École polytechnique — X2022 — INF412 Fondements de l'informatique

Problèmes NP-complets, problèmes NP-durs

1 Le problème du chemin hamiltonien est NP-complet

Le problème du *chemin hamiltonien* est le suivant : étant donné un graphe orienté, et deux sommets A et B distincts dans ce graphe, dire s'il existe un chemin de A à B qui passe par tous les sommets du graphe, et ne visite chacun qu'une fois.

Question 1.1. *Montrez que le problème du chemin hamiltonien est dans NP.*

Pour montrer que ce problème est NP-complet, il suffit de prouver en outre que 3SAT est plus facile que lui.

On appelle TRIPLETTE un graphe orienté avec trois nœuds d'entrée e_1, e_2 et e_3 et trois nœuds de sortie s_1, s_2 et s_3, qui possède les propriétés suivantes.

1. Pour tout nœud d'entrée e_i, il existe un unique chemin c_i qui part de e_i, finit par l'un des trois nœuds de sortie, et tel que chaque nœud de TRIPLETTE est visité exactement une fois par le chemin ; par ailleurs, c_i finit sur le nœud s_i.
2. Pour toute paire de nœuds d'entrée e_i et e_j, il existe une unique paire de chemins disjoints c_i et c_j, démarrant respectivement de e_i et de e_j, finissant par des nœuds de sortie, et tels que chaque nœud de TRIPLETTE est visité exactement une fois par l'un des deux chemins ; par ailleurs c_i et c_j finissent respectivement sur s_i et s_j.
3. Il existe un unique triplet de chemins disjoints c_1, c_2, et c_3, démarrant respectivement de e_1, e_2, et e_3, finissant par des nœuds de sortie, et tels que chaque nœud de TRIPLETTE est visité exactement une fois par l'un des trois chemins ; par ailleurs c_1, c_2, et c_3 finissent respectivement sur s_1, s_2, et s_3.

Question 1.2. *Définir un graphe orienté TRIPLETTE avec les propriétés ci-dessus [indication : il en existe un à 6 sommets].*

Soit $C_1 \wedge \cdots \wedge C_m$ un problème 3SAT, chaque clause C_i étant une disjonction de trois littéraux formée sur les variables propositionnelles $x_1, \ldots, x_n$. On construit le graphe formé :

— Des nœuds $v_1, \ldots, v_{n+1}$. Les nœuds $v_1, \ldots, v_n$ sont respectivement associés aux variables $x_1, \ldots, x_n$.

— De m copies de TRIPLETTE, chacune correspondant à une clause C_i, avec les 3 entrées et sorties respectives étiquetées par les variables de la clause.

— Pour chaque variable x_i, on répertorie les clauses $C_{i_1}, \ldots, C_{i_p}$ ($p \geq 0$) dans laquelle elle intervient positivement, avec $i_1 < \cdots < i_p$; on connecte v_i à l'entrée correspondante à x_i dans C_{i_1}, la sortie correspondante de C_{i_1} à l'entrée correspondante de C_{i_2}, etc., et la sortie de C_{i_p} est connectée à v_{i+1}. On appelle ces branchements le « sous-graphe positif associé à x_i ».

— On opère exactement de la même manière pour les clauses où elle intervient négativement. On appelle ces branchements le « sous-graphe négatif associé à x_i ».

Question 1.3. *Construisez le graphe associé à la formule* $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_4)$.

Question 1.4. *Dans l'exemple de la question précédente, quel est le chemin hamiltonien associé à la valuation* $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$ *?*

Question 1.5. *Montrez qu'à tout chemin hamiltonien de* v_1 *à* v_{n+1} *on peut associer une valuation de* $x_1, \ldots, x_n$ *satisfaisant* $C_1 \wedge \cdots \wedge C_m$.

Question 1.6. *Montrez qu'à toute valuation de* $x_1, \ldots, x_n$ *satisfaisant* $C_1 \wedge \cdots \wedge C_m$ *on peut associer un chemin hamiltonien.*

2 Langages creux et NP-difficulté

Soit Σ un alphabet fini, et $\mathcal{L} \subseteq \Sigma^\star$ un langage. On dit que $\mathcal{L}$ est *creux* s'il existe un polynôme P tel que pour tout n le nombre de mots de $\mathcal{L}$ de longueur au plus n est borné par $P(n)$. L'objectif de cet exercice est de montrer que si un tel langage est NP-difficile alors $\mathsf{P} = \mathsf{NP}$ (théorème de Mahaney), et d'en tirer quelques conséquences pratiques. On considère donc une fonction F calculable en temps polynomial qui envoie les formules de SAT dans $\mathcal{L}$ et les autres en dehors.

Question 2.1. *Montrer qu'il existe un polynôme D (pour dilatation) croissant tel que pour toute formule ϕ, $|F(\phi)| \leq D(|\phi|)$, où $|w|$ désigne la longueur du mot w.*

Question 2.2. *Montrer que si $F(\phi_1 \vee \phi_2) = F(\phi_1 \vee \phi_3)$ pour trois formules ϕ_1, ϕ_2, ϕ_3 alors la satisfiabilité de $\phi_1 \vee \phi_2 \vee \phi_3$ est équivalente à celle de $\phi_1 \vee \phi_2$.*

Question 2.3. *On suppose que la fonction de taille est telle que $|\phi \vee \psi| = |\phi| + |\psi| + 1$. Montrer que si les formules $\phi_1, \ldots, \phi_n$ ont toutes longueur bornée par N, sont telles que les valeurs de $F(\phi_1 \vee \phi_j)$, $j \in \{2, \ldots, n\}$ sont toutes distinctes et $P(D(2N+1)) < n-1$ alors ϕ_1 n'est pas satisfiable.*

Question 2.4. *Montrer comment, pour toute formule ϕ en les variables $x_1, \ldots, x_n$, on peut construire un arbre, en temps polynomial en $|\phi|$, dont les nœuds sont des formules, dont la racine est ϕ, et tel que pour chaque niveau k de l'arbre,*

1. *il y a r nœuds $\phi_1, \ldots, \phi_r$, où $r \leq P(D(2|\phi|+1))+1$;*
2. *$\phi_1 \vee \cdots \vee \phi_r$ est satisfiable si et seulement si ϕ est satisfiable ;*
3. *seules les variables $x_k, \ldots, x_n$ apparaissent.*

[On rappelle que la satisfiabilité d'une formule $\phi(x_1, \ldots, x_n)$ est équivalente à celle de $\phi(0, x_2, \ldots, x_n) \vee \phi(1, x_2, \ldots, x_n)$.]

Question 2.5. *Conclure.*

Une borne de complexité dans le cas le pire ne renseigne pas tellement sur le paysage de la complexité dans la classe étudiée : cette borne pourrait être atteinte sur quelques instances, alors que toutes les autres sont très faciles. Les questions suivantes montrent que, dans le cas de la satisfiabilité au moins, la situation ne peut pas être trop souvent très simple.

Question 2.6. *Montrer que, si $\mathsf{P} \neq \mathsf{NP}$, il n'existe pas d'algorithme résolvant SAT en complexité polynomiale dans toutes les instances sauf un nombre polynomial d'entre elles. [Indication : raisonner par l'absurde, exhiber un langage creux approprié, et réduire SAT à ce langage.]*

Même en relâchant les attentes concernant l'algorithme, on obtient un résultat négatif.

Question 2.7. *Montrer que, si $\mathsf{P} \neq \mathsf{NP}$, il n'existe pas d'algorithme de complexité polynomiale qui réponde correctement au problème SAT, sauf sur un nombre polynomial d'instances, où il se trompe. [Indication : comme à la question 2.4, on peut construire un arbre qui exploite un tel algorithme et un langage creux à expliciter pour réduire le nombre de formules à chaque niveau, et finalement résoudre SAT en complexité polynomiale.]*