MPI: Classes de Complexités

Cours

- Problème de décision, d'optimisation, de maximisation/minimisation. Comment transformer un problème d'optimisation en un problème de décision.
- Taille d'une instance.
- Définition d'une classe de complexité. Définition de P, NP, NP-Dur, NP-Complet
- Réduction polynomiale
- k-SAT, SAT, certificat

Question de Cours

- Comment transformer un probleme de maximisation en un problème de décision ?
- Montrer que 3-SAT est NP-Complet

Max 2-SAT

On consdière le problème d'optimisation suivant:

Entrée: φ une formule en FNC **Sortie:** Une valuation φ qui maximise le nombre de clauses satisfaites

1. Donner la version problème de décision
2. Montrer que si k est le nombre de clauses le problème deviens polynomial
3. Montrer que la version décisionnel du problème est NP-Complet

SUBSET-SUM

On considère le problème SUBSET-SUM suivant:

- **Entrée** $n \in \mathbb{N}$ et $a_1, a_2, ..., a_n, S \in \mathbb{N}$
- **Sortie:** Est-ce qu'il existe $I \subseteq [\![1; n]\!]$ tel que $\sum_{i \in I} a_i = S$

1. Pour les suites suivante de $(a_n)_n$ et les valeurs de S suivantes, indiquer si le problème est satisfiable ou non:
   - $(a_n)_n = (1, 2, 4, 8, 16), S = 7$
   - $(a_n)_n = (31, 24, 2, 43, 12, 12, 18), S = 29$
   - $(a_n)_n = (1001, 1010, 101, 100, 11), S = 1111$
2. Montrer que le problème SOMME est dans la classe NP
3. Donner un algorithme qui en $O(nS)$ résoud le problème

On cherche à prouver que le problème précédent est NP-Complet. Pour cela on considère une généralisation du problème sur des k-uplets intitulé SUBSET-SUM-VECT:

- **Entrée** $k, n \in \mathbb{N}$ et $a_1, a_2, ..., a_n, S \in \mathbb{N}^k$
- **Sortie:** Est-ce qu'il existe $I \subseteq [\![1; n]\!]$ tel que $\sum_{i \in I} a_i = S$

4. Montrer que on peut réduire le problème **SOMME-VECT** au problème **SOMME**
5. Montrer que **SOMME-VECT** est NP-Complet à partir de 3-SAT.
6. Est-ce que avec la question 3 on a démontré que P=NP ? Justifier.

INDEPENDANT-SET et CLIQUE

On considère les deux problèmes CLIQUE-MAX et INDEPENDANT-SET suivant:

CLIQUE:

- **Entrée:** Un graphe $G = (V, E)$ non orienté
- **Sortie:** Le plus grand $S \subseteq V$ en cardinal tel que $V^2 \subseteq E$

INDEPENDANT-SET:

- **Entrée:** Un graphe $G = (V, E)$ non orienté et un $k \in \mathbb{N}$
- **Sortie:** Est-ce qu'il existe $S \subseteq V$ tel que $S^2 \cap E = \emptyset$ avec $|S| \geq k$?

1. Donner la version problème de décision de CLIQUE. Montrer que elle est NP.
2. Donner une réduction polynomiale de INDEPENDANT-SET dans la version porblème de décision de CLIQUE.
3. En réduisant depuis 3-SAT, et en considérant un graphe avec $3k$ noeuds avec k le nombre de clauses, montrer que CLIQUE est NP-Complet.

MIN-COLOR-PATH

On considère le problème MIN-COLOR-PATH suivant:

- **Entrée:** $G = (V, E)$ un graphe, $s, t \in V$ deux sommets et $c : V \longrightarrow \mathbb{N}$ une coloration
- **Sortie:** Un chemin $s = x_1, .., x_n$ de s à t qui minimise $\mathrm{Card}(\{c(x_1), ..., c(x_n)\})$

1. Donner la version problème de décision, et montrer qu'elle est NP.
2. Donner un algorithme pour résoudre le problème si G est un cycle.
3. Montrer que le problème de décision associé à MIN-COLOR-PATH est NP-Complet.

On suppose maintenant que l'on rajoute la condition $|\mathrm{Im}(c)| \leq k$ avec $k \in \mathbb{N}$ fixé.

4. Montrer maintenant que le problème est P.

On considère l'algorithme qui prend des couleurs uniformément au hasard dans $\mathrm{Im}(c)$ jusqu'à ce qu'il existe un chemin de s à t

5. Quel est ce type d'algorithme ? Proposer des exemples arbitrairement grand ou cet algorithme renvoie une solution d'éspérence en $O(n)$ au lieu de $O(1)$.

DOMINATING-SET

Soit $G = (V, E)$, on dit que $S \subseteq V$ est un *ensemble dominant* si tous les sommets de V sont soit dans S, soit voisin d'un sommet de S. On note $\gamma(G)$ le cardinal minimum d'un ensemble dominant de G.

On considère le problème DOMINATING-SET suivant:

- **Entrée:** $G = (V, E)$ un graphe
- **Sortie:** $\gamma(G)$

1. Donner la version problème de décision de DOMINATING-SET. Montrer qu'elle est NP.
2. Montrer que $\gamma(G) \leq |V|/2$
3. Montrer que la version problème de décision de DOMINATING-SET est NP-Complet.
4. Montrer que si l'on restrain l'entrée du problème à des arbres, alors le problème est dans P.
5. Pour G un graphe, on note Δ_G son degré maximal. Donner une Δ_G-approximation de DOMINATING-SET.
6. Donner similairement une $\log(|V|)$-approximation. Est-ce qu'une des approximations fait les deux ?

Langages NP-Complets sur $\Sigma = \{a\}$[1]

Un langage L est dit unaire si c'ets un langage sur un alphabet à une lettre, donc si $L \subseteq \{a\}^*$. L est dit *NP-Difficile* si le problème de tester si $x \in L$ est NP-Difficile. Dans ce problème, la taille de l'entrée est $|x|$.

1. On note SAT^+ l'ensemble des entrée positives au problème SAT et S. Montrer que L est NP-Difficile ssi il existe une fonction f de complexité polynomiale telle que

$$\forall \varphi, \varphi \in \mathrm{SAT}^+ \iff f(\varphi) \in L$$

On cherche à montrer le théorème suivant

Théorème de Berman: S'il existe un langage unaire NP-Dur alors $\mathrm{P} = \mathrm{NP}$

Soit $\varphi(x_1, x_2, ..., x_n)$ une instance de SAT. On suppose que L est un langage unaire NP-Difficile.

2. On note $\varphi_\top(x_2, ..., x_n) \coloneqq \varphi(\top, x_1, ..., x_n)$ et $\varphi_\bot(x_2, ..., x_n) \coloneqq \varphi(\bot, x_1, ..., x_n)$. Montrer que

$$\varphi \in \mathrm{SAT}^+ \iff \varphi_\top \in \mathrm{SAT}^+ \vee \varphi_\bot \in \mathrm{SAT}^+$$

On note $|\varphi|$ la taille d'une formule φ, et on suppose que $\forall i, |\top|, |\bot| < |x_i|$

3. Soit φ une formule à n variables. Donner un algorithme qui, pour tout $i \leq n$, construit un ensemble de formules Φ_i tel que toute formule de Φ_i contient $n - i$ variables et $\varphi \in \mathrm{SAT}^+ \iff \exists \varphi \in \Phi_i, \varphi \in \mathrm{SAT}^+$
4. Montrer qu'il existe un polynome p tel que $\sum_{\varphi \in \Phi_i} |\varphi| \leq p(|\varphi|)$
5. En déduire que $\mathrm{P} = \mathrm{NP}$, et donc le théorème de Berman.

[1] Mallory Marin, doc de colles

Arbre couvrant minimum en couleurs (MLST)

Un *multi-graphe* est un graphe dans lequel on peut avoir plusieurs fois une arête entre 2 sommets. On considère le problème MIN-COLOR-TREE suivant:

- **Entrée:** $G = (S, A)$ un multigraphe et $c : A \longrightarrow \mathbb{N}$ une coloration des arrettes.
- **Sortie:** Un arbre couvrant $T = (S, A')$ de G qui minimise $\mathrm{Card}(\{c(e) : e \in A'\})$

On note n le nombre de sommets et m le nombre d'arêtes.

1. Donner la version problème de décision, et montrer qu'elle est NP.
2. Montrer que si toutes les arêtes sont de couleur différentes alors il existe un algorithme polynomial.
3. Soit $k \in \mathbb{N}$ fixé. Montrer que si le nombre de couleurs distinctes de c est au plus k, il existe un algorithme en $O\big(2^k(n+m)\big)$ résolvant le problème.

On considère le problème SET-COVER suivant :

- **Entrée:** $n, m, k \in \mathbb{N}$ et des ensembles $U_1, ..., U_m \subseteq [\![1; n]\!]$
- **Sortie:** Est-ce qu'il existe $I \subseteq [\![1; m]\!]$ avec $|I| \leq k$ tel que $\bigcup_{i \in I} U_i = [\![1; n]\!]$?

4. On admet que le problème SET-COVER est NP-dur. Montrer que le problème de décision associé à MIN-COLOR-TREE est NP-Dur.

Activation de Processus[2]

Soit un système temps réel à n processus asynchrones $i \in [\![1; n]\!]$ et m ressources $\left(r_j\right)_{j \leq m}$. Quand un processus i est actif, il bloque un certain nombre de ressources listées dans un ensemble P_i et une ressource ne peut être utilisée que par un seul processus. On cherche à activer simultanément le plus de processus possible. Le problème de décision `ACTIVATION` correspondant ajoute un entier k aux entrées et cherche à répondre à la question : "Est-il possible d'activer au moins k processus en même temps ?""

1. Soit $n = 4$ et $m = 5$. On suppose que $P_1 = \{r_1, r_2\}$, $P_2 = \{r_1, r_3\}$, $P_3 = \{r_2, r_4, r_5\}$ et $P_4 = \{r_1, r_2, r_4\}$. Est-il possible d'activer 2 processus en même temps ? Même question avec 3 processus.
2. Dans le cas où chaque processus n'utilise qu'une seule ressource, proposer un algorithme résolvant le problème `ACTIVATION`. Évaluer la complexité de votre algorithme.

On souhaite montrer que `ACTIVATION` est NP-complet

3. Donner un certificat pour ce problème.
4. Ecrire en pseudo code un algorithme de vérification polynomial. On supposera disposer de trois primitives, toutes trois de complexité polynomiale :
   - `appartient(c,i)` qui renvoie Vrai si le processus `i` est dans l'ensemble d'entiers `c`.
   - `intersecte(`P_i`,R)` qui renvoie `Vrai` si le processus `i` utilise une ressource incluse dans un ensemble de ressources `R`.
   - `ajoute(`P_i`,R)` qui ajoute les ressources P_i dans l'ensemble `R` et renvoie ce nouvel ensemble.

En théorie des graphes, le problème `STABLE` se pose la question de l'existence dans un graphe non orienté $G = (S, A)$ d'un ensemble d'au moins k sommets ne contenant aucune paire de sommets voisins. En d'autres termes, existe-t-il $S' \subseteq S, |S'| \geq k$ tel que $\forall s, p \in S', (s, p) \notin A$?

5. En utilisant le fait que `STABLE` est NP-complet, montrer par réduction que le problème `ACTIVATION` est également NP-complet.

[2] copie presque exacte CCINP 2023 sujet A

Intersection d'automates[3]

On considère le problème NONEMPTY-REGULAR-INTERSECTION suivant:

Entrée: $A_1, ..., A_n$ une liste d'automate sur $\Sigma = \{a, b\}$
Sortie Est-ce que

$$\bigcap_{i \leq n} L(A_i) \neq \emptyset$$

1. Montrer que NONEMPTY-REGULAR-INTERSECTION est NP
2. Donner une instance du problème tel que le plus petit mot dans l'intersection est de longueur exponentielle en la taille de l'entrée.
3. Montrer que si $\Sigma = \{a\}$ alors le problème est polynomial.
4. Montrer que NONEMPTY-REGULAR-INTERSECTION est NP-Complet à partir de 3-SAT

[3] Algo 1 ENS Lyon TD 8

Sheffer

On définit l'opréateur de sheffer $X \uparrow Y := \neg(X \wedge Y)$ et l'opérateur $X \oplus Y := (X \vee Y) \wedge (\neg X \vee \neg Y)$. Indiquer si les problèmes suivant sont Np-Complet ou Polynomial:

- **Entrée:** Une formule φ avec que des $\oplus$ **Sortie:** Si φ est satisfiable.
- **Entrée:** Une formule φ avec que des $\wedge$ et $\neg$ **Sortie:** Si φ est satisfiable.
- **Entrée:** Une formule φ avec que des $\uparrow$ **Sortie:** Si φ est satisfiable.

Plus dur, essayer de trouver une réduction de SAT (pas 3-SAT) aux formules avec $\uparrow$.