UNIVERSITÉ GRENOBLE ALPES **M1-info**

COMPLEXITÉ **2016-17**

Examen du 2 mai 2017 — corrigé

On rappelle les notions de base de la logique propositionnelle concernant les clauses.

- Soit un vocabulaire V constitué de v **variables propositionnelles**, $V = \{ p_1, \ldots, p_v \}$. Une **interprétation** I de V est une application (fonction totale) de V dans les "valeurs de vérité" : elle associe à chaque p_i la valeur $Vrai$ (aussi notée 1) ou $Faux$ (aussi notée 0).
- Un **littéral** h est soit une variable propositionnelle p de V, soit la négation $\neg p$ d'une variable propositionnelle p de V.
  Littéraux conjugués : deux littéraux h et h' sont dits *conjugués* l'un de l'autre s'il existe une variable propositionnelle p telle que $h = p$ et $h' = \neg p$ ou $h = \neg p$ et $h' = p$.
  L'interprétation I d'un littéral h est naturellement $I(h) = I(p)$ si $h = p$ et $I(h) = \neg I(p)$ si $h = \neg p$:
  - Un littéral positif p_i est évalué à $Vrai$ dans I si $I(p_i) = Vrai$, à $Faux$ sinon.
  - Un littéral négatif $\neg p_i$ est évalué à $Vrai$ dans I si $I(p_i) = Faux$, à $Vrai$ sinon.
- Une **clause** est une disjonction (finie et sans répétition) de littéraux : $c = h_1 \vee \ldots h_k$. La *taille* de c est k.
  Une clause peut être restreinte à un seul littéral (clause "unaire"), ou à au plus 2 littéraux (clause "binaire").
- Une **conjonction de clauses** est une formule (finie) $C = c_1 \ldots c_n$ où les c_i sont des clauses. Elle peut être restreinte à une seule clause.
  La taille d'une conjonction de clauses est la somme des tailles des clauses présentes dans la conjonction.
- L'évaluation d'une clause dans une interprétation I renvoie $Vrai$ si *au moins* un de ses littéraux est évalué à $Vrai$ dans I et $Faux$ sinon.
- L'évaluation d'une conjonction de clauses dans une interprétation I de ses variables propositionnelles renvoie $Vrai$ si *toutes* les clauses sont évaluées à $Vrai$ dans I, et $Faux$ sinon.
- **Satisfaisabilité :** Une conjonction de clauses est satisfaisable s'il existe une interprétation I de ses variables propositionnelles dans laquelle toutes les clauses sont évaluées à $Vrai$.
  SAT et 3-SAT :
  Le problème **SAT** est le test de satisfaisabilité d'une conjonction de clauses quelconques.
  Le problème **3-SAT** est le test de satisfaisabilité des conjonctions de clauses comportant exactement 3 littéraux par clauses.

Réduction de 3-SAT vers STABLE

Le problème STABLE se définit de la façon suivante : étant donnés un graphe non orienté $G = (V, E)$ et un entier $k \leq |V|$, déterminer si le graphe contient un *ensemble stable* de taille supérieure ou égale à k.

Un ensemble de sommets $V' \subseteq V$ est un ensemble *stable* si deux sommets quelconques de V' ne sont pas joints par une arête : pour tout $u \in V'$ et pour tout $v \in V'$, $(u, v) \notin E$.

1. Donner un exemple de graphe à 4 sommets contenant un ensemble stable de taille 3. Donner ensuite un exemple d'un graphe à 4 sommets ne contenant pas d'ensemble stable de taille 3.
   ► L'ensemble {1,2,3} est stable pour le graphe dont les arêtes sont : {(1,4), (2,4, (3,4)}.
   Le graphe suivant ne contient pas d'ensemble stable de taille 3 : {(1,4), (2,4, (3,4), (1,2)}. ◄

2. On convient de coder un graphe G à n sommets sous la forme d'un tableau E de dimension $n \times n$ tel que $E[i, j] = 1$ s'il existe une arête entre le nœud numéro i et le nœud numéro j, et $E[i, j] = 0$ sinon. On convient également de coder un sous-ensemble $\{i_1, \ldots, i_k\}$ de k numéros de sommets de G par un tableau `sousEns` de taille k tel que `sousEns`$[j] = i_j$ pour tout $j \in [1..k]$.
   Ecrire un programme pseudo-Pascal qui prend en entrée le codage E d'un graphe G à n sommets et le codage `sousEns` d'un sous-ensemble $\{i_1, \ldots, i_k\}$ de $[1 \ldots n]$, et qui vérifie si $\{i_1, \ldots, i_k\}$ est un ensemble stable de G.
   ► On suppose le sous-ensemble codé par un tableau `sousEns[1..k]`.

```
stable := 1 ;
Pour j de 1à k
    Pour l de j à k
        si E[sousEns[j], sousEns[l]] = 1 alors stable := 0
retourner(stable).
```

   ◄

3. Donner la complexité de l'algorithme précédent et en déduire que STABLE est dans NP.

   ► L'algorithme pr'ecédent teste si un sous-ensemble donné de taille k est un ensemble stable en $\Theta(k^2)$ donc en $O(n^2)$. Il est donc de complexité polynomiale. L'algorithme non déterministe qui choisit un ensemble de sommets V' de taille k et qui vérifie s'il est stable (par l'algorithme précédent) est non déterministe polynomial, ce qui prouve l'appartenance du problème à NP. ◄

4. Rappelons qu'une instance de 3-SAT est une conjonction de clauses telle que chaque clause est une disjonction $x_{i_1} \vee x_{i_2} \vee x_{i_3}$ d'exactement 3 littéraux, tous distincts 2 à 2. Par exemple, $(\neg p_1 \vee p_2 \vee p_3) \wedge (p_1 \vee p_2 \vee \neg p_3)$ est une instance de 3-SAT constituée de 2 clauses et construite à partir de 3 variables propositionnelles : p_1, p_2 et p_3. Elle est de taille 6.
   On considère la transformation suivante d'une instance de 3-SAT en un graphe :
   — toute clause $x_{i_1} \vee x_{i_2} \vee x_{i_3}$ est transformée en un triangle (graphe complet à 3 sommets) dont les sommets sont étiquetés par les littéraux x_{i_1}, x_{i_2} et x_{i_3} de la clause ;
   — une arête supplémentaire est ajoutée entre deux sommets s'ils sont étiquetés par deux littéraux conjugués.
   Dessiner le graphe (à 6 sommets) résultant de la transformation de la conjonction de clauses $(\neg p_1 \vee p_2 \vee p_3) \wedge (p_1 \vee p_2 \vee \neg p_3)$
   ► Triangle entre 3 noeuds étiquetés respectivement par $\neg p_1$, p_2 et p_3, un autre triangle entre 3 autres noeuds étiquetés respectivement par p_1, p_2 et $\neg p_3$, une arête entre le noeud du premier triangle étiqueté par $\neg p_1$ et le noeud du second triangle étiqueté par p_1, et une arête entre le noeud du premier triangle étiqueté par p_3 et le noeud du second triangle étiqueté par $\neg p_3$ ◄

5. Donner l'ordre de grandeur de la complexité de la construction précédente en fonction de la taille n de l'instance 3-SAT à transformer.

► Il s'agit de scanner la conjonction de clauses fournie en entrée, clause par clause, et pour chaque clause créer un triangle : on obtient un graphe de n sommets. Puis pour chaque sommet, scanner l'ensemble des sommets à la recherche d'étiquettes conjugées, et si c'est le cas de rajouter une arête. Donc $0(n^2)$. ◄

6. Soit F une instance de 3-SAT qui est une conjonction de c clauses et G le graphe résultat de la construction précédente à partir de F.
   Montrer que si F est satisfaisable alors G contient un ensemble stable de taille c.

   ► Si F est satisfaisable, il existe une interprétation I telle que, pour chacune des c clauses, au moins un des littéraux est évalué à Vrai. Choisissons un de ces littéraux par clause et considérons l'ensemble des o=c sommets du graphe correspondants. Ils forment un stable : aucun de ces sommets ne fait partie d'un même triangle par construction ; il ne peut pas y avoir d'arête supplémentaire entre eux car ces arêtes ne relient que des littéraux conjugués et il ne peut y avoir deux littéraux conjugués évalués à Vrai dans une interprtation donnée. ◄

7. Montrer que si G contient un ensemble stable de taille c alors F est satisfaisable.
   ► Soit S un ensemble stable de taille $\geq c$ de sommets de G. Au plus un sommet par triangle est dans S. Donc S est de taille exactement c et contient exactement un sommet de chaque triangle. Comme S est stable, il ne contient pas de paires de sommets étiquetés par des littéraux conjugués, qui par construction sont reliés. On peut donc définir l'interprétation I qui rend Vrai chacun des littéraux correspondants aux étiquettes des sommets de S : I rend Vrai F puisque chacune de ses clauses contient un littéral qui est Vrai dans I. ◄

8. On rappelle que le problème 3-SAT est NP-complet. En s'appuyant sur certaines des questions précédentes que vous préciserez, montrer que le problème STABLE est NP-complet.

   ► Les questions précédentes montrent que la transformation polynomiale (question 5), qui associe à chaque instance F de 3-SAT de c clauses une instance (G, c) est telle que F est satisfaisable si (question 7) et seulement si (question 6) G contient un ensemble stabe de taille $\geq c$. Il s'agit donc d'une réduction de 3-SAT vers STABLE, et donc STABLE est NP-dur. Il est dans NP (cf question 3). Donc STABLE est NP-complet. ◄

Réduction du problème de Sudoku vers SAT

Le problème *Sudoku* consiste à placer les nombres $1 \dots n^2$ sur une grille $n^2 \times n^2$ (pour le sudoku classique n = 3) dont certaines valeurs sont fixées. La figure 1 est un exemple d'une instance positive du problème de Sudoku classique : partant de la grille initiale partiellement remplie, il existe une solution, c'est-à-dire une façon de compléter les cases vides avec des nombres de 1 à 9 de sorte que les nombres d'une même ligne, ainsi que d'une même colonne et d'une même sous-grille 3×3 soient tous différents.

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

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

FIGURE 1 – Une instance du problème de Sudoku (gauche) pour laquelle il existe une solution (droite).

Pour un n donné, une instance du problème Sudoku sera codée par un tableau T de taille $n^2 \times n^2$ représentant la grille initiale : $T[i, j]$ représentera le contenu de la case à l'intersection de la ligne numéro i ($1 \leq i \leq n^2$) et de la colonne numéro j ($1 \leq j \leq n^2$), en représentant un contenu vide par 0. Dans l'exemple de la figure 1, $T[1, 1] = 5$, $T[1, 2] = 3$, $T[1, 3] = 0$.
La grille initiale est partitionnée en n^2 sous-grilles disjointes, chacune de taille $n \times n$. Chaque sous-grille est identifiée par la case (l,c) qui en est son coin supérieur gauche. Ainsi, la sous-grille (l,c) correspond au sous-tableau :

T[l,c]	...	T[l,c+j]	...	T[l,c+n-1]
...	...	...	...	...
T[l+i,c]	...	T[l+i,c+j]	...	T[l+i,c+n-1]
...	...	...	...	...
T[l+n-1,c]	...	T[l+n-1,c+j]	...	T[l+n-1, c+n-1]

Une solution doit vérifier les contraintes suivantes :

- Toutes les cases de la grille doivent contenir un et un seul nombre compris entre 1 et n^2 inclus.
- Les nombres sur une même ligne sont tous différents.
- Les nombres sur une même colonne sont tous différents.
- Les nombres dans une même sous-grille sont tous différents.

1. Pour la grille solution de la figure 1, indiquer le contenu du sous-tableau correspondant à la sous-grille dont le coin supérieur gauche est la case (4,7).

   ► | | | |
   |---|---|---|
   | 4 | 2 | 3 |
   | 7 | 9 | 1 |
   | 8 | 5 | 6 |

   ◄

2. Pour la transformation d'une instance du problème Sudoku codée par un tableau T (de taille $n^2 \times n^2$), on introduit autant de variables propositionnelles $p^v_{i,j}$ qu'il y a de cases (i, j) dans le tableau et de nombres v possibles à mettre dans chaque case. L'interprétation à $Vrai$ de $p^v_{i,j}$ signifiera que $T[i, j] = v$, et à $Faux$ que $T[i, j] \neq v$.

(a) Donner en fonction de n le nombre de variables propositionnelles $p^v_{i,j}$ ainsi introduites.
► n^6, car les indices i , j et v varient chacun de 1 à n^2. ◄

(b) Dans le cas $n = 3$, traduire par une conjonction de clauses unaires que $T[1, 2] = 3$ et que 3 est le seul chiffre de 1 à 9 que l'on peut placer dans la case $(1, 2)$.
► $p^3_{1,2} \wedge \neg p^1_{1,2} \wedge \neg p^2_{1,2} \wedge \neg p^4_{1,2} \wedge \neg p^5_{1,2} \wedge \neg p^6_{1,2} \wedge \neg p^7_{1,2} \wedge \neg p^8_{1,2} \wedge \neg p^9_{1,2}$ ◄

3. Dans le cas général, traduire par une conjonction $Contient(i, j, v)$ de clauses unaires le fait qu'une case (i, j) contient le nombre v, et ne peut contenir aucun autre nombre v' de $[1..n^2]$. On pourra utiliser la notation $\bigwedge^{v' \neq v}_{v' \in [1..n^2]} l^{v'}$ pour représenter une conjonction de littéraux indicés par v' où v' prend les valeurs entre 1 et n^2, sauf la valeur v.

► $Contient(i, j, v) : p^v_{i,j} \wedge \bigwedge^{v' \neq v}_{v' \in [1..n^2]} \neg p^{v'}_{i,j}$
◄

4. On s'intéresse maintenant à traduire par une conjonction de clauses les contraintes portant sur une case vide (i, j) dans la grille initiale, et qu'il faut donc remplir pour obtenir une solution.

(a) Traduire par une clause $C(i, j)$ de la forme $\bigvee_{v \in [1..n^2]} l^v$ que la case (i,j) doit être remplie par un nombre compris entre 1 et n^2.
► $C(i, j) : \bigvee_{v \in [1..n^2]} p^v_{i,j}$ ◄

(b) Traduire par une conjonction $AllDiff(i, j)$ de clauses binaires que la case (i, j) ne doit pas être remplie par deux nombres différents.

► Il faut exprimer que, pour toute valeur v et toute valeur v' distincte de v, $p^v_{i,j} \Rightarrow \neg p^{v'}_{i,j}$ (sous forme de clause : $\neg p^v_{i,j} \vee \neg p^{v'}_{i,j}$).
$AllDiff(i, j) : \bigwedge^{v<v'}_{v,v' \in [1..n^2]} (\neg p^v_{i,j} \vee \neg p^{v'}_{i,j})$ ◄

5. On s'intéresse à présent à traduire par une conjonction de clauses les contraintes portant sur une ligne i. Traduire par une conjonction $Ligne(i)$ de clauses binaires que pour tout nombre $v \in [1..n^2]$, il ne peut y avoir deux cases distinctes (i, j) et (i, j') de la ligne numéro i contenant v.

► $Ligne(i) : \bigwedge^{v \in [1..n^2]}_{j \neq j' \in [1..n^2]} (\neg p^v_{i,j} \vee \neg p^v_{i,j'})$ ◄

6. Traduire par une conjonction $Colonne(j)$ de clauses binaires que pour tout nombre $v \in [1..n^2]$, il ne peut y avoir deux cases distinctes (i, j) et (i', j) de la colonne numéro j contenant v.

► $Colonne(j) : \bigwedge^{v \in [1..n^2]}_{i \neq i' \in [1..n^2]} (\neg p^v_{i,j} \vee \neg p^v_{i',j})$ ◄

7. Traduire par une conjonction $SousGrille(l, c)$ de clauses binaires que pour tout nombre $v \in [1..n^2]$, il ne peut y avoir deux cases distinctes (i, j) et (i', j') contenant v dans la sous-grille dont le coin supérieur gauche est la case (l, c) .

► $SousGrille(l, c) : \bigwedge^{v \in [1..n^2]} \bigwedge^{l \leq i, i' \leq (l+n-1), i \neq i'}_{c \leq j, j' \leq (c+n-1), j \neq j'} (\neg p^v_{i,j} \vee \neg p^v_{i',j'})$ ◄

8. Après avoir rappelé la signification de la notation $B \leq_p B'$, justifier succinctement pourquoi les questions précédentes permettent de montrer que $Sudoku \leq_p SAT$?
► $B \leq_p B'$, où B et B' sont des problèmes de décision, signifie qu'il existe une fonction de complexité polynomiale de l'ensemble X des instances du problème B vers l'ensemble Y des instances du problème B' telle que : pour toute instance $x \in X$, B(x) est vrai si et seulement si B'(f(x)) est vrai.
Soit la fonction f, qui transforme une grille S de Sudoku partiellement remplie en la conjonction de clauses constituée :
- des conjonctions de clauses $Contient(i, j, v)$ pour toutes les cases (i,j) remplies par une valeur v dans la grille initiale (il y a au plus n^4 telles conjonction de clauses, chacune étant de taille n^2)

- des conjonctions de clauses $C(i,j) \wedge AllDiff(i,j)$ pour toutes les cases (i,j) vides dans la grille initiale (il y a au plus n^4 telles conjonctions de clauses, chacune de taille $n^2 + n^2$)

- des conjonctions de clauses $Ligne(i)$ (au plus n^2 telles conjonctions de clauses de taille $2 \times n^2 \times (n^2 - 1)$).

- des conjonctions de clauses $Colonne(j)$ (au plus n^2 telles conjonctions de clauses de taille $2 \times n^2 \times (n^2 - 1)$).

- des conjonctions de clauses $SousGrille(l,c)$ (au plus n^2 telles conjonctions de clauses de taille $2 \times n^2 \times (n^2 - 1)$).

cette fonction f est :

- de complexité polynomiale (par rapport à la taille n^4 de la grille initiale

- telle que : il existe une solution à partir de S vérifiant les contraintes de Sudoku si et seulement il existe une interprétation I qui sartisfait f(S).

◄

9. Que peut-on en déduire sur la classe de complexité de *Sudoku* ?

► On peut juste dire qu'on peut résoudre Sudoku à l'aide d'un algorithme qui résout SAT, pour lequel on connait un algorithme non déterministe de complexité polynomiale, et donc que Sudoku est dans NP. Attention, on ne peut absolumlent pas en déduire que Sudoku est NP-complet. D'ailleurs, l'existence d'un algorithme de complexité polynomiale pour résoudre Sudoku a été montrée très récemment (vérifier l'information). ◄