CH.4 PROBLEMES $\mathcal{NP}$ - COMPLETS

- 4.1 Les classes $\mathcal{P}$ et $\mathcal{NP}$: problèmes $\mathcal{NP}$-complets
- 4.2 Deux exemples de problèmes $\mathcal{NP}$-complets
- 4.3 La méthode de séparation-évaluation
- 4.4 La programmation dynamique

4.1 Les classes $\mathcal{P}$ et $\mathcal{NP}$: problèmes $\mathcal{NP}$-complets

Pour un certain nombre de problèmes (chemins hamiltoniens, coloriage des graphes, problèmes d'emplois du temps, factorisation des nombres entiers), une solution peut être difficile à trouver, mais, si on en propose une, il est facile (= polynomial) de vérifier si elle convient.

La plupart des problèmes utiles sont soit résolubles au moyen d'un algorithme polynomial, soit on peut vérifier si une solution proposée en est bien une en un temps polynomial.

Un examen attentif de ce deuxième type de problèmes a donné lieu, en 1971, à un travail de Cook, à Toronto, intitulé "The Complexity of Theorem Proving Procedures" où est introduite la notion de $\mathcal{NP}$-complétude. De façon inattendue, cette propriété abstraite d'un problème très artificiel se trouve être satisfaite par plusieurs centaines ou milliers de problèmes concrets.

.

La question de savoir si $\mathcal{P} = \mathcal{NP}$ est le problème non résolu le plus important de l'informatique théorique, figurant parmi les 7 problèmes mathématiques non résolus jugés les plus importants par l'institut Clay et dotés chacun d'un prix d'un million de dollars.

La classe $\mathcal{P}$ est celle des problèmes pour lesquels il existe un algorithme polynomial en la taille des données permettant de les résoudre. Les problèmes décrits dans les chapitres précédents sont polynomiaux.

La classe $\mathcal{NP}$ est celle des problèmes pour lesquels on peut vérifier en un temps polynomial si une solution proposée convient.

Un problème *P* est polynomialement réductible à un autre problème *Q* s'il existe un algorithme polynomial pemettant de ramener la recherche d'une solution de *P* à celle d'une solution de *Q*.

Si *P* est polynomialement réductible à un problème $\mathcal{P}$, il est lui-même $\mathcal{P}$.
Le **théorème de Cook** établit que tout problème $\mathcal{NP}$ est polynomialement réductible au problème SAT de la satisfiabilité des fonctions booléennes.

En fait, beaucoup de problèmes $\mathcal{NP}$ sont **équivalents** à SAT. Ces problèmes sont appelés **$\boldsymbol{\mathcal{NP}}$-complets**. C'est-à-dire que si un seul de ces problème est résoluble au moyen d'un algorithme polynomial, **tous** les problèmes $\mathcal{NP}$ le sont .

Lorsqu'un problème est connu pour être $\mathcal{NP}$-complet, ou pire, on peut donc raisonnablement penser qu'il est inutile d'en chercher une solution sous forme d'un algorithme de complexité polynomiale.

C'est le cas pour deux problèmes qui nous serviront d'exemples :
le problème du *sac à dos* et celui du *voyageur de commerce*.

4.2 Deux exemples de problèmes $\mathcal{NP}$-complets

Les problèmes considérés sont polynomialement réductibles à des problèmes d'optimisation linéaire en nombres entiers.

Optimisation linéaire :
trouver un n-uplet $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ de nombres réels qui maximisent une forme linéaire $\boldsymbol{c}\,\boldsymbol{x}$ en respectant des conditions $\boldsymbol{A}\,\boldsymbol{x} \leq \boldsymbol{b}$ et $\boldsymbol{x} \geq \mathbf{0}$.
En fait, si les coefficients sont rationnels, les solutions éventuelles sont rationnelles. Des algorithmes polynomiaux existent. Une variante de l'algorithme du pivot de Gauss permet de résoudre de tels problèmes.

Si les coefficients sont des entiers relatifs et les solutions des entiers positifs ou nuls, on sait que le problème de l'optimisation linéaire (en nombres entiers) est $\mathcal{NP}$-complet (attention! la taille en mémoire d'un entier est le logarithme de cet entier...)

- Le problème du sac à dos.

On a des objets de volumes $v_1, \ldots, v_n$ et d'utilités $u_1, \ldots, u_n$. On veut mettre dans un sac à dos de volume V un nombre x_i de chaque objet.
On veut maximiser l'utilité totale.
C'est un problème d'optimisation linéaire en nombres entiers :
Maximiser $\sum_i u_i\, x_i$ sous les $n + 1$ contraintes $x_i \geq 0$ et $\sum_i v_i\, x_i \leq V$.

Par exemple, maximiser $10\,x_1 + 8\,x_2 + 5\,x_3$ avec les contraintes de positivité (sous-entendues) et la contrainte $6\,x_1 + 5\,x_2 + 4\,x_3 \leq 9$.

- Le problème du voyageur de commerce.

On considère le graphe non-orienté complet à n sommets K_n. Chaque arête de ce graphe a un poids positif ou nul p_{uv}. On cherche un cycle passant une fois et une seule par chaque sommet (on dit hamiltonien) et de poids total minimum.

Ce problème peut être traduit en un problème d'optimisation linéaire en nombres entiers.

Les variables sont x_{uv}, avec $x_{uv} \geq 0$ et $x_{uv} \leq 1$. La forme est à minimiser. c'est $\sum_{u,\,v} p_{uv}\, x_{uv}$, avec les contraintes supplémentaires qu'en chaque sommet deux arêtes exactement sont choisies, $\sum_u x_{uv} = 2$ pour tout v, et que le cycle ne se décompose pas en cycles plus petits, donc pour tout sous-ensemble Y de sommets le nombre d'arêtes ayant ses deux extrémités dans Y est strictement plus petit que le cardinal de Y, c'est-à-dire que pour tout $Y \neq K_n$, $\sum_{u,\,v} x_{uv} < |Y|$, la somme portant sur tous les sommets u et v de Y.

La recherche d'une solution exacte de ces deux problèmes peut être impraticable. Néanmoins, il existe des méthodes permettant d'obtenir des solutions approchées. Ce sont les diverses méthodes *heuristiques*.

Même pour trouver une solution exacte, la connaissance de solutions approchées permet de rendre l'algorithme meilleur, tout en en conservant, bien sûr, la complexité exponentielle.

- Heuristique pour le sac à dos.

On peut chercher une solution rationnelle par une méthode de simplexe et arrondir à l'entier inférieur. En fait, dans ce cas, la méthode du simplexe est inutile. On trouve une approximation entière en considérant l'objet ayant le plus grand rapport utilité/volume et en remplissant le sac avec le maximum d'objets par ordre décroissant de ces rapports, par un algorithme glouton. On vérifie aussi facilement que, si on ne se limite pas aux solutions entières, on trouve la solution optimale en ne prenant que l'objet ayant ce plus grand rapport.

Dans l'exemple maximiser 10 x_1 + 8 x_2 + 5 x_3 avec les contraintes de positivité (sous-entendues) et la contrainte 6 x_1 + 5 x_2 + 4 $x_3 \leq 9$,

On a comme rapports respectivement 10/6, 8/5 et 5/4, déjà classées par ordre décroissant. Une solution rationnelle est ici $x_1 = 1{,}5$, $x_2 = x_3 = 0$, d'utilité totale 15.

Une solution entière approchée est $x_1 = 1$, $x_2 = x_3 = 0$, d'utilité totale 10.

On peut déjà dire que, pour une solution entière optimale, l'utilité est de 10 au moins et de 15 au plus.

- Heuristique pour le voyageur de commerce.

On peut partir d'un cycle hamiltonien, voire en essayer quelques-uns et les modifier localement en essayant d'améliorer le poids total (on y reviendra).

Une autre heuristique (due à Christofidès) consiste à remarquer que, si on enlève une arête à un cycle hamiltonien de poids minimum, on a un arbre recouvrant de poids minimum pour le graphe restant.

On peut donc partir d'un arbre recouvrant de poids minimum pour le graphe, puis coupler les sommets de degré impair de cet arbre par un couplage de poids minimum, deux choses qu'on sait faire en temps polynomial. Le résultat n'est pas nécessairement hamiltonien. Mais on peut sauter par-dessus des sommets déjà rencontrés. On obtient ainsi un circuit hamiltonien dont on peut prouver que le poids est au plus 1,5 fois le poids d'un circuit de poids minimum. Le tout est accompli en un temps cubique.

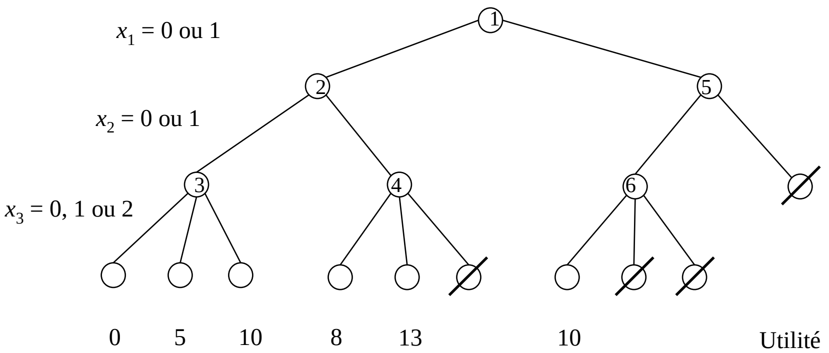
4.3 La méthode de séparation-évaluation

On peut représenter les solutions d'un problème sous forme d'un arbre. à chaque noeud, on fait un choix d'une valeur possible pour une variable.

Les solutions sont donc disposées aux feuilles. La recherche d'une solution optimale revient alors à regarder toutes les feuilles et à retenir celle qui maximise (ou minimise) l'utilité (ou le poids). Pour le sac à dos, l'ensemble des solutions est en nombre au moins 2^n. D'où la complexité exponentielle. Pour le voyageur de commerce, il y a 2^a solutions au maximum.

La méthode de séparation-évaluation consiste à abandonner l'exploration de branches de l'arbre lorsqu'on peut établir que toutes les solutions qui en découlent sont moins bonnes qu'une solution qu'on possède déjà.
Cette solution *a priori*, obtenue par une méthode heuristique est par ailleurs actualisée lors du parcouurs de l'arbre.

Dans l'exemple du sac à dos, $10\,x_1 + 8\,x_2 + 5\,x_3$ avec les contraintes de positivité (sous-entendues) et la contrainte $6\,x_1 + 5\,x_2 + 4\,x_3 \leq 9$, l'arbre des solutions complet est :



Certains noeuds sont barrés parce qu'on sait déjà que la contrainte de volume sera violée.

Pour limiter l'exploration, on va utiliser une borne inférieure (borne) correspondant à une solution exacte connue et une borne supérieure (évaluation) fournie par une solution rationnelle.

Ces deux bornes sont au départ de 10 et de 15. On explore l'arborescence en profondeur. A chaque nouveau noeud, on réévalue la borne supérieure, en fonction des choix déjà faits. Cette évaluation peut être approchée (solution non entière) ou exacte (solution entière).

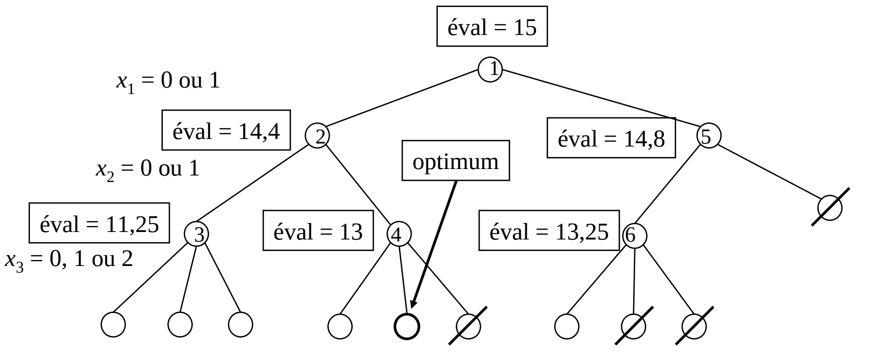
Si cette évaluation est strictement inférieure à la borne inférieure courante + 1, il est inutile de continuer à explorer sous le noeud.

Si l'évaluation est exacte et supérieure à la borne inférieure courante, on a trouvé une solution entière meilleure. On actualise donc la borne inférieure et on continue.

Sinon, on continue.

Dans notre exemple, en faisant une exploration en profondeur, on trouve successivement :

Maximiser $10\ x_1 + 8\ x_2 + 5\ x_3$ avec la contrainte $6\ x_1 + 5\ x_2 + 4\ x_3 \leq 9$



On arrive à 2. Le choix $x_1 = 0$ fait réévaluer. On trouve l'optimum pour $x_2 = 9/5$, soit une évaluation de 14,4 non exacte donc on continue.

On arrive à 3. L'évaluation donne $x_3 = 9/4$ donc évaluation de 11,25, on continue et on trouve trois solutions d'utilités 0, 5 et 10.

On retourne en 2, puis 4. Avec les choix $x_1 = 0$ et $x_2 = 1$, l'optimum est $x_3 = 1$, solution exacte donnant l'utilité 13. On actualise donc la borne à 13.

On retourne en 1, puis en 5. L'optimum est ici $x_2 = 3/5$, soit une évaluation de 14,8. On continue vers 6, où l'optimum est $x_3 = 3/4$, donc évaluation de 13,25. On peut s'arrêter car on n'aura aucune amélioration possible de la borne 13.

Dans le cas d'un circuit hamiltonien, il faut définir la fonction d'évaluation. La résolution exacte rationnelle du problème linéaire serait en effet trop longue.

Il s'agit de minorer un poids. L'évaluation doit donc être un minorant du poids d'un cycle hamiltonien. On choisit un sommet, puis on cherche un arbre recouvrant de poids minimum pour le graphe privé de ce sommet. On y ajoute la somme des poids des deux arêtes de poids minimum qui arrivent à ce sommet. Ceci fournit un minorant (évaluation).

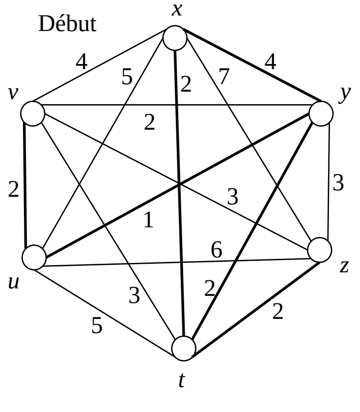
On va ensuite faire des branchements selon des arêtes à exclure (=leur poids devient infini). On réévalue en recalculant un arbre de poids minimum qu'on complète par les arêtes de poids minimum issues du premier sommet. Si on obtient ainsi un cycle hamiltonien de poids inférieur, on actualise la borne, et ainsi de suite.

Exemple $G = K_6$

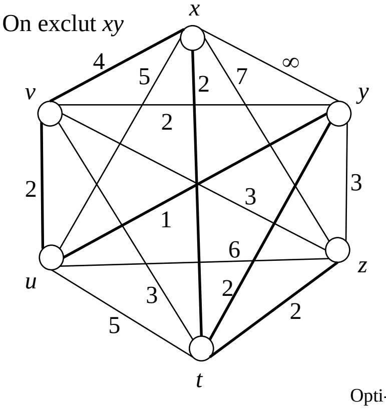
On part de *x*. Un ARPM de $G - x$ est indiqué en gras. On y ajoute les deux arêtes issus de *x* de poids minimum. Le poids total est 13, qui fournit l'évaluation de la racine. Pour la borne, on part d'un cycle hamiltonien. Par exemple *xyztuv* de poids 20.

Le sommet y est sur le graphe recouvrant de degré 3. On sait que dans une solution, au moins l'une des arêtes en gras incidentes à y doit être exclue. On branche en excluant successivement chacune de ces arêtes (en passant son poids à ∞). On réévalue en prenant un ARPM du graphe ainsi modifié moins *x* et en ajoutant les arêtes de poids minimum issues de *x*. Evaluation 13, non exacte.

Début

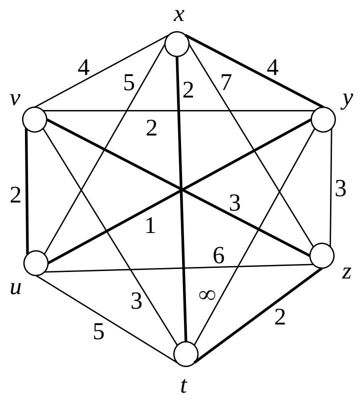


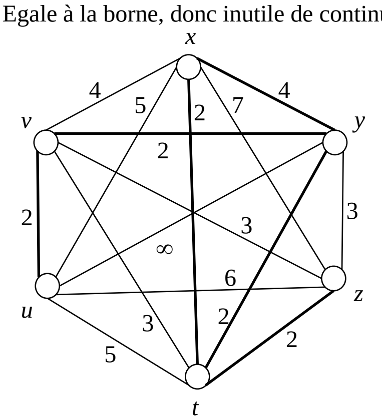
On exclut *xy*

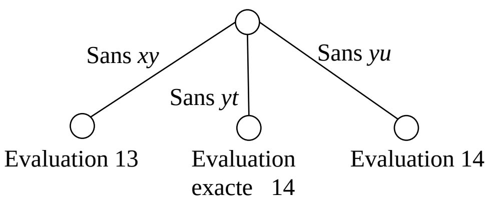


On exclut *yt*. Evaluation exacte 14 (cycle hamiltonien). On actualise la borne.

On exclut *yu*. Evaluation non exacte 14. Egale à la borne, donc inutile de continuer.



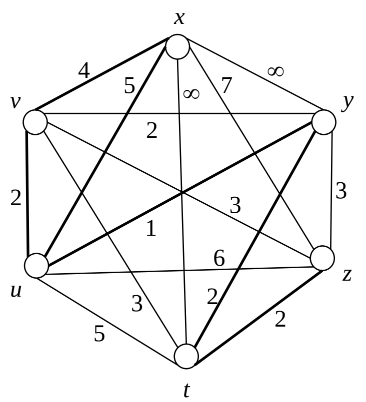


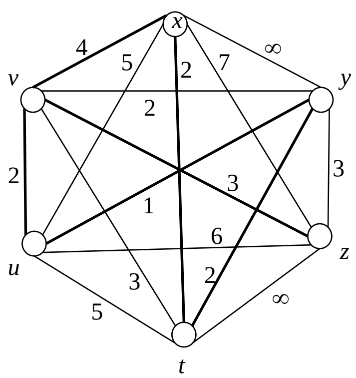


Seul le premier branchement conduit à une évaluation inférieure à la borne. Le premier sommet de degré 3 sur le GRPM est alors t. On branche en supprimant des arêtes de t.

On exclut *xy* et *xt*. Nouvelle évaluation 16. Inutile de continuer.

Il est inutile d'essayer d'exclure *yt*. On a déjà envisagé d'exclure cette arête sans gain. On exclut alors *zt*. L'évaluation vaut 14, donc inutile de continuer.





Conclusion : cycle hamiltonien de poids 14.

4.4 La programmation dynamique

On introduit un paramètre supplémentaire tel que le problème à résoudre corresponde à la dernière valeur du paramètre. Ce paramètre est choisi de façon que la résolution du problème pour une valeur quelconque ne dépende que de la solution pour les valeurs plus petites.

Exemple : le problème du sac à dos

P : Maximiser $\sum_i u_i x_i$ sous les $n + 1$ contraintes $x_i \geq 0$ et $\sum_i v_i x_i \leq V$.
On choisit le paramètre k et les problèmes
$P(k, v)$: Maximiser $\sum_i u_i x_i$ sous les $k + 1$ contraintes $x_i \geq 0$ et $\sum_i v_i x_i \leq v$, où les sommes ne portent que sur les indices de 1 à k.

$P(1, v)$ est facile à résoudre et pour $P(k, v)$, on essaie les diverses valeurs possibles de x_k, qui à chaque fois ramènent à un problème de paramètre $k - 1$. Plus précisément, si $z(k, v)$ est le maximum pour $P(k, v)$, $z(k, v) = \max\{ z(k - 1, v - v_k x_k) + u_k x_k \}$, pris sur les valeurs possibles de x_k.

Maximiser $10 x_1 + 8 x_2 + 5 x_3$ avec les contraintes de positivité (sous-entendues) et la contrainte $6 x_1 + 5 x_2 + 4 x_3 \leq 9$,
On choisit $k = 1$, 2 ou 3 et on s'intéresse seulement aux k premières variables.
v peut prendre toutes les valeurs de 0 à 9 (en fait de 4 à 9).

Pour $k = 1$, on doit maximiser $10 x_1$ avec $6 x_1 \leq v$.
Pour $k = 2$, on a $z(2, v) = \max\{ z(1, v - 5 x_2) + 8 x_2 \}$, avec $v - 5 x_2 \geq 0$.
Pour $k = 3$, on a $z(3, v) = \max\{ z(2, v - 4 x_3) + 5 x_3 \}$, avec $v - 4 x_3 \geq 0$

	v =	0	1	2	3	4	5	6	7	8	9
k =	1	0	0	0	0	0	0	10	10	10	10
	2	0	0	0	0	0	8	10	10	10	10
	3	0	0	0	0	5	8	10	10	10	13

On retrouve le maximum 13, correspondant à (0, 1, 1)

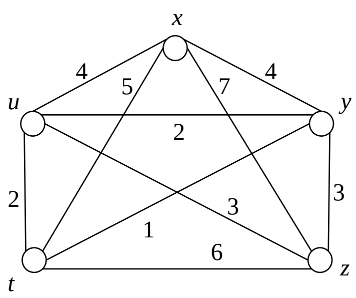
Exemple : le voyageur de commerce

L'idée est de partir d'un sommet *x* et de construire tous les sous-ensembles *S* contenant ce sommet et, pour chacun d'entre eux et pour chaque sommet *u* en-dehors de *S*, de chercher un chemin hamiltonien allant de *x* à *u*, en n'utilisant que des sommets intermédiaires dans *S*. On retient le plus petit poids de ces chemins. On fait de proche en proche le calcul pour tous les sous-ensembles. Un cycle hamiltonien de poids minimum est un tel chemin jusqu'à un sommet *t*, suivi de l'arête *tx*. On cherche donc le minimum des poids des chemins hamiltoniens de *x* à *u* (qu'on a trouvés à la dernière étape) plus le poids de *ux*.

L'étape d'actualisation ressemble un peu à celle de Prim. Mais on raisonne sur des ensembles de sommets...

Exemple

	x	*xy*	*xz*	*xt*	*xu*	*xyz*	*xyt*	*xyu*	*xzt*	*xzu*	*xtu*	*xyzt*	*xyzu*	*xytu*	*xztu*
y	4	x	10	6	6	x	x	x	14	12	7	x	x	x	13
z	7	7	x	11	7	x	9	9	x	x	10	x	x	10	x
t	5	5	13	x	6	11	x	7	x	12	x	x	12	x	x
u	4	6	10	7	x	10	7	x	14	x	x	12	x	x	x



Par exemple, le 13 en dernière colonne, ligne *y* est la valeur minimale entre $T(xzt, u) + p(u, y)$, $T(xzu, t) + p(t, y)$ et $T(xtu, z) + p(z, y)$.

La longueur du cycle hamiltonien de plus faible poids est donc le minimum entre $T(xyzt, u) + p(u, x)$, $T(xyzu, t) + p(t, x)$, $T(xytu, z) + p(z, x)$ et $T(xztu, y) + p(y, x)$.

Dans l'exemple, les valeurs sont 16, 17, 17, 17. Le minimum est donc 16.

En gardant trace de la façon dont sont calculés les nombres du tableau T, on peut reconstituer le cycle hamiltonien. Ici, c'est *xtyzux*.

Au point de vue de la complexité, pour le cycle hamiltonien on a $O(n\ 2^n)$, ce qui est mieux que la force brutale. Il y a en effet $(n - 1)!$ permutations circulaires des n sommets. Les regarder toutes conduit à une complexité $O(n^n\ e^{-n} \sqrt{n})$ d'après la formule de Stirling.

Pour le sac à dos, la solution semble polynomiale. En fait, elle est polynomiale en les *paramètres entiers*. Mais chaque paramètre est une fonction exponentielle de la place qu'il occupe en mémoire.
L'algorithme est donc bien exponentiel en la *taille* des données.
Un tel algorithme est appelé *pseudo-polynomial*.