Résolution exacte de MinSAT

Chu Min Li, Felip Manya, Zhe Quan, Zhu Zhu

Résolution exacte de MinSAT *

Chu Min Li [1] **Felip Manyà** [2] **Zhe Quan** [1] **Zhu Zhu** [1]

[1] MIS, Université de Picardie Jules Verne, 5 Rue du Moulin Neuf, 80000 Amiens, France
[2] IIIA-CSIC, Campus UAB, 08193 Bellaterra, Spain

{chu-min.li,zhu.zhu}@u-picardie.fr felip@iiia.csic.es quanzhe@gmail.com

Résumé

MinSAT est le problème consistant à trouver une affectation qui minimise le nombre de clauses satisfaites dans une formule CNF. Nous présentons une approche originale pour résoudre MinSAT de façon exacte, qui repose sur le codage MaxSAT et les solveurs MaxSAT. Notre étude empirique fournit la preuve que, en l'absence de spécifiques solveurs exacts pour MinSAT, l'approche générique proposée dans ce papier est compétitive.

Abstract

MinSAT is the problem of finding a truth assignment that minimizes the number of satisfied clauses in a CNF formula. We present an original approach for exact MinSAT solving, which relies on solving MinSAT using MaxSAT encodings and MaxSAT solvers. Our empirical investigation provides evidence that, in the absence of genuine exact solvers for MinSAT, the generic approach proposed in this paper is competitive.

1 Introduction

MaxSAT est le problème consistant à trouver une affectation qui maximise le nombre de clauses satisfaites dans une formule CNF, et MinSAT est le problème consistant à trouver une affectation qui minimise le nombre de clauses satisfaites. En dépit que MaxSAT est plus populaire et a attiré l'intérêt de la communauté SAT ces dernières années [8], nous pensons qu'il vaut la peine d'étudier MinSAT et de mettre au point des techniques de résolution exacte rapide pour MinSAT.

MinSAT a des intérêts théoriques et pratiques. Du point de vue théorique, nous soulignons les travaux existants sur les algorithmes d'approximation pour MinSAT (voir e.g. [2, 6, 11] et les références y citées). Du point de vue pratique, nous aimerions mettre l'accent sur les réductions existantes à problème MinSAT dans les domaines tels que la bio-informatique [5] et l'apprentissage supervisé [4].

Malheureusement, au meilleur de notre connaissance, il n'existe pas de solveur exact pour MinSAT similaire aux solveurs modernes séparation et évaluation développés pour MaxSAT. Compte tenu de l'énorme quantité d'efforts consacrés jusqu'à présent à la conception et au développement de solveurs compétitifs exacts pour MaxSAT, il semble difficile d'avoir de spécifiques solveurs exacts efficaces pour MinSAT dans un proche avenir. Donc, nous pensons qu'il vaut la peine d'explorer la résolution exacte de MinSAT à l'aide d'une méthode générique de résolution de problème.

Notre approche générique pour la résolution exacte de MinSAT s'appuie sur la définition des codages efficaces et originaux de MinSAT en MaxSAT, résoudre les problèmes codés par un solveur MaxSAT moderne, puis en déduire une solution optimale de MinSAT à partir d'une solution optimale de MaxSAT. Pour être plus précis, nous définissons trois codages. Le premier est une réduction directe de MinSAT en MaxSAT partiel. Les deux autres codages sont plus complexes, car ils commencent par une réduction de MinSAT en MaxClique, puis par une seconde réduction de MaxClique en MaxSAT partiel. La différence entre les deux codages est que le premier utilise un codage classique de MaxClique en MaxSAT partiel, tandis que le deuxième utilise un nouveau codage de Maxclique en MaxSAT qui repose sur une partition du graphe concerné en cliques maximales. L'étude empirique que nous avons réalisée montre que cette approche générique est compétitive.

Notre objectif est de profiter de l'évolution récente de MaxSAT, qui ont produit une technologie qui devient très compétitive dans la résolution de problème d'optimisation. D'autre part, il existe une large gamme de problèmes de minimisation qui ont un codage plus naturel et plus com-

*Ce travail est partiellement financé par la Generalitat de Catalunya avec la subvention 2009-SGR-1434, et les projets de recherche *Ministerio de Ciencia e Innovación* CONSOLIDER CSD2007-0022, INGENIO 2010, Acción Integrada HA2008-0017, TIN2007-68005-C04-04 et TIN2009-14704-C03-01.

pact en MinSAT qu'en MaxSAT. Résoudre directement et efficacement MinSAT peut faciliter le travail de l'utilisateur. Notre approche permet ainsi de contribuer à diffuser les résultats de la recherche et la technologie de la communauté SAT à d'autres communautés travaillant sur les problèmes d'optimisation.

Ce papier est structuré comme suit. Dans la section 2, nous donnons quelques définitions. Dans la section 3, nous définissons le codage direct de MinSAT en MaxSAT partiel. Dans la section 4, nous définissons les codages qui sont fondés en premier lieu sur la réduction de MinSAT à MaxClique, puis MaxClique vers MaxSAT partiel. Dans la section 5, nous présentons l'étude empirique, afin d'évaluer notre approche de la résolution exacte de MinSAT. Dans la section 6, nous donnons les conclusions et suggérons quelques directions pour les futures recherches.

2 Preliminaires

Considérons un ensemble de variables booléennes $\{x_1$,x_2,...,$x_n\}$, un littéral est une variable x_i ou sa négation $\bar{x}_i$, une clause est un ou logique de quelques littéraux. Une formule CNF (Conjunctive Normal Form) ϕ est un ensemble de clauses. MaxSAT est le problème consistant à trouver une affectation de valeurs de vérité qui maximise le nombre de clauses satisfaites dans une formule CNF, et MinSAT est le problème consistant à trouver une affectation qui minimise le nombre de clauses satisfaites. Min-2-SAT (Min-3-SAT) est le problème MinSAT restreint à des clauses avec au plus 2 (3) littéraux.

Il est connu que, pour toute instance de MinSAT, il y a une affectation qui satisfait zéro clauses si chaque variable apparaît uniquement avec la polarité positive ou seulement avec la polarité négative. De même, si toutes les clauses sont unitaires, la solution à MinSAT est facilement obtenue en affectant une variable à vrai si elle est apparaît dans moins de clauses que sa négation, et la variable à faux sinon. Cependant, en général, MinSAT (même Min-2-SAT) est NP-difficile [6].

Nous considérons également l'extension de MaxSAT connu sous le nom de MaxSAT partiel, car cette extension permet de mieux représenter et résoudre certains problèmes NP-difficiles. Un problème MaxSAT partiel est une formule CNF dont certaines clauses sont *relaxables* ou *souples* et les restes sont *non-relaxables* ou *dures*. Les clauses dures sont représentées entre crochets. Alors que MaxSAT consiste à trouver une affectation de valeurs de vérité qui maximise le nombre de clauses satisfaites, résoudre une instance MaxSAT partiel revient à trouver une affectation qui satisfait toutes les clauses dures et le maximum clauses souples. Étant donné un graphe non orienté $G(V, E)$, V est l'ensemble de sommets et E est l'ensemble d'arêtes, le problème de couverture minimale de sommets consiste à trouver un sous-ensemble minimum V' de V, tel que pour chaque arête $\{u, v\}$ dans E, au moins un des sommets u et v est dans V'. Une clique dans un graphe non orienté $G = (V, E)$ est un sous-ensemble C de V tel que, pour tous les deux sommets de C, il existe une arête les reliant. Cela revient à dire que le sous-graphe induit par C est complet. Une clique maximale est une clique qui ne peut être étendue en ajoutant un sommet supplémentaire, et une clique maximum est une clique de la plus grande cardinalité possible. Le problème de clique maximum (MaxClique) pour un graphe non orienté G consiste à trouver une clique maximum dans G. Une partition en cliques d'un graphe non orienté $G = (V, E)$ est une partition de V en sous-ensembles disjoints $V_1, \ldots, V_k$ tels que, pour $1 \leq i \leq k$, le graphe induit par V_i est une clique.

3 Codage 1 : Coder directement MinSAT en MaxSAT partiel

Définition 1 *Étant donnée une instance MinSAT I composée de l'ensemble des clauses $\mathcal{C}_I = \{C_1, \ldots, C_m\}$ et l'ensemble de variables X_I, le codage direct de I en MaxSAT est défini comme suit :*

- *L'ensemble des variables propositionnelles est $X_I \cup \{c_1, \ldots, c_m\}$, où $\{c_1, \ldots, c_m\}$ est un ensemble de variables auxiliaires.*
- *Pour toute clause $C_i \in \mathcal{C}_I$, la clause dure $c_i \leftrightarrow C_i$ est ajoutée.*
- *Pour toute variable auxiliaire c_i, la clause souple unitaire $\neg c_i$ est ajoutée.*

L'ensemble des clauses dures implique que c_i est vraie si et seulement si C_i est satisfaite, ou de manière équivalente, c_i est faux si et seulement si C_i est falsifié. Puisque notre objectif est de maximiser le nombre de clauses falsifiées, nous ajoutons une unité clause de $\neg c_i$ pour chaque $C_i \in \mathcal{C}_I$. Par conséquent, si les variables auxiliaires affectées à vrai dans une solution optimale du codage MaxSAT sont $\{c_{i_1}, \ldots, c_{i_k}\}$, alors $\{C_{i_1}, \ldots, C_{i_k}\}$ est un ensemble minimum de clauses satisfaites dans l'instance I de MinSAT. Nous pouvons construire une affectation optimale de MinSAT en utilisant l'affectation des variables de X_I à partir d'une affectation optimale de MaxSAT partiel.

Exemple 1. Soit I l'instance MinSAT $\{C_1, C_2, C_3, C_4, C_5\}$, $C_1 = a \vee b$, $C_2 = a \vee \neg b$, $C_3 = \neg a \vee b$, $C_4 = \neg a \vee \neg b$, et $C_5 = a \vee c$. Le codage

FIGURE 1 – Un graphe auxiliaire (à gauche) et son complément (à droite)

direct MaxSAT pour I est :

$$[c_1 \leftrightarrow a \vee b]$$
$$[c_2 \leftrightarrow a \vee \neg b]$$
$$[c_3 \leftrightarrow \neg a \vee b]$$
$$[c_4 \leftrightarrow \neg a \vee \neg b]$$
$$[c_5 \leftrightarrow a \vee c]$$
$$\neg c_1$$
$$\neg c_2$$
$$\neg c_3$$
$$\neg c_4$$
$$\neg c_5$$

Puisque $c_2 = c_3 = c_4 = vrai$ et $c_1 = c_5 = a = b = c = faux$ est la solution optimale pour MaxSAT partiel, alors $\{C_2, C_3, C_4\}$ est un ensemble minimum de clauses satisfaites dans I, et $a = b = c = faux$ est une solution optimale de MinSAT.

4 Coder MinSAT en MaxSAT par l'intermédiare de MaxClique

Soit I l'instance MinSAT composée de l'ensemble de clauses $\mathcal{C}_I$ et l'ensemble de variables X_I. Le graphe auxiliaire $G_I(V_I, E_I)$ correspondant à I est construit comme suit [11] : chaque sommet de V_I correspond à une clause dans $\mathcal{C}_I$. Pour chaque deux sommets v_i et v_j de V_I, l'arête $\{v_i, v_j\}$ est dans E_I si et seulement si les clauses correspondantes c_i et c_j sont telles qu'il existe une variable $x \in X_I$ apparaîssant sous forme positive dans une clause et sous forme négative dans l'autre clause. Le complément d'un graphe auxiliaire G_I est le graphe $\overline{G_I}$ avec les même sommets tel que deux sommets de $\overline{G_I}$ sont adjacents si et seulement s'ils ne sont pas adjacents dans G_I.

L'intuition dans le graphe auxiliaire est la suivante : chaque arête dans le graphe auxiliaire entre les sommets correspondent aux clauses c_i et c_j partageant au moins une variable sous forme positive dans une clause et négative dans l'autre. Aucune affectation de variables ne peut contredire c_i et c_j en même temps ; Ainsi une affectation qui viole c_i doit nécessairement satisfaire c_j, et réciproquement.

Exemple 2. Soit I l'instance de MinSAT $\{C_1, C_2, C_3, C_4, C_5\}$, $C_1 = a \vee b$, $C_2 = a \vee \neg b$, $C_3 = \neg a \vee b$, $C_4 = \neg a \vee \neg b$, et $C_5 = a \vee c$. La figure 1 montre le graphe auxiliaire G_I correspondant à I (à gauche) et son complément $\overline{G_I}$ (à droite).

4.1 Codage 2 : Codage basé sur MaxClique

Marathe et Ravi [11] ont prouvé que le nombre de clauses d'une instance I de MinSAT satisfaites par une affectation optimale est égale à la cardinalité d'une couverture minimum de sommets dans le graphe auxiliaire G_I. D'autre part, l'ensemble des sommets n'appartenant pas à une clique maximum dans le graphe complémentaire $\overline{G_I}$ est une couverture minimum de sommets de G_I. Cela découle du fait que, pour n'importe quel graphe $G(V, E)$, $V' \subseteq V$ est une couverture de sommets si et seulement si $V - V'$ est une clique dans le complément de G. Par conséquent, le nombre de clauses d'une instance I MinSAT satisfaites par une solution optimale est égal au nombre total de sommets moins la cardinalité d'une clique maximum de $\overline{G_I}$.

En outre, une affectation optimale pour l'instance MinSAT I peut être déduite d'une couverture minimum de sommets pour G_I comme suit [11] : les variables apparaîssant dans les clauses correspondant aux sommets qui n'appartiennent pas à la couverture minimum de sommets doivent être affectées de façon à ce que ces clauses soient violées, ce qui est possible parce que, par construction de G_I, ces clauses ne contiennent à la fois x et $\bar{x}$ pour tout $x \in X_I$; d'autres variables sont affectées à une valeur arbitraire.

Par conséquent, nous pouvons déduire une affectation optimale pour l'instance I de MinSAT d'une clique maximum de $\overline{G_I}$: les variables apparaîssant dans les clauses correspondant à sommets appartenant à la clique maximum doivent être affectées en sorte que ces clauses soient violées, d'autres variables sont affectées arbitrairement. Par conséquent, afin de trouver une affectation optimale pour l'instance MinSAT I, nous pouvons chercher une clique maximum dans $\overline{G_I}$. Le problème de MinSAT, par l'intermédiare de MaxClique, est naturellement codé en un problème MaxSAT partiel comme suit.

Définition 2. *Étant donnée une instance I de MinSAT composée de l'ensemble de clause $\mathcal{C}_I = \{C_1, \ldots, C_m\}$, le codage MaxSAT de I basé sur MaxClique est défini comme suit :*

– *L'ensemble de variables propositionnelles est* $\{c_1, \ldots, c_m\}$.
– *Pour toutes les deux clauses* C_i, C_j *de* $\mathcal{C}_I$ *tel que* C_i *contient une occurrence de* l *et* C_j *contient une occurrence de* $\neg l$*, la clause dure* $\neg c_i \vee \neg c_j$ *est ajoutée.*
– *Pour chaque variable propositionnelle* c_i*, une clause unitaire souple* c_i *est ajoutée.*

Observons que chaque variable propositionnelle correspond à un sommet de $\overline{G_I}$, et qu'il existe une clause dure pour tous les deux sommets non adjacents dans $\overline{G_I}$, signifiant que les deux sommets ne peuvent pas être dans la même clique, et une clause unitaire souple pour chaque sommet de $\overline{G_I}$. L'ensemble des variables propositionnelles affectées à vrai dans une solution optimale de l'instance MaxSAT forme une clique maximum de $\overline{G_I}$.

Exemple 3. Soit I l'instance MinSAT $\{C_1, C_2, C_3, C_4, C_5\}$, $C_1 = a \vee b$, $C_2 = a \vee \neg b$, $C_3 = \neg a \vee b$, $C_4 = \neg a \vee \neg b$, et $C_5 = a \vee c$. Le codage MaxSAT pour I basé sur MaxClique est

$$
\begin{array}{l}
[\neg c_1 \vee \neg c_2] \\
[\neg c_1 \vee \neg c_3] \\
[\neg c_1 \vee \neg c_4] \\
[\neg c_2 \vee \neg c_3] \\
[\neg c_2 \vee \neg c_4] \\
[\neg c_3 \vee \neg c_4] \\
[\neg c_3 \vee \neg c_5] \\
[\neg c_4 \vee \neg c_5] \\
c_1 \\
c_2 \\
c_3 \\
c_4 \\
c_5
\end{array}
$$

Notons que contrairement au codage direct (codage 1), le sens prévu de c_i est que c_i est vraie si est seulement si C_i est violée. La partie dure implique que chaque paire de clauses C_i et C_j contenant des littéraux complémentaires ne peuvent pas être violées simultanément. Comme notre objectif est de maximiser le nombre de clauses falsifiées, nous ajoutons une clause souple unitaire c_i pour chaque clause $C_i \in \mathcal{C}_I$. Dans une solution optimale de I, $c_1 = c_5 = vrai$ et $c_2 = c_3 = c_4 = faux$, les sommets correspondant aux variables affectées à vrai appartiennent à une clique maximum de $\overline{G_I}$ et les autres sommets appartiennent à une couverture minimum de sommets en G_I. Puisque une couverture minimum de sommets de G_I est de taille 3, le nombre minimum de clauses satisfaites dans I est 3. Nous pouvons construire une affectation optimale pour MinSAT en falsifiant C_1 et C_5, qui sont les clauses correspondant aux sommets appartenant à une clique maximum de $\overline{G_I}$. Par conséquent, si a, b, et c sont affectées à faux, on obtient une affectation optimale pour l'instance I de MinSAT.

4.2 Codage 3 : Codage amélioré basé sur MaxClique

Le codage de MinSAT en MaxSAT basé sur MaxClique peut être amélioré en réduisant le nombre de clauses souples en considérant le fait suivant : si le graphe auxiliaire d'une instance de MinSAT contient une clique $C = \{c_{i_1}, \ldots, c_{i_k}\}$, alors la partie dure contient les clauses $\neg c_{i_j} \vee \neg c_{i_h}$ pour tout j, h tel que $1 \leq j < h \leq k$. Remarquons que ces clauses dures codent la au-plus-une condition : au plus un littéral dans $\{c_{i_1}, \ldots, c_{i_k}\}$ peut être satisfait. Par conséquent, nous pouvons remplacer les k clauses souples $c_{i_1}, \ldots, c_{i_k}$ par la clause souple $c_{i_1} \vee \cdots \vee c_{i_k}$, car le nombre de clauses satisfaites est le même dans les deux cas dans une solution optimale.

En général, étant donné un graphe auxiliaire $G_I(V_I, E_I)$ d'une instance MinSAT I, une partition en cliques $V_1, \ldots, V_k$ de G_I, le codage amélioré de MinSAT en MaxSAT basé sur MaxClique a, pour chaque clique V_i dans la partition, une clause souple formée par la disjonction des sommets de V_i.

Définition 3. *Étant donnée une instance* I *de MinSAT composée de l'ensemble de clause* $\mathcal{C}_I = \{C_1, \ldots, C_m\}$*, le codage MaxSAT amélioré de* I *basé sur MaxClique est défini comme suit :*

– *L'ensemble de variables propositionnelles est* $\{c_1, \ldots, c_m\}$.
– *Pour toutes les deux clauses* C_i, C_j *de* $\mathcal{C}_I$ *tel que* C_i *contient une occurrence de* l *et* C_j *contient une occurrence de* $\neg l$*, la clause dure* $\neg c_i \vee \neg c_j$ *est ajoutée.*
– *Dans une partition du graphe auxiliaire* $G_I(V_I, E_I)$ *en cliques, une clause souple est ajoutée pour chaque clique qui est un* ou *logique des variables correspondantes aux sommets dans la clique.*

Exemple 4. Soit I l'instance MinSAT $\{C_1, C_2, C_3, C_4, C_5\}$, $C_1 = a \vee b$, $C_2 = a \vee \neg b$, $C_3 = \neg a \vee b$, $C_4 = \neg a \vee \neg b$, et $C_5 = a \vee c$. Le codage amélioré de I en MaxSAT basé sur MaxClique est :

$$
\begin{array}{l}
[\neg c_1 \vee \neg c_2] \\
[\neg c_1 \vee \neg c_3] \\
[\neg c_1 \vee \neg c_4] \\
[\neg c_2 \vee \neg c_3] \\
[\neg c_2 \vee \neg c_4] \\
[\neg c_3 \vee \neg c_4] \\
[\neg c_3 \vee \neg c_5] \\
[\neg c_4 \vee \neg c_5] \\
c_1 \vee c_2 \vee c_3 \vee c_4 \\
c_5
\end{array}
$$

Nous remarquons que $\{\{c_1, c_2, c_3, c_4\}, \{c_5\}\}$ est une partition de clique du graphe auxiliaire de I (cf. Figure 1), et que les six premières clauses dures signifient qu'au plus un des littéraux c_1, c_2, c_3, c_4 est satisfait par une affectation optimale.

Puisque le problème de trouver une partition minimale de cliques d'un graphe est NP-difficile, nous proposons d'appliquer une méthode heuristique pour partitionner le graphe en cliques pour obtenir le codage amélioré. L'heuristique utilisée dans notre étude empirique est l'algorithme 1.

Étant donné un graphe G et une liste P de cliques disjointes de G, P est initialement vide, nous dénotons par #c(v) le nombre de cliques dans P dans lesquelles le sommet v peut être inséré pour les agrandir. Algorithm 1 partitionne G en cliques en sélectionnant le sommet v avec le minimum #c(v), dans le cas où il y a plusieurs sommets avec le minimum #c(v), celui qui a le moindre degré (c'est à dire avec le minimum sommets adjacents) est choisi. Puis l'algorithme insère v dans la première clique de P où v peut être inséré si #c(v)>0, et crée une nouvelle clique dans P pour y insérer v sinon. Soit C une clique dans P, et v_i et v_j sont deux sommets non adjacents qui peuvent être respectivement insérés dans C pour obtenir une plus grande clique, observons que l'insertion de v_i dans C empêche v_j d'être inséré dans C, et réciproquement. L'intuition dans la sélection de v dans l'algorithme 1 est que le sommet le plus contraint (c'est à dire avec le moins de possibilités dans P) est inséré en premier, en évitant de créer une nouvelle clique pour ce sommet après l'insertion d'autres sommets.

Nous avons comparé l'algorithme 1 avec les heuristiques naturelles suivantes :

`H-Min` : même que l'algorithme 1 sauf que chaque fois que le sommet avec le plus petit degré est sélectionné pour être inséré dans une clique ;

`H-Max` : même que l'algorithme 1 sauf que chaque fois que le sommet avec le plus grand degré est sélectionné pour être inséré dans une clique ;

`H-Maximal-Clique-Min` : une clique maximale est calculée à la fois. Tant qu'il y a des sommets qui peuvent être insérés dans une clique pour l'agrandir, un de ces sommets avec le plus petit degré est inséré dans la clique. Après que la clique devient maximale et retiré du graphe, d'autres cliques maximales sont calculées de la même manière dans le graphe restant jusqu'à ce que le graphe devienne vide ;

`H-Maximal-Clique-Max` : même que H-Maximal-clique-Min, sauf que chaque fois que le sommet avec le plus grand degré est inséré dans la clique en construction.

L'Heuristique H-Min est utilisée dans [14] pour partitionner un graph en ensembles indépendents, et les heuristiques H-Maximal-Clique-Min et H-Maximal-Clique-Max sont utilisées dans [3, 12] pour partitionner aussi un graphe en ensemble indépendents. Ces partitions sont utilisées pour donner une borne supérieure de la caridinalité d'une clique maximum dans le graphe. Leur complexité en temps est meilleure que l'algorithme 1, puisque le calcul de #c n'est pas très simple dans l'algorithme 1.

Algorithm 1: cliquePartition(G)

Input: Un graphe G=(V, E)
Output: Une partition en cliques de G

```
1  begin
2      P ← ∅;
3      while G n'est pas vide do
4          Pour chaque sommet v dans G, calculer le
           nombre de cliques #c dans P dans lesquelles v
           est adjacent à tous les sommets;
5          v ← le sommet de G avec le minimum #c et
           puis avec le minimum degré ;
6          supprimer v de G;
7          if il y a une clique C dans P dans laquelle v
           est adjacent à tous les sommets then
8              ajouter v à C ;
9          else
10             créer une nouvelle clique C;
11             ajouter v à C;
12             P ← P ∪ {C} ;
13     retourner P ;
14 end
```

Cependant, l'algorithme 1 est quand même rapide et est significativement meilleur que toutes les autres heuristiques, car il donne la plus petite partition selon nos résultats expérimentaux (non reportés ici). Nous aimerions également faire remarquer que nous obtenons une meilleure borne inférieure initiale de MaxSAT si nous réduisons le nombre de clauses souples. Ainsi l'algorithme 1 permet le minimum nombre de clauses souples dans le codage amélioré et la meilleure borne inférieure initial de MaxSAT parmi toutes heuristiques testées.

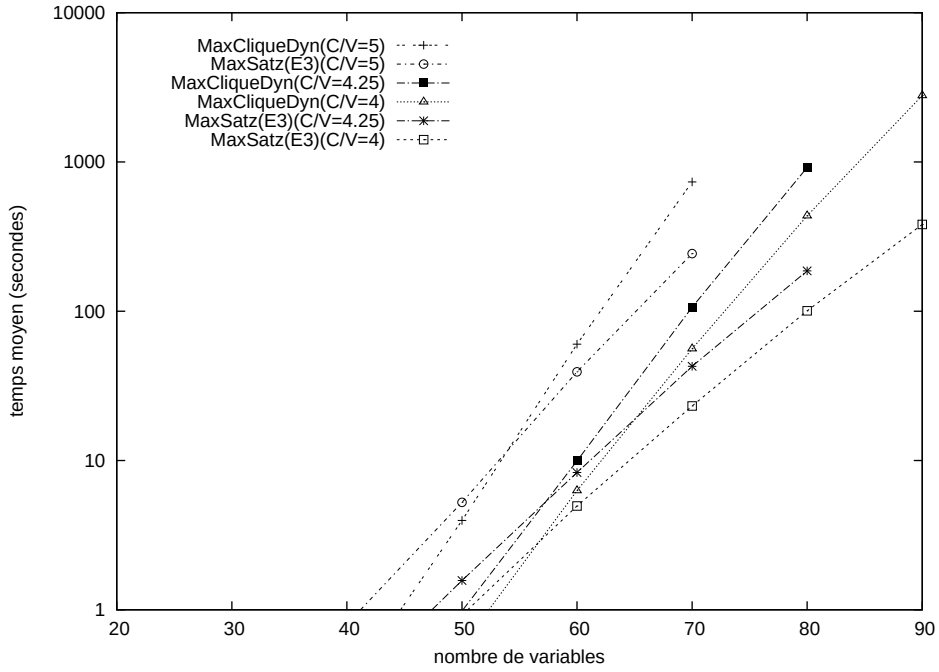
5 Résultats expérimentaux

Nous avons effectué une comparaison empirique des trois codages de MinSAT en MaxSAT sur plusieurs solveurs MaxSAT, et comparé nos résultats avec les résultats obtenus par la résolution de MinSAT par les deux meilleurs solveurs exacts de MaxClique à nos connaissances.

Les solveurs utilisés dans notre étude empirique sont les suivants :

– MaxSatz [9] : nous avons utilisé la version pour MaxSAT pondéré partiel qui a participé à l'évaluation

FIGURE 2 – *Min-3-SAT, échelle logarithmique*



MaxSAT 2009 avec un petit modification de l'heuristique utilisée pour sélectionner les variables dans la détection des littéraux d'échec : alors que la version de l'évaluation MaxSAT 2009 applique la détection des littéraux d'échec à toutes les variables apparaîssant au moins deux fois négativement et deux fois positivement dans les clauses binaire, dans la version utilisée dans ce papier, elle est appliqué à toutes les variables ayant au moins une occurrence positive et au moins deux occurrences négatives dans les clauses binaires, et à toutes les variables ayant au moins une occurrence négative et au moins deux occurrences positives dans les clauses binaires. Ceci parce que les variables dans une instance MaxSAT codant une instance MaxClique a seulement une occurrence positive, même si elles peuvent avoir plusieures occurrences négatives dans les clauses dures.

– WBO [10] : nous avons utilisé la version de ce solveur de MaxSAT pondéré partiel qui a participé à l'évaluation MaxSAT 2009.
– PM2 [1] : nous avons utilisé la version de ce solveur de MaxSAT partiel qui a participé à l'évaluation MaxSAT 2009.
– SAT4J-Maxsat : [1] nous avons utilisé la version de ce solveur de MaxSAT pondéré partiel qui a participé à l'évaluation MaxSAT 2009.
– MaxCliqueDyn [7] : nous avons obtenu le code source de ce solveur MaxClique de D. Janezic, qui est l'un de ses auteurs, en Janvier 2010.
– Cliquer [13] : nous avons utilisé la dernière version publique de ce solveur, qui a été publié en 2008. [2]

Les MaxClique solveurs MaxCliqueDyn et Cliquer résolvent une instance I de MinSAT en cherchant une clique maximum dans le graphe auxiliaire complémentaire $\overline{G_I}$. Les MaxSAT solveurs MaxSatz, WBO, PM2, et SAT4J-Maxsat résolvent I en utilisant trois codages de MinSAT en MaxSAT partiel qui sont définis dans les sections précédentes.

Comme benchmarks, nous avons généré des instances aléatoires de Min-2-SAT et Min-3-SAT dont chaque clause contient exactement 2 et 3 littéraux respectivement. Le nombre de variables dans les instances Min-2-SAT et Min-3-SAT variait de 20 à 100. Les ratios #clause sur #variable pour Min-2-SAT sont 1 et 3, et les ratios #clause sur #variable pour Min-3-SAT sont 4, 4,25 et 5. Des expérimentations ont été effectuées sur un MacPro avec un processeur Intel Xeon à 2,8 GHz et 4 Gb de mémoire avec Mac OS X 10.5. Le temps d'exécution est limité à 3 heures par instance.

Le tableau 1 (resp. le tableau 2) contient les résultats expérimentaux obtenus pour les instances de Min-2-SAT (resp. Min-3-SAT) avec les solveurs de MaxSAT et de MaxClique. Le tableau 1 et le tableau 2 montrent, pour

1. http ://www.sat4j.org/

2. http ://users.tkk.fi/pat/cliquer.html

TABLE 1 – Nombre d'instances résolues en moins de 3 heures et temps d'exécution en secondes de MaxSatz, MaxCliqueDyn (Dyn dans le tableau), Cliquer, PM2, WBO et SAT4J-Maxsat sur Min-2-SAT aléatoire. C/V est le ratio #clause sur #variable.

instance		*MaxSatz*			*Dyn*	*Cliquer*	*PM2*			*WBO*			*sat4j-maxsat*		
#var	C/V	*E*1	*E*2	*E*3			*E*1	*E*2	*E*3	*E*1	*E*2	*E*3	*E*1	*E*2	*E*3
20	1.0	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.02 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.07 *(50)*	0.07 *(50)*	0.04 *(50)*
30	1.0	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.02 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.18 *(50)*	0.14 *(50)*	0.05 *(50)*
40	1.0	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.02 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.48 *(50)*	0.27 *(50)*	0.05 *(50)*
50	1.0	0.01 *(50)*	0.01 *(50)*	0.00 *(50)*	0.02 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.92 *(50)*	0.87 *(50)*	0.06 *(50)*
60	1.0	0.01 *(50)*	0.08 *(50)*	0.00 *(50)*	0.02 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	2.71 *(50)*	2.73 *(50)*	0.06 *(50)*
70	1.0	0.02 *(50)*	0.21 *(50)*	0.00 *(50)*	0.02 *(50)*	0.00 *(50)*	0.01 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	0.00 *(50)*	17.40 *(50)*	27.02 *(50)*	0.05 *(50)*
80	1.0	0.03 *(50)*	1.18 *(50)*	0.00 *(50)*	0.02 *(50)*	0.00 *(50)*	0.01 *(50)*	0.00 *(50)*	0.00 *(50)*	0.02 *(50)*	0.00 *(50)*	0.00 *(50)*	505.9 *(50)*	190.6 *(50)*	0.08 *(50)*
90	1.0	0.58 *(50)*	9.38 *(50)*	0.00 *(50)*	0.02 *(50)*	0.00 *(50)*	0.01 *(50)*	0.00 *(50)*	0.00 *(50)*	0.02 *(50)*	0.01 *(50)*	0.00 *(50)*	2710 *(24)*	1995 *(45)*	0.08 *(50)*
100	1.0	0.26 *(50)*	27.34 *(50)*	0.00 *(50)*	0.02 *(50)*	0.00 *(50)*	0.01 *(50)*	0.01 *(50)*	0.00 *(50)*	0.02 *(50)*	0.01 *(50)*	0.00 *(50)*	2596 *(14)*	6358 *(12)*	0.11 *(50)*
20	3.0	0.01 *(50)*	0.03 *(50)*	0.00 *(50)*	0.02 *(50)*	0.00 *(50)*	0.02 *(50)*	0.02 *(50)*	0.00 *(50)*	0.01 *(50)*	0.01 *(50)*	0.00 *(50)*	0.81 *(50)*	1.20 *(50)*	0.07 *(50)*
30	3.0	0.03 *(50)*	1.39 *(50)*	0.01 *(50)*	0.02 *(50)*	0.00 *(50)*	0.14 *(50)*	0.14 *(50)*	0.00 *(50)*	0.14 *(50)*	0.22 *(50)*	0.00 *(50)*	112.5 *(50)*	1530 *(32)*	0.17 *(50)*
40	3.0	0.12 *(50)*	52.95 *(50)*	0.01 *(50)*	0.03 *(50)*	0.01 *(50)*	0.19 *(50)*	0.51 *(50)*	0.00 *(50)*	0.07 *(50)*	0.52 *(50)*	0.00 *(50)*	9904 *(4)*	- *(0)*	0.20 *(50)*
50	3.0	0.89 *(50)*	2053 *(48)*	0.01 *(50)*	0.04 *(50)*	1.23 *(50)*	5.25 *(50)*	7.89 *(50)*	0.00 *(50)*	42.35 *(50)*	51.12 *(50)*	0.01 *(50)*	- *(0)*	- *(0)*	0.43 *(50)*
60	3.0	7.42 *(50)*	- *(0)*	0.02 *(50)*	0.07 *(50)*	15.88 *(50)*	49.35 *(50)*	79.77 *(50)*	0.01 *(50)*	207.6 *(50)*	93.17 *(50)*	0.20 *(50)*	- *(0)*	- *(0)*	2.41 *(50)*
70	3.0	43.25 *(50)*	- *(0)*	0.03 *(50)*	0.13 *(50)*	68.01 *(50)*	144.3 *(50)*	135.8 *(50)*	0.01 *(50)*	225.1 *(50)*	143.9 *(50)*	0.08 *(50)*	- *(0)*	- *(0)*	2.42 *(50)*
80	3.0	201.2 *(50)*	- *(0)*	0.05 *(50)*	0.25 *(50)*	302.4 *(50)*	199.9 *(50)*	231.8 *(50)*	0.02 *(50)*	416.8 *(50)*	432.9 *(50)*	0.70 *(50)*	- *(0)*	- *(0)*	3.35 *(50)*
90	3.0	1355 *(49)*	- *(0)*	0.08 *(50)*	0.91 *(50)*	895.5 *(42)*	353.1 *(50)*	330.3 *(50)*	0.05 *(50)*	558.8 *(50)*	295.1 *(50)*	1.37 *(50)*	- *(0)*	- *(0)*	5.38 *(50)*
100	3.0	3258 *(46)*	- *(0)*	0.11 *(50)*	1.51 *(50)*	1185 *(44)*	256.7 *(10)*	455.3 *(50)*	0.09 *(50)*	548.5 *(50)*	665.9 *(50)*	29.27 *(50)*	- *(0)*	- *(0)*	96.91 *(50)*

TABLE 2 – Nombre d'instances résolues en moins de 3 heures et temps d'exécution en secondes de MaxSatz, MaxCliqueDyn (Dyn dans le tableau) et Cliquer sur Min-3-SAT aléatoire. C/V est le ratio #clause sur #variable.

instance		*MaxSatz*			*Dyn*	*Cliquer*
#var	C/V	*E*1	*E*2	*E*3		
20	4.00	0.02*(50)*	0.10*(50)*	0.01*(50)*	0.02*(50)*	0.00*(50)*
30	4.00	0.24*(50)*	7.73*(50)*	0.04*(50)*	0.03*(50)*	0.04*(50)*
40	4.00	3.28*(50)*	507.8*(50)*	0.19*(50)*	0.09*(50)*	4.40*(50)*
50	4.00	49.30*(50)*	- *(0)*	0.92*(50)*	0.57*(50)*	454.4*(50)*
60	4.00	742.4*(50)*	- *(0)*	4.96*(50)*	6.29*(50)*	3886*(11)*
70	4.00	5735*(34)*	- *(0)*	23.21*(50)*	56.01*(50)*	- *(0)*
80	4.00	- *(0)*	- *(0)*	100.7*(50)*	436.0*(50)*	- *(0)*
90	4.00	- *(0)*	- *(0)*	381.5*(50)*	2788*(50)*	- *(0)*
100	4.00	- *(0)*	- *(0)*	1658*(33)*	5610*(9)*	- *(0)*
20	4.25	0.02*(50)*	0.14*(50)*	0.01*(50)*	0.02*(50)*	0.00*(50)*
30	4.25	0.30*(50)*	12.05*(50)*	0.05*(50)*	0.03*(50)*	0.07*(50)*
40	4.25	4.67*(50)*	992.5*(50)*	0.28*(50)*	0.12*(50)*	15.94*(50)*
50	4.25	75.6*(50)*	- *(0)*	1.57*(50)*	0.98*(50)*	945.0*(49)*
60	4.25	1153*(50)*	- *(0)*	8.31*(50)*	9.94*(50)*	5385*(10)*
70	4.25	5989*(5)*	- *(0)*	42.77*(50)*	106.4*(50)*	- *(0)*
80	4.25	- *(0)*	- *(0)*	186.3*(50)*	917.4*(50)*	- *(0)*
90	4.25	- *(0)*	- *(0)*	760.4*(50)*	4453*(41)*	- *(0)*
100	4.25	- *(0)*	- *(0)*	2819*(26)*	- *(0)*	- *(0)*
20	5.00	0.03*(50)*	0.39*(50)*	0.02*(50)*	0.02*(50)*	0.00*(50)*
30	5.00	0.63*(50)*	55.91*(50)*	0.14*(50)*	0.04*(50)*	0.26*(50)*
40	5.00	10.87*(50)*	5693*(48)*	0.80*(50)*	0.30*(50)*	63.98*(50)*
50	5.00	226.8*(50)*	- *(0)*	5.24*(50)*	3.97*(50)*	3035*(35)*
60	5.00	3803*(48)*	- *(0)*	39.3*(50)*	60.14*(50)*	- *(0)*
70	5.00	- *(0)*	- *(0)*	243.4*(50)*	735.2*(50)*	- *(0)*
80	5.00	- *(0)*	- *(0)*	1512*(50)*	6355*(41)*	- *(0)*
90	5.00	- *(0)*	- *(0)*	5167*(39)*	- *(0)*	- *(0)*
100	5.00	- *(0)*	- *(0)*	- *(0)*	- *(0)*	- *(0)*

FIGURE 3 – *Min-2-SAT*

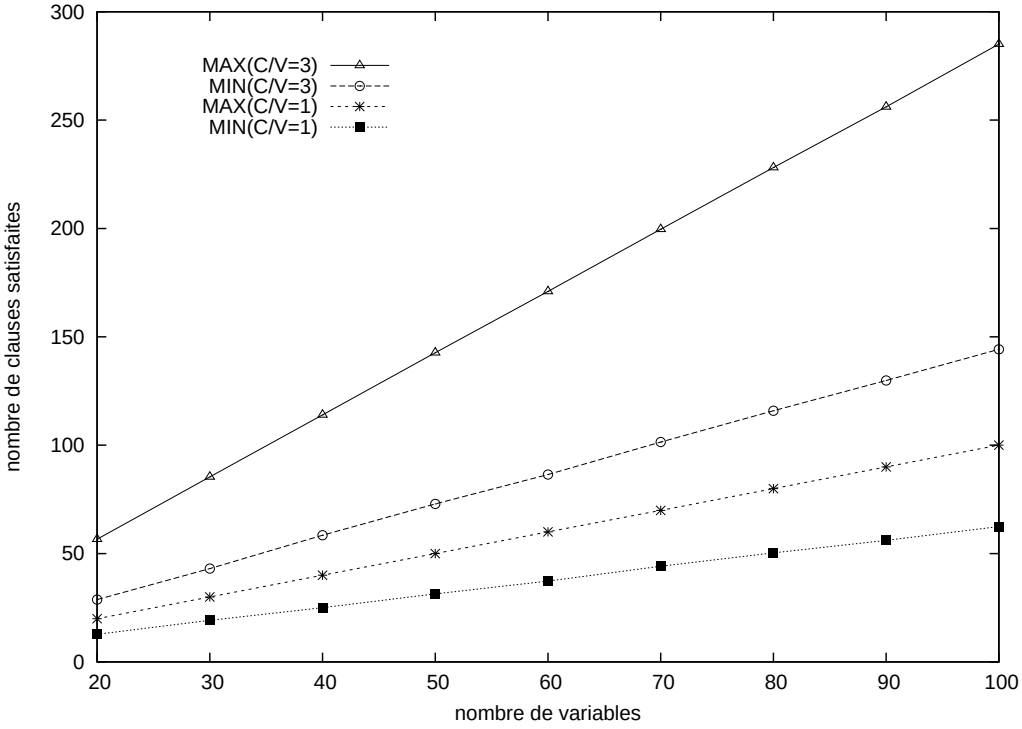
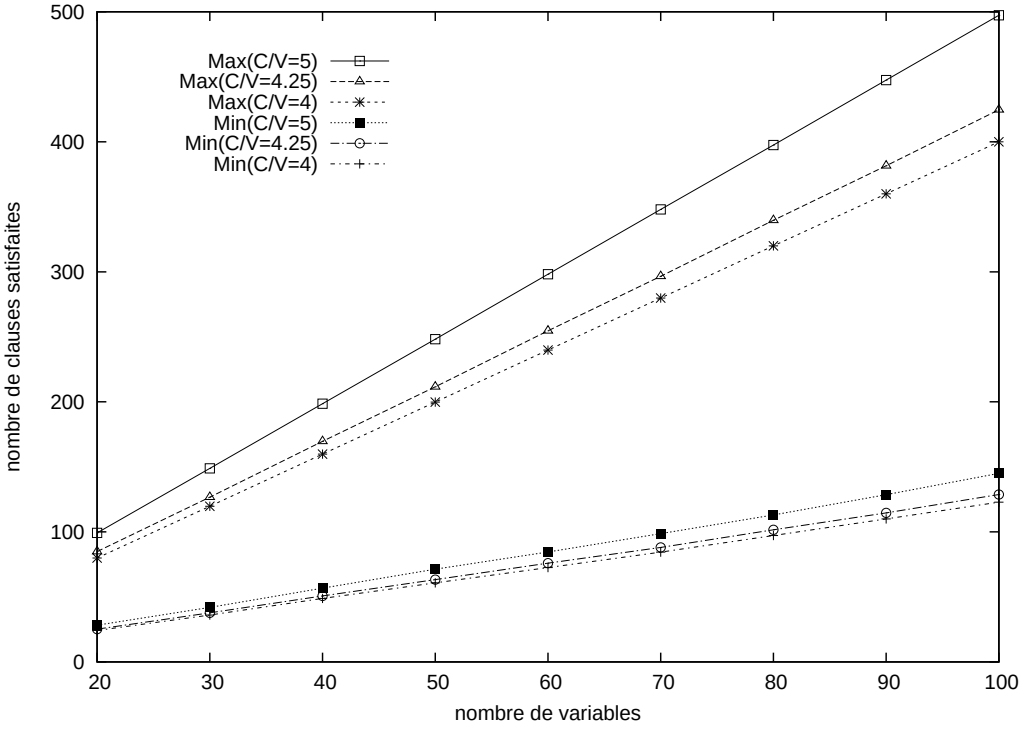


FIGURE 4 – *Min-3-SAT*



chaque solveur, le nombre d'instances résolues en moins de 3 heures (entre parenthèses) sur un ensemble de 50 instances à chaque point, et le temps moyen nécessaire pour résoudre ces instances résolues. Les solveurs MaxSAT qui ne sont pas inclus dans le tableau 2 ont été loin d'être compétitifs sur les instance Min-3-SAT. Les trois codages sont désignés par $E1$ (codage 1), $E2$ (codage 2), et $E3$ (codage 3).

Les résultats expérimentaux montrent que le codage direct de MinSAT en MaxSAT partiel (Codage 1) est meilleur que le codage basé sur MaxClique (Codage 2) pour tous les solveurs MaxSAT. Toutefois, lorsque le codage basé sur MaxClique est amélioré en utilisant une bonne partition en cliques de G_I, le codage basé sur MaxClique (Codage 3) est le plus performant pour tous les solveurs de MaxSAT. Plus important encore, le codage 3 fait MaxSatz significativement meilleur pour trouver une clique maximum dans $\overline{G_I}$ que les solveurs spécifiques de MaxClique comme MaxCliqueDyn et Cliquer. La figure 2 montre que, pour les trois ratios #clause sur #variable de Min-3-SAT, la différence de performance entre MaxSatz (utilisant le codage 3) et MaxCliqueDyn croît avec le problème de la taille. Ce phénomène surprenant mérite une étude future.

Les figures 3 et 4 montrent les nombres maximum et minimum de clauses satisfaites (pour 2-SAT et 3-SAT). Dans ces figures, Min représente le nombre minimum de clauses satisfaites et Max représente le nombre maximum de clauses satisfaites. Nous pouvons voir que si le ratio #clause sur #variable (C/V) augmente, alors l'écart entre Min et Max ne cesse d'augmenter.

6 Conclusions

Nous avons présenté une approche originale générique pour résoudre le problème MinSAT de façon exacte, qui utilise les codages MaxSAT et des solveurs MaxSAT, et avons montré qu'en particulier, l'approche basée sur le réduction de MaxSAT à MaxClique, puis réduire MaxClique à MaxSAT en utilisant le codage amélioré (le codage 3) est meilleure que d'utiliser directement les algorithmes exacts de MaxClique pour résoudre MinSAT, au moins pour les instances testées. En l'absence d'un solveur efficace de séparation et évaluation spécifique pour MinSAT, nous pensons que notre Codage 3 fournit une alternative qui est appropriée et compétitive pour résoudre des problèmes réels de MinSAT, et permet de promouvoir la recherche sur MinSAT.

Ce papier, malgré son caractère préliminaire, contient d'importantes contributions et suggère futures directions de recherche telles que l'élaboration de systèmes d'inférence, la conception des méthodes de calcul de bornes inférieures et supérieures spécifiques pour MinSAT, l'extension de notre approche au formalisme des clauses dures et clauses souples pondérées, et les applications des technologies MinSAT pour résoudre des problèmes industriels.

Remerciements : le problème MinSAT a, à l'origine, été demandé par Laurent Simon au premier auteur.

Références

[1] Carlos Ansótegui, María Luisa Bonet, and Jordi Levy. Solving (weighted) partial MaxSAT through satisfiability testing. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT-2009, Swansea, UK*, pages 427–440. Springer LNCS 5584, 2009.

[2] Adi Avidor and Uri Zwick. Approximating min 2-sat and min 3-sat. *Theory of Computing Systems*, 38(3) :329–345, 2005.

[3] T. Fahle. Simple and fast : Improving a branch-and-bound algorithm for maximum clique. In *Proceedings of ESA-2002*, pages 485–498, 2002.

[4] Giovanni Felici and Klaus Truemper. A minsat approach for learning in logic domains. *INFORMS Journal on Computing*, 14(1) :20–36, 2002.

[5] Avraham Goldstein, Petr Kolman, and Jie Zheng. Minimum common string partition problem : Hardness and approximations. *Electr. J. Comb.*, 12, 2005.

[6] Rajeev Kohli, Ramesh Krishnamurti, and Prakash Mirchandani. The minimum satisfiability problem. *SIAM J. Discrete Mathematics*, 7(2) :275–283, 1994.

[7] J. Konc and D. Janezic. An improved branch and bound algorithm for the maximum clique problem. *Communications in Mathematical and in Computer Chemistry*, 58 :569–590, 2007.

[8] Chu Min Li and F. Manyà. Max-SAT, hard and soft constraints. In Armin Biere, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, pages 613–631. IOS Press, 2009.

[9] Chu Min Li, Felip Manyà, and Jordi Planes. New inference rules for Max-SAT. *Journal of Artificial Intelligence Research*, 30 :321–359, 2007.

[10] Vasco M. Manquinho, João P. Marques Silva, and Jordi Planes. Algorithms for weighted boolean optimization. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT-2009, Swansea, UK*, pages 495–508. Springer LNCS 5584, 2009.

[11] M. V. Marathe and S. S. Ravi. On approximation algorithms for the minimum satisfiability problem. *Information Processing Letters*, 58 :23–29, 1996.

[12] P. R. J. Ostergard. A fast algorithm for the maximum clique problem. In *Discrete Applied Mathematics 120 (2002)*, pages 197–207, 2002.

[13] P. R. J. Ostergard. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, 120 :197–207, 2002.

[14] E. Tomita and T. Seki. An efficient branch-and-bound algorithm for finding a maximum clique. In *Proc. Discrete Mathematics and Theoretical Computer Science, LNCS 2731*, pages 278–289, 2003.