HAL
archives-ouvertes.fr

Approximation d'ensembles horn strong backdoor par recherche locale

Lionel Paris, Richard Ostrowski, Lakhdar Saïs, Pierre Siegel

▶ **To cite this version:**

Lionel Paris, Richard Ostrowski, Lakhdar Saïs, Pierre Siegel. Approximation d'ensembles horn strong backdoor par recherche locale. Deuxièmes Journées Francophones de Programmation par Contraintes (JFPC06), 2006, Nîmes - Ecole des Mines d'Alès / France, 2006. <inria-00085807>

HAL Id: inria-00085807
https://hal.inria.fr/inria-00085807

Submitted on 14 Jul 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Approximation d'ensembles horn strong backdoor par recherche locale

Lionel Paris [1] **Richard Ostrowski** [1] **Lakhdar Saïs** [2] **Pierre Siegel** [1]

[1] Équipe Inférence Contraintes et Application,
Laboratoire des Sciences de l'Information et des Systèmes (LSIS UMR 6168),
CMI, Université de Provence,
Technopôle de Château Gombert,
39, rue Joliot Curie, 13453 Marseille Cedex 13 France
[2] Centre de Recherche en Informatique de Lens (CNRS FRE 2499)
Université d'Artois , Faculté des sciences Jean Perrin
Rue Jean Souvraz SP-18, 62307 Lens Cedex France
{ Lionel.Paris, Richard.Ostrowski, Pierre.Siegel}@cmi.univ-mrs.fr, sais@cril.univ-artois.fr

Résumé

Dans ce papier, nous proposons une nouvelle approche pour calculer un strong backdoor pour des formules mises sous forme normale conjonctive (CNF). Elle est basée sur une utilisation originale d'une méthode de recherche locale qui fournit un renommage maximisant la sous-formule horn-renommable d'une CNF donnée. Plus précisément, à chaque étape, on choisit de renommer la variable qui fait le plus diminuer le nombre de clauses non-horn. S'il ne reste plus de clauses strictements positives(ou strictement négatives) ou de clauses non-horn dans la formule, notre méthode répond au problème de satisfaisabilité de la formule originale; sinon, on utilise la plus petite sous-formule qui ne soit pas de horn pour en extraire un ensemble de variables (strong backdoor) tel qu'une fois ces variables instanciées, le reste du problème appartient à une classe polynômiale. Les premiers résultats éxpérimentaux montrent que notre approche est prometteuse sur un grand nombre d'instances SAT.

1 Introduction

Le problème de satisfaisabilité de formule propositionnelle (SAT) qui consiste à décider si une formule booléenne sous forme normale conjonctive (CNF) est satisfaisable est un des problème NP-complet les plus étudiés à cause de son importance pratique et théorique. Encouragée par une forte progression de la résolution pratique du problème SAT, un grand nombre d'applications, allant de la vérification formelle à la plannification, sont encodées et résolues en utilisant ce formalisme. De tels améliorations de la résolution de SAT ont été obtenues en combinant deux approches complémentaires : la recherche locale stochastique et les techniques basées sur la recherche systématique. La plupart des solveurs complets actuels est basée sur un algorithme de recherche de type backtrack appelé procédure de Davis, Putnam, Logemann et Loveland (DPLL) [2]. Cette algorithme basique est couplé avec plusieurs méthodes de filtrage importantes comme l'apprentissage, formes étendues de propagation de contraintes booléennes, prétraitements, détection des symétries, etc. L'impact de ces différentes améliorations dépend du type d'instance à résoudre. Par exemple, l'apprentissage est bien plus efficace pour résoudre des instances codant des problèmes réels qu'il ne l'est pour les problèmes générés aléatoirement. La seconde classe d'algorithme pour tester la satisfaisabilité est donc basée sur des méthodes de recherche locale. Pour ces méthodes incomplètes, l'espace de recherche est exploré de manière non systématique. Elles génèrent aléatoirement une interprétation initiale des variables booléennes et à chaque étape, une nouvelle interprétation est obtenue en inversant (" flippant ") la valeur de vérité d'une variable choisie. Ce processus est répété jusqu'à ce qu'un modèle soit trouvé, ou qu'un nombre fixé d'opérations soit

atteint. Plusieurs variantes de ce schéma de base ont été proposées pour le problème SAT (*e.g.* [15, 9]). Ces techniques ont montré des performances impressionnantes, particulièrement sur des instances de taille importante satisfaisables (*e.g.* instances aléatoires difficile).

Un autre facteur important pour l'efficacité des solveurs SAT concerne l'exploitation de la structure des problèmes. Sur plusieurs instances, particulièrement sur celles codant des problèmes réels, l'efficacité d'un solveur SAT est fortement liée à sa capacité à exploiter la structure cachée de ces instances. Récemment, une forme de structure intéressante, appelée (strong) backdoor, a été proposée dans [18]. Calculer un tel ensemble est un sujet de recherche très étudié, à cause de sa connexion avec la complexité du problème. Un ensemble de variables forme un backdoor pour une formule donnée si il existe une affectation de ces variables telle que la formule simplifiée peut être résolue en temps polynômiale. Un tel ensemble est appelé strong backdoor si toute affectation de ces variables mène à une sous-formule de classe polynômiale. Ce genre de structure est liée à la notion de variable indépendantes [6, 3]. Rappelons que calculer le plus petit ensemble backdoor est un problème NP-difficile. En pratique, approximer (en temps polynômial) un ensemble strong backdoor de taille "raisonnable" est un challenge intéressant et important. Plusieurs travaux précédents ont abordé cette question. L'approche proposée dans [4] 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 auncune porte logique n'est identifiée, le strong backdoor coïncide avec l'ensemble des variables du problème. D'autres approches ont été proposées qui utilisent différentes techniques, telles que les algorithmes de recherche systématique adaptée [17, 7]). Récemment, dans [8] un concept avancé de fraction reverse Horn sub-optimal de formules CNF a été proposé, et 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.

Le but principal de ce papier est de concevoir une approche polynômiale qui fournit un ensemble strong backdoor de taille raisonnable. À cet effet, l'approche que nous proposons commence par considèrer une formule CNF donnée comme la conjonction de deux sous-formules différentes, où la première appartient à une classe polynômiale et la seconde est composée du reste des clauses. Un ensemble strong backdoor peut être obtenue à partir des variables de la seconde sous-formule. Cependant, un tel ensemble strong backdoor peut être très grand dans certain cas *i.e.* la plupart des variables de la seconde sous-formule apparaissent dans la première. Pour pallier à ce phénomène, notre approche tente de réduire le nombre de variables communes à ces deux sous-formules.

Nous considérons que la classe polynômiale à laquelle appartient la première sous-formule est la classe des formules de Horn (sous-formule H). Pour réduire la taille de l'ensemble strong backdoor, l'approche que nous proposons est basée sur les deux problèmes suivants:

1. trouver un renommage des variables qui maximise (resp. minimise) la taille de la partie (resp. non) horn de la formule en terme de nombre de clauses (problème appelé Maximum Renammable Horn et noté MRH).

2. calculer, à partir de la sous-formule non-horn réduite (obtenue en 1.) un ensemble strong backdoor minimal *i.e.* un ensemble minimal de variables B tel que toute affectation de B transforme la formule en formule simplifiée appartenant à la classe des formules de Horn.

Malheureuesment, dans le cas général ces deux problèmes s'avèrent NP-Difficile. En fait, calculer la sous-formule Horn-renommable maximale pour une CNF donnée est équivalente au problème MAX2SAT [12, 5] *i.e.* pour un ensemble de clauses binaires données, trouver un modèle qui satisfait le plus grand nombre de clauses. Le problème de décision du second problème est défini dans [11] par Nishimura et al et il y est démontré qu'il est NP-Complet. Tout d'abord, il appartient à la classe NP car on peut vérifier en temps polynômial que l'ensemble B est un strong backdoor. Ensuite, la NP-Complétude est démontré en utilisant une réduction polynômiale du problème Vertex-Cover au problème du calcul d'un strong backdoor. Par conséquent, trouver un ensemble strong backdoor de taille minimale est aussi NP-Difficile.

Pour contourner la difficulté de ces deux problèmes, notre approche utilise de manière originale une méthode de recherche locale pour approximer la sous-formule Horn-renommable maximale (premier problème). Puis, une approche heuristique efficace pour calculer un ensemble strong backdoor (second problème) est proposée. Cette méthode heuristique ne prends en compte pour le calcul du strong backdoor que les occurences positives des variables figurant dans la partie non-horn du problème.

Le papier est organisé comme suit. Dans la première partie nous introduisons et définissons le contexte dans lequel se situe ce papier. Puis nous présentons notre approche basée sur une recherche locale pour l'approximation de la sous-formule Horn maximale. Ensuite nous présentons une méthode heuristique pour le calcul d'un ensemble Horn strong backdoor. Enfin nous présentons les premiers résultats expérimentaux de cette approche pour le calcul d'ensemble strong backdoor sur un large panel d'instances SAT, avant de conclure et de discuter des poursuites possibles a donner à ce travail.

2 Définitions préliminaires

Soit $\mathcal{B}$ un langage Booléen (*i.e.* propositionnel) de formules formées de manière standard, en utilisant les connecteurs usuels ($\vee$, $\wedge$, $\neg$, $\Rightarrow$, $\Leftrightarrow$) 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 occurence 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 [16]. 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. 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 de variables (resp. littéraux) apparaissant 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 $\neg l \notin \mathcal{L}^-(\Sigma)$.

Une *interprétation* 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. $I[x] = faux$). Un *modèle* pour une formule est une interprétation qui satisfait la formule. En conséquent, le problème SAT consiste à trouver un modèle pour une CNF quand il existe, ou de prouver qu'il n'en existe pas.

3 Approximation de MRH par recherche locale

Notre approche dans ce papier pour approximer la sous-formule Horn Maximale (MRH) est basée sur l'exploitation de l'algorithme bien connu *WalkSat* [15]. Initialement, pour une formule CNF donnée, cet algorithme incomplet est conçu pour trouver une interprétation qui satisfait autant de clauses que possible. Il commence par générer aléatoirement une interprétation complète, puis à chaque étape une variable est choisie et sa valeur de vérité est flippée. Différentes stratégies ont été proposées pour choisir cette variable (*e.g. Random Walk Strategy*[15]).Plusieurs améliorations ont été proposées. Toutes ces stratégies essaient de choisir la "meilleure" variable, c'est à dire celle dont le fait de changer sa valeur de vérité fera décroître au maximum le nombre de clauses falsifiées du problème. Cette étape est répétée jusqu'à ce que (i) un nombre maximum (fixé à l'avance) de flips (*MAX_FLIPS*) soit atteind ou (ii) un modèle soit trouvé *i.e.* la formule est satisfaisable. Dans le premier cas, le processus est répété avec une autre interprétation générée aléatoirement jusqu'à ce qu'un nombre maximum de tentatives (fixé à l'avance) soit atteind (*MAX_TRIES*). Si aucun modèle n'est trouvé, l'algorithme fournit "no model is found" comme réponse au problème de décision de SAT.

Pour présenter notre approche, nous devons introduire quelques définitions.

Définition 1 *Soient Σ une formule CNF, $x \in Var(\Sigma)$, I une interprétation et I' l'interprétation obtenue à partir de I en inversant la valeur de vérité de x. On définit $breakCount(x, I) = |\{c|I[c] = vrai, I'[c] = faux\}|$ et $makeCount(x, I) = |\{c|I[c] = faux, I'[c] = vrai\}|$. On définit $score(x, I) = makeCount(x, I) - breakCount(x, I)$ comme le score de x pour I*

La prochaine variable à flipper est choisie dans une clause falsifiée. C'est celle qui possède le meilleur score ou n'importe laquelle choisie au hasard en fonction d'une certaine probabilité. Ensuite la variable est flippée et les compteurs *makeCount* et *breakCount* sont mis à jour.

Le *renommage* d'un ensemble de variables V peut être défini comme une application :

Définition 2 *Soit Σ une formule CNF. On définit un renommage R de $Var(\Sigma)$ comme l'application de V dans $\{vrai, faux\}$.*

On remarque qu'un renommage R peut être assimilé à une interprétation classique.

Définition 3 *Soient Σ une formule CNF, R un renommage de $Var(\Sigma)$. On définit Σ_R comme la formule obtenue en substituant, pour toutes les variables x telles que $R[x] = vrai$, toutes les occurences 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.*

Pour calculer la sous-formule de horn maximale, notre approche, basée sur la méthode de recherche locale décrite ci-dessus, diffère sur certains points. Tout d'abord, l'*interprétation* retournée par notre algorithme représente tantôt un *horn-renommage* de la formule, tantôt le meilleur *renommage* trouvé du point de vue de la taille de la sous-formule de horn. Dans le premier cas, la formule appartient à la classe des formules horn-renommables et sa satisfaisabilité peut être testée en temps linéaire. Dans le second cas, on obtient une approximation de la sous-formule de horn maximale.

Définition 4 *Soient Σ une formule CNF, I une interprétation et une clause $c \in \Sigma$. On définit le nombre de littéraux satisfaisant (resp. falsifiant) c pour I par $nbLS(c, I)$ (resp. $nbLU(c, I)$).*

Définition 5 *Soient Σ une formule CNF et I une interprétation. Une clause $c \in \Sigma$ est dite horn-satisfied par I (noté $h\_sat(c, I)$) si $nbLU(c, I) \leq 1$ i.e. c est satisfaite par au plus un littéral; sinon elle est dite horn-unsatisfied par I (noté $h\_unsat(c, I)$). On définit $nbHorn(\Sigma, I)$ comme le nombre de clauses de Σ horn-satisfied par I.*

Définition 6 *Soient Σ une formule CNF, $x \in Var(\Sigma)$, I une interprétation et I' l'interprétation obtenue à partir de I en inversant la valeur de vérité de x. On définit $h\_breakCount(x, I) = |\{c | h\_sat(c, I), h\_unsat(c, I')|$ et $h\_makeCount(x, I) = |\{c | h\_unsat(c, I), h\_sat(c, I')\}|$. On définit $h\_score(x, I) = h\_makeCount(x, I) - h\_breakCount(x, I)$*

Dans les techniques de recherche locale traditionnelles, une clause est considérée comme satisfaite sous une interprétation I donnée si *au moins une de ses variables* est vraie pour I; sinon elle est considérée comme falsifiée. S' il ne reste plus de clauses falsifiées dans la formule, ces algorithmes retournent un modèle.

Dans l'approche "Horn locale" que nous proposons, on n'a besoin de ne maintenir à jour, pour chaque clause, que le nombre de littéraux interprétés à vrai. Ainsi, les clauses falsifiées pour les techniques de recherche locale traditionnelles peuvent être identifiées aux clauses *h_unsat* dans notre approche. En conséquence, on peut obtenir de manière simple des variantes de techniques de recherche "Horn locales" (*e.g.* tabu, novelty, rnovelty...). L'algorithme 1 décrit la version horn de Walksat pour calculer la sous-formule de horn maximale (MRH).

De plus, les conditions d'arrêt de notre algorithme sont légèrement différentes de celles des algorithmes classiques. *WalkHorn* termine dans trois cas différents :

Algorithm 1 Algorithme WalkHorn

Function WalkHorn
Require: Une formule CNF Σ
Ensure: Le meilleur *renommage* trouvé pour *MRH* de Σ

```
1:  Initialiser R_max avec toutes les variables de Σ à vrai
2:  for i = 1 to MAX_TRIES do
3:    R = interprétation générée àléatoirement
4:    for j = 1 to MAX_FLIPS do
5:      if nbHorn(Σ, R) = nbCla(Σ) then
6:        return R {Σ est horm renommable}
7:      end if
8:      if (∀c ∈ Σ, nbLU(c, R) > 0) or (∀c ∈ Σ, nbLS(c, R) > 0) then
9:        return R {Σ est satisfaisable}
10:     end if
11:     {Horn Random Walk Strategy}
12:     Avec une probabilité p do
13:     Sélectionner aléatoirement une clause non horn c et un littéral l de c
14:     R = R − {l} ∪ {¬l}
15:     done
16:     Avec une probabilité 1 − p do
17:     Soit l ∈ R tel que ∀l' ∈ R avec l ≠ l', h_score(l, R) > h_score(l', R)
18:     R = R − {l} ∪ {¬l}
19:     done
20:     if nbHorn(Σ, R) > nbHorn(Σ, R_max) then
21:       R_max = R
22:     end if
23:   end for
24: end for
25: return R_max
```

1. le nombre maximum de tentatives (*MAX_TRIES*) est atteint. Dans ce cas, le meilleur renommage trouvé est retourné (ligne 25) ou,

2. un Horn-renommage est trouvé (ligne 6). La formule Σ est Horn-renommable ou,

3. un renommage R tel que pour toute clause c de Σ, $nbLU(c, R) > 0$, ou tel que pour toute clause c de Σ, $nbLS(c, R) > 0$. En conséquence, toutes les clauses de Σ_R contiennent au moins un littéral positif, ou toutes les clauses de Σ_R contiennent au moins un littéral négatif (ligne 9).

Dans le troisième cas, la formule n'est pas forcément Horn-renommable, mais un modèle existe pour Σ. Si toute les clauses de Σ_R contiennent au moins un littéral positif, alors Σ_R est satisfaisable. Étant donné que Σ et Σ_R sont équivalents pour le problème SAT, Σ

est aussi satisfaisable. Dans ce cas, l'ensemble *strong backdoor* de Σ est vide.

Exemple 1 *Considérons la formule CNF* Σ:$\{C_1 = (\neg a \vee b \vee c), C_2 = (a \vee b), C_3 = (a \vee c), C_4 = (\neg b \vee \neg c)\}$ *et l'interprétation* $I = \{a, b, c\}$.
Nous avons nbLU(C_1,I) = 1, nbLU(C_2,I) = 0, nbLU(C_3,I) = 0, nbLU(C_4,I) = 2. Les clauses C_1, C_2 *et* C_3 *sont h-satisfied par I i.e.* C_1, C_2 *et* C_3 *deviennent des clauses de Horn dans* Σ_R. *Si on flippe la valeur de la variable a, on obtient :* $I' = \{\neg a, b, c\}$. *Alors nous avons nbLU(C_1,I') = 0, nbLU(C_2,I') = 1, nbLU(C_3,I') = 1, nbLU(C_4,I') = 2 et les clauses* C_1 C_2 *et* C_3 *sont toujours de horn dans* $\Sigma_{I'}$. *Maintenant, si on flippe la valeur de la variable b on obtient :* $I'' = \{\neg a, \neg b, c\}$ *et nbLU(C_1,I'') = 1, nbLU(C_2,I'') = 2, nbLU(C_3,I'') = 1, nbLU(C_4,I'') = 1 et , les clauses* C_1 C_3 *et* C_4 *sont de Horn dans* $\Sigma_{I''}$. *Si on applique le renommage I'', on obtient :* $\Sigma_{I''} = \{C_1 = \neg a \vee b \vee \neg c, C_2 = a \vee b, C_3 = a \vee \neg c, C_4 = \neg b \vee c\}$ *où il y a au moins un littéral positif par clause, donc* $\Sigma_{I''}$ *est satisfaisable. Un modèle de* $\Sigma_{I''}$ *est obtenue en affectant toutes les variables à vrai. En conséquence, le problème original est satisafisable.*

4 Calcul d'ensembles Backdoor

Bien que le problème SAT soit NP-Complet dans le cas général, les instances codant des problèmes réels contiennent souvent des structures cachées qui peuvent être utilisées pour résoudre le problème efficacement. Dans cette section, nous décrivons comment calculer un ensemble strong backdoor en utilisant la sous-formule non-horn obtenue avec WalkHorn dans la section 3. Les ensembles backdoor ont été introduits par Williams, Gomes et Selman dans [17].

Définition 7 *Soit* Σ *une formule CNF.* $B \subseteq \mathcal{L}(\Sigma)$ *est un ensemble backdoor s'il existe une interprétation des variables de B telle que la formule simplifiée appartient à une classe polynômiale pour le problème SAT.*

Définition 8 *Soit* Σ *une formule CNF.* $B \subseteq \mathcal{L}(\Sigma)$ *est un ensemble strong backdoor si pour toute interprétation des variables de B la formule simplifiée appartient à une classe polynômiale pour le problème SAT.*

Dans la suite, nous considérons la sous-formule non-horn pour calculer un ensemble strong backdoor. Le problème qui consiste à trouver le meilleur ensemble strong backdoor revient à trouver un ensemble de littéraux positifs tel qu'une fois retirés de la sous-formule, elle devient une formule de Horn.

Exemple 2 *Considérons l'ensemble de clauses* $\{a \vee b \vee \neg c, a \vee c \vee \neg d, a \vee e \vee f \vee \neg g\}$ *et l'ensemble* $B = \{a, e\}$. *Pour toute interprétation des variables de B, soit l'ensemble de clause devient vide, soit il se transforme en un ensemble de clauses de Horn. L'ensemble B est donc un ensemble horn strong backdoor pour cette ensemble de clauses.*

Trouver le plus petit ensemble strong backdoor pour une formule Σ donnée est un problème NP-Difficile [11]. Une façon d'accomplir cette tâche consiste à considérer une formule Σ' telle que $\forall c' \in \Sigma'$, $\exists c \in \Sigma$, avec $c' \subseteq c$ et $\forall l \in c'$, l est un littéral *i.e.* Σ' est une formule monotone. Alors la formule Σ' peut être transformée en une formule de cardinalité où chaque clause obtenue à partir d'une clause c' de taille k exprime qu'au moins $k-1$ littéraux sont vrais parmis l'ensemble des littéraux de c'. De telles formules de cardinalité sont équivalentes à des formules monotones 2-SAT (toutes les clauses sont binaires). Trouver un ensemble strong backdoor minimal de Σ se ramène au problème de trouver le modèle de taille minimale de formules monotones 2-SAT.

Exemple 3 *Considérons à nouveau l'ensemble de clauses de l'exemple 2, la contrainte de cardinalité peut être représentée par l'ensemble de clauses binaires* $\{a \vee b, a \vee c, a \vee e, a \vee f, e \vee f\}$. $\{a,e\}$ *représente une interprétation minimale qui satisfait toutes les clauses. Il représente donc un strong backdoor.*

Évidemment, une telle méthode n'est pas utilisable en pratique. Nous proposons un algorithme glouton pour calculer un bon sous-ensemble de variables. Cet algorithme consiste à choisir la variable qui apparaît le plus dans la sous-formule et d'en retirer toutes ses occurences positives jusqu'à ce que la sous-formule devienne de Horn.

L'algorithme 2 montre comment calculer un ensemble strong backdoor en utilisant cette méthode gloutonne.

Algorithm 2 calcul du strong backdoor

Function Backdoor
Require: NHF : ensemble de clauses non-horn
Ensure: B : strong backdoor
1: init $B = \emptyset$
2: **repeat**
3: v=choixVariable()
4: retirer de NHF toutes les occurences positives de v
5: ajouter v à B
6: **until** NHF soit horn
7: **return** B

# V	# C	avg # B	avg % B/V	avg # NH	avg % NH/C
100	425	54,43	54,43	122,42	28,80
150	637	81,38	54,25	187,64	29,43
200	850	109,31	54,65	255,87	30,10
250	1062	136,70	54,68	322,92	30,39
300	1275	164,43	54,81	391,04	30,67
350	1487	191,81	54,80	459,66	30,90
400	1700	219,26	54,81	528,59	31,09

Table 1: Strong backdoor sur les instances 3-SAT aléatoires.

Instances	# V	# C	# B	% B/V	# NH	% NH/C	Heuristic
ewddr2-10-by-5-8	22500	123329	225	1	225	0,18	best
ewddr2-10-by-5-1	21800	118607	225	1,03	225	0,19	rnovelty+
enddr2-10-by-5-8	21000	113729	225	1,07	225	0,20	best
enddr2-10-by-5-1	20700	111567	225	1,09	225	0,20	best
e0ddr2-10-by-5-4	19500	104527	225	1,15	225	0,22	rnovelty+
e0ddr2-10-by-5-1	19500	103887	225	1,15	225	0,22	best
fvp-unsat.shuffled	24065	731850	128	0,53	74	0,01	tabu 10
bc56-sensors-k391-unsat	561371	1778987	6913	1,23	12716	0,71	rnovelty+
motors-stuck-k407-unsat	654766	2068742	39894	6,09	79973	3,87	rnovelty+
fifo8_300.shuffled-as.sat03	194762	530713	13670	7,02	18884	3,56	rnovelty+
ii8b4	1068	8214	70	6,55	195	2,37	best
ii8b3	816	6108	65	7,97	155	2,54	best
ii8b2	576	4088	70	12,15	105	2,57	best
ii8b1	336	2068	64	19,05	54	2,61	tabu 10
rope_5000-as.sat03	180000	420000	34994	19,44	27736	6,60	rnovelty+
rope_5000.shuffled	180000	420000	35214	19,56	27645	6,586	rnovelty+
bw_large.d	6325	131973	1231	19,46	14446	10,95	best
ssa7552-159	1363	3032	266	19,52	257	8,48	rnovelty+
ssa7552-160	1391	3126	273	19,63	272	8,70	rnovelty+
ssa7552-158	1363	3034	269	19,74	260	8,57	rnovelty+
dp02u01.shuffled	213	376	44	20,66	71	18,88	rnovelty+
vmpc_21.shuffled	441	45339	439	99,56	42	$< 0,01$	all
vmpc_23.shuffled	529	59685	510	96,41	46	$< 0,01$	all
vmpc_25.shuffled	625	76775	603	96,48	50	$< 0,01$	all
vmpc_27.shuffled	729	96849	706	96,84	54	$< 0,01$	all
vmpc_29.shuffled	841	120147	815	99,91	58	$< 0,01$	all

Table 2: Strong backdoor sur les instances industrielles.

5 Expérimentations

Les premières éxpérimentations ont été effectuées sur plusieurs classes d'instances 3-SAT générées aléatoirement. Chaque classe étant générée au seuil *i.e.* le rapport entre le nombre de clauses et le nombre de variables est égal à 4,25. Chaque classe d'instances est identifiée par son nombre de variables (100, 150, ... 400). Pour chaque classe, nous avons générés 400 instances. Chaque colonne du tableau 1 donne la moyenne des résultats obtenus pour une classe.

Les deux premières colonnes donnent le nombre de de variables (# V) et le nombre de clauses (# C) de la classe. (avg # B) donne la taille moyenne du strong backdoor pour les 400 instances de la classe testée. (avg % B/V) donne la moyenne du rapport entre la taille de l'ensemble strong backdoor et le nombre de variables. (avg # NH) donne le nombre moyen de clauses qui ne sont pas de Horn. Enfin, (avg % NH/C) donne la moyenne du rapport entre le nombre de clauses non-Horn et le nombre de clauses total. On remarque que, sans exception, le meilleur

MRH est trouvé en utilisant l'heuristque *rnovelty+*. En moyenne sur toutes les classes, l'ensemble strong backdoor obtenu contient 54,64% de toutes les variables, et le nombre de clauses non-Horn représente 30,2% du nombre total de clauses. En ce qui concerne ces résultats, notre méthode pour calculer MHR obtient des résultats comparables à ceux de la méthode de E. Boros dans [1] pour les instances aléatoires.

Les autres expérimentations ont été effectuées sur un ensemble de plus de 8000 instances issues des dernières compétitions SAT (industrial, crafted) [13, 14]. En moyenne sur tous ces problèmes, l'ensemble strong backdoor contient 57,67% des variables du problème, et le nombre de clauses non-Horn représente 38,44% du nombre total de clauses. On remarque que pour certains problèmes, notre méthode donne des résultats très satisfaisants. Le tableau 2 donne quelques exemples de problèmes donnant de bons résultats. Pour chaque instances, nous donnons la taille du problème (# V,# C), la taille de l'ensemble strong backdoor (# B, % B/V), la taille de la partie non-Horn (# NH, % NH/C) et le nom de l'heuristique de WalkSat ayant donné le meilleur MRH. D'un autre côté, nous avons découvert des problèmes pour lesquels cette méthode n'est pas intéressante, car la taille de l'ensemble strong backdoor est quasiment égale au nombre de variables, bien que le MRH soit de très bonne qualité. Par exemple, les instances vmpc* ont une proportion de clauses non-Horn inférieure à 0.01%, mais le rapport entre la taille de l'ensemble strong backdoor et le nombre de variables dépasse les 99,5%.

6 Conclusions et perspectives

Dans ce papier, nous avons proposé une nouvelle approche pour approximer un ensemble strong backdoor pour une formule CNF. Cette méthode utilise de manière originale une adaptation d'une méthode de recherche locale pour calculer une bonne approximation de la sous-formule de horn maximale d'une CNF donnée. Un ensemble strong backdoor est ensuite extrait des clauses de la formule non-Horn restante. Des résultats préliminaires très encourageant ont été obtenus. Sur certaines instances, la recherche locale trouve de très bonnes sous-formules Horn-renommée maximales. Ensuite, de bons ensembles strong backdoor sont extraits à l'aide d'un algorithme glouton. De plus, nous pensons que l'étude des ensembles strong backdoor peut être essentiel pour comprendre la complexité d'instances SAT difficiles. Par exemple les instances vmpc*, bien qu'elles aient de très petites sous-formules non-Horn, ont des ensembles strong backdoor de la taille des instances. Ces instances sont difficiles pour tous les solveurs connus. Les travaux présentés dans ce papier suggèrent de nombreuses perspectives intéressantes. Nous projetons d'intégrer notre approche dans un des meilleurs solveur SAT actuel (*e.g.* zchaff [10],...) en utilisant l'ensemble strong backdoor ainsi calculé pour guider la recherche lors du choix de la variable à instancier. Enfin, nous pensons que le fait de considérer les clauses non-Horn peut fournir une nouvelle fonction objective pouvant être ajouté aux techniques de recherche locale et permettrait de sortir des extrémums locaux.

References

[1] Boros. Maximum renamable horn sub-CNFs. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 96, 1999.

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

[3] Enrico Giunchiglia, Marco maratea, and Armando tacchella. Dependent and independent variables in propositional satisfiability. In *JELIA'2002*, pages 296–307, 2002.

[4] Éric. Grégoire, Bertrand. Mazure, Richard. Ostrowski, and Lakhdar. Saïs. Automatic extraction of functional dependencies. In *proc. of SAT*, volume 3542 of *LNCS*, pages 122–132, 2005.

[5] Edward A. Hirsch. A new algorithm for MAX-2-SAT. *Lecture Notes in Computer Science*, 1770:65–73, 2000.

[6] Henry Kautz, David McAllester, and Bart Selman. Exploiting the variable dependency in local search. In *International Joint Conference on Artificial Intelligence (IJCAI'1997)*, 1997.

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

[8] Hans Van Maaren and Linda 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.

[9] 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.

[10] 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.

[11] 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*, 2004.

[12] Ch. H. Papadimitriou. *Computational complexity.* Addison–Wesley, 1994.

[13] Sat 2002 : Fifth international symposium on theory and applications of satisfiability testing, May 2002. http://gauss.ececs.uc.edu/Conferences/SAT2002/.

[14] Sat 2003 : Sixth international symposium on theory and applications of satisfiability testing, May 2003. http://www.mrg.dist.unige.it/events/sat03/.

[15] Bart Selman and Henry A. Kautz. An empirical study of greedy local search for satisfiability testing. In *Proceedings of theEleventhNational Conference on Artificial Intelligence(AAAI-93)*, 1993.

[16] 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.

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

[18] 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.