HAI802I Algorithmique Avancée — TDs

Ivan Lejeune

27 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: Une formule conjonctive 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. Le problème de Non Egal Satisfaisabilité est dans $\mathcal{NP}$ car on peut vérifier en temps polynomial si deux formules booléennes sont non équivalentes. Pour montrer qu'il est $\mathcal{NP}$-complet, on effectue une réduction à partir du problème de Satisfaisabilité. On montre que si on pouvait résoudre Non Egal Satisfaisabilité en temps polynomial, alors on pourrait résoudre Satisfaisabilité en temps polynomial, ce qui contredit l'hypothèse que $\mathcal{P} \neq \mathcal{NP}$. Par conséquent, Non Egal Satisfaisabilité est $\mathcal{NP}$-complet.
Soit φ une formule conjonctive de 3-SAT. On construit une formule ψ pour le problème de Non Egal Satisfaisabilité de la manière suivante :

$$\psi_i = \varphi_i \wedge (x_1 \vee \ldots \vee x_n) \wedge (\neg x_1 \vee \ldots \vee \neg x_n), \quad \forall i \in \{1, \ldots, m\}$$

Ensuite, on pose $\psi = \bigwedge_{i=1}^{m} \psi_i$.
Si φ est satisfaisable, alors il existe une affectation de valeurs de vérité aux variables qui satisfait φ. Cette même affectation satisfait également ψ car les clauses supplémentaires garantissent que chaque clause de ψ a un littéral vrai et un littéral faux.
Inversement, si ψ est satisfaisable, alors il existe une affectation de valeurs de vérité aux variables qui satisfait ψ. Cette affectation doit également satisfaire φ (car si elle satisfait chaque clause de ψ), alors elle satisfait également chaque clause de φ.
Par conséquent, φ est satisfaisable si et seulement si ψ est satisfaisable, ce qui montre que le problème de Non Egal Satisfaisabilité est $\mathcal{NP}$-complet (car il est au moins aussi dur que Satisfaisabilité, qui est lui-même $\mathcal{NP}$-complet).

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 Steiner Tree est NP-complet. Procéder à une réduction à partir de Exact Cover by 3-Sets.
2. Montrer que Steiner Tree est NP-complet par une réduction à partir de Vertex Cover.

Solution.

Exercice 4 Autour de Coupe maximum.

Algorithme 8 – Coupe maximum (CUT)
Input: Soit $G = (V, E)$ un graphe non orienté et $k \in \mathbb{N}$.
Question: Existe-t-il une partition des sommets en deux sous-ensembles V_1 et V_2 tels que le nombre d'arêtes entre V_1 et V_2 soit k ?

Réduire Non Egal 3-SAT à Coupe maximum. Conclure. *Remarque* : vous verrez en master que coupe min est polynomiale meme si les aretes ont des poids.

Solution.

Exercice 5 title.

Solution.

Exercice 6 title.

Solution.

Exercice 7 title.

Solution.

Exercice 8 Problème de la coloration.

1. Montrer que 3-Coloration est $\mathcal{NP}$-complet en effectuant une réduction à partir de 3-SAT.
   Nous proposons la transformation polynomiale suivante à partir de 3-SAT :
   Soit I une instance de 3-SAT constitué d'un ensemble $B = \{C_1, \ldots, C_m\}$ de clauses sur un ensemble $X = \{x_1, \ldots, x_n\}$ de variables. Nous allons construire, à partir de la formule B, un graphe $G = (S, A)$ de manière que B soit satisfaisable si et seulement si G admet une coloration en 3 couleurs.
   - Pour chaque variable x_i, on pose $T_i = (S_{i,1}, A_{i,1})$ où $S_{i,1} = \{x_i, \overline{x_i}\}$ et $A_{i,1} = \{(x_i\overline{x_i})\}$, voir figure 1.
   - Pour chaque clause C_j, on pose $V_j = (S_{j,2}, A_{j,2})$ où $S_{j,2} = \{y_{j,1}, y_{j,2}, y_{j,3}, y_{j,4}, y_{j,5}, y_{j,6}\}$ et $A_{j,2} = \{(y_{j,1}y_{j,2}), (y_{j,1}y_{j,4}), (y_{j,2}y_{j,4}), (y_{j,4}y_{j,5}), (y_{j,5}y_{j,6}), (y_{j,5}y_{j,3}), (y_{j,3}y_{j,6})\}$, voir figure 2.
   - V_j est isomorphe à $G_0 - \{u_0, u_1, u_2\}$.
   - On pose $T = (S_3, A_3)$ où $S_3 = \{v_1, v_2\}$ et $A_3 = \{(v_1v_2)\}$.
   - Pour chaque variable x_i on pose $A_{i,4} = \{v_2x_i, v_2\overline{x_i}\}$ et $A_{i,5} = \{(v_1y_{j,6}, v_2y_{j,6})\}$ pour tout $j \in \{1, \ldots, m\}$.
   - Enfin, $A_{j,6}$ est un ensemble de trois arêtes reliant, pour chaque clause C_j, les trois sommets de type x_i ou $\overline{x_i}$ correspondant aux littéraux de C_j. Le premier littéral de C_j est relié à $y_{j,1}$, le deuxième à $y_{j,2}$ et le troisième à $y_{j,3}$, voir figure 3.

   Tous les graphes présentés ci-dessus sont donnés dans les figures ci-dessous :

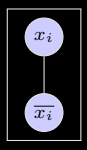

FIGURE 1 – Graphe T_i

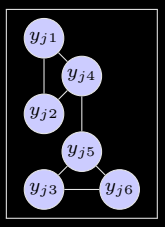

FIGURE 2 – Graphe V_j

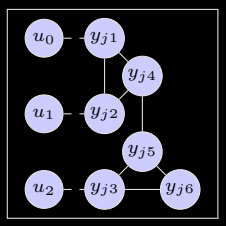

FIGURE 3 – Graphe G_0

Donner le nombre de sommets et le nombre d'arêtes de G. Appliquer la construction sur l'instance $X = \{x_1, x_2, x_3, x_4, x_5\}$ et $B = \{C_1, C_2\}$ où $C_1 = (x_1 \vee \overline{x_3} \vee \overline{x_4})$ et $C_2 = (\overline{x_4} \vee x_2 \vee \overline{x_1})$.

2. Montrer que le problème de 4-coloration est $\mathcal{NP}$-complet en effectuant une réduction à partir de 3-COLORATION.
3. Regarder la preuve pour des graphes planaires.

Solution. On propose de visualiser le schéma de construction avec la figure 4 suivante :

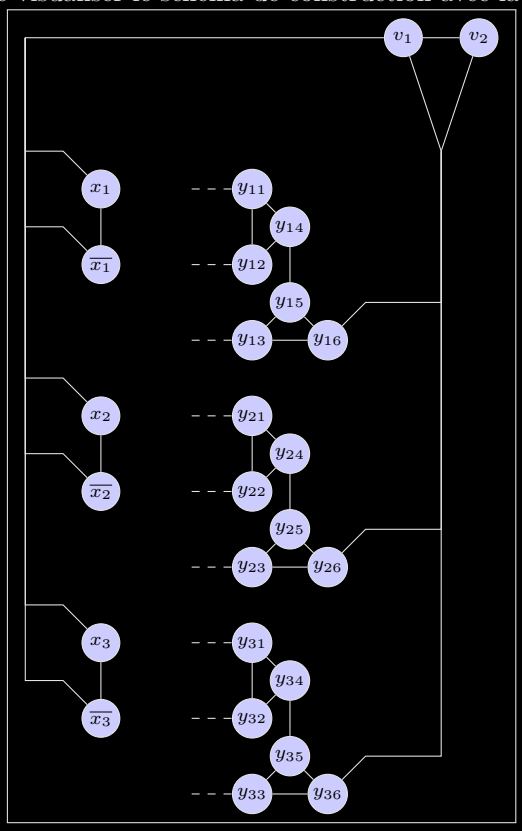

FIGURE 4 – Construction de la réduction de 3-COLORATION à partir de 3-SAT

En appliquant le schéma de construction à l'instance donnée, on obtient le graphe suivant :

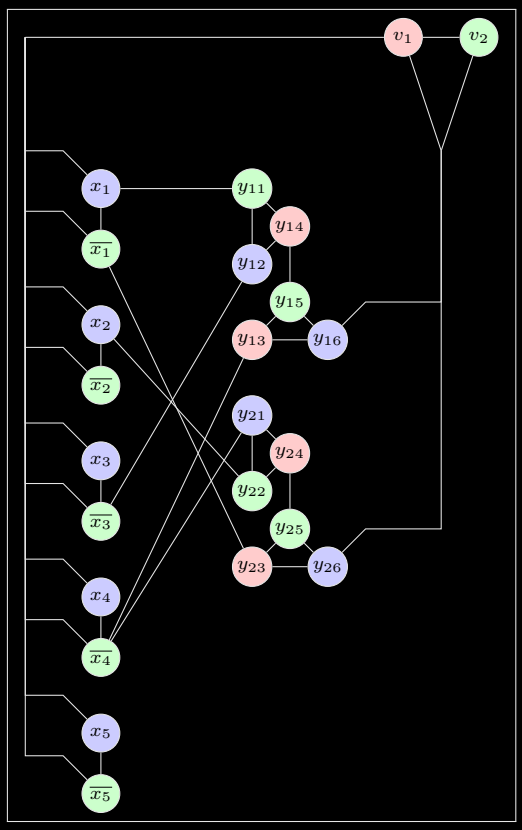

FIGURE 5 – Application de la construction à l'instance donnée avec un exemple de coloration

On peut aussi retirer les couleurs de ce graphe et essayer de les assigner manuellement. Les paires $(x_i, \overline{x_i})$ doivent être colorées avec des couleurs différentes, et v_1 et v_2 doivent être colorés avec des couleurs différentes. Les $y_{j,6}$ doivent être colorés différemment de v_1 et v_2. Avec ces seules contraintes, on peut trouver une coloration en 3 couleurs du graphe, ce qui correspond à une assignation de vérité pour les variables x_i qui satisfait les clauses C_j et on peut passer de l'une à l'autre de manière bijective.
Les seules choses à considérer pour passer d'une coloration à une autre est la permutation des couleurs $y_{j,1}, y_{j,2}$ et celles des x_i et $\overline{x_i}$, l'extension de cette construction à des instances plus grandes est alors facile.

1. Il est facile de voir qu'avec la construction proposée on a bien une réduction de 3-SAT vers 3-COLORATION et donc ce dernier est $\mathcal{NP}$-complet.
2. Il suffit de considérer le graphe G' qui a les memes sommets et arêtes que G mais avec un sommet supplémentaire v connecté à tous les autres sommets de G. Alors si G est 3-colorable, alors G' est 4-colorable (en coloriant v avec une nouvelle couleur). Inversement, si G' est 4-colorable, alors v doit être colorié avec une couleur différente de celle de tous les autres sommets de G, ce qui implique que G est 3-colorable.

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 Approximation pour Maximum Independent : algorithme glouton.

1. Donner la complexité de l'algorithme et montrer que la solution donnée par l'algorithme est un stable.
2. A partir de l'algorithme, proposer un graphe qui donne le pire cas.
3. Soit $G = (V, E)$ un graphe ayant n sommets et m arêtes, soit $\delta = \frac{m}{n}$ la densité du graphe de G. Montrer que toute solution S est d'au moins $\frac{n}{2\delta+1}$.
4. Montrer maintenant que le ratio est $\delta + 1$.

Solution.

1.
2. On propose le graphe suivant Où $A = K_n, B = IS_n, C = \{x\}$ où C domine B et A et B sont complètement connectés. L'algorithme peut choisir x et ne pas choisir les sommets de B alors que la solution optimale est de choisir tous les sommets de B. Ainsi on a bien un ratio de $\frac{n}{2}$ qui peut être arbitrairement grand.
3. Sachant que l'algorithme s'arrête, lorsqu'on a plus de sommets à traiter, on a :

$$\sum_{i \in S}(d_i + 1) = n \tag{1}$$

où d_i est le degré du sommet i dans le graphe restant.
De plus, on a

$$\sum_{i \in S} \frac{d_i(d_i+1)}{2} \leq m \tag{2}$$

En combinant les équations (1) et (2), on a

$$\begin{aligned}\sum_{i \in S}(d_i + 1) \cdot \sum_{i \in S} d_i(d_i + 1) &\leq 2m + n \\ &\leq 2\delta m + n - n(2S + 1)\end{aligned}$$

En appliquant l'inégalité de Cauchy-Schwarz avec $a_k = (d_i + 1)$ et $b_k = 1$, on a

$$\begin{aligned}\left(\sum_{i \in S}(d_i + 1) \cdot 1\right)^2 &\leq \left(\sum_{i \in S}(d_i + 1)^2\right) \cdot \left(\sum_{i \in S} 1^2\right) \\ &\leq \left(\sum_{i \in S}(d_i + 1)^2\right)\end{aligned}$$

et on a bien

$$\frac{n}{2\delta + 1} \leq |S|$$

Exercice 22 title.

Solution.

Exercice 23 title.

Solution.

Exercice 24 title.

Solution.

Exercice 25 Algorithme glouton pour SAT. Montrer que l'algorithme admet une performance relative de $\frac{1}{2}$. On pourra utiliser une preuve par réduction.

Solution. On fait une preuve par récurrence.
Dans le cas de base, le résultat est trivial.
Supposons maintenant qu'on a n variables. Soit v la variable dans φ qui apparait le plus. Soit

c_1 le nombre de clauses où v apparaît positivement et c_2 le nombre de clauses où v apparaît négativement. Sans perte de généralité, supposons que $c_1 \geq c_2$. En assignant v à vrai, on satisfait au moins c_1 clauses, il reste alors $c - c_1$ clauses à satisfaire sur les $n-1$ variables restantes, où c est le nombre de clauses satisfaites par une affectation optimale. Par hypothèse de récurrence, l'algorithme satisfait au moins $\frac{1}{2}(c - c_1)$ clauses parmi les $n-1$ variables restantes. Le nombre de clauses satisfaites au total par l'algorithme est alors au moins

$$c_1 + \frac{1}{2}(c - c_1) = \frac{1}{2}c + \frac{1}{2}c_1 \geq \frac{1}{2}c.$$

Exercice 26 title.

Solution.

Exercice 27 title.

Solution.

Exercice 28 title.

Solution.

Exercice 29 title.

Solution.

2.3 Mesure différentielle

Exercice 30 Propriétés liées à la mesure au pire cas.

1. Soit Π un problème de maximisation appartenant à la classe $\mathcal{NPO}$ ayant un ratio δ pour la mesure. Montrer que cela implique que le ratio dans le pire cas est de δ.
2. Sur le problème du TSP, considérons une instance $I = (K_n, \vec{d})$ avec $\vec{d}$ le vecteur des arêtes-distances. Alors, pour n'importe quelle transformation affine

$$\vec{d} \coloneqq \gamma \cdot \vec{d} + \mu \cdot \vec{1}$$

avec $\gamma, \mu \in \mathbb{Q}$ produit une approximation différentielle équivalente à celle obtenue pour I.

Solution.

Exercice 31 Mesure différentielle : borne inférieure pour le problème du Bin Packing.
Montrer que Bin Packing ne peut être résolu par un algorithme polynomial à rapport différentiel minoré par $\frac{n-1}{n}$.
Pour cela, on pourra supposer l'existence d'un tel algorithme et arriver à une contradiction.

Solution. On suppose qu'il existe un algorithme polynomial A pour Bin Packing tel que

$$\frac{n - A(I)}{n - \mathrm{OPT}(I)} \geq \frac{n-1}{n}$$

alors

$$A(I) - \text{OPT}(I) \leq 1 - \frac{\text{OPT}(I)}{n} < 1$$

or $A(I)$ et $\text{OPT}(I)$ sont des entiers, donc $A(I) = \text{OPT}(I)$ et A est un algorithme polynomial pour Bin Packing, ce qui est impossible.

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 Solutions semi-intégrales pour le problème de vertex cover.
Par définition, une solution réalisable x et appelée semi-intégrale si les valeurs appartiennent à l'ensemble $\{0, \frac{1}{2}, 1\}$.

1. Donner la formulation du problème de vertex cover par un programme linéaire en nombres entiers

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 Non-approximation du TSP en fonction de paramètres.
Nous voulons montrer que si $\mathcal{P} \neq \mathcal{NP}$, alors aucun algorithme polynomial pour le problème de la minimisation du VOYAGEUR DE COMMERCE ne peut garantir un rapport d'approximation classique inférieur à $\frac{d_{\max}}{nd_{\min}}$, où $d_{\max}$ et $d_{\min}$ sont respectivement la plus grande et la plus petite des distances sur les arêtes de l'instance du TSP, et n est le nombre de villes.

Pour cela, procéder par l'absurde et utiliser le même principe qu'en cours. A partir d'un graphe quelconque $G = (V, E)$:

1. Construire un graphe complet valué en complétant les arêtes.
2. Executer votre algorithme sur votre instance et conclure sur l'existence d'un cycle Hamiltonien dans G.

Solution.

1. On considère le graphe $G' = (V', E')$ suivant : $V' = V$ et $E' =$ l'ensemble de toutes les paires de sommets de V. On définit la fonction de coût c sur les arêtes de G' de la manière suivante :
$$c(u, v) = \begin{cases} 1 & \text{si } (u, v) \in E, \\ d_{\max} & \text{sinon.} \end{cases}$$
Par simplification, on peut supposer que $d_{\min} = 1$ et on peut poser $d_{\max} = M$.
2. Supposons que nous avons un algorithme polynomial A pour le TSP qui garantit un rapport d'approximation inférieur à $\frac{M}{n}$. Si G contient un cycle Hamiltonien, alors le coût optimal du TSP sur G' est n (car on peut parcourir les arêtes de coût 1). Par conséquent, l'algorithme A doit trouver une solution de coût au plus $\frac{M}{n} \cdot n = M$.
Si G ne contient pas de cycle Hamiltonien, alors le coût optimal du TSP sur G' est au moins $n - 1 + M$ (car on doit utiliser au moins une arête de coût M). Par conséquent, l'algorithme A doit trouver une solution de coût au plus $\frac{M}{n} \cdot (n - 1 + M)$ qui est strictement supérieur à M pour $n > 1$.
On a donc séparé les cas où G contient un cycle Hamiltonien de ceux où il n'en contient pas et on a trouvé un écart de coût, il est donc impossible d'approcher une solution du TSP avec un rapport d'approximation inférieur à $\frac{M}{n}$, ce qui contredit l'existence de l'algorithme

A. Par conséquent, si $\mathcal{P} \neq \mathcal{NP}$, aucun algorithme polynomial pour le TSP ne peut garantir un rapport d'approximation inférieur à $\frac{d_{\max}}{nd_{\min}}$.

Exercice 46 TSP asymétrique.
On rappelle que le problème du TSP asymétrique s'énonce comme suit :

Algorithme 9 – Asymmetric TSP Problem (ATSP)
Input: Un graphe $G = (V, E)$ orienté et muni d'une fonction de coût positive sur les arcs.
Question: Trouver un circuit de cout minimum qui passe par chaque sommet exactement une fois.

Montrer que ATSP est $\mathcal{NP}$-complet et non approximable.

Solution. Le problème du ATSP est un cas particulier du TSP (en imposant des coûts infinis pour les arcs non présents), et que le TSP est déjà connu pour être $\mathcal{NP}$-complet et non approximable. Par conséquent, ATSP hérite de ces propriétés.

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 10 – $P|\cdot|C_{\max}$
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 11 – 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

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

Autre correction possible :
On a

$$C_{\max}^{OPT} \geq \frac{T_{\text{seq}}}{m}$$

et

$$C_{\max}^{OPT} \geq p_j, \ \forall j$$

Si on considère l'agencement des taches on trouve

$$T_{\text{seq}} \geq (C_{\max} - p_j)\, m + p_j$$

et alors

$$\frac{T_{\text{seq}}}{m} \geq C_{\max} - p_j + \frac{p_j}{m}$$

en combinant ces deux inégalités on trouve

$$\begin{aligned} C_{\max}^{LS} &\leq \frac{T_{\text{seq}}}{m} + p_j - \frac{p_j}{m} \\ &\leq \frac{T_{\text{seq}}}{m} + p_j \left(1 - \frac{1}{m}\right) \\ &\leq C_{\max}^{OPT} + C_{\max}^{OPT} \left(1 - \frac{1}{m}\right) \\ &\leq C_{\max}^{OPT} \left(2 - \frac{1}{m}\right) \\ &\leq 2C_{\max}^{OPT} \end{aligned}$$

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.