***Instances* du problème du sac à doc** chaque instance est définie par :
• le nombre d'objets, leurs poids et leurs valeurs • la capacité du sac.

**Problème d'optimisation combinatoire**

- un ensemble d'instances $\mathcal{I}$ ; chaque instance a une *taille* qu'on sait calculer (nb objets, taille du graphe, ...)
- pour chaque $I \in \mathcal{I}$, un ensemble $\mathcal{S}(I)$ de solutions « faisables »/« possibles » DISCRET (souvent fini) ;
- une fonction d'objectif $v : I \times \mathcal{S}(I) \to \mathbf{R}$.

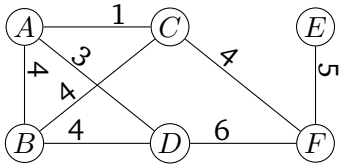
Pour une instance $I$ donnée, on cherche $S \in \mathcal{S}(I)$ qui maximise/minimise $v(I, S)$.

**Classes de problèmes**

NPO**:** la classe des problèmes d'optimisation dont les solutions faisables ont une taille bornée par un polynôme, et pour lesquels on a un algorithme qui sait vérifier si $S \in \mathcal{S}(I)$ et calculer $v(I, S)$ en *temps polynomial*.
PO**:** la sous-classe des problèmes de NPO pour lesquels on a un algorithme qui retourne une solution optimale en *temps polynomial*

(Temps polynomial = temps borné par un polynôme fonction de la taille des instances)

**Exemple** Étant donné un graphe $G$ :
NPO **:** trouver un circuit de longueur minimal passant par tous les sommets
PO **:** trouver un arbre couvrant minimal




**Algorithme de Kruskal**
1. Tant que c'est possible :
   (a) ajouter une arête de poids minimal qui ne créée pas un cycle

Les problèmes du sac-à-dos, de coloration de graphe, des déménageurs, ..., la PLNE sont dans NPO.

***Question théorique ouverte :*** par définition PO $\subseteq$ NPO ; mais a-t-on PO $=$ NPO ?
*On pense que non : il existe de nombreux problèmes pour lesquels on ne sait pas faire beaucoup mieux qu'énumérer toutes les solutions faisables (en nombre exponentiel)*

**SAT – un problème de référence en informatique théorique :**

- un ensemble de $n$ variables booléennes $x_1, \ldots, x_m$ ; $\Rightarrow 2n$ *littéraux* : $x_i$, $\neg x_i$
- un ensemble de $m$ clauses, c'est-à-dire de disjonctions de littéraux

$\Rightarrow$ cet ensemble est-il *satisfiable* ? C'est-à-dire : existe-t-il une affectation de valeurs aux $x_i$ telle que toutes les clauses soient vraies ?

Par exemple : $\mathcal{C} = \{\neg a \vee \neg c, \neg b \vee \neg c, \neg a \vee c, \neg b \vee c\}$ est satisfiable, mais pas $\mathcal{C} \cup \{a \vee b \vee \neg c\}$.

**Problème de décision =** un ensemble d'instances $\mathcal{I}$, et une fonction $d : \mathcal{I} \longrightarrow \{\text{vrai}, \text{faux}\}$.
Pour une instance $I$ donnée, on cherche à savoir si $d(I) = \text{vrai}$.

**Les classes `P` et `NP`**

NP **:** la classe des problèmes de décision pour lesquels on a un *vérificateur* $V$ qui termine en temps polynomial : pour chaque instance $I$, il existe un ensemble $\mathcal{S}(I)$ tel que pour toute $S \in \mathcal{S}(I)$, $V(S) \in \{\text{vrai}, \text{faux}\}$ est calculé en temps polynomial, et $d(I) = \text{vrai}$ si et seulement si il existe $S \in \mathcal{S}(I)$ telle que $V(S) = \text{vrai}$.
P **:** la sous-classe des problèmes de NP pour lesquels on a un algorithme qui calcul $d$ en *temps polynomial*.

**Lien entre optimisation et décision** pour un problème de minimisation $(\mathcal{I}, \mathcal{S}, v)$ donné, on peut étudier le problème de décision :

pour $k \in \mathbf{N}$ et $I \in \mathcal{I}$ donnés, existe-t-il $S \in \mathcal{S}(I)$ telle que $v(S) \leq k$ ?

- Si on sait résoudre le problème d'optimisation en temps polynomial, alors on sait résoudre aussi le problème de décision en temps polynomial
- Réciproquement, si on sait résoudre le problème de décision en temps polynomial, et si on connait une borne supérieur $K$ de l'optimum (et en général, on le sait), alors on sait résoudre le problème d'optimisation en résolvant au plus $\log_2(K)$ instances du problème de décision

**Problèmes NP-complets :** il y a, dans la classe NP, une classe de problèmes tels que si on savait résoudre l'un de ces problèmes en temps polynomial, alors on saurait résoudre en temps polynomial tous les problème de la classe NP. Ces problèmes sont dit *NP-complets*.

**Théorème** SAT est NP-complet. *(Théorème de Cook / Levin)*

**Remarque** On montre facilement que SAT est dans NP. Montrer que c'est NP-complet est plus compliqué...

**Théorème** 3SAT est NP-complet : dans 3SAT, on n'autorise que des clauses ayant au plus 3 littéraux.

**Preuve** On montre d'abord que 3SAT est dans NP... On montre ensuite qu'on peut *réduire* 3SAT à SAT, c'est-à-dire qu'on peut traduire *en temps polynomial* toute instance de SAT en une instance de 3SAT... ■

On dit qu'un problème d'optimisation est NP-complet lorsque le problème de décision correspondant l'est.
Les problèmes suivants sont NP-complets : le voyageur de commerce, le sac-à-dos, les déménageurs, la PLNE, ... (dans leur version « décision »)

**Exercice 1** Un *graphe biparti* est un graphe $(S, A)$ dont l'ensemble de sommets $S$ est partitionné en deux sous-ensembles $F$ et $G$, tels que chaque arête de $A$ a une extrémité dans $F$ et l'autre dans $G$. Le problème de l'*appariement maximal biparti* (ou *couplage* maximal biparti) est alors de trouver un sous-ensemble maximal $C$ de $A$ tel qu'aucun sommet du graphe n'apparaît sur deux arêtes de $C$ (ou, autrement dit, tel que $C$ ne contient pas deux arêtes adjacentes).

Question 1.1 Montrer que ce problème peut être résolu en temps polynomial.

Le problème d'*appariement 3D* est similaire, sauf que $S$ est maintenant partitionné en trois sous-ensembles $S_1$, $S_2$ et $S_3$, on a des arêtes entre $S_1$ et $S_2$, d'autres arêtes entre $S_2$ et $S_3$ : le but est de former le plus grand nombre possible de triplets de $S_1 \times S_2 \times S_3$ qui correspondent à des arêtes du graphe, et sans qu'aucun sommet n'apparaissent deux fois.

Question 1.2 Montrer que toute instance de 3-SAT peut être réduite en temps polynomial à une instance d'appariement 3D. Qu'en conclut-on ?

**Exercice 2** La version « décision »du problème des déménageurs consiste à décider, pour un entier naturel $k$ donné, s'il existe une solution avec moins de $k$ boites.

Question 2.1 Démontrez que le problème des déménageurs (dans sa version décision) est dans NP.

Question 2.2 Démontrez que le problème des déménageurs est NP-complet, en donnant une réduction du problème « 2-partition », connu pour être lui-même NP-complet :

> **2-partition** On a un ensemble d'entiers $S$, on veut savoir s'il existe une partition de $S$ en deux sous-ensembles $S_1$ et $S_2$ tels que la somme des entiers de $S_1$ est égale à la somme des entiers de $S_2$.

Question 2.3 Donnez une expression du problème en programmation linéaire en nombres entiers. On prendra soin de bien expliquer la signification des valeurs de chacune des variables introduites, ainsi que des contraintes nécessaires.