Solveurs SAT

1 Rappels théoriques

1.1 Problème $\mathbb{SAT}$

Le problème de *satisfiabilité* de formules propositionnelles (ou simplement de satisfiabilité, noté $\mathbb{SAT}$) est défini comme suit :

Définition 1 (Problème $\mathbb{SAT}$) *Etant donnée une formule propositionnelle quelconque ϕ définie sur l'ensemble des variables propositionnelles $p_1, p_2, \ldots, p_n$. Est-ce que ϕ est satisfiable ? (c à d : est-ce qu'il existe au moins une interprétation I qui, pour chaque variable propositionnelle p_i elle affecte une valeur $I(p_i) = vrai$ ou $I(p_i) = faux$ telle que $I(\phi) = vrai$).*

D'après la définition, il est clair que $\mathbb{SAT}$ est un problème de décision. Par ailleurs, on peut donner une verison *constructive* du problème comme suit :

Définition 2 (Problème $\mathbb{SAT} - \mathbb{UNSAT}$) *Etant donnée une formule propositionnelle quelconque ϕ définie sur l'ensemble des variables propositionnelles $p_1, p_2, \ldots, p_n$. Trouver une interprétation I qui satisfait ϕ si elle existe, ou prouver qu'elle n'existe pas, sinon.*

Le problème $\mathbb{SAT}$ est très simple à décrire mais très délicat à résoudre à cause de l'explosion combinatoire dû à l'évaluation des différentes variables. Si l'on a n variables propositionnelles, au pire des cas, on doit évaluer 2^n cas possibles pour décider si la formule ϕ est satisfiable ou non.

Du point de vue théorique, $\mathbb{SAT}$ est très imporant dans l'évaluation de la complexité des problèmes. D'ailleurs, c'est le premier est l'unique problème démontré directement qu'il est $\mathsf{NP}-$Complet[1], c à d, jusqu'au jour d'aujourd'hui, on ne connait pas un algorithme qui résoud $\mathbb{SAT}$ dans une temps polynomial (c à d : un temps qui s'exprime sous forme d'un polynôme en fonction du nombre de variables).

En outre, on peut très facilement concevoir un algorithme *exponentiel* pour $\mathbb{SAT}$. Il est donné comme suit :

{TRUE, FALSE} SAT(ϕ : formule)
if ($\phi == \top$) **then**
  return TRUE ;
else if ($\phi == \bot$) **then**
  return FALSE ;
else
  p=choisir_variable(ϕ) ;
  ψ=remplacer($\phi, p, \top$) ;
  γ=remplacer($\phi, p, \bot$) ;
  SAT(ψ) ;
  SAT(γ) ;
end if

Dans cette procédure, les tests ($\phi ==$ TRUE) et ($\phi ==$ FALSE) signfient que la formule ϕ est évaluée et réduite, si possible, à une constante $\top$ ou $\bot$ que l'on peut comparée à TRUE ou FALSE. Dans le cas contraire, la fonction choisir_variable() sélectionne une variable propositionnelle *quelconque* p, ensuite deux formules ψ et γ sont construites selon la valeur donnée à p dans ϕ : $\top$ ou $\bot$, respectivement. Deux appels récursifs qui prennent comme arguments ψ et γ sont lancés.

1. sans utiliser le théorème de Cook

1.2 Méthode de résolution

Il est établit que toute formule propositionnelle ϕ peut être transformée (dans un *temps polynomial*) en un formule ψ sous *Forme Normale Conjonctive* (FNC).

Définition 3 (Equisatisfiabilité) *Deux formules propositionnelles ϕ et ψ sont équisatisfiables si : ϕ est satisfiable ssi ψ l'est aussi.*

Définition 4 (FNC) *La forme normale conjonctive d'une formule ϕ est la formule ψ équisatisfiable de ϕ, et qui s'écrit sous la forme*

$$\bigwedge_i (\bigvee_j l_{ij})$$

Où l_{ij} sont des littéraux.

Définition 5 (Litéral) *Un litéral est une variable propositionnelle p ou sa négation $\neg p$.*

Définition 6 (Clause) *Une clause est une formule qui s'écrit sous la forme $\bigvee_j l_{ij}$, où l_{ij} sont des litéraux.*

La méthode de résolution dû à Robinson (1965) utilise une seule règle d'inférence sur des formules écrites en FNC pour prouver par *réfutation* qu'une formule ϕ est une conséquence logique d'une formule ψ en FNC. La procédure est décrite comme suit :

1. Ecrire $\neg\phi$ en FNC ;
2. Construire l'ensemble des clauses $\mathbb{C} = \{C_1, C_2, \ldots, C_n\}$ à partir de $\{\psi, \neg\phi\}$;
3. Appliquer la règle de résolution sur deux clauses C_i, C_j choisies arbitrairement, puis rajouter la clause résolvante C_k à $\mathbb{C}$.

1.3 Algorithme DPLL

Cette méthode est dû à leur autheur(s) : Davis, Putnam, Loveland et Logemann. Elle est basée sur la *recherche systématique* : procédure récursive qui consiste en deux règles :

1. **Règle de la clause unité :** si l'on a une clause $C \equiv l_1 \vee l_2 \vee \cdots \vee l_n$ où tous les litéraux sont évalués à "faux" sauf un, noté l_i, alors C sera réduite à l_i qui prendra la valeur "vrai".
   Exemple. Nous avons la formule $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_5) \wedge (x_2 \vee \neg x_5)$, transformée en 5 clauses : $C_1 \equiv x_1 \vee \neg x_2, C_2 \equiv \neg x_1 \vee x_3 \vee x_4, C_3 \equiv \neg x_2 \vee \neg x_4, C_4 \equiv \neg x_1 \vee x_5$ et $C_5 \equiv x_2 \vee \neg x_5$. Si on donne la valuation "vrai" à x_1, alors la clause $C1$ sera réduite à la clause unité $\top \vee \neg x_2 \equiv \neg x_2$, donc x_2 aura la valuation "faux". De même, la clause $C3$ sera réduite à un seul litéral $\neg x_4$, et x_4 au la valuation "faux". La même règle est appliquéée à la clause C_4 qui sera réduite au litéral x_5 qui prendra la valeur "vrai".
2. **Règle des conflits :** La clause C_5 sera évaluée à faux car x_2 est à "faux" et x_5 est à "vrai". Cette clause est conflictuelle, et nécessite un retour arrière (*backtracking*) pour résoudre le conflit

Des améliorations à la procédure de base ont été introduites, notamment *Conflict-Driven Learning* et *Non-Chronological Backtracking*

1.4 Solveurs $\mathbb{SAT}$

Nous utilisons les solveurs Open Source Cadical. Ce solveur accepte des fichiers d'entrée de format DIMACS :

```
c commentaire
p cnf nb_variables nb_clauses
-1 2 -3 0
4 5 0
```

La première ligne est un commentaire, précédée par `c`. La deuxième ligne signfie le début de la description sous forme CNF précédée par `p` : `nb_variables` est le nombre de variables et `nb_clauses` est le nombre de clause. Ensuite viennent les clauses : chaque clause se termine par 0 et comprend des nombres dans [`-nb_variables`...`+nb_variables`]. Les nombres négatifs correspondent aux litéraux négatifs et les nombres positifs correspondent aux litéraux positifs. Une clause n'accepte pas une nombre et sa négation en même temps.

Exemple. La formules $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_5)$ est représentée par l'entrée suivante :

```
p cnf 5 4
1 -2 0
-1 3 4 0
-2 -4 0
-1 5
c CTRL-D
```

2 Jeu Sodoku

2.1 Principe du jeu

Sodoku est un jeu de grille composée de 9×9 cellules, chacune doit contenir un chiffre décimal compris entre 1 et 9. Le but du jeu est de remplire la grille par des chiffres décimaux (1-9) de telle sorte que :

1. Chaque ligne de la grille doit contenir tous les chiffres 1-9.
2. Chaque colonne de la grille doit contenir tous les chiffres 1-9.
3. Chaque carré composé de 3×3 cellules (on retient uniquement les 8 carrés de bordures et le carré du centre) doit contenir tous les chiffres 1-9.

Certaines cellules sont pré-remplies comme le montre la figure ci-dessous.

	6		7			1	5	
		3	9			8		
		2	3				4	9
		7			4			
	4			9			8	
			1			4		
6	7				9	3		
		9			2	5		
	2	8			7		6	

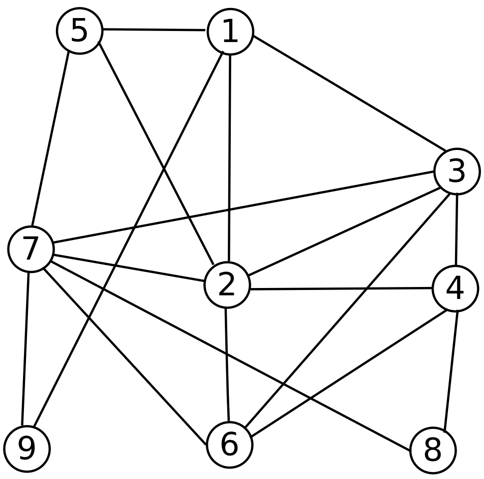
1. Implémenter et résoudre le jeu Sodoku dans Cadical.
2. Est-ce qu'il admet une solution ? laquelle ?

3 Coloriage de graphes

Le problème de coloriage de graphes est énoncé comme suit :

Définition 7 (Graphe $c-$Colorable) *Soit un ensemble de couleurs C tels que $|C| = c$. Etant donné un graphe $G = (V, E)$, où V est l'ensemble des sommets et E est l'ensemble des arrêtes. Trouver une affectation de couleur parmi l'ensemble C à chauque sommet de G telle que chauqe deux sommets adjecents ne soient pas colorés dans la même couleur.*

Définition 8 (Coloriage de graphe) *Etant donné un graphe $G = (V, E)$, où V est l'ensemble des sommets et E est l'ensemble des arrêtes. Trouver la valeur minimale de c telle que G soit $c-$Colorable.*



1. Modéliser le problème $4 - colorable$ pour la graphe donné ci-dessus.
2. Implémenter le modèle dans Cadical et donner la solution.
3. Quelle est la valeur minimale de m pour que ce graphe soit $m-$colorable.