5.1 Le problème de la satisfiabilité ou SAT

Définition 6 : Le problème de satisfiabilité ou SAT en abrégé est défini comme suit :

Instance : $V = \{v_1, v_2, \ldots, v_n\}$ un ensemble de variables booléennes, $C = \{c_1, c_2, \ldots, c_m\}$ un ensemble de clauses, c_i est une disjonction de littéraux, un littéral est une variable booléenne avec ou sans le connecteur de négation.

Question : Existe-il une instanciation (un ensemble de valeurs booléennes associées aux variables) de V telle que la conjonction des clauses de C est vraie ?

Théorème de Cook (1971) : Le problème de satisfiabilité est NP-complet.

Preuve :

1- SAT appartient à NP ?
2- Tout problème appartenant à NP peut être réduit en SAT via une transformation polynomiale.

Prouvons que SAT appartient à NP. Pour cela il faut trouver un algorithme non déterministe pour SAT. L'algorithme est le suivant :

Début

- Engendrer une instanciation des variables booléennes, en l'occurrence Val = (val1, val2, …, valn)
- Remplacer chaque variable des clauses par sa valeur et calculer sa valeur
- Si la conjonction des valeurs trouvées pour les clauses est égale à 1 alors l'instance est satisfiable sinon elle est contradictoire.

L'algorithme de vérification utilise une matrice m*n appelée SAT pour stocker la donnée SAT. SAT[i,j]=1 si le jème littéral de la ième clause existe sous la forme positive, il est égal à 0 si ce dernier existe sous la forme négative et est égal à -1 s'il n'existe pas. L'algorithme s'écrit comme suit :

Algorithme :

```
début
        i = 1 ; satisfiable = vrai ;
        tant que ( i <= m) et (satisfiable) faire
        début
                satclause = vrai ; j=1 ;
                tant que satclause et j <= n faire
                 si (SAT[i,j] = 0) alors satclause = faux
                                 sinon j = j+1;
                si satclause = faux alors satisfiable = faux
                                 sinon i = i+1 ;
        fin ;
        si (satisfiable) alors imprimer ‘oui’
                    sinon imprimer ‘non’ ;
fin ;
```

La complexité de l'algorithme est en O(n*m) car au pire cas, chaque clause possède m littéraux.