Des ensembles Horn strong backdoor aux ensembles ordonnés strong backdoor

Lionel Paris, Richard Ostrowski, Pierre Siegel

Des ensembles Horn strong backdoor aux ensembles ordonnés strong backdoor

Lionel Paris **Richard Ostrowski** **Pierre Siegel**
LSIS, Université de Provence
CMI, Technopôle de Château Gombert, 13013 Marseille, France
{lionel.paris, richard.ostrowski, pierre.siegel}@lsis.org

Résumé

L'identification et l'exploitation de structures cachées dans un problème est reconnue comme étant un moyen fondamental pour contrecarrer l'explosion combinatoire de sa résolution. Récemment, une structure particulière appelée (strong) backdoor a été identifiée pour le problème de satisfaisabilité de formule CNF (SAT). Certaines connexions entre les ensembles strong backdoor et la difficulté intrinsèque des problèmes SAT ont été mises en évidence, permettant une meilleure approximation de la borne de complexité en temps dans le pire des cas. On peut calculer des ensembles strong backdoor pour chaque classe polynomiale. Dans [13], une méthode d'approximation d'ensembles strong backdoor pour la classe des formules de Horn a été proposée. Cette approximation est réalisée en deux étapes. Dans un premier temps, on calcule le meilleur Horn renommage du point de vue du nombre de clauses de Horn de la CNF de départ. Ensuite on extrait un ensemble Horn strong backdoor de la partie non Horn de la formule renommée. Dans cet article, nous proposons de calculer des ensembles Horn strong backdoor en utilisant le même procédé mais en minimisant le nombre de littéraux positifs dans la partie non Horn de la formule renommée au lieu du nombre de clauses. Puis nous étendons cette méthode à la classe des formules ordonnée [2] qui est une extension de la classe des formules de Horn. Cette méthode nous garantit l'obtention d'ensembles Ordonné strong backdoor de taille plus petite ou égale à ceux des ensembles Horn strong backdoor (jamais plus grande). Les résultats expérimentaux montrent que ces nouvelles méthodes permettent de réduire la taille des ensembles strong backdoor sur certaines instances et que leur exploitation permet également d'améliorer les performances des solveurs SAT.

Abstract

Identifying and exploiting hidden problem structures is recognized as a fundamental way to deal with the intractability of combinatorial problems. Recently, a particular structure called (strong) backdoor has been identified in the context of the satisfiability problem. Connections has been established between backdoors and problem hardness leading to a better approximation of the worst case time complexity. Strong backdoor sets can be computed for any tractable class. In [13], a method for the approximation of strong backdoor sets for the Horn-Sat fragment was proposed. This approximation is realized in two steps. First, the best Horn renamming of the original CNF formula, in term of number of clauses, is computed. Then a Horn strong backdoor set is extracted from the non Horn part of the renamed formula. in this article, we propose computing Horn strong backdoor sets using the same scheme but minimizing the number of positive literals in the non Horn part of the renamed formula instead of minimizing the number of non Horn clauses. Then we extend this method to the class of ordered formulas [2] which is an extension of the Horn class. This method insure to obtain ordered strong backdoor sets of size less or equal than the size of Horn strong backdoor sets (never greater). Experimental results show that these new methods allow to reduce the size of strong backdoor sets on several instances and that their exploitation also allow to enhance the efficiency of satisfiability solvers.

1 Introduction

Le problème de satisfaisabilité de formules propositionnelles (SAT) qui consiste à décider si une formule Booléenne mise sous forme normale conjonctive (CNF) est satisfaisable est un des problèmes NP-complet les plus étudiés de part son importance pratique et théorique. Du fait des récents progrès effectués sur la résolution pratique du problème SAT, de nombreuses applications (planification, vérification formelle, équivalence et simplification de circuits, etc.) sont codées et résolues en utilisant ce formalisme. La plupart des solveurs complets actuels sont basés sur un algorithme de recherche de type backtrack, appelé procédure de DPLL (Davis, Putnam, Logemann et Loveland) [3]. Cet algorithme basique est couplé avec plu-

sieurs méthodes d'apprentissage ou de filtrages importantes comme des formes étendues de propagation de contraintes Booléennes, de pré-traitements, de détection de symétries, etc.

La seconde classe d'algorithme pour tester la satisfaisabilité est basée sur des méthodes de recherche locale [14, 10]. Ces techniques ne peuvent pas prouver l'insatisfaisabilité, car l'espace de recherche est exploré de manière non systématique. Cependant, ces algorithmes atteignent des performances remarquables sur des instances difficiles satisfaisables, même de grandes tailles (y compris sur les instances aléatoires).

Récemment, plusieurs auteurs se sont intéressés à la détection d'éventuelles structures cachées dans des instances SAT (*e.g.* backbones [4], backdoors [17], équivalences [7] et dépendances fonctionnelles [5]), permettant d'expliquer en partie et d'améliorer l'efficacité des solveurs SAT sur des instances issues du monde réel de grandes tailles. D'autres résultats théoriques importants (*e.g.* heavy tailed phenomena [17], backbones [6] et backdoors [16]) ont été obtenus, contribuant à une meilleur compréhension de la difficulté intrinsèque des problèmes. L'efficacité des solveurs pouvant être encore une fois expliquée en partie par ces résultats.

Dans ce papier, nous nous intéressons au problème du calcul d'ensembles strong backdoor. Rappelons qu'un ensemble de variables forme un backdoor pour une formule donnée s'il existe une interprétation des ces variables, pour laquelle la formule simplifiée peut être résolue en temps polynomial. Un tel ensemble de variables est appelé ensemble strong backdoor si toute interprétation de ces variables mène à une sous-formule polynomiale [1].

Calculer le plus petit ensemble strong backdoor est un problème NP-difficile [12]. L'approximation (en temps polynomial) du plus petit ensemble strong backdoor est un challenge intéressant et important. De précédents travaux ont abordé cette question. Par exemple, l'approche proposée dans [5] essaie tout d'abord d'identifier un ensemble de portes logiques (fonctions Booléennes) à partir d'une CNF donnée. Puis, en appliquant des heuristiques pour déterminer un ensemble coupe-cycle pour le graphe représentant la formule hybride. On obtient un ensemble strong backdoor composé à la fois de l'ensemble des variables indépendantes et de celles de l'ensemble coupe-cycle. Cependant, sur certaines instances où aucune porte logique n'est identifiée, le strong backdoor coïncide avec l'ensemble des variables du problème. D'autres approches ont été proposées. Elles utilisent différentes techniques, telles que les algorithmes de recherche systématique adaptée [16, 6]. Récemment, dans [9] un phénomène de corrélation intéressant a été observé entre la satisfaisabilité et les performances des solveurs SAT pour des problèmes 3-SAT générés aléatoirement avec une densité fixée en fonction de la partie reverse Horn sub-optimale d'une formule CNF. Dans [8], la relation entre la complexité de résolution d'instances aléatoires 3-SAT insatisfaisables, la taille des noyaux insatisfaisables et les ensembles strong backdoor a été mise en valeur.

Les buts principaux de ce papier sont dans un premier temps de concevoir une approche polynomiale qui fournit un ensemble strong backdoor de taille raisonnable. En se basant sur les travaux de [13], nous proposons une nouvelle heuristique permettant d'améliorer, de manière significative, la taille de l'ensemble strong backdoor en considérant la classe polynomiale des formules de Horn. Puis nous étendons ces résultats à une autre classe polynomiale qui est une extension naturelle des formules de Horn : les formules ordonnées [2].

Le papier est organisé comme suit. Dans une première partie nous donnons les définitions préliminaires au contexte dans lequel se situe ce papier et rappelons la méthode utilisée dans [13]. Nous présentons ensuite notre approche, basée sur une recherche locale, pour le calcul d'ensembles Horn strong backdoor. Dans une troisième partie nous présentons une généralisation des formules de Horn à savoir les formules ordonnées. Pour chacune de ces parties, nous présentons quelques résultats expérimentaux montrant l'intérêt de notre approche. Enfin, nous présentons des résultats expérimentaux, d'un point de vue pratique, montrant que cette approche, pour le calcul et l'exploitation d'ensembles Horn strong backdoor, apporte des améliorations notables sur l'efficacité d'un des meilleurs solveurs SAT. Pour finir, nous discutons de la portée de ces résultats et donnons quelques perspectives à donner à ce travail.

2 Définitions préliminaires et notations

Soit $\mathcal{B}$ un langage Booléen (*i.e.* propositionnel) de formules formées de manière standard, en utilisant les connecteurs usuels ($\vee$, $\wedge$, $\neg$) et un ensemble de variables propositionnelles. Une *formule CNF* Σ est un ensemble (interprété comme une conjonction) de *clauses*, où chaque clause est un ensemble (interprété comme une disjonction) de *littéraux*. Un littéral est une occurrence d'une variable propositionnelle soit sous forme positive, soit sous forme négative. Rappelons que toute formule Booléenne peut se réécrire sous forme d'une CNF en temps linéaire en utilisant la transformation de Tseitin [15]. La taille d'une CNF Σ est définie par $\sum_{c\in\Sigma} |c|$ où $|c|$ est le nombre de littéraux de c. Une clause *unitaire* (resp. *binaire*) est une clause de taille 1 (resp. 2). Un *littéral unitaire* est l'unique littéral d'une clause unitaire. Une clause unitaire $\mathcal{C} = \{l\}$ est dite positive si son littéral unitaire l est positif ($l \in \mathcal{V}$). On note $nbVar(\Sigma)$ (resp. $nbCla(\Sigma)$) le nombre de variables (resp. clauses) de Σ. $\mathcal{V}(\Sigma)$ (resp. $\mathcal{L}(\Sigma)$) est l'ensemble des variables (resp. littéraux) ap-

[1] Pour laquelle le test de satisfaisabilité peut être réalisé en temps polynomial.

paraissant dans Σ. L'ensemble $\mathcal{L}(\Sigma)$ est l'union des littéraux positifs $\mathcal{L}^+(\Sigma)$ et des littéraux négatifs $\mathcal{L}^-(\Sigma)$ présents dans la formule Σ. Un ensemble de littéraux $S \subset \mathcal{L}(\Sigma)$ est consistant $\Leftrightarrow \forall l \in S, \neg l \notin S$. Un littéral l est dit monotone si $\neg l \notin \mathcal{L}^-(\Sigma)$. On note $\mathcal{V}^+(\Sigma)$ (resp. $\mathcal{V}^-(\Sigma)$) l'ensemble des variables apparaissant au moins une fois positivement (resp. négativement) dans Σ. Pour tout littéral l, $Occ_\Sigma(l)$ représente l'ensemble $\{C \in \mathcal{F} | l \in C\}$ *i.e.* l'ensemble des clauses de Σ dans lesquelles le littéral l apparaît. On notera simplement $Occ(l)$ lorsqu'aucune confusion n'est possible.

Une *interprétation complète* d'une formule Booléenne est une affectation de ses variables à une valeur de vérité $\{vrai, faux\}$. Une variable x est satisfaite (resp. falsifiée) pour I si $I[x] = vrai$ (resp. $I[x] = faux$). Une interprétation I peut être représentée comme un ensemble de littéraux. Un littéral $l \in I$ (resp. $\neg l \in I$) si $I[l] = vrai$ (resp. $I[l] = faux$). Une *interprétation incomplète* d'une formule Booléenne est une affectation d'un sous-ensemble de ses variables. Si une variable n'apparaît ni sous forme positive, ni sous forme négative dans une interprétation, on dit que cette variable n'est pas affectée. On définit $I(\Sigma)$ comme la formule simplifiée par I. Formellement $I(\Sigma) = \{C | C \in \Sigma, I \cap C = \emptyset\} \cup \{C \setminus \sim I | C \in \Sigma, C \cap \sim I \neq \emptyset\}$. Un *modèle* pour une formule est une interprétation qui satisfait la formule. Le problème SAT est le problème de décision de la satisfaisabilité d'une CNF (existence d'un modèle). On définit un *renommage* R de $\mathcal{V}(\Sigma)$ comme l'application de $\mathcal{V}(\Sigma)$ dans $\{vrai, faux\}$. l_R est le littéral correspondant au renommage du littéral l par R. Si $R(\mathcal{V}(l)) = vrai$, alors $l_R = \neg l$, sinon $l_R = l$. Les variables renommées sont donc celles qui ont une valeur à $vrai$ dans R.

L'idée proposée dans [13], consiste à calculer un ensemble Horn strong backdoor par une approche stochastique. Dans une première phase, un renommage est calculé afin de minimiser la partie non Horn. Pour ce faire, les définitions suivantes sont données dans [13] :

Définition 1 (Formule renommée) *Soient Σ une formule CNF, R un renommage de $\mathcal{V}(\Sigma)$. On définit la formule renommée Σ_R comme la formule obtenue en substituant, pour toutes les variables x telles que $R(x) = vrai$, toutes les occurrences de x (resp. $\neg x$) par $\neg x$ (resp. x). x est alors renommée dans Σ. Quand Σ_R est une formule de Horn, R est appelé un Horn-renommage de Σ.*

On dit qu'un littéral l est positif dans c pour R si $l \in c$ et $\neg l \in R$, ou bien si $\neg l \in c$ et $l \in R$. On note ceci $estPos(l, c, R)$. Ce même littéral est négatif dans c pour R si $l \in c$ et $l \in R$, ou bien $\neg l \in c$ et $\neg l \in I$. On note ceci $estNeg(l, c, R)$.

On note le nombre de littéraux positifs (resp. négatif) dans c pour R par $nbPos(c, R)$ (resp. $nbNeg(c, R)$).

On note $nbPosTot(\Sigma, R)$ le nombre total de littéraux positifs présents dans la partie non Horn de Σ pour le renommage R.

Ce qui permet de définir la notion de *Horn-renommabilité* :

Définition 2 (Horn-renommabilité) *Soient Σ une formule CNF et R un renommage. Une clause $c \in \Sigma$ est dite Horn-renommée par R (notée $h\_ren(c, R)$) si $nbPos(c, R) \leq 1$ i.e. c contient au plus un littéral positif pour R ; sinon elle est dite Horn-falsifiée par R (notée $h\_fal(c, R)$).*

On note $nbHorn(\Sigma, R)$ le nombre de clauses de Σ Horn-renommées par R.

L'algorithme 1 récapitule ce principe de calcul du meilleur renommage selon la fonction objectif (notée $h\_score()$) suivante :

Définition 3 (Fonction Objectif) *Soient Σ une formule CNF, $x \in \mathcal{V}(\Sigma)$, R un renommage et R_x le renommage obtenu à partir de R en inversant la valeur de vérité de x. On définit $h\_breakCount(x, R) = |\{c | h\_ren(c, R), h\_fal(c, R_x)|$ et $h\_makeCount(x, R) = |\{c | h\_fal(c, R), h\_ren(c, R_x)\}|$. On définit $h\_score(x, R) = h\_makeCount(x, R) - h\_breakCount(x, R)$.*

Ensuite une fois ce renommage calculé, une méthode gloutonne est utilisée afin de construire un ensemble strong backdoor. Le principe est de choisir la variable apparaissant le plus souvent dans la partie non Horn mais en ne considérant que les littéraux positifs de ces clauses. L'algorithme 2 montre comment on construit cet ensemble Horn strong backdoor. Les résultats expérimentaux ont montré l'utilité de calculer cet ensemble. D'une part, cela permet de raffiner la borne de complexité pour les différentes instances, et d'autre part, du point de vue résolution, de nombreuses améliorations en terme de temps et de nombre de nœuds ont été constatées.

3 Les formules de Horn

Dans cette partie, nous présentons une nouvelle fonction objectif pour le calcul du meilleur Horn renommage. L'approche utilisée pour calculer l'ensemble Horn strong backdoor à partir de la formule renommée avec un meilleur renommage reste la même.

3.1 Une Nouvelle Fonction Objectif pour le calcul du Renommage

Contrairement à celle proposée dans [13], cette nouvelle fonction objectif prend en compte la taille de la sous-formule non Horn, et plus précisément le nombre de littéraux positifs apparaissant dans la partie non

Algorithm 1 Algorithme WalkHorn

Fonction WalkHorn
Entrée : Une formule CNF Σ
Sortie : Un *renommage* de Σ

Initialiser R_{max} avec toutes les variables de Σ à vrai
pour $i = 1$ à *MAX_TRIES* **faire**
  $R =$ interprétation générée aléatoirement
  pour $j = 1$ à *MAX_FLIPS* **faire**
    si $nbHorn(\Sigma, R) = nbCla(\Sigma)$ **alors**
      retourner R {Σ *est Horn renommable*}
    fin si
    si $(\forall c \in \Sigma, nbPos(c, R) > 0)$ **ou** $(\forall c \in \Sigma, nbNeg(c, R) > 0)$ **alors**
      retourner R {Σ *est satisfaisable*}
    fin si
    {*Horn Random Walk Strategy*}
    Avec une probabilité p **faire**{p est un paramètre fixé arbitrairement}
    Sélectionner aléatoirement une clause non Horn c et un littéral l de c
    $R = R - \{l\} \cup \{\neg l\}$
    fait
    Avec une probabilité $1 - p$ **faire**
    Soit $l \in R$ tel que $\forall l' \in R$ avec $l \neq l'$, $h\_score(l, R) > h\_score(l', R)$
    $R = R - \{l\} \cup \{\neg l\}$
    fait
    si $nbHorn(\Sigma, R) > nbHorn(\Sigma, R_{max})$ **alors**
      $R_{max} = R$
    fin si
  fin pour
fin pour
retourner R_{max}

Horn de la formule. Cette fonction objectif va tenter de trouver un renommage qui minimise cette taille car il parait raisonnable de penser que si l'on arrive à diminuer le nombre total de littéraux présents dans la partie non Horn (quitte à avoir plus de clauses non Horn), on devrait aussi diminuer le nombre de variables présentes dans l'ensemble Horn strong backdoor que l'on va calculer à partir de cette partie non Horn.

La seule chose que nous ayons à faire pour mettre la nouvelle fonction objectif en œuvre, c'est de redéfinir les compteurs de *Horn-Break* et de *Horn-Make*, permettant à la fonction *h_score* de l'algorithme 1 d'être en adéquation avec cette fonction objectif.

Définition 4 (Min {Break/Make} Count) *Soient Σ une formule CNF, $x \in \mathcal{V}(\Sigma)$, R un renommage et R_x le renommage obtenu à partir de R en inversant la valeur de vérité de x. Soient* $h\_Break_1 = \{c \in \Sigma | nbPos(c, R) = 1 \text{ et } estNeg(x, c, R)\}$, $h\_Break_2 = \{c \in \Sigma | nbPos(c, R) > 1 \text{ et } estNeg(x, c, R)\}$, $h\_Make_1 = \{c \in \Sigma | nbPos(c, R) = 2 \text{ et } estPos(x, c, R)\}$ *et* $h\_Make_2 = \{c \in \Sigma | nbPos(c, R) > 2 \text{ et } estPos(x, c, R)\}$. *Les compteurs sont définis comme suit :*

$hM\_breakCount(x, R) = 2 \times |h\_Break_1| + |h\_Break_2|$ *et* $hM\_makeCount(x, R) = 2 \times |h\_Make_1| + |h\_Make_2|$. *Puis on définit* $hM\_score(x, R) = hM\_makeCount(x, R) - hM\_breakCount(x, R)$

Algorithm 2 Horn Strong Backdoor

Fonction Backdoor
Entrée : Σ : une formule CNF
Sortie : B : Un Ensemble Horn Strong Backdoor

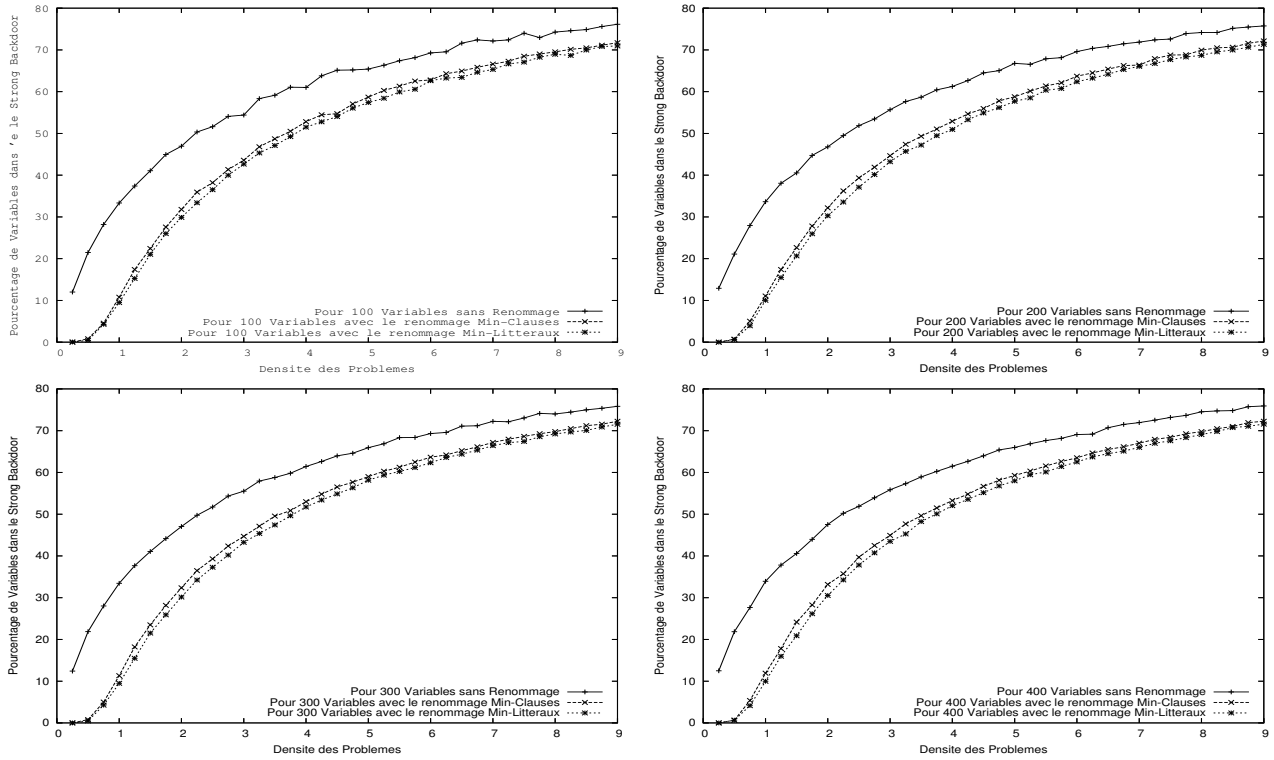
init $B = \emptyset$
$\psi = \{c | c \in \Sigma, |c^+| > 1\}$
répéter
  choisir une variable v de $\mathcal{V}^+(\psi)$
  Soit l un littéral positif de v
  $\psi = \{c \backslash \{l\} | c \in \psi, l \in c^+\} \cup \{c | c \in \Sigma, l \notin c^+\}$
  $B = B \cup \{v\}$
jusqu'à ce que ψ soit de Horn
retourner B

L'ensemble h_Break_1 correspond à l'ensemble des clauses qui sont Horn-renommées par R et qui ne le sont pas par R_x. Cela signifie que chacune de ces clauses contient *deux* littéraux positifs pour R_x, et qu'il faut rajouter *deux* littéraux pour chaque clause de $Break_1$ à $nbPosTot(\Sigma, R)$ si l'on passe de R à R_x (*i.e.* $nbPosTot(\Sigma, R_x) = 2 + nbPosTot(\Sigma, R)$ pour chacune de ces clauses). L'ensemble h_Break_2 correspond à l'ensemble des clauses qui ne sont pas Horn-renommées par R et dans lesquelles x est négatif pour R. Ces clauses ne sont pas Horn-renommées par R_x non plus et contiennent toutes un littéral positif de plus par rapport à R_x (x en l'occurrence). Il faut donc rajouter *un* littéral à $nbPosTot(\Sigma, R)$ pour chacune de ces clauses dans l'hypothèse où l'on passe de R à R_x.

D'un autre côté, l'ensemble h_Make_1 correspond à l'ensemble des clauses qui ne sont pas Horn-renommées par R et qui le sont pour R_x. Ce qui signifie que chacune de ces clauses contient *deux* littéraux positifs pour R qu'il faut soustraire à $nbPosTot(\Sigma, R)$ pour chacune de ces clauses lors du passage de R à R_x. Et enfin, l'ensemble h_Make_2 correspond à l'ensemble des clauses qui ne sont Horn-renommées ni par R, ni par R_x et dans lesquelles x est positif pour R. Il faut donc retirer *un* littéral à $nbPosTot(\Sigma, R)$ pour chacune des clauses de cet ensemble.

Il ne reste plus qu'à remplacer l'appel à la fonction *h_score* de l'algorithme 1 par la fonction hM_score, et nous pouvons calculer le meilleur renommage minimisant le nombre de littéraux positifs des clauses qui ne sont pas de Horn.

FIG. 1 – Tailles des ensembles Horn strong backdoor pour des instances 3-SAT aléatoires



3.2 Expérimentations

Nous avons testé cette méthode de calcul d'ensemble strong backdoor dans un premier temps sur les instances générées aléatoirement. Nous avons testé sur des instances comprenant 100, 200, 300 et 400 variables. Pour chaque classe nous avons fait varier le rapport $\frac{\#C}{\#V}$ de 0.25 jusque 9. Pour chaque ratio, nous avons générés 50 instances et calculé la moyenne du nombre de variables apparaissant dans l'ensemble Horn strong backdoor suivant trois approches :

1. nous avons considéré la formule originale (sans renommage) et calculé un strong backdoor à l'aide de l'algorithme 2.
2. nous avons considéré la formule renommée par le meilleur renommage fournie par l'algorithme 1 selon le critère du nombre de clauses Horn-renommées et calculé un strong backdoor à l'aide de l'algorithme 2. Ce renommage sera appelé renommage *Horn_Min_Clause* (HMC).
3. nous avons considéré la formule renommée par le meilleur renommage fournie par l'algorithme 1 selon le critère du nombre de littéraux positifs figurant dans la partie non Horn de la formule renommée et calculé un strong backdoor à l'aide de l'algorithme 2. Ce renommage sera appelé renommage *Horn_Min_Littéraux* (HML).

La figure 1 récapitule les résultats pour chaque classe d'instances. On remarque que comparativement à la formule originale, nous obtenons de meilleurs résultats en terme de taille de l'ensemble Horn strong backdoor quelque soit le renommage choisi. De plus la nouvelle heuristique que nous proposons se trouve être légèrement meilleure que celle proposée dans [13] en terme de taille des ensembles Horn strong backdoor. Le tableau 1 montre quelques résultats expérimentaux de notre approche sur des instances issues des précédentes compétitions pour le problème SAT. Pour chaque instance, nous donnons la taille de l'instance ($\#V$ et $\#C$) ainsi que la taille de l'ensemble Horn strong backdoor proposé dans [13] *i.e.* en utilisant HMC) et par notre nouvelle approche (*i.e.* en utilisant HML). Les résultats expérimentaux montrent clairement l'intérêt de minimiser le nombre de variables apparaissant dans la partie non Horn, plutôt que d'essayer de minimiser le nombre de clauses dans la partie non Horn dans un premier temps et de calculer ensuite un strong backdoor pour cet ensemble de clauses non Horn. De manière générale notre approche donne toujours un nombre de clauses dans la partie non Horn supérieure à [13], et nous donne au pire une taille d'ensemble Horn strong backdoor égale à celle donnée dans [13].

Instance	$\#V$	$\#C$	$\|SB_{HMC}\|$	$\|SB_{HML}\|$
fifo8_100	64762	176313	22933	21065
Mat323	33831	117126	17396	16164
Mat317	24435	85050	12606	11727
comb2	31933	112462	14652	13476
ip36	47273	153368	21584	20667
ip38	49967	162142	22911	21894
ip50	66131	214786	30280	29301
f2clk_50	34678	101319	13923	13028
vda_gr_rcs_w8	5776	116522	4245	3807
too_large_gr_rcs_w9	4671	64617	3397	3024
dp12u11	11137	30792	3584	3315
dp11u10	9197	25271	2828	2664
dp10u09	7611	20770	2385	2200

TAB. 1 – Tailles des ensembles Horn strong backdoor pour des instances réelles.

4 Formules ordonnées

Dans cette section, nous montrons comment il est possible de calculer des ensembles strong backdoor pour une autre classe polynomiale du problème SAT : la classe des formules ordonnées introduite par Benoist et Hébrard en 1999 [2].

4.1 Description

La classe des formule ordonnées peut être vue naturellement comme une extension de la classe des formules de Horn. Elles est basée sur la notion de littéraux libres et de littéraux liés [2].

Définition 5 (Littéraux libres/liés) *Soient $C \in \Sigma$ une clause et $l \in C$ un littéral. On dit que l est lié dans C (par rapport à Σ) si $Occ(\neg l) = \emptyset$; ou s'il existe $t \in C (t \neq l)$ tel que $Occ(\neg l) \subseteq Occ(\neg t)$. Si l n'est pas lié dans C, on dit que l est libre dans C.*

Les formules ordonnées sont définies ainsi [2] :

Définition 6 (Formules Ordonnées) *Une formule CNF Σ est ordonnée si chaque clause $C \in \Sigma$ contient au plus un littéral positif libre dans C.*

En rappelant qu'une formule CNF est une formule de Horn si chacune de ses clauses ne contient pas plus d'un littéral positif, on voit bien que toute formule de Horn est forcément ordonnée, ce qui étaye l'idée que la classe des formules ordonnées est une extension de la classe des formules de Horn.

On sait que si Σ est une formule de Horn et ne contient aucune clause unitaire positive, alors Σ est satisfaisable. Cette propriété reste vraie si Σ est ordonnée. Pour les formules ordonnées, nous avons également la proposition suivante (prouvée dans [2]) :

Proposition 1 *Si Σ est ordonnée et ne contient aucune clause unitaire positive $C = \{x\}$ telle que x est libre dans C, alors Σ est satisfaisable.*

De plus, il est montré dans [2] qu'une formule ordonnée est stable par propagation unitaire *i.e.* une formule ordonnée reste ordonnée quelque soient les propagations unitaire que l'on y réalise. De ce fait, avec la proposition 1, on voit que le problème de satisfaisabilité d'une formule ordonnée peut être résolu en temps linéaire.

Enfin, comme pour les formules de Horn, la classe des formules ordonnées peut être étendue par la classe des formules ordonnées renommables. Il existe un algorithme polynomial pour la reconnaissance de ces formules et un autre, polynomial également, pour décider de la satisfaisabilité d'une formule ordonnée renommable [2].

Ainsi, nous pouvons envisager de calculer des ensembles ordonnés strong backdoor en utilisant la même approche que pour les ensembles Horn strong backdoor *i.e.* pour une formule CNF originale, trouver un renommage maximisant la sous-formule ordonnée et extraire un ensemble ordonné strong backdoor de la sous-formule non-ordonnée de la formule renommée.

4.2 Approximation de la sous-formule ordonné-renommable maximale

Comme dans le cas des formules de Horn, il existe un algorithme de complexité polynomial pour savoir si une formule est ordonné-renommable ([2]), et comme pour les formule de Horn, si la formule n'est pas ordonné-renommable, calculer la sous-formule ordonné-renommable maximale est un problème NP-difficile. En effet, dans le pire des cas, si aucun littéral n'est lié, la classe des formule ordonnée est équivalente à la classe des formules de Horn.

La proposition suivante nous permet d'exploiter l'algorithme 1 pour approximer la sous-formule ordonné-renommable maximale :

Proposition 2 *Soient Σ une formule CNF et $c \in \Sigma$ une clause de Σ. Soient l et t deux littéraux de c. Soit R un renommage des variables de Σ ($\mathcal{V}(\Sigma)$). Si l est lié à t dans c par rapport à Σ, alors l_R est toujours lié à t_R dans c_R par rapport à Σ_R.*

Preuve *Il suffit de constater que si une variable $v \in \mathcal{V}(\Sigma)$ est renommée dans R, alors, pour les littéraux qui lui sont associés v et $\neg v$, on a $Occ_\Sigma(v) = Occ_{\Sigma_R}(v_R) = Occ_{\Sigma_R}(\neg v)$ et $Occ_\Sigma(\neg v) = Occ_{\Sigma_R}(\neg v_R) = Occ_{\Sigma_R}(v)$.*

Cette proposition nous assure que la liaison entre deux littéraux est stable par renommage. Ainsi, pour une formule CNF donnée, il suffit de calculer une seule fois au préalable une table des liaisons pour chaque littéral de la formule, et à tout moment, il suffira de regarder dans cette table si un littéral est lié ou non (en fonction du renommage).

Ici encore, deux critères pour calculer le meilleur renommage sont possibles : minimiser le nombre de

clauses non ordonnées, ou bien minimiser le nombre de littéraux positifs libres dans l'ensemble des clauses non ordonnées.

Nous avons besoin des définitions suivantes avant de pouvoir décrire les deux fonctions objectifs que nous proposons.

Définition 7 (Littéral renommé libre) *Soient Σ une formule CNF, R un renommage de $\mathcal{V}(\Sigma)$. On dit qu'un littéral l est positif et libre dans c pour R si $estPos(l, c, R)$ et que l n'est pas lié dans c par rapport à Σ. On note cela $estPos\&Libre(l, c, R)$. Ce même littéral est négatif et libre dans c pour R si $estNeg(l, c, R)$ et que l n'est pas lié dans c par rapport à Σ. On note cela $estNeg\&Libre(l, c, R)$.*

On note le nombre de littéraux positifs (resp. négatif) et libres dans c pour R par $nbPos\&Libre(c, R)$ (resp. $nbNeg\&Libre(c, R)$).

On note $nbPos\&LibreTot(\Sigma, R)$ le nombre total de littéraux positifs et libres présents dans la partie non Ordonnée de Σ pour le renommage R.

Cette définition nous permet de définir la notion d'*ordonné-renommabilité* :

Définition 8 (ordonné-renommabilité) *Soient Σ une formule CNF et R un renommage. Une clause $c \in \Sigma$ est dite ordonné-renommée par R (notée $o\_ren(c, R)$) si $nbPos\&Libre(c, R) \leq 1$ i.e. c contient au plus un littéral positif libre pour R ; sinon elle est dite ordonné-falsifiée par R (notée $o\_fal(c, R)$).*

On note $nbOrd(\Sigma, R)$ le nombre de clauses de Σ ordonné-renommées par R.

Il ne nous reste plus qu'à définir les compteurs de gain (*Makecount*) et de perte (*Breakcount*), ainsi que les fonctions *score* associées pour chacune des deux fonctions objectifs.

Pour la minimisation du nombre de clauses non ordonnées :

Définition 9 ({Break/Make}Count) *Soient Σ une formule CNF, $x \in \mathcal{V}(\Sigma)$, R un renommage et R_x le renommage obtenu à partir de R en inversant la valeur de vérité de x. On définit $o\_breakCount(x, R) = |\{c | o\_ren(c, R), o\_fal(c, R_x)|$ et $o\_makeCount(x, R) = |\{c | o\_fal(c, R), o\_ren(c, R_x)\}|$. On définit $o\_score(x, R) = o\_makeCount(x, R) - o\_breakCount(x, R)$*

Pour la minimisation du nombre de littéraux positifs libres dans les clauses non ordonnées :

Définition 10 *Soient Σ une formule CNF, $x \in \mathcal{V}(\Sigma)$, R un renommage et R_x le renommage obtenu à partir de R en inversant la valeur de vérité de x. Soient $o\_Break_1 = \{c \in \Sigma | nbPos\&Libre(c, R) = 1$ et $estNeg\&Libre(x, c, R)\}$, $o\_Break_2 = \{c \in \Sigma | nbPos\&Libre(c, R) > 1$ et $estNeg\&Libre(x, c, R)\}$, $o\_Make_1 = \{c \in \Sigma | nbPos\&Libre(c, R) = 2$ et $estPos\&Libre(x, c, R)\}$ et $o\_Make_2 = \{c \in \Sigma | nbPos\&Libre(c, R) > 2$ et $estPos\&Libre(x, c, R)\}$. Les compteurs sont définis comme suit : $oM\_breakCount(x, R) = 2 \times |o\_Break_1| + |o\_Break_2|$ et $oM\_makeCount(x, R) = 2 \times |o\_Make_1| + |o\_Make_2|$. Puis on définit $oM\_score(x, R) = oM\_makeCount(x, R) - oM\_breakCount(x, R)$.*

Les ensembles o_Break et o_make sont analogues aux ensembles h_Break et h_make (définition 4) respectivement, pour la classe des formules ordonnées avec la contrainte supplémentaire que x doit être libre à chaque fois, car tous les littéraux liés sont ignorés.

Pour une formule CNF donnée, en remplaçant $nbHorn(\Sigma)$ par $nbOrd(\Sigma)$ et h_score par o_score dans l'algorithme 1, on obtient un algorithme pour calculer un renommage de Σ maximisant le nombre de clauses ordonné-renommées. Ce renommage sera appelé renommage *Ordonné_Min_Clause* (OMC).

En y remplaçant o_score par oM_score, on obtient un algorithme pour calculer un renommage minimisant le nombre de littéraux positifs et libres dans les clauses non ordonné-renommées. Ce renommage sera appelé renommage *Ordonné_Min_Littéraux* (OML).

4.3 Calcul d'ensembles ordonnés strong backdoor

Cette fois encore, le problème du calcul d'un ensemble ordonné strong backdoor de taille minimale est NP-difficile. Nous allons donc utiliser le même procédé que pour les ensembles Horn strong backdoor *i.e.* pour une formule CNF, calculer un sous-ensemble de taille minimale des littéraux *libres* positifs apparaissant dans la sous-formule non-ordonnée telles qu'une fois retirées les littéraux de cet ensemble, le reste de la formule est ordonné. Cependant, ceci n'est valable que si le retrait de ces littéraux, par leur affectation à vrai ou à faux, ne rend pas libres d'autres littéraux de la formule qui été liés. La proposition suivante nous assure que c'est le cas :

Proposition 3 *Soit Σ une formule CNF. Soit $\mathcal{U}$ un ensemble de clauses unitaires sur $\mathcal{V}$. Soit I une interprétation telle que $I = \mathcal{V}(\mathcal{U})$. Soit $C \in \Sigma$, $l \in C$ tel que l est lié dans C par rapport à Σ, alors l est aussi lié dans C' par rapport à $\Sigma' = I(\Sigma)$ ($C' = I(C)$), ou bien C est retirée (satisfaite) de Σ'.*

Preuve *Du fait que l est lié, deux cas sont possibles :*

- *$Occ_\Sigma(\neg l) = \emptyset$: dans ce cas, il est trivial que $Occ_{\Sigma'}(\neg l) = \emptyset$, et l est toujours lié dans Σ' dans toutes les clauses où il apparaît.*
- *$Occ_\Sigma(\neg l) \neq \emptyset$: $\exists t \in C$ tel que $Occ_\Sigma(\neg l) \subseteq Occ_\Sigma(\neg t)$. Ici trois cas sont possibles :*
  - *$t \in \mathcal{U}$: dans ce cas, la clause C est satisfaite dans Σ'.*

Instance	$\#V$	$\#C$	$\|SB_{HML}\|$	$\|SB_{OMC}\|$	$\|SB_{OML}\|$
fifo8_100	64762	176313	21065	21203	19551
f2clk_50	34678	101319	13028	13453	12582
ip50	66131	214786	29301	29720	28886
ip38	49967	162142	21894	22429	21496
w10_70	32745	103556	13739	13850	13511
ca256	4584	13236	1949	1984	1834
ca128	2282	6586	977	980	897
example2_gr_2pin_w6	3603	41023	2702	2579	2633
9symml_gr_2pin_w5	2604	32450	2666	1848	1872
too_large_gr_rcs_w9	4671	64617	3024	3385	3000
hanoi5	1931	14468	1077	1115	1053

TAB. 2 – Tailles des ensembles strong backdoor pour des instances réelles.

- $\neg t \in \mathcal{U}$ *: dans ce cas, chaque clause contenant* $\neg t$ *sont satisfaites (retirées) in* Σ'*. Du fait que* $Occ_\Sigma(\neg l) \subseteq Occ_\Sigma(\neg t)$*, toutes les clauses contenant* $\neg l$ *sont aussi satisfaites dans* Σ'*. Ainsi* $Occ_{\Sigma'}(\neg l) = \emptyset$ *et* l *est toujours liée dans* C' *par rapport à* Σ'*.*
- $t \notin \mathcal{U}$ *et* $\neg t \notin \mathcal{U}$ *: soit* S *l'ensemble des clauses de* Σ *satisfaites (retirées) dans* Σ'*. On a* $Occ_{\Sigma'}(\neg t) = Occ_{\Sigma \setminus S}(\neg t)$ *et* $Occ_{\Sigma'}(\neg l) = Occ_{\Sigma \setminus S}(\neg l)$*. Du fait que* $Occ_\Sigma(\neg l) \subseteq Occ_\Sigma(\neg t)$*, on a* $Occ_{\Sigma \setminus S}(\neg l) \subseteq Occ_{\Sigma \setminus S}(\neg t)$*. Donc* $Occ_{\Sigma'}(\neg l) \subseteq Occ_{\Sigma'}(\neg t)$*, et* l *est toujours lié dans* C' *par rapport à* Σ'*.*

Avec cette proposition, on est assuré de la stabilité de la liaison pour l'affection *i.e.* dans une formule CNF, toute variable liée dans une clause reste liée dans une clause de la formule simplifiée par n'importe quelle affectation. Ainsi, on peut calculer un ensemble ordonné strong backdoor S de taille minimale comme suit : pour une formule CNF Σ quelconque, on ne considère que la partie non-ordonnée ψ de Σ. On construit S en y ajoutant des variables dont les occurrences positives libres figurent dans $\mathcal{V}^+(\psi)$ que l'on retire de ψ itérativement jusqu'à ce que l'on obtienne une sous-formule de ψ qui soit ordonnée. À chaque étape, la variable apparaissant le plus souvent sous forme positive et libre dans ψ est choisie. On utilise l'algorithme 2 en rajoutant simplement la condition que les variables choisies soient libres.

On voit donc que par rapport à la classe des formules de Horn, les ensembles strong backdoor que l'on calcule pour les formules ordonnées sont de taille au pire égale. En effet, dans le cas où aucun littéral n'est lié, les ensembles ordonné strong backdoor et Horn strong backdoor seront identiques. Mais dès lors qu'il y a des littéraux liés, on peut les ignorer donc réduire la taille des ensembles strong backdoor.

4.4 Expérimentations

Tout comme dans le cas des formules de Horn, nous avons testé cette approche sur les instances aléatoires pour un nombre de variables allant de 100 à 400 et avons fait varier le rapport $\frac{\#C}{\#V}$. Nous obtenons exactement les même courbes que dans la figure 1 sauf pour quand le ratio est inférieur à 2 où la taille des ensembles ordonné strong backdoor est très légèrement inférieure à celle des ensembles Horn strong backdoor (environ 1% de différence). En effet, dès lors que le ratio dépasse 2, le nombre de littéraux liés devient quasinul et tend très rapidement vers 0 lorsque le ratio augmente. Dans ce cas, les formules ordonnées deviennent strictement équivalentes au formules de Horn et il est donc normal que l'on obtienne les même résultats que pour les formules de Horn en terme de taille des ensemble strong backdoor.

En ce qui concerne des problèmes réels issus des précédentes compétitions SAT, le tableau 2 montre que la généralisation du calcul d'un ensemble strong backdoor, en considérant cette fois ci la classe des formules ordonnées, nous permet d'obtenir de meilleurs résultats sur de nombreuses instances. Tout comme le tableau 1, nous donnons pour chaque instance, la taille du problème ($\#C$ et $\#V$), la taille des ensembles strong backdoor calculé en utilisant le renommage HML, celle calculée en utilisant le renommage OMC et celle utilisant OML. Comparativement à l'approche utilisant HML, on remarque que sur de nombreux problèmes, la généralisation aux formules ordonnés permet d'obtenir un meilleur ensemble strong backdoor.

Ceci met en évidence le phénomène suivant : bien que l'on ne trouve jamais d'instances ordonnées (renommables) en pratique, un certain nombre d'instances codant des applications réelles contiennent un nombre plutôt important de littéraux liés, comme en atteste la taille des ensembles ordonné strong backdoor par rapport à celle des ensemble Horn strong backdoor.

5 Expérimentations pratiques

Dans cette section, nous présentons les résultats expérimentaux que nous avons obtenus en exploitant les ensembles strong backdoor calculés précédemment lors de la résolution d'instances SAT. Dans ces expérimentions, les ensembles strong backdoor ont été utilisés pour forcer le solveur *Zchaff*[2] [11] à choisir ses variables à instancier. Une fois toutes les variables contenues dans l'ensemble strong backdoor instanciées, étant donné que *Zchaff* intègre la propagation unitaire, on peut stopper l'exploration et décider de la satisfaisabilité s'il n'y a pas de clauses falsifiée, ou sinon de déclencher un retour arrière.

Les premières expérimentation ont été réalisées sur des instances aléatoires toutes non satisfaisables, avec un ratio $\frac{\#C}{\#V} = 4.25$. Comme nous l'avons constaté au paragraphe 4.4, étant donné que le ratio est supérieur à 2, les ensembles strong backdoor pour les deux classes des formules de Horn et des formules ordonnées sont identiques. C'est pourquoi

[2]Version 2003.

#V	#C	**Zchaff**			**Zchaff+SB_{MC}**			**Zchaff+SB_{ML}**		
		Temps(s)	Noeuds	MxD	Temps(s)	Noeuds	MxD	Temps(s)	Noeuds	MxD
200	850	5,15	55610,76	28,26	4,18	47071,1	25,52	4,07	46215,26	25,88
250	1062	192,18	388970,5	34,18	133,21	328879,58	30,46	138,4	332056,52	30,62
300	1275	4499,91	2287975,5	39	3601,22	2016263,68	34,9	3302,74	1978685,62	34,7

TAB. 3 – ZChaff sur les instances aléatoires

Instance	#V	#C	S/U	**Zchaff**		**Zchaff+SB_{HMC}**		**Zchaff+SB_{HML}**		**Zchaff+SB_{OMC}**		**Zchaff+SB_{OML}**	
				Temps (s)	Noeuds	Temps (s)	Noeuds	Temps (s)	Noeuds	Temps (s)	Noeuds	Temps (s)	Noeuds
okgen-c1300-v650	650	1300	S	0	431	0	194	0	166	0	178	0	165
dp07u06	3193	8308	U	1,11	6990	1,6	6636	1,38	5532	1,29	5481	1,44	6113
dp03s03	637	1468	S	0	59	0	87	0	44	0	41	0	35
dp12s12	12065	33520	S	*time*	*out*	*time*	*out*	*time*	*out*	333,46	274302	2416,93	1291718
rand_net40-30-1	2400	7121	U	1,06	8973	1,7	8716	1,95	10464	1,68	8281	1,85	9821
rand_net40-40-10	3200	9521	U	137,57	311358	152,01	314143	151,21	297820	167,47	320576	137,8	285891
rand_net40-40-5	3200	9521	U	53,46	133375	93,78	175079	56,07	127578	78,85	158466	82,52	156776
rand_net70-25-1	3500	10361	U	113,36	228410	77,98	163133	49,6	126285	72,73	151790	164,66	267168
rand_net40-25-5	2000	5921	U	198,08	344425	200,5	329849	40,99	132043	164,02	296932	62	169776
rand_net60-30-5	3600	10681	U	1295,94	1249286	727,51	833383	1034,97	1085851	633,2	749447	1173,38	1124490
w10_70	32745	103556	U	178,49	137310	282,19	145002	427,46	167610	277	126026	511,79	201524
fifo8_100	64762	176313	U	74,12	151084	264,06	105492	258,47	108329	355,34	141962	344,99	136466
hanoi6	7086	78492	S	181,4	248892	1020,28	766762	110,5	182502	1462,21	825750	184,21	248646
hanoi5	1931	14468	S	*time*	*out*	913,19	859670	784,78	769299	626,06	757422	2915,62	1755395
ip38	49967	162142	U	3010,57	1082808	2233,59	650405	3077,5	808930	*time*	*out*	*time*	*out*
unif-c2600-v650	650	2600	S	10,76	83141	274,48	468181	0,43	7053	250,03	450142	11,48	76866
unif-c2450-v700	700	2450	S	0,02	755	0,01	302	0,03	649	0,05	895	0,05	995

TAB. 4 – ZChaff sur les instances issues des compétitions SAT précédentes

dans le tableau 3, nous ne précisons pas pour quelle classe l'ensemble strong backdoor a été calculé, mais seulement le renommage utilisé [3] : MC pour minimisation du nombre de clauses et ML pour minimisation du nombre de littéraux. Dans ce tableau, chaque ligne fournie des données moyennes sur 50 instances. On peut y trouver le temps nécessaire en seconde pour la résolution (Temps), le nombre de nœuds explorés pendant la recherche (Noeuds), ainsi que la profondeur maximum de l'arbre de recherche atteint par le solveur pendant la recherche, et ceci pour les trois méthodes expérimentées. On peut constater qu'à tout point de vue, l'exploitation des ensembles strong backdoor améliore significativement les performances de Zchaff, et il semblerait que plus la taille des problèmes augmente, plus l'approche ML se différencie en mieux de l'approche MC. On voit que les ensembles strong backdoor contiennent en moyenne 50% des variables des instances alléatoires, ce qui nous permet de d'avancer une complexité de $2^{n/2}$ dans le pire des cas. Or, comme il a été remarqué dans [1], la profondeur de l'arbre de recherche développé par un solveur sur une instance 3-SAT aléatoire au seuil est bien inférieure à $n/2$, ce qui pourrait laisser imaginer qu'il existe des ensembles strong backdoor bien inférieur en taille. Cependant, du fait des heuristiques de choix dynamiques, il se peut que 100% des variables de l'instance soient instanciées au moins une fois pendant la recherche. Notre approche nous permet de décider de la satisfaisabilité en ne s'intéressant qu'à 50% des variables.

[3] En sachant que les deux classes ont produit les même résultats

Nous avons ensuite expérimenté la résolution d'instances issues des dernières compétitions SAT en exploitant les ensembles strong backdoor. Le tableau 4 indique les résultats que nous avons obtenus sur certaines instances, en utilisant les ensembles strong backdoor calculés pour les deux classes de formules de Horn et de formules ordonnées à l'aide des deux méthodes utilisant les renommages orientés nombre de clauses ($*MC$) et nombres de littéraux ($*ML$). Enfin, la colonne S/U nous renseigne sur la satisfaisabilité éventuelle de l'instance (S pour satisfaisable et U pour non satisfaisable).

Ce que l'on peut en conclure, c'est que l'exploitation des ensemble strong backdoor en général semble améliorer soit le temps de résolution, soit la taille de l'espace de recherche (nombre de noeuds). Il semble néanmoins difficile de conclure que les ensembles d'une classe sont plus pertinents que ceux d'une autre. On constate que cela dépend beaucoup de l'instance considérée, comme le montre les instances *rand_net*.

6 Conclusions et perspectives

Nous avons présenté dans cet article une nouvelle méthode de calcul d'ensembles *Horn strong backdoor*, qui minimise le nombre de littéraux positifs contenus dans la partie non Horn. Cette méthode nous a permis de calculer des ensembles Horn strong backdoor de taille plus petite, bien que le nombre de clauses non Horn soit légèrement supérieur.

Nous avons ensuite proposé une extension de la méthode de calcul d'ensembles *Horn strong backdoor* permettant de calculer des ensembles *ordonné strong backdoor*. Cette méthode nous garantie que le nombre de

variables contenues dans les ensembles ordonné strong backdoor est dans le pire des cas égal à celui des ensemble Horn strong backdoor. Cette méthode nous a permis de réduire encore la taille des ensembles strong backdoor sur certaines instances.

Nous avons ensuite expérimenté l'exploitation des ensembles strong backdoor pour les deux classes de formules de Horn et des formules ordonnées. Cette expérimentation nous a permis de constater que sur les instances aléatoire, l'exploitation des ensembles strong backdoor permet d'améliorer de plus de 20% le temps de résolution d'instances difficiles à 300 variables insatisfaisables. Sur les instances issues de problèmes réels (tirées des précédentes compétitions SAT), nous avons constaté de grandes disparités. Mais de manière assez générale, on a pu constater que l'exploitation des ensembles strong backdoor permet de gagner au moins en nombre de nœuds parcourus, si ce n'est en terme de temps de résolution. Cependant, il est difficile de dire pour quelle classe les ensembles strong backdoor sont les plus pertinents.

La principale perspective que nous envisageons est l'intégration des ensemble strong backdoor pour la résolution du problème MAX-SAT. Bien que la classe des formules de Horn ne soit pas une classe polynomiale pour ce problème, il nous semble qu'il est possible d'utiliser les ensembles Horn strong backdoor pour réduire la complexité de sa résolution, sans pour autant la rendre polynomiale.

Références

[1] Gilles Audemard, Belaid Benhamou, and Pierre Siegel. AVAL : An enumerative method for SAT. In *Proceedings of first international conference on computational logic CL00, London*, volume 1861 of *LNCS*, pages 373–383. Springer Verlag, July 2000.

[2] Emmanuel Benoist and Jean-Jacques Hébrard. Ordered formulas. In *Les cahiers du GREYC, CNRS - UPRES-A 6072 (search report)*, number 14, Université de Caen - Basse-Normandie, 1999.

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

[4] Olivier Dubois and Gilles Dequen. A backbone-search heuristic for efficient solving of hard 3–sat formulae. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI–01)*, August 4–10 2001.

[5] É. Grégoire, B. Mazure, R. Ostrowski, and L. Saïs. Automatic extraction of functional dependencies. In *proc. of SAT*, volume 3542 of *LNCS*, pages 122–132, 2005.

[6] Philip Kilby, John Slaney, Sylvie Thiebaux, and Toby Walsh. Backbones and backdoors in satisfiability. In *AAAI'2005*, 2005.

[7] Chu Min Li. Integrating equivalency reasoning into davis-putnam procedure. In *proceedings of Conference on Artificial Intelligence AAAI*, pages 291–296, Austin, USA, 2000. AAAI Press.

[8] I. Lynce and J. P. Marques-Silva. Hidden structure in unsatisfiable random 3-sat : An empirical study. In *Proc. of ICTAI'04*, 2004.

[9] H. Van Maaren and L. Van Norden. Correlations between horn fractions, satisfiability and solver performance for fixed density random 3-cnf instances. *Annals of Mathematics and Artificial Intelligence*, 44 :157– 177, 2005.

[10] Bertrand Mazure, Lakhdar Saïs, and Éric Grégoire. Tabu search for SAT. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97)*, pages 281–285. AAAI Press, July 27–31 1997.

[11] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff : Engineering an efficient sat solver. In *Proceedings of 38th Design Automation Conference (DAC01)*, 2001.

[12] Naomi Nishimura, Prabhakar Ragde, and Stephen Szieder. Detecting backdoor sets with respect to horn and binary clauses. In *7th International Conference on theory and Application of Satisfiability Testing (SAT'04)*, 2004.

[13] Lionel Paris, Richard Ostrowski, Pierre Siegel, and Lakhdar Saïs. Computing and exploiting horn strong backdoor sets thanks to local search. In *Proceedings of the 18th International Conference on Tools with Artificial Intelligence (ICTAI'2006)*, pages 139–143, Washington DC, United States, 2006.

[14] Bart Selman and Henry A. Kautz. An empirical study of greedy local search for satisfiability testing. 1993.

[15] G.S. Tseitin. On the complexity of derivations in the propositional calculus. In H.A.O. Slesenko, editor, *Structures in Constructives Mathematics and Mathematical Logic, Part II*, pages 115–125, 1968.

[16] Ryan Williams, Carla Gomes, and Bart Selman. Backdoors to typical case complexity. In *Proccceding of Intenrational Joint Conference on Artificial Intelligence (IJCAI'2003)*, 2003.

[17] Ryan Williams, Carla Gomes, and Bart Selman. On the connections between heavy-tails, backdoors, and restarts in combinatorial search. In *International Conference on Theory and Applications of Satisfiability Testing (SAT'2003)*, 2003.