Réordonnancement dynamique basé sur l'apprentissage

Youssef Hamadi, Said Jabbour, Lakhdar Saïs

Réordonnancement dynamique basé sur l'apprentissage

Youssef Hamadi[1] **Said Jabbour**[2] **Lakhdar Sais**[2]

[1] Microsoft Research
7 J J Thomson Avenue
Cambridge, United Kingdom
youssefh@microsoft.com

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

Abstract

Cet article propose un nouveau schéma d'apprentissage. La particularité de cette approche réside dans sa capacité à opérer sans se référer aux conflits. Ceci contraste avec les approches d'apprentissage traditionnelles qui sont généralement basées sur l'analyse de conflits. Pour permettre un tel apprentissage, des clauses pertinentes issues de la partie satisfaites de la formule sont conjointement combinées au graphe d'implications classique afin de générer une meilleure raison d'implication pour un littéral donné. A partir de cette extension, un premier schéma d'apprentissage appelé réordonnancecement dynamique basé sur l'apprentissage (RDBA) est proposé. Il exploite les nouvelles raisons générées pour réordonner dynamiquement les affectations partielles. Des résultats expérimentaux montrent que l'intégration de RDBA sur des solveurs issus de l'état de l'art de SAT permet d'obtenir d'intéressantes améliorations, notamment sur les instances satisfiables.

1 Introduction

Le problème SAT, qui consiste à vérifier si un ensemble de clauses est satisfiable, est central dans plusieurs domaines de l'informatique. Il est important en intelligence artificielle, en satisfaction de contraintes (CSP), planification, raisonnement non monotone, vérification de circuits, etc. SAT est devenu un domaine attractif. Ceci s'explique par l'apparition d'une nouvelle génération de solveurs appelés solveurs SAT modernes [9, 4] particuliérement efficaces. Ces solveurs sont basés sur la propagation unitaire [2] combinée efficacement à des structures de données comportant : (i) des stratégies de redémarrage [6, 7], (ii) des heuristiques de choix de variables (comme VSIDS) [9], et (iii) l'apprentissage de clauses [8, 9]. Les solveurs SAT modernes sont spécialement efficaces sur les instances issues d'applications industrielles. Sur ces problèmes, Selman et al. [5] ont identifiés le phénomène de longue traîne (heavy tailed), i.e., diffèrents ordres de variables mènent souvent à des temps de résolution différents. Ceci explique entre autre l'introdution des stratégies de redémarrage dans les solveurs SAT modernes, dans le but de découvrir un bon ordre des variables. Par ailleurs, VSIDS et d'autres variantes d'heuristiques de choix de variables ont été introduites pour éviter le trashing et focaliser la recherche : lorsqu'il s'agit d'instances de grande taille ces heuristiques tentent de diriger la recherche vers la partie la plus contrainte de la formule. Les redémarrages et VSIDIS jouent un rôle complémentaire dès lors qu'ils implémentent respectivement les deux principes de diversification et d'intensification. L'apprentissage dirigé par les conflits (Conflict Driven Clause Learning (CDCL)) est le troisième composant, conduisant au retour-arrière non-chronologique. Dans CDCL une structure de données centrale est le *graph d'implications*, qui mémorise l'interprétation partielle en cours de construction ainsi que les implications faites. À chaque fois qu'un échec est rencontré, une clause conflit ou nogood est apprise grâce au parcours par le bas du graphe d'implications. Ce parcours est aussi utilisé pour mettre à jour les activités des variables concernées, permettant à l'heuristique VSIDS de sélectionner en priorité la variable la plus active lors de la prochaine décision. Dans ce papier nous traitons de l'apprentissage dans les solveurs SAT modernes. Notre but est de montrer que de façon similaire à la vie réelle, on peut apprendre non seulement à partir des échecs (ou conflits) mais aussi à partir des décisions menant à des succès. Comme l'apprentissage renvoi classiquement à l'analyse de conflits, le cadre que nous proposons contraste avec les approches classiques d'apprentissage proposées dans le cadre SAT.

Dans l'apprentissage classique, le but principale est de calculer la raison du conflit courant et de permettre d'effectuer un retour-arrière non chronologique. À l'opposé, notre approche a pour but principal de déduire de meilleures raisons pour l'implication d'un littéral donné permettant d'exploiter ces raisons pour réarranger (ou corriger) l'affectation partielle courante. Usuellement, à un nœud donné de l'arbre de recherche, lorsqu'un littéral de décision est affecté et après application de la propagation unitaire, la raison enregistrée pour un littéral impliqué x contient au moins un littéral du niveau courant i en plus du littéral x. Cette raison est codée dans le graphe d'implications. L'approche que nous proposons tend à découvrir de nouvelles raisons pour x, incluant seulement des littéraux de niveaux ($j < i$). Autrement dit, notre approche est capable de découvrir qu'un littéral propagé au niveau i aurait dû être propagé plus tôt, au niveau j du processus de recherche. En pratique, ces nouvelles raisons lorsqu'elles sont codées dans le graphe d'implication pourraient améliorer l'analyse de conflits elle-même. e.g., en améliorant les niveaux des retours-arrière. Contrairement à l'analyse de conflits classique où le littéral assertif est affecté au niveau du retour-arrière à sa valeur de vérité opposée (littéral refuté), dans l'approche que nous proposons, les nouvelles raisons générées sont utilisées pour réarranger l'interprétation courante en affectant quelques littéraux aux niveaux précédents en gardant les mêmes valeurs de vérité. En considérant le niveau d'implication de quelques littéraux, notre approche tend à repousser les mauvaises décisions en bas de l'arbre de recherche. Ce processus améliore la qualité de l'affectation partielle courante et favorise l'obtention des solutions. Le papier est organisé comme suit. Après quelques notations et définitions préliminaires, nous présentons le graphe d'implications et le schéma d'apprentissage classiques section 2.1 ensuite nous décrivons notre approche dans la section 3. Dans la section 4, l'intégration de RDBA au sein des solveurs SAT modernes est présentée. Finalement, avant la conclusion, des résultats expérimentaux quantifiant les performances de notre approche sont présentées.

2 Definitions

2.1 Definitions préliminaires 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 $\bot$, 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 $\rho \vDash \mathcal{F}$. Les notations suivantes seront largement utilisés par la suite :

- $\mathcal{R}[x, c_i, c_j]$ dénote la *résolvante* entre la clause c_i contenant le litéral x et la clause c_j contenant l'opposé de ce même litéral $\neg x$. Autrement dit $\mathcal{R}[x, c_i, c_j] = c_i \cup c_j \backslash \{x, \neg x\}$. Une résolvante est appelé *tautologique* si elle contient à la fois un litéral et son opposé.
- $\mathcal{F}|_x$ dénote la formule $\mathcal{F}$ simplifiée par l'affectation du litéral x à la valeur *vrai*. Formellement $\mathcal{F}|_x = \{c \mid c \in \mathcal{F}, \{x, \neg x\} \cap c = \emptyset\} \cup \{c \backslash \{\neg x\} \mid c \in \mathcal{F}, \neg x \in c\}$ (les clauses contenant x et qui sont donc satisfaites sont supprimées ; et celles contenant $\neg x$ sont simplifiées). Cette notation est étendue aux interprétations : soit $\rho = \{x_1, \ldots, x_n\}$ une interprétation, on définit $\mathcal{F}|_\rho = (\ldots((\mathcal{F}|_{x_1})|_{x_2})\ldots|_{x_n})$.
- $\mathcal{F}^*$ 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) $\mathcal{F}^* = \bot$ si $\mathcal{F}$ contient deux clauses unitaires $\{x\}$ et $\{\neg x\}$, (3) autrement, $\mathcal{F}^* = (\mathcal{F}|_x)^*$ tel que x est le litéral qui apparaît comme clause unitaire dans $\mathcal{F}$.
- $\models_*$ dénote la déduction logique par propagation unitaire : $\mathcal{F} \models_* x$ signifie que le litéral x est déduit par propagation à partir de $\mathcal{F}$, i.e. $x \in \mathcal{F}^*$. On écrit $\mathcal{F} \models_* \bot$ si la formule est inconsistance (insatisfaisable) par propagation. En particulier, si on dispose d'une clause c tel que $\mathcal{F} \wedge \overline{c} \models_* \bot$, alors c est une conséquence logique de $\mathcal{F}$. On dit alors que c est déduite *par réfutation*.

2.2 Recherche DPLL

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*" ; À chaque nœud les litéraux affectés (le litéral de décision et les litéraux 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 défaites et le niveau de décision courant est décrémenté en consèquence (égal au niveau du backtrack). Au niveau i, l'interprétation partielle courante ρ peut être représentée comme une séquence décision–propagations de la forme $\langle (x_k^i), x_{k_1}^i, x_{k_2}^i, \ldots, x_{k_{n_k}}^i \rangle$ 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 litéraux 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 litéraux affectés au niveau $\leq i$.

2.3 Analyse du Conflit et Graphe d'Implication

Les graphes d'implications 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. À chaque fois qu'un littéral y est propagé, on sauvegarde la clause à l'origine de cette propagation, clause que l'on note $\overrightarrow{imp}(y)$. La clause $\overrightarrow{imp}(y)$ est donc de la forme $(x_1 \vee \cdots \vee x_n \vee y)$ où chaque littéral x_i est faux sous l'interprétation courante ($\forall i \in 1 \ldots n \quad \rho(x_i) = \textit{faux}$) et $\rho(y) = \textit{vrai}$. Quand un littéral y n'est pas obtenu par propagation unitaire mais qu'il correspond à un point de choix, $\overrightarrow{imp}(y)$ est indéfini, que l'on note par convention $\overrightarrow{imp}(y) = \bot$.

Lorsque $\overrightarrow{imp}(y) \neq \bot$, on note par $exp(y)$ l'ensemble $\{\overline{x} \mid x \in \overrightarrow{imp}(y) \setminus \{y\}\}$, appelé l'ensemble des *explications* de y. Autrement dit, si $\overrightarrow{imp}(y) = (x_1 \vee \cdots \vee x_n \vee y)$ alors les explications sont les litéraux $\overline{x_i}$ qui constituent la condition sous laquelle $\overrightarrow{imp}(y)$ est devenue une clause unitaire $\{y\}$. Notons que pour tout i on a $l(\overline{x_i}) \leq l(y)$, i.e., toutes les explications de la déduction ont un niveau inférieur à celui de y.

Quand $\overrightarrow{imp}(y)$ est indéfini, $exp(y)$ est égal à l'ensemble vide. Les explications peuvent, alternativement, être vues comme un graphe d'implications dans lequel l'ensemble des prédécesseurs d'un nœud 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., there is exactly one node for every literal, decided or implied ;*
- *$\mathcal{E} = \{(x, y) \mid x \in \rho, y \in \rho, x \in \textit{exp}(y)\}$*

Par souci de simplicité, exp est éliminé dans le reste du papier, et le graphe d'implication est tout simplement noté $\mathcal{G}_{\mathcal{F}}^{\rho}$.

Exemple 1 *$\mathcal{G}_{\mathcal{F}}^{\rho}$, montré dans la Figure 1 est le graphe d'implications de la formule $\mathcal{F}$ et l'affectation partielle ρ donnée ci-dessous : $\mathcal{F} \supseteq \{c_1, \ldots, c_9\}$*

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

$\rho = \{\langle \ldots \neg x_6^1 \ldots \neg x_{17}^1 \rangle \langle (x_8^2) \ldots \neg x_{13}^2 \ldots \rangle \ldots \langle (x_{11}^5) \ldots \rangle\}$.
Le niveau de décision courant est 5 et $\rho(\mathcal{F}) = faux$.

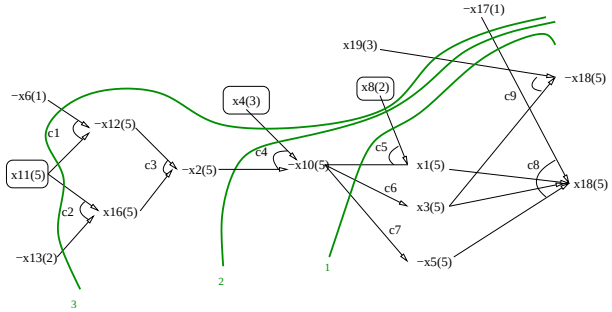

FIG. 1 – Implication Graph $\mathcal{G}_{\mathcal{F}}^{\rho} = (\mathcal{N}, \mathcal{E})$

L'analyse du conflit est le résultat de l'application de la résolution sur la clause devenue fausse (à partir des nœuds conflictuels du graphe d'implications) en utilisant diffèrentes clauses de la forme $(\overline{exp(y)} \vee y)$ implicitement codées à chaque nœud $y \in \mathcal{N}$. On appelle ce processus preuve par résolution basée sur les conflits. En utilisant l'exemple 1, on illuste à présent le schéma d'apprentissage basé sur les conflits (conflict driven clause learning) et ces concepts sous-jacents de clause assertive et unique point d'implication. Le parcours du graphe $\mathcal{G}_{\mathcal{F}}^{\rho}$ permet de générer les clauses suivantes :

- $r_1 = \mathcal{R}[x_{18}, c_8, c_9] = (x_{17}^1 \vee \neg x_1^5 \vee \neg x_3^5 \vee x_5^5 \vee \neg x_{19}^3)$
- $r_2 = \mathcal{R}[x_1, r_1, c_5] = (x_{17}^1 \vee \neg x_3^5 \vee x_5^5 \vee \neg x_{19}^3 \vee \neg x_8^2 \vee x_{10}^5)$
- $r_3 = \mathcal{R}[x_5, r_2, c_7] = (x_{17}^1 \vee \neg x_3^5 \vee \neg x_{19}^3 \vee \neg x_8^2 \vee x_{10}^5)$
- $r_4 = \Delta_1 = \mathcal{R}[x_3, r_3, c_6] = (x_{17}^1 \vee \neg x_{19}^3 \vee \neg x_8^2 \vee x_{10}^5)$

La clause Δ_1 est une *clause assertive* car tout ces littéraux sont affectées à $faux$ avant le niveau du conflit excepté un (x_{10}) qui est affecté au niveau courant 5. À partie de la première clause assertive, on déduit qu'on peut effectué un retour-arrière (backjumping) au niveau 3 (le niveau maximum des littéraux restant dans Δ_1) et affecter x_{10} (appelé littéral assertif) à $vrai$. Le nœud $\neg x_{10}$ dans le graphe (voir coupe 1 dans la Figure. 1) est appelé le premier *Unique Point d'Implication (UIP)*. Toutes les clauses intermidiares r_1, r_2 et r_3 contiennent plus qu'un seul littéral du niveau de décision actuel 5. Si nous continuons ce processus, nous obtenons d'autres clauses assertives $\Delta_2 = (x_{17}^1 \vee \neg x_8^2 \vee \neg x_{19}^3 \vee \neg x_4^3 \vee x_2^5)$ et $\Delta_3 = (x_{17}^1 \vee x_6^1 \vee \neg x_8^2 \vee x_{13}^2 \vee \neg x_4^3 \vee \neg x_{19}^3 \vee \neg x_{11}^5)$ correspon-

dant au second $\neg x_2^5$ (coupe 2 dans la Figure. 1) et au troisième UIP $\neg x_{11}^5$ (coupe 3 dans Figure. 1) respectivement. Le nœud $\neg x_{11}$ est le dernier UIP car il correspond au littéral de décision (voir coupes 2 et 3 dans la Fig. 1).

Notons que pour une clauses assertive $c = (\alpha \vee x)$ déduite par l'analyse de conflit au niveau m, on peut déduire par propagation unitaire x au niveau de décision $i < m$ où $i = l(\alpha)$ i.e., $(\mathcal{F}|_{\rho^i}) \wedge \overline{x} \models_* \bot$. Comme x est affecté à faux au niveau de décision courant m, l'analyse de conflit tend à *corriger* cette affectation en générant une meilleure explication pour le littéral x.

3 Génération de nouvelles explications

Dans cette section, nous montrons que de nouvelles explications pour les littéraux impliqués (littéraux propagés unitairement) peuvent être générées même si une affectation partiel ne mène pas forcèment à un conflit. Ces nouvelles capacités offertes par notre approche sont motivées dans l'exemple ci-dessous.

3.1 Example

Exemple 2 *Soit $\mathcal{F}'$ la nouvelle formule obtenue à partir de $\mathcal{F}$ (voir exemple 1) en substituant la clause c_9 par la nouvelle clause $c_{10} = (x_6 \vee \neg x_{10} \vee x_{18})$. Le graphe d'implications $\mathcal{G}^{\rho}_{\mathcal{F}'}$ est le sous-graphe de $\mathcal{G}^{\rho}_{\mathcal{F}}$ (voir Figure 1) induit par l'ensemble de nœuds $\mathcal{N}\backslash\{\neg x_{18}, x_{19}\}$. Clairement, l'affectation partielle ρ ne mène pas à un conflit au niveau de décision 5. Nous pouvons remarquer que la clause c_{10} n'est pas mémorisée dans le graphe d'implication $\mathcal{G}^{\rho}_{\mathcal{F}'}$ car elle contient deux littéraux $\neg x_{10}$ et x_{18} affectés à $vrai$. En plus, le littéral x_{18} est affecté par propagation unitaire au niveau 5 grâce à la clause c_8 et ρ. Illustrons à présent comment nous pouvons déduire une nouvelle raison (clause assertive) permettant de propager x_{18} à un niveau inférieur à 5. Premièrement, partant de la clause $\overrightarrow{imp}(x_{18})$, nous générons les résolvantes suivante :*

- $r'_1 = \mathcal{R}[x_1, c_8, c_5] = (x_{17}^1 \vee \neg x_8^2 \vee x_{10}^5 \vee \neg x_3^5 \vee x_5^5 \vee x_{18}^5)$
- $r'_2 = \mathcal{R}[x_3, r'_1, c_6] = (x_{17}^1 \vee \neg x_8^2 \vee x_{10}^5 \vee x_5^5 \vee x_{18}^5)$
- $r'_3 = \Delta'_1 = \mathcal{R}[x_5, r'_2, c_7] = (x_{17}^1 \vee \neg x_8^2 \vee x_{10}^5 \vee x_{18}^5)$

La clause Δ'_1 contient seulement deux littéraux x_{10} et x_{18} du niveau de décision actuel 5 et tous les littéraux sont affectés à $faux$ excepté x_{18} qui est affecté à $vrai$. Maintenant, si on applique une étape supplémentaire de résolution entre Δ'_1 et $c_{10} = (x_6 \vee \neg x_{10} \vee x_{18}) \in \mathcal{F}'$, on obtient une nouvelle clause unitaire et assertive $\Delta''_1 = (x_6^1 \vee x_{17}^1 \vee \neg x_8^2 \vee x_{18}^5)$. Cette dernière clause exprime le fait que le littéral x_{18} affecté au niveau 5 peut être propagé au niveau 2. En effet, comme $\mathcal{F}' \models \Delta''_1$ et les deux littéraux x_6 et x_{17} (resp. le littéral $\neg x_8$) sont affectés à $faux$ au niveau 1 (resp. niveau 2), on peut déduire que le littéral x_{18} affecté au niveau 5 peut être affecté par propagation unitaire au niveau 2 grâce à la nouvelle raison Δ''_1.

À partir de l'exemple précédent et contrairement au schéma d'apprentissage classique, on peut effectuer les premières observations suivantes :

- r'_3 n'est pas une clause conflit i.e., $\rho(r'_3) = vrai$
- le littéral x_{18} n'est pas réfuté i.e., la clause déduite Δ''_1 indique que x_{18} doit être affecté à la même valeur de vértié $vrai$ au niveau 2 au lieu du niveau 5

3.2 Présentation Formelle

Donnons à présent une présentation formelle du schéma d'apprentissage de nouvelles raisons. Dans toutes les définitions et propriétés suivantes, on considère $\mathcal{F}$ une formule CNF, ρ une interprétation partielle telle que $(\mathcal{F}|_\rho)^* \neq \bot$ et m le niveau de décision courant.

Définition 2 (clause bi-assertive) *Une clause $c = (\alpha \vee x \vee y)$ est dite clause bi-assertive ssi $\rho(\alpha) = faux$, $l(\alpha) < m$ et $l(x) = l(y) = m$.*

Une notion similaire de clause bi-assertive est définit par Knot Pipatsrisawat et Adnan Darwiche. Dans [11], une clause bi-assertive est définit comme une clause conflit avec deux littéraux affectés au niveau du conflit alors que dans la définition 2, nous ne faisons aucune hypothèse sur les valeurs de vérité des deux littéraux x et y. Suivant notre définition , la clause r'_3 (exemple 2) satisfaite par x_{18} est une clause bi-assertive. En résumé, et contrairement au travail présenté dans [11] notre définition n'est pas lié à la notion de clause-conflit. Elle peut être générée par preuve bi-assertive (Définition 3) *à chaque nœud* de l'arbre de recherche, même si le grahe d'implications n'est pas un graphe de conflit.

Définition 3 (résolution bi-assertive) *Soit x un littéral tel que $l(x) = m$ et $\overrightarrow{imp}(x)$ est définie. Une résolution bi-assertive $\pi_b(x)$ est une séquence de clauses $\langle r_1, r_2, \ldots r_k \rangle$ satisfaisant les conditions suivantes :*

1. $r_1 = \overrightarrow{imp}(x)$
2. *r_i, pour $i \in 2..k$, est construite en sélectionnant un littéral $y \in r_{i-1}$. La clause r_i est définie donc comme $\mathcal{R}[y, r_{i-1}, \overrightarrow{imp}(\bar{y})]$;*
3. *r_k est une clause bi-assertive*

Comme le littéral $x \in r_1$ ne peut pas être éliminé par résolution, on a $x \in r_k$. En effet, si cette élimination est possible, cela signifie que $\overrightarrow{imp}(x)$ et $\overrightarrow{imp}(\bar{x})$ sont définies. Ceci contredit l'hypothèse $(\mathcal{F}|_\rho)^* \neq \bot$. De plus, x est l'unique littéral de r_k affecté à $vrai$.

Dans [1], nous avons montré que le littéral assertif x généré en utilisant l'analyse de conflit peut être déduit par propagation unitaire au niveau i, où i est le niveau du saut arrière (ou backjumping) associé à la clause assertive $(\alpha \vee x)$ i.e, $\mathcal{F}|_{\rho^i} \models_* x$. De façon similaire, pour une clause

bi-assertive $(\alpha \vee y \vee x)$, la clause binaire $(y \vee x)$ (voir la propriété 1) peut être aussi déduite par propagation unitaire au niveau $i = l(\alpha)$.

Propriété 1 *Soit $\pi_b(x) = \langle r_1, r_2, \ldots, r_k \rangle$ une résolution bi-assertive telle que $r_k = (\alpha \vee y \vee x)$ où $i = l(\alpha)$. Nous avons $\mathcal{F}|_{\rho^i} \models^* (y \vee x)$*

Preuve 1 *Par définition de la résolution bi-assertive, $\pi_b(x)$ est dérivée en parcourant le graphe d'implications associé à $\mathcal{F}$ et ρ à partir du nœud x. Il est évident que l'affectation $\neg y$ au niveau i mène à la propagation du littéral x. En effet, toutes les clauses utilisées dans la résolution bi-assertive $\pi_b(x)$ contiennent seulement des littéraux de $\bar{\alpha}$ et des littéraux propagés au niveau de décision courant m. Par conséquent, on a $\mathcal{F}|_{\rho^i} \wedge \neg y \models^* x$. Donc $\mathcal{F}|_{\rho^i} \wedge \neg y \wedge \neg x \models^* \bot$.*

Illustrons la propriété 1 en utilisant l'exemple 2. À partir de la clause bi-assertive $r'_3 = (x^1_{17} \vee \neg x^2_8 \vee x^5_{10} \vee x^5_{18})$, il est facile de montrer que la clause $(x^5_{10} \vee x^5_{18})$ peut être déduite par propagation unitaire au niveau 2 i.e, $\mathcal{F}|_{\rho^2} \models^* (x_{10} \vee x_{18})$. A partir de la formule $\mathcal{F}$ et ρ, on peut voir que la formule $\mathcal{F}|_{\rho^2}$ contient les clauses suivantes $\{(x_{10} \vee x_1), (x_{10} \vee x_3), (x_{10} \vee \neg x_5), (\neg x_1 \vee \neg x_3 \vee x_5 \vee x_{18})\}$ (voir le graphe d'implications $\mathcal{G}^{\rho}_{\mathcal{F}'}$). Donc, on a $\mathcal{F}|_{\rho^2} \wedge \neg x_{10} \models^* x_{18}$. Par conséquent, $\mathcal{F}'|_{\rho^2} \wedge \neg x_{10} \wedge \neg x_{18} \models^* \bot$.

Pour dériver une nouvelle raison pour l'implication d'un littéral x donné, on procède en deux étapes. Dans la première, nous générons une résolution bi-assertive $\pi_b(x) = \langle r_1, r_2, \ldots, r_k = (\alpha \vee y \vee x) \rangle$, et dans la seconde, nous éliminons y de r_k par résolution. Ce processus de résolution est appelé résolution assertive, et est formellement défini ci-dessous :

Définition 4 (résolution assertive) *Soit $\pi_u(x)$ une séquence de clauses $\langle r_1, \ldots, r_{k-1}, r_k \rangle$. $\pi_u(x)$ est une résolution assertive si elle satisfait les deux conditions suivantes :*

1. *$\langle r_1, \ldots, r_{k-1} = (\alpha \vee y \vee x) \rangle$ est une résolution bi-assertive*
2. *$\exists c = (\beta \vee \neg y \vee x) \in \mathcal{F}$ une clause bi-assertive $r_k = \mathcal{R}[y, r_{k-1}, c] = (\gamma \vee x)$ où $\gamma = \alpha \cup \beta$.*

Il suit de la définition 2 que tous les littéraux de α et β sont affectés à $faux$ avant le niveau m. Par conséquent, la résolvante r_k est une clause assertive unitaire. Cela signifie que lorsque $\pi_u(x)$ existe, on déduit que le littéral x propagé au niveau m aurait dû être propagé à $vrai$ au niveau $k < m$ où $k = l(\gamma)$. La propriété suivante fait état de ce résultat.

Propriété 2 *Soit x un littéral tel que $l(x) = m$ et $\overrightarrow{imp}(x)$ existe. Si $\pi_u(x) = \langle r_1, \ldots, r_{k-1}, r_k = (\gamma \vee x) \rangle$ est une résolution assertive alors $\mathcal{F}|_{\rho^i} \models^* x$ où $i = l(\gamma)$.*

Algorithm 1: algorithme RDBA

```
   Données: A CNF formula F;
   Résultat: vrai if F is satisfiable ; faux otherwise
1  begin
2      currentLevel = 0, Δ = ∅;
3      while (vrau) do
4          if (propagate()==faux) then
5              if (currentLevel==0) then return faux;
6              c=(α ∨ ¬d)=analyze();
7              Δ = Δ ∪ c;
8              backJumpLevel = l(α);
9              bactrack(backJumpLevel+1);
10             (newJumpLevel, Δ′) =
               learnForNewReasons(backJumpLevel+1);
11             Δ = Δ ∪ Δ′;
12             bactrack(newJumpLevel);
13         else
14             currentLevel++;
15             if (decide()== faux) then
16                 return vrai
17 end
```

Preuve 2 *Premièrement, comme $\langle r_1, \ldots, r_{k-1} \rangle$ est une résolution bi-assertive, la résolvante r_{k-1} est de la forme $(\alpha \vee y \vee x)$ où $l(\alpha) \leq i$. En utilisant la propriété 1, on peut déduire que $\mathcal{F}|_{\rho^i} \models^* (y \vee x)$, alors $\mathcal{F}|_{\rho^i} \wedge \neg x \models^* y$. De plus, par définition de $\pi_u(x)$, la résolvante r_k est obtenue par résolution sur y entre r_{k-1} et $c \in \mathcal{F}$ de la forme $(\beta \vee \neg y \vee x)$ où $l(\beta) \leq i$. Comme $\forall z \in \beta\ \rho(z) = faux$, donc $(\neg y \vee x) \in \mathcal{F}|_{\rho^i}$ et $\mathcal{F}|_{\rho^i} \wedge \neg x \models^* \neg y$. Par conséquent, $\mathcal{F}|_{\rho^i} \wedge \neg x \models^* \bot$.*

Dans la propriété 2, nous avons montré que le littéral x peut être déduit par propagation unitaire au niveau i. Cependant, vérifier si chaque littéral non affecté peut être déduit par propagation unitaire à chaque nœud de l'arbre de recherche est coûteux. Dans la section suivante, on montre comment quelques unes de ces déductions peuvent être obtenues à la volée en utilisant notre approche d'apprentissage de nouvelles raisons.

4 Réordonnancement dynamique à base d'apprentissage

Nous présentons dans cette section comment le schéma d'apprentissage de nouvelles raisons peut être utilisé pour réordonner dynamiquement les affectations partielles d'un solveur SAT moderne. L'amélioration du solveur SAT est esquissée dans l'algorithme 1. Pour des raisons de simplicité, l'algorithme ne contient pas l'une des composante essentiel des solveurs SAT à savoir le redémarrage. Nous rappelons néanmoins les principales fonctions qui y sont incorporées :

- *propagate* : effectue la propagation unitaire et retourne $faux$ en cas de conflit et $vrai$ sinon.
- *analyze* : retourne la clause apprise c calculée en utilisant le schéma classique d'apprentissage. Le littéral

assertif est dénoté d.

- *learn* : ajoute une clause apprise c à la base des clauses apprises
- *learnFromSuccess* : applique notre schéma d'apprentissage de nouvelles raisons présenté dans l'algorithme 2
- *backtrack* : effectue un retour-arrière au niveau passé en paramètre.
- *decide* : sélectionne et affecte le littéral de décision suivant. La fonction retourne $vrai$, si la formule est satisfaite (tous les littéraux sont affectés), et $faux$ sinon.

Comme on l'a vu dans la section précédente, l'apprentissage de nouvelles raisons peut être appliqué à chaque nœud de l'arbre de recherche. En d'autres termes, pour chaque littéral unitaire y propagé à un certain niveau i, on peut appliquer le schéma d'apprentissage de nouvelles raisons pour déduire une implication à un niveau précédent i. Cependant, une application systématique peut dégrader les performances des solveurs. Dans l'algorithme 1, on applique l'apprentissage de nouvelles raisons (ligne 10) seulement lorsqu'un conflit est détecté i.e, si l'appel à *propagate()* retourne $faux$. Plus précisément, on utilise le composant classique d'analyse de conflit (ligne 6). La clause assertive c générée est ajoutée à la base des clauses apprises Δ (ligne 7). L'algorithme effectue donc un retour-arrière au niveau $backJumpLevel+1$ (ligne 9) et l'apprentissage de nouvelles raisons est seulement appliqué à ce niveau particulier. En restreignant ce schéma d'apprentissage aux littéraux y propagés au niveau $backJumpLevel + 1$, on garantit que chaque nouveau niveau d'implication i pour x est inférieur ou égal au niveau de retour-arrière courant. Un ensemble de nouvelles raisons Δ' et de nouveaux niveaux (appelé $newJumpLevel$) est retourné par la fonction $learnForNewReasons$ (ligne 10). le nouveau niveau de retour-arrière est calculé en choisissant le meilleur niveau (le plus petit) de ces nouvelles raisons Δ'. Avant d'effectuer un retour-arrière à ce nouveau niveau (ligne 12), l'ensemble de ces nouvelles raisons Δ' est ajouté à la base des clauses apprises (ligne 11). Si la procédure *learnForNewReasons* ne trouve pas de nouvelles implications, $newJumpLevel$ est égale à $backJumpLevel$, et l'algorithme suit le schéma classique d'apprentissage CDCL.

La fonction $learnFromNewReasons$ est détaillée dans l'algorithme 2. Premièrement, y (resp. U) désigne le littéral de décision (resp. l'ensemble des littéraux propagés à ce niveau) affecté au niveau $currentLevel$ (initialisé à $backJumpLevel + 1$) (ligne 2 et ligne 3). Pour chaque littéral $w \in U$, si on dispose d'une clause bi-assertive de la forme $c = (\beta \vee y \vee w) \in \mathcal{F}$ telle que $\rho(\beta) = faux$, $\rho(w) = \rho(y) = vrai$ est détecté, alors une résolution assertive $\pi_u(w) = \langle r_1, \ldots, r_k \rangle$ est opérée (voir ligne 7). La nouvelle implication r_k de w est ajoutée à la base des clauses apprises (ligne 8) et le $newJumpLevel$ est mis à jour (ligne 9). Tandis que les littéraux propagés à ce niveau $backJumpLevel + 1$ sont traités, le meilleur niveau d'implication ($newJumpLevel$) est retourné (ligne 10), et l'algorithme 1 peut effectuer un retour-arrière à ce nouveau niveau (ligne 12).

Algorithm 2: learnForNewReasons

Données: $currentLevel$: application level of learning for new reasons
Résultat: $newJumpLevel$: the highest implication level found; R : a set of new reasons;

```
1  begin
2      y = decisionLiteral(currentLevel);
3      U=UPLiterals(currentLevel);
4      newJumpLevel = currentLevel-1, R = ∅;
5      for (w ∈ U) do
6          if (∃c = (β ∨ y ∨ w) ∈ F s.t. ρ(β) = faux,
           ρ(w) = ρ(y) = vrai) then
7              Soit π_u(w) = ⟨r_1, ..., r_{k-1} = (α ∨ ¬y ∨ w), r_k⟩
           où r_k = (γ ∨ w) = R[y, r_{k-1}, c] tq. γ = α ∪ β;
8          R = R ∪ r_k;
9          newJumpLevel = min(l(γ),newJumpLevel);
10     return (newJumpLevel, R);
11 end
```

Pour réarranger l'affectation partielle, on utilise le système de sauvegarde similaire à celui proposé dans [10]. En effet, durant le retour-arrière, tous les littéraux de décisions entre $backJumpLevel$ (calculé à partir de l'apprentissage classique) et le $newJumpLevel$ (calculé par $learnForNewReasons$) sont enregistrés. La fonction $decide()$ les sélectionne en priorité dans le but de maintenir approximativement les mêmes décisions. Les littéraux unitaires avec les nouvelles implications sont affectés au bon niveau (le nouveau niveau calculé) grâce aux raisons enregistrées (voir ligne 12 dans l'algorithme 2). Ce processus définit notre approche de réordonnancement dynamique basé sur l'apprentissage (RDBA).

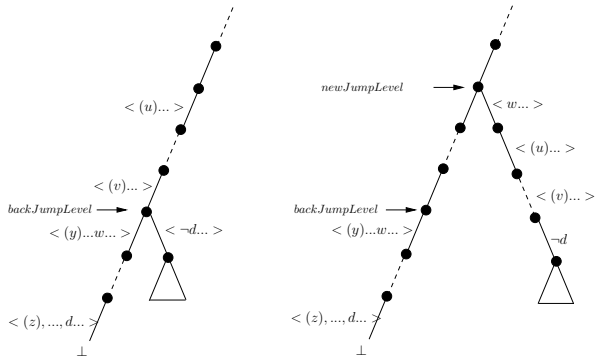

FIG. 2 – CDCL Vs RDBA

La figure 2 illustre la différence entre un branchement CDCL (à gauche) et RDBA (à droite). Soit $(\alpha \vee \neg d)$ la clause assertive obtenue par l'analyse de conflit classique. Dans l'approche CDCL, l'algorithme effectue un retour-

arrière au niveau $backJumpLevel = l(\alpha)$ et affecte le littéral assertif $\neg d$. Dans l'approche RDBA, l'algorithme effectue un retour-arrière au niveau $backJumpLevel + 1$ et applique le processus d'apprentissage de nouvelles raisons sur tous les littéraux propagés à ce niveau (littéral w dans la figure). Toutes les nouvelles implications pour ces littéraux sont enregistrées dans la base des clauses apprises. Supposons que le meilleur niveau de retour-arrière ($newJumpLevel$) correspond à la nouvelle implication pour w. L'algorithme effectue un retour-arrière au niveau $newJumpLevel$ et affecte le littéral w et les autres littéraux grâce à la mémorisation de l'implication. Finalement, la recherche continue en affectant en priorité les littéraux de décision entre $backJumpLevel$ et $newJumpLevel$ seulement et dans l'ordre tant que c'est possible.

5 Expérimentations

Nous avons effectués des expérimentations sur un large panel de problèmes industriels issus des deux dernières compétitions SAT-Race'06 et SAT'2007. Avant les tests de résolution toutes les instances sont pré-traiter par *SatELite* [3]. Le temps limite de calcul est fixé à 1800 secondes et les résultats sont indiqués en secondes. Nous avons intégré l'extension RDBA (section 4) dans Minisat et nous avons effectué une comparaison entre Minisat classique et Minisat incluant cette approche. Ces tests ont été menés sur un cluster Xeon 3.2GHz (2 GB RAM).

Les représentations en echelle logarithmique dans les figures 3 et 4 illustrent une comparaison des résultats de Minisat et Minisat+RDBA sur des instances satisfiables et insatisfiables respectivement. L'axe des x (resp. y) correspond à Minisat+RDBA (resp. Minisat). Chaque instance est associée à un point de coordonnées (tx, ty) qui représente les temps de calcul obtenus par les deux approches sur l'instance donnée.

La figure 3 montre que sur les instances satisfiables, l'apprentissage de nouvelles raisons permet d'accélérer le processus de résolution par rapport à Minisat. La majorité des points de la figure sont situés au-dessus de la diagonale i.e., Minisat+RDBA est meilleur. Ces résultats ne sont pas surprenants car notre approche vise à corriger le niveau et non pas la valeur de vérité des littéraux impliqués. Ce qui est diffèrent de l'analyse de conflit classique, où on modifie à la fois le niveau d'affectation (niveau assertif) et la valeur de vérité du littéral assertif. D'un autre côté, pour les instances insatisfiables (voir figure 4), il n'y a pas de séparation claire entre les performances des deux approches.

Les figures 3 et 4 (figures du bas) montrent le $temps$ mis par chaque solveur pour résoudre un certain nombre d'instances. Cette vision globale confirme que notre approche est meilleure sur les instances satisfiables en terme de temps de calcul. Au contraire sur les instances insatisfiables, la différence entre les performances est moins

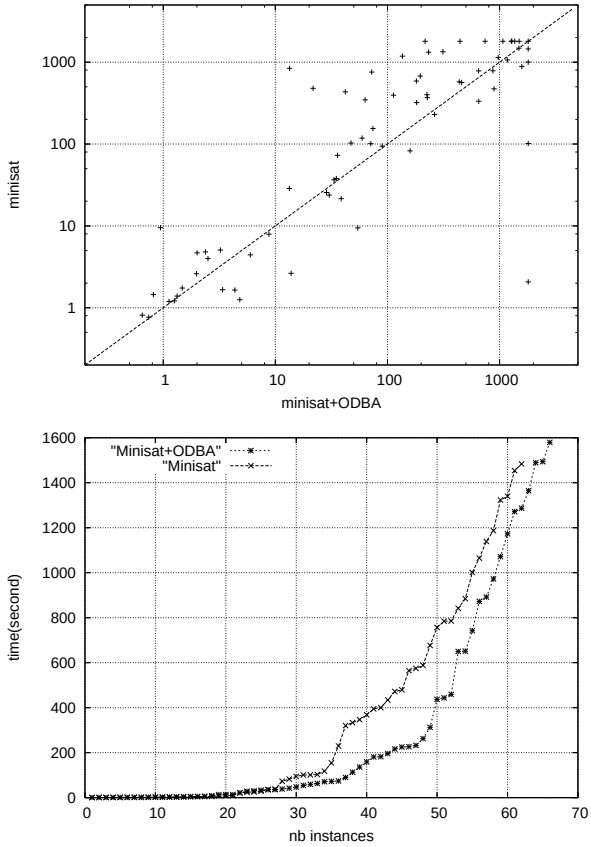

FIG. 3 – Satisfiables instances

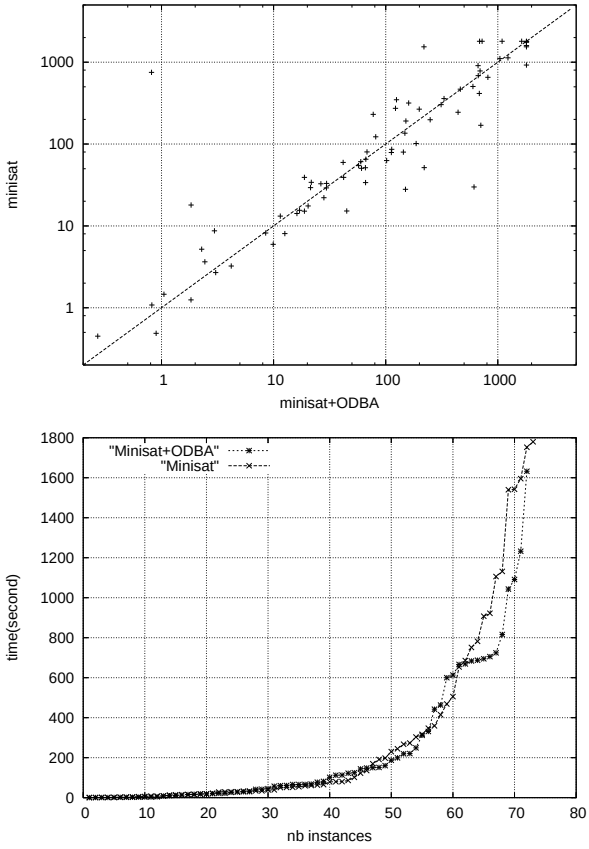

FIG. 4 – Unsatisfiables instances

flagrante mais en moyenne Minisat+RDBA est légèrement meilleur.

Finalement, la table 1 résume quelques résultats obtenus par Minisat+RDBA et Minisat. Pour chaque famille (première colonne), on indique le nombre d'instances (deuxième colonne), la moyenne du temps de calcul sur les instances résolues et le nombre d'instances résolues (entre parenthèses). Les résultats sont donnés sur des classes (SAT) et (UNSAT) et sur toutes les instances (SAT-UNSAT). Pour chaque famille, les meilleurs résultats en terme de temps moyen ou de nombre d'instances résolues sont mises en gras pour plus de lisibilité. Comme on peut le remarquer sur ces résultats Minisat+RDBA obtient de meilleurs résultats en général sur des familles satisfiables. Sur la famille $safe-*$ par exemple, RDBA permet un gain de près de deux ordres de grandeur.

Pour résumer, l'intégration de RDBA sur un solveur moderne est assez prometteuse car elle a mis en lumière une question principale sur l'intérêt de l'ordre d'affectation dans les solveurs modernes. Cette approche est complémentaire au schéma classique d'apprentissage dans la mesure où elle offre une alternative, qui, combinée à CDCL, permet comme cela a été dit de corriger l'interprétation courante.

6 Conclusion

Dans cet article, un nouveau cadre d'apprentissage de nouvelles raisons d'implication est proposé. Ce nouveau cadre permet de dériver de meilleures implications pour la propagation unitaire des littéraux. Ceci peut être vu comme une méthode de réarrangement de l'interprétation courante. Alors que dans les solveurs SAT la partie satisfaite de la formule est ignorée, notre approche suggère que ces clauses connectées au reste de la formule peuvent être exploitées avantageusement durant la recherche. Notre première intégration de RDBA dans un solveur moderne (Minisat) montre une nette amélioration du temps de calcul sur certaines classes de problèmes industriels. Plus intéressant encore, nous avons constaté une relation entre la résolution assertive et la déduction basée sur la propagation unitaire. Comme notre approche essaye de réarranger l'interprétation courante, nous envisageons d'approfondir la question du rôle joué par les redémarrages dans ce contexte. Finalement, étendre ce schéma pour récupérer d'autres formes de consistance plus fortes que la propagation unitaire nous semble être une piste de recherche pouvant permettre la résolution de problèmes difficiles.

Références

[1] G. Audemard, L. Bordeaux, Y. Hamadi, S. Jabbour, and L. Sais. Generalized framework for conflict analysis. In *In proceedings of eleventh International*

		SAT		UNSAT		SAT-UNSAT	
familles	# inst.	Minisat+RDBA	Minisat	Minisat+RDBA	Minisat	Minisat+RDBA	Minisat
mizh_*	10	**501**(10)	720(10)	–	–	**501**(10)	720(10)
manol_*	20	–	–	**336(17)**	344(14)	**336(17)**	344(14)
partial_*	20	**1272(1)**	–	–	–	**1271(1)**	–
total_*	20	262**(7)**	94(4)	66(3)	66(3)	203**(10)**	82(7)
vmp_grieu_*	12	**462**(4)	801(4)	–	–	**462(4)**	801(4)
APro_*	16	13(1)	**2**(1)	323(7)	557**(9)**	284(8)	502**(10)**
dated_*	20	160(9)	**151**(9)	135(2)	661(3)	156(11)	279**(12)**
clause_*	4	**233**(4)	303(4)	–	–	**233**(4)	303(4)
cube_*	4	891(1)	**472**(1)	**59**(1)	60(1)	475(2)	**266**(2)
gold_*	7	–	–	597(4)	**577**(4)	597(4)	**577**(4)
safe_*	4	**13**(1)	841(1)	–	–	**13(1)**	841(1)
ibm_*	20	**98**(10)	121(10)	**36**(9)	102(9)	**69**(19)	112(19)
IBM_*	35	1295**(4)**	1113(3)	**331**(1)	359(1)	1102**(5)**	924(4)
simon_*	6	**149**(2)	277(2)	204(3)	**128**(3)	**182**(5)	188(5)
block_*	2	–	–	27(2)	27(2)	27(2)	27(2)
dspam_*	2	–	–	315(2)	**34**(2)	315(2)	**34**(2)
schup_*	2	**71**(1)	100(1)	**666**(1)	907(1)	**368**(2)	504(2)
sort_*	5	**312**(1)	1339(1)	1232(1)	**1131**(1)	**772**(2)	1235(2)
velev_*	12	–	**2(1)**	25(3)	**19**(3)	25(3)	**15(4)**

TAB. 1 – Highlighted results

Conference on Theory and Applications of Satisfiability Testing (SAT'2008), 2008.

[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] 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.

[4] 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)*, pages 502–518, 2002.

[5] C. P. Gomes, B. Selman, N. Crato, and H. Kautz. Heavy-tail phenomena in satisfiability and constraint satisfaction. *Journal of Automated Reasoning*, pages 67 – 100, 2000.

[6] Carla P. Gomes, Bart Selman, and Henry Kautz. Boosting combinatorial search through randomization. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'97)*, pages 431–437, Madison, Wisconsin, 1998.

[7] H. Kautz, E. Horvitz, Y. Ruan, C. Gomes, and B. Selman. Dynamic restart policies. In *AAAI'02*, pages 674–682, 2002.

[8] Joao P. Marques-Silva and Karem A. Sakallah. GRASP - A New Search Algorithm for Satisfiability. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 220–227, November 1996.

[9] 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.

[10] Knot Pipatsrisawat and Adnan Darwiche. A lightweight component caching scheme for satisfiability solvers. In *Proceedings of 10th International Conference on Theory and Applications of Satisfiability Testing(SAT)*, pages 294–299, 2007.

[11] Knot Pipatsrisawat and Adnan Darwiche. A new clause learning scheme for efficient unsatisfiability proofs. In *AAAI'2008*, pages 1481–1484, 2008.