# TD5 – Réductions, $\mathsf{NP}$-difficulté, $\mathsf{NP}$-complétude – Correction

**Exercice 1.** *Problèmes* $\mathsf{NP}$*-complets*

1. **Ensemble indépendant** $\in \mathsf{NP}$ car il est résolu par l'algorithme non-déterministe suivant :

   Sur l'entrée $(G, k)$ :

   1. *deviner* un ensemble $X$ de $k$ sommets de $G$ ;
   2. *vérifier* que $X$ est un ensemble dominant de $G$.

   Cette procédure non-déterministe correspond bien à une machine de Turing non-déterministe polynomiale puisque :

   — La taille de l'objet deviné est polynomiale en la taille de l'entrée $(G, k)$ : $|X|$ est inférieure à $|G|$ et *a fortiori* à $|(G, k)|$ ;

   — La phase de vérification peut se faire en temps polynomial en $|(G, k)|$ : il suffit de vérifier que les $k$ sommets devinés sont deux à deux non-adjacents, ce qui se fait facilement en temps linéaire en la taille de $G$.

   **Clique $\leq_{\mathrm{p}}^{m}$ Ensemble indépendant**

   Pour un graphe non orienté $G = (V, E)$, on note $\overline{G}$ le graphe obtenu à partir de $G$ en inversant les arêtes et les « non-arêtes ». Autrement dit : $\overline{G} = (V, V^2 \setminus E)$. Un ensemble de sommets $X$ est une clique (toutes les arêtes) de taille $\geq k$ de $G$ si, et seulement si, c'est un ensemble indépendant (aucune arête) de taille $\geq k$ de $\overline{G}$. Par conséquent, l'application $r$ qui à toute instance $(G, k)$ de **Clique** associe l'instance $(\overline{G}, k)$ de **Ensemble indépendant** satisfait :

   $$(G, k) \in \textbf{Clique} \iff r(G, k) \in \textbf{Ensemble indépendant}.$$

   Cette application est donc une réduction du premier problème au second. De plus, cette réduction est calculable en temps polynomial : pour calculer $r(G, k) = (\overline{G}, k)$ à partir de $(G, k)$, il suffit « d'inverser » $G$ et ceci peut se faire en temps linéaire (par exemple, on obtient la matrice d'adjacence de $\overline{G}$ en inversant tous les $0$ et les $1$ de la matrice de $G$, à l'exception des $0$ placés sur la diagonale puisque les graphes que nous considérons sont sans boucle). Il s'ensuit que **Clique** se réduit polynomialement à **Ensemble indépendant**, et la $\mathsf{NP}$-difficulté de **Clique** entraîne celle de **Ensemble indépendant**.

   Finalement, **Ensemble indépendant** appartient à $\mathsf{NP}$ et est $\mathsf{NP}$-difficile : c'est un problème $\mathsf{NP}$-complet.

2. **Node Cover** $\in \mathsf{NP}$ car ce problème est résolu par l'algorithme non-déterministe suivant :

   Sur l'entrée $(G, k)$ :

   1. *deviner* un ensemble $X$ de $k$ sommets de $G$ ;
   2. *vérifier* que $X$ est une couverture (des arêtes par les sommets) de $G$.

   Cette procédure non-déterministe correspond bien à une machine de Turing non-déterministe polynomiale puisque :

   — La taille de l'objet deviné est polynomiale en la taille de l'entrée $(G, k)$ : $|X|$ est inférieure à $|G|$ et *a fortiori* à $|(G, k)|$ ;

— La phase de vérification peut se faire en temps polynomial en $|(G, k)|$ : il suffit de vérifier que chaque arête de $G$ a une extrémité dans $X$, ce qui se fait facilement en temps $\mathcal{O}(|G|)$.

**Clique $\leq_{\mathrm{p}}^{m}$ Node Cover.**

On a vu qu'un ensemble $X$ de sommets de $G$ est une clique de $G$ ssi c'est un ensemble indépendant de $\overline{G}$. Or $X$ est un ensemble indépendant dans $\overline{G}$ si aucune arête de $\overline{G}$ ne lie deux sommets de $X$, c'est-à-dire si toute arête de $\overline{G}$ a au moins une extrémité dans $V \setminus X$. Par conséquent : $X$ est une clique de $G$ si et seulement si toute arête de $\overline{G}$ a une extrémité dans $V \setminus X$. Il s'ensuit : $X$ est une clique de $G$ de taille $\geq k$ si et seulement si $V \setminus X$ est une couverture de $\overline{G}$ de taille $\leq |V| - k$. Par conséquent, l'application $r : (G, k) \mapsto (\overline{G}, |V| - k)$ satisfait

$$(G, k) \in \textbf{Clique} \iff r(G, k) \in \textbf{Node Cover}.$$

C'est donc une réduction de **Clique** vers **Node Cover**.

Cette réduction est clairement calculable en temps polynomial : pour calculer $r(G, k) = (\overline{G}, |V| - k)$, il suffit d'inverser $G$ en $\overline{G}$ et de remplacer $k$ par $|V| - k$, ce qui se fait en temps linéaire en $|(G, k)|$. Il s'ensuit que **Clique** se réduit polynomialement à **Node Cover**, et la NP-difficulté de **Clique** entraîne celle de **Node Cover**.

Finalement, **Node Cover** appartient à NP et est NP-difficile : c'est un problème NP-complet.

**3.** **Set packing** $\in$ NP, puisqu'étant donnés $k$ sous-ensembles, un certificateur (dans la caractérisation existentielle de NP) peut vérifier efficacement (en temps polynomiale) qu'ils sont disjoints deux à deux.

Montrons maintenant que **Ensemble indépendant** $\leq_m^p$ **Set packing**, ce qui nous permettra de conclure car **Independent Set** est NP-complet.

Étant donné une instance d'ensemble indépendant sur un graphe $G(V, E)$, on créé une collection d'ensembles, où pour chaque sommet $v \in V$, il y a un ensemble $S_v$ contenant toutes les arêtes de $E$ adjacentes à $v$. On prend $\ell = k$.

$\boxed{\Rightarrow}$ Soit $(G = (V, E), k) \in$ **Ensemble Indépendant**, et soit $V' \subseteq V'$ un ensemble indépendant de taille $k$. Alors $\{S_{v'}\}_{v' \in V'}$ est un set packing de taille $\ell = k$ : les éléments de ces $\ell$ ensembles sont les arêtes adjacentes à des sommets qui ne sont pas voisins dans $G$, donc aucune arête n'apparaît deux fois (pour cela il faudrait deux sommets voisins) et les ensembles $\{S_{v'}\}_{v' \in V'}$ sont mutuellement disjoints. C'est-à-dire $(\{S_v\}_{v \in V}, \ell) \in$ **Set packing**.

$\boxed{\Leftarrow}$ Si $(\{S_v\}_{v \in V}, \ell) \in$ **Set packing**, alors il existe $\ell$ sous-ensemble mutuellement disjoints. L'ensemble de sommets correspondant à ces $\ell$ sous-ensembles est un ensemble indépendant de taille $k = \ell$ : deux sommets n'ont aucune arête adjacente en commun, ils ne sont donc pas voisins. C'est-à-dire $(G = (V, E), k) \in$ **Ensemble Indépendant**.

**4.** **Set covering** $\in$ NP, puisqu'étant donnés $k$ sous-ensembles $\{S_{j_i}\}_{i \in \{1,\ldots,k\}}$, un certificateur (dans la caractérisation existentielle de NP) peut vérifier efficacement (en temps poly) que l'union de ces $k$ sous-ensemble contient (recouvre) tous les éléments de l'union de tous les sous-ensembles de la famille $\{S_j\}_{j \in \{1,\ldots,m\}}$.

Montrons maintenant que **Node cover** $\leq_m^p$ **Set covering**, ce qui nous permettra de conclure car **Node cover** est NP-complet.

Etant donnée une instance $(G = (V, E), \ell)$ de **Node cover**, nous allons construire une instance du problème **Set covering**. Nous définissons une famille de $m = |V|$ sous-ensembles comme suit : pour chaque sommet $v \in V$ on définit l'ensemble $S_v$ des arêtes adjacentes à $v$. On prend $k = \ell$. Cette construction se fait en temps poly (parcours des sommets et des voisins de chaque sommet).

$\boxed{\Rightarrow}$ Supposons que $G$ ait une couverture des arêtes par les sommets de taille $\ell$. Soit $V'$ un tel ensemble de sommets. Alors $\{S_{v'}\}_{v' \in V'}$ est une couverture de taille $k = \ell$ des ensembles de la famille $\{S_v\}_{v \in V}$. En effet, pour toute arête de $G$ (rappelons que les éléments des ensembles $S_v$ sont des arêtes de $G$), puisque $V'$ est une couverture des arêtes par les sommets de $G$, on a une extrémité dans $V'$, et donc l'arête dans un $S_{v'}$ avec $v' \in V'$.

$\boxed{\Leftarrow}$ Supposons que $\{S_v\}_{v \in V}$ ait une couverture de taille $\ell$. Alors en prenant $V'$ l'ensemble des sommets associés aux sous-ensembles de cette couverture (chaque sous-ensemble correspond à un sommet), on obtient une couverture des arêtes par les sommets dans $G$. En effet, chaque arête de $E$ est élément d'un sous-ensemble, donc est couverte (dans le monde des sous-ensemble) par un $\{S_{v'}\}_{v' \in V'}$, donc est couverte (dans le monde des graphes) par un $v' \in V'$.

5. **Feedback node set** $\in$ NP, car on a l'algorithme non-déterministe suivant : deviner un sous ensemble $V' \subseteq V$ (taille poly), et vérifier qu'il s'agit bien d'un feedback node set :
   **(a)** vérifier la taille de $V'$ inférieure ou égale à $\ell$ (1 étape de comparaison),
   **(b)** vérifier que tout cycle dirigé de $G$ a une extrémité dans $V'$ avec des parcours : depuis tout sommet on doit visiter un sommet de $V'$ avant de cycler, c'est-à-dire avant de revister un sommet (temps polynomial en la taille du graphe).

   Refuser si une des conditions n'est pas vérifiée, accepter sinon. On aura bien une branche d'exécution non-déterministe qui accepte ssi il existe un feedback node set $V'$.

   Montrons maintenant que **Node cover** $\leq_m^p$ **Feedback node set**, ce qui nous permettra de conclure car **Node cover** est NP-complet.

   Soit $G = (V, E), \ell$ une instance de **Node cover**. On construit l'instance $G' = (V', A'), k'$ de **Feedback node set** suivante :

$$V' = V \text{ et } A' = \{(u, v) \mid \{u, v\} \in E\} \text{ et } k' = \ell.$$

   (Nous créons un petit cycle orienté dans $G'$ pour chaque arrête de $G$.) Cette construction se fait en temps poly avec un simple parcours des arêtes de $E$.

   $\boxed{\Rightarrow}$ Si $(G = (V, E), \ell) \in$ **Node cover**, alors il existe une couverture $C \subseteq V$ des arrêtes $E$ par les sommets de $C$, telle que $|C| \leq \ell$. Alors en prenant le même $C \subseteq V'$ on obtient un feedback node set de $G'$ tel que $|C| \leq \ell = k'$ : tous les petits cycles orientés de longueur deux dans $G'$ ont un sommet dans $C$ (puisque qu'ils correspondent aux arrêtes de $E$ qui sont couvertes par $C$ dans $G$). Les plus grands cycles ont également un sommet dans $C$ car ils contiennent les deux sommets d'un cycle de longueur deux, et d'après ce qui précède l'un de ces deux sommets est dans $C$. Donc $(G' = (V', A'), k') \in$ **Feedback node set**.

   $\boxed{\Leftarrow}$ Si $(G' = (V', A'), k') \in$ **Feedback node set**, alors il existe un feedback arc set $F \subseteq V'$ tel que $|F| \leq k'$ et tout cycle orienté a un sommet dans $F$. Puisque dans $G'$ nous avons un cycle de longueur deux pour chaque arrête de $G$, toutes les arrêtes de $G$ ont un sommet dans $F$. Donc $F'$ est une couverture des arrêtes $E$ telle que $|F| \leq \ell$, c'est-à-dire $(G = (V, E), \ell) \in$ **Node cover**.

6. **Exactly-1 3-SAT** $\in$ NP car on peut déviner une affectation aux variables et vérifier si elle satisfait *exactement un littéral par clause* en temps polynomial (il s'agit d'une simple variante de l'algorithme correspondant pour **3-SAT**).

   Montrons que **3-SAT** $\leq_m^p$ **Exactly-1 3-SAT**. Soit $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ une formule en forme normale conjonctive avec trois littéraux par clause et $n$ variables. Pour chaque clause $C_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$ de trois littéraux, on construit les trois clauses suivantes :

   $$T_i = (\neg \ell_{i,1} \vee x_i \vee y_i) \wedge (\ell_{i,2} \vee y_i \vee z_i) \wedge (\neg \ell_{i,3} \vee z_i \vee w_i)$$

   avec quatre nouvelles variables $x_i, y_i, z_i, w_i$. Soit $\phi' = T_1 \wedge T_2 \wedge \cdots \wedge T_m$. Alors $\phi'$ est aussi une formule en FNC avec trois littéraux par clause, $n + 4m$ variables et $3m$ clauses, et on peut la construire en temps polynomial à partir de $\phi$. On va prouver qu'il existe une affectation qui satisfait $\phi$ si et seulement si il existe une affectation qui satisfait exactement un littéral par chaque clause de $\phi'$.

   Supposons que $\phi$ ne soit pas satisfaisable. Alors, pour chaque affectation des variables de $\phi$, il existe une clause $C_i$ qui n'est pas satisfaite, donc les trois littéraux $\ell_{i,1}$, $\ell_{i,2}$ et $\ell_{i,3}$ sont faux. Montrons que c'est impossible de satisfaire correctement (c'est-à-dire, en satisfaisant exactement un littéral par clause) au moins l'une des trois clauses correspondants de $T_i$ dans $\phi'$. Comme $\ell_2$ est faux, pour satisfaire $(\ell_2 \vee y_i \vee z_i)$ il faut que soit $y_i$, soit $z_i$ soit vraie. Mais si $y_i$ est vraie, alors deux littéraux dans $(\neg \ell_1 \vee x_i \vee y_i)$ seraient vrais, et si $z_i$ est vraie, alors deux littéraux dans $(\neg \ell_e \vee z_i \vee w_i)$ seraient vrais. Donc $\phi'$ n'est pas satisfaisable avec exactement un vrai littéral par clause.

   Vice-versa, supposons que $\phi$ soit satisfaisable. Alors il existe une affectation qui rend vraie chaque clause $C_i$. Montrons comment construire une affectation qui rend vrai exactement un littéral par clause dans $\phi'$.
   - Si $C_i$ est satisfaite parce que $\ell_{i,2}$ est vrai, alors on donne la valeur faux à $y_i$ et $z_i$, et on choisit la valeur de $x_i$ et $w_i$ en fonction de la valeur de $\ell_{i,1}$ et $\ell_{i,1}$, pour avoir exactement un vrai littéral par clause dans $T_i$.
   - Si $\ell_{i,2}$ est faux et $\ell_{i,1}$ et $\ell_{i,3}$ sont vrais, alors on choisit la valeur vraie pour $x_i$ et $z_i$, fausse pour $y_i$ et $w_i$. Cela satisfait correctement un littéral par clause dans $T_i$.
   - Si seulement $\ell_{i,1}$ est vrai, on donne la valeur vraie à $y_i$ et fausse à $x_i$, $z_i$ et $w_i$.
   - Symétriquement, si seulement $\ell_{i,3}$ est vrai, on donne la valeur vraie à $z_i$ et fausse à $x_i$, $y_i$ et $w_i$.

   Dans chaque cas, on obtient une affectation qui satisfait exactement un littéral par clause dans $T_i$ et, comme le raisonnement est le même pour chaque clause $C_i$, dans la totalité de formule $\phi'$.

7. **0-1 integer programming** $\in$ NP, car on a l'algorithme non-déterministe suivant : deviner un vecteur booléen $x$ de taille $n$ (taille poly), puis accepter si $Mx = d$, refuser sinon (la multiplication d'une matrice par un vecteur prend un temps poly, tout comme le test d'égalité). On aura bien une branche d'exécution non-déterministe qui accepte ssi il existe un tel vecteur $x$.

   Montrons maintenant que **Exactly-1 3-SAT** $\leq_m^p$ **0-1 integer programming**, ce qui nous permettra de conclure car **3-SAT** est NP-complet. Notre idée pour cette réduction sera qu'une valuation sur $n$ variables peut être vue comme un vecteur booléens de taille $n$.

   Soit $\phi$ une formule en 3-CNF, avec $m$ le nombre de clauses et $n$ le nombre de variables $(x_1, \ldots, x_n)$. Nous allons construire une matrice de taille $m \times n$. Pour $i \in \{1, \ldots, m\}$ et

$j \in \{1, \ldots, n\}$ on prend

$$M_{i,j} = \begin{cases} 1 & \text{si } x_j \in C_i \\ -1 & \text{si } \neg x_j \in C_i \\ 0 & \text{sinon} \end{cases} \quad \text{avec } C_i \text{ la } i^{\text{ème}} \text{ clause.}$$

La matrice $M$ encode la formule : la ligne $i$ est la clause $C_i$ où la colonne $j$ vaut $1$ si la variable $x_j$ apparaît positivement dans $C_i$, vaut $-1$ si elle apparaît négativement, et vaut $0$ si elle n'apparaît pas. Le vecteur objectif est $d_i = 1-$(nombre de littéraux négatifs dans $C_i$) pour tout $i$. Cette construction se fait en temps polynomial, en parcourant la formule pour construire la matrice et le vecteur objectif.

**Un conseil :** Si vous ne comprenez pas où l'on veut en venir, construisez la matrice $M$ et le vecteur $d$ correspondants à la formule

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

et calculez $Mx$ pour la valuation $x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1$ (qui satisfait la formule avec exactement un littéral qui satisfait chaque clause).

$\boxed{\Leftarrow}$ Si il existe un vecteur booléen $x$ de taille $n$ tel que $Mx = d$ (c'est-à-dire $(M, d) \in$ **0-1 integer programming**), alors $x$ correspond à une valuation satisfaisant $\phi$. En effet, puisque $Mx = d$ on aura pour chaque ligne $i$ de $M$ un littéral qui satisfait la clause $C_i$ (soit $x_j = 1$ et $M_{i,j} = 1$, soit $x_j = 0$ et $M_{i,j} = -1$). Chaque clause est satisfaite par exactement un littéral, donc $\phi \in$ **3-SAT**.

$\boxed{\Rightarrow}$ Si $\phi \in$ **Exactly-1 3-SAT** (c'est là que la restriction est utile), alors la valuation qui satisfait $x$ correspond à un vecteur booléen tel que $Mx = d$. En effet, parmi les trois littéraux, si aucun ne satisfait la formule on aurait $(Mx)_i = -$nombre de littéraux négatifs dans $C_i$), et si un littéral satisfait la formule il donnera $(Mx)_i = d_i$. Donc si $\phi \in$ **Exactly-1 3-SAT** on aura $(M, d) \in$ **0-1 integer programming**.