HAI802I Algorithmique Avancée — TDs

Ivan Lejeune

2 février 2026

Table des matières

Liste des Algorithmes

1 Complexité

1.1 Quelques preuves en complexité des problèmes

Exercice 1 Quelques preuves de NP-complétude autour de 3-SAT.

Algorithme 1 – 3-SAT

Input: Une formule $\varphi(x_1, \ldots, x_n)$ en 3-CNF
Question: Existe-t-il une affectation des variables $(x_1, \ldots, x_n)$ satisfaisant φ ?

Algorithme 2 – Half 3-SAT

Input: Une formule $\varphi(x_1, \ldots, x_n)$ en 3-CNF
Question: Existe-t-il une affectation des variables $(x_1, \ldots, x_n)$ satisfaisant φ à 50% ?

Algorithme 3 – ¬3-SAT

Input: Une formule $\varphi(x_1, \ldots, x_n)$ en 3-CNF
Question: Existe-t-il une affectation des variables $(x_1, \ldots, x_n)$ qui ne satisfait pas φ ?

1. Montrer que ¬3-SAT appartient à la classe P.
2. Montrer que Half 3-SAT est NP-complet. Que dire du problème Half 3-SAT dans le cas où le nombre de clauses satisfaites est au moins de 50% ?

Solution.

1. On rappelle que tous les litéraux sont positifs, il suffit alors de tout mettre à faux pour trouver une solution, donc le problème appartient à la classe P.
2. On fait une réduction de 3-SAT vers Half 3-SAT.
   On commence par faire la chose suivante :

$$I \rightsquigarrow I + \underbrace{(x \vee y \vee z)}_{m \text{ copies}}, \quad x, y, z \notin I$$

   Si on a oui à 3-SAT alors on a m clauses satisfaites. On en déduit que sur $2m$ clauses, il suffit de mettre x, y, z à faux et alors on a bien 50% des clauses qui sont satisfaites par la solution à 3-SAT.
   Dans l'autre sens, si on a une solution à Half 3-SAT on a plusieurs cas :
   - Premier cas, on satisfait une clause dans I. Alors toutes les m clauses sont validées dans I et on a une solution à 3-SAT.
   - Second cas, on ne satisfait aucune clause de I. Mais alors si dans I il n'y a aucune variable négative, on peut résoudre 3-SAT en mettant tout à positif, de même dans le cas où il n'y a aucune variable positive. Il y a donc forcément une variable présente de manière positive et négative dans I. Alors, au moins une clause est satisfaite. Donc on est dans le premier cas et forcément les m clauses satisfaites sont dans I.

   Le problème de au moins 50% est facile. Il suffit de prendre une affectation aléatoire (ou son complémentaire).

Exercice 2 Autour de Satisfaisabilité.

Algorithme 4 – Non Egal Satisfaisabilité (NAESAT)

Input: Und vofmuld onjonctive φ sur n variables et m clauses
Question: Existe-t-il une affectation de valeurs de vérité aux variables qui satisfasse φ tel que chaque clause a un littéral vrai et un littéral faux ?

Montrer que Non Egal Satisfaisabilité est $\mathcal{NP}$-*complet*. La preuve se fera à partir de Satisfaisabilité.

Solution.

1.2 Quelques problèmes dans les graphes

Exercice 3 Quelques preuves de NP-complétude pour Steiner tree.

Algorithme 5 – Exact Cover by 3-sets
Input: Un univers X, une collection C de triplés de X
Question: Trouver une collection $C' \subset C$ telle que chaque élément de X apparait exactement une fois dans les triplés de C'

Algorithme 6 – Vertex Cover
Input: Un graphe $G = (V, E)$, un entier k
Question: Trouver $V' \subset V$ telle que chaque arête est couverte par un sommet de V avec $|V'| = k$

Algorithme 7 – Steiner tree
Input: Un graphe $G = (V, E)$, une fonction de poids w sur les arêtes, un ensemble de terminaux $X \subset V$, un entier k
Question: Trouver $T \subset E$ tel que la somme des poids des arêtes de T soit au plus de k, $G[T]$ est connexe et $X \subset V(G[T])$

1. Montrer que

Solution.

Exercice 4 title.

Solution.

Exercice 5 title.

Solution.

Exercice 6 title.

Solution.

Exercice 7 title.

Solution.

Exercice 8 title.

Solution.

1.3 La classe de complexité $co\mathcal{NP}$

Exercice 9 title.

Solution.

Exercice 10 title.

Solution.

Exercice 11 title.

Solution.

1.4 Rappels en complexité

Exercice 12 title.

Solution.

2 Algorithmes d'approximation

2.1 La classe Log-APX

Exercice 13 title.

Solution.

2.2 La classe APX

Exercice 14 title.

Solution.

Exercice 15 title.

Solution.

Exercice 16 title.

Solution.

Exercice 17 title.

Solution.

Exercice 18 title.

Solution.

Exercice 19 title.

Solution.

Exercice 20 title.

Solution.

Exercice 21 title.

Solution.

Exercice 22 title.

Solution.

Exercice 23 title.

Solution.

Exercice 24 title.

Solution.

Exercice 25 title.

Solution.

Exercice 26 title.

Solution.

Exercice 27 title.

Solution.

Exercice 28 title.

Solution.

Exercice 29 title.

Solution.

2.3 Mesure différentielle

Exercice 30 title.

Solution.

Exercice 31 title.

Solution.

Exercice 32 title.

Solution.

2.4 Construction d'un FPTAS

Exercice 33 title.

Solution.

Exercice 34 title.

Solution.

2.5 Polynomial-Time Asymptotic Scheme : PTAS

Exercice 35 title.

Solution.

Exercice 36 title.

Solution.

2.6 Points extrêmes

Exercice 37 title.

Solution.

Exercice 38 title.

Solution.

Exercice 39 title.

Solution.

2.7 FPTAS et programmation dynamique

Exercice 40 title.

Solution.

2.8 Schéma primal-dual

Exercice 41 title.

Solution.

3 Inapproximabilité ou seuil d'approximation

3.1 Seuil d'approximation

Exercice 42 title.

Solution.

Exercice 43 title.

Solution.

Exercice 44 title.

Solution.

3.2 Utilisation du principe Gap-réduction

Exercice 45 title.

Solution.

Exercice 46 title.

Solution.

Exercice 47 title.

Solution.

Exercice 48 title.

Solution.

Exercice 49 title.

Solution.

4 Méthodes exactes

4.1 Programmation dynamique

Exercice 50 title.

Solution.

Exercice 51 title.

Solution.

4.2 Arbres de branchement

Exercice 52 title.

Solution.

Exercice 53 title.

Solution.

Exercice 54 title.

Solution.

Exercice 55 title.

Solution.

5 Réductions divers

5.1 L-réduction

Exercice 56 title.

Solution.

Exercice 57 title.

Solution.

6 Quelques problèmes d'ordonnancement

6.1 Ordonnancement sur un processeur

Exercice 58 title.

Solution.

6.2 Ordonnancement sur m processeurs

Exercice 59 Problème d'ordonnancement sans contrainte de précédence.
On pose $P|\cdot|C_{\max}$ définie de la manière suivante :

Algorithme 8 – $P|\cdot|C_{\max}$
Input: n taches de durées $p_1, \ldots, p_n$ indépendantes
Question: Ordonnancer ces tâches sur m machines identiques en une durée totale minimale notée $C_{\max}$.

Par la suite on va utiliser un ordonnancement par liste. Le principe est le suivant :

- On construit une liste de priorité de taches.
- A chaque étape la première machine disponible est sélectionnée pour executér la première tâche de la liste.

1. Montrer que le problème est $\mathcal{NP}$-complet meme pour deux machines.
2. Pour n'importe quel ordonnancement de liste LS on a $\frac{C_{\max}^{LS}}{C_{\max}^{*}} \leq 2$. Montrer que la borne est supérieure est atteinte.
3. Nous considérons le nouvel algorithme dont le principe est le suivant :
   - On construit une liste de priorité de taches dans l'ordre décroissant.
   - A chaque étape la première machine disponible est sélectionnée pour exéctuer la première tache de la liste.

   Pour n'importe quel ordonnancement de liste on a $\frac{C_{\max}^{LPT}}{C_{\max}^{*}} \leq \frac{4}{3} - \frac{1}{3m}$.
4. Un schéma d'approximation polynomiale pour le problème $P|\cdot|C_{\max}$

Solution.

1. On va faire une réduction du problème de 2-Partition vers $P|\cdot|C_{\max}$. On rappelle l'énoncé du problème 2-Partition :

Algorithme 9 – 2-Partition
Input: Un ensemble de n objets a_i de poids $s(a_i)$ de somme $2p$
Question: Est-il possible de trouver une partition en deux sous-ensemble de poids total p

Dans le problème de de 2-Partition, on a

$$\sum_{i=1}^{n} s(a_i) = 2p$$

On pose

$$s(a_i) = p_i \quad \text{et} \quad w = \sum_{i=1}^{n} p_i$$

Montrons que si il est possible de partager les poids en deux sous-ensembles de même poids alors il existera une distribution des taches pour le problème $P|\cdot|C_{\max}$ telle que

$$C_{\max} \leq T \leq \frac{w}{2}$$

Si on peut partager les poids en deux sous-ensembles P_1, P_2 alors il existe S tel que

$$\mathsf{Load}(M_1) = \sum_{i \in S} P_i = \sum_{a_i \in P_1} s(a_i) = p$$

de même pour l'autre machine. Alors on a bien trouvé une solution.
Pour le sens indirect, soit C une distribution telle que

$$C_{\max} \leq \frac{w}{2}$$

Soit S un sous-ensemble de travaux de sorte que $S \subseteq \{1, \ldots, n\}$ et

$$\mathsf{Load}(M_1) = \sum_{i \in S} p_i = \frac{w}{2}$$

Alors, comme pour tout i on a $p_i = s(a_i)$, l'ensemble des taches sur le processeur P_1 forme une solution du problème de $P|\cdot|C_{\max}$ avec une bipartition

$$\sum_{i \in P_1} s(a_i) = p$$

Les éléments restants sont dans P_2 ce qui donne une solution à 2-Partition.

2. On peut déjà trouver deux bornes inférieures intéressantes :

$$C_{\max}^{OPT} \geq \max_{i \in I} p_i, \quad \text{et} \quad C_{\max}^{OPT} \geq \frac{1}{m} \sum_{i=1}^{n} p_i$$

Il faut au moins la durée de la tache de la plus longue et on ne peut pas faire mieux que la répartition parfaite de toutes les taches sur tous les processeurs de manière continue.
On a

$$C_{\max} = t_j + p_j \leq \max_i p_i + t_j \leq C_{\max}^{OPT} + t_j$$

Mais de par la nature de notre algorithme, avant t_j tous les processeurs sont actifs d'où

$$C_{\max} = t_j + p_j \leq 2C_{\max}^{OPT}$$

Exercice 60 title.

Solution.

Exercice 61 title.

Solution.

Exercice 62 title.

Solution.

7 Algorithmes randomisés

7.1 Sur le problème Maximum Satisfaisabilité

Exercice 63 title.

Solution.

7.2 Sur le problème Couverture d'ensembles pondérés

Exercice 64 title.

Solution.

Exercice 65 title.

Solution.

Exercice 66 title.

Solution.

Exercice 67 title.

Solution.

Exercice 68 title.

Solution.

8 Bornes inférieures et ETH

Exercice 69 title.

Solution.

Exercice 70 title.

Solution.

Exercice 71 title.

Solution.