohérent ou Σ est cohérent) ne sont pas décrites. Notre présentation n'inclut pas non plus le prétraitement syntaxique qui consiste à supprimer les occurences de clauses identiques dans Σ, sauf une (celle appartenant à Γ si une telle clause existe), permettant de réduire exponentiellement le nombre de MUS de la CNF. Il est important de traiter de telles situations pour éviter des cas pathologiques, mais celles-ci sont simples à implémenter et ne présentent que peu d'intérêt algorithmique. L'algorithme se réfère également à la technique `HYCAM` [8] qui calcule l'ensemble des MUS d'une instance SAT. En pratique, une nouvelle version de `HYCAM` capable de gérer les clusters de clauses a été implémentée et utilisée dans les études empiriques.

Détaillons les différentes étapes de l'approche décrite dans l'Algorithme 1. Cette technique prend en entrée une CNF Σ et l'un de ses sous-ensembles Γ. De plus, elle nécessite l'entrée de 3 paramètres entiers qui sont m, inc et s.

Tout d'abord, une recherche locale est exécutée pour fournir un score heuristique aux clauses suivant la contrainte que chacune exerce au sein de Σ. Plus précisément, les scores de chaque clause sont tout d'abord mis à 0. Durant le processus de recherche locale, les scores des clauses *critiques* [9] par rapport à l'interprétation courante sont incrémentés (cf. fonction `scoreClause` pour plus de détails).

Les clauses sont ensuite groupées en un clustering Π (ligne 3), via l'appel de la fonction `clusterClauses`. Cette fonction consiste à faire la conjonction des clauses ayant les scores les plus importants dans un premier cluster, les clauses restantes ayant les scores maximum dans un deuxième cluster, etc. jusqu'à ce que m clusters soient formés. Puis, la version modifiée de `HYCAM`, notée `HYCAM*` (ligne 4) est appelée pour calculer l'ensemble des MUSC de Π.

À présent, l'ensemble $\cup_{MUSC_\Pi}$ a été calculé et plusieurs cas sont à considérer. Soit Π est globalement incohérent (lignes 9-10) : à ce niveau, aucune information ne peut être extraite pour trouver un sous-ensemble de clauses conflictuelles impliquant Γ, le clustering est donc raffiné en ajoutant inc clusters supplémentaires (dans l'algorithme, ceci est effectué à travers un appel récursif à `look4MUS`). Dans le cas contraire, nous

Algorithme 1 : L'algorithme `look4MUS`.

Données :
Σ : une CNF
Γ : une CNF t.q. $\Gamma \subset \Sigma$
m : taille du clustering initial
inc : incrément du clustering (incohérence globale)
s : nombre de clusters à diviser (incohérence locale)
Résultat : Un MUS de Σ chevauchant Γ si un tel MUS existe, sinon $\emptyset$

```
1  début
2      S_Σ ⟵ scoreClauses(Σ) ;
3      Π ⟵ clusterClauses(Σ \ Γ, S_Σ, m) ∪{Γ} ;
4      ∪MUSC_Π ⟵ HYCAM*(Π) ;
5      si Π est un |Σ|-clustering alors
6          si ∃M ∈ ∪MUSC_Π t.q. Γ ⊆ M alors retourner M ;
7          sinon retourner ∅ ;
8      sinon
9          si ∀Π_i ∈ Π, Σ \ Π_i est satisfiable alors
               // incohérence globale
10             retourner look4MUS(Σ, Γ, m + inc, inc, s) ;
11         sinon
12             tant que ∄M ∈ ∪MUSC_Π t.q. Γ ∈ M faire
                   // incohérence locale
13                 Π' ⟵ ∅ ;
14                 pour j ∈ [1..s] faire
15                     Π_best ⟵ ∅ ;
16                     pour chaque Π_i ∈ Π t.q. Π ≠ Γ faire
17                         si MI(Π_i,∪MUSC_Π)>MI(Π_best,∪MUSC_Π) alors Π_best ⟵ Π_i ;
18                     Π ⟵ Π \ Π_best ;
19                     Π' ⟵ Π' ∪ clusterClauses(Π_best, S_Σ, 2) ;
20                 Π ⟵ Π ∪ Π' ;
21                 ∪MUSC_Π ⟵ HYCAM*(Π) ;
22             M_Γ ⟵ selectOneMUSC(∪MUSC_Π,Γ) ;
23             retourner look4MUS(M_Γ, Γ, m, inc, s) ;
24 fin
```

sommes en présence d'une incohérence locale. Tant que Γ n'est impliqué dans aucun $MUSC_\Pi$ (lignes 12-21), les s clusters les plus conflictuels (par rapport à la mesure d'incohérence de Shapley de la fonction `MI`) sont divisés en deux parties dans le but de révèler une source d'incohérence impliquant Γ. Quand de telles incohérences locales sont découvertes, un MUSC M_Γ contenant Γ est sélectionné (fonction `selectOneMUSC`, qui retourne le MUSC contenant le moins de clauses possible). Le processus continue en ne considérant que la CNF M_Γ, qui est une sous-formule de Σ qui contient un MUS impliquant Γ.

À moins que le temps de calcul imparti soit épuisé, l'algorithme ne peut stopper qu'aux lignes 6 et 7, quand chaque cluster est en fait une simple clause ($|\Sigma|$-clustering). À ce point, un appel « classique » à `HYCAM` est effectué, fournissant des MUS (les « MUSC » ligne 6 sont en fait des MUS puisque chaque cluster est une clause) contenant de l'information de Γ, ou prouvant que de tels MUS n'existe pas. Dans notre implémentation, nous n'attendons pas nécessairement qu'un $|\Sigma|$-clustering soit atteint pour utiliser la procédure classique ; quand le nombre de clusters est suffisamment proche du nombre de clauses (i.e. quand $m < 2 \times |\Sigma|$), alors la procédure se termine par un appel à `HYCAM`. De plus, l'implémentation possède une caractéristique *any-time*, dans le sens où elle fournit la dernière sous-formule calculée quand la quantité de ressources allouées à cette tâche est épuisée.

Fonction `scoreClauses`

Données : Σ : une CNF
Résultat : Un vecteur de scores pour chaque clause de Σ

début
  pour chaque $c \in \Sigma$ **faire**
    $S_\Sigma(c) \longleftarrow 0$;
  $I \longleftarrow$ une affectation aléatoire pour chaque variable de Σ ;
  tant que *le nombre de flip imparti n'est pas dépassé* **faire**
    pour chaque $c \in \Sigma$ **faire**
      si *c est critique p/r à I dans* Σ **alors**
        $S_\Sigma(c)++$;
    $I \longleftarrow I'$ t.q. I et I' diffèrent d'une seule valeur de vérité ;
  retourner S_Σ ;
fin

Fonction `clusterClauses`

Données : Σ : une CNF, S_Σ : un vecteur de scores, m : la taille du clustering
Résultat : Un m-clustering de Σ

début
  $\Pi \longleftarrow \emptyset$;
  pour $i \in [1..m-1]$ **faire**
    $\Pi_i \longleftarrow$ les $(\frac{|\Sigma|}{m})$ clauses de Σ ayant les plus hauts scores ;
    $\Pi \longleftarrow \Pi \cup \{\Pi_i\}$;
    $\Sigma \longleftarrow \Sigma \setminus \Pi_i$;
  $\Pi \longleftarrow \Pi \cup \{\Sigma\}$;
  retourner Π ;
fin

Fonction `selectOneMUSC`

Données : $\cup_{MUSC_\Pi}$: l'ensemble des MUSC p/r Π, Γ : une CNF
Résultat : Un MUSC de $\cup_{MUSC_\Pi}$ contenant Γ

début
  $M_\Gamma \longleftarrow \Pi$;
  pour chaque M *t.q.* $M \in \cup_{MUSC_\Pi}$ **faire**
    si $\Gamma \subseteq M$ *et* $|M| < |M_\Gamma|$ **alors**
      $M_\Gamma \longleftarrow M$;
  retourner M_Γ ;
fin

Fonction `MI`

Données : Π_i : un élément du clustering Π, $\cup_{MUSC_\Pi}$: l'ensemble des $MUSC_\Pi$
Résultat : La mesure d'incohérence de Π_i p/r Π

début
  $mi \longleftarrow 0$;
  pour chaque $MUSC_\Pi \in \cup_{MUSC_\Pi}$ **faire**
    si $\Pi_i \in MUSC_\Pi$ **alors**
      $mi \longleftarrow mi + \frac{1}{|MUSC_\Pi|}$;
  retourner mi ;
fin

5 Résultats expérimentaux

Une implémentation en `C` de l'algorithme présenté a été réalisée. Comme cas d'étude, les valeurs suivantes ont été sélectionnées comme paramètres $m = 30$, $inc = 10$ et $s = \frac{m}{10}$. Cette implémentation a été exécutée sur de nombreux benchmarks provenant de `http://www.satcompetition.org`. Toutes nos expérimentations ont été conduites sur des processeurs Intel Xeon 3GHz sous Linux CentOS 4.1. (kernel 2.6.9) avec une limite de mémoire RAM de 2 Go.

Pour chaque instance SAT testée, nous essayons de prouver que les clauses apparaissant en $1^{ère}$, $3^{ème}$, $5^{ème}$ et $7^{ème}$ position de son fichier au format DIMACS participent effectivement à l'incohérence de la CNF, et d'extraire un MUS contenant au moins l'une de ces clauses. Un échantillon des résultats obtenus est reporté en Table 1 avec pour chaque instance, la taille en nombre de clauses du MUS extrait ($\#\text{cla}_{MUS}$) et le temps en secondes. Ces résultats permettent de constater que l'approche est capable de localiser des MUS spécifiques pour des CNF de taille conséquente. Notons de plus que l'ensemble exhaustif des MSS (et donc des MUS) ne peut être extrait en temps raisonnable ni par `CAMUS` [17] ni par `HYCAM` [8], pour la plupart de ces problèmes. Aucune approche jusqu'à présent ne permet donc d'extraire une source d'incohérence contenant exactement telle ou telle information. Notre technique permet au contraire de répondre à cette question de la participation de clauses spécifiques dans l'incohérence des formules testées.

Grouper les clauses en cluster dans le but de « dissimuler » des sources d'incohérence, et se concentrer sur les clusters impliquant les clauses désirées se montre particulièrement viable en pratique. En effet, cette stratégie permet de réduire de manière drastique l'effort calculatoire en limitant l'explosion combinatoire du nombre de sources d'incohérence. Par exemple, les problèmes `dp05u04` et `dp06u05` (qui encodent le célèbre problème du dîner des philosophes dans le cadre du *bounded model checking*) possède un si grand nombre de MUS qu'il est impossible de les calculer

nom	#var	#cla	#cla$_{MUS}$	temps
1dlx_...bp_f	776	3725	1440	705
bf0432-007	1040	3668	1223	119
bf1355-075	2180	6778	151	22
bf1355-638	2177	6768	154	21
bf2670-001	1393	3434	134	8
dp05u04	1571	3902	820	1254
dp06u05	2359	6053	1105	2508
ezfact16_1	193	1113	319	28
ezfact16_2	193	1113	365	43
ezfact16_3	193	1113	297	31
ezfact16_10	193	1113	415	54

TAB. 1 – Extraire un MUS of Σ chevauchant Γ : quelques résultats expérimentaux

tous (voire même de les lister) avec la technologie actuelle. `look4MUS` réussit au contraire à extraire l'une des sources d'incohérence voulues – sans les extraire toutes – en un temps de calcul très raisonnable (environ respectivement 20 et 42 minutes). La situation est similaire pour les autres CNF. Sur la famille d'instances `ezfact16_*` (factorisation de circuits), une explication impliquant des clauses particulières peut être extraite en moins d'une 1 minute alors que les approches exhaustives montrent leurs limites sur de tels problèmes. Néanmoins, les problèmes considérés dans la Table 1 sont loin des problèmes de très grande taille (plusieurs dizaines de milliers de clauses) que peuvent résoudre les meilleurs solveurs SAT actuels. Notons tout de même que sur certains de ces problèmes difficiles, notre technique est parvenue à itérer plusieurs fois, permettant de retourner en temps raisonnable une sous-formule conflictuelle avec Γ, grâce à son caractère *any-time*.

6 Travaux connexes

Les approches existantes pour extraire un MUS ou en faire l'approximation ont été décrites dans les sections précédentes. Soulignons ici que le travail présenté dans ce papier induit une forme d'*abstraction* pour réduire le coût calculatoire du problème initial. De plus, via l'aspect *any-time* de cette technique, elle peut être utilisée comme une stratégie d'*approximation*. De la même manière, un grand nombre de techniques d'abstraction et d'approximation ont été proposées pour traiter divers problèmes liés à la logique propositionelle. Nous présentons ici quelques unes d'entre elles.

Le papier séminal de l'approximation sous ressources limitées de l'inférence en logique propositionelle est [19]. Une autre approximation, où chaque étape de calcul peut être décidée en temps polynomial, a été proposée dans le cadre clausal dans [2, 3], et plus tard étendue à l'ensemble de la logique classique dans [6]. Pour un survol des techniques d'approximation, nous renvoyons le lecteur à [7]. Notons enfin que l'approximation de raisonnement dans les systèmes à base de connaissances a également été un sujet très étudié (voir par exemple [14, 13]).

Pour finir, l'idée de grouper les clauses en clusters a déjà été utilisée dans le contexte de contraintes de haut niveau codée sous forme de clauses [17]. De plus, il apparaît que les cas où aucune clause de Γ ne participe à l'incohérence sont les pires pour notre approche. Toutefois, si Γ est composée exclusivement de clauses non nécessaires, alors cette sous-formule peut être satisfaite par une *autarky* [15], qui est définie comme une interprétation partielle qui satisfait toutes les clauses ayant au moins un littéral affecté. Un algorithme a récemment été proposé pour calculer des autarkys en modifiant la CNF et en considérant un problème d'optimisation [18].

7 Conclusions et travaux futurs

Dans ce papier, une technique originale permettant de vérifier si un ensemble de clauses chevauche au moins une sous-formule minimale incohérente d'une formule CNF a été décrite. Un tel problème peut se montrer très utile quand l'utilisateur désire restaurer la cohérence d'une telle formule et possède (ou peut obtenir) certaines informations à propos des clauses en conflit de la CNF.

L'approche proposée se montre plus efficace que les approches existantes de plusieurs ordres de grandeur, grâce à son paradigme d'abstraction et de reformulation. Elle ne nécessite en particulier pas l'extraction de *tous* les MUS de l'instance. Un autre attrait de la technique est son caractère *any-time*, permettant de produire des solutions plus grossières si les ressources qui lui sont allouées sont limitées.

Les pistes de recherche liées à cette approche incluent la prise en compte de plusieurs variantes du cadre de travail. Par exemple, l'un des points cruciaux de l'approche repose sur la façon dont les clauses sont regroupées en clusters. Bien que notre heuristique basée sur la recherche locale et le concept de clause critique se soit montrée très fructueuse en pratique, il pourrait être intéressant de réfléchir à de nouveaux regroupements tenant compte de différentes formes d'interaction entre les clauses. De plus, cette étude se concentre sur le cas booléen classique, où chaque clause a la même importance que les autres. L'utilisateur peut cependant avoir certaines préférences qualitative ou quantitative à propos des informations contenues dans Σ. Dans ce cas, il préfèrera extraire le MUSC

qui correspond le mieux à ses préférences. Nous prévoyons d'étudier l'extension de notre approche à ce cadre.

Références

[1] Armin Biere. Boolforce. `http://fmv.jku.at/booleforce`.

[2] Mukesh Dalal. Anytime families of tractable propositional reasoners. In *AI/MATH'96*, pages 42–45, 1996.

[3] Mukesh Dalal. Semantics of an anytime family of reasponers. In *ECAI'96*, pages 360–364, 1996.

[4] Thomas Eiter and Georg Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactual. *Artificial Intelligence*, 57 :227–270, 1992.

[5] Niklas Eén and Niklas Sörensson. Minisat home page. `http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat`.

[6] Marcelo Finger. Towards polynomial approximations of full propositional logic. In *SBIA'04*, pages 11–20. LNAI, 2004.

[7] Marcelo Finger and Renata Wassermann. The universe of propositional approximations. *Theoretical Computer Science*, 355(2) :153–166, 2006.

[8] Éric Grégoire, Bertrand Mazure, and Cédric Piette. Boosting a complete technique to find MSS and MUS thanks to a local search oracle. In *IJCAI'07*, volume 2, pages 2300–2305. AAAI Press, 2007.

[9] Éric Grégoire, Bertrand Mazure, and Cédric Piette. Local-search extraction of MUSes. *Constraints Journal*, 12(3) :325–344, 2007.

[10] Éric Grégoire, Bertrand Mazure, and Cédric Piette. On approaches to explaining infeasibility of sets of Boolean clauses. In *ICTAI'08*, volume 1, pages 74–83, 2008.

[11] Jinbo Huang. MUP : A minimal unsatisfiability prover. In *ASP-DAC'05*, pages 432–437, 2005.

[12] Anthony Hunter and Sébastien Konieczny. Measuring inconsistency through minimal inconsistent sets. In *KR'08*, pages 358–366, 2008.

[13] Frédéric Koriche. Approximate coherence-based reasoning. *Journal of Applied Non-Classical Logics*, 12(2) :239–258, 2002.

[14] Frédéric Koriche and Jean Sallantin. A logical toolbox for knowledge approximation. In *TARK'01*, pages 193–205. Morgan Kaufmann Publishers Inc., 2001.

[15] Oliver Kullmann. Investigations on autark assignments. *Discrete Applied Mathematics*, 107 :99–137, 2000.

[16] Oliver Kullmann, Ines Lynce, and Joao Marques Silva. Categorisation of clauses in conjunctive normal forms : Minimally unsatisfiable sub-clause-sets and the lean kernel. In *SAT'06*, pages 22–35, 2006.

[17] Mark Liffiton and Karem Sakallah. Algorithms for computing minimal unsatisfiable clause sets. *Journal of Automated Reasoning*, 40(1) :1–33, 2008.

[18] Mark Liffiton and Karem Sakallah. Searching for autarkies to trim unsatisfiable clause sets. In *SAT'08*, pages 182–195, 2008.

[19] Marco Schaerf and Marco Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74 :249–310, 1995.

[20] Lloyd Shapley. A value for n-person games. *Contributions to the Theory of Games II (Annals of Mathematics Studies 28)*, pages 307–317, 1953.

[21] Hans van Maaren and Siert Wieringa. Finding guaranteed MUSes fast. In *SAT'08*, pages 291–304, 2008.

[22] Lintao Zhang and Sharad Malik. Extracting small unsatisfiable cores from unsatisfiable Boolean formula. In *SAT'03*, 2003.