REPUBLIQUE DU CAMEROUN
REPUBLIC OF CAMEROON
Peace-Work-Fatherland
****
UNIVERSITE DE DSCHANG
UNIVERSITY OF DSCHANG
Scholar-Thesaurus DschangensisIbiCordum
BP 96, Dschang(Cameroun)- Tel/Fax:(237)232 45 1381
Site web: http://www.univ.dschang.org/iutfv

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE
FOTSO VICTOR DE BANDJOUN
FOTSO VICTOR UNIVERSITY INSTITUTE OF YECHNOLOGY
Département de Mathématiques- et Informatique
Departement of Computer Engineering
BP 134, Bandjoun- Tel. / Fax (237) 699 31 61 30 /670 64 23 92
Website: http://www.univ.dschang.org/iutfv

EXPOSÉ DE COMPLEXITÉS ET ALGORITHMIQUES AVANCÉES

THEME :**NP-COMPLETUDE**

Effectué par le **GROUPE 5**

Membres du Groupe

NOMS ET PRENOMS	**Matricule**
HAPI TCHAKOUNTIO ROMÉO JOËL	CM-UDS-17SCI0936
NGUETOUM LIKEUFACK Anderson	CM-UDS-18SCI0680
FOKA SIN-DZE GODWILL MC JORDAN	CM-UDS-18SCI0096
KANA GUIMAPI Yvan	CM-UDS-18SCI1700

Sous la coordination de :

Dr. KENGNE Vianney

Année académique 2021 / 2022

SOMMAIRE

Liste des figures

A. Cours

Introduction

Dans l'étude de la complexité des problèmes, le point important qui concerne aussi bien le théoricien que le praticien est de savoir si, pour un problème donné, il existe un algorithme de Complexité polynomiale pour le résoudre c'est-à-dire le temps d'exécution dans le pire des cas est en O(n^k) pour une certaine constante k.

Jusqu'ici pour des algorithmes tels que ceux des tris par exemple, nous avons constaté que certains s'exécutaient en temps polynomial en fonction de la taille des données en entrées, il est naturel de se demander si est-ce tous les problèmes qui admettent un algorithme dont l'exécution se fait en temps polynomial en fonction de la taille des données. La réponse est Non car depuis la conférence ***<<The complexity of theorem-proving procedures>>*** en 1971 aucun participant jusqu'à nos jours n'a pu démontrer ou infirmer cette assertion. S'il est vrai que la résolution de certains problèmes se fait en temps polynomial, qu'en est-il de ceux dont l'exécution ne se fait pas en temps polynomial généralement appelés problèmes Non Déterministe Polynomiale(NP) voir Non Déterministe Polynomial Complet (NP-Complet), donc nous verrons qu'il existe certains problèmes dont la complexité s'évaluent a l'ordre exponentiel et d'autres dont seulement la vérification d'une solution se fait en temps polynomial, notamment les problèmes NP complet.

Notre objectif dans ce chapitre serait de passer en revu les différentes classes de complexités à savoir : Les classes **P, NP, NP-Complet** avec des exemples a l'appui. Par la suite nous allons nous concentrer sur les techniques de réductions pour montrer l'improbabilité d'existence d'algorithme efficace pour résoudre un problème. En fin nous présenterons quelques exemples de problèmes **NP-Complet**.

Définition formelle

Un problème de décision est une question mathématique dont la réponse est soit « oui », soit « non ». Les logiciens s'y sont intéressés à cause de l'existence ou de la non-existence d'un algorithme répondant à la question posée. Les problèmes de décision interviennent dans deux domaines de la logique : la théorie de la calculabilité et la théorie de la complexité. Certains problèmes de décision décidables sont cependant considérés comme non traitables en pratique pour des raisons de trop grande complexité des calculs. La théorie de la complexité des algorithmes donne une hiérarchie des complexités formelles.

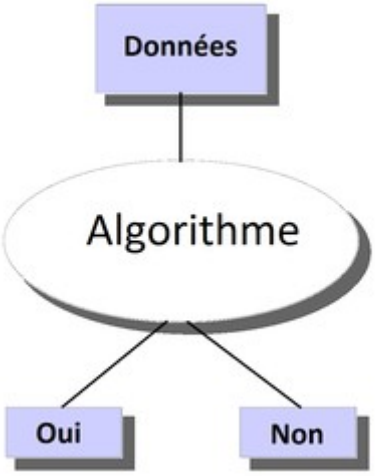


Figure 1: Illustration d'un problème de décision

Par abus de langage, on qualifie souvent de « NP-complets » des problèmes d'optimisation et non des problèmes de décision. Ainsi, on dira que le problème du voyageur de commerce, qui consiste à trouver un plus court-circuit passant une seule fois par chacun des sommets d'un graphe connexe fini, est NP-complet. Cela se justifie par le fait qu'on peut transformer tout problème d'optimisation en problème de décision et réciproquement.

Un vérificateur V est un algorithme qui prend une information en plus (certificat)
pour vérifier qu'une instance est positive.

Les classes P et NPC sont incluses dans NP . Si $p \neq NP$, elles sont disjointes et il existe des problèmes NP-intermédiaires.

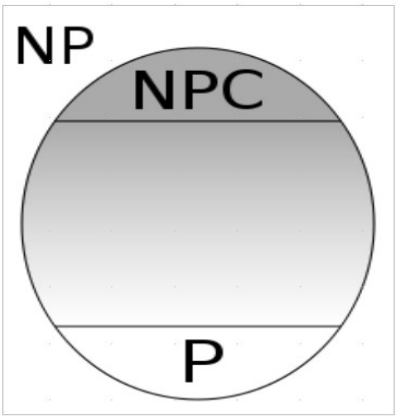


Figure 2: Classes de complexités algorithmiques

En particulier, un problème NP-complet est considéré comme tellement difficile qu'il est conjecturé qu'il n'existe pas d'algorithme efficace qui puisse le résoudre.

Exemple : Graphe Hamiltonien

Problème de décision :

Données: un graphe non-orienté G = (V, E).
Question: G a-t-il un cycle hamiltonien ?

- Le certificat correspond à une suite S de sommets. Le vérificateur V vérifie que S
  est un cycle et qu'il traverse chaque sommet une unique fois. V fonctionne bien en
  temps polynomial.
- Graphe Hamiltonien est dans NP.

I. Historique

Le concept de NP-complétude a été introduit en 1971 par Stephen Cook dans une communication intitulée <<**The complexity of theorem-proving procedures>>** (La complexité des procédures de démonstration de problèmes), bien que le mot « NP-complet » n'apparaisse pas explicitement dans l'article. Lors de la conférence à laquelle il a été présenté, une discussion acharnée a eu lieu entre les chercheurs présents pour savoir si les problèmes NP-complets pouvaient être résolus en temps polynomial sur machine de Turing déterministe. John Hopcroft a finalement convaincu les participants que la

question devait être remise à plus tard, personne n'ayant réussi à démontrer ou infirmer le résultat. La question de savoir si P = NP n'est toujours pas résolue. Elle fait partie des problèmes du prix du millénaire, un ensemble de sept problèmes pour lesquels l'Institut de mathématiques Clay offre un prix d'un million de dollars. Il « suffirait » de trouver un seul problème NP qui soit à la fois NP-complet et P pour démontrer cette hypothèse, ou d'exhiber un seul problème NP qui ne soit pas dans P pour démontrer sa négation. Le résultat de l'article de Cook, démontré de manière indépendante par Leonid Levin en URSS, est maintenant connu sous le nom de théorème de Cook-Levin. Ce théorème affirme qu'il existe un problème NP-complet. Cook a choisi le problème SAT et Levin un problème de pavage. En 1972, Richard Karp a prouvé la NP-complétude de plusieurs autres problèmes fondamentaux en informatique et très disparates, connus comme la liste des 21 problèmes NP-complets de Karp. Depuis, on a démontré la NP-complétude des milliers d'autres problèmes.

II. Présentation des classes

1. La classe P(Polynomial)

Un algorithme est polynomial si et seulement si quelle qu'en soit la complexité de cet algorithme il existe un entier K tel que la complexité en temps dans le pire des cas de cet algorithme est dominé par $O(n^k)$, avec n la taille de données en entrée. Pour qu'un problème a le droit d'appartenir à cette classe, il faut qu'il existe un algorithme permettant de résoudre ce problème en $O(n^k)$ au pire des cas.

Exemple : Le problème Prime

Soit un ensemble B possédant n entier naturel classé en désordre (n appartient N) $b=\{0,10,28,13,45,50,6,80,49,100\}(n=100)$ .Il est question dans cet exercice de ressortir tous les nombres premiers de l'ensemble B (c'est ce qu'on appelle le problème Prime.

Solution

Tout d'abord notre algorithme doit parcourir le tableau en vérifiant tous les nombre qui ne sont divisibles que par 1 et par lui-même et c'est cette liste qui sera retourné.

En évaluant la complexité on a :

- Le parcours du tableau se fait en $O(n)$, n étant le nombre chiffre
- La vérification du nombre de diviseur lui à son tour s'effectuant en $O(k)$, k étant le nombre dont on veut obtenir son nombre de diviseur.

De tous ceci on obtient $T(n)=O(n)+O(k)$ Or n est supérieur a k alors on a la complexité $T(n)=O(n)$.

D'où Prime est un problème de la classe P

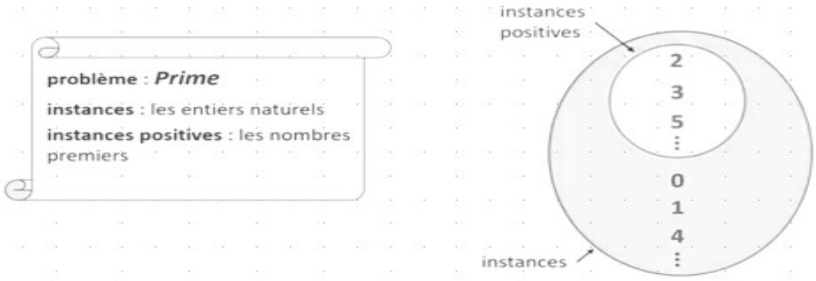


Figure 3: Problème prime

En guise d'autres exemples, nous pouvons citer : le problème du plus court chemin (Dijkstra) et le problème de la détermination de l'arbre de recouvrement de poids minimum (Prim et Kruskal).

2. La classe NP(Non déterministe Polynomial)

NP signifie Non déterministe Polynomial. Pour qu'un problème Q appartienne à cette classe, il faut qu'il existe en plus de l'ensemble d'instance de Q un ensemble de donnée appelé certificat et un algorithme de décision appelé vérificateur acceptant en entrée une instance et un certificat de Q tel qu'un certain nombre de condition soit vérifié

- Le vérificateur doit avoir une complexité polynomiale en fonction de la taille de l'instance placée en entré.
- Si l'instance placée en entrée est négative, alors quel que soit le signe du certificat, le vérificateur doit produire la valeur fausse
- Si l'instance placée en entré est positive, alors il doit exister au moins un certificat pour lequel le vérificateur produit la valeur vraie.

En guise d'exemple le problème sorted est un exemple de présentation de la classe NP ; car il existe en effet pour ce problème un solveur polynomial qui

peut être utilisé comme vérificateur capable de fonctionner sans certificat ou avec un unique certificat vide qui ne sera pas pris en compte.

Théorème : $P \subseteq NP$

Par contre il existe dans la classe NP les problèmes dont on ne peut résoudre en temps polynomial

Exemple :

Dans le problème du circuit hamiltonien, étant donné un graphe orienté G =(S, A), une solution certifiée serait une suite $(v_1, v_2, v_3, \ldots, v_{|S|})$ de |S| sommets. Il est facile de vérifier en temps polynomial que $(v_i, v_{i+1}) \in A$ pour $i = 1,2,3,\ldots|S|-1$ et que $(v_{|S|}, v_1) \in A$ aussi.

3. La classe NP-Complet(Non déterministe polynomial complet)

La classe NP-complet contient les problèmes les plus difficiles de NP. Ces problèmes semblent vraiment ne pas faire partie de P (bien que à l'heure actuelle on ne possède pas une réponse définitive, voir la question irrésolue P = NP ?). Si nous sommes capables de trouver un moyen de résoudre efficacement (algorithme polynomial) un problème de NP-complet, nous pouvons alors utiliser cet algorithme pour résoudre tous les problèmes de NP.

En effet, un problème X appartient à NP-complet s'il est dans NP et si tout autre problème dans NP peut s'y réduire. On n'en déduit qu'une méthode "assez facile" pour prouver qu'un nouveau problème appartient à NP-complet est de montrer d'abord qu'il est dans NP, ensuite le réduire en un problème connu dans NP-complet. Par conséquent il est très utile de connaître une variété de problèmes NP-complets.

III. Théorie de la NP-complétude

1. Présentation

La raison qui pousse le plus les théoriciens de l'informatique à croire que P ≠ NP est sans doute l'existence de la classe des problèmes «NP-complets ». Cette classe possède une propriété surprenante : si un problème NP-complet peut être résolu en temps polynomial, alors tous les problèmes de NP peuvent être résolus en temps polynomial, autrement dit, P = NP. Pourtant, malgré des années de recherches, aucun algorithme polynomial n'a jamais été découvert pour quelque problème NP-complet que ce soit.

Les langages NP-complets sont, dans un sens, les langages les plus «difficiles » de NP. La réductibilité en temps polynomial va donc nous permettre de comparer des problèmes.

Pour montrer qu'un problème de décision A est NP-complet il faut :

- Montrer que A appartient à la classe NP. C'est-à-dire que le certificat (ou la solution)

Peut-être vérifié en temps polynomial. Généralement pour ce faire il suffit de proposer algorithme polynomial.

- Montrer que B ≤p A, où B est un autre problème NP-complet. Le symbole ≤p signifie "B

se réduit en temps polynomial à A". Autrement dit, pour toute instance I_B du problème B on peut construire en temps polynomial une instance I_A du problème A tel que I_B admet comme réponse OUI si et seulement si I_A , admet comme réponse OUI.

2. Exemple : problème de 3-Coloration

On rappelle qu'une coloration de sommets d'un graphe est une application qui associe à chaque sommet une couleur (un nombre) tel que deux sommets adjacents n'ont pas la même couleur. Le problème de décision k-COLORATION (ou k-COL en plus simple) est le suivant :

INSTANCE : Un graphe G.

QUESTION : Est-ce que G peut être colorié avec au plus k couleurs ?

1. Prouver que k-COL appartient à la classe NP.

Remarquez que 1-COL et 2-COL sont des problèmes qui peuvent être résolus en temps polynomial, voire linéaire…

On souhaite prouver que 3-COL est un problème NP-complet. Pour ce faire nous allons utiliser 3-SAT.

SAT est un problème qui consiste en un ensemble $X=\{X_1,X_2,\ldots,X_n\}$ de variables Booléennes et un ensemble $C=\{C_1,C_2,\ldots,C_n\}$ de clauses. Chaque clause C_i contient un ensemble de littéraux X_i ou X_j avec un OU entre eux. Voici un exemple de clause : $C_i=(X_j\vee X_k\vee X_1)$. Les clauses sont reliées entre elles avec un ET : $C_i=(C_1\wedge C_2\wedge\ldots\wedge C_m)$.

Le problème de décision est le suivant :

INSTANCE : Un ensemble de variables booléennes X et un ensemble de clauses C.

QUESTION : Est-ce que C est satis-fiable ? Plus précisément est-ce qu'il existe une assignation de variables tel que chaque clause de C est VRAI.

Dans le cas de 3-SAT la restriction est qu'il y a au plus 3 littéraux par clause.

Maintenant nous allons montrer comment on peut encoder l'instance I de 3-SAT dans un graphe G tel que I est satis-fiable si et seulement si G admet une coloration avec au plus

3 couleurs.

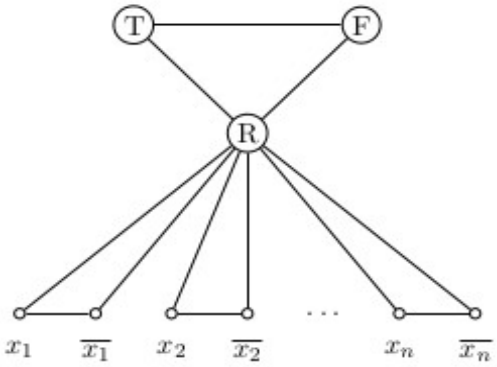


Figure 4: Réduction du problème 3-SAT en 3-coloriage: étape 1

2. L'ensemble des variables de I ,va être représenté par le graphe donné dans la Figure 4. Sans perte de généralité, nous allons supposer que le triangle est colorié tel que c'est indiqué dans la figure. Quelles peuvent être les couleurs de $X_1,X_2,\ldots,X_n$?
3. Pour chaque clause dans l'instance de 3-SAT on va construire le graphe donné dans la Figure 5. Dans l'exemple de la figure la clause est $(X_j\vee X_k\vee\bar{X}_1)$. Les sommets X_i ou $\bar{X}_1$ ainsi que le sommet colorié avec T sont les mêmes que ceux dans la Figure 4.

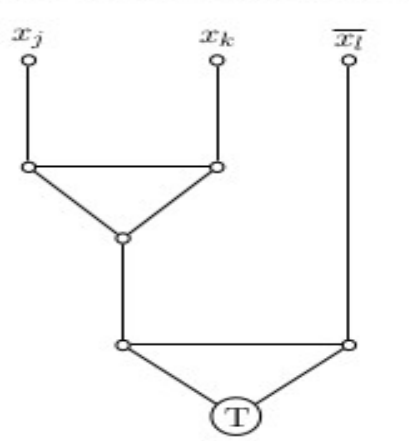


Figure 5: Réduction du problème 3-SAT en 3-coloriage: étape 2

En utilisant la réponse à la question précédente qu'en déduisez-vous ?

4. Montrer que si I est satis-fiable alors il existe une coloration de G avec au plus 3 couleurs.
5. Montrer que si G admet une coloration avec au plus 3 couleurs alors I est satis-fiable.

IV. Réduction polynomiale

1. *Notion de réduction polynomiale*

Une réduction polynomiale est un outil d'information théorique, plus particulièrement de théorie de la complexité. C'est une classe particulière de réductions particulièrement importante, notamment pour le problème $P=NP$. Elle permet de montrer qu'un problème n'est pas plus difficile ou plus facile qu'un autre, elle s'applique même quand les deux problèmes sont des problèmes de décision. Nous exploiterons cette idée dans la quasi-totalité des démonstrations de NP-complétude, et ce de la façon suivante. Soit un problème de décision, disons A, que l'on voudrait résoudre en temps polynomial. L'entrée d'un problème particulier est dite instance de ce problème;
Supposons maintenant qu'il y ait un autre problème de décision, disons B, que nous savons déjà comment résoudre en temps polynomial. Enfin, supposons que nous ayons une procédure qui transforme toute instance a de A en une certaine instance b de B et qui a les caractéristiques suivantes :

- La transformation prend un temps polynomial.

- Les réponses sont les mêmes. C'est-à-dire, la réponse pour a est « oui » si et seulement si la réponse pour b est « oui ».

  Une telle procédure porte le nom de réduction à temps polynomial et, comme le montre la figure ci-dessous, elle donne un moyen de résoudre le problème A en temps polynomial : Étant donné une

- Instance a de A, utiliser une réduction à temps polynomial pour la transformer en une instance b de B.
- Exécuter l'algorithme de décision à temps polynomial de B sur l'instance b.
- Utiliser la réponse pour b comme réponse pour a.
- Si chacune de ces étapes prend un temps polynomial, il en est de même de l'ensemble ; on a donc un moyen de décider pour a en temps polynomial. Autrement dit, en « réduisant » la résolution du problème A à celle du problème B, on se sert de la « facilité » de B pour prouver la « facilité » de A.

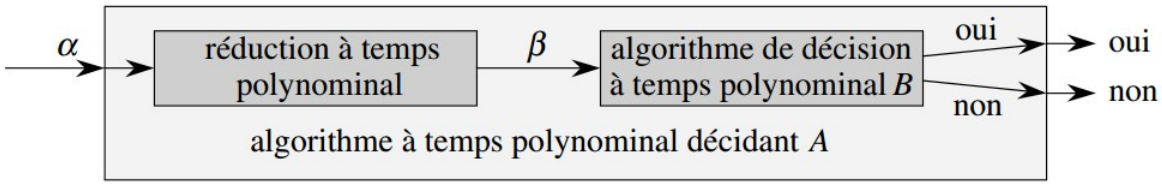


Figure 6: Utilisation d'une réduction à temps polynomial pour résoudre un problème

En nous rappelant que la NP-complétude consiste à démontrer le niveau de difficulté d'un problème et non son niveau de facilité, nous pouvons utiliser des réductions à temps polynomial, en sens inverse, Pour prouver qu'un problème est NP-complet. Poussons l'idée un cran plus loin et montrons comment nous pourrions employer des réductions à temps polynomial pour prouver qu'il ne peut exister d'algorithme à temps polynomial pour un certain problème B. Supposons que l'on ait un problème de décision A pour lequel on sait déjà qu'il ne peut pas exister d'algorithme à temps polynomial. (Nous laisserons de côté, pour l'instant, le problème de savoir comment trouver un tel problème A.) Supposons, en outre, que l'on ait une réduction à temps polynomial qui transforme des instances de A en instances de B. On peut alors utiliser un raisonnement simple par l'absurde pour prouver qu'il ne peut pas exister d'algorithme à temps polynomial pour B. Supposons le contraire, c'est-à-dire que B ait un algorithme à temps polynomial. Alors, en utilisant la méthode illustrée à la Figure 6, on aurait un moyen de résoudre A en temps polynomial, ce qui est contraire à l'hypothèse selon laquelle il n'y a pas d'algorithme à

temps polynomial pour A. Pour la NP-complétude, on ne peut pas partir du principe qu'il n'y a absolument pas d'algorithme à temps polynomial pour le problème A. Cependant, la méthodologie de la démonstration est la même : nous prouvons que le problème B est NP-complet en supposant que le problème A est lui aussi NP-complet.

2. Définition formelle

Soient A et B deux problèmes de décision. On dit que A se réduit pronominalement à B et on note $A \leq p\, B$ s'il existe une fonction $f : \sum * \sum * ¿$ calculable en temps polynomial telle que $x \in A ssi\, f(x) \in B$.

NP-complet

A est NP-Complet si :

- $A \in NP$
- A est aussi difficile que n'importe quel problème NP c'est-à-dire $B \leq_p A, B \in NP$

Tout problème P ∈ NP ; en effet, si un problème appartient à P alors on peut le résoudre en temps polynomial même si l'on ne nous donne pas de solution ; d'où la relation $P \subseteq NP$; Mais on n'a pas encore pu prouver que NP = P car bien qu'on puisse vérifier en temps polynomial toute solutions proposées, on ne sait pas en trouver efficacement. Ainsi la classe NP regroupe la classe NP et la classe NPC (voir la figure ci-dessous).

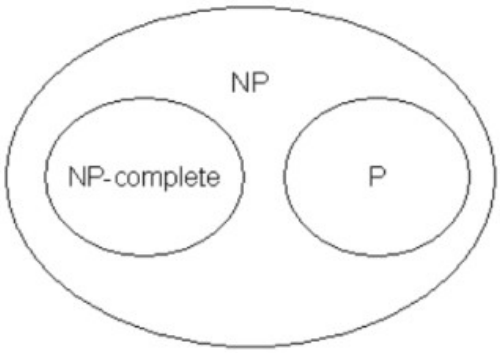


Figure 7: Structure de classe

V. Quelques problèmes NP-complet

Le véritable problème dans cette partie du cours est de se poser la question à savoir lorsqu'une solution à un problème est rapidement vérifiable peut-elle être rapidement trouvée ?

Un problème NP est complet si elle est en NP et si le problème le plus difficile . En d'autres termes, nous pouvons dire qu'une langue L est NP-complet si les déclarations suivantes sont remplies :

1. L est NP
2. Pour chaque langue L « dans NP il y a une réduction polynomiale de L » à L

Ce qu'on appelle en particulier Karp-complet, il peut être considéré comme un cas particulier de Cook-complet qui définit un problème NP-complet si, étant donné un oracle pour le problème P, c'est un mécanisme capable de répondre à toute demande sur l'appartenance à une chaîne P dans une unité de temps, il est possible de reconnaître en temps polynomial une langue dans NP. La définition de la cuisson est plus complète assez général pour inclure compléments problèmes NP-complets dans la classe des problèmes NP-complets.« Réductible » signifie que pour chaque problème L, il y a une réduction polynomiale, un algorithme déterministe qui transforme les instances l ∈ L dans les cas c ∈ C, de sorte que la réponse à c il est oui si et seulement si la réponse à l ll est oui. Pour prouver qu'un problème NP A est en fait un problème NP-complet est suffisant pour montrer qu'un problème NP-complet est déjà connu est réduit à A.

Une conséquence de cette définition est que si nous avions un algorithme de temps polynomial C, Nous pourrions résoudre tous les problèmes de NP en temps polynomial.

Cette définition a été donnée par Stephen Cook, dans un document intitulé « La complexité des procédures théorème-expérimentation » aux pages 151-158 de Actes du 3e Symposium ACM annuel sur la théorie de l'informatique en 1971, bien que le terme NP-complet Il ne figure nulle part dans son journal. Lors de cette conférence de l'informatique, il y avait un débat houleux entre les informations scientifiques si les problèmes NP-complets pourraient être résolus en temps polynomial par une machine de Turing déterministe. John Hopcroft Il a convaincu tous présents à la conférence de mettre de côté cette question pour reprendre plus tard, puisque personne n'avait la preuve formelle de prouver ce qu'il disait. Ce problème est connu comme le problème si P = NP.

Personne n'a encore été en mesure de prouver si les problèmes NP-complets sont en effet résolubles en temps polynomial, ce qui en fait l'un des plus

grands problèmes de mathématiques. Cependant, il y a une forte suspicion parmi les scientifiques que ces problèmes ne peuvent être résolus en temps polynomial ; selon un vote informel parmi les 100 chercheurs en 2002, pour 61 d'entre eux, il semblait plus probable P ≠ NP alors que seulement 9 que P = NP. La Clay Mathematics Institute à Cambridge, MA offre la récompense d'un million de dollars à quiconque peut prouver que P = NP ou P ≠ NP.

Au début, il semble tout à fait surprenant que les problèmes NP-complets devraient même exister, mais dans le célèbre Théorème de Cook-Levin (Quel que soit démontré par Leonid Levin), Cook a prouvé que le problème de satisfiabilité booléenne Il est NP-complet. En 1972, Richard Karp Il a estimé que d'autres problèmes étaient les mêmes NP-complet, donc il y a une classe de problèmes NP-complets (en plus du problème de satisfiabilité booléenne). A partir du résultat original de Cook, des milliers d'autres problèmes se sont révélés être NP-complet ; beaucoup de ces problèmes sont donnés dans le livre par Garey et Johnson 1979 Ordinateurs et indocilité : Guide de NP-Complétude.

Un problème qui satisfait à la condition 2, mais pas nécessairement la condition 1 est dit NP-dur. Officieusement, un problème NP-dur est « au moins aussi difficile que » tout problème NP-complet, et peut-être encore plus difficile. Par exemple, choisissez le mouvement parfait dans certains jeux de société côté arbitraire est NP-dur ou même strictement plus dur que les problèmes NP-difficile . Comme exemple de problème NP-Complet , nous avons :

1. *Satisfaisabilité d'une formule booléenne*

Nous illustrons la méthode de la réduction en donnant une preuve de la NP-Complétude du problème consistant à déterminer si une formule booléenne, et non plus un circuit, est satisfaisable. Ce problème eut l'honneur d'être le premier dans l'histoire dont on a démontré qu'il était NP-complet. Nous exprimons le problème de la satisfaisabilité (de formule) en fonction du langage SAT de la manière suivante.

Une instance de SAT est une formule booléenne f composée de

- n variables booléens : $x_1, x_2, \ldots, x_n$;
- m connecteurs booléens : toute fonction booléenne avec une ou deux entrées et une sortie, telle que
$\wedge\,(ET), \vee\,(OU), \neg\,(NON), \rightarrow\,(implication), \leftrightarrow\,(si\ et\ seulement\ si)$; et
- des parenthèses. (Sans nuire à la généralité, on suppose qu'il n'y a pas de parenthèses redondantes, c'est à dire qu'il y a au plus une paire de parenthèses par connecteur booléen.)

Il est facile d'écrire une formule booléenne f dans une longueur qui est polynomiale en$n+m$. Comme c'est le cas avec les circuits combinatoires booléens, une assignation de vérité pour une formule booléenne f est un ensemble de valeurs pour les variables de f, et une assignation satisfaisante est une assignation pour laquelle l'évaluation de la formule donne 1. Une formule possédant une assignation satisfaisante est une formule satisfaisable. Le problème de la satisfaisabilité consiste à savoir si une formule booléenne donnée est satisfaisable ; en termes des langages formels, on dira

$SAT = \{\langle\phi\rangle \, tel\, que\, est\, une\, formule\, booléenne\, satisfaisable\}$.

Exemple : La formule

$\phi = ((x_1 \to x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$

possède l'assignation satisfaisante$\langle x_1=0, x_2=0, x_3=1, x_4=1\rangle$, puisque

$\phi = \big((0 \to 0) \vee \neg((\neg 0 \leftrightarrow 1) \vee 1)\big) \wedge \neg 0$

$\phi = (1 \vee \neg(1 \vee 1)) \wedge 1$

$\phi = (1 \vee 0) \wedge 1$

$\phi = 1$

et donc cette formule ϕ appartient à SAT.

L'algorithme naïf permettant de déterminer si une formule booléenne arbitraire est satisfaisable ne s'exécute pas en temps polynomial. Il existe 2^n assignations possibles d'une formule ϕ à n variables. Si la longueur de f est polynomiale en n, alors la vérification de chaque assignation demande un temps $\Omega(2^n)$, ce qui est suprapolynomial par rapport à la longueur de ϕ. Comme nous allons le montrer, un algorithme polynomial a très de chance d'exister .

Pour montrer que La satisfaisabilité d'une formule booléenne est un problème NP-Complet. Nous allons :

a $SAT \in NP$

On montre qu'un certificat consistant en une assignation satisfaisante pour une formule f en entrée peut être validé en temps polynomial.

L'algorithme de vérification remplace tout simplement chaque variable de la formule par sa valeur correspondante, puis évalue l'expression.

Ce travail peut facilement s'effectuer en temps polynomial. Si l'expression a pour évaluation 1, la formule est satisfaisable.

Donc, $SAT \in NP$

2. Problème de *clique*

Une clique dans un graphe non orienté G = (S, A) est un sous-graphe complet de G c'est-à-dire que deux sommets quelconques de la clique sont toujours adjacents. La taille d'une clique est le nombre de sommets qu'elle contient. Le problème de la clique est le problème d'optimisation consistant à trouver une clique de taille maximale dans un graphe. Le problème de décision associé est celui de savoir s'il existe une clique de taille K dans le graphe. De manière formelle la définition d'une clique est:

$CLIQUE = \{ <G, k> ¿\, G \, est \, un \, graphe \, contenant \, une \, Clique \, de \, taille \, k \}$

- En entrée : Graphe $G=(S,A)$et une constante k
- Sortie : exist-il dans G une clique $S' \subseteq S$avec¿ $S' \vee$¿ k ?

Montrer que CLIQUE est NP-Complet, revient à montrer que $CLIQUE \in NP$ puis que CLIQUE est NP-Difficile

a Montrons que clique est dans NP

Pour montrer que $CLIQUE \in NP$, pour un graphe donné $G=(S,A)$, on

utilise l'ensemble $S' \subseteq S$ des sommets de la clique comme certificat pour G. Vérifier que S' est une clique peut être accompli en temps polynomial, en testant si pour toute paire u, v ∈ S' l'arête (u, v) appartient à A.

b Montrons la NP-difficulté de clique

Nous allons montrer que 3-CNF-SAT se réduit polynomialement à CLIQUE c'est-à-dire 3-CNF-SAT $\leq_P CLIQUE$. L'algorithme de réduction commence avec une instance de 3-CNF-SAT. Soit $f = C_1 \wedge C_2 \wedge \cdots \wedge C_k$ une formule booléenne sous forme 3-CNF à k clauses. Pour r = 1, 2, . . . k, chaque clause C_r possède exactement trois littéraux distincts $l_{r1}, l_{r2} et \, l_{r3}$.

Nous allons construire un graphe G tel que f soit satisfaisable si et seulement si G possède une clique de taille k. Le graphe $G=(S,A)$ est construit de la manière suivante. Pour chaque clause $C_r = (l_1 \vee l_2 \vee l_3)$ de f, on place un triplé de sommets v_{r1}, v_{r2} et v_{r3} dans S. On place une arête entre deux sommets v_{ri} et v_{sj} si les deux conditions suivantes sont satisfaites :

- v_{ri} et v_{sj} sont dans des triplets différents, autrement dit si r est diffèrent de s ;
- Leurs littéraux correspondants sont cohérents, autrement dit l_{ri} n'est pas la négation de l_{sj}.

Ce graphe peut se calculer facilement à partir de f en temps polynomial. À titre d'exemple de cette construction, si l'on se donne

$f=(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$, alors G est le graphe représenté à la figure suivante

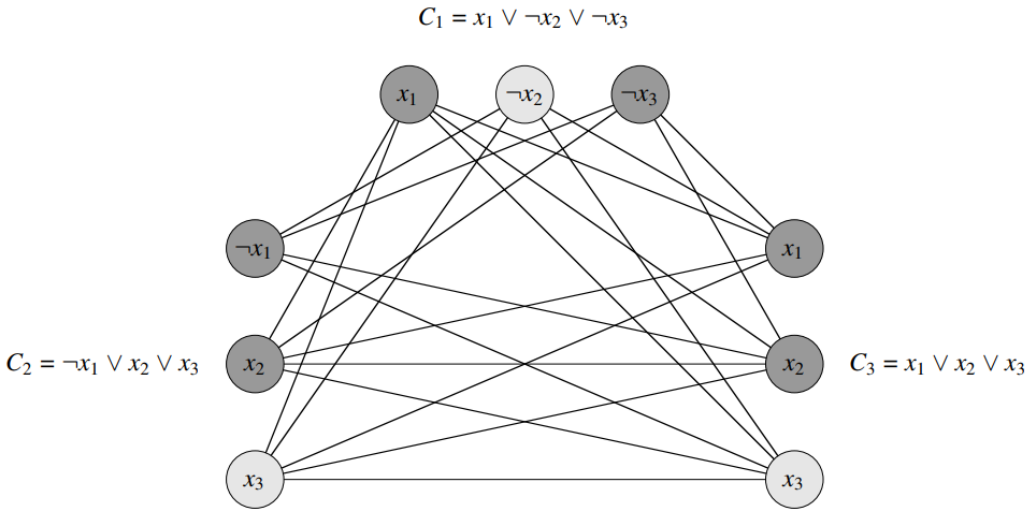


Figure 8: Problème clique(graphe issu de la formule)

Montrons que ce passage de f à G est une réduction.

- D'abord, supposons que f est une assignation satisfaisante. Alors, chaque clause C_r contient au moins un littéral l_{ri} ayant la valeur 1, et chaque littéral de ce type correspond à un sommet v_{ri}. En prenant l'un de ces littéraux ayant la valeur vraie dans chaque clause, on obtient un ensemble S de k sommets. Nous affirmons que S est une clique. Pour deux sommets quelconques v_{ri}, $v_{sj} \in$ S, où r s, l'assignation satisfaisante associe la valeur 1 aux deux littéraux l_{ri} et l_{sj} correspondants, et les littéraux ne peuvent donc pas être compléments.

Donc, d'après la construction de G, l'arête (v_{ri}, v_{sj}) appartient à A. Le graphe précèdent issu de la formule 3-CNF $f=C_1 \wedge C_2 \wedge C_3$, où $C_1=(x_1 \vee \neg x_2 \vee \neg x_3)$, $C_2=(\neg x_1 \vee x_2 \vee x_3)$et $C_3=¿$), lors de la réduction de 3-CNF-SAT à CLIQUE. L'assignation $¿x1=0 ou 1, x2=0, x3=1>¿$ est satisfaisante.

Cette assignation vérifie C_1 avec $\neg x_2$ et vérifie C_2 et C_3 avec x_3, ce qui correspond à la clique dont les sommets sont en clair.

- Supposons maintenant que G contienne une clique de taille k. Comme les littéraux d'une même clause ne sont pas reliés, cette clique contient un littéral exactement dans chaque clause. Montrons alors qu'il existe une assignation qui rend tous ces littéraux vrais. Chaque littéral de cette clique est égal à x_i ou à $\neg x_i$. Pour que ce littéral soit vrai, on impose la valeur 1 ou 0 à la variable correspondante x_i. Comme tous les littéraux de la clique sont reliés par une arête, ils ne sont pas contradictoires deux à deux. Ceci signifie que deux littéraux quelconques de la clique concernent deux variables distinctes x_i et x_j avec $i \neq j$ ou alors ils concernent la même variable x_i mais ils imposent la même valeur à la variable x_i. On obtient alors une assignation cohérente des variables apparaissant dans la clique. En affectant n'importe quelle valeur à chacune des autres variables, on obtient une affectation qui donne la valeur 1 à la formule f. On pourrait penser que l'on a montré que CLIQUE est NP- difficile uniquement pour les graphes où les sommets vont par trois et où il n'y a pas d'arêtes entre deux sommets d'un même triplet.

En fait, on a montré que CLIQUE est NP-difficile uniquement pour ce cas particulier, mais cette preuve suffit pour montrer que CLIQUE est NP-difficile pour des graphes quelconques.

D'où le problème de la clique est NP-complet.

3. Satisfaisabilité d'un circuit

Un circuit est dit satisfiable s'il existe une assignation aux valeurs d'entrées telles que le circuit global produit « OUI » ou « 1 ».

Pour montrer que CUIRCUIT-SAT est NP-complet, nous allons d'abord :

a **Montrons que CUIRCUIT-SAT** ϵ *NP*

Soit C un circuit constitué de n portes logiques et A une assignation qui satisfait le circuit.

Montrons que la vérification de A sur C se fait en temps polynomial.

C est constitué de n portes logiques, la vérification de A sur C implique que nous allons évaluer les résultats de ces n portes logiques en fonction de leurs entrées. D'où la vérification de A sur C se fait dans l'ordre n portes.

$O(n)$ est un temps polynomial. D'où CUIRCUIT-SATϵ NP.

b **Montrons que CUIRCUIT-SAT est NP-difficile (complexité exponentielle)**

Étant donné que CUIRCUIT-SAT est satisfaisable, on regarde simplement la porte qui donne le résultat global du circuit, on exprime de manière récurrente chacune des entrées en tant que formule. La formule correspondante au circuit est alors obtenue en écrivant une expression qui applique la fonction de la porte aux formules de ces entrées.

Malheureusement cette méthode directe n'est pas une résolution en temps polynomial.

Les sous-formules partagées peuvent faire croître exponentiellement la taille de la formule générée pour peu qu'il ait des portes donc les files de sorties est de facteur d'éventail de 2 ou plus (croit de façon exponentielle). D'où la complexité dans le cas est de l'ordre d'exponentielle.

D'où CUIRCUIT-SAT est NP-difficile.

Nous avons montré que CUIRCUIT-SATϵ NP d'après a, et d'après b nous avons montré qu'il est NP-difficile. D'où CUIRCUIT-SAT est NP-complet.

4. Le problème du k-center

Dans cette partie, nous allons nous concentrer sur le problème du k-centre : étant donné un ensemble de villes dont les distances sont spécifiées, choisir k villes afin d'installer des entrepôts de façon à minimiser la distance maximale d'une ville à l'entrepôt le plus proche. Un tel ensemble de k villes est appelé k-centre.

Le problème associé de décision est le suivant k-centre

Données: un graphe complet K = (V, E) muni d'une fonction de poids w sur les arêtes , et des entiers strictement positifs k et b.

Question : Existe-t-il un ensemble S de sommets tel que |S| = k et tel que tout sommet v de V satisfait la condition suivante

min{w(v, u) : u ∈ S} ≤ b

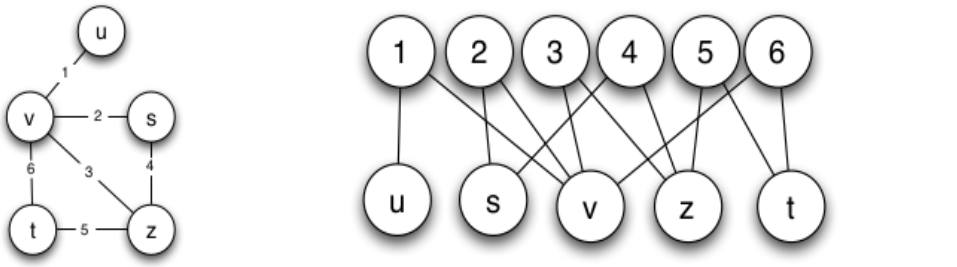
a) Le problème ensemble dominant est NP-complet

Rappelons les définitions suivantes :

- une couverture de sommets *S* du graphe *G* est un sous-ensemble de sommets tel que toutes les arêtes de *G* sont incidentes à au moins un sommet de *S*.
- un ensemble dominant *C* du graphe *G* est un sous-ensemble de sommets tel que tout sommet est soit dans *C* soit voisin d'un sommet de *C*.

Soit *G* un graphe $G = (V, E)$. Nous allons construire un graphe $G^0 = (V^0, E^0)$ à partir de *G* tel que

- $V^0 = V \cup E$;
- $E^0 = E \cup \{(v, a) | v \in V, a \in E, v$ est extrémité de l'arête a dans $G\}$



Graphe G Graphe G' (les arêtes dans E ne sont pas dessinées)

Question : Montrer que si *S* est une couverture de sommets du graphe *G*, alors *S* est un ensemble dominant de G^0.

Solution

Soit S une couverture de sommets du graphe G. Il faut prouver que tous les sommets du graphe soit voisin de S ou soit dans S.

Toutes les arêtes de G ont au moins un de ses extrémités dans S. Par conséquence, tous les sommets de G sont soit dans S ou soit voisins de S.

Comme toutes les arêtes de G ont au moins un de ses extrémités dans S, tous les sommets correspondant à une arête de G sont des voisins de S.

Donc S est un ensemble dominant de G^0.

Question : Montrer que si S^0 est un ensemble dominant de G^0, alors il existe un ensemble S de meme cardinalité de S^0 qui est une couverture de graphe G,

Solution

Si $S^0 \subseteq V$, alors tous les sommets du graphe G sont soit voisin de S ou soit dans S.

Donc toutes les arêtes ont une de ces extrémités dans S.

Donc S^0 est une couverture de graphe G,Sinon, il existe un sommet a dans S^0 qui n'appartient pas à V mais qui appartient à V^0 .

Cela signifie que le sommet a représente une arête (u, v) dans G et que couvre uniquement

les arêtes (a, u) et (a, v) dans G^0 . Considérons un sous-ensemble S^{00} tel que

S^{00} = (S 0 ∩ V) ∪ {u : a = (u, v) ∧ a ∈ S 0 }

Remarquons que | S^{00} | = | S^0 |. Soit a ∈ S^0 \V . Comme le sommet a arêtes (a, u) et (a, v) dans G^0 , alors le sommet u couvre les mêmes arêtes. Donc S^{00} est une couverture de graphe G.

Question : Montrer que ce problème est dans NP.

Solution

On peut vérifier en temps polynomial si un ensemble de sommets est un ensemble dominant et s'il est de cardinalité inférieure à k.

Question : Montrer que ce problème est dans NP. NP-complet.

Solution

Soit (G, k) une instance du problème Couverture sommet. On peut construire en temps polynomial le graphe G^0 .

Les questions précédentes ont permis de prouver que le fait suivant : Le graphe G admet est une couverture de sommets S de taille k si et seulement si le graphe G 0 admet est un ensemble dominant de taille k. Donc le problème est dans NP-complet.

b) Le problème k-centre est NP-complet

Question : Montrer que le problème k-centre est dans NP.

Solution

On peut vérifier en temps polynomial si S est de cardinalité

Question : Montrer que k-centre est NP-complet sachant que DOMINANT est NP-complet.

Conclusion

Parvenu au terme de notre étude où il était question de faire une étude sur la NP-Complétude, force est de constater bien que bien qu'il existe les classe P (polynomiale), NP (Non polynomiale) il existe aussi un type de classe dite NP-COMPLET classe dans laquelle les classe P ne sont pas incluses. Le problème de cette classe prend un temps polynomial pour tester une solution mais un temps exponentiel pour écrire son algorithme.

Les mathématiciens ont démontré qu'il est possible de transformer n'importe quel problème NP en un problème de 3 coloriages et c'est parce qu'il peut représenter ainsi n'importe quel problème NP qu'il est dit NP-COMPLET. Et c'est aussi le cas du problème **SAT**, du problème du **Sac à do** et de beaucoup d'autre problème.

Webographie

- http://www.uqac.ca/rebaine/8INF806/NPcompletudecours.pdf
- https://fr.wikipedia.org/wiki/Probl%C3%A8me_de_d%C3%A9cision
- http://www.enseignement.polytechnique.fr/informatique/INF550/Cours1112/INF550-2011-4.pdf

B. Fiche de TD

Exercice 1 :

Montrer que la classe P, vue en tant qu'ensemble de langages, est fermée pour l'union, l'intersection, la concaténation, le complément et l'opération Kleene étoile. Autrement dit, si L1, L2 $\in$ P, alors L1 $\cup$ L2 $\in$ P,

Exercice 2 : Réduction polynomiale

.

i Propriétés sur l'ordre $\leq$

Soient A et B deux problèmes de décision.

Une réduction de A vers B est une fonction f : IA $\rightarrow$ IB calculable en temps polynomial telle que w $\in$ Oui(A) si et seulement si f(w) $\in$ Oui(B). Nous notons A $\leq$ B lorsque A se réduit à B.

a) Montrer que $\leq$ est réflexive (L $\leq$ L).

b) Montrer que $\leq$ est transitive (c'est-à-dire L1 $\leq$ L2, L2 $\leq$ L3 impliquent L1 $\leq$ L3.)

c) Montrer que P = NP si et seulement si 3-SAT $\in$ P.

ii Réduction

Soient A, B et Q des problèmes de décision. Supposons que A est dans P, et que B est NP-dur. Dire si les affirmations suivantes sont vraies ou fausses :

1. Si A se réduit polynômialement à Q, alors Q est dans P.
2. Si Q se réduit polynômialement à A, alors Q est dans P.
3. Si Q se réduit polynômialement à B, alors Q est NP −dur.
4. Si B se réduit polynômialement à Q, alors Q est NP –dur

Exercice 3 : Graphe eulérien

Le graphe G est eulérien si il existe un cycle en empruntant exactement une fois chaque arête du graphe G. On rappelle qu'un graphe connexe est eulérien si et seulement si chacun de ses sommets a un degré pair.

Question 1.1. Ecrire le problème de décision qui lui est associé et donner la taille de l'instance

Question 1.2. Trouver un algorithme polynomial qui détermine si le graphe est eulérien.

Exercice 4: Formulation des problèmes de décisions

Mettre sous forme de problème de décision et évaluer la taille de leurs instances.

Question 2.1. Problème de savoir s'il existe un chemin entre deux sommets disjoints dans un graphe ;

Question 2.2. Problème de connaitre la distance entre deux sommets disjoints dans un graphe ;

Question 2.3. Problème de connaitre la longueur de la chaîne maximum dans un graphe pondéré.

Exercice 5 Problèmes dans NP ou dans P

Les problèmes suivants sont-ils dans NP, dans P ? Justifier votre réponse.

Problème P1

onnées : Un graphe G = (V, E)
Question : Existe-t-il cycle de longueur égale à b |V 2 | c ?

Problème P2

Données : Un graphe G = (V, E)
Question : Existe-t-il cycle de longueur égale à 4 ?

Problème P3

Données : Un graphe G = (V, E), deux sommets u et v distincts de G et un entier k.
Question : Existe-t-il un simple chemin entre u et v de longueur inférieure ou égale à k ?

Problème P4

Données : Un graphe G = (V, E), et un entier k.
Question : Existe-t-il un arbre couvrant tous les sommets de G ayant moins de k feuilles ?

Exercice 3 : Le problème 2-SAT est dans P

Nous allons considérer de cet exercice des fonctions booléennes, c'est-à-dire de fonctions de f de $\{1,0\}n \to \{1,0\}$.

- Les variables ne peuvent prendre que deux valeurs, vrai (codé par 1) ou faux (codé par 0).
- La fonction booléenne f est composée de variables et d'opérateurs comme négation (¬) la conjonction ($\wedge$) la disjonction ($\vee$), l'implication $\to$.

Voici la table de vérité pour les opérateurs booléens cités ci-dessus :

u	v	$\neg u$	$(u \wedge v)$	$u \vee v$	$u \to v$
0	0	1	0	0	1
0	1	1	0	1	1
1	0	0	0	1	0
1	1	0	1	1	1

Considérons une fonction t : $U \to \{0,1\}$. On dira que la fonction t satisfait la fonction f si la fonction f retourne 1 avec les valeurs de t en entrée.

1 Nous allons considérer la fonction φ_1 : telle que $\{1,0\}^3 \to \{1,0\}$.
$\varphi_1(x,y,z)=(y \vee \neg z) \wedge (\neg y \vee z) \wedge (y \vee \neg x)$.

Donner une fonction t : U → {0,1} telle que t satisfait φ_1.

2 Que se passe-t-il pour φ_2

$\varphi_2(x,y,z)=(y \vee z) \wedge (\neg y \vee z) \wedge (\neg z \vee x) \wedge (\neg z \vee \neg x)$?
Une clause est une fonction de f de $\{1,0\}^n \to \{1,0\}$ composée de variables et d'opérateurs comme négation (¬) et la disjonction ($\vee$).
Par exemple $C(u_2,u_3)=(u_2 \vee \neg u_3)$ est une clause.
Donnons la définition du problème 2-SAT
Données : un ensemble U de variables $\{u_1,u_2,\ldots,u_n\}$ et une formule logique $\varphi=C_1 \wedge \cdots \wedge C_l$ des clauses de 2 littéraux.
Existe-t-il une fonction t : $U \to \{0,1\}$ telle que t satisfait φ ?.

3 Montrer que $u \vee v=(\neg u \Rightarrow v) \wedge (\neg v \Rightarrow u)$

A partir d'une instance (U,φ) de 2-SAT, nous construisons un graphe orienté $G_\varphi=(V,E)$ tel que

- un sommet par un littéral de U.
- un arc par implication (en transformant chaque clause par deux implications)

4 Dessiner les graphes $G_{\varphi 1}$ et $G_{\varphi 2}$

5 Montrer que si G_φ possède un chemin entre *u* et *v*, alors il poss`ede aussi un chemin entre $\neg v$ *et* $\neg u$.

6 Montrer qu'il existe une variable u de U tel que $G_\varphi$contient un cycle entre *u* vers $\neg u$ si et seulement si φ n'est pas satisfiable.

Exercice 4 : Différentes Variantes du problème cycle hamiltonien

Nous allons supposer que le problème Cycle Hamiltonien est NP-Complet.

Cycle Hamiltonien

Données : un graphe non-orienté G.

Question : G contient-il un cycle hamiltonien?

a Considérons le problème Chaine Hamiltonien suivant :

Chaine Hamiltonienne

Données : un graphe non-orienté G, deux sommets u et v distincts de G.

Question : G contient-il une chaine hamiltonienne entre u et v?

Montrer que le problème Chaine Hamiltonien est NP-complet

b Le problème Chaine est le problème de décision suivant

Chaine

Données : un graphe non-orienté G de n sommets, deux sommets u et v distincts de G.

Question : G contient-il une chaine de longueur n/2 entre u et v?

Montrer que le problème Chaine est NP-complet

c Chevaliers de la table ronde

Etant données n chevaliers, et connaissant toutes les paires de féroces ennemis parmi eux, est-il possible de les placer autour d'une table circulaire de telle sorte qu'aucune paire de féroces ennemis ne soit côte à côte?

d Considérez le problème Voyageur de Commerce :

Données : Un graphe complet G muni d'une fonction d de coût positif sur les arrêtes, et un entier k.

Question : Existe-t-il un circuit passant au moins une fois par chaque ville et dont la somme des coûts des arrêtes est au plus k?

- Montrer que Voyageur de Commerce est NP-complet même si la fonction poids respecte l'inégalité triangulaire. (C'est-`a-dire que si les arrêtes vérifient l'inégalité triangulaire, c'est-à-dire si pour tout triplet de sommets $(u,v,w), on\ a\ d(u,v)\leq d(u,w)+d(w,v).$ ¿
- Montrer que le problème du voyageur de commerce ayant une métrique respectant l'inégalité triangulaire est dans NP-complet.

Exercice 5 :

Montrer que le problème du chemin hamiltonien peut être résolu en temps polynomial sur les graphes orientés sans circuits. Donner un algorithme efficace pour le problème.

Exercice 6 : Problème de clique maximum

Considérons le problème de décision ***CLIQUE***:

Données : un graphe non-orienté $G=(V,E)$ et un entier ***k***.

Question : Existe-t-il une clique de taille *k*?

1 Nous noterons par G^c le complémentaire du graphe **G**.
Montrer que **G** a une clique de taille k si et seulement si G^c a une couverture de sommets de taille $n-k$.

2 Montrer que le problème **CLIQUE** est NP-complet.

Nous allons travailler sur une restriction du problème **CLIQUE** en considérant uniquement les graphes dans lesquels tous leurs sommets sont de degré au plus 3. Nous le noterons ***3-CLIQUE***.

3 Montrer que ***3-CLIQUE*** est dans **NP**

4 Trouver l'erreur dans le raisonnement suivant :

Nous savons que le problème **CLIQUE** est NP-complet, il suffit donc de présenter une réduction de ***3-CLIQUE*** à **CLIQUE.** Etant donné un graphe G' dont les sommets sont de degré inférieur à 3, et un entier **k**, la réduction laisse inchangée le graphe et le paramètre **k** : clairement le résultat de la réduction est une entrée possible pour le problème CLIQUE. Par ailleurs, la réponse aux deux problèmes est identique. Cela prouve la justesse de la réduction et, par conséquent, la NP-complétude de la **3-CLIQUE**.

5 Donner un algorithme polynomial pour le problème ***3-CLIQUE***