
Travail TP : Le problème de satisfaisabilité booléenne (SAT)

1 Présentation du Problème

Problème SAT (Satisfaisabilité)

Soit F une formule logique exprimée en **forme normale conjonctive (CNF)**, c'est-à-dire une conjonction de clauses où chaque clause est une disjonction de littéraux.

Formulation formelle :

- $F = C_1 \wedge C_2 \wedge \dots \wedge C_k$, où chaque clause C_i est de la forme :

$$C_i = (l_{i1} \vee l_{i2} \vee \dots \vee l_{im_i}),$$

avec l_{ij} un littéral (une variable x ou sa négation $\neg x$).

Objectif : Trouver une affectation des variables de F telle que F soit évaluée à vrai, c'est-à-dire qu'il existe au moins un littéral vrai dans chaque clause C_i .

Sortie :

- "Oui" si une telle affectation existe, avec éventuellement l'affectation trouvée.
- "Non" sinon.

Exemple pour le problème SAT

Considérons la formule suivante en forme normale conjonctive (CNF) :

$$F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee x_3).$$

Une affectation possible qui satisfait F est :

$$x_1 = \text{vrai}, x_2 = \text{vrai}, x_3 = \text{vrai}.$$

Avec cette affectation, chaque clause contient au moins un littéral évalué à vrai, donc F est satisfaisable.

Problème 3-SAT

Le problème 3-SAT est une version particulière du problème SAT où chaque clause contient **exactement trois littéraux**.

Formulation formelle :

- $F = C_1 \wedge C_2 \wedge \dots \wedge C_k$, où chaque clause C_i est de la forme :

$$C_i = (l_{i1} \vee l_{i2} \vee l_{i3}),$$

avec l_{ij} un littéral (une variable x ou sa négation $\neg x$).

Objectif : Déterminer s'il existe une affectation des variables de F telle que F soit évaluée à vrai, c'est-à-dire qu'il existe au moins un littéral vrai dans chaque clause C_i .

Sortie :

- "Oui" si une telle affectation existe, avec éventuellement l'affectation trouvée.
- "Non" sinon.

Exemple pour le problème 3-SAT

Considérons la formule suivante en 3-SAT (chaque clause contient exactement 3 littéraux) :

$$F = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_3).$$

Une affectation possible qui satisfait F est :

$$x_1 = \text{faux}, x_2 = \text{faux}, x_3 = \text{vrai}.$$

Avec cette affectation, chaque clause contient au moins un littéral évalué à vrai, donc F est satisfaisable.

Objectifs de ce travail :

- Implémenter les solutions pour résoudre les deux problèmes.
- Implémenter la réduction de 3SAT vers SAT et l'étudier pratiquement à travers SATLIB.
- Analyser et comparer les complexités en temps et en espace de chaque algorithme de manière empirique.

2 Questions

• Développement d'Algorithmes

1. Etant donnée une instance I du problème SAT, écrire un programme qui trouve la solution de cette instance.
2. Etant donnée une instance du problème SAT, écrire un programme qui vérifie si une **solution** potentielle s satisfait toutes la clauses I .
3. Etudier la complexité pratique, temporelle et spatiale, de deux programmes précédents en spécifiant les structures de données utilisées pour représenter l'instance et les solutions.
4. Pareil pour le problème du 3SAT :
 - Implémenter l'algorithme qui trouve la solution d'une instance donnée.
 - Implémenter l'algorithme qui vérifie si une solution potentielle donnée satisfait une instance donnée.
 - Etudier la complexité pratique, temporelle et spatiale, des deux programmes en spécifiant les structures de données utilisées.
5. Implémenter un algorithme de réduction (linéaire) qui permet de transformer les instances du SAT en instances 3SAT (ainsi que leurs solutions) et étudier sa complexité pratique.

• Analyse Empirique de la Complexité

1. Comparaison de la Performance :

- Générer des instances de tailles différentes pour les deux problèmes.
- Enregistrez et analysez le temps d'exécution et l'utilisation de la mémoire de chaque approche pour chaque taille d'entrée.
- Créez des graphes pour montrer la croissance empirique des performances en temps et en espace en fonction de la taille de l'entrée.

2. Exploration d'applications du monde réel :

- Utiliser la bibliothèque SATLIB¹ pour tester vos programmes.

¹<https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html>

3 Livrables

- **Code** : Remettre code de chaque solution et l'application.
- **Rapport** : Inclure un rapport (max 8 pages) qui couvre :
 - L'analyse de la complexité et les résultats empiriques (graphiques inclus). En spécifiant l'environnement d'expérimentation (Matériel et Logiciel).
 - La discussion sur les optimisations et les conditions qui réduisent la complexité pour l'application.

Ce travail doit être remis au plus tard Le 02/01/25 à 00:00.
Ce travail se fait en *trinôme ni plus ni moins*.