Université de Lille

FST- Département Informatique

ACT

2018–2019
Master1 d'Informatique

Fiche TD-Les propriétés NP-dures, les réductions polynomiales

---

Exercice 1 : Vrai ou Faux

Q 1. Une propriété X est NP. Parmi les affirmations suivantes, quelle est celle qui est vraie?

(Af1) Il n'y a pas d'algorithme polynomial pour décider X.

(Af2) Si X est NP-dure, elle est NP-complète.

(Af3) Il n'y a pas d'algorithme déterministe pour décider X.

Q 2. Pour chaque affirmation suivante, dire si elle est vraie, fausse, vraie si et seulement si $P = NP$, vraie si et seulement si $P \neq NP$. Justifier brièvement.

(Af1) Toute propriété NP est aussi P.

(Af2) Toute propriété P est aussi NP.

(Af3) Une propriété NP est une propriété non polynomiale.

Q 3. Soient P_1, P_2, P_3, Q des propriétés.

Supposons qu'on sache que P_1 est P, P_2 est NP-dur, P_3 NP.

Pour chacune des affirmations suivantes, dire si elle vous semble "Vraie", "Fausse" ou "fausse si $NP \neq P$":

(Af1) Si P_1 se réduit polynomialement en Q, Q est P.

(Af2) Si Q se réduit polynomialement en P_1, Q est P. (Af3) Si Q se réduit polynomialement en P_2, Q est NP-dure. (Af4) Si P_2 se réduit polynomialement en Q, Q est NP-dure. (Af5) Si Q se réduit polynomialement en P_3, Q est NP. (Af6) Si P_3 se réduit polynomialement en Q, Q est NP.

Exercice 2 : Les chevaliers de la Table Ronde

n chevaliers veulent s'installer autour d'une table ronde à n places mais certains chevaliers sont ennemis. On veut savoir si on peut les installer autour de la table sans que deux ennemis se retrouvent côte à côte -la donnée du problème est donc les n chevaliers et toutes les "paires" d'ennemis.

Q 1. Montrer que le problème est NP.

Q 2. Montrer que le problème est $NP-$ dur en utilisant le fait que le problème de l'existence d'un cycle Hamiltonien dans un graphe est $NP-$complet.

Q 3. Montrer que le problème est $NP-$complet.

Exercice 3 : Noyau d'un graphe

Le *noyau* d'un graphe dirigé est un ensemble de sommets indépendant - aucun arc ne relie deux sommets du noyau, et si tout sommet est soit dans le noyau soit a un successeur dans le noyau.

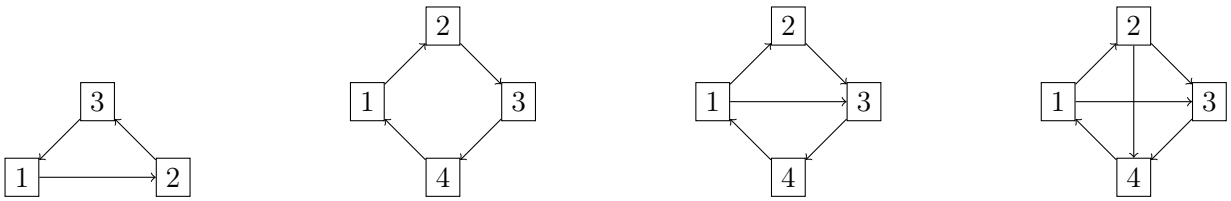
Le noyau a par exemple des applications pour certains jeux à deux joueurs (Nim, échecs, ...): un sommet correspond à une configuration du jeu, un arc relie une configuration à ses successeurs, une configuration perdante n'a pas de successeur.; si un joueur peut jouer un coup qui l'amène dans une position du noyau, il a une stratégie gagnante (Pourquoi?).

Le problème que nous appellerons ici le problème du noyau sera donc:

Entrée: $G = (S, A)$ - -un graphe dirigé

Sortie: Oui, Ssi il existe un noyau de G.

Q 1. Parmi les exemples suivants, lesquels ont un noyau?



Q 2. Montrer que le problème du noyau est NP.
Q 3. Montrer que 3-Sat se réduit polynomialement dans le problème du noyau. Qu'en conclure pour le problème du noyau?

Exercice 4 : Processus

Rappel du problème (cf feuille de TD précédente):

Soit un système temps réel avec n processus asynchrones et m ressources. Quand un processus est actif, il bloque un certain nombre de ressources et une ressource ne peut être utilisée que par un seul processus. On cherche à activer simultanément k processus.

Le problème de décision DecProc est donc:

Donnée:

n, le nombre de processus

m le nombre de ressources

pour chaque processus i, la liste P_i des ressources qu'il bloque.

k le nombre de processus que l'on souhaite activer

Sortie:

Oui, si on peut activer k processus simultanément, non sinon.

Q 1. En utilisant le fait que le problème Independent Set, montrer que le problème DecProc est NP-dur.

Rappel: Le problème qui consiste à tester l'existence d'un ensemble de k noeuds indépendants (i.e. non reliés) dans un graphe est $NP-$complet. Formellement, le problème de décision Independent Set est:

Entrée:

$G = (S, A)$ - -un graphe

k - -un entier

Sortie: Oui, si il existe un sous-ensemble indépendant de cardinal k, i.e. une partie I de S de cardinal k telle qu'aucun arc ne relie deux sommets de I ($(I \times I) \cap A = \emptyset$).

Q 2. Que pensez-vous de la complexité du problème d'optimisation associé OptProc:

Donnée:

n, le nombre de processus

m le nombre de ressources

pour chaque processus i, la liste P_i des ressources qu'il bloque.

Sortie:

k maximum tel qu'on puisse activer k processus simultanément.

Exercice 5 : Programmation Linéaire en entiers

Un problème de Programmation linéaire en entiers sera du type:

Donnée:

A une matrice $m \times n$ d'entiers (relatifs),

B un vecteur de m entiers (relatifs).

Sortie:

Oui, si il existe un vecteur de n entiers(relatifs) X tel que $AX \leq B$, Non sinon.

Montrer que 3-CNF-SAT (le problème de la satisfiabilité d'une expression booléenne sous forme conjonctive) se réduit polynomialement dans le problème de Programmation linéaire en entiers. Qu'en déduire?

Exercice 6 : Chemin Hamiltonien: de la décision à l'optimisation

Soit le problème de décision $DecHam$

Entrée: G, un graphe non dirigé, A et B deux sommets du graphe

Sortie: Oui, si il existe un chemin dans G partant de A arrivant en B et visitant une et une seule fois chaque sommet

Soit maintenant le problème $ConstrHam$

Entrée: G, un graphe non dirigé, A et B deux sommets du graphe

Sortie: Un chemin partant de A arrivant en B et visitant une et une seule fois chaque sommet si il en existe un, et "Non" sinon

Q 1. Supposons qu'on ait un algorithme polynomial pour $ConstrHam$: existerait-il un algorithme polynomial pour $DecHam$?
Q 2. Réciproquement, supposons qu'on ait un algorithme polynomial pour $DecHam$: comment en déduirait-on un algorithme polynomial pour $ConstrHam$?

Note Biblio: Un cas particulier de ce problème est celui du "KnightTour" où le but est de parcourir toutes les cases d'un échiquier en se déplaçant avec un cavalier, problème aussi appelé problème de Rudrata, poéte du Cachemire du 9e siècle qui posa la question dans un poème.

Exercice 7 : Histoire de Cliques

Rappel: dans un graphe, une *clique* est un ensemble de sommets qui sont **tous** reliés deux à deux- chaque paire est reliée. Le problème Clique consiste à déterminer s'il existe dans un graphe donné une clique de cardinal égal à un entier k donné. Le problème de décision Clique est donc défini par:

Entrée:
$G = (S, A)$ - -un graphe
k - -un entier
Sortie: Oui, si il existe une clique de cardinal k, i.e. une partie C de S de cardinal k telle que toute paire de sommets de S soit reliée par un arc de A, Non sinon.

Clique est connu NP-complet.

Soient les quatre problèmes suivants:

1000-Clique
Entrée: $G = (S, A)$ –un graphe
Sortie: Oui, si il existe une clique de cardinal égal à 1000, Non sinon.

Demi-Ind
Entrée: $G = (S, A)$ –un graphe
Sortie: Oui, si il existe un sous-ensemble indépendant de cardinal au moins $|S|/2$.

Double-Clique
Entrée:
$G = (S, A)$ –un graphe
k - -un entier
Sortie: Oui, si il existe deux cliques distinctes de cardinal égal à k, Non sinon.

Presque-Clique
Entrée: $G = (S, A)$ –un graphe.
Sortie: Oui, si il existe une clique de cardinal égal à $|S| - 1$ ($|S|$ est le cardinal de S), Non sinon.
Q 1. Parmi ces quatre problèmes, 1000-Clique, Demi-Ind, Double-Clique, Presque-Clique, quels sont les problèmes NP-durs, quels sont les problèmes P? Justifier.
Q 2. - *1 pt* - Les quatre problèmes sont-ils NP? Justifier.

Exercice 8 : Paquets logiciels

Préliminaire: Cet exercice est inspiré de la problématique d'installation des paquets logiciels et en particulier d'une infime partie des travaux de L'Initiative pour la Recherche et l'Innovation sur le Logiciel Libre, l'IRILL: c'est une version très simplifiée et schématique de la problématique: voir le site de l'IRILL, http://www.irill.org/.

On supposera dans l'exercice qu'un paquet est identifié par un numéro, un entier positif.

Le cadre général: Pour un paquet logiciel P on connait:

- l'ensemble des paquets avec qui il est en conflit, c.à.d. qui ne peuvent être installés en même temps que lui.
- un ensemble de dépendances qui est un **ensemble d'ensembles** de paquets: si les dépendances de P sont $\{E_1, \ldots, E_k\}$, cela signifie que pour une installation réussie de P, il faut pour chaque i, qu'au moins un paquet de E_i soit installé. Par exemple, si les dépendances sont $\{\{P_1, P_4\}, \{P_2\}, \{P_1, P_3\}$, P_1, P_2 ou P_4, P_2, P_3 suffisent pour installer le paquet, P_4, P_2 ne suffisent pas.

Ces informations sur le paquet P sont incluses dans le paquet P. On supposera dans l'exercice qu'un paquet est limité à ces informations: un paquet sera donc la donnée d'un triplet $(Id, Conf, Dep)$ avec Id son identifiant, $Conf$ la donnée de l'ensemble des identifiants des paquets en conflit avec P et Dep l'ensemble des dépendances de P. La taille d'un paquet est donc ici la taille de sa représentation.

Par exemple $(2, \{1, 7\}, (\{3, 4\}, \{5\})$ correspond au paquet de numéro 2, qui est en conflit avec les paquets 1 et 7 et qui nécessite d'une part le paquet 3 ou le paquet 4, d'autre part le paquet 5.

On supposera disposer des primitives sur les ensembles (appartenance d'un élément à un ensemble, intersection, union, complémentaire, ...).

On supposera pour simplifier que les conflits sont donnés explicitement de façon symétrique: si le paquet Q se dit en conflit avec le paquet R, le paquet R se dit en conflit avec le paquet Q.

Une **installation** I est juste un ensemble de paquets. On dit qu'elle est **correcte** si elle respecte les contraintes de conflit - elle ne contient pas deux paquets en conflit-, et de dépendance - pour tout paquet de I, les contraintes de dépendances sont vérifiées: si les dépendances du paquet sont $\{E_1, \ldots, E_k\}$, $E_i \cap I$ est non vide pour tout i, $1 \leq i \leq k$. On va étudier la complexité de différents problèmes liés à l'installation de paquets.

Q 1. *Installation immédiate*

Soit le problème suivant:

Entrée: un paquet $P = (id, Conf, Dep)$, une installation correcte I.

Sortie: Oui, Ssi P peut être installé directement, i.e. $I \cup P$ est-elle correcte. Non , sinon.

Montrer que le problème est P.

Q 2. *Installation correcte*

Soit le problème suivant:

Entrée: une installation I.

Sortie: Oui, Ssi I est correcte. Non , sinon.

Montrer que le problème est P.

Q 3. *Installation de k paquets (problème pas trés réaliste)*

Soit maintenant le problème de décision suivant $k - Inst$:

Entrée:

n, entier positif

$P_1, \ldots, P_n$, n paquets.

k, entier , $0 \leq k \leq n$

Sortie: Oui, Ssi on peut choisir k paquets distincts parmi $P_1, ..., P_n$ qui forment une installation correcte.

Q 3.1. Montrer que le problème $k - Inst$ est NP.

Q 3.2.

Montrer qu'$Independent$ se réduit polynomialement dans le problème $k - Inst$. Qu'en déduire?

Q 4. *Installation possible?*

Soit maintenant le problème de décision $Poss$ suivant:

Entrée: un ensemble de paquets disponibles $Dispo$, un paquet P de $Dispo$

Sortie: Oui, Ssi il existe une installation correcte de P, i.e. $I \subset Dispo$ qui soit correcte et qui contienne P.

Q 4.1. Montrer que le problème $Poss$ est NP.

Q 4.2. Supposons qu'il n'y ait aucun conflit, juste des dépendances. Quelle est la complexité de la restriction du problème $Poss$ à ce cas?

Q 4.3. Montrer qu'$Independent$ se réduit polynomialement dans le problème $Poss$.

Qu'en déduire pour la complexité du problème $Poss$?

Q 5. Que pensez-vous de la complexité du problème DecProc si chaque processus utilise une seule ressource?

Q 6. Que pensez-vous de la complexité du problème DecProc si chaque ressource est utilisée par au plus deux processus?