
TP 02 – Utiliser un solveur SAT, approximation

Langage de programmation au choix (à valider avec votre chargé de TP).

Exercice 1.*Sudoku*

Le Sudoku est un jeu de placement des chiffres de 1 à 9 dans une grille de 9 lignes et 9 colonnes, également découpée en 9 zones de taille 3×3 . Les chiffres de 1 à 9 doivent être placés de façon à ce que chaque ligne, chaque colonne et chaque zone contienne tous les chiffres de 1 à 9. Étant donnés des chiffres initialement placés dans la grille, le problème de décision du Sudoku consiste à savoir si la grille peut être complétée ou non en respectant les règles ci-dessus.

1. Donner un exemple d'instance négative du problème de décision du Sudoku.
2. En notation asymptotique (avec une expression simplifiée autant que possible), quelle est la complexité d'un algorithme de résolution du Sudoku sur une grille de taille 9×9 ?

On généralise le Sudoku aux grilles de taille $n^2 \times n^2$, qui peuvent être découpées en n^2 zones de taille $n \times n$. Le jeu utilise alors n^2 symboles différents pour remplir les cases de la grille.

3. Expliquer comment modéliser une instance du problème Sudoku généralisé à la taille $n^2 \times n^2$, comme une formule propositionnelle qui est satisfaisable si et seulement si la grille a une solution. Compter son nombre de variables et de clauses en fonction de n .
4. Implémenter votre réduction vers SAT de la question 3, pour la taille $n = 3$ (le Sudoku classique), en suivant la structure suivante :
 - (a) Votre programme prendra en entrée une grille 9×9 comme une chaîne de 81 caractères sur l'alphabet $\{1, 2, 3, 4, 5, 6, 7, 8, 9, .\}$, où $.$ indique une case initialement vide (les lignes de la grille sont concatenées, voir benchmarks en question 5), et produira une formule au format DIMACS (<https://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html>).
 - (b) Votre programme appelle `minisat` (<http://minisat.se/>) pour décider la satisfaisabilité, et obtenir un modèle si possible.
 - (c) Votre programme doit inclure une fonctionnalité d'affichage (dans le terminal, c'est suffisant) de la grille de départ et de sa solution (si celle-ci existe).
5. Mesurer expérimentalement le nombre de grilles que votre procédure peut résoudre en 1 seconde, pour chacun des deux jeux de données de grilles suivants :
 - https://pageperso.lis-lab.fr/kevin.perrot/benchmarks/benchmark_sudoku_1.txt
 - https://pageperso.lis-lab.fr/kevin.perrot/benchmarks/benchmark_sudoku_2.txtSource : <https://github.com/t-dillon/tdoku/benchmarks> 17_clue et forum_hardest_1905_11+.
6. Expliquer comment modéliser la contrainte additionnelle suivante pour la taille $n^2 \times n^2$: deux cases adjacentes horizontalement ne doivent pas contenir des entiers consécutifs. Ajouter à votre code une fonctionnalité permettant de produire ces clauses.

Exercice 2.

Emploi du temps

Nous allons modéliser et résoudre un problème de construction d'emploi du temps compatible avec les contraintes des étudiants et des enseignants. Soient n étudiants, m enseignants, et ℓ cours. Chaque étudiant suit un ensemble de cours, et chaque enseignant donne un ensemble de cours. Dans l'emploi du temps de la semaine, nous disposons de 10 demi-journées pour placer les cours. Un étudiant ne peut pas suivre deux cours simultanément, et un enseignant ne peut pas donner deux cours simultanément.

Étant donnés les cours suivis par chacun des étudiants, et les cours donnés par chacun des enseignants, est-il possible de contruire un emploi du temps compatible ?

1. Donner une instance positive et une instance négative du problème.
2. Écrire le code d'un générateur aléatoire d'instances du problème.

Pour un entier naturel k donné, le problème **k -Coloration** consiste, étant donné un graphe non-orienté $G = (V, E)$, à décider si on peut assigner à chacun de ses sommets une couleur parmi $\{1, 2, \dots, k\}$, de façon à ce que si deux sommets sont reliés par une arête alors ils ont des couleurs différentes. Mathématiquement, $\exists c : V \rightarrow \{1, 2, \dots, k\}, \forall \{u, v\} \in E, c(u) \neq c(v)$?

3. Comment modéliser la construction d'emploi du temps comme un problème de coloration de graphe ? (Quel graphe ? Combien de couleurs ?)
4. Sachant que le problème **3-Coloration** est NP-complet, montrer que le problème d'emploi du temps est NP-complet.
5. Expliquer comment réduire en temps polynomial **10-Coloration** à **SAT**. C'est-à-dire, à partir d'un graphe G , comment construire une formule φ_G qui soit satisfaisable si et seulement si le graph G est k -coloriable.
 - (a) Quelles variables ?
 - (b) Quelle conjonction de contraintes (clauses) ?
 - (c) Exprimer en CNF.

Donner le nombre de variables, ainsi le nombre et la taille des clauses, comme une fonction de $n = |V|$ et $m = |E|$. Analyser le temps de calcul de la transformation.

6. Écrire le code de cette réduction, d'un graphe vers une formule au format DIMACS.
7. Brancher votre générateur d'instance du problème d'emploi du temps au solveur `minisat`, via par votre réduction de **k -Coloration** vers **SAT**. Produire une solution (si possible).

Exercice 3.

Réseau sans-fil

Un réseau sans-fil est constitué de n antennes, dont certaines sont reliées entre elles et d'autres non. Les connexions sont données par un graphe non-orienté $G = (V, E)$ avec $|V| = n$ sommets (un sommet par antenne). Dans le but d'effectuer du routage, un administrateur souhaite construire une colonne vertébrale du réseau (*network backbone*). Il s'agit d'un sous-ensemble d'antennes reliées à toutes les antennes.

On cherche à minimiser le nombre d'antennes nécessaires. Le problème de décision associé s'appelle **Ensemble-dominant** : étant donné un graphe $G = (V, E)$ et un entier k , existe-t-il un sous-ensemble de sommets $X \subseteq V$ de taille k tel que pour tout $v \in V$ on a $(\{v\} \cup \{u \mid \{u, v\} \in E\}) \cap X \neq \emptyset$?

1. Montrer que ce problème appartient à la classe NP.
2. La taille de l'ensemble-dominant est-elle difficile à maximiser ou à minimiser ?

On rappelle qu'un stable (ensemble-indépendant) est un sous-ensemble de sommets qui n'ont aucune liaison entre eux.

3. Montrer que tout stable maximal est également un ensemble-dominant minimal.
4. Sachant que le problème **Stable** (étant donnés G et k , est-ce que G possède un stable de taille k ?) est NP-difficile, peut-on en déduire que **Ensemble-dominant** est NP-difficile?
5. Utiliser `minisat` pour résoudre ce problème (réduction vers **SAT**).
 - (a) Quelles variables?
 - (b) Quelle conjonction de contraintes (clauses)?
 - (c) Exprimer en CNF.

Nous allons désormais approximer le problème, en tâchant de calculer un ensemble-dominant dont la taille n'est pas trop éloignée du minimum.

6. Implémenter un algorithme glouton pour trouver un ensemble-dominant, où à chaque étape on sélectionne le sommet qui domine le plus de nouveaux sommets.
7. Analyser la complexité de votre algorithme en question 6.

Nous allons maintenant démontrer que l'algorithme glouton est une $(\log \Delta + 2)$ -approximation, où Δ est le degré maximum du graphe.

8. Que signifie être une $(\log \Delta + 2)$ -approximation?

Fixons G un graphe arbitraire de degré maximum Δ . Soit S^* un ensemble-dominant optimal, et S l'ensemble-dominant retourné par l'algorithme glouton. On a $|S^*| \leq |S|$.

Soit $W = S \setminus S^*$. Puisque S^* est un ensemble-dominant, tout sommet $w \in W$ a un voisin dans S^* , que l'on notera $s(w)$. L'ensemble des sommets $w \in W$ et leur voisin $s(w)$ dans S^* forment une collection d'étoiles. Nous allons analyser ces étoiles.

9. Dans chaque étoile, combien de sommets sont dans S^* ?

Nous allons maintenant réaliser pour chaque étoile un calcul amorti du nombre de sommets dans S , qui montrera qu'il y en a (globalement) au plus $\log \Delta + 2$ par étoile.

10. Pourquoi cela permettra bien de conclure?

Considérons le déroulement de l'algorithme glouton. Quand un sommet u est sélectionné dans S par l'algorithme glouton, nous allons distribuer uniformément son coût (de 1 dans la taille de S) sur les sommets nouvellement dominés par l'ajout de u à S .

11. Quels sont les plus petits et plus grands coûts que peuvent recevoir un sommet?

On se concentre maintenant sur une étoile, dont on note s^* le centre et $d(s^*)$ son degré.

12. Montrer que quand un premier sommet de cette étoile est dominé par un sommet u ajouté à S par l'algorithme glouton, il reçoit un coût d'au plus $\frac{1}{d(s^*)+1}$.
13. Montrer que quand un deuxième sommet de cette étoile est dominé par un sommet u ajouté à S par l'algorithme glouton, il reçoit un coût d'au plus $\frac{1}{d(s^*)}$.
14. Montrer que quand un troisième sommet de cette étoile est dominé par un sommet u ajouté à S par l'algorithme glouton, il reçoit un coût d'au plus $\frac{1}{d(s^*)-1}$.
15. En déduire une borne sur le coût total de l'étoile dont le centre est s^* .

On note $H(n) = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$ le n -ième nombre harmonique.

16. En groupant les termes par 1, 2, 4, 8, 16... montrer que $H(2^m) \leq m + 1$.
17. Conclure.