Analyse de conflits dans le cadre de la recherche locale

Gilles Audemard, Jean-Marie Lagniez, Bertrand Mazure, Lakhdar Saïs

Analyse de conflits dans le cadre de la recherche locale*

Gilles Audemard **Jean-Marie Lagniez** **Bertrand Mazure** **Lakhdar Saïs**

Université Lille-Nord de France, Artois, F-62307 Lens
CRIL, F-62307 Lens
CNRS, UMR 8188, F-62307 Lens
{audemard,lagniez,mazure,sais}@cril.fr

Résumé

Dans cet article, nous présentons une nouvelle approche pour sortir des minimums locaux dans le cadre de la recherche locale. Cette approche est basée sur le principe d'analyse de conflits utilisé dans les solveurs SAT modernes. Nous proposons une extension du graphe d'implications au cadre de la recherche locale où plusieurs conflits sont présents pour une interprétation donnée. Nous présentons ensuite une méthode basée sur la propagation unitaire, permettant de construire et d'exploiter de tels graphes. Enfin, nous étendons le schéma classique de WSAT pour y intégrer notre analyse de conflits. Les résultats expérimentaux montrent que l'intégration de notre système d'analyse de conflits améliore sensiblement les performances de WSAT sur les problèmes structurés. De plus, cette méthode isolant des sous-problèmes inconsistants, est capable de montrer que l'instance n'admet pas de modèle.

1 Introduction

La résolution pratique du problème SAT peut être divisée en deux catégories. D'un coté, les méthodes de recherche locale [16, 15, 10], incomplètes, et de l'autre les méthodes complètes, basées la plupart du temps sur la procédure de Davis et Putnam [3]. Les premières explorent l'espace de recherche de manière stochastique et ne savent, dans la plupart des cas, répondre uniquement lorsque la formule est satisfaisable. Autant en recherche opérationnelle elles ont montré leur efficacité, autant, dans le cas du problème SAT, les méthodes de recherche locale ne sont compétitives que pour les problèmes aléatoires.

*supporté par le projet ANR UNLOC

D'un autre coté, l'amélioration constante des méthodes complètes permet depuis quelques années la résolution de larges instances issues de problèmes industriels. L'un des composants clés de ces solveurs modernes est l'analyse de conflits [18]. Elle fait partie des contributions ayant le plus apportées, en terme d'efficacité, pour la résolution pratique de telles instances. Les solveurs actuels de type CDCL (*Conflict Driven, Clause Learning*), tel que Berkmin [8], Minisat [5] et d'autres, implantent une telle approche. Lors de l'analyse d'un conflit un niveau de backtrack est calculé, un « *nogood* » est extrait et ajouté à la base. Cette sauvegarde permet d'éviter de parcourir des sous-espaces insatisfaisables.

Dans cet article, nous proposons d'étendre l'analyse de conflits [18] dans le cadre des algorithmes de recherche locale stochastiques. L'objectif est double. Premièrement, nous souhaitons exploiter cette analyse et les nogoods résultants pour sortir des minimums locaux. En effet, la stratégie utilisée pour sortir des minimums locaux joue un grand rôle dans l'efficacité des méthodes de recherche locale. Il est à noter qu'une approche semblable à déjà été proposée dans [2] mais était limitée à des résolutions sur deux clauses. L'autre objectif est de répondre à un des challenges importants de la communauté scientifique : proposer une méthode de recherche locale qui puisse répondre à l'insatisfaisabilité d'une formule [17]. Cela est possible car les nogoods déduits par l'analyse des conflits vont être ajoutés à la formule initiale. Cette extension présente des difficultés inhérentes aux méthodes de recherche locales. Celles-ci utilisent des interprétations complètes dans lesquelles plusieurs clauses peuvent être falsifiées. L'absence de notion

de niveau de décision participe également à cette difficulté.

L'article est organisé de la façon suivante. Après quelques définitions préliminaires (section 2), nous présentons le schéma de base d'une recherche locale de type WSAT ainsi que le graphe d'implications classique et son utilisation (section 3). Dans la section 4, nous proposons dans un premier temps une façon d'étendre la notion de graphe d'implications à une interprétation complète. Ensuite, une méthode basée sur la propagation unitaire pour la construction de tels graphes est proposée. Pour terminer, l'intégration de cette analyse à un solveur de type WSAT est décrite. Avant de conclure, nous donnons des résultats expérimentaux dans la section 5.

2 Définitions et notations

Le problème SAT est un problème de décidabilité consistant à savoir si une formule propositionelle Σ mise sous forme CNF (*Conjontive Normal Form*) est logiquement valide. Une formule *CNF* Σ est un ensemble (interprété comme une conjonction) de *clauses*, où une clause est un ensemble (interprété comme une disjonction) de *littéraux*. Un littéral est soit positif (x) soit négatif ($\overline{x}$). Les deux littéraux x et $\overline{x}$ sont appelés littéraux *complémentaires*. L'ensemble des variables apparaissant dans Σ sera noté $\mathcal{V}_\Sigma$. Une interprétation $\mathcal{I}$ d'une formule booléenne Σ associe une valeur $\mathcal{I}(x)$ aux variables $x \in \mathcal{V}_\Sigma$. L'interprétation $\mathcal{I}$ est dite *complète* si elle associe une valeur de vérité à chaque $x \in \mathcal{V}_\Sigma$, sinon elle est dite *partielle*. Une interprétation peut également être représentée par un ensemble de littéraux. Un *modèle* d'une formule de Σ est une interprétation $\mathcal{I}$ qui satisfait la formule, ce qui est noté $\mathcal{I} \models \Sigma$. La négation d'une formule Γ sera notée $\overline{\Gamma}$. Les notations suivantes sont utilisées par la suite :

- $\Sigma_{|_x}$ dénote la formule Σ simplifiée par l'affectation du littéral x à la valeur vrai. Cette notation est étendue aux interprétations : soit $\mathcal{P} = \{x_1, ..., x_n\}$ une interprétation, on définit $\Sigma_{|_\mathcal{P}} = (...(\Sigma|_{x_1})...|_{x_n})$;
- Σ^* dénote la formule close Σ après application de la propagation unitaire ;
- $\models_*$ dénote la déduction logique par propagation unitaire : $\Sigma \models_* x$ signifie que le littéral x est déduit par propagation unitaire à partir de Σ. On écrit $\Sigma \models_* \bot$ si la formule est inconsistante (insatisfaisable) par propagation unitaire.

3 Préliminaires

3.1 Algorithmes de recherche locale

Le schéma de recherche locale est relativement simple. Celui-ci consiste à se déplacer stochastiquement dans l'ensemble des interprétations jusqu'à l'obtention d'une solution. À chaque étape, on essaie de réduire le nombre de clauses non satisfaites (c'est l'étape communément appelé descente). L'interprétation suivante est choisie parmi l'ensemble des interprétations voisines (c'est-à-dire différant uniquement sur la valeur d'une seule variable) de l'interprétation courante. Lorsqu'aucune descente n'est possible, on se trouve alors dans un minimum local. L'un des points cruciaux des méthodes de recherche locale est la technique utilisée pour s'échapper de ces minimums locaux. De nombreuses approches ont été proposées dans la littérature : liste tabou [13], recuit simulé, ajout des résolvantes [2]... D'autres améliorations (choix de la variable à flipper, propagation unitaire, ...) ont également été proposé, le lecteur intéressé pourra se référer à [14] pour une description plus détaillée. Nous présentons ci-dessous l'algorithme WSAT que nous utilisons comme base de la nouvelle approche que nous proposons dans ce papier.

Algorithme 1 : WSAT

```
   Input : Σ une formule CNF
   Output : SAT si un modèle de Σ est trouvé,
            UNKNOWN sinon
1  for i ← 1 to MaxTries do
2      I_c ← interprétation complète de Σ;
3      for j ← 1 to MaxFlips do
4          if I_c ⊨ Σ then
5              return SAT;
6          α ∈ Σ tel que I_c ⊭ α;
7          if ∃x ∈ α permettant une descente then
8              flipper(x)
9          else                    /* minimum local */
10             Remplacer I_c par une interprétation
               obtenu selon un critère d'échappement;
11 return UNKNOWN;
```

L'algorithme WSAT (algorithme 1) génère une interprétation complète et et la modifie jusqu'à obtenir un modèle ou jusqu'à ce que le nombre maximal de réparations autorisées soit atteint. Pour ce faire, une clause est choisie aléatoirement parmi l'ensemble des clauses falsifiées par l'interprétation courante (ligne 6). S'il existe une variable appartenant à cette clause permettant de réduire le nombre de clauses falsifiées (descente) cette dernière est

flippée[1] (ligne 8). Sinon, on se trouve dans un minimum local et un critère d'échappement est utilisé (ligne 10). Cette opération est répétée un certain nombre de fois ($MaxFlips$) fixé au départ. Ce processus peut se répéter un nombre maximal de fois ($MaxTries$) fixé au démarrage de la procédure. Un des avantages de cette méthode est la vitesse à laquelle la prochaine variable à flipper est choisie. En effet, le choix de la prochaine variable est limité à un sous-ensemble de variables restreint. De cette manière, WSAT parcourt rapidement de nombreuses interprétations.

3.2 Analyse de conflits et graphe d'implications

Nous introduisons maintenant la notion de graphes d'implications utilisée par les solveurs complets de type CDCL pour analyser les conflits, déduire de nouvelles clauses (ou nogoods) et effectuer un retour arrière non chronologique. Le graphe d'implications est un graphe orienté acyclique (DAG) permettant une représentation de l'affectation des variables et des propagations unitaires issues de ces affectations. Dans un graphe d'implications, chaque sommet représente un littéral et possède un niveau de décision. Dès que l'on affecte une variable, celle-ci est appelée variable de décision et elle donne son rang d'instanciation à toutes les variables qui seront affectées par propagation unitaire résultant de cette affectation. Pour chaque littéral y propagé, on garde en mémoire la clause à l'origine de cette propagation. Cette clause, notée $\overrightarrow{cla}(y)$, est de la forme $(x_1 \vee ... \vee x_n \vee y)$ tel que $\forall x_i, 1 \leq i \leq n, x_i \notin \mathcal{I}$ et $y \in \mathcal{I}$. Lorsqu'un littéral y n'est pas obtenu par propagation unitaire mais qu'il correspond à un point de décision, $\overrightarrow{cla}(y)$ est indéfini et par convention on notera $\overrightarrow{cla}(y) = \bot$.

Lorsque $\overrightarrow{cla}(y) \neq \bot$, on note par $exp(y)$ l'ensemble $\{\overline{x_i} | x_i \in \overrightarrow{cla}(y) \setminus \{y\}\}$, appelé l'ensemble des *explications* de y. Autrement dit, $\overrightarrow{cla}(y) = (x_1 \vee ... \vee x_n \vee y)$ alors les explications sont les littéraux $\overline{x_i}$ qui constituent la condition sous laquelle $\overrightarrow{cla}(y)$ devient une clause unitaire $\{y\}$.

Quand $\overrightarrow{cla}(y)$ est indéfini, $exp(y)$ est égal à l'ensemble vide. Dans un graphe d'implications, chaque noeud admet un ensemble d'explications qui correspond à l'ensemble de ses prédécesseurs dans le graphe. De manière formelle on définit le graphe d'implications de la manière suivante :

Définition 1 (graphe d'implications) *Soit Σ une formule sous forme CNF, $\mathcal{I}_p$ une interprétation partielle. Le graphe d'implications associé à Σ, $\mathcal{I}_p$ et exp est $\mathcal{G}_\Sigma^{\mathcal{I}_p} = (\mathcal{N}, \mathcal{A})$ où :*

1. *$\mathcal{N} = \{x | x \in \mathcal{I}_p\}$ i.e. un noeud pour chaque littéral de $\mathcal{I}_p$;*
2. *$\mathcal{A} = \{(x, y) | x \in \mathcal{I}_p, y \in \mathcal{I}_p, x \in exp(y)\}$.*

[1] Flipper une variable consiste à inverser sa valeur.

Exemple 3.1 *Soient $\Sigma = \{\phi_1, ..., \phi_{11}\}$ une formule CNF tel que :*

$$\begin{array}{ll} \phi_1 : (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) & \phi_2 : (x_1 \vee \overline{x_4} \vee \overline{x_5}) \\ \phi_3 : (x_2 \vee \overline{x_1}) & \phi_4 : (x_4 \vee \overline{x_7} \vee \overline{x_6}) \\ \phi_5 : (x_3 \vee \overline{x_5}) & \phi_6 : (x_5 \vee \overline{x_7}) \\ \phi_7 : (x_6 \vee \overline{x_8}) & \phi_8 : (x_7 \vee \overline{x_8}) \\ \phi_9 : (x_8 \vee \overline{x_4}) & \phi_{10} : (x_1 \vee \overline{x_8}) \\ \phi_{11} : (x_7 \vee \overline{x_9}) & \end{array}$$

et $\mathcal{I}_p$ l'interprétation partielle suivante $\mathcal{I}_p = \{\langle(x_2^1)\rangle\langle(x_6^2)\rangle\langle(x_9^3)\ x_7^3\ x_4^3\ x_5^3\ x_3^3\ x_1^3\ \overline{x_1^3}\rangle\}$ où x_i^j indique que le littéral x_i est affecté au niveau j. Les littéraux de décision sont indiqués entre parenthèses et les autres littéraux représentent ceux affectés par propagation unitaire. Le niveau de décision courant est 5 et $\Sigma_{|\mathcal{I}_p} \models_* \bot$. La figure 1 représente le graphe d'implications $\mathcal{G}_\Sigma^{\mathcal{I}_p}$ associée à Σ, $\mathcal{I}_p$ et exp (l'ensemble des explications).

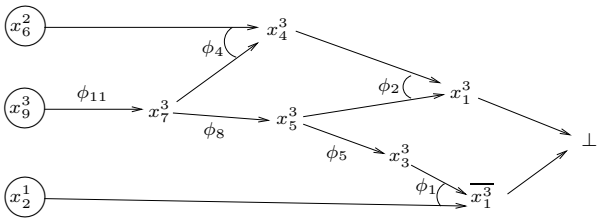


FIG. 1 – Graphe d'implication (exemple 3.1)

Le graphe d'implications est utilisé pour extraire des nogoods. Ceux-ci sont construits par l'application de plusieurs résolutions sur la clause devenue fausse et en remontant ce graphe. Divers types de nogoods peuvent alors être générés. L'un des plus utilisés et des plus efficaces, est le « *first UIP* » (Unique Implication Point). Pour plus de détails, se référer à [18, 1].

4 Analyse de conflits et recherche locale

4.1 Définition des graphes conflits

Nous avons vu dans la section 3.1 qu'un des points importants des méthodes de recherche locales est le critère d'échappement. Nous proposons une nouvelle méthode qui, partant de l'interprétation complète, génère un graphe d'implications. Celui-ci va nous permettre d'ajouter des nogoods

à la formule mais également de choisir les variables à flipper pour sortir de ce minimum local.

Néanmoins, dans le cadre de la recherche locale, le fait de considérer des interprétations complètes rend la définition et la construction de graphes d'implications problèmatique. D'une part, on ne retrouve pas les notions de niveau d'affectation, de littéraux de décision ou propagés. D'autre part, une interprétation complète peut falsifier plusieurs clauses. Dans cette section, nous proposons une nouvelle définition des graphes d'implications adaptée au contexte de la recherche locale stochastique. Nous donnons ci-dessous quelques définitions préalables en considérant une formule CNF Σ et une interprétation complète $\mathcal{I}_c$. On dit qu'un littéral ℓ satisfait (respectivement falsifie) une clause $\beta \in \Sigma$ si $\ell \in \mathcal{I}_c \cap \beta$ (respectivement $\ell \in \mathcal{I}_c \cap \overline{\beta}$). On note l'ensemble des littéraux satisfaisant (resp. falsifiant) une clause β pour une interprétation $\mathcal{I}_c$ par $\mathcal{L}^+_{\mathcal{I}_c}(\beta)$ (respectivement $\mathcal{L}^-_{\mathcal{I}_c}(\beta)$).

Définition 2 (clause unisatisfaite) *Une clause β est dite* unisatisfaite *si $|\mathcal{L}^+_{\mathcal{I}_c}(\beta)| = 1$.*

Définition 3 (clause critique et liée [9]) *Une clause α est dite critique si $|\mathcal{L}^+_{\mathcal{I}_c}(\alpha)| = 0$ et $\forall \ell \in \alpha$, $\exists \alpha' \in \Sigma$ avec $\sim \ell \in \alpha'$ et $|\mathcal{L}^+_{\mathcal{I}_c}(\alpha')| = 1$ (α' est satisfaite en $\sim \ell$). Les clauses α' sont dites* liées *à α pour l'interprétation $\mathcal{I}_c$.*

Exemple 4.1 *Soient $\Sigma = (\overline{a} \vee \overline{b} \vee \overline{c}) \wedge (a \vee \overline{b}) \wedge (b \vee \overline{c}) \wedge (c \vee \overline{a})$ et $\mathcal{I}_c = \{a, b, c\}$. La clause $\alpha_1 = (\overline{a} \vee \overline{b} \vee \overline{c})$ est critique. Les trois autres clauses sont liées à α_1 pour $\mathcal{I}_c$.*

Il est important de noter que dans un minimum local toutes les clauses falsifiées sont critiques [9]. On peut maintenant définir la notion de graphe conflit pour les interprétations complètes.

Définition 4 (graphe conflit sur z) *Soient Σ une formule sous forme CNF, $\mathcal{I}$ une interprétation complète conflictuelle de Σ. Étant données deux clauses de Σ, $\beta = \{\beta_1, \ldots, \beta_k, z\}$ falsifiée par $\mathcal{I}$ et $\gamma = \{\gamma_1, \ldots, \gamma_l, \overline{z}\}$ unisatisfaite en z pour $\mathcal{I}$, on construit le graphe conflits $\mathcal{G}^z_{\Sigma,\mathcal{I}} = (\mathcal{N}, \mathcal{A})$ de la manière suivante :*

1. – $\{z, \overline{z}, \bot\} \subseteq \mathcal{N}$;
   – $\{\overline{\gamma_1}, \ldots, \overline{\gamma_l}\} \subseteq \mathcal{N}$;
   – $\{\overline{\beta_1}, \ldots, \overline{\beta_k}\} \subseteq \mathcal{N}$;
2. – $\{(\underline{z}, \bot), (\overline{z}, \bot)\} \subseteq \mathcal{A}$;
   – $\{(\overline{\beta_1}, z), \ldots, (\overline{\beta_k}, z)\} \subseteq \mathcal{A}$;
   – $\{(\overline{\gamma_1}, \overline{z}), \ldots, (\overline{\gamma_k}, \overline{z})\} \subseteq \mathcal{A}$;
3. *$\forall x \in \mathcal{N}$, si $x \neq z$ et $\alpha = \bigwedge\{y \in \mathcal{N} \mid (y, x) \in \mathcal{A}\} \not\models \bot$ alors $\overline{\alpha} \vee x \in \Sigma$ et est unisatisfaite en x.*

Il est clair que pour un littéral z donné, les clauses β et γ ne sont pas uniques. De plus, il est possible que plusieurs clauses unisatisfaites expliquent un même littéral. On peut donc remarquer qu'il n'y a pas unicité du graphe conflit.

Exemple 4.2 *Considérons de nouveau l'exemple 3.1. Soit l'interprétation complète $\mathcal{I}_c \subseteq \mathcal{V}_\Sigma$ tel que $\mathcal{I}_c = \{x_1, \ldots, x_9\}$. La figure 2 schématise un graphe conflit $\mathcal{G}^{x_1}_{\Sigma,\mathcal{I}}$ construit sur la variable x_1.*

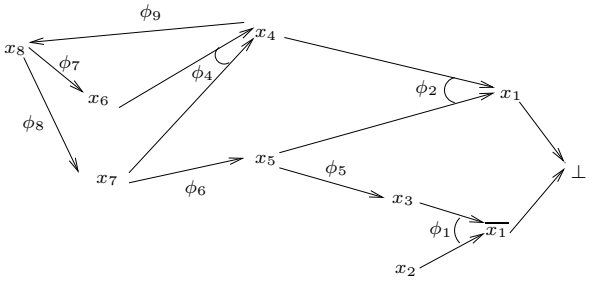


FIG. 2 – Graphe conflit défini sur la variable x_1

Il est important de noter que l'existence d'un tel graphe n'est pas toujours assurée, et la proposition suivante exprime les conditions sous lesquelles la construction de ce graphe conflit est possible.

Proposition 1 *Soient Σ une formule sous forme CNF, $\mathcal{I}$ une interprétation complète construite sur les variables de Σ. Le graphe conflit $\mathcal{G}^x_{\Sigma,\mathcal{I}}$ est constructible si et seulement si x et $\overline{x}$ apparaissent respectivement dans une clause fausse et une clause unisatisfaite.*

Preuve 1 *Cette proposition découle directement de la définition d'un graphe conflit.*

Corrolaire 2 *Soit $\alpha \in \Sigma$ une clause critique. $\forall x \in \alpha$ il est possible de construire un graphe conflit défini sur x.*

Preuve 2 *Par définition d'une clause critique $\alpha \in \Sigma$ on a $\forall x \in \alpha$, $\exists \beta \in \Sigma$ unisatisfaite en x. D'après la proposition 1, il est facile de voir que $\forall x \in \alpha$ il existe un graphe conflit en x.*

Lorsque l'on se trouve dans un minimum local, toutes les clauses non satisfaites sont critiques. Le corrolaire 2 nous assure donc que, dans ce cas, il est toujours possible de créer un graphe conflit. Ceci est très important, puisque cela va nous permettre de construire un graphe afin de générer un nogood qui va lui même permettre de s'extraire du minimum local. Néanmoins, générer de tels nogoods en

partant d'un graphe conflit est un problème difficile. En effet, la notion de niveau étant absente, l'analyse de conflit classique basée sur la notion d'UIPs ne peut être étendue. De plus, les graphes conflits, tel que définis précédemment, peuvent contenir des cycles. La présence de cycles au sein d'un graphe conflit peut conduire à produire par résolution des clauses tautologiques. Ces clauses sont par nature inutiles. Pour palier à ces problèmes, nous proposons dans la section suivante, une méthode, qui, à partir du graphe conflit d'une interprétation complète, extrait le graphe d'implications associé. De cette manière, on retrouve le cas classique des graphes d'implications utilisés dans les solveurs SAT modernes qui permettent d'extraire facilement et rapidement des nogoods importants pour simplifier la recherche.

4.2 Construction des graphes conflits

Afin de palier aux problèmes énoncés dans la section précédente, nous proposons une première méthode basée sur la propagation unitaire. Partant de l'interprétation complète conflictuelle $\mathcal{I}$, nous allons construire une interprétation partielle par propagation unitaire dont les points de choix ont les mêmes valeurs dans les deux interprétations. Cette interprétation, que nous définissons ci-dessous, permettra ensuite de définir un graphe conflit acyclique.

Définition 5 (interprétation partielle dérivée) *Soient Σ une formule sous forme CNF, $\mathcal{I}$ une interprétation complète conflictuelle construite sur $\mathcal{V}_\Sigma$. L'interprétation partielle dérivée de $\mathcal{I}$, noté $\mathcal{I}'$, est construite de manière incrémentale de la façon suivante :*

- $\mathcal{I}'_0 = \emptyset$;
- $\mathcal{I}'_{i+1} = \mathcal{I}'_i \cup \{(x_{i+1}), x^1_{i+1}, ..., x^k_{i+1}\}$ *tel que* $x_{i+1} \in \mathcal{V}_\Sigma \setminus \mathcal{V}_{\mathcal{I}'_i}$ *et* $\forall j, 1 \leqslant j \leqslant k$ *on a* $\Sigma_{|_{\mathcal{I}'_i \cup \{x_{i+1}\}}} \models_* x^j_{i+1}$ *avec* $x^j_{i+1} \in \mathcal{I}$;
- $\mathcal{I}' = \mathcal{I}'_i \cup \{(x_{i+1}), x^1_{i+1}, ..., x^l_{i+1}\}$ *tel que* $x_{i+1} \in \mathcal{V}_\Sigma \setminus \mathcal{V}_{\mathcal{I}'_i}$ *et* $\forall j, 1 \leqslant j \leqslant l$ *on a* $\Sigma_{|_{\mathcal{I}'_i \cup \{x_{i+1}\}}} \models_* x^j_{i+1}$ *avec* $x^j_{i+1} \in \mathcal{I}$ *pour* $j \neq l$ *et* $x^l_{i+1} \notin \mathcal{I}$.

L'ensemble des points de choix sera nommé ensemble conflit et on appellera littéral conflit le littéral $x \in \mathcal{I}' \setminus \mathcal{I}$.

Le choix des variables apparaissant dans l'ensemble conflit est un choix heuristique et conduit à différentes interprétations partielles dérivées.

Exemple 4.3 *Reprenons la formule Σ de l'exemple 3.1 et l'interprétation complète et conflictuelle $\mathcal{I}_c = \{x_1, \dots x_9\}$.*

Dans un premier temps, considérons l'ordre lexicographique pour générer l'interprétation partielle dérivée. On obtient :

- $\mathcal{I}'_0 = \emptyset$
- $\mathcal{I}'_1 = \{(x_1), x^1_2, \overline{x^1_3}\}$

La variable conflit est alors x_3 et l'ensemble conflit est limité $\{x_1\}$. Si nous considérons maintenant l'ordre lexicographique inverse, alors l'interprétation partielle dérivée est :

- $\mathcal{I}'_0 = \emptyset$
- $\mathcal{I}'_1 = \{(x^1_9), x^1_7, x^1_5, x^3_1\}$
- $\mathcal{I}'_2 = \{(x^1_9), x^1_7, x^1_5, x^3_1, (x^2_8), x^2_6, x^2_4, x^2_1, x^2_2, \overline{x^2_2}\}$

Le littéral conflit est x_2 et on a comme ensemble conflit $\{x_9, x_8\}$.

L'interprétation complète étant conflictuelle, son interprétation partielle dérivée va différer sur au moins un littéral (le littéral conflit). La proposition suivante exprime le fait qu'il existe au moins une clause falsifiée contenant ce littéral. C'est à partir de celui-ci que l'on construira notre graphe conflit.

Proposition 3 *Soient Σ une formule CNF, $\mathcal{I}$ une interprétation complète conflictuelle de Σ et $\mathcal{I}'$ une interprétation partielle dérivée de $\mathcal{I}$. Soit x le littéral conflit, alors $exp(x) \subseteq \mathcal{I}$ et la clause $\overrightarrow{cla}(x)$ est falsifiée par $\mathcal{I}$.*

Preuve 3 *Tout d'abord il est facile de voir que par construction de $\mathcal{I}'$ on a $exp(x) \subseteq \mathcal{I}$. En effet, supposons que $exp(x) \not\subseteq \mathcal{I}$ alors $\exists y \in exp(x)$ tel que $y \notin \mathcal{I}$. Par définition $exp(x) \subseteq \mathcal{I}'$, donc $y \in \mathcal{I}'$ et par conséquent le littéral y est aussi un littéral conflit. Par construction de $\mathcal{I}'$ il ne peut y avoir qu'un seul littéral conflit, il en résulte alors $y = x$. Cette conclusion est absurde car une variable propagée ne peut pas appartenir à son explication.*

Montrons à présent que $\mathcal{I} \not\models \overrightarrow{cla}(x)$. Pour cela, raisonnons également par l'absurde et supposons $\overrightarrow{cla}(x)$ soit satisfait par $\mathcal{I}$. En premier lieu, on peut noter que x est une variable obtenue par propagation unitaire. Il est donc possible de trouver une explication $exp(x)$ et une clause $\overrightarrow{cla}(x) \in \Sigma$ tel que $\overrightarrow{cla}(x) = \overline{exp(x)} \vee x$. On sait par ailleurs que $exp(x) \subseteq \mathcal{I}$, donc $\overline{exp(x)} \not\subseteq \mathcal{I}$. Puisque $\overrightarrow{cla}(x)$ est satisfaite sous l'interprétation $\mathcal{I}$, elle ne peut l'être qu'en x. Ce qui est absurde puisque $x \notin \mathcal{I}$.

La proposition suivante exprime le fait que toutes les clauses (hormis la clause falsifiée) ayant servi à la construction du graphe sont unisatisfaites.

Proposition 4 *Soient Σ une formule CNF, $\mathcal{I}$ une interprétation complète conflictuelle de Σ, $\mathcal{I}'$ une*

interprétation partielle dérivée de $\mathcal{I}$ en fonction de Σ et x le littéral conflit associé à $\mathcal{I}'$. Considérons le graphe d'implications $\mathcal{G}_\Sigma^{\mathcal{I}'} = (\mathcal{N}, \mathcal{A})$, alors $\forall y \in \mathcal{N} \setminus \{x\}$ on a $\overrightarrow{cla}(y) = \bot$ ou $\overrightarrow{cla}(y)$ est unisatisfaite par $\mathcal{I}'$ en y.

Preuve 4 *Deux cas sont à considérer :*

1. *y est un point de choix et alors $\overrightarrow{cla}(y) = \bot$;*
2. *y n'est pas un littéral propagé. Il existe $\overrightarrow{cla}(y) \in \Sigma$ tel que $\overrightarrow{cla}(y) = \overline{exp(y)} \vee y$. Par construction de $\mathcal{I}'$, on a $x \notin exp(y)$ et $\mathcal{I}' \setminus \{x\} \subset \mathcal{I}$. Puisque $y \neq x$ on a $\{exp(y), y\} \subseteq \mathcal{I}' \setminus \{x\}$. Par transitivité on a $\{exp(y), y\} \subseteq \mathcal{I} \setminus \{x\}$. Donc la clause $\overrightarrow{cla}(y)$ est unisatisfaite par $\mathcal{I}$ en y.*

Nous prouvons dans la proposition 5 que le graphe d'implications obtenu à partir de l'interprétation partielle dérivée peut être étendu en un graphe conflit sur le littéral conflit. Ayant maintenant un graphe d'implications comme ceux utilisés dans les solveurs CDCL classiques [5], il est possible d'analyser celui-ci pour générer un nogood qui sera ensuite ajouté à la base de clauses.

Proposition 5 *Soient Σ une formule CNF, $\mathcal{I}$ une interprétation complète de Σ, $\mathcal{I}'$ une interprétation partielle dérivée et x le littéral conflit associé à $\mathcal{I}'$. Si $\exists \alpha \in \Sigma$ unisatisfaite par $\mathcal{I}$ en x, alors il est possible d'étendre le graphe d'implication $\mathcal{G}_\Sigma^{\mathcal{I}'} = (\mathcal{N}, \mathcal{A})$ associé à $\mathcal{I}'$ en un graphe conflit $\mathcal{G}^x_{\mathcal{I},\Sigma} = (\mathcal{N}', \mathcal{A}')$ de la manière suivante :*

- *$\mathcal{N}' = \mathcal{N} \cup \{y \in \overline{\alpha} \setminus x\} \cup \{\overline{x}, \bot\}$;*
- *$\mathcal{A}' = \mathcal{A} \cup \{(y, \overline{x}) | y \in \overline{\alpha} \setminus x\} \cup \{(x, \bot), (\overline{x}, \bot)\}$.*

Preuve 5 *Avant de vérifier que $\mathcal{G}^x_{\mathcal{I},\Sigma}$ est bien un graphe conflit valide, il faut identifier les clauses $\beta = \{\beta_1, \ldots, \beta_k, z\} \in \Sigma$ falsifiée par $\mathcal{I}$ et $\gamma = \{\gamma_1, \ldots, \gamma_l, \overline{z}\} \in \Sigma$ unisatisfaite en z pour $\mathcal{I}$. Par hypothèse, la clause α est unisatisfaite par $\mathcal{I}$ en x. On peut donc poser $\gamma = \alpha$. D'après la proposition 3, nous pouvons prendre pour β la clause $\overrightarrow{cla}(x)$. En effet, $x \in \overrightarrow{cla}(x)$ et la clause $\overrightarrow{cla}(x)$ est bien falsifiée par $\mathcal{I}$. Les deux clauses servant à la construction du graphe conflit étant identifiées, pour valider que $\mathcal{G}^x_{\mathcal{I},\Sigma} = (\mathcal{N}', \mathcal{A}')$ est un graphe conflit, il suffit, d'après la définition 4, de vérifier les propriétés suivantes :*

1. *– $\{x, \overline{x}, \bot\} \subseteq \mathcal{N}'$. Par hypothèse, on a $\{\overline{x}, \bot\} \subseteq \mathcal{N}'$, reste à monter que $x \in \mathcal{N}'$. On sait que $x \in \mathcal{I}'$, par construction du graphe $\mathcal{G}_\Sigma^{\mathcal{I}'}$, on a $x \subseteq \mathcal{N}$. Donc, puisque $\mathcal{N} \subseteq \mathcal{N}'$, $x \in \mathcal{N}'$;*
   – $\{\overline{\gamma_1}, \ldots, \overline{\gamma_l}\} \subseteq \mathcal{N}'$. On a $\{y \in \overline{\alpha} \setminus x\} \subseteq \mathcal{N}'$ et $\gamma = \alpha$. Donc $\{y \in \overline{\gamma} \setminus x\} \subseteq \mathcal{N}'$;
   – $\{\overline{\beta_1}, \ldots, \overline{\beta_k}\} \subseteq \mathcal{N}'$. Par hypothèse, $\beta = \overrightarrow{cla}(x) = \beta_1 \vee \cdots \vee \beta_k \vee x = \overline{exp}(x) \vee x$. D'où $exp(x) = \{\overline{\beta_1}, \ldots, \overline{\beta_k}\}$. D'après la proposition 3, on a $exp(x) \subseteq \mathcal{I}'$, d'où $exp(x) \subseteq \mathcal{N}$ (voir définition 1). Puisque $\mathcal{N} \subseteq \mathcal{N}'$, par transitivité on a $exp(x) \subseteq \mathcal{N}'$;
2. *– $\{(x, \bot), (\overline{x}, \bot)\} \subseteq \mathcal{A}'$. Par construction ;*
   – $\{(\overline{\gamma_1}, \overline{x}), \ldots, (\overline{\gamma_k}, \overline{x})\} \subseteq \mathcal{A}'$. Idem ;
   – $\{(\overline{\beta_1}, x), \ldots, (\overline{\beta_k}, x)\} \subseteq \mathcal{A}'$. On sait que $exp(x) = \{\overline{\beta_1}, \ldots, \overline{\beta_k}\}$ et que $\{exp(x), x\} \subseteq \mathcal{I}'$. Par construction de $\mathcal{A}'$ et d'après la définition 1, on a $\{(\overline{\beta_1}, x), \ldots, (\overline{\beta_k}, x)\} \subseteq \mathcal{A} \subseteq \mathcal{A}'$;
3. *$\forall x \in \mathcal{N}$, si $x \neq z$ et $\alpha = \bigwedge\{y \in \mathcal{N} \mid (y, x) \in \mathcal{A}\} \not\models \bot$ alors $\overline{\alpha} \vee x \in \Sigma$ et est unisatisfaite en x. D'après la proposition 4, $\forall y \in \mathcal{N}$ tel que $y \neq x$ et $\overrightarrow{cla}(y) \neq \bot$ on a $\overrightarrow{cla}(y)$ unisatisfaite par $\mathcal{I}$ en y. Par construction de $\mathcal{G}_\Sigma^{\mathcal{I}}$ et $\mathcal{G}^x_{\mathcal{I},\Sigma}$ la propriété précédente est vérifiée.*

Exemple 4.4 *Reprenons l'exemple 3.1 et les interprétations partielles dérivées obtenues dans l'exemple 4.3. Nous pouvons alors étendre les graphes d'implications associés à ces deux interprétations en deux graphes conflits schématisés dans les figures 3 (ordre lexicographique) et 4 (ordre lexicographique inverse).*

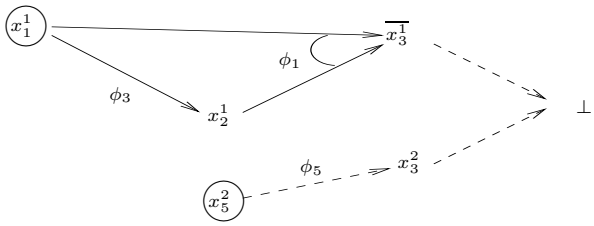


FIG. 3 – Graphe conflit construit à l'aide de la propagation unitaire (ordre lexicographique)

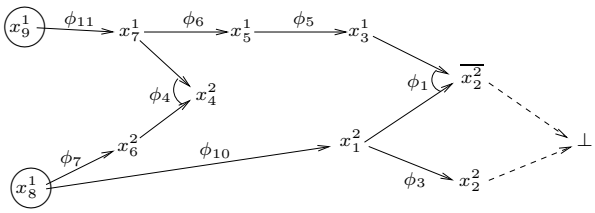


FIG. 4 – Graphe conflit construit à l'aide de la propagation unitaire (ordre lexicographique inverse)

4.3 Implantation

Nous proposons d'intégrer le graphe conflit défini dans la section précédente au sein d'un solveur de type WSAT, le graphe conflit sera construit

Algorithme 2 : CDLS

Input : Σ une formule CNF
Output : SAT ou $UNSAT$ si la satisfaisabilité de Σ a pu être décidée, $UNKNOWN$ sinon

1 **for** $i \leftarrow 1$ *to* $MaxTries$ **do**
2 $\mathcal{I}_c \leftarrow$ interpretationCompletePU(Σ);
3 **for** $j \leftarrow 1$ *to* $MaxFlips$ **do**
4 **if** $\mathcal{I}_c \models \Sigma$ **then**
5 **return** SAT;
6 $\Gamma = \{\alpha \in \Sigma | \ \mathcal{I}_c \not\models \alpha\}$;
7 **if** $\exists x \in \mathcal{V}_\Gamma$ *permettant une descente* **then**
8 $flipper(x)$;
9 **else**
 `/* minimum local */`
10 $\alpha \in \Gamma$;
11 $\beta \leftarrow analyseConflitRL(\Sigma, \mathcal{I}_c, \alpha)$;
12 **if** $\beta = \bot$ **then**
13 **return** UNSAT;
14 $\Sigma \leftarrow \Sigma \cup \{\beta\}$;
15 **return** $UNKNOWN$;

Fonction `analyseConflitRL`

Input : - Σ une formule CNF ;
- $\mathcal{I}_c$ une interprétation complète ;
- $\alpha \in \Sigma$ une clause critique pour$\mathcal{I}_c$.

Output : β une clause construite sur $\mathcal{V}_\Sigma$

1 $\mathcal{E} \leftarrow ensembleConflit(\Sigma, \mathcal{I}_c, \alpha)$;
2 $\gamma \leftarrow \emptyset$;
3 $\mathcal{I}_p \leftarrow \emptyset$;
4 **while** $(\gamma = \emptyset)$ *and* $(\mathcal{I}_p \subset \mathcal{I}_c)$ **do**
5 $\mathcal{E} \leftarrow \mathcal{E} \setminus \mathcal{I}_p$;
6 $\mathcal{I}_p \leftarrow \mathcal{I}_p \cup \{x\}$ tel que $x \in \mathcal{E}$;
7 $\gamma \leftarrow BCP()$;
8 **if** $\gamma \neq \emptyset$ **then** `/* CAS 1 */`
9 $\beta \leftarrow analyseConfitClassique(\mathcal{G}_\Sigma^{\mathcal{I}_p})$;
10 $flipper(x)$ avec x littéral assertif;
11 **else** `/* CAS 2 */`
12 $\mathcal{I}' \leftarrow$ interprétation dérivé associée à $\mathcal{I}_p$;
13 $y \leftarrow$ littéral conflit de $\mathcal{I}'$;
14 $\mathcal{G}^y_{(\Sigma,\mathcal{I}')} \leftarrow$ graphe conflit étendu de $\mathcal{G}^{\mathcal{I}'}_\Sigma$;
15 $\beta \leftarrow analyseConfitClassique(\mathcal{G}^y_{(\Sigma,\mathcal{I}')})$;
16 **forall** $x \in \mathcal{I}_p \setminus \mathcal{I}_c$ **do**
17 $flipper(x)$;
18 **return** β;

lorsque WSAT aura atteint un minimum local afin de s'échapper de celui-ci. Cette méthode est nommée CDLS (*Conflict Driven for Local Search*).

Lors de la construction de l'interprétation partielle dérivée, deux cas peuvent se présenter. Soit un conflit est atteint lors de la propagation unitaire (voir figure 4) et dans ce cas, le graphe d'implications résultant est utilisé pour analyser le conflit comme le font les solveurs SAT modernes et extraire une clause assertive associée au premier UIP. Sinon (voir figure 3), nous étendons le graphe d'implications de l'interprétation partielle dérivée en un graphe conflit sur le littéral conflit, et de la même manière, on extrait une clause assertive. Dans les deux cas, une clause assertive est ajoutée à la base de clauses de la formule. Lorsque celle-ci est égale à la clause vide l'insatisfaisabilité de la formule est ainsi démontrée. Contrairement à WSAT classique qui n'autorise qu'un seul flip à chaque étape, CDLS va pouvoir flipper un ensemble de variables afin de s'échapper d'un minimum local. Ces variables sont celles qui diffèrent entre l'interprétation complète et l'interprétation partielle dérivée.

Il est important de noter que l'algorithme CDLS n'est pas un algorithme hybride comme [6, 7]. C'est une méthode de recherche locale stochastique. La propagation unitaire est uniquement utilisée pour construire et analyser le graphe conflit et ainsi générer des nogoods.

L'algorithme CDLS (algorithme 2) prend en paramètre une CNF Σ et peut retourner trois valeurs, SAT, $UNSAT$ ou $UNKNOWN$. L'algorithme reprend le schéma de base de WSAT (voir algorithme 1). Notons que les interprétations initiales sont construites en utilisant la propagation unitaire (ligne 2 de l'algorithme CDLS et fonction `interpretationCompletePU`). En effet, l'utilisation de la propagation unitaire pour initialiser l'interpération courante de WSAT permet de s'approcher plus rapidement d'un minimum local et de prendre en compte certaines dépendances fonctionnelles entre les variables. Tant qu'une descente est possible on flippe une variable permettant de diminuer le nombre de clauses fausses. Une fois, un minimum local atteint, on analyse le conflit comme expliqué précédemment afin de générer un nogood β qui est ajouté à la base de clauses. La fonction `analyseConflitRL` est au coeur de notre contribution. Cette fonction commence par sélectionner un ensemble des variables conflits qui vont nous permettre de construire une interprétation partielle dérivée conduisant à un conflit. C'est la fonction `ensembleConflit` qui s'en charge. Elle choisit les variables dans les clauses liées à la clause falsifiée α, afin de forcer l'interprétation partielle qui va en être déduite, à rester dans le voisinage de α. La fonction `BCP` effectue la propagation unitaire. Elle renvoie une clause falsifiée si la propagation uni-

Fonction `interpretationCompletePU`

Input : Σ une formule CNF
Output : $\mathcal{I}_c$ une interprétation complète de Σ

1 $\Sigma' \leftarrow \Sigma$;
2 $\mathcal{I}_c \leftarrow \emptyset$;
3 **while** $|\mathcal{I}_c| \neq |\mathcal{V}_\Sigma|$ **do**
4 $\quad \mathcal{I}_c \leftarrow \mathcal{I}_c \cup \{x | x \in \mathcal{V}_\Sigma \setminus \mathcal{I}_c\}$;
5 $\quad \mathcal{P} \leftarrow \{x\} \cup \{y |\ \Sigma'_{|_x} \models_* y\}$;
6 $\quad$ **foreach** $y \in \mathcal{P}$ **do**
7 $\quad\quad$ **if** $\overline{y} \in \mathcal{I}_c$ **then**
8 $\quad\quad\quad \mathcal{P} \leftarrow \mathcal{P} \setminus \{y\}$;
9 $\quad \mathcal{I}_c \leftarrow \mathcal{I}_c \cup \mathcal{P}$;
10 $\quad \Sigma' \leftarrow \Sigma'_{|\mathcal{P}}$;
11 **return** $\mathcal{I}_c$;

Fonction `ensembleConflit`

Input : - Σ une formule CNF ;
- $\mathcal{I}_c$ une interprétation complète ;
- $\alpha \in \Sigma$ une clause critique pour $\mathcal{I}_c$.

Output : $\mathcal{C}$ un ensemble de littéraux conflit

1 $\mathcal{C} \leftarrow \emptyset$;
2 **forall** $x \in \overline{\alpha}$ **do**
3 $\quad \beta \in \Sigma$ unisatisfaite en x;
4 $\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{\beta \setminus \{x\}\}$;
5 **return** $\mathcal{C}$;

taire conduit à un conflit et la clause vide sinon. C'est cette fonction qui permet de construire ensuite l'interprétation partielle $\mathcal{I}_p$ (ligne 4-7). L'interprétation partielle conduit donc, soit à un conflit par propagation unitaire, soit à une discordance entre l'interprétation partielle et l'interprétation complète. Dans le premier cas, une analyse de conflit classique est effectuée (ligne 9) et le littéral assertif est flippé. Dans le second, on extrait l'interprétation partielle dérivée de $\mathcal{I}_p$ et on génère le graphe conflit associé (ligne 12-14 et proposition 5). À partir de là, il est possible d'analyser le conflit. Toutes les variables qui diffèrent de valeur entre l'interprétation partielle et l'interprétation complète sont alors flippées.

5 Expérimentations

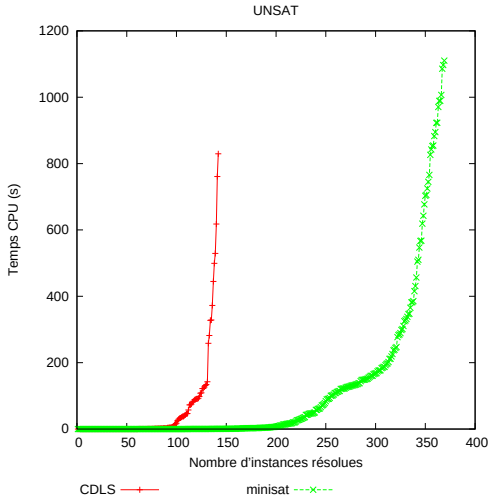
Les résultats expérimentaux reportés dans cette section ont été obtenus sur un Xeon 3.2 GHz (2 Go RAM). Le temps CPU est limité à 1200 secondes et les résultats sont reportés en secondes. Nous comparons CDLS à 3 méthodes incomplètes : WSAT [16], rsaps [11] et adaptg2 [12], et à minisat [5] un des solveurs complets CDCL les plus efficaces. Les instances utilisées sont issues des dernières compétitions SAT et sont divisées en trois catégories : *crafted* (1439 instances), *industrial* (1305) et *random* (2172). Toutes les instances sont prétraitées à l'aide de SATElite [4].

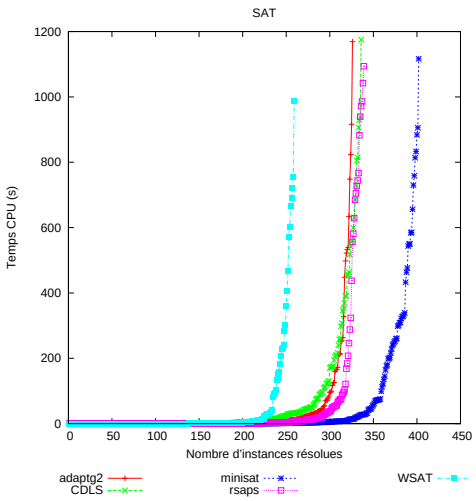
La figure 5 montre, pour chaque catégorie d'instances et en fonction de la satisfaisabilité de celles-ci, le nombre d'instances résolues en fonction du temps. Rappelons, tout d'abord que adaptg2, rsaps et WSAT sont des méthodes qui ne peuvent répondre qu'à la satisfaisabilité d'une instance. En ce qui concerne la catégorie *crafted*, le solveur minisat mis à part, notre approche est compétitive et résoud sensiblement le même nombre d'instances que les approches de recherche locale récentes adaptg2 et rsaps. Deux autres points sont à noter. D'une part, CDLS résoud beaucoup plus d'instances SAT que WSAT sur lequel il est basé. De l'autre, CDLS est capable de résoudre un grand nombre d'instances UNSAT.

Pour les instances de la catégorie *industrial*, CDLS résoud deux fois plus d'instances SAT que les solveurs stochastiques classiques et est capable de résoudre près de 160 instances UNSAT. L'analyse des conflits dans le cadre de la recherche locale permet donc de résoudre efficacement des instances structurées, qu'elles soient satisfaisables ou pas.

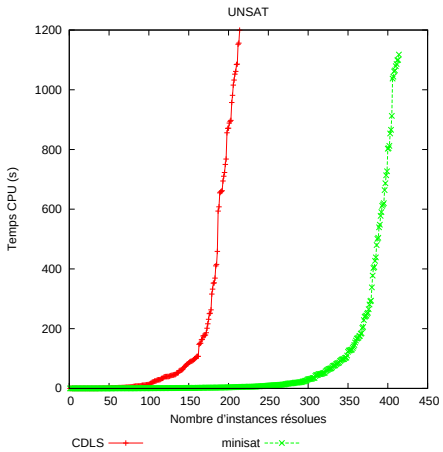
Enfin, concernant la catégorie *random*, notons, tout d'abord, que notre méthode n'arrive pas à prouver l'insatisfaisabilité pour les instances de cette catégorie. Nous pensons que cela provient du choix heuristique de l'ensemble conflit (voir fonction `ensembleConflit`). Il semble que CDLS ne se focalise pas assez sur le même sous-ensemble inconsistant. Au niveau des instances random satisafaisables, CDLS est le moins efficace des solveurs stochastiques. Ceci peut s'expliquer par le fait que la construction du graphe conflit est coûteuse en temps et qu'elle ralentit donc notre méthode. De ce fait, CDLS parcourt moins d'interprétations que les autres méthodes de recherche locale et a donc moins de chance de trouver un modèle. Notons également les mauvaises performances de minisat sur ce type d'instances.

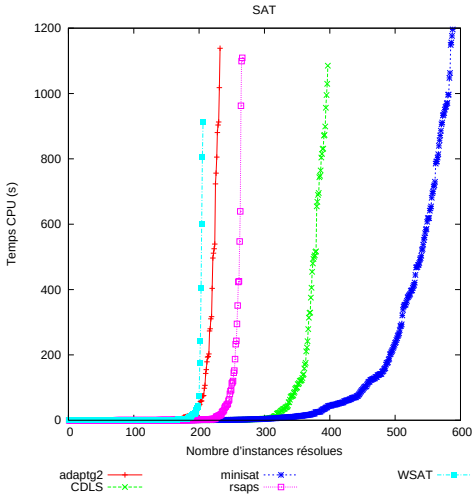
Globalement, CDLS est plus efficace que les autres solveurs basés sur la recherche locale. Il améliore grandement les performances de WSAT sur lequel il est basé. Il serait donc intéressant de greffer notre analyse de conflits à d'autres types d'algorithme de recherche locale, comme rsaps, afin d'améliorer leur performances. Il reste que CDLS ne rivalise pas encore avec les approches CDCL comme minisat dans les catégories crafted et industriels. Néanmoins, les premiers résultats obtenus sont extrêmement encourageants.



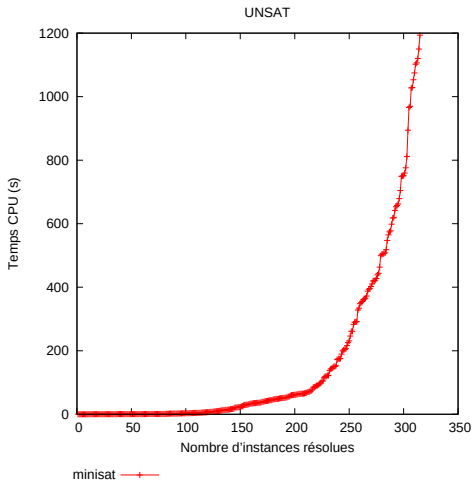


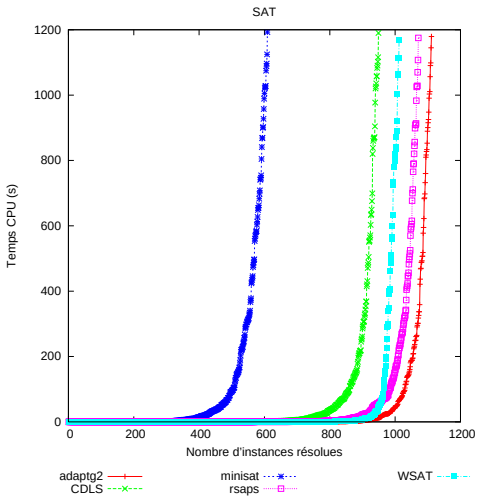
catégorie CRAFTED





catégorie INDUSTRIAL





catégorie RANDOM

FIG. 5 – Nombre d'instances résolues en fonction du temps

6 Conclusion

Dans cet article nous proposons une nouvelle approche pour sortir des minimums locaux dans le cadre des méthodes de recherche locale pour SAT. Notre approche est basée sur une extension de l'analyse des conflits utilisé par les solveurs SAT modernes complets. L'objectif est double : améliorer les méthodes de recherche locale sur les problèmes structurés et répondre à un des challenges les plus importants de la communauté SAT, à savoir, proposer une méthode incomplète efficace répondant à l'insatisfaisabilité d'une formule.

Les premiers résultats obtenus sur un large panel d'instances sont très encourageants. Notre solveur CDLS résout beaucoup d'instances insatisfaisables, de plus, il est beaucoup plus efficace que les autres méthodes de recherche pour résoudre des problèmes industriels. Dans les travaux futurs, nous comptons améliorer l'heuristique de choix des variables appartenant à l'ensemble conflit. Nous souhaitons également analyser et extraire les nogoods sans avoir recours à la propagation unitaire. Enfin, l'utilisation du premier UIP n'est pas primordial ici, et nous allons étudier diverses pistes pour choisir le meilleur nogood à extraire.

Références

[1] Gilles Audemard, Lucas Bordeaux, Youssef Hamadi, Saïd Jabbour, and Lakhdar Saïs. Un cadre général pour l'analyse de conflits. Dans *Journées Francophones de Programmation par Contraintes*, pages 267–276, 2008.

[2] Byungki Cha and Kazuo Iwama. Adding new clauses for faster local search. Dans *AAAI/IAAI, Vol. 1*, pages 332–337, 1996.

[3] Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7) :394–397, 1962.

[4] N. Eén and A. Biere. Effective preprocessing in SAT through variable and clause elimination. Dans *SAT*, pages 61–75, 2005.

[5] Niklas Eén and Niklas Sörensson. An extensible sat-solver. Dans *SAT*, pages 502–518, 2003.

[6] Lei Fang and Michael S. Hsiao. A new hybrid solution to boost sat solver performance. Dans *2007 Design, Automation and Test in Europe Conference and Exposition (DATE 2007)*, pages 1307–1313, 2007.

[7] Eugene Goldberg. A decision-making procedure for resolution-based sat-solvers. Dans *SAT*, pages 119–132, 2008.

[8] Eugene Goldberg and Yakov Novikov. Berkmin : A fast and robust sat-solver. *Discrete Applied Mathematics*, 155(12) :1549–1561, 2007.

[9] Eric Gregoire, Bertrand Mazure, and Cedric Piette. Extracting muses. Dans *European Conference on Artificial Intelligence (ECAI'06)*, pages 387–391, Trento (Italy), August 2006.

[10] Edward A. Hirsch and Arist Kojevnikov. Unitwalk : A new sat solver that uses local search guided by unit clause elimination. *Ann. Math. Artif. Intell.*, 43(1) :91–111, 2005.

[11] Frank Hutter, Dave A. D. Tompkins, and Holger H. Hoos. Scaling and probabilistic smoothing : Efficient dynamic local search for sat. Dans *CP*, pages 233–248, 2002.

[12] Chu Min Li, Wanxia Wei, and Harry Zhang. Combining adaptive noise and look-ahead in local search for sat. Dans *SAT*, pages 121–133, 2007.

[13] Bertrand Mazure, Lakhdar Saïs, and Eric Grégoire. Tabu search for sat. Dans *AAAI*, pages 281–285, juillet 1997.

[14] Lakhdar Sais, editeur. *Problème SAT : Progrès et Défis*, chapitre 6. Hermes, 2008.

[15] Bart Selman and Henry A. Kautz. An empirical study of greedy local search for satisfiability testing. Dans *AAAI*, pages 46–51, 1993.

[16] Bart Selman, Henry A. Kautz, and Bram Cohen. Noise strategies for improving local search. Dans *AAAI*, pages 337–343, 1994.

[17] Bart Selman, Henry A. Kautz, and David A. McAllester. Ten challenges in propositional reasoning and search. Dans *IJCAI (1)*, pages 50–54, 1997.

[18] Lintao Zhang, Conor F. Madigan, Matthew W. Moskewicz, and Sharad Malik. Efficient conflict driven learning in boolean satisfiability solver. Dans *ICCAD*, pages 279–285, 2001.