Les classes de problèmes

Classe de complexité P:

La classe de complexité P est l'ensemble des problèmes concrets de décision qui sont résolubles en temps polynomial.

Classe de complexité P

Exemple :

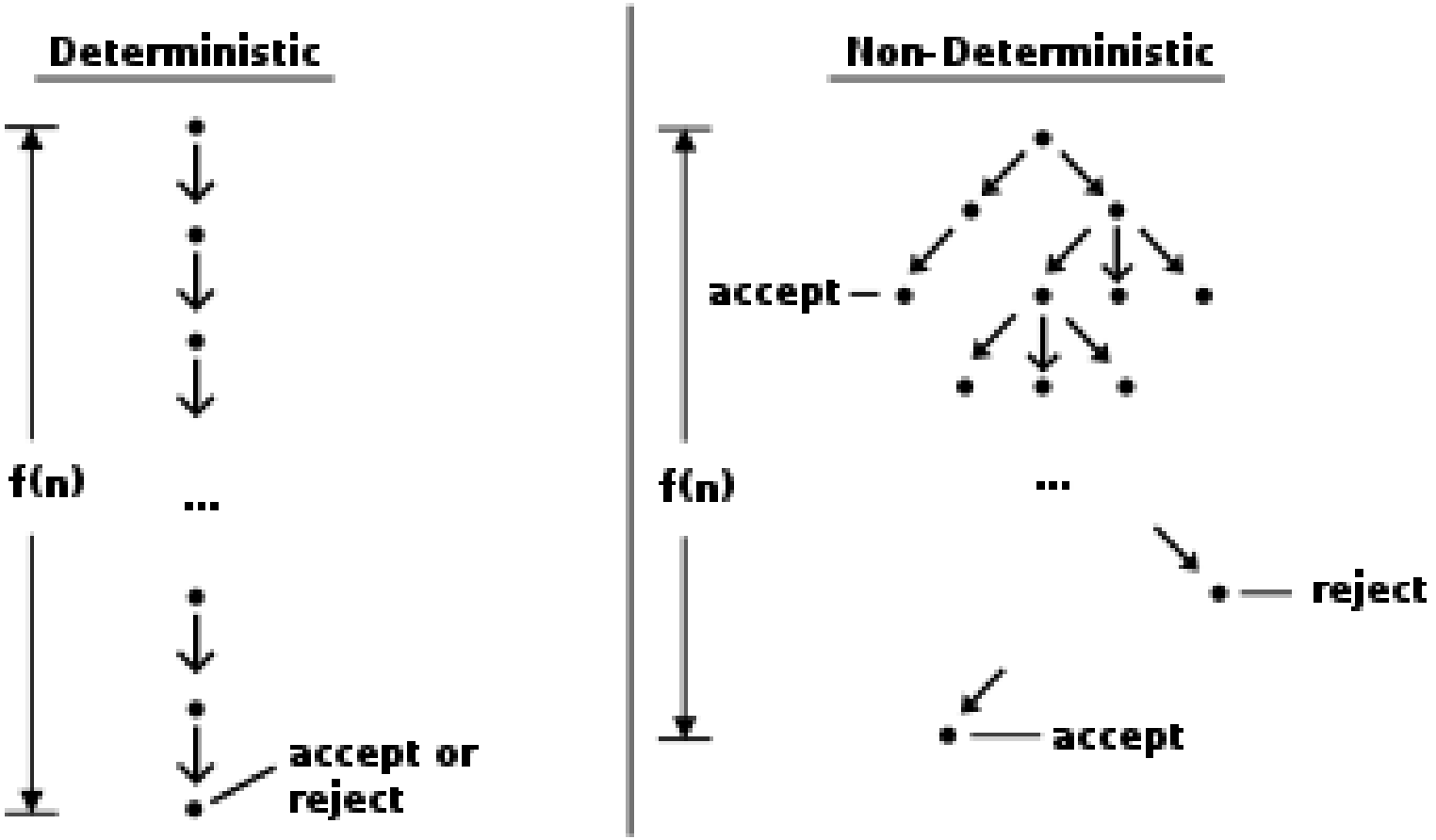
prenons le problème CHEMIN qui répond à la question « étant donné un graphe G, deux sommets u et v et un entier positif k, existe-t-il dans G un chemin de u à v de longueur au plus k ? »

Classe de complexité P

De nombreux problèmes abstraits ne sont pas des problèmes de décisions mais des problèmes d'optimisation. Pour leur appliquer la théorie de la NP-complétude, le plus souvent on les reformulera sous la forme d'un problème d'optimisation en imposant une borne sur la valeur à optimiser, comme nous l'avons fait en passant du problème PLUS-COURT-CHEMIN au problème CHEMIN.

La classe de complexité NP(nondeterministic polynomial time) *:est l'ensemble des problèmes concrets de décision Q pour lesquels il existe un algorithme polynomial de validation A.*

NP


Deterministic
f(n)
...
accept or
reject
Non-Deterministic
accept
f(n)
...
reject
accept

NP

3	2		1				3	1
3	2	1	2		3			1
	2	3		1		2	2	1
			1	2				2
		3		3		3		2
1	1	2	3	2	3		3	2
		1	3	1		2	2	
1	1		2		2	1	2	
2		2	2	3			1	3

NP-complétude

NP-complétude

Une des raisons qui laissent à penser que P != NP est l'existence de la classe des problèmes NP-complets : si un seul problème NP-complet peut être résolu en temps polynomial, alors tous les problèmes de NP peuvent être résolus en temps polynomial et P = NP. Mais aucun algorithme polynomial n'a jamais été découvert pour aucun problème

NP-complet. Les problèmes NP-complets sont, dans un certain sens, les problèmes les plus « difficiles » de NP.

NP-complets

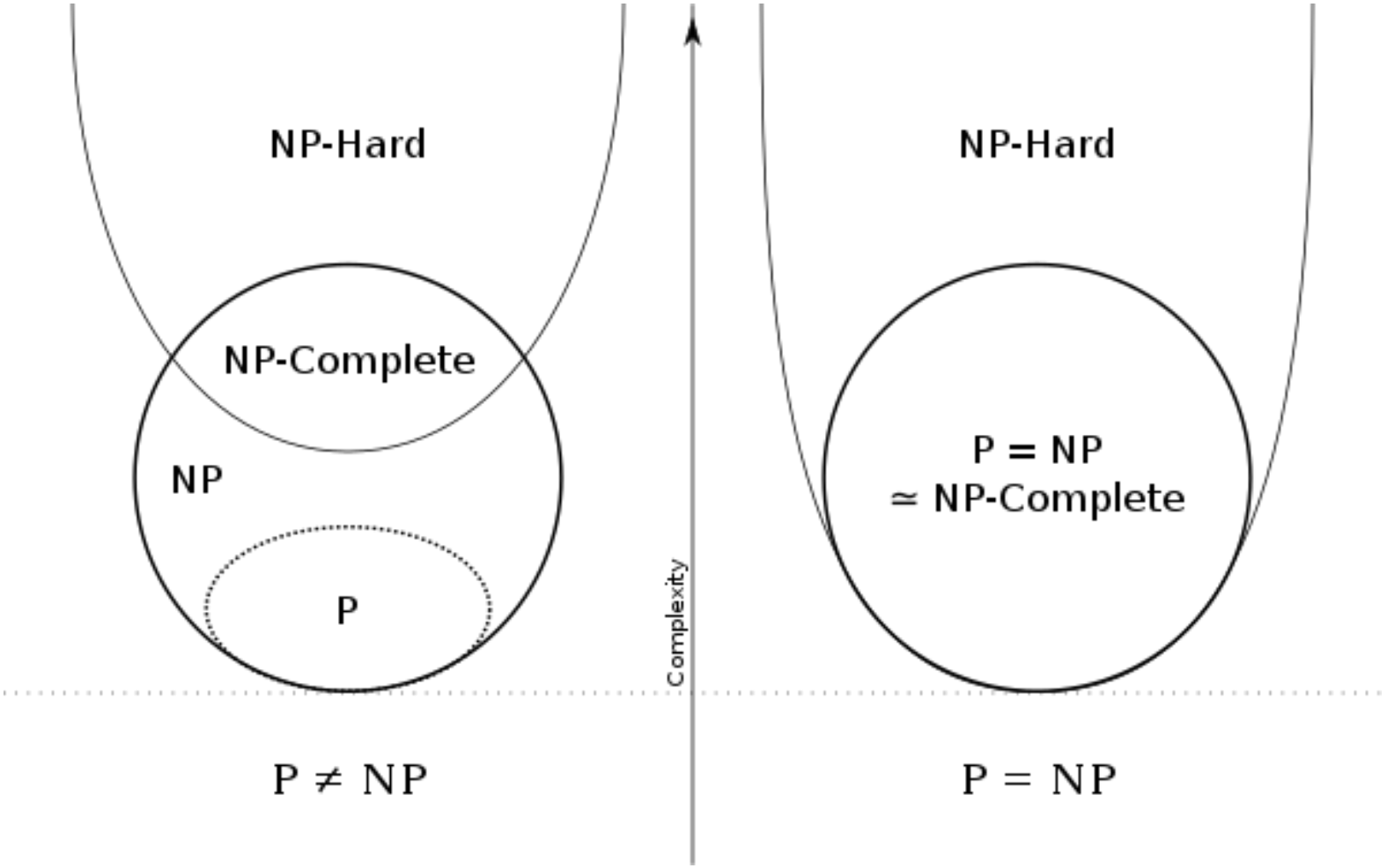
Un problème Q est ***NP-complet*** *si*

1. $Q \in NP$.
2. $\forall Q' \in NP,\ Q' \leq_P Q$.

On note NPC la classe des problèmes NP-complets.

Un problème concret qui vérifie la propriété 2 mais pas nécessairement la propriété 1 est dit ***NP-difficile(NP-HARD)***

NP-complets


NP-Hard
NP-Complete
NP
P
Complexity
NP-Hard
P = NP
≃ NP-Complete
P ≠ NP
P = NP

P = NP?

Si un problème de NP est résoluble en temps polynomial, alors P = NP. De façon équivalente, si un problème quelconque de NP n'est pas résoluble en temps polynomial, alors aucun problème NP-complet ne peut se résoudre en temps polynomial

$P = NP?$

Le Problème du Millénaire (1 million de dollars)

P : Problèmes solubles rapidement (temps polynomial)

NP : Problèmes dont les solutions sont **vérifiables** rapidement

Question : P = NP ? Est-ce que "vérifier" = "trouver" ?

Si P = NP : Cela révolutionnerait le monde :

- Cryptographie actuelle cassée
- Intelligence Artificielle parfaite
- Optimisation extrême de tout
- Découvertes scientifiques accélérées

Si P ≠ NP :

Confirmation que certains problèmes sont **intrinsèquement difficiles**

comprendre les limites de l'IA(Planification, raisonnement = souvent NP-complet)

Méthode pour montrer la NP-complétude d'un problème $Q1$

1. Prouver que $Q_1 \in$ NP.
2. Choisir un problème NP-complet Q_2.
3. Décrire un algorithme polynomial capable de calculer une fonction f faisant correspondre toute instance de Q_2 à une instance de Q_1.
4. Démontrer que la fonction f satisfait la propriété :

$$Q_2(x) = vrai \quad \text{si et seulement si} \quad Q_1(f(x)) = vrai.$$

5. Démontrer que l'algorithme calculant f s'exécute en temps polynomial.

NP-complets(SAT)

.SAT : soit une formule booléenne composée de variables x1, ..., xn et de connecteurs (et, ou, non, implication, équivalence) et de parenthèses ; existe-t-il une affectation des x1, ..., xn pour laquelle la formule soit vraie ?

.Premier problème dont la NP-complétude ait été démontrée, par Cook en 1971.

3-SAT
3-SAT est un cas particulier de SAT où la formule doit être sous forme normale conjonctive (FNC) et chaque clause a exactement 3 littéraux.

FNC
Littéral : une variable (x_1) ou sa négation ($\neg x_1$)

Clause : une disjonction (OU) de littéraux, par exemple ($x_1 \vee \neg x_2 \vee x_3$)

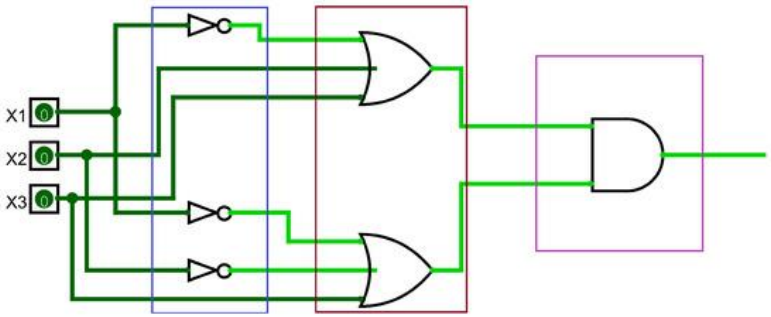
FNC : conjonction (ET) de clauses, par exemple $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee x_4)$

Importance: Centrale en théorie de la calculabilité, utilisée dans la vérification matérielle/logicielle, la planification, les puzzles, etc.

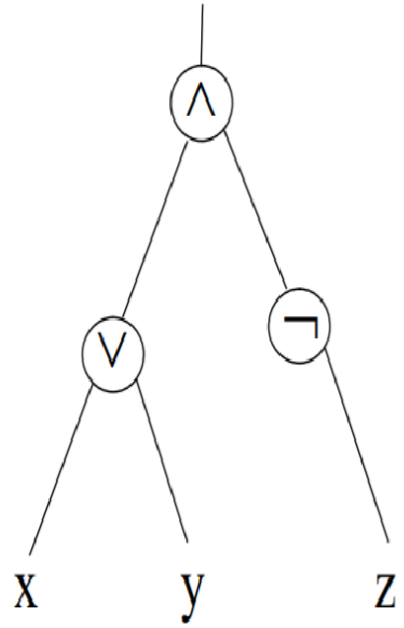
NP-complets

4)

3 SAT: Centrale en théorie de la calculabilité, utilisée dans la vérification matérielle/logicielle, la planification, les puzzles, etc.



$f(x_1, x_2, x_3) = (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$



, WHERE ∨=OR, ∧=AND, ¬=NOT.

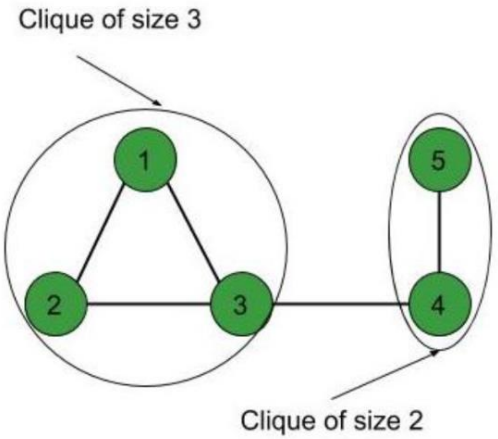
NP-complets(**CLIQUE)**

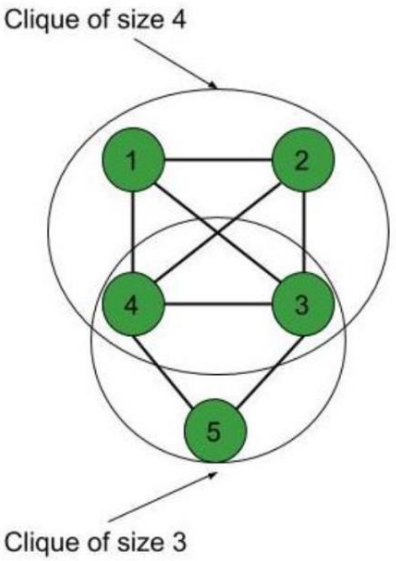
.CLIQUE

.Une **clique** est un sous-ensemble de sommets dans un graphe où **chaque paire de sommets est connectée par une arête** (sous-graphe complet).

Problème de Décision :

"Étant donné un graphe G et un entier k, est-ce que G contient une clique de taille ≥ k ?"



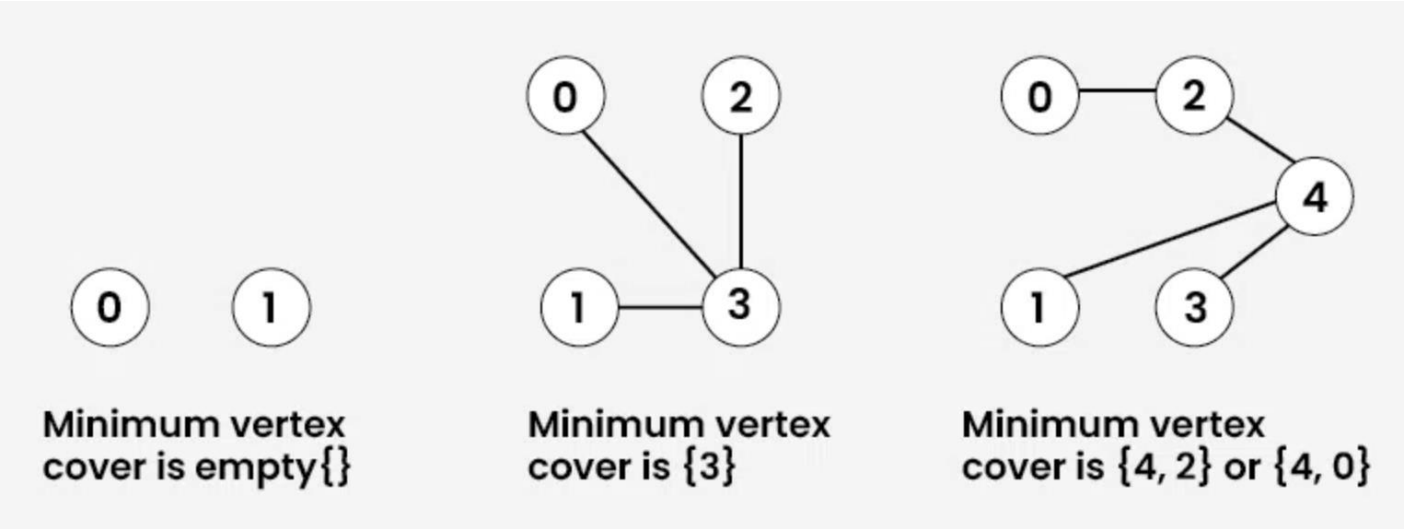


COUVERTURE DE SOMMETS (Vertex Cover)

Une **couverture de sommets** est un ensemble de sommets qui **touche chaque arête** du graphe (au moins une extrémité de chaque arête est dans l'ensemble).

Problème de Décision :

"Étant donné un graphe G et un entier k, est-ce que G a une couverture de sommets de taille ≤ k ?"



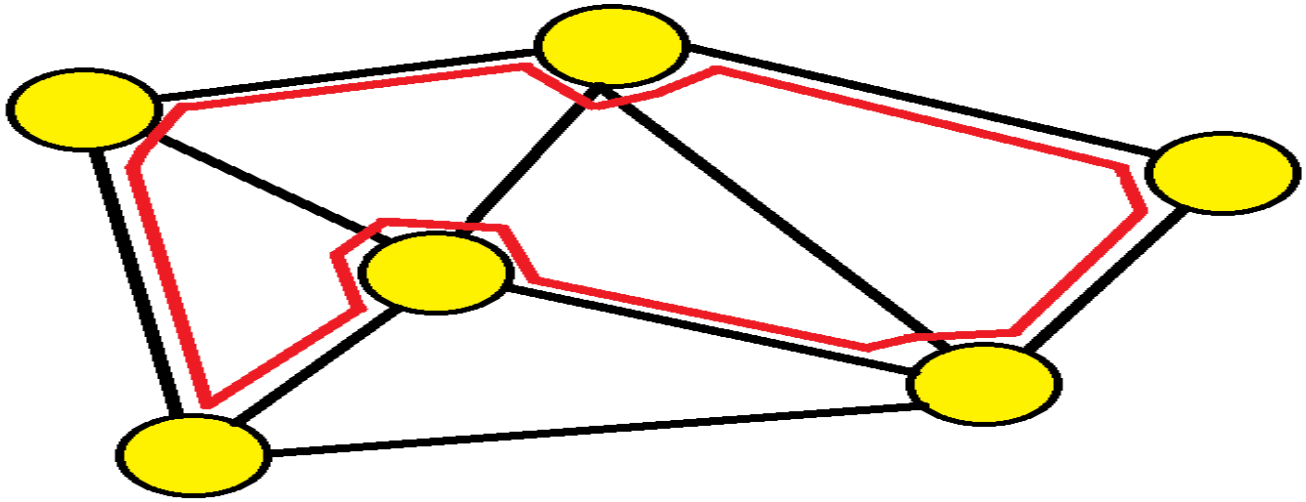
NP-complets(**HAM Cycle)**

.CYCLE HAMILTONIEN

.Un cycle hamiltonien est un cycle qui visite chaque sommet exactement une fois et retourne au point de départ.

.Problème de Décision :

."Étant donné un graphe G, contient-il un cycle hamiltonien ?"



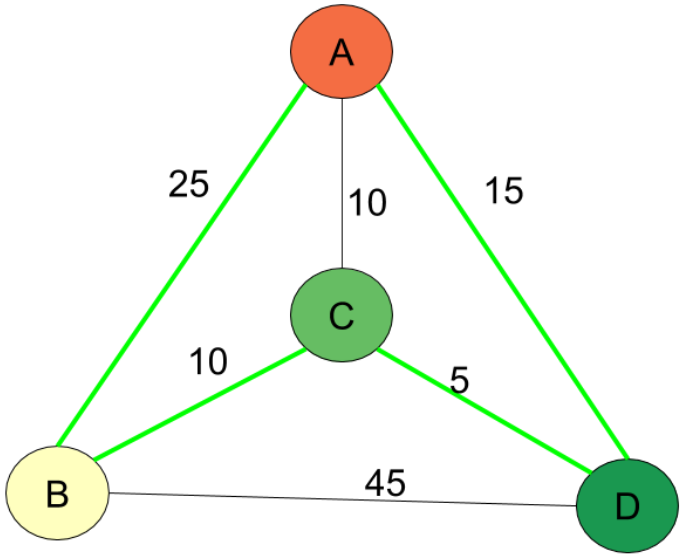
4. PROBLÈME DU VOYAGEUR DE COMMERCE (TSP)

Qu'est-ce que c'est ?

Étant donné des villes et les distances entre elles, trouver le **parcours le plus court possible** qui visite chaque ville exactement une fois et revient à l'origine.

Version Décision :

"Étant donné des villes, des distances, et une borne k, existe-t-il un tour de longueur ≤ k ?"



NP-complets(**Graph coloring**)

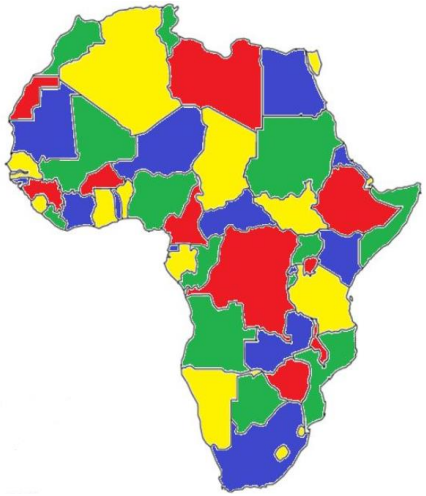
Le **coloriage d'un graphe** consiste à attribuer une couleur à chaque sommet de sorte que **deux sommets adjacents** (reliés par une arête) ne partagent pas la même couleur.

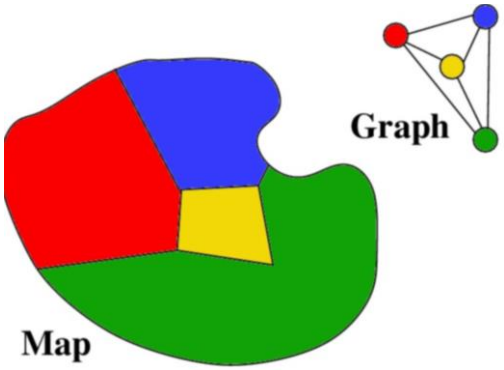
Problème de Décision :

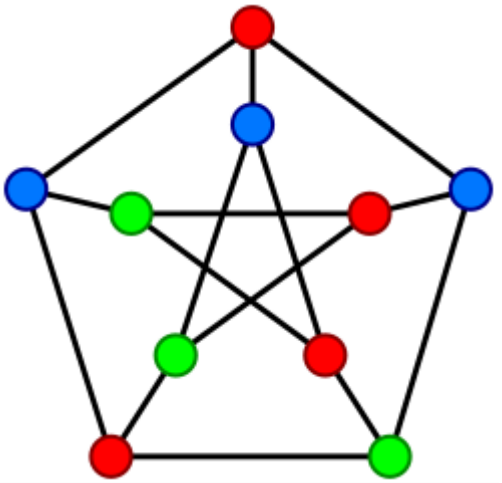
"Étant donné un graphe G et un entier k, peut-on colorier G avec k couleurs ?"

Notation : **k-COLORIAGE**

Objectif : Utiliser le **nombre minimum** de couleurs → **nombre chromatique** χ(G)



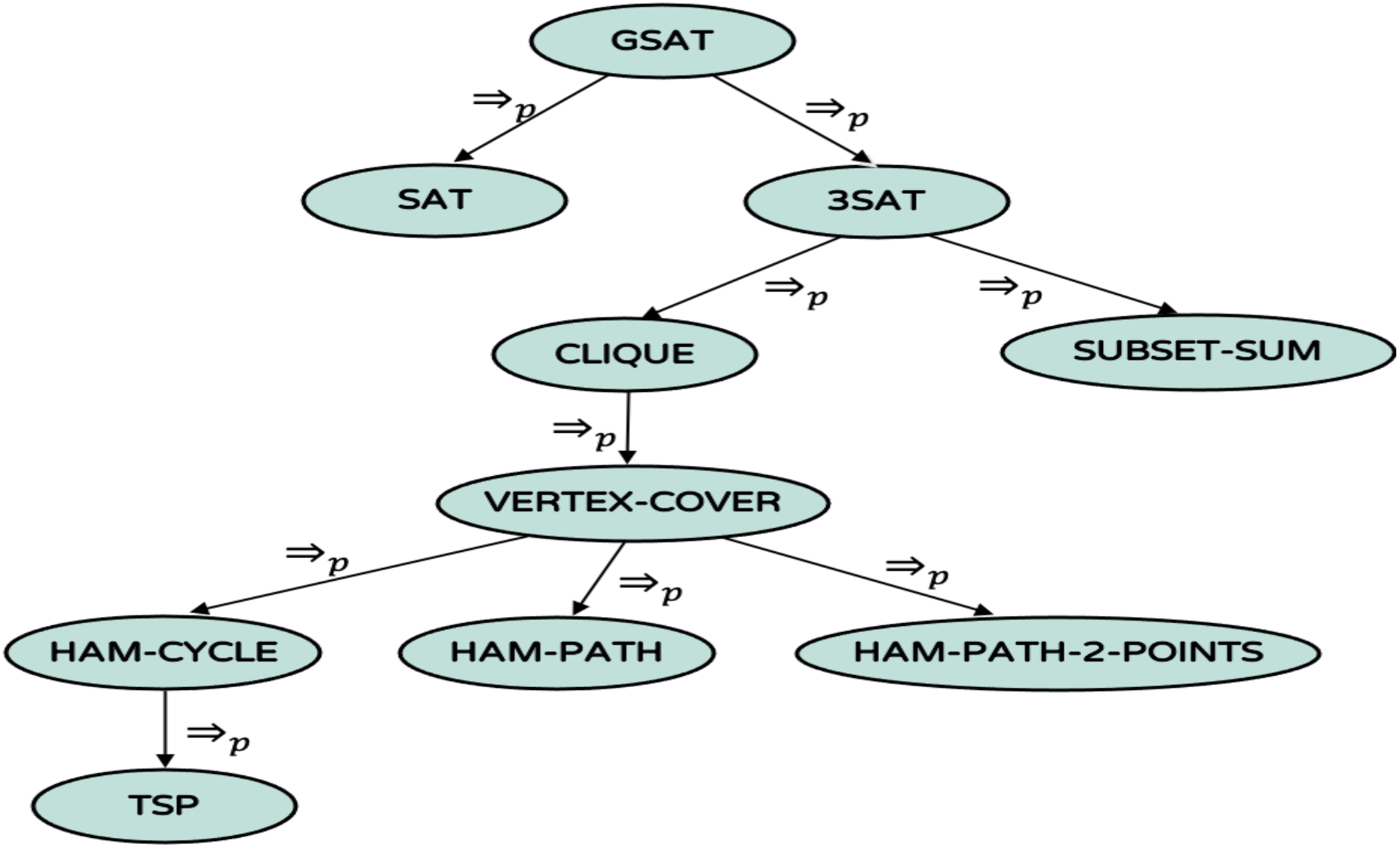




NP-complets

Problème	Domaine Industriel	Exemple Concret	
3-SAT	Aérospatial, Électronique	Vérification de circuits Airbus	
CLIQUE	Réseaux sociaux, Biotech	Détection de communautés Facebook	
COUVERTURE	Sécurité, Santé	Placement de caméras de sécurité	
CYCLE HAMILTONIEN	Logistique, Génétique	Assemblage de génomes	
TSP	Transport, E-commerce	Tournées UPS/FedEx	
3-COLORIAGE	Télécoms, Éducation	Attribution fréquences 5G	

NP-complets


GSAT
$\Rightarrow_p$
$\Rightarrow_p$
SAT
3SAT
$\Rightarrow_p$
$\Rightarrow_p$
CLIQUE
SUBSET-SUM
$\Rightarrow_p$
VERTEX-COVER
$\Rightarrow_p$
$\Rightarrow_p$
$\Rightarrow_p$
HAM-CYCLE
HAM-PATH
HAM-PATH-2-POINTS
$\Rightarrow_p$
TSP

Réductibilité

Réductibilité:

Nous avons besoin de pouvoir comparer la difficulté de problèmes. Intuitivement, un problème $Q1$ peut être ramené à un problème $Q2$ si une instance quelconque x de $Q1$ peut être « facilement reformulée » comme une certaine instance y de $Q2$. Dans ce cas, la résolution du problème $Q2(y)$ nous fournira la solution du problème $Q1(x)$ et le problème $Q1$ n'est, dans un certain sens, « pas plus difficile à résoudre » que le problème $Q2$.

Problème réductible à un autre en temps polynomial: *Soient $Q1$ et $Q2$ deux problèmes concrets. $Q1$ est **réductible en temps polynomial** à $Q2$ (ce que l'on note $Q1 \leq_P Q2$) s'il existe une fonction calculable en temps polynomial f / $Q1(x) = vrai$ si et seulement si $Q2(f(x)) = vrai$:*

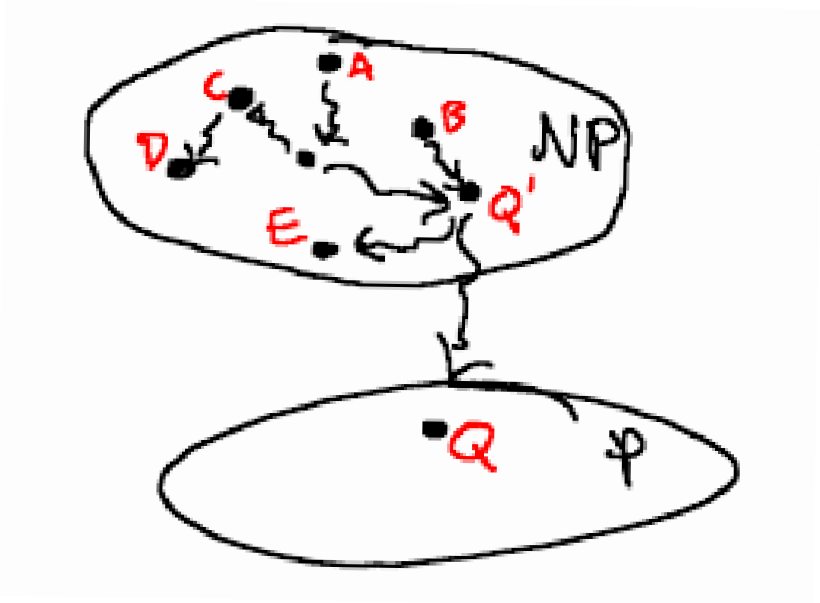
Réductibilité

Les réductions permettent de **classer la difficulté** des problèmes :

Problème A (déjà prouvé NP-difficile)

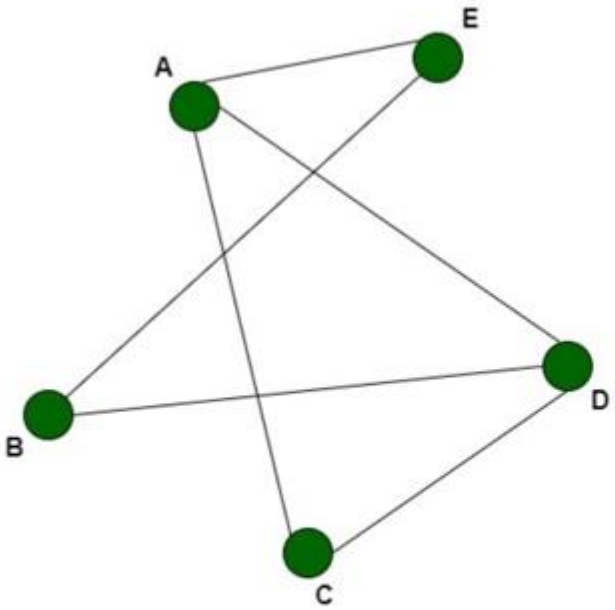
↓ réduction polynomiale

Problème B (nouveau problème)

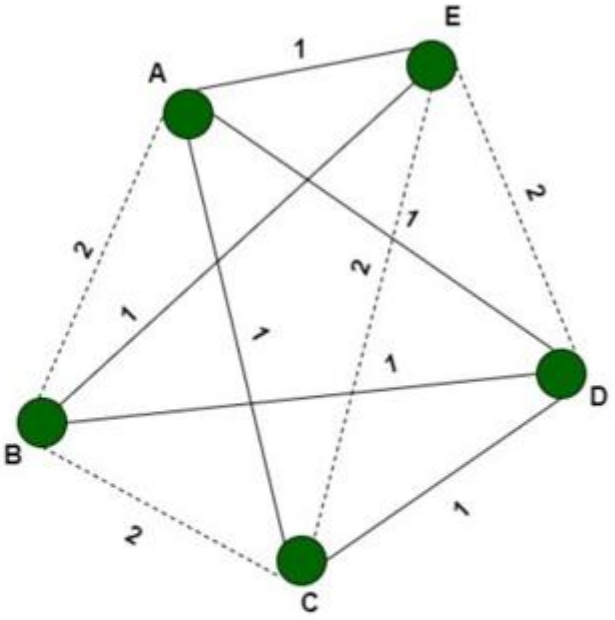


Si : A → B et A est NP-difficile
Alors : B est aussi NP-difficile

Réductibilité



G =
Hamiltonian cycle {EACDBE}



G' =
TSP {EACDBE}
Cost = 5 (=n)

Réductibilité

1. **Problème *Q*1** : problème de l'existence d'un cycle Hamiltonien (cycle simple qui comprend tous les sommets) dans un graphe donné *G*1.

2. **Problème *Q*2** : étant donné un graphe *G*2 de villes valué des distances inter-villes, existe-t-il un cycle (pas forcément simple) passant par toutes les villes, et de longueur inférieure à une valeur fixée *M* ?

3. **Réduction de *Q*1 à *Q*2** :

- On crée un graphe *G*2 de villes contenant autant de villes que *G*1 de sommets. On associe chaque sommet de *G*1 à une ville de *G*2.
- Si deux sommets de *G*1 sont reliés par une arête, on relie les deux villes correspondantes de *G*2 par une arête valuée de la distance interville « 1 », et sinon valuée par la distance interville « 2 ».

4. On exécute l'algorithme résolvant *Q*2 sur *G*2 avec $M = n$, le nombre de sommets de *G*1. S'il existe un tel cycle il est hamiltonien ! Et si *G*1 admet un cycle hamiltonien, *G*2 admet un cycle tel que recherché.

5. *G*2 contient autant de villes que *G*1 de sommets, et le nombre d'arêtes de *G*2 est égal au nombre de paires de sommets de *G*1 : $n(n-1)/2$. La réduction est linéaire en la taille de *G*2 et est donc bien polynomiale en la taille de *G*1.

NP-complet

Savoir qu'un problème est NP-complet ne signifie pas 'abandonner', mais 'changer de stratégie