

## Compte Rendu du TP 2 - Complexité

### Exercice 1.

- Le fichier **exo1a.cnf** est en pièce jointe.
- la satisfaisabilité de la formule avec Minisat :

```
redouane@redouane-350V5C-351V5C-3540VC-3440VC:~/Master1/Complexite/TP2$ minisat exo1a.cnf
WARNING: for repeatability, setting FPU to use double precision
===== [ Problem Statistics ] =====
|
| Number of variables:          4
| Number of clauses:           3
| Parse time:                  0.00 s
| Eliminated clauses:          0.00 Mb
| Simplification time:         0.00 s
|
===== [ Search Statistics ] =====
| Conflicts | ORIGINAL | LEARNT | Progress |
| Vars  | Clauses | Literals | Limit  | Clauses | Lit/Cl |
=====
restarts      : 1
conflicts     : 0 (-nan /sec)
decisions     : 1 (0.00 % random) (inf /sec)
propagations  : 0 (-nan /sec)
conflict literals : 0 (-nan % deleted)
Memory used   : 22.00 MB
CPU time      : 0 s

SATISFIABLE
```

- La formule :  $\varphi = (\neg t \rightarrow \neg s) \rightarrow (((b \vee t) \rightarrow s) \wedge ((r \wedge m) \rightarrow (b \vee a)) \wedge \neg r)$ 
  - FNC :  $\varphi = (\neg t \vee s) \wedge (\neg t \vee \neg r) \wedge (\neg b \vee s) \wedge (\neg r \vee s)$
  - Le fichier **exo1c.cnf** est en pièce jointe.
  - la satisfaisabilité de la formule avec Minisat :

```
redouane@redouane-350V5C-351V5C-3540VC-3440VC:~/Master1/Complexite/TP2$ minisat exo1c.cnf
WARNING: for repeatability, setting FPU to use double precision
===== [ Problem Statistics ] =====
|
| Number of variables:          4
| Number of clauses:           4
| Parse time:                  0.00 s
| Eliminated clauses:          0.00 Mb
| Simplification time:         0.00 s
|
===== [ Search Statistics ] =====
| Conflicts | ORIGINAL | LEARNT | Progress |
| Vars  | Clauses | Literals | Limit  | Clauses | Lit/Cl |
=====
restarts      : 1
conflicts     : 0 (-nan /sec)
decisions     : 1 (0.00 % random) (inf /sec)
propagations  : 0 (-nan /sec)
conflict literals : 0 (-nan % deleted)
Memory used   : 22.00 MB
CPU time      : 0 s

SATISFIABLE
```

En ajoutant une clause unitaire positive (qui était négative dans le premier résultat) , on confirme qu'il existe au moins deux valuations satisfaisantes.

(d) Algorithme en pseudo Code pour vérifier si une formule est une tautologie :

```
isTautologie( $\varphi$ )
{
    n = CalculerNbrVariables( $\varphi$ )
    m = CalculerNbrClauses( $\varphi$ )
    Pour i = 1 à  $2^n$  // Création de la table de vérité de la formule  $\varphi$ 
    |   Pour j = 1 à n+m
    |   |   M[i,j] = CalculerValeurClauses(M[i,j])
    |   FinPour
    FinPour
    tauto = VRAI
    Pour i = 1 à  $2^n$ 
    |   Si( M[i,n+m] == FAUX )
    |   |   Alors tauto = FAUX
    FinPour
}
```

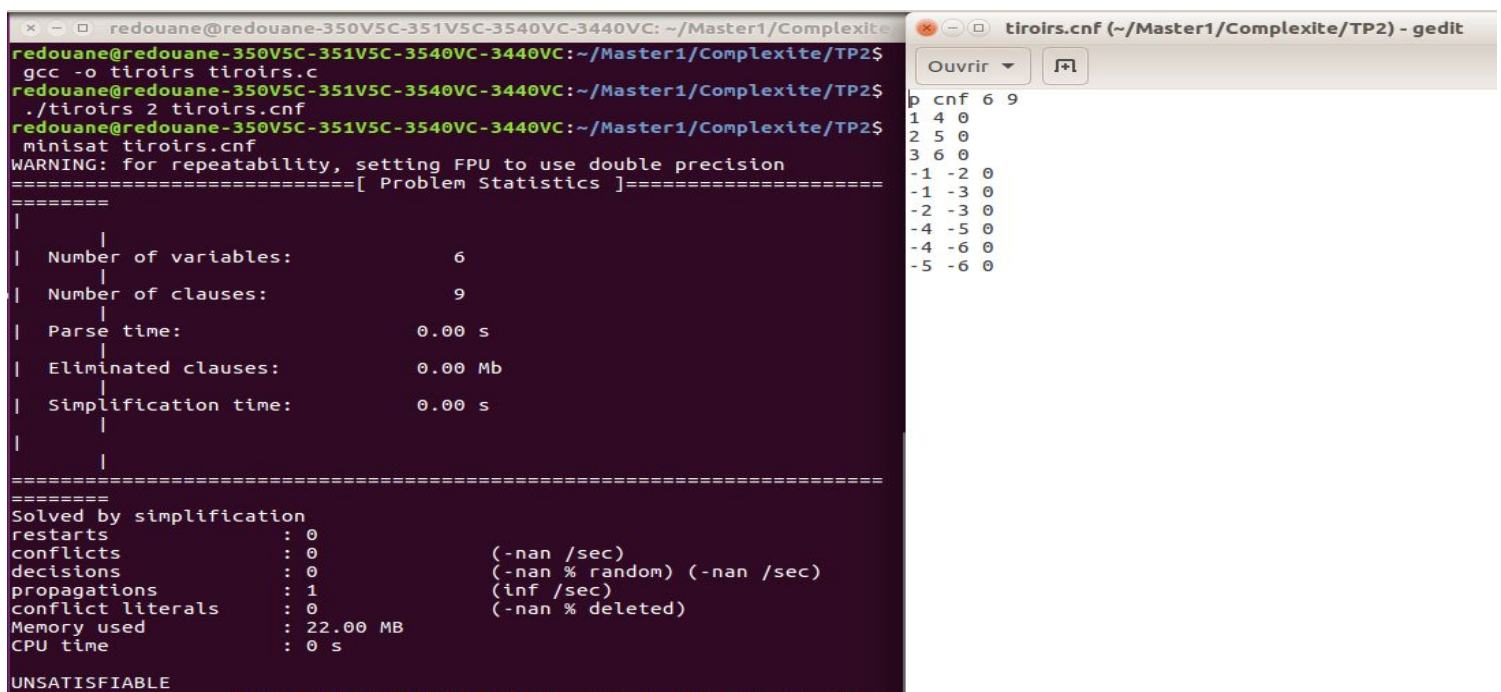
(e) Le problème des tiroirs :

- i. La formule propositionnelle :
  - i : le nombre de chaussettes (1 ..n+1)
  - j : le nombre de tiroir (1..n)
  - $C_{ij}$  : la chaussette i dans le tiroir j

$$\varphi_n = \bigwedge_{j=1..n} \left( \bigvee_{i=1..n+1} (C_{ij}) \right) \wedge \bigwedge_{j=1..n} \left( \bigwedge_{1 \leq i \neq i' \leq n+1} (\neg C_{ij} \vee \neg C_{i'j}) \right)$$

- ii. Pour tester le principe des tiroirs avec plusieurs valeur de n, on a créé un fichier .c (tiroirs.c) qui va à son tour créer un fichier .cnf (nom\_du\_fichier.cnf).

Test avec **Nombre\_Tiroirs** = 2 et **Nom\_fichier.cnf** = tiroirs.cnf



The screenshot shows two windows. The left window is a terminal running the program 'tiroirs.c' with 'n=2'. It shows the creation of 'tiroirs.cnf' and the execution of 'minisat tiroirs.cnf'. The output shows the problem statistics and the result 'UNSATISFIABLE'.

The right window is a gedit editor showing the content of 'tiroirs.cnf'. It is a 6-variable, 9-clause CNF formula.

```
b cnf 6 9
1 4 0
2 5 0
3 6 0
-1 -2 0
-1 -3 0
-2 -3 0
-4 -5 0
-4 -6 0
-5 -6 0
```

iii. Le résultat pour des valeurs croissantes de **n** est décrit dans le tableau suivant :

<b>n</b> Nombre tiroirs	Nombre chaussettes	Nombre Variables	Nombre Clauses	Temps d'Exécution "Solveur Minisat"
<b>2</b>	3	6	9	<b>0 sec</b>
<b>5</b>	6	30	81	<b>0 sec</b>
<b>8</b>	9	72	297	<b>0.368 sec</b>
<b>9</b>	10	90	415	<b>2.712 sec</b>
<b>10</b>	11	110	561	<b>166.54 sec (3 min)</b>
<b>11</b>	12	132	738	<b>2866.33 sec (48 min)</b>

On remarque que dès que le nombre de tiroirs dépasse 9, le temps d'exécution du solveur minisat augmente considérablement.

## Exercice 2.

1. La réduction polynomiale de 3-COLOR au problème SAT :

- i. Soit  $r$  une application tel que  $r : 3\text{-COLOR} \rightarrow \text{SAT}$ ,  $r$  associe une formule propositionnelle au graphe. Nous donnons la formule comme suit :

$$\varphi_n = \bigwedge_{i=1..n} (C_i^1 \vee C_i^2 \vee C_i^3) \wedge \bigwedge_{ii' \in E} ((\neg C_i^1 \vee \neg C_{i'}^1) \wedge (\neg C_i^2 \vee \neg C_{i'}^2) \wedge (\neg C_i^3 \vee \neg C_{i'}^3))$$

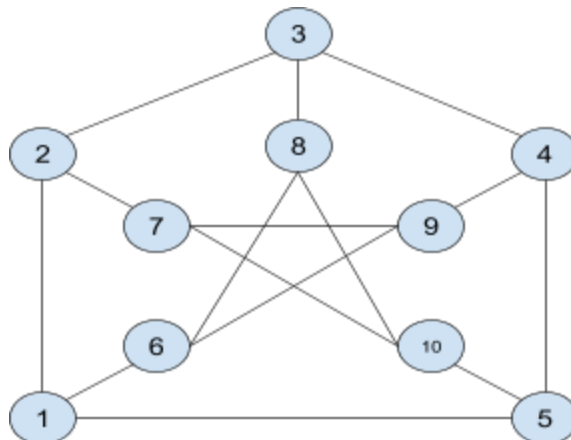
- ii. si la formule est satisfaisable alors le graphe est 3-COLOR c-a-d :

$$x \in I^+(3\text{-COLOR}) \quad \text{si et seulement si} \quad r(x) \in I^+(\text{SAT})$$

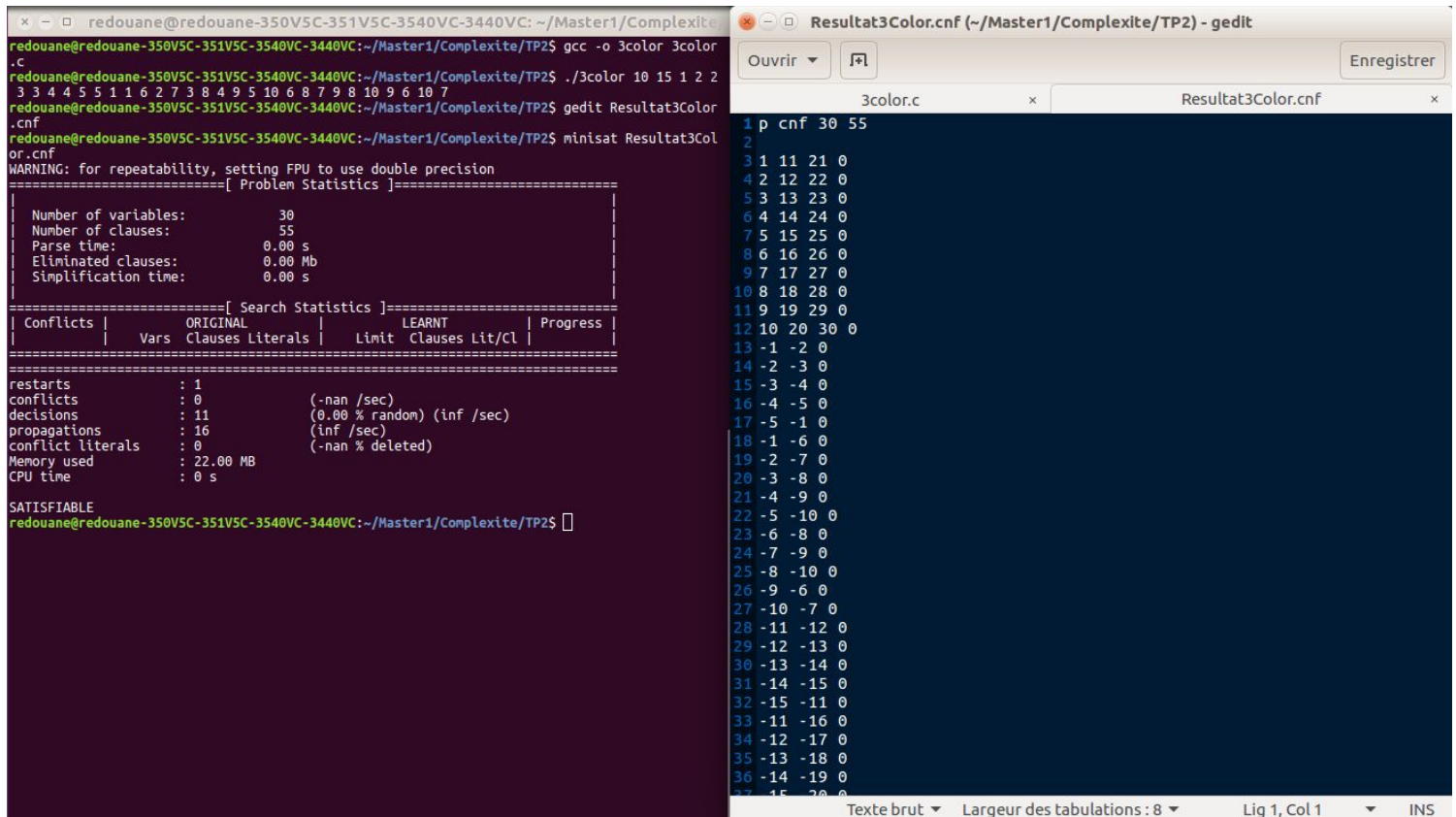
- iii. L'application " $r$ " se calcule en un temps polynomial.

2. Le graphe de la séquence suivante :

10 15 1 2 2 3 3 4 4 5 5 1 1 6 2 7 3 8 4 9 5 10 6 8 7 9 8 10 9 6 10 7



3. L'implémentation de la procédure qui étant donné un graphe construit un fichier .cnf (Resultat3Color.cnf) est décrite dans le fichier joint nommé **3color.c**
4. En exécutant le fichier **3color.c** avec le graphe donné dans l'énoncé on obtient un fichier **Resultat3Color.cnf** et en utilisant le solveur minisat on obtiendra le résultat suivant :



```

redouane@redouane-350V5C-351V5C-3540VC-3440VC: ~/Master1/Complexite
redouane@redouane-350V5C-351V5C-3540VC-3440VC:~/Master1/Complexite/TP2$ gcc -o 3color 3color.c
redouane@redouane-350V5C-351V5C-3540VC-3440VC:~/Master1/Complexite/TP2$ ./3color 10 15 1 2 2
3 3 4 4 5 5 1 1 6 2 7 3 8 4 9 5 10 6 8 7 9 8 10 9 6 10 7
redouane@redouane-350V5C-351V5C-3540VC-3440VC:~/Master1/Complexite/TP2$ gedit Resultat3Color.cnf
redouane@redouane-350V5C-351V5C-3540VC-3440VC:~/Master1/Complexite/TP2$ minisat Resultat3Color.cnf
WARNING: for repeatability, setting FPU to use double precision
===== [ Problem Statistics ] =====
Number of variables:      30
Number of clauses:       55
Parse time:              0.00 s
Eliminated clauses:      0.00 Mb
Simplification time:     0.00 s
===== [ Search Statistics ] =====
Conflicts | ORIGINAL | LEARNT | Progress |
Vars  | Clauses | Literals | Limit | Clauses | Lit/Cl |
=====
restarts      : 1
conflicts     : 0
decisions    : 11
propagations  : 16
conflict literals : 0
Memory used   : 22.00 MB
CPU time      : 0 s

SATISFIABLE
redouane@redouane-350V5C-351V5C-3540VC-3440VC:~/Master1/Complexite/TP2$

```

```

1 p cnf 30 55
2
3 1 11 21 0
4 2 12 22 0
5 3 13 23 0
6 4 14 24 0
7 5 15 25 0
8 6 16 26 0
9 7 17 27 0
10 8 18 28 0
11 9 19 29 0
12 10 20 30 0
13 -1 -2 0
14 -2 -3 0
15 -3 -4 0
16 -4 -5 0
17 -5 -1 0
18 -1 -6 0
19 -2 -7 0
20 -3 -8 0
21 -4 -9 0
22 -5 -10 0
23 -6 -8 0
24 -7 -9 0
25 -8 -10 0
26 -9 -6 0
27 -10 -7 0
28 -11 -12 0
29 -12 -13 0
30 -13 -14 0
31 -14 -15 0
32 -15 -11 0
33 -11 -16 0
34 -12 -17 0
35 -13 -18 0
36 -14 -19 0
37 -15 -20 0

```

5. Pour mieux voir le résultat et pour optimiser le temps des testes, nous avons mis en place un script qui lance l'exécution de la procédure qui construit le graphe dans un fichier .cnf et la deuxième procédure qui récupère le résultat du solveur minisat pour en déduire la distribution des 3 couleurs sur le graphes si il 3 coloriable.

Les trois fichiers sont joints et nomme **3color.c** , **coloriage.c** et **script.sh** . Ainsi nos résultats de test avec le graphe de l'énoncé sont décrit comme suit :





```
redouane@redouane-350V5C-351V5C-3540VC-3440VC:~/Master1/Complexite/TP2$ gcc -o 3color 3color.c
redouane@redouane-350V5C-351V5C-3540VC-3440VC:~/Master1/Complexite/TP2$ gcc -o coloriage coloriage.c
redouane@redouane-350V5C-351V5C-3540VC-3440VC:~/Master1/Complexite/TP2$ sh script.sh 10 15 1 2 2 3 3 4 4 5 5 1 1 6 2 7 3 8 4 9 5 10 6 8 7 9 8 10
9 6 10 7
WARNING: for repeatability, setting FPU to use double precision
===== [ Problem Statistics ] =====
|
| Number of variables:      30
| Number of clauses:       55
| Parse time:              0.00 s
| Eliminated clauses:      0.00 Mb
| Simplification time:     0.00 s
|
|===== [ Search Statistics ] =====
| Conflicts | ORIGINAL | LEARNT | Progress |
|           | Vars  Clauses Literals | Limit  Clauses Lit/Cl |
|===== [ Search Statistics ] =====
restarts      : 1
conflicts     : 0          (-nan /sec)
decisions     : 11         (0.00 % random) (inf /sec)
propagations  : 16         (inf /sec)
conflict literals : 0          (-nan % deleted)
Memory used   : 22.00 MB
CPU time      : 0 s

SATISFIABLE
redouane@redouane-350V5C-351V5C-3540VC-3440VC:~/Master1/Complexite/TP2$
```

Resultat3Color.cnf (~/Master1/Complexite/TP2) - gedit

Ouvrir ▾

Enregistrer

```
1 p cnf 30 55
2
3 1 11 21 0
4 2 12 22 0
5 3 13 23 0
6 4 14 24 0
7 5 15 25 0
8 6 16 26 0
9 7 17 27 0
10 8 18 28 0
11 9 19 29 0
12 10 20 30 0
13 -1 -2 0
14 -2 -3 0
15 -3 -4 0
16 -4 -5 0
17 -5 -1 0
18 -1 -6 0
19 -2 -7 0
20 -3 -8 0
21 -4 -9 0
22 -5 -10 0
23 -6 -8 0
24 -7 -9 0
25 -8 -10 0
26 -9 -6 0
27 -10 -7 0
28 -11 -12 0
29 -12 -13 0
30 -13 -14 0
```

Resultatminisat.txt (~/Master1/Complexite/TP2) - gedit

Ouvrir ▾

Enregistrer

```
1 SAT
2 -1 -2 3 -4 -5 6 7 -8 -9 -10 -11 12 -13 -14 15 -16 -17 18 19 -20 21 -22 -23 24 -25 -26 -27 -28 -29 30 0
```

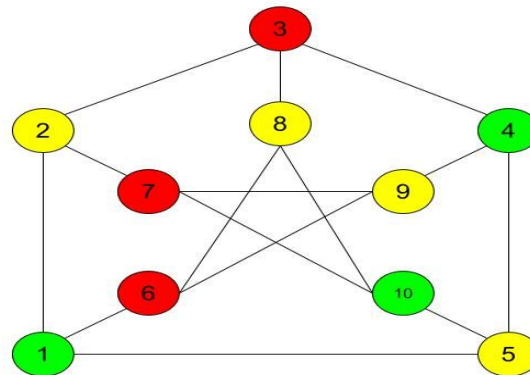
ResultatFinal.txt (~/Master1/Complexite/TP2) - gedit

Ouvrir ▾

Enregistrer

```
1 couleur 1 : 3 6 7
2 couleur 2 : 2 5 8 9
3 couleur 3 : 1 4 10
```

Texte brut ▾ Largeur des tabulations : 8 ▾ Lig 1, Col 1 ▾ INS



6. Pour tester notre solution sur des graphes aléatoires on a créé un nouveau fichier **construireGraphe.c** qui retourne un graphe généré aléatoirement dans un fichier **grapheRand**, ainsi on récupère le graphe généré et on le teste avec le script.