### **Les classes de complexité :**

**P** : Les problèmes dits polynomiaux, les problèmes résolus à l’optimum avec des algorithmes de complexité polynomiale. (en O(nk)).

**P-Complet** : Classe des problèmes dans P qui sont les plus difficiles à résoudre, vers lesquels tous les problèmes de P peuvent être réduits. Π ϵ P et pour tout Π’ ϵ P, Π’ <logΠ.

**NP** : Classe des langages pouvant êtres validés par un algorithme polynomial. Un langage L appartient à NP ssi il existe un algorithme polynomial A à deux entrées et une constante c telle que : L = {x ϵ {0,1}\* : Il existe y avec |y| = O(|x|c)|A(x,y) = 1}. Un problème est dans NP si toute solution de Π est vérifiable en temps polynomial.

**NP-Complet** : Problèmes de NP les plus difficiles à résoudre, tous les problèmes de NP peuvent être réduits à ces problèmes.

**NPO** : Classe contenant les problèmes d'optimisation dont la formulation en problème de décision se trouve dans NP.

**CoNP** : Notons ~Π (~ barre) le problème ayant comme oui-instance l’ensemble : IΠ/OΠ. Tous ces problèmes forment la classe coNP, {~Π : Π ϵ NP}.

**CoNP-Complet** : Problèmes de CoNP les plus difficiles à résoudre, tous les problèmes de CoNP peuvent être réduits à ces derniers.

**NP-intermédiaire** : Si P != NP, alors il existe une problème un problème Π ϵ NP tel que : Π !ϵ P, Π n’est pas NP-Complet.

**L** : La classe regroupe l’ensemble des langages reconnus en espace logarithmique O(log n). Cad L = PSPACE(log n).

**NL (NPSPACE (log n))** : C'est l'ensemble des problèmes de décision qui peuvent être décidés par des machines de Turing non déterministes dont l'espace de travail est borné par une fonction logarithmique.

**NL-Complet** : Problèmes de NL les plus difficiles à résoudre, tous les problèmes de NL peuvent être réduits à ces problèmes.

**CoNL** :

**DP** : Un langage L est dans la classe DP ssi il y a deux langages L1 ϵ NP et L2 ϵ coNP tel que L = L1 ∩ L2.

**PLS** : Contient les problèmes de recherche locale de la manière suivante : Un problème de recherche locale s’il existe trois applications polynomiales AL, BL, CL telles que :

* pour une instance I ϵ ΣΠ, l’application AL : ΣΠ → sol(I) donne une solution initiale pour I.
* pour toute solution x ϵ sol(I), l’application BL : ΣΠ x sol(i) → R calcul le score de x;
* pour toute solution x l’application CL : ΣΠ x sol(i) 26 sol(I) détermine si S un optimum local et, si ce n’est pas le cas, retourne la solution avec le meilleur score.

**PLS-Complet** : L’ensemble des problèmes d’optimums les plus difficiles à résoudre.

**SPACEf(n) (L = SPACE(log n))** : La classe des de problèmes pouvant être résolus, sur une instance de taille n, en utilisant un espace mémoire en O(f(n)).

**PSPACE** : Classe des langages décidés par une machine de Turing déterministe dont la complexité en espace (nombre de cases du ruban) est bornée par un polynôme. Autrement dit la classe de tous les problèmes qui peuvent être résolus en utilisant en espace mémoire polynomial en taille de leur instances : PSPACE = U SPACEnk

**NPSPACE** : Classe des langages acceptés par une machine de Turing déterministe dont la complexité en espace (nombre de cases du ruban utilisées) est bornée par un polynôme. On pose NL = NLSPACE(log n).

Nous avons les inclusions suivantes : L ⊆ NL ⊆ NC ⊆ P ⊆ NP ⊆ PSPACE = NPSPACE ⊆ EXPTIME ⊆ NEXPTIME ⊆ EXPSPACE = NEXPSPACE ;

Et : P ⊆ Co-NP ⊆ PSPACE.

**Les problèmes de décision** : On cherche à décider de l’existence d’une solution vérifiant certaines propriétés. La solution d’un problème de décision est : soit oui une telle solution existe, soit non elle n'existe pas.

**Les problèmes d’optimisation** : La catégorie de problèmes ou l’on cherche à déterminer la meilleure solution parmi celles vérifiant certaines propriétés imposées par la définition même du problème.

**Les problèmes d’optimums locaux** : Des problèmes de recherche locales permettant de trouver un optimum local.

### **Définitions :**

**Langages** : Un langage L sur S est un ensemble quelconque de chaînes construites à partir de symboles de S.

**Problème non trivial :**

**Classe de complexité close/fermée** : (classe de la définition C, < réduction (se réduit)) Cette dernière est close pour une réduction r si pour n’importe quelle paire de problèmes de décision Π1 et Π2 tel que Π1 < Π2 et Π2 appartient à C → Π1 appartient à C.

**Equivalence** : Si deux problèmes Πet Π’ vérifient Π<Π’ et Π’<Π on notera Π équivalent à Π’, ils seront équivalents pour les réductions many one. (Ex : Clique et Stable).

**NPO** : une problème de NPO est NP-Difficile ssi sa variante décisionnelle est une problème NP-Complet.

**Difficulté** : Soit une classe de complexité C, un problème est C difficile si il est au moins aussi difficiles que tous les autres problèmes de C. Si le problème est C-difficile et qu’il appartient à C, il est C-Complet.

**Certificat** : Étant donné un problème de décision, un certificat est une information que l'on ajoute à une donnée du problème, pour certifier (au sens où la vérification devient « facile »), que la réponse au problème pour cette donnée soit « oui » ou « non ».

**Hamiltonien** : En mathématiques, dans le cadre de la théorie des graphes, un chemin hamiltonien d'un graphe orienté ou non orienté est un chemin qui passe par tous les sommets une fois et une seule. Un cycle hamiltonien est un chemin hamiltonien qui est un cycle. Un graphe hamiltonien est un graphe qui possède un cycle hamiltonien.

**Circuit** : Cycle dans un graphe orienté. Ou cycle dans des graphes non orientés.

**Co-Problème** : question formulée de manière négative;

**Graphe complet :** En théorie des graphes, un graphe complet est un graphe simple dont tous les sommets sont adjacents deux à deux, c'est-à-dire que tout couple de sommets disjoints est relié par une arête. Si le graphe est orienté, on dit qu'il est complet si chaque paire de sommets est reliée par exactement deux arcs (un dans chaque sens).

**Chemin** : Dans un graphe orienté, un chemin d'origine x et d'extrémité y, est défini par une suite finie d'arcs consécutifs, reliant x à y. La notion correspondante dans les graphes non orientés est celle de chaîne.

**Stable** : En informatique théorique, le problème du stable maximum ou maximum independent set problem en anglais, est un problème d'optimisation qui consiste étant donné un graphe non orienté à trouver un stable de cardinal maximum, c'est-à-dire un sous-ensemble de sommets du graphe, le plus grand possible, tel que les éléments de ce sous-ensemble ne soient pas voisins.

### **Théorèmes :**

* Les classes P et NP sont closes pour la réduction
* Les réductions de Karp et de Turing sont transitives
* La variante décisionnelle d’un problème de NPO appartient à NP
* Le problème satisfaisabilité est NP-Complet
* Le problème 3-satisfaisabilité est NP-Complet
* Les problèmes Vertex-Cover, Clique et Stable sont des problèmes NP-Complet
* Le problème du circuit Hamiltonien est NP-Complet
* P ⊆ NP et P ⊆ coNP
* Le problème Tautologie est CoNP-Complet
* Accessibilité est NL-Complet pour les réductions en espace logarithmiques
* Les classes de complexité en espace NL et CoNL sont égales.
* EXACT Voyageur de commerce est DP-Complet.
* Max Circuit/Flip et Min Circuit/Flip sont PLS-Complets.

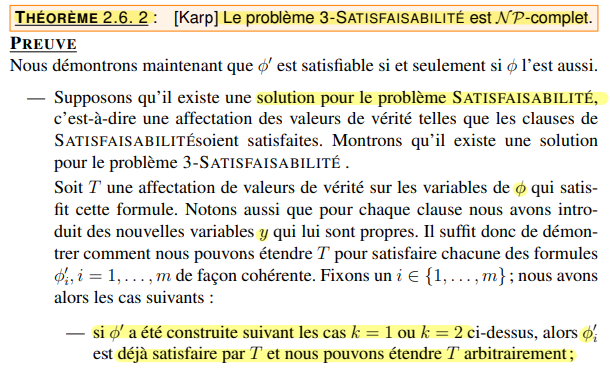
### **Remarques et Faits :**

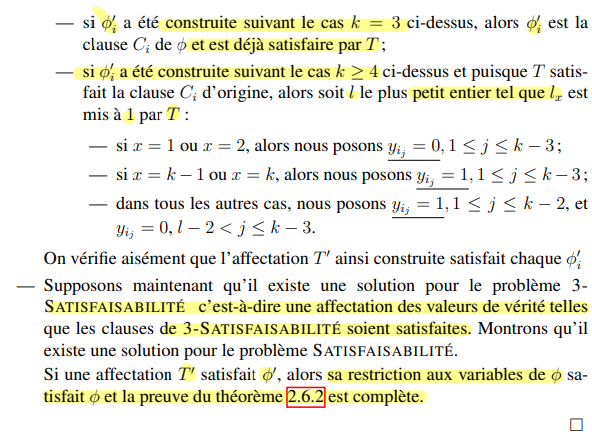
1. Tout langage L peut se voir comme un problème de décision, de la manière suivante : Entrée : Mot W, Question : Décider si W appartient à L
2. La relation de réduction < est réflexive et transitive (p17)
3. Si Π est NP-Complet alors ~Π est coNP-Complet
4. NL = coNL
5. SAT-UNSAT est DP-Complète.

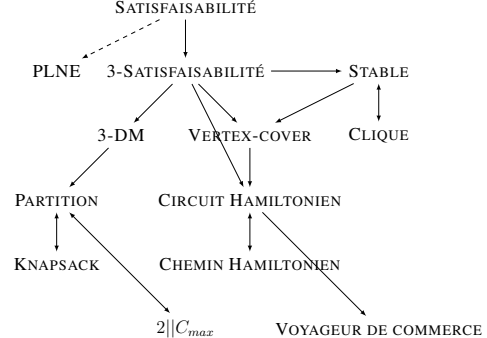
### **Exemples de réductions :**

**Réduction de Karp** : Étant donnée deux problèmes de décision Π1 et Π2, une réduction de Karp est une **fonction** f:IΠ1 → IΠ1, calculable en temps polynomial, telle que étant donné une solution pour f(I), nous sommes capables de trouver une solution I en temps polynomial en |I| (la taille de l’instance I).

**Réduction de Turing** : Une réduction de Turing d’un problème Π1 à un problème Π2 est un **algorithme** A1 qui résout Π1 en faisant appel à un algorithme A2 pour Π2 de manière telle que si A2 est polynomial, alors A1 l’est aussi.

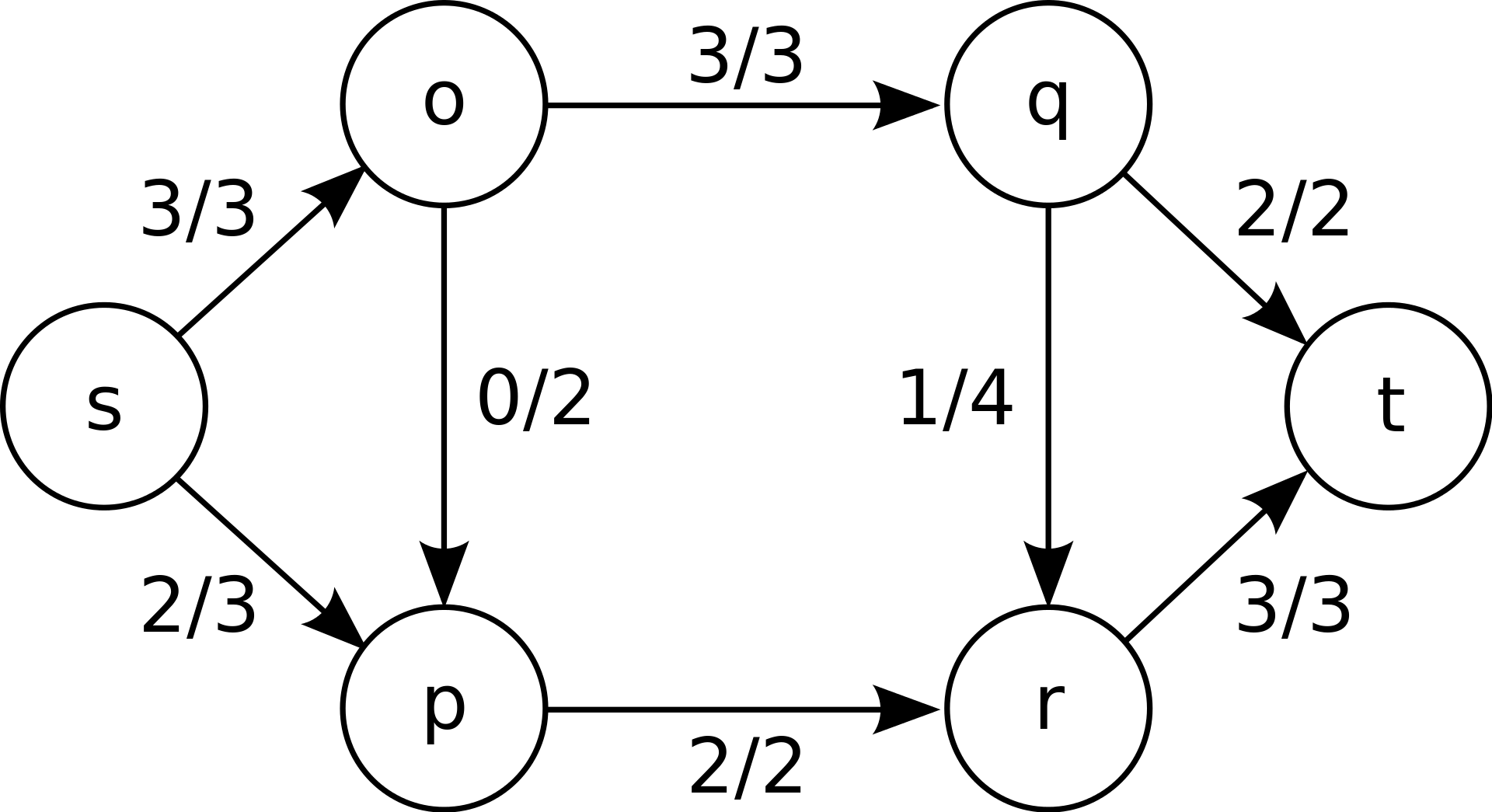






**Clique :** En informatique, le problème de la clique est un problème algorithmique qui consiste à trouver des cliques (sous-ensembles de sommets d'un graphe tous adjacents les uns aux autres, également appelés sous-graphes complets) dans un graphe.

**Flow** : Le problème de flot maximum consiste à trouver, dans un réseau de flot, un flot réalisable depuis une source unique et vers un puits unique qui soit maximum. Quelquefois[Quand ?], on ne s'intéresse qu'à la valeur de ce flot. Le s-t flot maximum (depuis la source s vers le puits t) est égal à la s-t coupe minimum du graphe, comme l'indique le théorème flot-max/coupe-min.



**Couplage parfait** : Soit un graphe simple. On appelle couplage un ensemble d'arêtes tel que deux arêtes quelconques de ne sont pas adjacentes. Un sommet est dit saturé par un couplage s'il est extrémité d'une arête de ce couplage. Un couplage est parfait s'il sature tous les sommets du graphe.

**Problème de sous-ensembles :** Le problème de la somme de sous-ensembles (en anglais : subset sum problem) est un problème de décision important en complexité algorithmique et en cryptologie. Le problème peut être décrit de la manière suivante : étant donné un ensemble {\displaystyle E}E de {\displaystyle n}n entiers, existe-t-il un sous-ensemble de {\displaystyle E}E dont la somme des éléments est nulle ? Par exemple, pour l'ensemble {-8, -3, -2, 4, 5}, la réponse est oui car la somme des éléments du sous-ensemble {-3, -2, 5} est nulle, par contre pour {-6, -1, 2, 3, 8} la réponse est non.