Dénombrement des Données du Problème de Satisfiabilité

Eric HUMBERT

UFR de Mathématiques et Informatique
Université René Descartes
45, rue des Saints-Pères
75270 PARIS Cedex 06
FRANCE

Abstract

The satisfiability problem (SAT) is at the centre of the complexity theory and of the NP-completeness in computer science. It consists in determining if a boolean expression (a SAT instance) can be true (or satisfiable) by assigning its variables to the value true or false. The polynomial resolution of SAT would imply the existence of polynomial algorithms for a great number of hard problems (in graph theory, Artificial Intelligence, etc.). In this paper, we calculate the number of SAT instances by induction on the number of clauses and variables. An instance is characterised by four parameters: the exact number of clauses, the exact number of variables, the exact or the maximal number of literals per clause and finally the exact number of clauses having a maximum number of literals.

Résumé

Le problème de satisfiabilité (SAT) est au coeur de la théorie de la complexité et de la NP- complétude en Informatique. Il consiste à déterminer si une expression booléenne (donnée SAT) peut être vraie (ou satisfiable) par assignation de ses variables à la valeur vraie ou fausse. La résolution polynomiale de SAT signifierait l'existence d'algorithmes de complexité polynomiale pour un grand nombre de problèmes difficiles (en théorie des graphes, Intelligence Artificielle, etc.). Dans ce papier, nous calculons le nombre de données SAT par récurrence sur le nombre de variables et de clauses. Une donnée est caractérisée par quatre paramètres : le nombre exact de clauses, le nombre exact de variables, le nombre exact ou maximal de littéraux par clause et enfin le nombre exact de clauses comportant un maximum de littéraux.

1 Introduction

Le problème booléen de satisfiabilité (SAT) consiste à déterminer si une donnée SAT peut être vraie [3] (satisfiable) . Une donnée SAT est une formule logique sous forme normale conjonctive (conjonction de clauses). Une donnée k-SAT est caractérisée par exactement k littéraux par clause. Cook [1] a montré que le problème de la détermination de la satisfiabilité d'une donnée SAT et plus spécifiquement 3-SAT est NP-complet (complexité exponentielle). Le problème 2-SAT est dans P [2] (complexité polynomiale).
Le problème SAT peut être étudié sur des données restreintes. Ainsi, Papadimitriou [5] a montré que 3-SAT avec au plus cinq occurrences par variable reste NP-complet. Tovey [7] a réduit ce nombre à au plus quatre. La restriction des données permet de mieux connaître la limite qui sépare les problèmes P des problèmes NP. De plus, elle facilite la détermination de nouveaux problèmes NP-complets.
Deux axes de recherche peuvent justifier et prolonger l'étude du dénombrement des données SAT.
Premièrement, étudier le lien qui pourrait exister entre la complexité d'une sous-classe de données SAT (ou restriction SAT) et sa taille (nombre de données). Il est curieux de constater que l'on obtient une bien meilleure complexité au pire (et sans doute en moyenne) par restriction par rapport au nombre maximal d'occurrences par variable (au plus trois : *,3-SAT en $o(1,194^p)$ avec p pour nombre de variables [4]) plutôt que par restriction par rapport au nombre maximal de littéraux par clause (au plus trois : 3-SAT en $o(1,579^p)$ [6]). Qu'en est-il de la différence de taille entre les deux sous-classes ? Un travail préparatoire consiste à dénombrer toutes les données SAT.
Deuxièmement, étudier les données SAT de manière plus exhaustive et moins contestable. En particulier, tester les algorithmes de résolution du problème SAT sur des "échantillons représentatifs", plus exactement sur toutes les données d'une sous-classe, plutôt qu'en faisant appel a des générateurs aléatoires de données. Ou encore, faire une étude précise sur la proportion des données satisfiables et insatisfiables d'une sous-classe. Actuellement, les ordinateurs ne sont pas encore assez puissants pour étudier de manière systématique toutes les données d'une sous-classe (pour un nombre fixé suffisamment grand de variables ou de clauses). Mais il est possible de restreindre considérablement les données SAT à d'autres données équivalentes. Par exemple, par permutation des indices des variables, la satisfiabilité est inchangée. Une étude systématique suppose la réduction du nombre de données à étudier donc en particulier un calcul de dénombrement.
Ce papier n'introduit pas de nouvelles techniques combinatoires. Il met à jour des formules récursives de complexité polynomiale afin d'éviter une énumération systématique de complexité exponentielle. Nous calculons le nombre de données SAT par rapport à quatre paramètres. Le nombre exact p de variables, le nombre exact n de clauses, le nombre exact ou maximal k de littéraux par clause (longueur de la clause) et enfin le nombre exact q de clauses ayant k littéraux.

2 Définition des données SAT

Soit x_i une variable booléenne. Deux complémentations sont possibles pour x_i : la forme directe (littéral positif x_i) ou la forme complémentée (littéral négatif $\neg x_i$). Une clause est une disjonction de littéraux. La variable x_i appartient à la clause Cl si et seulement si l'un des deux littéraux x_i ou bien $\neg x_i$ appartient à Cl. Nous représenterons une conjonction de clauses par un ensemble de clauses. La notation k-SAT(n,p) pour n, p, $k \geq 1$ désigne une conjonction de n clauses. La donnée comporte p variables distinctes notées x_1,x_2,...,x_p. Chaque clause est de longueur k c'est à dire comprend exactement k littéraux. Il n'y a ni ordre ni répétition des variables dans une clause et des clauses dans la donnée. Une donnée k^+-SAT(n,p) est une donnée SAT(n,p) dont la longueur maximale atteinte par les clauses est k. Ce qui suppose qu'au moins une clause a pour longueur k. Les clauses sont non vides. La clause vide est la seule donnée SAT qui ne soit pas de type k^+-SAT. Une donnée k^+-SAT(n,p,q) est de type k^+- SAT(n,p) et comporte exactement (i.e. ni plus ni moins) $q \geq 1$ clauses de longueur k. Notons $S(n,p,k)$ le nombre de données k^+- SAT(n,p) et $S_q(n,p,k)$ le nombre de données k^+-SAT(n,p,q). Le paramètre q peut varier de 1 à n donc :

$$S(n,p,k) = \sum_{q=1}^{n} S_q(n,p,k). \tag{2.1}$$

Le nombre de variables distinctes est majoré par le nombre maximum de littéraux. Le nombre maximal de clauses l-SAT distinctes est $2^l C_p^l$. Nous en déduisons une condition nécessaire d'existence pour une donnée k-SAT(n,p) :

$$\begin{cases} k \geq 1 \\ k \leq p \leq kn \\ 1 \leq n \leq 2^k C_p^k \end{cases} \tag{2.2}$$

Puis, respectivement, pour une donnée k^+- SAT(n,p) et k^+-SAT(n,p,q) :

$$\begin{cases} k \geq 1 \\ k \leq p \leq kn \\ 1 \leq n \leq \sum_{l=1}^{k} 2^l C_p^l \end{cases} \tag{2.3}$$

et

$$\begin{cases} k \geq 1 \\ 1 \leq q \leq n \sum_{l=1}^{k} 2^l C_p^l \\ k \leq p \leq kq + (n-q)(k-1) \end{cases} \tag{2.4}$$

En effet, le nombre de variables est d'au plus $kq+(n-q)(k-1)$ si au plus $n-q$ clauses ont une longueur k-1. $S(n,p,k) = 0$ et $S_q(n,p,k) = 0$ si n, p, k, q ne vérifient pas respectivement (2.3) et (2.4).

3 Calcul de $S(n,p,1)$ et $S(n,kn,k)$

Dans des cas simples, le calcul de $S(n,p,k)$ peut se faire directement comme le montre les deux lemmes qui suivent. Voici quelques valeurs pour $k=1$:

- $S(1,1,1)=2$, ($\{x_1\}$ et $\{\neg x_1\}$).
- $S(2,1,1)=1$, ($\{x_1,\neg x_1\}$).
- $S(2,2,1)=4$, ($\{x_1,x_2\},\{x_1,\neg x_2\},\{\neg x_1,\neg x_2\},\{\neg x_1,x_2\}$).
- $S(3,2,1)=4$, ($\{x_1,\neg x_2\,x_2\},\{\neg x_1,x_1,x_2\},\{\neg x_1,\neg x_2,x_2\},\{x_1,\neg x_2,\neg x_1\}$).
- $S(3,3,1)=8$, ($\{x_1,x_2,x_3\},\{x_1,\neg x_2\,x_3\},\{\neg x_1,x_2,x_3\},\{x_1,x_2,\neg x_3\},\{\neg x_1,\neg x_2,x_3\}$, $\{x_1,\neg x_2,\neg x_3\},\{\neg x_1,x_2,\neg x_3\},\{\neg x_1,\neg x_2,\neg x_3\}$).

Lemme 3.1 $S(n,p,1)=2^{2p-n}C_p^{n-p}$.

Preuve. Aucune clause ne peut comporter moins d'un littéral, d'où $S_q(n,p,1)=0$ si $q<n$, donc $S(n,p,1)=S_n(n,p,1)$. Dans une donnée 1-SAT(n,p), les p variables sont présentes au moins une fois sous la forme de p clauses. Une variable x peut apparaître sous la forme x (forme directe) ou $\neg x$ (forme complémentée). Le nombre de variables présentes sous les deux formes est n-p. Le nombre de variables ayant une seule occurrence est p-(n-p)=2p-n. D'où le lemme 3.1. D'autre part, C_p^{n-p} existe car $n\leq 2p$ (d'après (2.3) pour $k=1$) donc $n-p\leq p$.

Lemme 3.2 $S(n,kn,k)=\frac{2^{kn}\prod_{i=1}^{n}C_{kn-(i-1)k}^{k}}{n!}$.

Preuve. Dans ce cas le nombre de variables est égal au nombre maximal de littéraux. Chaque variable n'est présente qu'une seule fois dans la donnée. Toutes les clauses sont de longueur k. Le nombre de possibilités pour la première clause de la donnée correspond au choix de k variables parmi kn et à leur complémentation, soit C_{kn}^{k} choix possibles de k variables et 2^k complémentations possibles. Pour la deuxième clause : C_{kn-k}^{k} choix possibles de variables et 2^k complémentations possibles. Pour la $i$ème clause : $C_{kn-(i-1)k}^{k}$ choix et 2^k complémentations. L'ordre des n clauses n'est pas pris en compte. Il faut donc diviser par le nombre de permutations $n!$. D'où le lemme 3.2.

Les sections suivantes ont pour but de calculer S(n,p,k) et $S_q(n,p,k)$ pour $k>1$ au moyen de formules de récurrence sur n et p.

4 Calcul de $S_q(n,p,k)$ par récurrence sur n avec $k>1$

Nous allons étudier la construction d'une donnée comportant exactement n clauses à partir de données comportant moins de n clauses. Une approche possible est de décomposer toute donnée k^+-SAT(n,p,q) notée S en deux données S_1 et S_2. S_1 comporte les q clauses de S de longueur k et S_2 les n-q clauses de longueur au plus $k'<k$. Soit $P=\{x_1,x_2,...,x_p\}$ l'ensemble des variables de S, P_1 et P_2 les ensembles de variables de S_1 et S_2. Nous avons $P=P_1\cup P_2$

avec $p = |P|$. Posons $p_1 = |P - P_2|$ et $p_2 = |P - P_1|$. p_1 est le nombre de variables appartenant exclusivement à S_1 et p_2 le nombre de variables appartenant exclusivement à S_2. Le nombre de choix possibles des p_1 et p_2 variables parmi p est $C_p^{p_1} \times C_{p-p_1}^{p_2}$ avec $0 \leq p_1, p_2, p_1 + p_2 \leq p$. S_1 est un ensemble de q clauses de longueur k et comportant $p - p_2$ variables. Le nombre de données S_1 possibles pour p_2 fixé est : $S_q(q, p - p_2, k)$. S_2 est un ensemble de $n - q$ clauses de longueur au plus $k' < k$ et comportant $p - p_1$ variables. Le nombre de données S_2 possibles pour p_1 et k' fixés est $S(n - q, p - p_1, k')$. D'où :

Proposition 4.1 *Si $q < n, k > 1$ et si (n, p, k, q) vérifie (2.4) alors*

$$S_q(n, p, k) = \sum_{\substack{0 \leq p_1, p_2, p_1 + p_2 \leq p \\ 0 < k' < k}} S_q(q, p - p_2, k) \times S(n - q, p - p_1, k').$$

Il nous reste à traiter le cas $n = q$ pour lequel $S_2 = \emptyset$. D'après (2.1) :

$$S_n(n, p, k) = S(n, p, k) - \sum_{q=1}^{n-1} S_q(n, p, k). \tag{4.1}$$

Le calcul de $S(n, p, k)$ est effectué à la section suivante.

5 Calcul de $S(n, p, k)$ par récurrence sur p avec $k > 1$

Examinons la construction d'une donnée comportant exactement $p+1$ variables à partir d'une donnée comportant exactement $p + 1$ variables. Autrement dit, nous voulons introduire la variable x_{p+1} dans une donnée k^+-SAT(n,p) pour en faire une donnée k^+-SAT(n,p+1). L'introduction de la variable x_{p+1} se fait par ajout d'un littéral à une clause de longueur strictement inférieure à k. Une donnée peut comporter les deux clauses x_{p+1} et $\neg x_{p+1}$ alors qu'une clause ne peut pas comporter à la fois les deux littéraux x_{p+1} et $\neg x_{p+1}$ (variables distinctes dans une clause). Notons $Q(n,p,k)$ le nombre de données k^+-SAT(n,p+1) ne comportant ni la clause x_{p+1} ni la clause $\neg x_{p+1}$.

Proposition 5.1 *Si $k > 1$ et si (n, p, k, q) vérifie (2.4) alors*

$$S(n, p+1, k) = Q(n, p, k) + 2Q(n-1, p, k) + Q(n-2, p, k) + 2S(n-1, p, k) + S(n-2, p, k).$$

Preuve. Remarquons d'abord que si n=1 alors seul le premier terme est non nul. Les différents types de données k^+-SAT(n,p+1) sont les suivants : $Q(n,p,k)$ données k^+-SAT(n,p+1) ne comportant ni la clause x_{p+1} ni la clause $\neg x_{p+1}$; $2Q(n-1, p, k)$ données k^+-SAT(n-1,p+1) auxquelles on ajoute la clause x_{p+1} ou bien la clause $\neg x_{p+1}$; $Q(n$-2,$p,k)$ données k^+-SAT(n-2,p+1) auxquelles on ajoute la clause x_{p+1} et la clause $\neg x_{p+1}$; $2S(n-1, p, k)$ données k^+-SAT(n-1,p) auxquelles on ajoute la clause x_{p+1} ou bien $\neg x_{p+1}$ la clause ; $S(n-2, p, k)$ données k^+-SAT(n-2,p) auxquelles on ajoute la clause x_{p+1} et la clause $\neg x_{p+1}$. D'où la proposition 5.1.

Il reste à calculer la valeur de $Q(n,p,k)$. Construisons une donnée T de type k^+- SAT$(n,p+1)$ à partir d'une donnée S de type k^+-SAT(N,p,q) en posant

$$\begin{cases} N = n_1 + n_2 + n_3 \\ n = n_1 + 2n_2 + 3n_3 \end{cases} \tag{5.1}$$

Où $n_i \geq 0$ $(i = 1,2,3)$ représente un nombre de clauses de S présentes i fois dans T et complétées éventuellement par un littéral x_{p+1} ou $\neg x_{p+1}$ pour les rendre distinctes. Les q clauses de longueur k ne peuvent pas être complétées sinon la donnée obtenue ne serait pas k^+-SAT. Les q clauses font donc partie des n_1 clauses non répétées $(0 \leq q \leq n_1)$. Une même clause peut donner naissance à deux clauses distinctes par ajout en fin de clause de x_{p+1} ou $\neg x_{p+1}$. Soit l le nombre d'ajouts dans les $n_1 - q$ clauses de S non répétées dans T. Le nombre de possibilités d'ajouts dépend du choix des l clauses parmi les $n_1 - q$ possibles ($C^l_{n_1-q}$ possibilités) et du choix du littéral ajouté (x_{p+1} ou $\neg x_{p+1}$). Remarquons que l'ajout de la variable x_{p+1} si $n_2 + n_3 = 0$ doit impérativement se faire dans les $n_1 - q$ clauses. La valeur minimale de l vaut donc 0 si $n_2 + n_3 > 0$ et 1 sinon. Il existe trois façons de rendre distinctes deux clauses identiques : compléter une seule clause par x_{p+1}, compléter une seule clause par $\neg x_{p+1}$, compléter les deux clauses à la fois. Le nombre de possibilités d'ajouts dans les n_2 clauses est donc 3^{n_2}. Enfin, il existe une seule façon de différencier trois clauses identiques. Compléter deux d'entre elles, l'une par x_{p+1} et l'autre par $\neg x_{p+1}$. Le nombre de possibilités d'ajouts dans les n_3 clauses est $1^{n_3} = 1$. Soit $W_q(N)$ le nombre de possibilités d'ajouts dans une donnée $k^+ - SAT(N,p,k)$, nous pouvons écrire $W_q(N) = 3^{n_2} \sum_{l=l_{min}}^{n_1-q} 2^l C^l_{n_1-q}$ avec

$$l_{min} = \begin{cases} 0 & \text{si } n_2 + n_3 > 0 \\ 1 & \text{sinon} \end{cases}$$

Pour calculer $Q(n,p,k)$, il faut prendre en compte toutes les valeurs possibles de N donc de (n_1, n_2, n_3) et de q :

Lemme 5.1

$$Q(n,p,k) = \sum_{\substack{n_1,n_2,n_3 \\ vérifiant(5.1)}} \left\{ \sum_{q=q_{min}}^{q_{max}} T_q(N,p,k) \right\}$$

avec $T_q(N,p,k) = S_q(N,p,k) \times W_q(N)$.

$T_q(N,p,k)$ est le nombre de données k–SAT$(n,p+1)$ ne contenant pas les clauses x_{p+1} et $\neg x_{p+1}$ dans q clauses de longueur k. Il nous reste à déterminer q_{max} et q_{min} respectivement majorant et minorant de q. La relation (2.4) nous permet d'écrire $p \leq qk + (N-q)(k-1)$ c'est à dire $p \leq qk + Nk - N - qk + q$ d'où $q \leq p - N(k-1)$. Finalement $q_{min} = p - N(k-1)$. Déterminons q_{max}. L'ajout d'au moins une variable x_{p+1} se fait obligatoirement parmi les n_1 clauses si $n_2 + n_3 = 0$. Il faut donc prévoir dans ce cas au moins une clause de longueur inférieure à k. Les n_2 clauses doublées et les n_3 clauses triplées ne peuvent être de longueur k car cela exclurait la possibilité d'un ajout. Nous en déduisons que

$$q_{max} = \begin{cases} n_1 & \text{si } n_1 + n_2 > 0 \\ n_1 - 1 & \text{sinon} \end{cases}$$

6 Algorithme de dénombrement

Les formules de récurrence qui précèdent permettent de calculer $S_q(n,p,k)$ pour des valeurs croissantes de q, p, n et enfin k. Voici l'algorithme de dénombrement pour des valeurs maximales n_{max} et k_{max} pour n et k.

Algorithme 6.1.

Begin
  For n:= 1 **to** n_{max} **do**
    for p:=1 **to** n **do**
    begin
      $S_n(n,p,1) := 2^{2p-n}C_p^{n-p}$;
      $S(n,p,1) := 2^{2p-n}C_p^{n-p}$
    end;
  For k:=2 **to** k_{max} **do**
    for n:= 1 **to** n_{max} **do**
      for p:=1 **to** kn **do**
        for q:=1 **to** n **do**
          If TEST(n,p,k,q)
          then
          begin
            if n=q
            then
            begin
              $S_n(n,p,k) := SNP(n,p,k)$
              for i:=1 **to** n-1 **do**
                $S_n(n,p,k) := S_n(n,p,k)$-$SNPQ(n,p,k,i)$
            end
            else $S_q(n,p,k) := SNPQ(n,p,k,q)$
          end
          else $S_q(n,p,k) = 0$
End.

La fonction TEST a pour but de vérifier si les variables n, p, k, q vérifient les conditions d'existence (relation (2.4)). Si q=n, $S_n(n,p,k)$ est calculée grâce à la relation (4.1). La fonction SNP calcule $S(n,p,k)$ par la proposition 5.1. La fonction SNPQ calcule $S_q(n,p,k)$ si $q < n$ par la proposition 4.1.

Bibliographie

[1] S. A. Cook, The complexity of theorem- proving procedures, Proc 3rd. Ann. ACM Symp. on Theory of computing (Assoc. Comput. Mach., New York, 1971) 151-158.

[2] S. Even, A. Itai and A. Shamir, On the complexity of timetable and multicommodity flow problems, SIAM J. Comput. 5 (1976) 691-703.

[3] M.R. Garey and D. S. Johnson, Computers and Intractability, A Guide to the Theory of NP-completeness (Freeman and Co., San Francisco) 1979.

[4] E. Humbert, Solving *,3-satisfiability in less than $o(1,194^p)$ steps, soumis à FCT'95.

[5] C. H. Papadimitriou, The Euclidian traveling salesman problem is NP-complete, Theor. Comput. Sci 4 (1977) 237-244.

[6] I. Schiermeyer, Solving 3-satisfiability in less than $o(1,579^p)$, Proc. of CSL'92. LNCS 702, pp. 379-394.

[7] C. A. Tovey, A simplified NP-complete satisfiability problem, Discrete Applied Math. 8 (1984) 85-89.