INF4705 — Analyse et conception d'algorithmes
CHAPITRE 9 : ALGORITHMES D'APPROXIMATION

- Pour certains problèmes, on ne connait pas d'algorithme d'approximation (p.ex. coloriage de graphe).
- Pour d'autres, il existe des algorithmes d'approximation donnant une solution à au plus une certaine distance fixe de la meilleure (p.ex. voyageur de commerce métrique).
- Pour d'autres encore, on peut concevoir un algorithme d'approximation donnant une solution de qualité aussi proche de l'optimum qu'on le désire (p.ex. sac à dos).

9.1 Classification des algorithmes d'approximation

Soient c et ϵ des constantes positives et supposons que la valeur des solutions à notre problème est également positive.

Définition 1 algorithme d'approximation c-absolue. *Un* algorithme d'approximation c-absolue *calcule pour un problème d'optimisation une solution de valeur V dont l'erreur absolue par rapport à la valeur $V^\star$ d'une solution optimale est au plus c :*

$$V^\star - c \leq V \leq V^\star \text{ si on maximise}$$

$$V^\star \leq V \leq V^\star + c \text{ si on minimise}$$

Définition 2 algorithme d'approximation ϵ-relative. *Un* algorithme d'approximation ϵ-relative *calcule pour un problème d'optimisation une solution de valeur V dont l'erreur relative par rapport à la valeur $V^\star$ d'une solution optimale est au plus ϵ :*

$$(1-\epsilon)V^\star \leq V \leq V^\star \text{ si on maximise}$$

$$V^\star \leq V \leq (1+\epsilon)V^\star \text{ si on minimise}$$

Définition 3 algorithme d'approximation mixte. *Un* algorithme d'approximation mixte *calcule pour un problème d'optimisation une solution de valeur V dont l'erreur par rapport à la valeur $V^\star$ d'une solution optimale s'exprime à la fois de manière absolue et relative :*

$$(1-\epsilon)V^\star - c \leq V \leq V^\star \text{ si on maximise}$$

$$V^\star \leq V \leq (1+\epsilon)V^\star + c \text{ si on minimise}$$

Des valeurs de zéro pour c et ϵ correspondent à un algorithme exact. Plus c et ϵ se rapprochent de zéro, meilleures sont les approximations.

9.2 Exemples d'algorithmes d'approximation

Voici quelques problèmes d'optimisation, certains nous étant déjà familiers, pour lesquels on connaît des algorithmes d'approximation et parfois même de plusieurs types.

9.2.1 Mise en boîte ("Bin Packing")

Étant donné n objets, chacun de poids w_i, et des boîtes de capacité W ...

pour un nombre donné de boîtes, k, quel est le plus grand nombre d'objets qu'on puisse y mettre ?

L'algorithme vorace choisissant les objets en ordre croissant de poids garantit une approximation $(k-1)$-absolue (décrit et démontré en classe).

quel est le plus petit nombre de boîtes nécessaires pour y mettre tous les n objets ?

L'algorithme vorace "First-Fit-Decreasing" (décrit en classe) est d'approximation mixte :

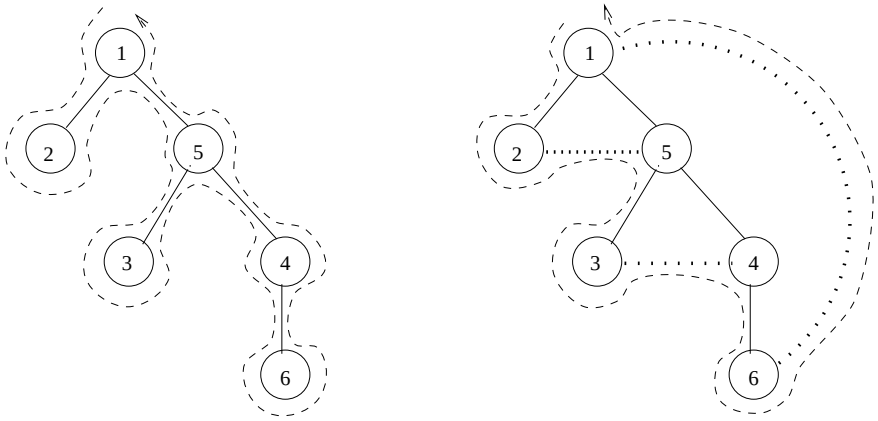
$$V^\star \leq V \leq \frac{11}{9}V^\star + \frac{6}{9}$$

9.2.2 Voyageur de commerce (métrique)

Soit G un graphe complet sur n sommets et dont chaque arête (u, v) a une longueur $\ell((u, v)) \geq 0$. Soit $\tau^\star$ la tournée la plus courte, de longueur $\ell(\tau^\star)$ (par extension naturelle de la définition de ℓ). On remarque que si on retire une arête de $\tau^\star$, on obtient un chemin (Hamiltonien) reliant tous les sommets de G et de longueur au plus $\ell(\tau^\star)$. La première remarque clef est que ce chemin est un arbre sous-tendant particulier, appelons-le a. Conséquemment, un arbre sous-tendant minimum de G, appelons-le $a^\star$, est au plus aussi long que ce chemin, qui lui-même est au plus aussi long que $\tau^\star$:

$$\ell(a^\star) \leq \ell(a) \leq \ell(\tau^\star).$$

Considérons une promenade le long de $a^\star$ (comme pour déterminer les parcours pré-ordre, post-ordre et en-ordre) : puisque chaque arête est empruntée deux fois, la longueur de la promenade est de $2\,\ell(a^\star)$.



La figure de gauche illustre le chemin obtenu pour un petit exemple, $\langle 1, 2, 1, 5, 3, 5, 4, 6, 4, 5, 1\rangle$. Ce chemin n'est pas une tournée légale puisqu'il passe plusieurs fois par le même sommet. Puisque G est complet, on pourrait par contre éliminer les visites multiples d'un sommet en prenant des "raccourcis", comme illustré à la figure de droite, nous donnant une tournée $\tilde{\tau} = \langle 1, 2, 5, 3, 4, 6, 1\rangle$.

Mais s'agit-il vraiment de raccourcis ? Par exemple, le chemin direct de 2 à 5 est-il nécessairement plus court que celui passant par 1 ? Non.

Considérons donc le problème du voyageur de commerce *métrique*, c'est-à-dire où les longueurs respectent l'inégalité du triangle :

$$\ell((u, v)) + \ell((v, w)) \geq \ell((u, w)).$$

La seconde remarque clef est que dans ce cas, qui n'est pas si restrictif puisqu'il est vérifié entre autres par la distance Euclidienne, il s'agit bel et bien de raccourcis et nous obtenons

$$\ell(\tilde{\tau}) \leq 2\,\ell(a^\star) \leq 2\,\ell(\tau^\star).$$

Autrement dit,

$$\ell(\tau^\star) \leq \ell(\tilde{\tau}) \leq (1+1)\,\ell(\tau^\star).$$

Nous venons donc de décrire un algorithme d'approximation 1-relatif correspondant au voyageur de commerce métrique.

En fait, le résultat final de l'algorithme devrait vous sembler familier : il s'agit simplement d´un parcours en pré-ordre d'un arbre sous-tendant minimum de G. Son temps de calcul est donc dans $\mathcal{O}(n^2)$.

Voici une troisième remarque : la solution que nous obtiendrons dépend du sommet de l'arbre sous-tendant minimum que nous choisirons comme racine. En effet, la promenade décrit essentiellement le même chemin pour tout choix de racine mais à un décalage près. Ce décalage signifie que les visites multiples éliminées ne seront pas nécessairement les mêmes et donc que la tournée résultante sera composée d'arêtes différentes, menant à des solutions de coût différent. Par exemple, le choix du sommet 4 comme racine correspond à la promenade $\langle 4,5,1,2,1,5,3,5,4,6\rangle$, qu'on obtient en décalant $\langle 1,2,1,5,3,5,4,6,4,5\rangle$ de deux sommets (le retour final à la racine a été omis afin de mieux visualiser la similitude). L'élimination des visites multiples dans le premier chemin mènera à $\langle 4,5,1,2,3,6,4\rangle$, une tournée différente.

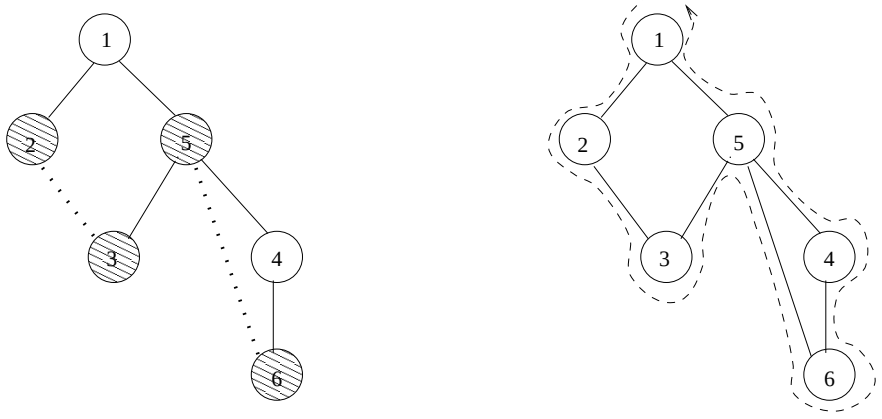
Mais il y a encore mieux. D'abord quelques définitions :

- Une tournée *Eulérienne* dans un graphe passe par chaque *arête* exactement une fois (et possiblement plusieurs fois par certains sommets). Un graphe n'admet pas nécessairement une tournée Eulérienne.
- Un graphe est dit *Eulérien* si chacun de ses sommets est de degré pair. On l'appelle Eulérien parce qu'on peut démontrer qu'un graphe admet une tournée Eulérienne si et seulement si chacun de ses sommets est de degré pair.

En somme, l'algorithme précédent

1. construit un arbre sous-tendant minimum
2. en fait un graphe Eulérien en doublant chaque arête
3. détermine une tournée Eulérienne
4. la transforme en tournée (sur les sommets) en prenant des raccourcis

Tout en conservant cette procédure, il existe une façon plus économe de transformer l'arbre en un graphe Eulérien (étape 2). Considérons l'ensemble des sommets de degré impair dans l'arbre, I. Il y a nécessairement un nombre pair de tels sommets (Pourquoi ?). On pourrait donc les regrouper deux à deux et ajouter les arêtes correspondantes, ce qui rendrait le graphe Eulérien puisque leur degré deviendrait pair. Un *couplage de coût minimum* donne la façon la plus économe d'accomplir cela et on connaît des algorithmes efficaces pour le calculer.



Pour l'exemple ci-haut, on obtiendrait la tournée $\bar{\tau} = \langle 1,2,3,5,6,4,1\rangle$. Quelle garantie peut-on offrir quant à la longueur de $\bar{\tau}$? Une dernière remarque clef : les arêtes du couplage ont une longueur totale qui ne peut dépasser la moitié de celle de $\tau^\star$, la tournée optimale. En effet, si on retire de $\tau^\star$ les sommets ne faisant pas partie de I, on obtient, en introduisant des raccourcis, une tournée de I, appelons-la τ_I, de longueur

inférieure à $\ell(\tau^\star)$. En choisissant en alternance des arêtes de τ_I, on dérive deux couplages disjoints, dont le plus court des deux n'excède pas la moitié de $\ell(\tau_I)$. Or le couplage de coût minimum que nous avions calculé est au moins aussi court, par définition.

Il s'ensuit que

$$\ell(\tau^\star) \leq \ell(\bar{\tau}) \leq (1 + \frac{1}{2})\, \ell(\tau^\star).$$

Voilà donc un algorithme d'approximation $\frac{1}{2}$-relatif.

Pour clore le sujet, soulignons qu'il ne peut exister d'algorithme d´approximation c-absolue efficace pour résoudre le problème du voyageur de commerce métrique (à moins qu'il en existe également un pour le problème originel). Si on permet que l'inégalité du triangle ne soit pas respectée (problème du voyageur de commerce) alors c'est vrai autant pour les approximations ϵ-relative que c-absolue.

9.2.3 Sac à dos

L'algorithme vorace que nous connaissons peut être arbitrairement mauvais.
Voici une façon simple de l'améliorer qui retourne une solution valant au moins la moitié de la valeur optimale (donc d'approximation 1/2-relative) :

retourner $V = \max(\text{algorithme vorace}(w, v, W),\ \max\{v_i : 1 \leq i \leq n\})$.

Supposons les objets classés en ordre décroissant de densité de valeur et soit ℓ le premier objet à devoir être exclus du sac ($\sum_{i=1}^{\ell} w_i > W$).

$$\begin{aligned} V &\geq (\text{algorithme vorace}(w, v, W) + \max\{v_i : 1 \leq i \leq n\})/2 \\ &\geq (\sum_{i=1}^{\ell-1} v_i + v_\ell)/2 \\ &= (\sum_{i=1}^{\ell} v_i)/2 \\ &\geq V^\star/2 \end{aligned}$$

Nous avons donc :

$$(1 - 1/2)V^\star \leq V \leq V^\star$$

Schéma d'approximation

Un *schéma d'approximation* est un algorithme auquel on donne, en plus de l'exemplaire, l'ϵ désiré pour l'algorithme d'approximation ϵ-relative à résoudre.
On dira que le schéma d'approximation est *pleinement polynomial* s'il prend un temps dans $\mathcal{O}(p(n, 1/\epsilon))$ en pire cas, où n est la taille de l'exemplaire et p est un polynôme fixe en deux variables.

idée : pour le problème du sac à dos, combiner l'algorithme vorace amélioré et l'algorithme de programmation dynamique.

Algorithme de programmation dynamique (version duale) :

Définissons cette fois un tableau $U[1 \ldots n, 0 \ldots M]$ où M est une borne supérieure sur la valeur totale du sac à dos et $U[i, j]$ est le poids minimum d'un contenu du sac de valeur totale j et n'utilisant que les objets $1 \ldots i$.
Calcul de $U[i, j]$: nous pouvons soit utiliser l'objet i ($w_i + U[i-1, j-v_i]$) ou ne pas l'utiliser ($U[i-1, j]$).

$$U[i, j] = \min(w_i + U[i-1, j-v_i],\ U[i-1, j])$$

- Où est notre réponse ? À la case $U[n, j]$ pour le plus grand j tel que $U[n, j] \leq W$.
- Valeurs frontière ? $U[0, 0] = 0$; $U[i, j] = \infty, \forall i = 0, j > 0$ et $\forall j < 0$.

– Dans quel ordre remplir notre tableau ? Comme pour l'autre version.

analyse :
On doit remplir un tableau $n \times (M+1)$ et récupérer la réponse en parcourant une ligne du tableau, ce qui donne un temps dans $\Theta(nM)$.

2 questions :

Q1 : Pourquoi utiliser cette nouvelle version ?

Q2 : Et de toute façon, où trouver M ?

R1 : Son temps d'exécution dépend non plus de W mais de M et nous pouvons borner la taille de M par un polynôme en n et $1/\epsilon$ en autant qu'on se contente d'une solution approximative.

Soient ϵ l'erreur relative tolérée et k une constante entière à déterminer.
Définissons une modification de l'exemplaire identique en tous points sauf pour la valeur des objets : $v'_i = \lfloor v_i/k \rfloor$.
Soient $X^\star$ une solution optimale de valeur $V^\star$ pour l'exemplaire originel et X' une solution optimale pour l'exemplaire modifié ; soit V' la valeur de X' pour l'exemplaire *originel*.

Remarques :

1. $\Sigma_{i \in X'} v'_i \geq \Sigma_{i \in X^\star} v'_i$
2. $v_i \geq k v'_i > v_i - k$

On a donc

$$\begin{aligned} V' &= \Sigma_{i \in X'} v_i \\ &\geq k \Sigma_{i \in X'} v'_i, \text{ par remarque 2} \\ &\geq k \Sigma_{i \in X^\star} v'_i, \text{ par remarque 1} \\ &> \Sigma_{i \in X^\star} (v_i - k), \text{ par remarque 2} \\ &= \Sigma_{i \in X^\star} v_i - \Sigma_{i \in X^\star} k \\ &\geq V^\star - kn \end{aligned}$$

En choisissant $k \leq \epsilon V^\star / n$ on obtient :

$$V' > V^\star - \epsilon V^\star = (1 - \epsilon) V^\star$$

On aurait donc un algorithme d'approximation ϵ-relative.
Mais on a besoin de $V^\star$? ! ?

R2 : L'algorithme vorace amélioré nous retourne une solution de valeur $\geq 1/2 V^\star$

Utilisation des morceaux :

- La valeur A de la solution de l'algorithme vorace amélioré sert :
  - de sous-estimation de $V^\star$ pour le calcul de k ;
  - de sur-estimation de $1/2V^\star$ pour le calcul de M.
- L'algorithme de programmation dynamique dual sert :
  - à calculer une solution exacte pour l'exemplaire originel, ou plus souvent
  - à calculer une solution exacte pour l'exemplaire modifié, qui est aussi une solution approximative ϵ-relative pour l'exemplaire originel.

Voici donc le schéma d'approximation :

1. $A \leftarrow$ valeur de la solution de `algo_vorace_amélioré`(w, v, W) ;
2. **si** $\frac{\epsilon A}{n} < 2$ **alors**
   (a) $S \leftarrow$ `algo_prog_dyn_dual`$(w, v, W, M = 2A)$;
   (b) **retourner** S ; {qui sera une solution exacte}
3. $k \leftarrow \lfloor \frac{\epsilon A}{n} \rfloor$; $\{ \leq \frac{\epsilon V^\star}{n} \}$
4. **pour** $i \leftarrow 1$ **à** n **faire** $v'_i \leftarrow \lfloor \frac{v_i}{k} \rfloor$; {construire l'exemplaire modifié}
5. $S \leftarrow$ `algo_prog_dyn_dual`$(w, v', W, M = \lfloor \frac{2A}{k} \rfloor)$;
6. **retourner** S ; {qui sera une solution approximative ϵ-relative}

Analyse du temps d'exécution :

1. $\mathcal{O}(n \log n)$

2. $\Theta(nA)$ mais $A < 2n/\epsilon$ alors $\mathcal{O}(n^2/\epsilon)$

3. $\Theta(1)$

4. $\Theta(n)$

5. $\Theta(nA/k) = \Theta(nAn/\epsilon A) = \Theta(n^2/\epsilon)$ puisque $\frac{\epsilon A}{n} - 1 \leq k \leq \frac{\epsilon A}{n}$

Donc l'algorithme prend un temps dans $\mathcal{O}(n^2/\epsilon)$, ce qui est pleinement polynomial.