Redondance dans les CNF : laissons le solveur agir

Cédric Piette

Université Lille-Nord de France, Artois
CRIL-CNRS UMR 8188
F-62307 Lens
piette@cril.fr

Résumé : La redondance d'information en logique propositionnelle est une piste de recherche très active. La complexité de certains problèmes liés à la redondance a par exemple récemment été établie pour les CNF, ainsi que pour ses fragments 2-SAT et Horn par Liberatore (2005, 2008). Toutefois, ce phénomène de redondance n'est pas considéré par les solveurs SAT modernes, et est la plupart du temps tout simplement ignoré. Gérer la redondance au sein des CNF afin d'améliorer l'efficacité des solveurs est pourtant un challenge important. Dans ce papier est étudiée une solution adaptative, permettant de discrimer l'information redondante et de ne conserver que les éléments non fondamentaux qui se montrent *utiles* pendant la recherche.

1 Introduction

SAT est le problème de décision NP-complet qui consiste à vérifier si un ensemble de clauses booléennes (appelé CNF) admet au moins une affectation logique qui le safistasse. Il s'agit d'un problème central pour la discipline informatique, qui possède de nombreuses applications dans des domaines variés tels que la bioinformatique, la vérification formelle, l'intelligence artificielle ou la conception assistée par ordinateur (CAO). Une grande communauté de chercheurs s'intéresse de près à l'étude théorique et pratique de ce problème (cf. `http://www.SATlive.org`) et de ses applications.

Les problèmes liés à la redondance au sein des CNF ont été l'objet de nombreuses études, mais celles-ci se concentrent la plupart du temps sur les aspects théoriques de ces problèmes. En effet, le problème de minimisation d'une formule propositionnelle est connu depuis la première formalisation de la hiérarchie polynomiale (Meyer & Stockmeyer (1972)). Des résultats de complexité à propos de problèmes liés à la redondance ont été établis par Liberatore (2005). D'autres résultats ont également été proposés pour des cas restreints, par exemple quand la formule est exclusivement composée de clauses de Horn (cf. Ausiello *et al.* (1986); Hammer & Kogan (1993)).

L'importance de la redondance pour la résolution pratique de SAT a ensuite été soulignée par plusieurs études empiriques. La première, conduite par Boufkhad & Roussel

(2000) sur des instances générées aléatoirement, a par exemple permis d'observer que les formules irredondantes sont en général plus difficiles à résoudre qu'avec l'ajout d'information redondante. Ce travail a été ensuite étendu par Zeng & McIlraith (2005), en montrant en particulier que la « difficulté » de résolution des CNF peut souvent être mise en lien avec la taille de ses sous-formules irredondantes.

En dépit de ces travaux qui montrent une relation forte entre redondance et coût calculatoire de la satisfaisabilité, il s'agit un problème qui est ignoré en pratique par les solveurs, car il nécessiterait tout d'abord d'extraire, ou détecter, de l'information redondante. Il s'agit malheureusement d'un problème d'une forte complexité dans le pire cas. En effet, les problèmes liés à la redondance sont situés au moins au premier niveau de la hiérarchie polynomiale (cf. Liberatore (2005)), ce qui les rend aussi difficiles que le problème SAT lui-même. Dans ce papier, une première approche pratique pour gérer la redondance de clauses pendant une recherche pour la satisfaisabilité est présentée et expérimentalement évaluée. Tout d'abord, pour circonvenir à la trop grande complexité du test de redondance, une forme d'inférence incomplète, mais linéaire en temps est utilisée. Ensuite, comme l'ajout ou le retrait d'information redondante peut, suivant les cas, améliorer ou dégrader les performances d'un solveur, une stratégie adaptative est utilisée pour manipuler l'ensemble de clauses redondantes extrait polynomialement.

Ce papier est organisé comme suit. Dans la section suivante, les notations basiques concernant la logique propositionnelle ainsi que des définitions liées à la redondance sont données. Un nouveau schéma de résolution de SAT permettant de gérer la redondance est ensuite présenté en section 3. Dans la section 4, une étude empirique montre l'intérêt de l'approche. Enfin, nous concluons par quelques perspectives de recherche.

2 Sur la redondance des CNF

Soit $\mathcal{L}$ le langage booléen standard, construit sur un ensemble fini de variables. Une formule CNF est un ensemble (interprété comme une conjonction) de clauses, une clause étant un ensemble (interprété comme une disjonction) de littéraux. Un littéral est une variable booléenne x ou sa négation $\neg x$. Deux littéraux x et $\neg x$ sont dits *complémentaires*. On note $\bar{l}$ le complémentaire d'un littéral l. Pour un ensemble de littéraux L, $\bar{L}$ est défini comme $\{\bar{l} \mid l \in L\}$. Une clause *unitaire* est une clause qui ne contient qu'un seul littéral, tandis qu'une clause *vide*, notée $\bot$, est interprétée comme $faux$. L'ensemble des variables apparaissant dans une formule CNF Σ est notée V_Σ. Une *interprétation* ρ d'une formule Σ associe une valeur $\rho(x)$ à chaque variable $x \in V_\Sigma$ et est appelée *modèle* si elle la satisfait (noté $\rho \vDash \Sigma$).

Soit Σ une formule CNF. On note $\Sigma|_x$ la formule obtenue en affectant le littéral x à la valeur $vrai$. Formellement, $\Sigma|_x = \{c \mid c \in \Sigma, \{x, \neg x\} \cap c = \emptyset\} \cup \{c \backslash \{\neg x\} \mid c \in \Sigma, \neg x \in c\}$. Cette notation est étendue aux ensembles de littéraux : étant donné $\rho = \{x_1, \ldots, x_n\}$, on définit $\Sigma|_\rho = (\ldots((\Sigma|_{x_1})|_{x_2})\ldots|_{x_n})$. Par ailleurs, on note Σ^* la formule CNF Σ close pour la propagation unitaire, définie récursivement comme suit :

1. $\Sigma^* = \Sigma$ si Σ ne contient pas de clause unitaire ;
2. $\bot \in \Sigma^*$ si Σ contient deux clauses unitaires $\{x\}$ et $\{\neg x\}$;
3. sinon, $\Sigma^* = (\Sigma|_x)^*$ où x est un littéral inclus dans une clause unitaire de Σ.

De plus, une clause c est impliquée par propagation unitaire à partir de Σ, noté $\Sigma \models^* c$, si $\bot \in (\Sigma|_{\bar{c}})^*$.

La présence d'information redondante au sein d'une CNF peut jouer un rôle important pour décider de sa satisfaisabilité. En effet, rendre explicite un grand nombre de clauses induites mais non exprimées peut aider les solveurs. Néanmoins, la présence de ce type d'information trouve ses limites dans les problèmes de stockage et de mise-à-jour. Ainsi, la présence d'un grand nombre d'information redondante ne fait que ralentir le processus de résolution. Définissons maintenant formellement une clause redondante.

Définition 1
Soit Σ une formule CNF, et $c \in \Sigma$ une clause. c est dite redondante *(par rapport à Σ) si et seulement si $\Sigma \backslash \{c\} \models \{c\}$. Toute clause non redondante est dite* irredondante.

À travers cette définition, il semble clair que toute clause redondante peut être ôtée d'une formule CNF, en préservant non seulement sa satisfaibilité mais également l'ensemble de ses modèles.

Exemple 1
Soit $\Sigma = \{\neg a \vee b \vee c, \neg b \vee c, \neg c, a \vee \neg b, c \vee d, \neg a \vee c \vee \neg d\}$ une formule CNF. La clause $\phi_1 = \neg b \vee c$ est redondante par rapport à Σ, puisque $(\Sigma \backslash \{\phi_1\})|_{\bar{\phi_1}}$ est clairement insatisfaisable. Les clauses $\phi_2 = \neg a \vee b \vee c$, $\phi_3 = a \vee \neg b$ et $\phi_4 = \neg a \vee c \vee \neg d$ sont également redondantes par rapport à Σ.

Décider si une clause est redondante est clairement CoNP-complet, et en itérant un tel test sur toutes les clauses d'une CNF et en retirant chaque clause prouvée redondante, il est possible d'obtenir une *formule irredondante*.

Définition 2
Soit Σ une CNF. Σ est appelée formule irredondante *ssi $\forall c \in \Sigma$, c est irredondante.*

Cette notion de formule irredondante est utilisée depuis longtemps, bien qu'elle ait été baptisée de différentes façons dans la littérature. Par exemple, elle est définie sous le nom de *irredundant equivalent subset* par Liberatore (2005) et *satisfiable core* par Boufkhad & Roussel (2000). De plus, de nombreuses études se sont récemment concentrées sur le problème d'expliquer *pourquoi* une CNF ne possède aucune solution, à travers le concept de MUS (*Minimally Unsatisfiable Subformula*) (cf. Grégoire *et al.* (2007)). Un MUS est en fait une formule irredondante dans le cas particulier de l'insatisfaisabilité.

Une formule CNF peut posséder plusieurs formules irredondantes ; en fait, une CNF contenant m clauses peut posséder dans le pire cas $C_m^{m/2}$ formules irredondantes.

Exemple 2
Soit Σ la formule CNF de l'exemple précédent. Σ possède 3 sous-formules irredondantes, qui sont $\Sigma_1 = \Sigma \backslash \{\phi_1, \phi_2\}$, $\Sigma_2 = \Sigma \backslash \{\phi_2, \phi_3\}$ et $\Sigma_3 = \Sigma \backslash \{\phi_3, \phi_4\}$. Ces 3 formules sont illustrées dans la figure 1.

Le rôle des clauses redondantes au sein des CNF reste flou. Boufkhad & Roussel (2000) ont montré que les formules irredondantes sont typiquement plus difficiles à résoudre qu'avec la présence de clauses redondantes. Il est en effet connu que ces clauses

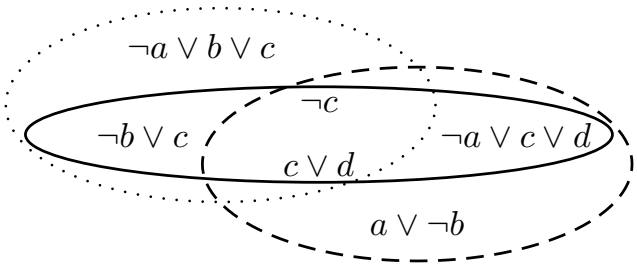

FIG. 1 – Les 3 formules irredondantes de l'exemple 2

peuvent en pratique aider les solveurs ; à titre d'exemple, les mécanismes d'apprentissage (Beame *et al.* (2003)), qui produisent une résolvante particulière après chaque conflit peuvent être vu comme un ajout dynamique de clauses redondantes pendant la recherche. Cette stratégie d'apprentissage est l'un des éléments clés de l'efficacité des solveurs modernes, ce qui montre l'intérêt de l'information redondante par rapport à la résolution pratique de SAT. Cependant, il est également connu qu'ajouter trop d'information redondante peut au contraire ralentir la recherche. D'ailleurs, une simple expérience consistant à ajouter à une CNF toutes les clauses apprises après sa résolution, montre que cette nouvelle formule est typiquement plus difficile à résoudre.

Ainsi, il paraît difficile de prédire si une clause redondante va se montrer utile pour la résolution d'une formule CNF donnée, en particulier parce que cela dépend de la façon dont l'espace de recherche est exploré. Dans la section suivante, une approche pour gérer l'information redondante est proposée.

3 Un nouveau schéma pour la gestion de la redondance

3.1 Extraire des clauses redondantes en temps polynomial

Comme présenté dans la section précédente, l'information redondante peut s'avérer utile pour la résolution de SAT, ou la ralentir considérablement. Nous proposons ici une technique originale permettant de gérer ce type d'information. Malheureusement, cela induit de pouvoir extraire des clauses redondantes au sein des CNF. S'agissant d'un problème aussi difficile que SAT lui-même, effectuer un tel test pour la résolution pratique de ce problème n'a donc aucun sens. Pourtant, une approche récente de Fourdrinoy *et al.* (2007) vise à détecter de manière incomplète des clauses redondantes en temps polynomial. L'idée générale de ce test est de n'effectuer ce test de redondance qu'à travers la propagation unitaire.

Définition 3
Soit Σ une formule CNF et $c \in \Sigma$ une clause. c est appelée U-redondante *si et seulement si $\Sigma \backslash \{c\} \models^* c$, i.e. $\bot \in (\Sigma \backslash \{c\})|_{\bar{c}})^*$.*

Données: Σ : une formule CNF
Résultat: $vrai$ si Σ est satisfaisable, $faux$ sinon

début
  $\Sigma_{ur} \leftarrow \emptyset$;
  $\Sigma_{uir} \leftarrow \Sigma$;
  pour chaque $c \in \Sigma$ **faire**
    $\Sigma_{uir} \leftarrow \Sigma_{uir} \backslash \{c\}$;
    Soit $c = \{l_1, \ldots, l_k\}$;
    $i \leftarrow 1$; *conflit* $\leftarrow$ *faux* ;
    tant que $i \leq k$ **et** *conflit* $=$ *faux* **faire**
      si $(\Sigma_{uir}|_{\{\bar{l_1},\ldots,\bar{l_i}\}})^*$ = $\bot$ **alors** $\Sigma_{ur} \leftarrow \Sigma_{ur} \cup \{l_1, \ldots, l_i\}$;
      $i \leftarrow i + 1$;
    fin
    si *conflit* $=$ *faux* **alors** $\Sigma_{uir} \leftarrow \Sigma_{uir} \cup \{c\}$;
  fin
  retourner `solve`$(\Sigma_{uir},\Sigma_{ur})$;
fin

Algorithm 1: Solveur U-redSAT

Considérer les clauses U-redondantes permet d'éviter toute explosion combinatoire tout en extrayant un grand nombre de clauses redondantes en pratique. En fait, toute formule CNF Σ peut être partitionnée en temps polynomial en $\Sigma_{uir} \cup \Sigma_{ur}$, où Σ_{uir} est une formule U-irredondante de Σ et Σ_{ur} est un ensemble de clauses redondantes par rapport à Σ. Assez clairement, l'ordre dans lequel les tests de U-redondance sont effectués peut mener à différentes partitions de Σ.

3.2 Gérer les clauses extraites de manière spécifique

L'intuition de la technique présentée ici est que l'un des mécanismes implémenté dans les solveurs modernes pourrait être en charge de la sélection des clauses redondantes pertinentes par rapport à la recherche en cours. En effet, les solveurs actuels produisent de l'information redondante après chaque conflit, à travers leur fonction d'apprentissage. Ils doivent donc faire face aux problèmes de stockage et de mise-à-jour de cette information additionnelle. Cet aspect est contrôlé via différentes techniques dont le but est d'assurer un bon compromis entre la quantité de clauses à mettre-à-jour et la capacité à propager de l'algorithme. Actuellement, la stratégie la plus largement utilisée est inspirée de l'heuristique de branchement de variables VSIDS, et vise à ne conserver que les clauses les plus actives, c'est-à-dire celles permettant de filtrer le plus l'espace de recherche (Goldberg & Novikov (2002)). Plus précisément, un compteur (initialisé à 0) appelé *activité* est associé à chaque clause apprise, et est incrémenté chaque fois que la clause correspondante est « utilisée » pour déclencher une propagation unitaire. Périodiquement, la base de clauses apprises est purgée des éléments possédant les scores les plus bas, suivant le principe que les clauses les plus utiles jusqu'à présent le seront également pendant le reste de la recherche.

Instance			Test de Redondance			Temps de résolution	
Name	#cla	S/U	#cla$_{red}$	$\%_{red}$	temps	`minisat`	`minisat`$_{red}$
hanoi5u	59 718	U	2 139	3,58%	1,21	214,87	164,15
bqwh.33.381	9 040	S	494	5,46%	0,06	456,5	38,32
5col100_15_6	3 473	U	397	11,43%	0,03	56,93	30,88
lksat-n2000-m6840-k3-l4	6 598	U	242	3,66%	0,02	28,72	29,66
Composite-024BitPrimes-0	9 689	S	793	8,18%	0,05	206,25	163,71
shuffling-1-s1025511904	42 818	U	4 818	11,3%	2,17	284,04	332,98
manol-pipe-c6id	238 142	U	3 899	1,64%	3,38	150,58	147,97
ferry12	23 285	S	7 285	31,3%	0,43	0,85	0,94
avg-checker-5-34	33 206	U	2 700	8,13%	1,09	55,09	28,77
2dlx_cc_mc_ex_bp_f2_bug015	149 693	S	37 664	25,2%	40,35	2,31	2,35
9vliw_bp_mc	156 921	U	22 542	14,4%	59,99	248,77	898,7
k2fix_gr_2pinvar_w8	270 136	U	0	0%	30,04	67,52	67,48
hanoi6	22 633	S	11 120	49,1%	0,21	132,56	87,69
shuffling-2-s1182968979	58 408	U	4 487	7,68%	5,05	405,65	348,83
frg2mul.miter	58 477	U	4 418	7,56%	2,02	540,28	305,04
CompositeRSA640	1 929 086	?	105 148	5,45%	26,04	*time out*	*time out*
lksat-n900-m6174-k4-l4-s...	6 084	U	90	1,48%	0,02	57,59	109,26
4pipe_1_ooo	60 564	U	13 956	23,04%	4,19	184,56	82,56
rand_net70-30-5	12 071	U	390	3,23%	0,33	266,33	344,21
gripper10	14 126	S	3 782	26,77%	0,21	*time out*	192,95

TAB. 1 – Taux de redondance et temps de résolution d'un échantillon de problèmes

Nous proposons donc d'utiliser la méthode de Fourdrinoy *et al.* (2007) pour capturer efficacement un ensemble de clauses redondantes au sein d'une CNF, et d'informer le solveur de la nature de ces clauses. Dans ce but, les clauses détectées sont éliminées de la CNF et placées dans la base de clauses apprises du solveur. Grâce à cette approche, les clauses redondantes les plus utiles (i.e. avec la plus haute *activité*) seront conservées, tandis que les clauses non pertinentes seront progressivement supprimées.

L'approche proposée est décrite dans l'algorithme 1. Tout d'abord, les clauses de Σ sont testées pour la U-redondance. En pratique, les opposés des littéraux de la clause courante c sont affectés un par un (littéral l_i avec $1 \leq i \leq k$), et dès qu'un conflit survient, la clause composée des littéraux l_1 à l_i est enregistrée dans la partie redondante Σ_{ur} (lignes 7 à 11). Si au contraire c n'est pas U-redondante, alors elle est replacée dans la formule courante Σ_{uir} (ligne 12).

À la fin de cette étape de prétraitement, deux CNF Σ_{uir} et Σ_{ur} sont obtenues. Supposons maintenant qu'un solveur SAT appelé `solve` soit modifié de sorte que son premier paramètre soit le problème donné en entrée sous forme CNF, et le second un ensemble de clauses apprises « initiales » : un appel classique à un tel solveur serait donc « `solve`(Σ,$\emptyset$) ». Au lieu de cet appel, après l'étape préliminaire le solveur modifié est invoqué avec les formules Σ_{uir} et Σ_{ur} (ligne 14).

Ce schema de résolution de SAT, du test de redondance à l'utilisation particulière des clauses U-redondantes, peut être très facilement greffé à la plupart des solveurs actuels. Dans la section 4, les idées proposées sont évaluées d'un point de vue empirique.

4 Résultats expérimentaux

Dans le but de valider la viabilité pratique des idées présentées, nous avons implémenté le test de U-redondance pour calculer (de manière incomplète) un ensemble de clauses redondantes au sein d'une CNF. Comme l'ordre dans lequel les tests de U-redondance sont effectués a un impact sur la sous-formule U-irredondante obtenue, nous avons choisi de trier les clauses dans l'ordre décroissant de leur taille, afin d'obtenir une sous-formule close pour la subsumption. La procédure retourne donc une partition clausale, la première partie étant une sous-formule U-irredondante et la seconde une base de clauses redondantes.

Nous avons ensuite modifié une méthode complète dans le but de tenir compte de cette information en considérant les clauses redondantes comme des clauses apprises initiales. Nous avons ensuite comparé le comportement de notre solveur modifié avec la version originale, qui considère toutes les clauses de la CNF comme fondamentales (aucune ne peut être supprimée). Comme cas d'étude, `minisat` (Eén & Sorensson (2008)) a été choisi comme implémentation de solveur moderne. Précisons en outre que nous avons volontairement conservé tel quel les paramètres de `minisat` dans le but d'obtenir une comparaison juste entre les 2 solveurs. En particulier, le nombre de clauses apprises conservées est initialement $|\Sigma|/3$ et est incrémenté régulièrement. Ceci signifie qu'à tout moment de la recherche, les 2 solveurs possèdent la même borne sur le nombre de clauses apprises, ce qui contribue à les exécuter dans les même conditions. Néanmoins, notre version commence son calcul avec un ensemble de clauses apprises non vide, et quand cet ensemble dépasse le tiers du nombre total de clauses, certaines sont supprimées dès qu'un conflit survient. Fort heureusement, comme le montrent nos résultats expérimentaux, ce cas ne se produit que de très rares fois.

Nos expérimentations ont été conduites sur une sélection de 940 problèmes réels (académiques et industriels) venant des dernières Compétitions SAT (2008). Tout d'abord, un échantillon des résultats obtenus est proposé dans la table 1, qui est divisé en trois parties. La première partie fournit des informations sur l'instance testée (nom, nombre de clauses et satisfaisabilité) La deuxième partie de la table se concentre sur le test de U-redondance, en reportant le nombre de clauses redondantes, le pourcentage de clauses qu'elles représentent par rapport à l'instance originale, et le temps en secondes nécessaire à cette détection. Enfin, dans la troisième partie de la table présente le temps (en secondes) requis pour résoudre la formule CNF avec le solveur original et modifié (noté $\texttt{minisat}_{red}$). Nos études expérimentales ont été conduites sur des processeurs Intel Xeon 3GHz sous Linux CentOS 4.1. (noyau 2.6.9) avec une limite mémoire de 2Go. Pour toutes ces expérimentations, une limite de temps de 900 secondes a été respectée. Si un calcul dépasse cette limite, alors la mention *time out* est reportée.

Tout d'abord, concentrons-nous sur le test de U-redondance. Les résultat obtenus montrent qu'il existe des modélisations générant un très grand nombre de clauses redondantes. À titre d'exemple, les benchmarks `2dlx...bug015` et `ferry12` possèdent respectivement au moins 25,2% et 31,3% de leurs clauses qui sont redondantes. Limiter ce test à la propagation unitaire permet donc de capturer un nombre important de clauses redondantes grâce à cet algorithme calculatoirement léger : la plupart du temps, ce test peut être effectué en moins de 5 secondes, bien que pour les très grandes instances,

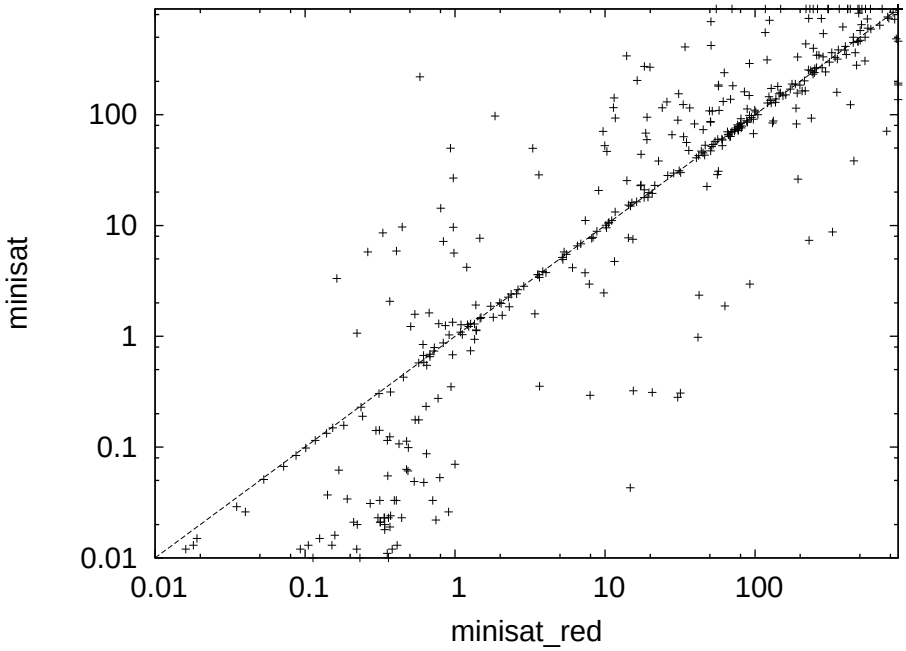

FIG. 2 – Minisat tenant (ou non) compte des clauses redondantes détectées

plusieurs dizaines de secondes peuvent s'avèrer nécessaires (`CompositeRSA640`). De tous les benchmarks testés, la formule CNF le plus redondante est l'encodage clausal propositionnel du problème des tours de Hanoi de taille 6. Ce problème possède pas moins de 22633 clauses, mais près de la moitié d'entre elles peut être inférée par l'autre moitié, et ne sont pas nécessaires pour coder le problème. Heureusement, pour la plupart des instances testées durant nos expérimentations, le « taux de redondance » n'excède pas 15%. En fait, dans certains cas, la procédure n'a été capable de détecter aucune clause redondante (`k2fix_gr_2pinvar_w8`).

Considérons maintenant les conséquences de cette information obtenue polynomialement. Ces quelques résultats montrent que discriminer les clauses U-redondantes accélère souvent la résolution d'instances. En outre, il a été observé que certaines clauses sont parfois conservées un long moment pendant la recherche ou peuvent éliminées assez rapidement, quand des clauses plus *utiles* sont apprises. Notons que même quand la CNF possède un taux de redondance relativement faible (< 15%), notre approche améliore souvent le comportement du solveur (`Composite-024BitPrimes-0, avg-checker-5-34`). Bien sûr, prendre en considération la redondance des clauses peut dans certains cas se révèler moins efficace qu'ignorer leur état. Par exemple, en dépit de la découverte de 14,3% de clauses U-redondantes, la version originale du solveur `minisat` prouve l'insatisfaisabilité de la formule `9vliw_bp_mc` en près de 4 minutes, alors que considérer ces clauses comme optionelles pour le même solveur le mène à un calcul de presque 15 minutes.

De manière plus générale, nous avons comparé le temps nécessaire à ces versions du

même solveur sur l'ensemble des tests réalisés. Les résultats obtenus sont fournis dans la figure 2 sous la forme d'un nuage de points. Chaque point de cette figure représente le temps de résolution d'un problème : le temps de $\texttt{minisat}_{red}$ est donné par la projection de ce point sur l'axe des abscisses, tandis que le temps de la version originale est donnée par sa projection sur l'axe des ordonnées. De cette manière, un point situé au dessus (dessous) de la première diagonale signifie que la version modifiée a nécessité moins (resp. plus) de temps pour résoudre le problème que la version originale. Notons en outre que les deux axes sont reportés à une échelle logarithmique.

Premièrement, on peut remarquer que pour les problèmes les plus faciles à résoudre (temps de résolution inférieur à 1 secondes), le solveur « classique » est en général la version la plus efficace. Ce phénomène est expliqué par le surcoût dû au test de redondance préliminaire. Celui-ci n'est en général pas très lourd, mais avec des problèmes si faciles à résoudre, ce temps passé a son importance. De plus, comme mentionné plus haut, pour de très grandes CNF, ce test peut nécessiter plusieurs secondes. Quant un tel benchmark est en réalité très simple pour une implémentation DPLL, le calcul inutile des clauses U-redondantes détériore les résultats. Heureusement, cette perte de temps ne représente dans les pires cas observés que quelques dizaines de secondes. Considérons maintenant les problèmes plus difficiles (temps de résolution supérieur à 10 secondes). Sur de tels problèmes, choisir parmi les clauses prouvées redondantes celles qui déclenchent le plus de propagations se montre très souvent viable. En effet, la plupart des points sont situés au dessus de la diagonale, ce qui montre l'amélioration de `minisat` par notre nouvelle approche. Enfin, notons que de nombreux points sont situés autour de la diagonale. Ceci est expliqué par la fait qu'il existe certaines CNF qui ne possèdent qu'un faible taux de U-redondance, voire aucune clause de ce type. Évidemment, dans de tels cas le comportement des deux solveurs est très similaire, voire identique dans le cas d'échec de l'étape préliminaire.

Néanmoins, de manière générale, ces premiers résultats plaident pour plus d'attention à propos de la redondance au sein des CNF pour la résolution pratique de SAT. Notre première implémentation fournit clairement des résultats prometteurs, et utiliser les mécanismes existants créés pour l'apprentissage semble adéquat pour la gestion de clauses redondantes. Assez clairement, de meilleurs résultats peuvent être attendus en réglant précisément les différents paramètres du solveur, et tout spécialement le nombre initial de clauses apprises.

5 Conclusion

Dans ce papier, une nouvelle stratégie pour la gestion de clauses redondantes au sein des CNF est présentée. Cette technique a l'avantage d'être facilement implémentable dans la plupart des solveurs actuels, grâce à une utilisation originale de leurs caractéristiques. Plus précisément, l'idée est d'extraire polynomialement un ensemble de clauses redondantes. La stratégie de type VSIDS des solveurs modernes est ensuite appliquée à cet ensemble de clauses, et si certaines d'entre elles ne permet pas de propager efficacement pendant la recherche, le solveur est alors libre de les supprimer. Cette technique est empiriquement validée par des expérimentations intensives qui montrent son intérêt pratique.

Ce travail ouvre de nombreuses pistes de recherche très intéressantes. D'abord, comme mentionné dans ce papier, l'ordre dans lequel les clauses sont testées pour la U-redondance peut être d'une grande importance. Le choix que nous avons fait, basé sur la subsumption, s'est montré pertinent, mais de nouveaux doivent être testés. De plus, les solveurs SAT proposent souvent un grand nombre de paramètres qui sont cruciaux pour l'efficacité de la procédure. Il a été choisi de ne pas modifier ces paramètres dans notre version test, mais certains d'entre eux, comme le nombre de clauses apprises autorisées à être conservées, pourraient être redéfinis pour tenir compte de la quantité d'information initialement fournie. Enfin, cette étude se concentre sur la façon de gérer l'information redondante déjà présente au sein des CNF. Il serait également intéressant de produire des clauses redondantes en utilisant des formes de résolution limitée, comme certains préprocesseurs tels que `HyPre` de Bacchus & Winter (2003) le font. Ces clauses produites pourraient être considérées comme apprises, dans le but d'aider le solveur quand elles se montrent effectivement utiles. Si certaines d'entre elles ne sont au contraire pas pertinentes pendant la recherche, le solveur serait alors capable de s'en débarasser. Nous prévoyons d'explorer ces différentes pistes de recherche.

Références

AUSIELLO G., D'ATRI A. & SACCÁ D. (1986). Minimal representation of directed hypergraphs. *SIAM Journal on Computing*, **15**(2), 418–431.

BACCHUS F. & WINTER J. (2003). Effective preprocessing with hyper-resolution and equality reduction. In *SAT'03*, volume 2919, p. 341–355.

BEAME P., KAUTZ H. & SABHARWAL A. (2003). Understanding the power of clause learning. In *IJCAI'03*, p. 1194–1201.

BOUFKHAD Y. & ROUSSEL O. (2000). Redundancy in random SAT formulas. In *AAAI'00*, p. 273–278.

Compétitions SAT (2008). http ://www.satcompetition.org.

EÉN N. & SORENSSON N. (2008). Minisat home page http ://www.cs.chalmers.se/cs/research/formalmethods/minisat.

FOURDRINOY O., GRÉGOIRE E., MAZURE B. & SAIS L. (2007). Eliminating redundant clauses in SAT instances. In *CP-AI-OR'07*, p. 71–83.

GOLDBERG E. P. & NOVIKOV Y. (2002). Berkmin : a fast and robust SAT-solver. In *DATE'02*, p. 142–149.

GRÉGOIRE E., MAZURE B. & PIETTE C. (2007). Local-search extraction of MUSes. *Constraints Journal*, **12**(3), 325–344.

HAMMER P. L. & KOGAN A. (1993). Optimal compression of propositional horn knowledge bases : Complexity and approximation. *Artificial Intelligence*, **64**(1), 131–145.

LIBERATORE P. (2005). Redundancy in logic I : CNF propositional formulae. *Artificial Intelligence*, **163**(2), 203–232.

LIBERATORE P. (2008). Redundancy in logic II : 2CNF and horn propositional formulae. *Artificial Intelligence*, **172**(2–3), 265–299.

MEYER A. R. & STOCKMEYER L. J. (1972). The equivalence problem for regular expressions with squaring requires exponential space. In *FOCS'72*, p. 125–129.

ZENG H. & MCILRAITH S. A. (2005). The role of redundant clauses in solving satisfiability problems. In *CP'05*, p. 873.