

Un cadre général pour l'analyse de conflits

G. Audemard¹ L. Bordeaux² Y. Hamadi² S. Jabbour¹ Lakhdar Sais¹

¹ CRIL-CNRS, Université d'Artois
rue Jean Souvraz SP18
F-62307 Lens Cedex France
{audemard, jabbour, sais}@cril.fr

² Microsoft Research
7 J J Thomson Avenue
Cambridge, United Kingdom
{lucasb, yousefjh}@microsoft.com

Résumé

Cet article présente plusieurs contributions au "Conflict Driven Clauses Learning" (CDCL), qui est une des composantes clés des solveurs SAT modernes. Tout d'abord, nous montrons que, à partir du graphe d'implication, les clauses assertives obtenues en utilisant le principe du premier point d'implication unique ("First Unique Implication Point" (FUIP)) sont optimales en terme de saut arrière. Puis nous proposons une extension du graphe d'implication contenant de nouveaux arcs appelés arcs arrières. Ces arcs sont obtenus en tenant compte des clauses satisfaites qui sont habituellement ignorées par l'analyse du conflit. Cette extension capture plus fidèlement l'ensemble du processus de propagation et ouvre de nouvelles perspectives pour les approches fondées sur CDCL. Entre autres avantages, notre extension du graphe d'implication conduit à un nouveau schéma d'analyse des conflits qui exploite les arcs ajoutés et permet des retours arrières plus haut dans l'arbre de recherche. Les résultats expérimentaux montrent que l'intégration de notre système d'analyse des conflits généralisés au sein de solveurs dernier-cri améliore sensiblement leurs performances.

1 Introduction

Cet article présente plusieurs contributions à l'un des points clés des solveurs SAT modernes, à savoir les techniques de type CDCL ("Conflict Driven-Clause Learning") [10, 6]. La structure de donnée principale d'un solveur basé sur CDCL est le graphe d'implication qui enregistre les différentes propagations faites durant la construction de l'interprétation partielle. Ce graphe d'implication permet d'analyser les conflits, il est utilisé pour faire du retour arrière ("backtracking") intelligent, pour apprendre des no-goods et pour mettre à jour les poids des variables dans les heuristiques dynamiques de type VSIDS [10, 7]. Il est

important de noter que ce graphe est construit de manière incomplète, il ne donne qu'une vue partielle des implications entre les littéraux. Regardons par exemple ce qu'il se passe lorsque un littéral y est déduit à un niveau donné. Cette déduction est faite parce qu'une clause, par exemple $(\neg x_1 \vee \neg x_2 \vee y)$, est devenue unitaire sous l'interprétation courante. Cette dernière contient donc les affectations $x_1 = true$ et $x_2 = true$ qui ont eu lieu à des niveaux inférieurs ou égaux au niveau de l'affectation de y . Lorsque y est impliqué, la raison de cette implication est enregistrée, on ajoute donc au graphe d'implication des arcs entre x_1 et y et entre x_2 et y . Supposons maintenant que $(x_1 \vee y)$ soit une autre clause de la formule, x_1 est donc une conséquence de l'affectation de y . Il serait correct d'ajouter l'arc (y, x_1) au graphe d'implication, mais cela n'est pas le cas. En effet, seule la première explication (clause) rencontrée d'un littéral déduit est enregistrée dans le graphe. Cette stratégie est donc fortement dépendante de l'ordre des clauses dans la formule. Nous proposons une extension du graphe d'implications dans lequel un littéral déduit peut avoir plusieurs explications. Contrairement au cas traditionnel, les arcs peuvent aller en arrière par rapport au niveau d'affectation, c'est le cas par exemple si nous rajoutons l'arc (y, x_1) , puisque y a un niveau supérieur (il est propagé après) celui de x_1 . Ces arcs particuliers sont appelés *arcs arrières*.

Nous proposons donc une extension forte des graphes d'implications, basée sur la notion de graphe d'implications généralisé que nous venons de présenter de manière informelle. Nous commençons par prouver que pour un graphe d'implication donné, la clause conflit assertive générée en utilisant le principe du premier UIP (*Unique implication point*) est optimale en terme de saut. Nous formalisons ensuite la notion de graphe d'implication étendu et étudions son utilisation lors de l'analyse de conflits. Les arcs arrières permettent de détecter des raisons pour des va-

riables de décision, ce qui ne peut être fait avec un graphe d'implications classique. Cela ouvre des nouvelles perspectives pour les solveurs de type CDCL. Les arcs arrières peuvent servir à minimiser les clauses assertives déduites du graphe d'implication classique ou à améliorer le saut arrière. Ces arcs peuvent également être utilisés comme raisons alternatives lors de la génération de la clause assertive.

Le reste de l'article est organisé comme suit. Après quelques définitions et notations (section 2), le graphe d'implication classique et son utilisation pour générer des nogoods sont présentés en section 3. Nous prouvons dans la section 4 l'optimalité en termes de saut de la clause assertive générée. Dans la section 5, nous introduisons le graphe d'implication étendu. Nous montrons ensuite comment utiliser les arcs arrières pour améliorer les sauts lors du retour arrière (section 6) et comment l'intégrer à des solveurs de type CDCL comme Rsat [11] ou Minisat [5]. Avant de conclure, nous donnons des résultats expérimentaux dans la section 7.

2 Définitions et notations

Une *formule CNF* \mathcal{F} est un ensemble (interprété comme une conjonction) de *clauses*, où chaque clause est un ensemble (interprété comme une disjonction) de *littéraux*. Un littéral est soit positif (x) ou négatif ($\neg x$). Les deux littéraux x et $\neg x$ sont appelés *complémentaires*. On note également $\neg l$ (ou \bar{l}) le littéral complémentaire de l . Pour un ensemble de littéraux L , \bar{L} désigne l'ensemble $\{\bar{l} \mid l \in L\}$. Une clause est dite *unitaire* si elle contient exactement un seul littéral (appelé *littéral unitaire*), et dans le cas d'une clause contenant seulement deux littéraux celle-ci est dite binaire. La *clause vide*, notée \perp , est interprétée comme fausse (insatisfaisable), tandis qu'une *formule CNF vide*, notée \top , est interprétée comme vraie (satisfaisable). L'ensemble des littéraux apparaissant dans \mathcal{F} est noté $V_{\mathcal{F}}$. Une *interprétation* ρ d'une formule booléenne \mathcal{F} associe une valeur $\rho(x)$ aux variables $x \in \mathcal{F}$. L'interprétation ρ est dite *complète* si elle associe une valeur à chaque $x \in \mathcal{F}$, sinon elle est dite *partielle*. Une interprétation peut également être représentée par un ensemble de littéraux. Un *modèle* d'une formule \mathcal{F} est une interprétation ρ qui satisfait la formule, ce qui est noté $\rho \models \Sigma$. Les notations suivantes seront largement utilisées par la suite :

- $\eta[x, c_i, c_j]$ dénote la *résolvante* entre la clause c_i contenant le littéral x et la clause c_j contenant l'opposé de ce même littéral $\neg x$. Autrement dit $\eta[x, c_i, c_j] = c_i \cup c_j \setminus \{x, \neg x\}$. Une résolvante est appelée *tautologique* si elle contient à la fois un littéral et son opposé.
- $\mathcal{F}|_x$ dénote la formule \mathcal{F} simplifiée par l'affectation du littéral x à la valeur *true*. Formellement $\mathcal{F}|_x = \{c \mid c \in \mathcal{F}, \{x, \neg x\} \cap c = \emptyset\} \cup \{c \setminus \{\neg x\} \mid c \in \mathcal{F}, \neg x \in c\}$ (les clauses contenant x et qui sont donc satisfaites sont supprimées ; et celles contenant $\neg x$ sont simpli-

fiées). Cette notation est étendue aux interprétations : soit $\rho = \{x_1, \dots, x_n\}$ une interprétation, on définit $\mathcal{F}|_{\rho} = (\dots((\mathcal{F}|_{x_1})|_{x_2}) \dots |_{x_n})$.

- \mathcal{F}^* dénote la formule close \mathcal{F} après application de la propagation unitaire, définie récursivement comme suit : (1) $\mathcal{F}^* = \mathcal{F}$ si \mathcal{F} ne contient aucune clause unitaire. (2) $\mathcal{F}^* = \perp$ si \mathcal{F} contient deux clauses unitaires $\{x\}$ et $\{\neg x\}$, (3) autrement, $\mathcal{F}^* = (\mathcal{F}|_x)^*$ tel que x est le littéral qui apparaît comme clause unitaire dans \mathcal{F} .
- \models_* dénote la déduction logique par propagation unitaire : $\mathcal{F} \models_* x$ signifie que le littéral x est déduit par propagation à partir de \mathcal{F} , i.e. $x \in \mathcal{F}^*$. On écrit $\mathcal{F} \models_* \perp$ si la formule est inconsistance (insatisfaisable) par propagation. En particulier, si on dispose d'une clause c tel que $\mathcal{F} \wedge \bar{c} \models_* \perp$, alors c est une conséquence logique de \mathcal{F} . On dit alors que c est déduite *par réfutation*.

Nous introduisons maintenant des notations et remarques terminologiques associées aux solveurs SAT basés sur la procédure de Davis Logemann Loveland, communément appelé DPLL [2]. DPLL est une procédure de recherche de type "*backtrack*"; A chaque noeud les littéraux affectés (le littéral de décision et les littéraux propagés) sont étiquetés avec le même *niveau de décision*, initialisé à 1 et incrémenté à chaque nouveau point de décision. Le niveau de décision courant est le niveau le plus élevé dans la pile de propagation. Lors d'un retour arrière ("*backtrack*"), les variables ayant un niveau supérieur au niveau du backtrack sont désaffectées et le niveau de décision courant est décrémenté en conséquence (égal au niveau du backtrack). Au niveau i , l'interprétation partielle courante ρ peut être représentée comme une séquence décision-propagations de la forme $\langle (x_k^i), x_{k_1}^i, x_{k_2}^i, \dots, x_{k_{n_k}}^i \rangle$ telle que le premier littéral x_k^i correspond au littéral de décision x_k affecté au niveau i et chaque $x_{k_j}^i$ de l'ensemble $1 \leq j \leq n_k$ représente les littéraux unitaires propagés à ce même niveau i .

Soit $x \in \rho$, On note $l(x)$ le niveau d'affectation de x , $d(\rho, i) = x$ si x est le littéral de décision affecté au niveau i . Pour un niveau donné i , ρ^i représente la projection de ρ aux littéraux affectés au niveau $\leq i$.

3 Analyse du Conflit et Graphe d'Implication

Les graphes d'implication sont des représentations capturant les variables affectées durant la recherche, que ce soit des variables de décision ou des variables propagées. Cette représentation est utile pour analyser les conflits. A chaque fois qu'un littéral y est propagé, on garde en mémoire la clause à l'origine de cette propagation, clause que l'on note $\overrightarrow{cl}_a(y)$. La clause $\overrightarrow{cl}_a(y)$ est donc de la forme $(x_1 \vee \dots \vee x_n \vee y)$ où chaque littéral x_i est faux sous

l'interprétation courante ($\forall i \in 1 \dots n \ \rho(x_i) = false$) et $\rho(y) = true$. Quand un littéral y n'est pas obtenu par propagation unitaire mais qu'il correspond à un point de choix, $\vec{cla}(y)$ est indéfini, que l'on note par convention $\vec{cla}(y) = \perp$.

Lorsque $\vec{cla}(y) \neq \perp$, on note par $exp(y)$ l'ensemble $\{\bar{x} \mid x \in \vec{cla}(y) \setminus \{y\}\}$, appelé l'ensemble des *explications* de y . Autrement dit, si $\vec{cla}(y) = (x_1 \vee \dots \vee x_n \vee y)$ alors les explications sont les littéraux \bar{x}_i qui constituent la condition sous laquelle $\vec{cla}(y)$ devient une clause unitaire $\{y\}$. Notons que pour tout i on a $l(\bar{x}_i) \leq l(y)$, i.e., toutes les explications de la déduction ont un niveau inférieur à celui de y .

Quand $\vec{cla}(y)$ est indéfini, $exp(y)$ est égal à l'ensemble vide. Les explications peuvent, alternativement, être vues comme un graphe d'implication dans lequel l'ensemble des prédécesseurs d'un noeud correspond aux explications du littéral correspondant :

Définition 1 (Graphe d'implication) Soient \mathcal{F} une formule CNF et ρ une interprétation partielle. Le graphe d'implication associé à \mathcal{F} , ρ et exp est $(\mathcal{N}, \mathcal{E})$ tel que :

- $\mathcal{N} = \rho$, i.e. il existe exactement un seul noeud pour chaque littéral, de décision ou impliqué ;
- $\mathcal{E} = \{(x, y) \mid x \in \rho, y \in \rho, x \in exp(y)\}$

Exemple 1 On considère une formule \mathcal{F} contenant, entre autres, les clauses suivantes ($\mathcal{F} \supseteq \{c_1, \dots, c_9\}$) :

$$\begin{array}{ll} (c_1) & x_6 \vee \neg x_{11} \vee \neg x_{12} & (c_2) & \neg x_{11} \vee x_{13} \vee x_{16} \\ (c_3) & x_{12} \vee \neg x_{16} \vee \neg x_2 & (c_4) & \neg x_4 \vee x_2 \vee \neg x_{10} \\ (c_5) & \neg x_8 \vee x_{10} \vee x_1 & (c_6) & x_{10} \vee x_3 \\ (c_7) & x_{10} \vee \neg x_5 & (c_9) & \neg x_3 \vee \neg x_{19} \vee \neg x_{18} \\ (c_8) & x_{17} \vee \neg x_1 \vee \neg x_3 \vee x_5 \vee x_{18} & & \end{array}$$

Soit ρ l'interprétation partielle suivante $\rho = \{(\dots \neg x_6^1 \dots \neg x_{17}^1) \langle (x_8^2) \dots \neg x_{13}^2 \dots \rangle \langle (x_4^3) \dots x_{19}^3 \dots \rangle \dots \langle (x_{11}^5) \dots \rangle\}$. Le niveau de décision courant est 5. Le graphe d'implication $\mathcal{G}_{\mathcal{F}}^{\rho}$ associé à \mathcal{F} et ρ est schématisé dans la figure 1 (notez que seule une partie du graphe est affichée).

3.1 Generation des clauses assertives

Dans cette section, on décrit formellement le schéma d'apprentissage classique utilisé dans les solveurs SAT modernes. Dans les définitions suivantes, on considère \mathcal{F} une formule CNF, ρ une interprétation partielle telle que $(\mathcal{F}|_{\rho})^* = \perp$ et $\mathcal{G}_{\mathcal{F}}^{\rho} = (\mathcal{N}, \mathcal{E})$ le graphe d'implication associé. Supposons que le niveau de décision courant est m , puisqu'on a atteint le conflit, il existe donc un littéral z tel que $\{z, \neg z\} \subset \mathcal{N}$ et $l(z) = m$ ou $l(\neg z) = m$. L'analyse du conflit est basée sur l'application de la résolution (par z)

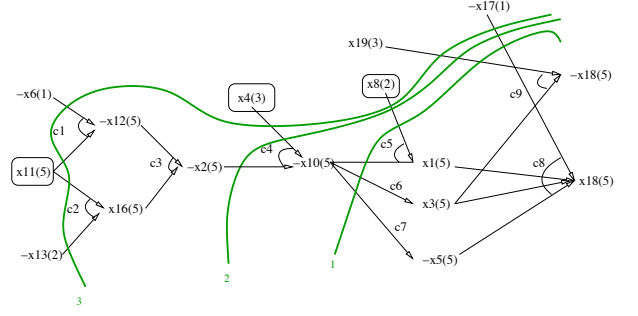


FIG. 1 – Graphe d'implication $\mathcal{G}_{\mathcal{F}}^{\rho} = (\mathcal{N}, \mathcal{E})$

sur la clause devenue fautive et en remontant dans le graphe d'implication, et ceci en utilisant les clauses $(exp(y) \vee y)$ à chaque noeud $y \in \mathcal{N}$. Le processus s'arrête lorsqu'il ne reste plus qu'un littéral x du dernier niveau 'affectation. On appelle ce processus la preuve par résolution basé sur les conflits. Définissons formellement les différents concepts de clause assertive, de preuve par résolution basée sur les conflits et de point d'implication unique (*Unique Implication Point*).

Définition 2 (Clause assertive) Une clause c de la forme $(\alpha \vee x)$ est dite une clause assertive ssi $\rho(c) = false$, $l(x) = m$ et $\forall y \in \alpha, l(y) < l(x)$. x est appelé littéral assertif, qu'on notera $\mathcal{A}(c)$. On définit également $jump(c) = max\{l(\neg y) \mid y \in \alpha\}$.

Définition 3 (Preuve par résolution basée sur les conflits)

Une preuve par résolution basée sur les conflits π est une séquence de clause $\langle \sigma_1, \sigma_2, \dots, \sigma_k \rangle$ qui satisfait les conditions suivantes :

1. $\sigma_1 = \eta[x, \vec{cla}(z), \vec{cla}(\neg z)]$, tel que $\{z, \neg z\}$ est un conflit.
2. pour tout $i \in 2..k$, σ_i est construit en sélectionnant un littéral $y \in \sigma_{i-1}$ pour lequel $\vec{cla}(\bar{y})$ est défini. On a alors $y \in \sigma_{i-1}$ et $\bar{y} \in \vec{cla}(\bar{y})$, deux clauses pouvant entrer en résolution. La clause σ_i est définie comme $\eta[y, \sigma_{i-1}, \vec{cla}(\bar{y})]$;
3. Enfin, σ_k est une clause assertive

Notons que chaque σ_i est une résolvente de la formule \mathcal{F} : par induction, σ_1 est la résolvente entre deux clauses qui appartiennent à \mathcal{F} ; pour chaque $i > 1$, σ_i est une résolvente entre σ_{i-1} (qui, par hypothèse d'induction, est une résolvente) et une clause de \mathcal{F} . Chaque σ_i est aussi un *implicat* de \mathcal{F} , c'est à dire : $\mathcal{F} \models \sigma_i$. Une autre remarque importante est que dans les solveurs SAT modernes, les littéraux y utilisés dans la condition 2 de la définition 3 sont restreint à ceux du niveau courant.

Définition 4 (Preuve élémentaire) Une preuve par résolution basé sur les conflits $\pi = \langle \sigma_1, \sigma_2, \dots, \sigma_k \rangle$ est dite

élémentaire ssi $\exists i < k$ tel que $\langle \sigma_1, \sigma_2, \dots, \sigma_i \rangle$ soit aussi une preuve par résolution basée sur les conflits.

En utilisant les définitions 3 et 4, on peut désormais définir les concepts du point d'implication unique (*Unique Point Implication* (UIP)) et du premier UIP :

Définition 5 (Point d'Implication Unique (UIP)) Un noeud $x \in \mathcal{N}$ est un UIP ssi il existe une preuve par résolution basé sur les conflits $\langle \sigma_1, \sigma_2, \dots, \sigma_k \rangle$ tel que $\bar{x} \in \sigma_k$ et $l(x)$ est égal au niveau de décision courant, m (Notez que σ_k est assertive, et a exactement un seul x de la sorte).

Définition 6 (Premier UIP) Un noeud $x \in \mathcal{N}$ est un premier UIP ssi il est obtenu à partir d'une preuve élémentaire ; i.e. $\exists \langle \sigma_1, \sigma_2, \dots, \sigma_k \rangle$ une preuve par résolution basé sur les conflits tq. $\bar{x} \in \sigma_k$ et $l(x) = m$.

Définition 7 (Dernier UIP) Le dernier UIP est défini comme étant le littéral $d(\rho, m)$, i.e. le littéral de décision au niveau du conflit m .

Pour illustrer les définitions précédentes, considérons à nouveau l'exemple 1.

Le parcours du graphe $\mathcal{G}_{\mathcal{F}}^{\rho}$ nous permet de générer trois clauses assertives correspondant aux trois UIPs possibles (voir figure 1). Illustrons la preuve par résolution basée sur les conflits qui conduit à la première clause assertive Δ_1 correspondant au premier UIP (voir coupe 1 de la figure 1).

$$\begin{aligned} - \sigma_1 &= \eta[x_{18}, c_8, c_9] = (x_{17}^1 \vee \neg x_1^5 \vee \neg x_3^5 \vee x_5^5 \vee \neg x_{19}^3) \\ - \sigma_2 &= \eta[x_1, \sigma_1, c_5] = (x_{17}^1 \vee \neg x_3^5 \vee x_5^5 \vee \neg x_{19}^3 \vee \neg x_8^2 \vee x_{10}^5) \\ - \sigma_3 &= \eta[x_5, \sigma_2, c_7] = (x_{17}^1 \vee \neg x_3^5 \vee \neg x_{19}^3 \vee \neg x_8^2 \vee x_{10}^5) \\ - \sigma_4 &= \Delta_1 = \eta[x_3, \sigma_3, c_6] = (x_{17}^1 \vee \neg x_{19}^3 \vee \neg x_8^2 \vee x_{10}^5) \end{aligned}$$

Comme on peut le voir, σ_4 permet de déduire une première clause assertive (qu'on appellera également Δ_1) car tous ses littéraux sont affectés avant le niveau actuel excepté un (x_{10}) qui est affecté au niveau courant 5 ; $\langle \sigma_1, \sigma_2, \sigma_3, \sigma_4 \rangle$ est une preuve élémentaire par résolution basée sur les conflits, et $\neg x_{10}$ est le premier UIP.

Si on continue un tel processus, on obtient en plus les deux clauses assertives $\Delta_2 = (x_{17}^1 \vee \neg x_{19}^3 \vee \neg x_8^2 \vee \neg x_4^3 \vee x_2^5)$, correspondant au second UIP $\neg x_2^5$; et $\Delta_3 = (x_{17}^1 \vee \neg x_{19}^3 \vee \neg x_8^2 \vee \neg x_4^3 \vee x_{13}^2 \vee x_6^1 \vee \neg x_{11}^5)$, correspondant au 3ième UIP ($\neg x_{11}^5$) mais également dernier UIP puisque x_{11} est la dernière variable de décision (voir coupes 2 and 3 dans la figure 1).

Propriété 1 (Clause assertive et retour arrière) Soit $c = (\alpha \vee x)$ une clause assertive déduite à partir de l'analyse du conflit et $i = \max\{l(\neg y) \mid y \in \alpha\}$. On peut sans danger faire un retour arrière au niveau i et considérer la nouvelle interprétation partielle $\rho^i \cup \{x\}$

Propriété 2 Soit $c = (\alpha \vee x)$ une clause assertive déduite à partir de l'analyse du conflit et $i = \max\{l(\neg y) \mid y \in \alpha\}$. alors x peut être déduite par réfutation au niveau i , i.e. $(\mathcal{F} \wedge \bar{x})|_{\rho^i} \models_* \perp$.

Preuve 1 L'affectation de tous les littéraux $y \in \alpha$ à faux conduit exactement au même conflit par propagation unitaire.

La propriété précédente montre que le littéral assertif peut être déduit par réfutation au niveau i . Cela signifie que si on décide de tester par propagation unitaire à chaque niveau un ensemble de littéraux (lookahead local), alors on peut détecter ce conflit bien avant d'atteindre un niveau plus bas dans l'arbre de recherche.

La vraie difficulté, cependant, réside dans la manière de sélectionner efficacement les littéraux à pré-traiter par propagation unitaire. Une tentative de réponse à cette question est donnée dans les approches proposées par Dubois et al, [3] et par Li et Anbulagan [8].

Cette propriété simple montre que les solveurs SAT modernes explorent un arbre de recherche binaire, et peuvent être considérés comme une variante de la procédure DPLL. Ce point de vue n'est pas partagé par tous les membres de la communauté SAT.

4 Résultat sur l'optimalité de la clause assertive

Pour motiver notre extension, nous prouvons dans cette section une propriété fondamentale concernant l'optimalité de la clause assertive générée grâce au premier UIP du schéma d'apprentissage : il est optimal en terme de niveau de backjumping¹. Cette simple propriété montre pourquoi le premier UIP habituellement considéré dans les schémas d'apprentissage classique est plus puissant que les autres UIPs.

Soit c une clause, on note $levels(c)$ l'ensemble des différents niveaux présents dans la clause c , i.e. $levels(c) = \{l(x) \mid x \in c\}$.

Propriété 3 Dans une preuve par résolution basé sur les conflits $\langle c_1, \dots, c_k \rangle$ on a pour tout $i < k$, $levels(c_i) \subseteq levels(c_{i+1})$.

Preuve 2 Chaque σ_{i+1} est obtenu à partir de σ_i en sélectionnant un littéral $x \in \sigma_i$ qui n'est pas un littéral de décision et en le remplaçant par l'ensemble de ces explications $exp(x)$. Ces explications contiennent toujours d'autres littéraux y tel que $l(y) = l(x)$.

¹Après discussions privées [9], il apparaît que ce résultat fait parti du folklore de la communauté, mais, d'après nos connaissances, il semble que ce soit la première fois que cela est formellement établi dans un article

Propriété 4 Soit $c = (\alpha, x)$ une clause assertive obtenue par preuve par résolution sur le conflit et dans laquelle x est le premier UIP. Le niveau du backjumping de c est optimal : toute autre assertive clause c' obtenue par résolution sur les conflits est telle que $\text{jump}(c') \geq \text{jump}(c)$.

Preuve 3 Il peut être démontré que toutes les clauses assertives contenant le premier UIP ont le même niveau de backjump (en général, il peut y avoir plusieurs clauses de telle sorte, mais leurs ensembles de niveaux peut être incomparables). Soit (c_1, \dots, c') une résolution basée sur les conflits de c' . Soit i l'index de la première clause assertive dans cette preuve. Comme c_i est une clause assertive, on a $\text{jump}(c) = \text{jump}(c_i)$ et en utilisant la propriété 3, on a $\text{jump}(c_i) \leq \text{jump}(c')$.

Notons que le premier UIP est le seul qui garantit ces deux points d'optimalité. On peut construire aisément des exemples dans lesquels n'importe quelle clause assertive associée à un UIP autre que le premier a un niveau de backjump strictement plus faible que celui du premier UIP.

5 Graphe d'implication étendu

On montre dans cette section comment les arcs provenant des clauses satisfaites peuvent être ajoutés au graphe d'implication et peuvent être exploités avantageusement². En particulier, ces arcs permettent parfois d'améliorer le niveau du backjump. Comme montré dans la section précédente, c'est quelque chose d'impossible si l'on considère le graphe d'implication classique.

5.1 Illustration

Dans les solveurs SAT modernes, seule la première clause $\text{exp}(x)$ rencontrée et impliquant un littéral x est prise en compte dans l'analyse. Or ce même littéral peut être impliqué par d'autres clauses et peut donc posséder plusieurs explications. Cependant ces dernières sont ignorées dans les solveurs. Ceci est dû principalement à une question de performances. C'est justement ces clauses qui représentent des arcs particuliers dans le graphe d'implication (qui ne sont pas représentés jusqu'à maintenant) qui nous intéressent et qui constituent la base de notre travail.

Pour expliquer notre idée considérons à nouveau la formule \mathcal{F} de l'exemple 1 On considère exactement la même interprétation partielle et on définit la nouvelle formule \mathcal{F}' comme suit : $\mathcal{F}' \supseteq \{c_1, \dots, c_9\} \cup \{c_{10}, c_{11}, c_{12}\}$ avec $c_{10} = (\neg x_{19} \vee x_8)$, $c_{11} = (x_{19} \vee x_{10})$ et $c_{12} = (\neg x_{17} \vee x_{10})$.

²Un notion de graphe d'implication étendu dans lequel plusieurs clauses "explicatives" sont maintenues pour chaque littéral a déjà été introduite dans la littérature [1]. Néanmoins, Le but de ce travail était théorique alors que le notre est de montrer que les arcs arrières ont un intérêt pratique

Les trois clauses ajoutées sont satisfaites par l'interprétation ρ . c_{10} est satisfaite par x_8 affecté au niveau 2, c_{11} est satisfaite par le littéral x_{19} affecté au niveau 3 quand à c_{12} elle est satisfaite par l'affectation de $\neg x_{17}$ au niveau 1. Ceci est décrit dans le graphe d'implication étendu de la figure 2 grâce aux arcs en pointillé. Illustrons maintenant l'utilité de notre extension proposée. Considérons une nouvelle fois la clause assertive Δ_1 qui correspond au premier UIP classique.

On peut générer la clause assertive forte suivante :

- $c_{13} = \eta[x_8, \Delta_1, c_{10}] = (x_{17}^1 \vee \neg x_{19}^3 \vee x_{10}^5)$
- $c_{14} = \eta[x_{19}, c_{13}, c_{11}] = (x_{17}^1 \vee x_{10}^5)$
- $\Delta_1^s = \eta[x_{17}, c_{14}, c_{12}] = x_{10}^5$

Dans ce cas on saute au niveau 0 et on affecte x_{10} à true. En effet $\mathcal{F}' \models x_{10}$.

Comme on peut le voir Δ_1^s subsume Δ_1 . Si on continue le processus, on obtient d'autres clauses assertives fortes $\Delta_2^s = (\neg x_4^3 \vee x_2^5)$ et $\Delta_3^s = (\neg x_4^3 \vee x_{13}^2 \vee x_6^1 \vee \neg x_{11}^5)$ qui subsument respectivement Δ_2 et Δ_3 . Cette première illustration permet d'obtenir une nouvelle voie pour minimiser la taille de la clause assertive.

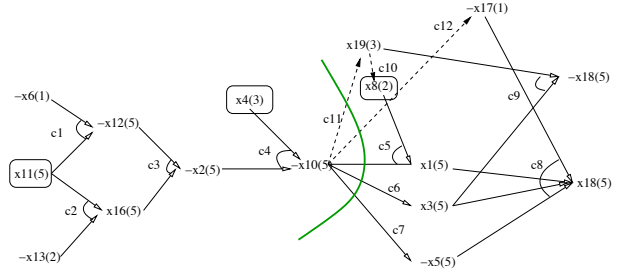


FIG. 2 – Graphe d'implication étendu : $\mathcal{G}_{\mathcal{F}'}^{\rho} = (\mathcal{N}', \mathcal{E}')$

Si nous jetons un coup d'oeil aux clauses utilisées dans le graphe d'implication classique $\mathcal{G}_{\mathcal{F}}^{\rho}$ (figure 1), toutes ont les propriétés suivantes : (1) $\forall x \in \mathcal{N}$ la clause $c = (\text{exp}(x) \vee x)$ est satisfaite par seulement un littéral i.e. $\rho(x) = \text{true}$ et $\forall y \in \text{exp}(x)$, On a $\rho(y) = \text{true}$ et (2) $\forall y \in \text{exp}(x)$, $l(\neg y) \leq l(x)$. Maintenant dans le graphe d'implication étendu (figure 2) les clauses ajoutées satisfont la propriété (1) et, en plus, la propriété (2') $\exists y \in \text{exp}(x)$ avec $l(\neg y) > l(x)$.

Expliquons brièvement comment ces arcs supplémentaires sont obtenus. Habituellement, la propagation unitaire ne garde pas de trace des implications apparaissant dans la sous-formule satisfaite par l'interprétation. Dans cette extension, on prend en compte la sous formule satisfaite pour déduire de nouvelles implications. Revenons à notre exemple, quand on déduit x_{19} au niveau 3, on découvre l'implication de x_8 (qui est le littéral de décision du niveau 2). De manière similaire, pour $\neg x_{10}$ déduit au niveau 5, on "redécouvre" les déductions x_{19} (faite au niveau 3) et $\neg x_{17}$ (faite au niveau 1).

Notre approche tient compte de ces déductions "redé-

couvertes". Notez quelque chose d'inhabituel : Les points de choix peuvent également avoir des arcs entrants (comme x_8 dans l'exemple), ce qui est impossible avec un graphe d'implication classique.

Avant d'introduire formellement la définition du graphe d'implication étendu, on va introduire le concept d'implication arrière (arc arrière).

On maintient en plus de la clause classique $\overrightarrow{cla}(x)$ une nouvelle clause $\overleftarrow{cla}(x)$ de la forme $(x \vee y_1 \vee \dots \vee y_n)$. Cette clause est sélectionnée de la façon suivante $\rho(y_i) = false$ pour tout $i \in 1 \dots n$; $\rho(x) = true$; et $\exists i. l(y_i) > l(x)$. Cette clause peut être indéfinie dans certains cas (ce qu'on note $\overleftarrow{cla}(x) = \perp$). Plusieurs clauses de cette forme peuvent être trouvées pour chaque littéral. Dans ce cas, une seule est sélectionnée arbitrairement : on peut par exemple choisir de considérer la première trouvée. (on peut toutefois définir des variantes si on veut tenir compte de toutes ces clauses, dans ce cas $\overleftarrow{cla}(x)$ est l'ensemble des clauses ; mais le but ici n'est pas de développer cette variante).

On note par $\overleftarrow{exp}(x)$ l'ensemble $\{\bar{y} \mid y \in \overleftarrow{cla}(x) \setminus \{x\}\}$, et, par soucis de clarté, par $\overrightarrow{exp}(x)$ l'ensemble précédemment noté exp . Un graphe d'implication étendu est défini comme suit (notons que ce graphe n'est pas acyclique dans le cas général) :

Définition 8 (Graphe d'implication étendu) Soit \mathcal{F} une formule CNF et ρ une interprétation partielle ordonnée. On définit le graphe d'implication étendu associé à \mathcal{F} et ρ comme $\mathcal{G}_{s_{\mathcal{F}}}^{\rho} = (\mathcal{N}, \mathcal{E} \cup \mathcal{E}')$ tel que,

- $\mathcal{N} = \rho$
- $\mathcal{E} = \{(x, y) \mid x \in \rho, y \in \rho, x \in \overrightarrow{exp}(y)\}$, $\mathcal{E}' = \{(x, y) \mid x \in \rho, y \in \rho, x \in \overleftarrow{exp}(y)\}$

6 de l'apprentissage au backjump : une première extension

Dans cette section, nous décrivons une première extension possible de l'approche CDCL utilisant le graphe d'implication étendu. Notre approche fait une utilisation originale des arcs arrières afin d'effectuer des sauts arrière intelligents plus importants, c'est-à-dire d'améliorer le niveau du backjumping généré par la clause assertive. Illustrons tout d'abord l'idée principale derrière cette extension. Notre approche est réalisée en trois étapes. Dans la première étape (1) : une clause assertive, dite $\sigma_1 = (\neg x^1 \vee \neg y^3 \vee \neg z^7 \vee \neg a^9)$ est apprise à partir du schéma d'apprentissage classique, le niveau de décision actuel est donc égal à 9. Comme $\rho(\sigma_1) = false$, habituellement on backtrack au niveau $jump(\sigma_1) = 7$. Dans la seconde étape (2), notre approche vise à éliminer le littéral $\neg z^7$ à partir de σ_1 en utilisant un arc arrière du graphe étendu. Expliquons à présent cette deuxième et nouvelle transformation. Soit $c = (z^7 \vee \neg u^2 \vee \neg v^9)$ tel que $\rho(z) = true$,

$\rho(u) = true$ et $\rho(v) = true$. La clause c est un arc arrière i.e. le littéral z affecté au niveau 7 est impliqué par les deux littéraux u et v affectés respectivement au niveau 2 et 9. A partir de c et de σ_1 , une nouvelle clause $\sigma_2 = \eta[z, c, \sigma_1] = (\neg x^1 \vee \neg u^2 \vee \neg y^3 \vee \neg v^9 \vee \neg a^9)$ est générée par résolution. On peut remarquer que la nouvelle clause σ_2 contient deux littéraux du niveau de décision actuel (9). En troisième étape (3), en utilisant le graphe d'implication classique, on peut faire une preuve par résolution basée sur les conflits à partir de σ_2 pour trouver une nouvelle clause assertive σ_3 avec un seul littéral du niveau de décision actuel.

Notons que cette nouvelle clause assertive σ_3 peut dégrader le niveau du backjumping. Pour éviter cet inconvénient majeur, l'arc arrière c est choisi sous conditions : i) les littéraux de c qui sont du niveau de décision actuel (v^9) sont déjà visités durant la première étape. et ii) tous les autres littéraux de c sont affectés avant le niveau 7 (niveau de z). Dans ce cas, on garantit que la nouvelle clause assertive satisfait la propriété suivante : $jump(\sigma_3) \leq jump(\sigma_1)$. Plus intéressant encore, le littéral assertif de σ_3 est $\neg a$.

On peut réitérer le processus précédant sur la nouvelle clause assertive σ_3 pour éliminer les littéraux de σ_3 affectés au niveau $jump(\sigma_3)$.

Afin d'introduire formellement notre approche, nous introduisons dans ce qui suit le concept d'arc arrière joint pour étendre la résolution basé sur les conflits classique.

Définition 9 (Arc arrière joint) Soit $\pi = \langle \sigma_1, \sigma_2, \dots, \sigma_k \rangle$ une preuve par résolution basé sur les conflits. Une clause $c \in \mathcal{F}$ est appelée arc arrière joint de π si les conditions suivantes sont satisfaites :

1. c est l'arc arrière associé à un des littéraux de σ_k dont le niveau est le niveau du backjumping : $\exists x \in \sigma_k$ tq. $c = \overleftarrow{cla}(x)$ et $l(x) = jump(\sigma_k)$;
2. $\forall y \in c$, soit $l(y) = m$ soit $l(y) < jump(\sigma_k)$.

Quand il existe un arc arrière joint c pour une preuve de résolution basé sur les conflits π , cela signifie qu'on peut l'utiliser pour étendre cette preuve : on obtient la clause $\sigma_{k+1} = \eta[x, \sigma_k, c]$, à partir de laquelle on commence une nouvelle preuve $\pi' = \langle \sigma_{k+1} \dots \sigma_l \rangle$ (Intuitivement c "joint" π et π'). Cet ensemble de séquences est appelé preuve de résolution jointe basée sur les conflits :

Définition 10 (preuve par résolution jointe) Une preuve par résolution jointe basée sur les conflits est une séquence de clauses $\langle \sigma_1 \dots \sigma_k, \sigma_{k+1}, \dots, \sigma_l \rangle$ telle que

- $\langle \sigma_1 \dots \sigma_k \rangle$ est une preuve de résolution basée sur les conflits ;
- $\sigma_{k+1} = \eta[x, \sigma_k, c]$, tel que c est un arc arrière joint ;
- Les clauses $\sigma_{k+2}, \dots, \sigma_l$ sont utilisées comme preuve de résolution basée sur les conflits i.e. elles respectent les conditions (2) et (3) de la définition 3.

Clairement, à partir de la définition 10, on peut dériver une nouvelle clause assertive. L'étape de jointure (appliquer la résolution entre σ_k et c) produit une nouvelle clause tel que le littéral qui a le plus haut niveau est éliminé. A partir de la nouvelle clause générée σ_{k+1} , on peut dériver par résolution (en utilisant les arcs du graphe d'implication classique) une autre clause conflit σ_l . Pour être sur que la nouvelle clause σ_l admette un niveau de backjump inférieur ou égal à $jump(c_k)$, une nouvelle condition sur l'arc arrière joint est introduite :

Définition 11 Soit $\pi = \langle \sigma_1, \sigma_2, \dots, \sigma_k \rangle$ une preuve par résolution basée sur les conflits. On définit $resLit(\pi)$ comme étant l'ensemble des littéraux du dernier niveau de décision m utilisés dans les résolutions de π

Propriété 5 Soit $\pi^e = \langle \sigma_1 \dots \sigma_k, \sigma_{k+1}, \dots, \sigma_l \rangle$ une preuve par résolution jointe basée sur les conflits tel que k est la position de l'arc arrière joint c . Si $\forall x \in c$ tq. $l(x) = m$, alors $x \in resLit(\pi_1) \cup \{\mathcal{A}(\sigma_k)\}$ alors $jump(\sigma_l) \leq jump(\sigma_k)$.

Preuve 4 Esquissons la preuve. Comme π_1 est une preuve par résolution basé sur les conflits, $\sigma_k = (\alpha \vee y \vee x)$ avec $\mathcal{A}(\sigma_k) = x$ et $l(\neg y) = jump(\sigma_k)$, est une clause assertive. Sans perte de généralité, supposons que $\forall z \in \alpha, l(\neg z) < l(\neg y)$. Comme π^e est une preuve par résolution basé sur les conflits, $\exists c = (\beta \vee \neg y)$ un arc inverse joint et $\sigma_{k+1} = \eta[y, \sigma_k, c] = (\alpha \vee \beta \vee x)$. Maintenant, à partir de σ_{k+1} , on applique la résolution uniquement sur l'ensemble des littéraux $\beta_1 = \beta \cap resLit(\pi_1)$. Comme ces littéraux sont précédemment utilisés dans la preuve par résolution π_1 , alors à partir de σ_{k+1} on peut dériver en suivant les mêmes arcs (par résolution) la clause assertive σ_l de la forme $(\alpha \vee \beta \setminus \beta_1 \vee x)$ ou $(\alpha \vee \beta \setminus \beta_1 \vee y \vee x)$. Dans le premier cas, $jump(\sigma_l) < jump(\sigma_k)$. par hypothèse et définition de c (voir condition 2), on déduit que les littéraux de α et $\beta \setminus \beta_1$ sont affectés à un niveau inférieur à $l(\neg y)$. Dans le deuxième cas, le littéral y qu'on veut éliminer de σ_k apparait dans σ_l . Par conséquent, les deux clauses assertives admettent le même backjumping level $l(y)$. Supposons que la clause assertive σ_k contient plusieurs littéraux avec le même niveau que y . Dans cec cas, en éliminant un seul littéral, on obtient le même niveau de backjumping.

A partir de la propriété 5, nous avons montré comment améliorer le niveau du backjumping en éliminant le littéral du niveau du backtrack. Naturellement, quand la clause assertive contient plusieurs littéraux du niveau du backtrack, on peut réitérer le processus sur la nouvelle clause assertive pour éliminer d'autres littéraux.

Plus intéressant encore, quand tous les littéraux du niveau le plus grand sont éliminés, on peut itérer le processus sur les littéraux du nouveau plus grand niveau et ainsi faire plusieurs backjumping successifs à partir d'un seul conflit.

Néanmoins, en pratique, un tel processus itératif peut nécessiter beaucoup de ressources.

Dans la section suivante, on introduit les choix pratiques et les restrictions adoptés lors de l'implémentation des arcs arrières dans les solveurs modernes.

6.1 Intégration pratique dans les solveurs modernes

Notre schéma d'apprentissage étendu peut être implémenté sur n'importe quel solveur de type CDCL. Rappelons les composantes essentielles constituant les solveurs SAT modernes : (1) l'analyse du conflit, (2) l'heuristique VISDS (prenant en compte l'activité des littéraux), (3) l'apprentissage des clauses assertives, (4) la politique de redémarrage. Les modifications suivantes sont effectuées :

1. *Analyse du conflit* : A chaque conflit, l'apprentissage classique est appliqué et une clause assertive $\sigma_1 = (\alpha \vee y^i \vee x^m)$, avec x le littéral assertif et $i = jump(\sigma_1)$, est générée grâce a une preuve par résolution basée sur les conflits π_1 (en utilisant le premier UIP). Avant d'effectuer le retour arrière au niveau i , on essaie d'utiliser la preuve de résolution jointe basée sur les conflits afin de générer une nouvelle clause assertive σ_2 . Dans notre implémentation, cette extension est seulement appliquée si σ_1 contient un seul littéral du niveau i . En général, on peut réitérer le processus pour tous les littéraux du niveau i . Une autre restriction est que l'arc inverse doit contenir des littéraux du niveau m ou de niveau j tel que $j < i$ i.e. $x \in resLit(\sigma_1) \cup \{\mathcal{A}(\sigma_1)\}$. Sans cette restriction, on peut générer d'autres clauses assertives qui peuvent être plus ou moins intéressantes en terme du niveau de backjumping.
2. *Activité des littéraux* : Les activités des littéraux introduits par l'arc arrière sont aussi pondérées.
3. *Apprentissage* Les deux clauses assertives sont ajoutées à la base des nogoods.

7 Experimentations

Les résultats expérimentaux reportés dans cette section sont obtenus sur un Xeon 3.2 GHz (2Go RAM) et ont été réalisés sur un large éventail d'instances (286) issus de la SATRACE 2006 et de la compétition SAT 2007 (uniquement les instances industrielles). Toutes les instances sont simplifiées à l'aide du pré-processeur SatElite [4]. Le temps CPU est limité à 1800 secondes et les résultats sont reportés en secondes. Nous avons implémenté l'approche proposée dans la section 6.1 aux solveurs Minisat [6] et Rsat [11] et proposons une comparaison entre les solveurs originaux et leur version étendue (appelés Minisat^E and Rsat^E).

Les deux graphiques (échelle logarithmique) de la figure 3 montre les comparaisons entre Minisat (figure du haut) et Rsat (figure du bas) par rapport à leur version étendue. Chaque point correspond à une instance. L'axe des abscisses (resp. des ordonnées) correspond au temps CPU t_x (resp. t_y) obtenu sur le solveur étendu (resp. original). Chaque point au dessus (resp. au dessous) de la diagonale indique que le solveur étendu est plus rapide (resp. plus lent) que la version originale sur l'instance donnée. La figure 3 montre clairement que l'utilisation des arcs arrières améliore les performances de Minisat. Cela semble beaucoup moins le cas pour Rsat, mais il est important de noter que Rsat^E est capable de résoudre 5 instances de plus que Rsat original (alors que Minisat^E résout une instance de moins que Minisat original).

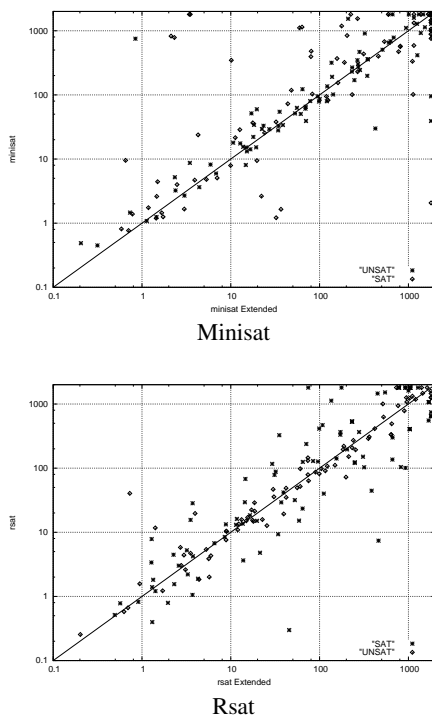


FIG. 3 – Résultats sur les instances de la compétition SAT'07 (industriel) et de la SATRACE'06

La figure 4 montre le temps T (figure du haut) et le temps cumulé CT (figure du bas) nécessaires pour résoudre un nombre donné d'instances ($\#instances$). T représente le nombre d'instances résolues en t secondes, CT représente la même chose mais en cumulant les temps de résolution. Rsat^E et Minisat^E permettent donc de résoudre plus d'instances et plus rapidement que les solveurs originaux.

Enfin, la table 1 exhibe les résultats obtenus sur un échantillon représentatif des instances. Pour chaque instance, on reporte le temps CPU nécessaire aux 4 différents solveurs.

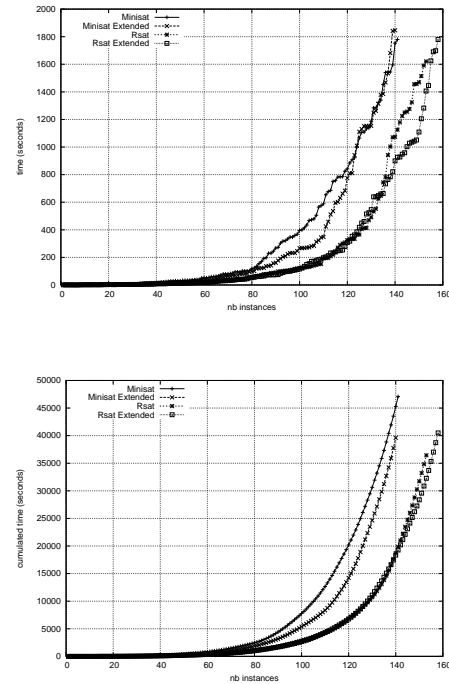


FIG. 4 – Temps et temps cumulé pour résoudre un nombre donné d'instances

Rappelons que nous appliquons notre méthode uniquement si la clause assertive contient un seul littéral du niveau de backjumping. C'est pour cela que l'utilisation des arcs arrières permet d'améliorer le saut arrière entre 1% et 10% des conflits. Malgré ce faible pourcentage, notre extension est clairement utile et montre le potentiel des arcs arrières. De plus, sur certaines classes d'instances le temps CPU est amélioré de plusieurs ordres de grandeur. Rsat^E est clairement le meilleur sur les familles *ibm*, *Velev*, *total* et *simon* alors que Minisat^E a de très bons résultats sur les familles *manol*, *vmvc*, *gold*, *APro*.

8 Conclusion

Dans ce papier, Nous avons proposé une généralisation du cadre de l'analyse du conflit. Cette généralisation est obtenue par une extension originale du graphe d'implication classique classiquement utilisé dans les solveurs CDCL. Cette extension, motivée par les résultats d'optimalité de la clause assertive (Premier UIP) en terme de niveau du backjump, est obtenue en considérant des clauses (appelées arcs inverses) qui proviennent de la partie satisfaite de la formule. Plusieurs schémas d'apprentissage peuvent être définis à partir de cette extension. Une première extension possible de l'apprentissage qui améliore les clauses

instance	SAT ?	Rsat ^{ES}	Rsat.	Minisat ^{ES}	Minisat
AProVE07-02	N	–	–	683.21	782.07
AProVE07-21	N	1406.02	–	511.14	505.41
AProVE07-22	N	232.68	208.17	136.19	316.98
clauses-6	Y	171.21	331.93	810.24	564.78
cube-9-h11-sat	Y	1282.38	–	774.78	472.25
dated-5-15	N	958.61	1074.07	–	1752.71
goldb-heqc-frg2mul	N	187.42	219.99	281.12	468.28
goldb-heqc-i8mul	N	1108.60	1324.17	941.759	1105.80
goldb-heqc-term1mul	N	–	1250.23	–	–
ibm-2002-19r-k100	Y	95.94	126.84	157.31	368.49
ibm-2002-21r-k95	Y	64.68	125.04	268.14	229.77
ibm-2002-26r-k45	N	3.96	19.65	0.84	751.33
IBM_04_30_dat.k8	Y	1042.88	–	1682.10	–
IBM_2004_30_SAT_dat.k100	Y	784.22	–	–	–
IBM_2004_30_SAT_dat.k55	Y	231.99	532.94	–	–
IBM_2004_30_SAT_dat.k60	Y	1698.74	1070.30	595.85	–
IBM_2004_30_dat.k80	Y	925.87	–	–	–
IBM_2004_30_dat.k85	Y	1042.88	–	1682.10	–
IBM_2004_30_dat.k90	Y	1051.16	–	–	–
manol-pipe-c7nidw	N	254.00	193.00	1469.20	–
manol-pipe-f7idw	N	1624.76	–	1010.01	–
manol-pipe-g10bidw	N	–	–	1313.43	–
manol-pipe-g10id	N	73.73	131.10	209.71	1539.87
mizh-md5-47-3	Y	107.98	469.35	1111.21	333.34
mizh-md5-47-4	Y	135.36	1125.72	267.50	1544.60
mizh-sha0-35-4	Y	278.60	365.84	1153.66	1323.37
partial-5-13-s	Y	546.21	1514.82	–	–
schup-12s-guid-1-k56	N	524.18	626.08	322.68	907.43
simon-s02b-dp11u10	N	515.00	1003.18	284.89	244.64
simon-s02b-r4b1k1.1	Y	1002.87	1620.79	458.83	400.30
total-10-17-s	Y	98.98	413.28	–	–
total-10-19-s	Y	74.46	–	–	–
total-5-17-s	Y	14.59	67.85	1384.98	–
velev-pipe-sat-1.1-b7	Y	70.85	238.16	224.04	–
velev-pipe-uns-1.0-9	N	764.74	941.76	–	–
velev-pipe-uns-1.1-7	N	733.14	–	–	–
vmpc_30	Y	176.49	–	–	–

TAB. 1 – Highlighted results

assertives classiques en terme de retour arrière intelligents montre le grand potentiel de ce nouveau cadre. Malgré les différentes restrictions, son intégration au sein des solveurs Minisat et Rsat montre des améliorations intéressantes sur les instances industrielles des deux dernières compétitions SAT'07 et SAT-race'06. Dans les travaux futurs, nous envisageons de définir d'autres schémas d'apprentissage, par exemple en considérant les arcs inverses comme une alternative à chaque noeud de l'arbre de recherche. En d'autres termes, le but serait d'améliorer les différentes résolvantes générées lors des "preuves par résolution basées sur les conflits" en termes de profondeur de backjumping et/ou de taille de clause. Enfin, nous envisageons d'exploiter les arcs arrière pour minimiser les clauses assertives.

Références

- [1] P. Beame, H. Kautz, and A. Sabharwal. Towards understanding and harnessing the potential of clause learning. *J. of Artificial Intelligence Research*, 22 :319–351, 2004.
- [2] M. Davis, G. Logemann, and D. W. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7) :394–397, 1962.
- [3] O. Dubois, P. André, Y. Boufkhad, and Y. Carlier. *Second DIMACS implementation challenge : cliques, coloring and satisfiability*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, chapter SAT vs. UNSAT, pages 415–436. American Mathematical Society, 1996.
- [4] N. Eén and A. Biere. Effective preprocessing in SAT through variable and clause elimination. In *Proceedings of the Eighth International Conference on Theory and Applications of Satisfiability Testing (SAT'05)*, pages 61–75, 2005.
- [5] N. Eén and N. Sörensson. An extensible SAT-solver. In *Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT'03)*, pages 502–518, 2003.
- [6] Niklas Eén and Niklas Sörensson. An extensible sat-solver. In *Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT'03)*, 2002.
- [7] E. Goldberg and Y. Novikov. BerkMin : A fast and robust SAT-solver. In *Proceedings of International Conference on Design, Automation, and Test in Europe (DATE '02)*, pages 142–149, 2002.
- [8] Chu Min Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *Procee-*

dings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97), pages 366–371, 1997.

- [9] J. Marques-Silva. Personal communication.
- [10] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff : Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (DAC'01)*, pages 530–535, 2001.
- [11] Knot Pipatsrisawat and Adnan Darwiche. Rsat 2.0 : Sat solver description. Technical Report D-153, Automated Reasoning Group, Computer Science Department, UCLA, 2007.