Pourquoi les solveurs SAT modernes se piquent-ils contre des cactus ?

Gilles Audemard, Mouny Samy Modeliar, Laurent Simon

Pourquoi les solveurs SAT modernes se piquent-ils contre des cactus ?*

Gilles Audemard[1] **Mouny Samy Modeliar**[1] **Laurent Simon**[2]

[1]Univ. Lille-Nord de France
CRIL/CNRS UMR8188
Lens, F-62307 CRIL-CNRS,

{audemard,modeliar}@cril.fr

[3] Univ. Paris-Sud
LRI/CNRS UMR 8623
INRIA Saclay
Orsay, F-91405

simon@lri.fr

Abstract

Malgré les progrès impressionnants observés ces dernières années autour de la résolution pratique d'instances SAT industrielles, les solveurs CDCL (*Conflict Driven, Clause Learning*) ont toujours un comportement global qui se doit d'être relativisé. Celui-ci est en effet souvent caractérisé par les figures *cactus*, qui suggèrent l'existence d'un seuil d'incompétence atteint par tout solveur après un certain temps de recherche. Les chances de résoudre une instance une fois passée cette limite semble décroître de manière exponentielle.

Dans cet article, nous rapportons une étude expérimentale intensive sur un solveur CDCL canonique, en nous focalisant sur un panel d'instances industrielles. Nous étudions particulièrement la décroissance du nombre de niveaux de décisions durant la recherche. Cela nous permet de proposer une hypothèse quant à la puissance du mécanisme d'apprentissage, à savoir, la production de clauses particulières, appelées "*clauses glues*". Nous montrons, sur de nombreuses instances, que de moins en moins de clauses glues sont produites, celle ci suivant une loi de type $log(x)^B$, ce qui peut être une bonne explication de l'inefficacité des solveurs passé un certain temps.

1 Introduction

La plupart des articles qui traitent du problème SAT commencent traditionnellement par rappeler les progrès extraordinaires mesurés ces dernières années, notamment sur les problèmes industriels. Les résultats récents sont en effet impressionnants, à tel point qu'aujourd'hui, la résolution de nombreux problèmes industriels passe par un codage SAT à la place des solveurs ad hoc [13] du domaine. Mais le tableau global n'est pas si positif. Si l'on revêt quelques instants le costume de l'avocat du diable, on peut souligner que, dans cette quête de la performance, seuls quelques progrès ont été fait sur la compréhension réelle des mécanismes de ces solveurs. Certes, nous savons quels sont les ingrédients nécessaires pour obtenir de bonnes performances, mais leurs rôles respectifs ne sont pas encore parfaitement connus. Si l'on suit ce raisonnement, on peut aussi faire remarquer que ces solveurs modernes, descendants de ZCHAFF [10], utilisent des mécanismes d'apprentissages, comme le premier UIP (*Unique Implication Point* [16]), déjà introduits 8 ans avant ZCHAFF [14]. Derrière les progrès certes impressionnants, se cache souvent une simple réécriture de ZCHAFF, accompagnée d'astuces plus ou moins importantes (écriture compacte des clauses, redémarrages rapides, sauvegarde de la phase, littéraux bloqués...).

Les structures de données paresseuses introduites dans ZCHAFF ont provoqué un bouleversement du monde SAT qui dépasse de loin leurs motivations initiales. En effet, ces structures de données, initialement introduites pour empêcher les défauts de cache [17] ont interdit toute information de type *lookahead* (comme le simple comptage d'occurrences des littéraux). Comme ces structures de données sont incontournables pour résoudre les problèmes industriels qui comptent un grand nombre de variables et de clauses, la plupart des solveurs n'ont pas eu d'autres choix que de suivre cette voie, devenant tous de type *lookback*. Dès lors, les variables (pour les choix heuristiques) et les clauses (pour connaître celles qu'il faut supprimer parmi les clauses apprises) sont estimées uniquement sur leur ac-

*supporté par le projet ANR UNLOC

tivité passée. Les solveurs sont donc maintenant conçus pour atteindre le plus rapidement possible les conflits afin d'apprendre de nouveaux nogoods et de mettre à jour les valeurs heuristiques connexes. L'un des effets de bord de ce schéma d'algorithme est que les solveurs sont devenus de plus en plus imprévisibles. Les *anciens* solveurs, basés sur les techniques de type *lookahead* (voir [9] par exemple), étaient beaucoup plus faciles à comprendre. En effet, la recherche était guidée, par exemple, pour réduire le plus possible la taille de l'arbre de recherche. Les techniques de type *lookback*, comme par exemple l'heuristique VSIDS (Variable State Independent Decaying Sum [10]) couplée à des redémarrages rapides [6], sont beaucoup plus difficiles à appréhender. En conclusion, nous savons donc implanter des solveurs très efficaces, qui plus est en moins de 1000 lignes de code, mais nous ne savons pas toujours quels sont les mécanismes qui sont vraiment importants et pourquoi ils le sont. De nombreuses questions restent donc en suspens. Que sont de bonnes clauses apprises ? pourquoi les redémarrages à la luby sont-ils si efficaces ? Est ce que certaines clauses apprises ne servent qu'à mettre à jour les heuristiques et simuler le retour arrière ?

Dans ce contexte, il nous semble de plus en plus important de construire une réelle science expérimentale autour des algorithmes [5] afin de comprendre le comportement des solveurs de type *lookback*. C'est l'idée principale que cet article tente de suivre. Nous proposons une étude expérimentale d'un solveur CDCL canonique (MINISAT [3]) afin d'essayer de comprendre ces forces et faiblesses. Notre travail, est un prolongement de remarques déjà faites dans [1], où nous montrions la relation entre la décroissance des niveaux de décisions et la performance des solveurs.

Notre hypothèse est que les solveurs CDCL réduisent, tout au long de la recherche, le nombre de décisions à faire avant d'atteindre un conflit. De plus, sur de nombreuses instances, cette décroissance des niveaux de décision semble suivre une loi exponentielle (par rapport au nombre de conflits déjà rencontrés), rendant de plus en plus difficile l'obtention du dernier conflit. Nous faisons une autre hypothèse, à savoir que cette loi exponentielle est guidée par l'incapacité des solveurs à garder un taux élevé de production de certaines clauses tout au long de la recherche. Nous montrons cela sur des variantes de MINISAT, suggérant que ces limites sont dues à l'architecture des solveurs CDCL. Néanmoins, il est important de noter que cet article risque de soulever plus de questions que de réponses. Il est en effet utopique de chercher des lois de régression qui puissent s'appliquer sur des problèmes très différents, certains de nos résultats pourront donc être vu comme négatifs. Malgré tout, nous pensons que ce travail est nécessaire pour aider la communauté à mieux comprendre comment les solveurs se comportent.

Le reste de l'article est organisé comme suit. La section 2 présente quelques notations et rappelle les résultats obtenus dans un travail annexe [1]. La troisième section étudie les différentes lois de régression utilisées. Avant de conclure, la section 4 étudie l'évolution des performances des solveurs sur des problèmes difficiles. Les performances sont mesurées par la capacité à produire des clauses que nous pensons importantes.

Algorithm 1: Solveur de type CDCL

```
   Input: Σ une formule CNF
   Output: SAT ou UNSAT
1  I = ∅ ;                    /* interprétation */
2  nd = 0 ;                   /* niveau de décision */
3  x_c = 0 ;                  /* numéro du conflit */
4  while (true) do
5      c = propagationUnitaire(Σ,I);
6      if (c !=null) then
7          x_c = x_c + 1;
8          γ =AnalyseConflit(Σ,I,c);
9          bl = CalculeRetourArriere(γ,I);
10         if (bl < 0) then return UNSAT;
11         Σ = Σ ∪ {γ};
12         if (redémarrage()) then bl = 0;
13         RetourArriere(Σ,I,bl);
14         nd = bl;
15     else
16         if (toutes les variables sont affectées) then
17             return SAT;
18         ℓ = choixLiteralDecision(Σ);
19         nd = nd + 1;
20         I = I ∪ {ℓ};
21
22 end
```

2 Travaux préliminaires et antérieurs

Soit $V = \{x_1, \ldots, x_n\}$ un ensemble de *variables booléennes*, un *littéral* ℓ_i est une variable x_i ou sa *négation* $\neg x_i$. Une *clause* est une disjonction de littéraux $c_i = l_1 \vee l_2 ... \vee l_{n_i}$. Une clause *unitaire* ne contient qu'un seul littéral. Le problème SAT est un problème de décision défini comme suit : étant donné une formule propositionelle Σ sous forme normale conjonctive (CNF, une conjonction de clauses), existe-t-il une affectation des variables V de Σ tel que Σ soit satisfaite, c.a.d., tel que toutes les clauses de Σ soient satisfaites ?

2.1 Solveurs CDCL

L'algorithme 1 montre l'organisation d'un solveur de type CDCL (nous ne rentrerons pas dans le détail de chaque fonction de cet algorithme). Une branche typique d'un tel solveur est une séquence de décisions suivies de propaga-

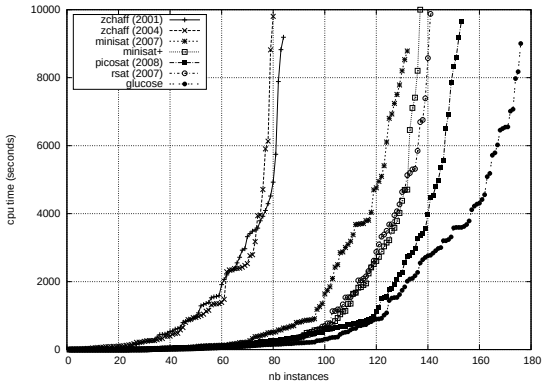

FIG. 1 – Figure cactus sur les 234 instances industrielles de la compétition SAT'07. MINISAT+ est MINISAT avec les redémmarages à la luby, la sauvegarde de phase et les litéraux bloqués. GLUCOSE est notre solveur [1].

Series	#Benchs	% Decr.	$m_{x_{jp}}$	r
een	8	62%	1.1×10^3	0.31
goldb	11	100%	1.4×10^6	0.22
grieu	7	71%	1.3×10^6	0.48
hoons	5	100%	7.2×10^4	0.30
ibm-2002	7	71%	4.6×10^4	0.21
ibm-2004	13	92%	1.9×10^5	0.24
manol-pipe	55	91%	1.9×10^5	0.40
miz	13	0%	–	0.37
schup	5	80%	4.8×10^5	0.38
simon	10	90%	1.1×10^6	0.34
vange	3	66%	4.0×10^5	0.06
velev	54	92%	1.5×10^5	0.25
all	199	83%	3.2×10^5	0.31

TAB. 1 – Décroissance des niveau de décision et justification à postériori

tions, répétée jusqu'à ce qu'un conflit survienne. Chaque littéral de décision (lignes 18–20) est affecté à un niveau de décision (nd), les littéraux qui se déduisent (par propagation unitaire) de cette décision ont le même niveau. Si tous les littéraux sont affectés, alors I est un modèle de Σ (lignes 16-17). À chaque fois qu'un conflit est atteint par propagation unitaire (c est alors la clause falsifiée, lignes 5-6), un nogood γ est calculé (ligne 8) en utilisant une méthode donnée, le plus souvent le premier UIP (Unique Implication Point) [16] et un niveau de backjump bl est calculé. À ce moment, on peut avoir prouvé l'inconsistance de la formule. Si ce n'est pas le cas, on fait un saut arrière et le nouveau niveau de décision devient égal au niveau de backjump (lignes 13-14). Enfin, de temps à autres, les solveurs CDCL forcent des redémarrages (diverses stratégies sont possibles, voir [6] par exemple) et dans ce cas, on remonte tout en haut de l'arbre de recherche (ligne 12).

2.2 Des figures Cactus ?

Les *figures cactus*, comme par exemple la figure 1, sont souvent utilisées comme un outil résumant les performances d'un ensemble de solveurs sur un ensemble de problèmes. Une telle figure montre clairement que tous les solveurs SAT atteignent un seuil d'incompétence I après un certain temps (il ne semble pas illogique de relier ce seuil d'incompétence au principe de PETER [11]). Avec les progrès récents réalisés, ce point d'incompétence est régulièrement poussé vers la droite, mais il existe toujours. Intuitivement, l'existence de ce point semble signifier que, si un solveur n'a pas résolu une instance donnée en I secondes, alors il y a peu de chances qu'il la résolve en $2 \times I$ secondes. La possibilité qu'il la résolve semble ainsi décroître exponentiellement avec le temps additionnel donné, une fois ce seuil d'incompétence franchi.

Dans [4], les auteurs suggèrent que ceci peut être relié aux phénomènes de type longue traîne (*heavy tailed*) observés dans les arbres de recherche. Les auteurs montrent qu'ils peuvent être dûs à de mauvais choix de variables en haut de l'arbre (Cette idée est également reliée à celle des "*backdoors*" d'une formule [15]). Néanmoins, les solveurs modernes utilisent des redémarrages extrêmement rapides (on notera que l'on ne devraient peut-être plus les appeler redémarrage, mais réorganisation des valeurs de littéraux [2]). Si les phénomènes de longues traînes sont effectivement dus à de mauvais choix en haut de l'arbre, ces solveurs devraient donc bien en être préservés.

D'après nos connaissances, il n'y a donc pas de justification théorique satisfaisantes des figures cactus. Nous pensons que, pour améliorer les solveurs CDCL et proposer de nouvelles stratégies, nous devons d'abord comprendre les raisons de leurs inefficacité sur certaines instances. Nous espérons que cet article sera un premier pas dans ce sens.

2.3 Une première observation : les niveaux de décision décroissent

Nous rappelons ici les premières observations faîtes dans [1]. Pour des raisons de simplicité, nous rappelons ici la même table (table 1) légèrement modifiée. Ces observations ont été faites sur de nombreuses instances issues des dernières compétitions SAT. Voici comment nous l'avons obtenu : nous lançons le solveur MINISAT [3] sur ces instances. Chaque fois qu'un conflit numéroté x_c est atteint (ligne 7 de l'algorithme 1), on enregistre le niveau de décision nd où il est survenu. A la fin de la recherche, nous calculons les caractéristiques (a et b) de la loi de régression linéaire $y = a \times x + b$ sur l'ensemble des points (x_c,nd). Dans ce travail préliminaire, nous ne nous sommes intéressés qu'au comportement général de cette droite, et particu-

lièrement à sa décroissance. Nous n'avons pas regardé si cette régression était la meilleure possible, ce qui est l'un des objets de cet article. Si cette droite décroît alors $a < 0$. Nous pouvons dans ce cas "*prédire*" quand le solveur résout l'instance (qu'elle soit satisfaisable ou pas). C'est le moment où la droite de régression intersecte l'axe des x (le conflit est trouvé au niveau de décision 0 de l'arbre de recherche). Nous appelons ce point, le point de "*justification à posteriori*" (connaître tous les points est en effet nécessaire pour calculer la droite de régression et donc la justification). Les coordonnées de ce point sont ($x_{jp} = -b/a$,0). Cette valeur donne une bonne intuition de la décroissance des niveaux de décision durant la recherche, une décroissance rapide donnera des justifications relativement petites.

La table 1 montre que sur une large majorité des instances les niveaux de décision décroissent bel et bien. "#Benchs" donne le nombre de benchmarks dans la série, "#Decr." donne le pourcentage de benchmarks qui voient une décroissance des niveaux de décision. La colonne 4 donne la médiane $m_{x_{jp}}$ des justifications calculées lorsque cela est possible (quand il y a effectivement décroissance). On peut noter que, dans la plupart des cas (167 sur 199), les droites de régression décroîssent. De plus, dans très peu de cas, la valeur $m_{x_{jp}}$ est importante.

La dernière colonne donne l'indice de Bravais-Person r. Cet indice vaut entre 0 et 1. Il montre l'adéquation entre les points (x_c, nd) et la droite de régression. Une valeur au-dessus de 0.71 montre une adéquation pas trop mauvaise, une valeur au dessus de 0.86 une bonne. Une conclusion s'impose : les régressions linéaires calculées sont très mauvaises.

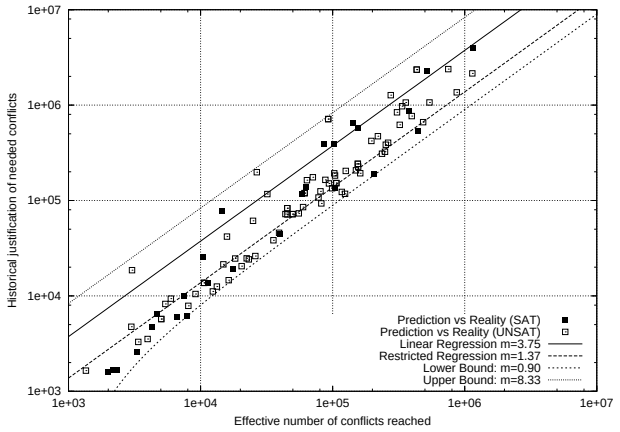

FIG. 2 – Relation entre justification à posteriori et nombre de conflits nécessaires à MINISAT pour résoudre l'instance.

Néanmoins, le fait d'avoir de mauvaises valeurs de r ne veut pas dire que la justification à posteriori est mauvaise. En effet, que pourrait-on déduire si les justifications sont de bonnes estimations du nombre réel de conflits nécessaires pour prouver l'inconsistance (ou plus incroyable encore, une solution) ? Cela voudrait peut-être dire que la décroissance des niveaux est une caractéristique importante et une des puissances des solveurs CDCL : Ils forcent les niveaux de décision à régulièrement décroître le long de la recherche.

La figure 2 (également issue de [1]) est construite comme suit : Chaque point (x, y) correspond à une instance. x correspond au nombre de conflits effectifs nécessaire à MINISAT pour résoudre l'instance donnée, y à la justification à posteriori. Seules les instances résolues en moins de 2 millions de conflits y sont représentées (autrement, le calcul de la justification serait biaisé). Cette figure montre clairement une étroite relation entre les justifications calculées et le nombre réels de conflits nécessaires pour terminer la recherche. La justification à posteriori est comprise entre 0.9 et 8.33 fois le nombre réels de conflits nécessaires à MINISAT pour résoudre le problème. Dans la plupart des cas, la justification est approximativement égale à 1.37 le nombre de conflits nécessaires. Ce qui est assez frappant, c'est que l'on ne peut pas faire de distinctions entre les instances SAT et UNSAT : quand une solution existe, elle n'est pas trouvée soudainement, par chance.

2.4 Notion de "clauses glues"

Dans le même article [1], nous montrons que le nombre de différents niveaux de décision d'une clause apprise, donne une bonne estimation de son utilité. Plus petit est ce nombre, "meilleure" semble être la clause apprise. Il est intéressant de noter que cette mesure est optimale pour la clause assertive obtenue par le premier UIP [7, 8]. Associée à une suppression agressive des "*mauvaises*" clauses apprises, cette mesure a donné lieu à un solveur nommé GLUCOSE [1] qui s'est avéré très efficace dans la pratique.

Il est déjà de notoriété que les clauses unaires et binaires sont importantes, mais les clauses de LBD égal à 2 le sont également (notez d'ailleurs que les clauses binaires ont un LBD égal à 2). Elles contiennent un littéral v du dernier niveau de décision (le littéral assertif) et un ensemble de littéraux d'un autre groupe de littéraux S (celui où sera fait le retour arrière), et après retour arrière, le littéral v sera fusionné (collé) avec le groupe de littéraux S, et ce, quelque soit la taille de la clause, que nous appellerons donc clause *glue*. Avec la technique de sauvegarde de la phase, il est fort possible que les variables de cet ensemble S soient toujours affectées avec les mêmes valeurs et, grâce à cette clause *glue*, v restera collé a cet ensemble de littéraux un grand nombre de fois. Cette intuition est juste une hypothèse, et elle n'est encore appuyée par une validation expérimentale, si ce n'est par les performances de solveur GLUCOSE (voir figure 1).

Série	F_1	F_2	F_3	F_4	F_5	F_6
een	78 (72–83)	83 (73–93)	13 (11–16)	76 (69–83)	81 (71–90)	81 (71–91)
goldb	57 (13–89)	53 (14–93)	26 (1–99)	54 (13–80)	50 (14–66)	40 (11–68)
grieu	62 (42–87)	64 (45–86)	11 (1–17)	59 (41–87)	60 (44–86)	59 (34–84)
hoons	43 (25–77)	46 (26–86)	13 (7–24)	39 (20–77)	44 (23–86)	47 (22–98)
ibm-2002	41 (9–94)	46 (9–93)	11 (0–52)	37 (8–93)	40 (8–91)	40 (4–92)
ibm-2004	52 (12–88)	54 (15–90)	16 (0–90)	47 (9–88)	49 (9–88)	46 (10–90)
manol-pipe	60 (9–98)	62 (11–98)	15 (0–91)	57 (5–98)	58 (2–97)	58 (3–97)
miz	66 (30–85)	65 (40–81)	11 (0–39)	63 (25–83)	62 (30–81)	59 (34–79)
schup	75 (53–94)	77 (54–93)	26 (4–79)	74 (53–94)	75 (54–93)	86 (55–95)
simon	72 (31–97)	73 (30–96)	49 (5–97)	64 (16–97)	63 (19–96)	61 (23–89)
vange	30 (14–38)	32 (14–42)	0 (0–2)	29 (10–38)	31 (11–42)	27 (4–38)
velev	50 (6–87)	50 (7–86)	10 (0–90)	45 (6–82)	44 (3–84)	39 (0–85)
all	56 (6–98)	58 (7–98)	15 (0–99)	52 (5–98)	53 (2–97)	50 (0–135)

TAB. 2 – Moyenne des valeurs de r (multiplié par 100 et arrondi pour plus de lisibilité) calculée sur les diverses lois de régression proposées. Les valeurs entre parenthèses sont les valeurs minimum et maximum obtenues.

3 Les niveaux de décision décroissent de plus en plus lentement

Nous avons donc mis en évidence une relation entre le passé de la recherche (en observant uniquement les niveaux de décision) et le nombre de conflits nécessaires pour terminer cette recherche. À partir de là, nous avons essayé d'aller plus loin, en regardant si, à partir d'un certain nombre de conflits, la justification permettait de prédire la fin de la recherche. C'est-à-dire, si il existe un nombre de conflits après lequel le nombre $-b/a$ se stabilise.

Ceci à échoué – sans surprise pourrait-on dire – au moins avec la régression linéaire. La prédiction calculée se déplace constamment vers la droite tout au long de la recherche. Sur certains problèmes, nous avons néanmoins observé une certaine régularité. La prédiction $p(X)$ faite au conflit X ($p(X)$ est égal à $-b/a$ avec a et b les caractéristiques de la loi de régression calculées sur les X premiers conflits) est reliée à celle faite au conflit $X+1$ de la manière suivante : $p(X+1) = p(X) + e$ avec e dépendant de l'instance.

Ce type d'observation suggère que plus le nombre de conflits augmente, plus la droite de régression s'aplatit. C'est pour cela que nous avons testé si une loi de décroissance exponentielle ne se cache pas derrière la décroissance des niveaux de décision.

3.1 Existe-t-il une loi générale guidant la décroissance des niveaux de décision ?

Même si la justification à posteriori donne de bons résultats (voir figure 2) les valeurs r sont très mauvaises. Nous avons donc essayé de faire correspondre notre nuage de points (numéro du conflit, niveau de décision) avec une large famille de loi de régression. La première difficulté est dûe à la grande variance de notre ensemble de données. Même si une tendance générale des lois de régression semble naturelle dans certains cas, nous devons réduire cette variance pour tous ces nuages de points. Il existe de nombreuses méthodes pour lisser ces fluctuations. Nous proposons d'utiliser la moyenne lissée. Chaque ordonnée de notre nuage de points (c'est le niveau de décision) est la moyenne des ordonnées de 2000 points précédents et suivants. Pour accentuer ce lissage nous répétons cette opération 5 fois.

Une fois le nuage de points lissé, nous pouvons tester les valeurs de Bravais sur plusieurs types de courbes. Le procédé que nous avons utilisé sur les courbes de type $F_1 : y = a \times x^b + c$ (a, b, c sont réels) est le suivant. Nous avons itéré la valeur de b entre -10 et 10 par pas de 0.1 et calculé à chaque fois la régression obtenue (nous avons donc calculé à chaque fois les valeurs de a et c) ainsi que le coefficient r. Nous gardons celle qui donne le meilleur coefficient r. À partir de là, nous pouvons tester d'autres types de courbes par de simples transformations mathématiques. Voici celles que nous avons choisies :

$$\begin{aligned} F_2 :&\quad y = exp(a \times x^b + c) \\ F_3 :&\quad y = ln(a \times x^b + c) \\ F_4 :&\quad y = a \times ln(x)^b + c \\ F_5 :&\quad y = exp(a \times ln(x)^b + c) \\ F_6 :&\quad y = a \times exp(x \times b) + c \end{aligned}$$

Le tableau 2 donne une idée des différentes valeurs de r obtenues. La première observation que l'on peut faire est que les valeurs des coefficients r obtenues sont meilleures qu'avec une simple régression linéaire. Ceci est du à l'exposant b, mais aussi au lissage effectué sur les courbes (ce n'est pas le cas dans la section 2). Si nous regardons attentivement, nous remarquons que les courbes F_1 et F_2 sont celles qui suivent le mieux nos nuages de points. Les autres fonctions peuvent donner de très bons résultats sur certains types d'instances (comme F_3 sur les séries `gold`), mais

Series	F_1	F_2
een	78 (72–83)	83 (73–93)
goldb	51 (13–89)	53 (14–93)
grieu	62 (42–87)	64 (45–86)
hoons	77 (77–77)	86 (86–86)
ibm-2002	45 (9–94)	48 (9–93)
ibm-2004	50 (12–84)	55 (15–90)
manol-pipe	59 (9–98)	64 (11–98)
miz	69 (56–80)	70 (60–81)
schup	69 (53–83)	77 (54–93)
simon	75 (55–86)	78 (63–89)
vange	14 (14–14)	32 (14–42)
velev	52 (12–80)	57 (7–86)
all	57 (9–98)	60 (7–98)

TAB. 3 – Valeurs de la table 2, en se focalisant uniquement sur les instances exhibant une décroissance des niveaux de décision.

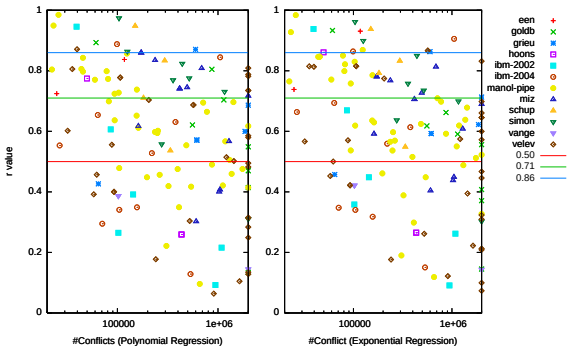

FIG. 3 – Valeurs de r pour deux lois de régressions sélectionnés (gauche : polynômiale (F_1), droite : exponentielle (F_2). Chaque point représente une instance.

leurs moyennes sur toutes les instances est moins bonne que celles des deux premières fonctions. Il est aussi important de noter que de grosses divergences existent sur le coefficient r même au sein d'une même famille.

Nos hypothèses sont basées sur une décroissance des niveaux de décision le long de la recherche, ce qui n'est pas le cas de toutes les instances. La table 3 se concentre sur les instances présentant ces décroissances et sur les régressions de type F_1 et F_2. Les valeurs des coefficients r sont bien meilleures, montrant un meilleur fit sur ces instances. La fonction F_2 semblent donner les meilleurs résultats.

Comme nous venons de le voir, les valeurs de r sont très difficiles à résumer sur un aussi grand nombre d'instances. Nous proposons, dans la figure 3, une vue aérienne de celles-ci, en se focalisant une nouvelle fois sur les régressions de type F_1 et F_2. Chaque point (x, y) correspond à une instance. x correspond au nombre de conflits nécessaires pour la résoudre, et y à la valeur de r. Il est tout d'abord intéressant de noter que les valeurs r semblent devenir de plus en plus mauvaises au fur et à mesure que le nombre de conflits nécessaires pour résoudre l'instance donnée croît (les points semblent se déplacer du haut,gauche vers le bas,droite). Nous pouvons également observer que de nombreuses instances ont un coefficient r relativement bon. Sur les 140 instances testées, la régression polynômiale (de type F_1) a 10 instances avec une valeur de r au dessus de 0.86 et 48 au dessus de 0.71. La régression exponentielle (de type F_2) en a 15 au dessus de 0.86 et 44 au dessus de 0.71.

Arrivés à ce point, il faut bien avouer que le tableau n'est pas très clair. Nous ne sommes pas capable, à partir de données statistiques, de préférer une régression polynômiale ou exponentielle. Néanmoins, et cela est important, il y a très peu de chance qu'une équation de type $y = exp(a \times x^b + c)$ ne puisse pas être approchée par une autre de type $y = a_2 \times x^{b_2} + c_2$, et ceci, lorsque x varie entre 0 et 2 millions et que les données sont très bruitées.

3.2 Courbes et régression

Comme nous venons de le voir, le coefficient r ne nous permet pas de choisir quel type de régression est le plus approprié. Néanmoins, ce coefficient a ses propres limites. En effet, comme nous l'avons montré dans la section 2.3, même de simples lois de régressions linéaires avec un coefficient r très mauvais sont viables dans la pratique. En effet, le coefficient r dépend de la variance des données, et même sur nos nuages de points lissés, les points extrêmes peuvent être très éloignés et avoir une grosse influence dans le calcul de r. Il y a aussi très peu de chances que ce coefficient puisse faire une différence entre deux régressions relativement proches. Pour se convaincre de cela, nous montrons dans la figure 4 les nuages lissés et les lois de régressions calculées sur quelques problèmes sélectionnés parmi les plus représentatifs, avec une préférence sur les instances difficilement résolues (jusqu'à 2 millions de conflits). Nous utilisons $y = exp(a \times x + b)$ comme loi de régression linéaire.

La première courbe `goldb-heqc-i10mul` montre deux caractéristiques intéressantes. Premièrement, après une décroissance régulière durant les 200 000 premiers conflits, il semble que les niveaux de décision restent relativement stables, avec une légère décroissance. Le deuxième point important est le nombre de pics présents sur cette courbe. Ce phénomène est présent sur de nombreux problèmes. Il semble être du à des restarts (soit voulus par le solveur, soit forcés après l'apprentissage d'une clause unaire). Notons tout de même que ce phénomène n'intervient pas systématiquement avec chaque restart. La deuxième courbe `goldb-heqc-i10mul` montre une décroissance qui commence rapidement et qui devient de plus en plus lente avec également quelques pics jusqu'à la fin.

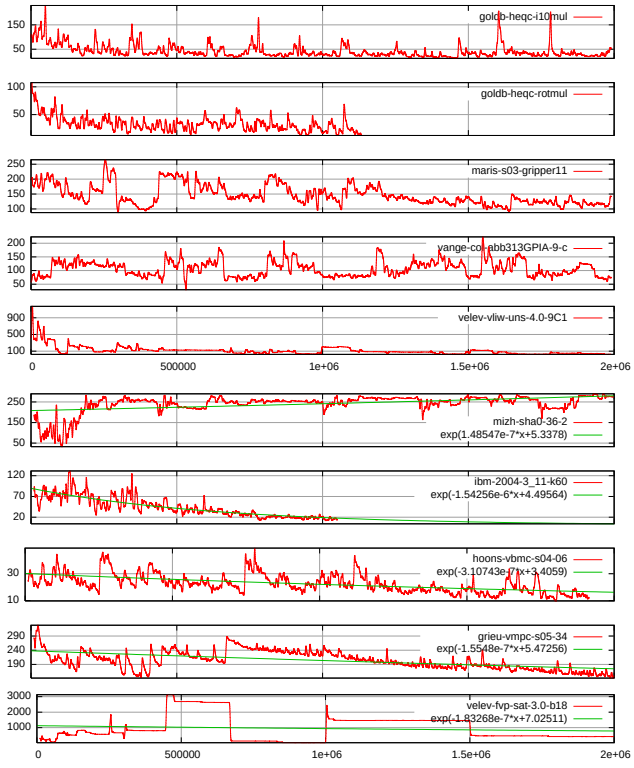

FIG. 4 – Sélection de courbes lissées. Les 5 dernières sont accompagnées de leur loi de régression.

Les courbes `maris` et `vange` montrent des cas typiques avec une très grande variance des données. Enfin, la courbe `velev-vliw` a également une décroissance de plus en plus lente ce qui est, une nouvelle fois, typique.

Les 5 dernières figures montrent, en plus du nuage de points, les lois de régressions calculés. La première (`mizh` montre un cas, relativement rare, de croissance tout au long de la recherche. De telles courbes semblent vraiment dépendre du type de problème encodé (ici des problèmes de cryptographie). La courbe `ibm-2004` est assez incroyable. La loi de régression associée la suit complètement. La courbe `grieu-vmpc-s05-34` est aussi intéressante. Après 700 000 conflits, la courbe se met à décroître régulièrement. Avant cela, une grande variance existe, il est alors difficile d'en déduire une loi de régression. Enfin, la courbe `velev-fvp-sat` montre un cas typique de grandes occiliations. Avoir un coefficient r de bonne qualité semble improbable dans ce cas. Néanmoins, on peut observer que notre hypothèse de départ est ici tout à fait justifiée : les niveaux de décision décroissent toujours et de plus en plus faiblement.

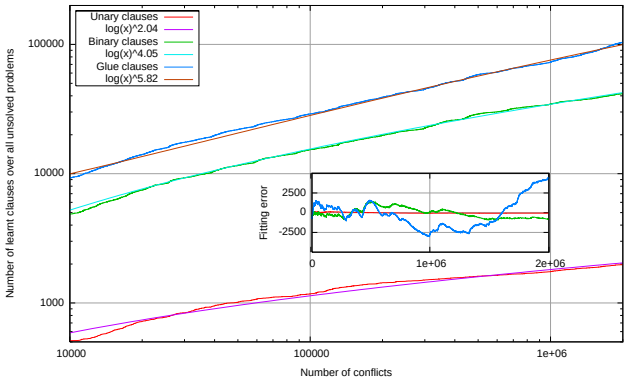

FIG. 5 – Nombre cumulé de clauses apprises (unaires, binaires, glues) produites durant la recherche, restreinte aux problèmes non résolus. En petit, nous avons représenté l'erreur de "fitting".

4 La fréquence des bonnes clauses décroît

La vitesse de décroissance du nombre de décision n'est pas l'unique moyen de mesurer les performances d'un solveur sur une instance donnée. Afin de mieux comprendre les forces et faiblesses des solveurs CDCL, nous avons aussi observé la capacité de ceux-ci de produire des "*bonnes clauses*" durant la recherche. Notre hypothèse est que la décroissance devient de moins en moins rapide parce que le solveur a de plus en plus de mal à générer des clauses intéressantes.

Dans cette section, nous mesurons la *fréquence* d'apparition de certaines clauses, de manière à connaître combien de clauses d'un certain type sont apprises à chaque conflit en moyenne pour l'ensemble des instances. Si la courbe résultante est une droite alors la fréquence d'apprentissage est maintenue tout au long de la recherche. Nous avons donc mesuré les clauses de taille plus petite que 10 et les clauses glues. Nous avons toujours une borne de 2 millions de conflits, mais nous n'avons pris en compte que les instances n'ayant pas été résolues en 2 millions de conflits. Sinon, la fréquence calculée aurait été biaisée. De plus, cela nous permet de nous focaliser sur les instances difficiles.

Les résultats obtenus sont visibles figure 5. On peut noter que la production des clauses unaires et binaires survient de moins en moins au fur et à mesure de la recherche. Nous avons ajouté à cette figure les régression de type $log(x)^b$ calculés sur la clauses unaires et glues. Comme le montre la sous figure interne à la figure 5, le taux d'erreur est très faible, ce qui montre bien que la production cumulée de clauses glues et unaires (c'est la même chose pour les binaires) suit une loi logarithmique : de moins en moins de clauses intéressantes sont produites au fur et à mesure que la recherche avance.

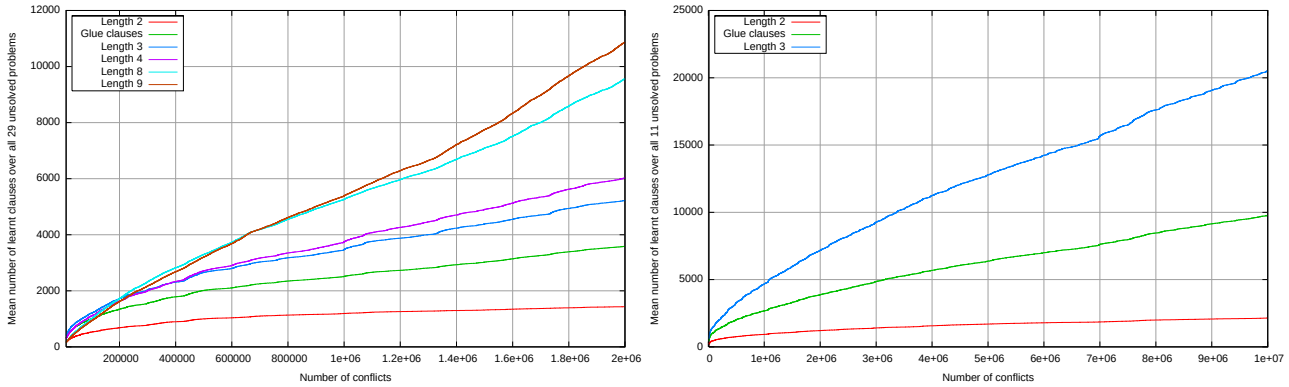

FIG. 6 – Moyenne cumulée de différentes types de clauses apprises avec un arrêt de la recherche à 2 000 000 de conflits (gauche) et 10 000,000 (droite).

La figure 6 de gauche montre le nombre moyen de clauses de différents types appris durant la recherche. La production de clauses glues, unaires et binaires suit une loi logarithmique, ce qui ne semble pas être le cas pour les clauses de plus grande taille. De plus, on produit moins de clauses glues que de clauses ternaires. Pourtant les clauses glues peuvent être de n'importe quelle taille. Pour autant, cette différence semble minime. Pour voir comment cette différence évolue, on a relancé MINISAT, mais cette fois avec un nombre limite de conflits fixé à 10 millions. Il est alors clair que MINISAT a beaucoup de mal à maintenir une bonne production de clauses glues.

Même si cela n'est pas reporté ici, nous avons observé qu'il se passe la même chose entre la génération de deux clauses unaires. Le nombre de clauses glues générés entre la production de deux clauses unaires est de plus en plus rare.

4.1 Obervation de différentes versions de MINISAT

La production de clauses intéressantes décroît donc tout au long de la recherche, pour dédouaner ce résultat des particularités de MINISAT, nous avons réalisé les mêmes expérimentations, mais avec différentes versions de ce solveur. La première version utilise des redémarrages très rapides (tous les 250 conflits), une avec un vardecay de 0.85 au lieu de 0.95 (cette constante contrôle la vitesse à laquelle le solveur oublie les variables lors du choix heuristique), et enfin une avec la sauvegarde de la phase [12] et des redémarrages à la luby [6]. Ici, une limite de 2 millions de conflits est à nouveau utilisée.

Les résultats sont visibles sur la figure 7. Il est frappant de noter que la version avec luby et sauvegarde de la phase produit peu de clauses unaires, mais beaucoup de clauses binaires et glues. Si on regarde attentivement, on peut également remarquer que la différence entre cette version et avec un restart250 n'est pas si importante. Il semble que la version phase+luby soit meilleure pour produire des clauses binaires, et que la production de clauses glues paie vraiment. Nous pouvons aussi noter que la version vardecay est meilleure que MINISAT dans la production de clauses glues, même si elle donne au final de plus mauvais résultats (voir la sous figure cactus). La notion de clauses glues n'est donc pas suffisante pour expliquer le bon ou le mauvais comportement des différentes versions. Nous pensons que ces clauses sont importantes mais il reste encore du travail pour appréhender totalement leur rôle dans les démonstrateurs CDCL.

5 Conclusion

Dans cet article, nous avons considéré le solveur MINISAT comme un système physique a étudier. Nous nous sommes focalisés sur deux points qui nous semblent importants : les niveaux de décision et certaines caractéristiques des clauses apprises durant la recherche (taille et nombre de niveaux de décision). A partir de là, nous avons construit un certain nombre d'expériences pour valider ou invalider certaines hypothèses.

Ces premiers résultats tendent à montrer que dans de nombreux cas, MINISAT réduit, au fur et à mesure que la recherche avance, le nombre de niveaux de décision nécessaire pour atteindre un conflit. Néanmoins, au plus cette recherche avance, au plus il semble difficile de réduire ces niveaux de décision. Nous suspectons que ces niveaux de décision suivent une courbe exponentielle qui expliquerai pourquoi les solveurs CDCL présentent des figures cactus. De plus, nous montrons, que plus la recherche avance, moins les clauses apprises semblent intéressantes.

Références

[1] G. Audemard and L. Simon. Predicting learnt clauses quality in modern sat solvers. In *proceedings of IJ-*

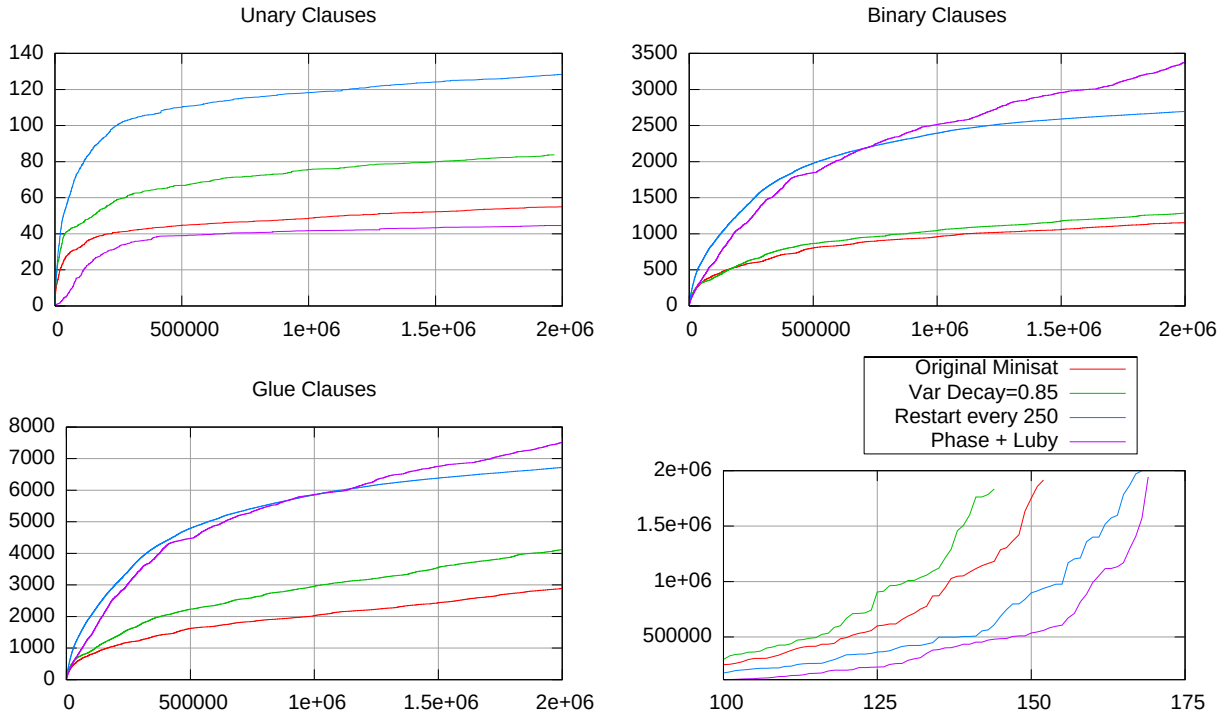

FIG. 7 – Fréquence de production de certains types de clauses pour les quatre versions de MINISAT. La figure en bas à gauche donne la figure de type cactus, du point de vue des conflits (sur tous les problèmes résolus).

CAI, 2009. to appear.

[2] A. Biere. PicoSAT essentials. *Journal on Satisfiability*, 4 :75–97, 2008.

[3] N. Eén and N. Sörensson. An extensible SAT-solver. In *proceedings of SAT*, pages 502–518, 2003.

[4] C. Gomes, B. Selman, N. Crato, and H. Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning*, 24 :67–100, 2000.

[5] J. N. Hooker. Needed : An empirical science of algorithms. *Operations Research*, 42 :201–212, 1994.

[6] J. Huang. The effect of restarts on the efficiency of clause learning. In *proceedings of IJCAI*, pages 2318–2323, 2007.

[7] S. Jabbour and L. Sais. personnal communication, February 2008.

[8] Said Jabbour. *De la satisfiabilité propositionnelle aux formules booléennes quantifiées*. PhD thesis, Université d'Artois, 2008.

[9] Chu Min Li and Anbulagan. Look-ahead versus look-back for satisfiability problems. In *Principles and Practice of Constraint Programming (CP97)*, volume 1330 of *LNCS*, pages 341–355. Springer Verlag, 1997.

[10] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff : Engineering an efficient SAT solver. In *proceedings of DAC*, pages 530–535, 2001.

[11] J. Peter and R. Hull. *Le principe de Peter*. 1969.

[12] K. Pipatsrisawat and A. Darwiche. A lightweight component caching scheme for satisfiability solvers. In *proceedings of SAT*, pages 294–299, 2007.

[13] M. Prasad, A. Biere, and A. Gupta. A survey of recent advances in SAT-based formal verification. *journal on Software Tools for Technology Transfer*, 7(2) :156–173, 2005.

[14] João P. Marques Silva and Karem A. Sakallah. Grasp - a new search algorithm for satisfiability. In *proceedings of ICCAD*, pages 220–227, 1996.

[15] R. Williams, C. Gomes, and B. Selman. Backdoors to typical case complexity. In *Proceedings of IJCAI'03*, 2003.

[16] L. Zhang and C. Madigan. Efficient conflict driven learning in a boolean satisfiability solver. In *proceedings of ICCAD*, pages 279–285, 2001.

[17] Lintao Zhang and Sharad Malik. Cache performance of sat solvers : a case study for efficient implementation of algorithms. In *proceedings of SAT*, pages 287–298, 2003.