Une recherche locale dirigée par l'analyse de conflits pour la satisfiabilité

Djamal Habet, Donia Toumi

▶ **To cite this version:**

Djamal Habet, Donia Toumi. Une recherche locale dirigée par l'analyse de conflits pour la satisfiabilité. JFPC 2012, May 2012, Toulouse, France. hal-00818035

HAL Id: hal-00818035
https://inria.hal.science/hal-00818035

Submitted on 25 Apr 2013

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.

Une recherche locale dirigée par l'analyse de conflits pour la satisfiabilité *

Djamal Habet **Donia Toumi**
LSIS – CNRS UMR 7296
Université Aix-Marseille
{Djamal.Habet, Donia.Toumi}@lsis.org

Résumé

Cet article présente un algorithme de recherche locale guidée par l'analyse de conflits pour résoudre le problème SAT. L'usage d'une telle analyse, permettrait d'exploiter les dépendances entre les variables particulièrement présentes dans des instances structurées et d'accroître l'effet de la propagation unitaire. Les premiers résultats expérimentaux sont prometteurs.

1 Introduction

Le problème de satisfiabilité (SAT) consiste à décider si une formule booléenne sous la forme normale conjonctive est satisfiable. SAT est l'un des problèmes NP-complets les plus étudiés à cause de son importance aussi bien sur le plan théorique que pratique.

L'intérêt des mécanismes de l'analyse des conflits et de l'apprentissage des clauses dans la résolution du problème SAT a été notamment démontré dans les démonstrateurs complets (par exemple dans [3, 7]). Cependant, peu de travaux exploitent de telles techniques dans le cadre de la recherche locale [1, 4]. Par ailleurs, des instances SAT, en particulier structurées, peuvent contenir des dépendances entre les variables dont la prise en compte permet d'améliorer leur résolution. Nous présentons dans ce papier une nouvelle intégration de l'analyse des échecs dans une recherche locale qui permet d'inclure la propagation unitaire sur une interprétation complète en cours de réparation captant ainsi d'éventuelles relations entre variables. Cette intégration introduit aussi un caractère tabou dans la recherche prévenant ainsi l'occurrence de cycles lors de l'exploration de l'espace de recherche. Les premiers résultats expérimentaux nous semblent prometteurs.

*Avec le soutien du projet ANR-08-BLAN-0289-04

Ce papier est organisé comme suit : dans la section 2 après la donnée de quelques définitions et notations, nous effectuons un rappel sur la recherche locale pour SAT, par la présentation de Walksat [8] qui est utilisé comme base dans notre approche, et un bref rappel du principe de l'analyse de conflits. La section 3 est consacrée à la description de notre approche, en décrivant les éléments qui la composent puis en expliquant son principe de fonctionnement. Les sections 4 et 5 présentent des travaux proches et des premiers résultats expérimentaux, avant de conclure dans la section 6.

2 Préliminaires

2.1 Définitions et notations

Une instance $\mathcal{F}$ du problème de satisfiabilité (SAT) est définie par la paire $\mathcal{F} = (\mathcal{X}, \mathcal{C})$, tels que $\mathcal{X} = \{x_1, x_2, \cdots x_n\}$ est un ensemble de variables booléennes (leurs valeurs appartiennent à l'ensemble $\{vrai,\ faux\}$) et $\mathcal{C} = \{c_1, c_2, \cdots c_m\}$ est un ensemble de clauses. Une clause $c_i \in \mathcal{C}$ est une disjonction finie de littéraux et un littéral est soit une variable (x_i) ou sa négation ($\neg x_i$). Une clause est aussi représentée par l'ensemble des littéraux y apparaissant. Pour un littéral donné l, $var(l) = \{x_i |\ l = x_i\ ou\ l = \neg x_i\}$ correspond à l'ensemble singleton de la variable sur laquelle porte l. De plus, pour une clause donnée c_j, $var(c_j) = \cup_{l\ \in\ c_j} var(l)$ est l'ensemble des variables apparaissant dans c. La clause vide est toujours fausse et notée par $\square$. Une interprétation $\mathcal{I}$ des variables de $\mathcal{F}$ est définie par un ensemble de littéraux, tel que $\forall \{l_1,\ l_2\} \subseteq \mathcal{I},\ var(l_1) \neq var(l_2)$. Une variable qui apparaît positivement (resp. négativement) dans $\mathcal{I}$ signifie qu'elle est fixée à $vrai$ (resp. à $faux$). $\mathcal{I}$ est dite partielle si $|\mathcal{I}| < n$ et complète sinon. Un modèle de $\mathcal{F}$

est une interprétation complète qui satisfait toutes les clauses de $\mathcal{F}$. Enfin, le problème de satisfiabilité (SAT) consiste à tester si $\mathcal{F}$ possède un modèle. Si c'est le cas alors $\mathcal{F}$ est dite satisfiable, sinon $\mathcal{F}$ est insatisfiable.

2.2 La recherche locale pour SAT

La description qui suit se limite à un rappel de l'algorithme Walksat [8] utilisé comme base dans cet article. Partant d'une configuration complète $\mathcal{I}_c$ falsifiant certaines clauses d'une instance $\mathcal{F}$ donnée, Walksat tente de minimiser le nombre de clauses falsifiées dans l'espoir d'atteindre un modèle. En effet, à chaque étape, Walksat tente de réparer $\mathcal{I}_c$ en inversant la valeur d'une variable (flip) apparaissant dans une clause falsifiée c_i, choisie aléatoirement. Pour toute variable $x_j \in var(c_i)$, une fonction *score* calcule le nombre de clauses qui seront falsifiées en cas de flip de x_j. Ces scores servent à choisir la variable à flipper. Ainsi, s'il existe une variable à score nul alors une telle variable est choisie (descente). Sinon, la recherche se trouve dans un minimum local. Dans ce cas, avec une probabilité p (*noise probability*), la variable avec le plus petit score est sélectionnée et avec une probabilité $1-p$ une variable est sélectionnée aléatoirement dans c_i (*marche aléatoire*). Etant incomplet, Walksat ne garantit pas la construction d'un modèle pour les instances satisfiables et ne peut prouver l'inconsistance des instances insatisfiables.

2.3 Analyse de conflits et graphe d'implications

L'analyse de conflits dans le cadre de la satisfiabilité a été introduite dans [7]. Elle est exploitée dans une recherche complète et arborescente, travaillant donc sur des interprétations partielles. Dans cette dernière, certaines variables sont fixées par décision et d'autres sont le résultat de la propagation des clauses unitaires induites par l'interprétation successive des variables. L'analyse des conflits est conduite lors de l'occurrence d'un échec correspondant à la falsification d'une clause. Dans ce cas, cette analyse identifie les littéraux (les faits) responsables de cet échec (conflit). Pour éviter que la recherche visite à nouveau la partie de l'espace de recherche contenant ce conflit, un *nogood* est ajouté sous forme d'une clause, dite *assertive*, aux clauses initiales de l'instance traitée. De plus, cette clause assertive permet d'augmenter l'effet de la propagation unitaire.

L'analyse de conflits nécessite la définition d'un graphe d'implications sur une interprétation partielle $\mathcal{I}_p$. Ce graphe est acyclique où les sommets sont des littéraux de $\mathcal{I}_p$ et il existe un arc d'un sommet l_i vers un sommet l_j si l'affectation de $var(l_i)$ implique celle de $var(l_j)$ par propagation unitaire. Ainsi, les variables de décision ne possèdent pas d'arcs entrants. Aussi, tout sommet (littéral) est étiqueté par la clause (devenue) unitaire ayant provoquée sa propagation et à tout littéral est associé un niveau de décision.

L'analyse de conflits exploite ce graphe d'implications (le plus souvent) par le mécanisme *Unique Implication Point* (UIP) [9]. Le nogood (clause assertive) responsable de l'échec est généré en effectuant des résolvantes entre les clauses participant à la génération du conflit. Au même temps, un niveau de retour-arrière non-chronologique est défini.

3 Notre contribution : AcWalk

3.1 Motivations

L'une des faiblesses de la recherche locale est son manque d'exploitation des dépendances entre les variables. Il est reconnu qu'une telle exploitation permet d'augmenter l'efficacité des démonstrateurs SAT, qu'ils soient complets ou à base d'une recherche locale. La prise en compte de ces dépendances passe notamment par la propagation unitaire, technique qui est difficile à mettre en œuvre dans une recherche locale qui, comme vu précédemment, agit sur des configurations complètes. Un autre problème "classique" de la recherche locale est sa tendance, dans certains cas, à se retrouver piéger dans une zone de l'espace de recherche sans pouvoir la quitter. L'une des solutions à ce problème est d'appliquer la métaheuristique tabou [5]. Nous verrons dans la suite la mise en place de cette dernière dans notre approche. Par ailleurs, l'intérêt de l'analyse de conflits et l'apprentissage des clauses (nogoods) n'est plus à démontrer. Ces techniques sont des composantes indispensables dans les démonstrateurs SAT actuels. Toutefois, on ne trouve dans la littérature que très peu de travaux sur l'intégration de ces mécanismes de l'analyse des conflits dans une recherche locale (par exemple [1, 4]).

La section suivante décrit le principe de notre algorithme, que nous appelons AcWalk, qui tente de répondre aux points cités ci-dessus.

3.2 Composantes d'AcWalk

Comme exposé précédemment, l'une des difficultés liées à la recherche locale est l'incapacité de celle-ci à intégrer la propagation unitaire lors de la recherche. Par ailleurs, l'analyse de conflits nécessite la construction d'un graphe d'implication, sur la base des propagations unitaires, à partir d'une interprétation partielle ce qui est incompatible avec l'usage d'interprétations complètes pour une recherche locale. D'où l'idée de maintenir à la fois une interprétation partielle $\mathcal{I}_p$

(servant pour la construction du graphe d'implications) et une interprétation complète $\mathcal{I}_c$ (servant pour la recherche locale). Tout le long de la recherche, la relation $\mathcal{I}_p \subseteq \mathcal{I}_c$ sera maintenue comme expliqué ci-après.

3.2.1 Construction de $\mathcal{I}_p$ et $\mathcal{I}_c$

Dans ce papier, l'algorithme de recherche locale que nous utilisons est Walksat qui effectue à chaque itération un flip sur une variable x_i selon la fonction score rappelée dans la section 2.2. Deux cas se distinguent, soit $score(x_i) = 0$ donc on est dans le cas d'une descente et dans le cas contraire ($score(x_i) > 0$) un minimum local est atteint.

Définition 1. (Clause conflit) Soit $\mathcal{F}$ une instance SAT et une interprétation complète $\mathcal{I}_c$ sur les variables de $\mathcal{F}$ et soit c_k une clause falsifiée par $\mathcal{I}_c$. c_k est dite clause conflit si $\forall x_j \in var(c_k),\ score(x_j) > 0$.

Autrement dit, une clause conflit est une clause falsifiée où toutes les variables ont des scores négatifs. Nous définissons ainsi une variable conflit comme suit :

Définition 2. (Variable conflit) Soit $\mathcal{F}$ une instance SAT et une interprétation complète $\mathcal{I}_c$ sur les variables de $\mathcal{F}$ et soit c_k une clause falsifiée par $\mathcal{I}_c$. Si c_k est une clause conflit alors toute variable de $var(c_k)$ est dite variable conflit.

Lors d'un flip par Walksat d'une variable conflit x_i dans $\mathcal{I}_c$, le littéral correspondant à ce flip est ajouté à $\mathcal{I}_p$ ainsi que les propagations unitaires obtenues par l'application de $\mathcal{I}_p$ à $\mathcal{F}$. Le résultat de ces propagations est aussi appliquée à $\mathcal{I}_c$ permettant ainsi à la recherche locale de tirer profit de cette opération et de capturer certaines dépendances hypothétiques entre les variables. Avec une telle construction, à chaque flip d'une variable conflit, d'autres variables peuvent être aussi flippées. Aussi, on a bien $\mathcal{I}_p \subseteq \mathcal{I}_c$.

Par opposition à une variable conflit, notons que si une variable x_j apparait dans une clause falsifiée tel que $score(x_j) = 0$, le flip de cette variable ne produira aucun conflit si elle venait à être ajoutée à $\mathcal{I}_p$ et ne provoquera ainsi aucune analyse de conflits.

3.2.2 Construction et exploitation du graphe d'implications

Le graphe d'implications est construit sur la base des variables conflits. Ainsi, à chaque flip d'une telle variable, le littéral correspondant est ajouté à $\mathcal{I}_p$ et le graphe d'implication est mis à jour, en l'augmentant par les sommets et les arcs correspondant à ce flip ainsi que les littéraux propagés et leurs causes.

Intéressons nous maintenant à son exploitation : durant l'extension du graphe d'implication, un conflit peut se produire sur la base de l'interprétation $\mathcal{I}_p$. Dans ce cas, une analyse de conflits est effectuée comme décrit dans la section 2.3, où le niveau de décision de chaque littéral dans $\mathcal{I}_p$ correspondant à l'itération de la recherche locale ayant provoqué son ajout à $\mathcal{I}_p$. Plus précisément, lors d'un conflit un nogood (clause assertive) est ajoutée à la base des clauses et un niveau de retour-arrière it est fourni. Il s'en suit la suppression de $\mathcal{I}_p$ de tout les littéraux ajoutés depuis l'itération it. Après cette opération, l'inclusion de $\mathcal{I}_p$ dans $\mathcal{I}_c$ est toujours vérifiée.

Notons que dans certains cas, la clause assertive peut-être vide ce qui conduit à prouver l'insatisfiabilité de l'instance SAT traitée. Cela confère à cette recherche locale une capacité qu'elle ne possédait pas de fait de son caractère incomplet.

3.2.3 Gestion tabou

Lors d'une nouvelle réparation par Walksat, ce dernier se restreint à choisir une variable dans une clause falsifiée, dont aucun littéral correspondant n'appartient à $\mathcal{I}_p$. Ce qui revient à utiliser $\mathcal{I}_p$ comme une liste tabou et la durée tabou de chaque variable (littéral) est dynamique : tant que ce littéral n'est pas effacé de $\mathcal{I}_p$ à la suite d'une analyse de conflits, la variable correspondante reste tabou. La volonté de marquer ce caractère tabou vient du fait que les valeurs des variables dans $\mathcal{I}_p$ refléteraient une certaine dépendance entre les variables qu'il conviendrait de maintenir. Un procédé comparable a été utilisé aussi dans [2, 6]. Toutefois dans notre approche, on applique un critère d'aspiration qui permet de lever le caractère tabou d'une variable x_i si le flip de celle-ci permet de réaliser une descente, autrement dit $score(x_i) = 0$. A la même occasion, on lève le caractère tabou de toutes les variables flippées ou propagées depuis le flip de x_i en supprimant de $\mathcal{I}_p$ les littéraux correspondant (en conséquence $\mathcal{I}_p \subseteq \mathcal{I}_c$ reste vérifiée) et en mettant à jour le graphe d'implications.

3.2.4 AcWalk résumé

L'algorithme AcWalk résultant est un algorithme de recherche locale basé sur Walksat. Il a pour objectif de tirer profit de la propagation unitaire et de l'analyse de conflits, et peut même prouver dans certains cas l'insatisfiabilité de l'instance. Cette capacité reste toutefois secondaire dans la pratique. AcWalk maintient tout le long de la recherche deux interprétations : une partielle ($\mathcal{I}_p$ initialisée à $\emptyset$) et une autre complète ($\mathcal{I}_c$ initialisée

aléatoirementt). Tant qu'un modèle n'est pas trouvé dans le cas de la satisfiabilité (ou que la clause assertive n'est pas vide, dans le cas d'une instance insatisfiable), une variable à flipper est choisie selon les critères évoqués et un graphe d'implications est maintenue à jour tout en appliquant l'effet des propagations sur les variables de l'interprétation complète. La configuration partielle sert aussi de liste tabou qui est toutefois remise en cause dans le cas d'une descente. A chaque essai, AcWalk effectue un nombre fixé de réparations sur une configuration initiale générée aléatoirement et ce processus est répété selon un critère d'arrêt prédéfini (limitation de temps, nombre maximum d'essai autorisés ...).

4 Travaux connexes

Cette courte étude bibliographique se limite à l'intégration de l'analyse des conflits dans une recherche locale. Dans [1], les auteurs proposent un algorithme de recherche locale CDLS (aussi basé sur Walksat) utilisant l'analyse de conflits que dans le but de s'échapper des minima locaux. Dans ce cas, une interprétation partielle est dérivée à partir de l'interprétation complète courante. Elle sert à la construction d'un graphe de conflits qui est analysé par la suite. Ainsi, tandis qu'AcWalk maintient un graphe d'implications et une interprétation partielle (et complète) tout le long de la recherche, CDLS reconstruit un graphe conflit à chaque minimum local. Dans [2], une méthode SatHys est définie par l'hybridation d'une recherche locale et un démonstrateur complet de type minisat [3]. Ces deux recherches travaillent alternativement selon des critères bien définis et partagent certaines informations. Cette hybridation constitue une première différence majeure avec AcWalk qui est un algorithme de recherche locale. Une autre distinction est la gestion de la liste tabou dans Sathys, qui n'introduit pas un mécanisme d'aspiration.

5 Résultats expérimentaux

Une étude expérimentale préliminaire a été effectuée sur une première implémentation d'AcWalk. Les tests sont effectués sur des instances structurées et satisfiables issues principalement de SATLIB. Les instances aléatoires présentant peu de relations structurelles entre les variables, elles sont de ce fait hors du cadre de nos objectifs. Nous comparons AcWalk (codé en C/C++) avec CDLS et Walksat. Le choix de CDLS est motivé par le fait qu'il soit (à notre connaissance) le seul algorithme de recherche locale connu exploitant l'analyse de conflits. AcWalk étant basé sur Walkat, la comparaison avec celui-ci est pertinente afin de mesurer l'apport de notre approche. Les tests sont réalisés sur des lames de calcul avec 2 processeurs Intel Xeon 2.4 Ghz et 24 GO de mémoire vive. Les trois algorithmes sont testés avec un nombre maximum de réparations de 10^6 par essai. Les essais sont répétés pendant un temps limite de 1800 secondes. Chaque algorithme est lancé 10 fois et nous mesurons le taux de réussite (t dans $[0,\ 1]$), le temps moyen ($tmoy$) et médian ($tmed$) en secondes sur les exécutions ayant abouties à la construction d'un modèle. Dans tout les tests, la probabilité p pour la marche aléatoire est fixée à 0.5.

Comparé à Walksat, les gains sur ces instances sont très significatifs, que ce soit en temps d'exécution ou en taux de réussite. Nous observons aussi des meilleurs résultats par rapport à CDLS. Même si d'autres tests sont nécessaires afin de positionner notre approche par rapport à d'autres démonstrateurs incomplets, ces premiers résultats montrent toutefois la pertinence de notre approche.

6 Conclusion

Pour mieux résoudre des instances structurées, nous avons présenté une nouvelle approche permettant d'intégrer l'analyse des conflits et l'apprentissage de clauses dans un algorithme de recherche locale pour SAT et les premiers résultats sont encourageants. Parmi les points à améliorer, la gestion des clauses apprises est l'un des points essentiels. En effet, dans la version actuelle d'AcWalk, les clauses asservies sont utilisées pour augmenter l'effet de propagation unitaire et comme nogoods. Le calcul des scores des variables ne se fait que sur les clauses originelles (non apprises). Ce choix est dû à une observation empirique qui a montrée qu'il n'était pas judicieux (du moins d'un point de vue efficacité pratique) d'inclure ces clauses dans le choix de la variable à flipper. Aussi, le nombre de clauses apprises doit être maitrisé afin d'éviter un accroissement trop important, provoquant ainsi une perte significative de l'efficacité pratique de notre solveur. Les techniques utilisées dans le cadre d'une recherche de type minisat sont une première solution mais qu'il est nécessaire d'affiner dans le cadre de la recherche locale.

Par ailleurs, une gestion plus étudiée de la liste tabou doit-être menée. En effet, telle que présentée, cette gestion peut-être agressive (lors de l'application du critère d'aspiration) si la variable à flipper figure parmi les premières à être insérées dans l'interprétation $\mathcal{I}_p$. Une première approche consisterait à introduire un second critère lors de choix de la variable flipper qui consiste à prendre en compte l'âge de la variable, qui correspond à sa date d'insertion dans $\mathcal{I}_p$.

Instances	**Walksat** ($t/tmoy/tmed$)	**CDLS** ($t/tmoy/tmed$)	**AcWalk** ($t/tmoy/tmed$)
bmc-ibm-1	0/-/-	1/50.9/49.74	**1/1.77/1.75**
bmc-ibm-2	0/-/-	1 /0.03/0.03	1/0.01/0.01
bmc-ibm-3	0/-/-	1/101.4/101.0	**1/6.33/6.23**
bmc-ibm-4	0/-/-	1/33.0/33.0	**1/1.50/1.17**
bmc-ibm-5	0/-/-	1/1.60/1.65	1/0.09/0.09
bmc-ibm-6	0/-/-	1/84.78/80.21	**1/2.34/2.29**
bmc-ibm-7	0/-/-	1/0.04/0.04	1/0.03/0.03
bmc-ibm-10	0/-/-	1/36.9/35.5	1/16.51/15.90
bmc-ibm-11	0/-/-	0.8/1451.8/1437.8	**1/11,66/10,98**
bmc-ibm-12	0/-/-	0 /- /-	**1/305.86/292.24**
bmc-ibm-13	0/-/-	0.7/1185.6/1314.1	**1/118.91/104.48**
bmc-gal-8	0/-/-	1/492.7/500.4	**1/7.47/7.41**
bmc-gal-9	0/-/-	1/715.8/725.0	**1/13.22/12.98**
qg1-07	1/24.4/18.6	1/0.18/0.15	1/0.03/0.03
qg1-08	0/-/-	1/472.76/398.37	**1/11.63/5.99**
qg2-07	1/18.4/6.5	1/0.17/0.15	1/0.02/0.02
qg2-08	0/-/-	0.6/819.2/836.9	**1/69.51/42.18**
qg3-08	1/9.5/7.7	1/0.08/0.05	1/0.002/0.002
qg4-09	1/63.2/14.0	1/0.78/0.21	1/0.007/0.004
qg5-11	0/-/-	1/0.47/0.27	1/0.22/ 0.14
qg6-09	0-/-	1/0.02/0.02	1/0.006/0.008
qg7-09	0.1/1078.9/1078.9	1/0.01/0.01	1/0.00/0,004
qg7-13	0/-/-	1/169.2/98.9	**1/12,8/6,87**
bw_large.c	0.3/926.7/941.4	1/4.17/ 4.14	1/0,40/0.38
bw_large.d	0-/-/-	1/138.5/121.2	**1/23.87/25.28**
par16-1	0-/-/-	1/96.0/ 28.1	**1/16.29/10.30**
par16-2	0-/-/-	1/176.1/122.2	**1/54.09/33.75**
par16-3	0-/-/–	1/180.3 /168.9	**1/19.27/16.98**
par16-4	0-/-/-	1/119.0/123.8	**1/13.02/7.57**
par16-5	0-/-/-	1/124.9/108.2	**1/29.78/31.79**
par16-1-c	0-/-/-	1/25.4/23.1	**1/39.86/30.21**
par16-2-c	0-/-/-	**1/78.4/46.2**	1/82,12/63,74
par16-3-c	0-/-/-	1/60.05/43.02	**1/0,42/0,06**
par16-4-c	0-/-/-	**1/31.5/26.5**	1/33,62/14,94
par16-5-c	0-/-/-	1/81.2/84.4	**1/34,29/26,68**
vmpc_24	0.1/1184.1/1184.1	0.7/504.53/447.86	**1/227.21/55.23**
vmpc_25	0-/-/-	0.7/280.28/302.86	**1/313.58/179.54**
vmpc_26	0-/-/-	0/-/-	**0.9/469.99/394.66**
vmpc_28	0-/-/-	0/-/-	**0.5/476.01/221.78**
vmpc_29	0-/-/-	0/-/-	**0.3/753.80/463.35**
vmpc_30	0-/-/-	0/-/-	**0.2/507.74/507.74**
vmpc_33	0-/-/-	0/-/-	0/-/-
vmpc_34	0-/-/-	0/-/-	0/-/-

TABLE 1 – Premiers résultats comparatifs

Références

[1] G. Audemard, JM. Lagniez, B. Mazure, and L. Sais. Analyse de conflits dans le cadre de la recherche locale. In *JFPC*, pages 215–224, 2009.

[2] G. Audemard, JM. Lagniez, B. Mazure, and L. Sais. Boosting local search thanks to cdcl. In *LPAR*, pages 474–488, 2010.

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

[4] Hai Fang. Complete local search for propositional satisfiability. In *AAAI*, pages 161–166, 2004.

[5] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[6] L. Kroc, A. Sabharwal, CP. Gomes, and B. Selman. Integrating systematic and local search paradigms : A new strategy for maxsat. In *IJCAI*, pages 544–551, 2009.

[7] JP. Marques-Silva and KA. Sakallah. Grasp : A search algorithm for propositional satisfiability. *IEEE Trans. Computers*, 48(5) :506–521, 1999.

[8] D. McAllester, B. Selman, and H. Kautz. Evidence for invariants in local search. In *AAAI*, pages 321–326, 1997.

[9] L. Zhang, CF. Madigan, and MH. Moskewicz. Efficient conflict driven learning in a boolean satisfiability solver. In *ICCAD*, pages 279–285, 2001.