Algorithmes d'approximation

François Schwarzentruber

10 mars 2021

Certains problèmes d'optimisation dont les problèmes de décision sont NP-complets admettent des algorithmes en temps polynomial qui calcule une solution suffisamment satisfaisante.

Définition 1 Un algorithme d'approximation est un algorithme qui calcule une solution quasi optimale.

Définition 2 Étant donné un problème de minimisation, un algorithme d'approximation est de ratio ρ si, sur toute entrée de taille n,

$$\frac{\text{cout(solution calculée)}}{\text{cout(solution optimale)}} \leq \rho(n)$$

Définition 3 Étant donné un problème de maximisation, idem mais

$$\frac{\text{cout(solution optimale)}}{\text{cout(solution calculée)}} \leq \rho(n)$$

1 Couverture de sommets

Application 4 Placer le minimum de gardiens pour surveiller tous les couloirs (= arêtes).

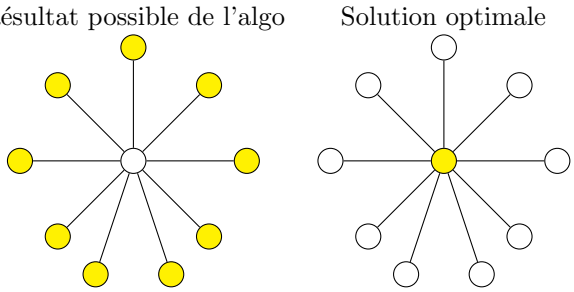
Définition 5 (couverture de sommets) Soit un graphe $G = (V, E)$. Une couverture de sommet est un ensemble $C \subseteq V$ tel que pour tout $(u, v) \in E$, $u \in C$ ou $v \in C$.

Définition 6 (problème d'optimisation) **VERTEX COVER**

entrée : un graphe $G = (V, E)$ non orienté
sortie : une couverture de sommets de cardinal minimal.

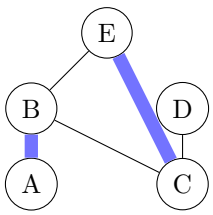
1.1 Algorithme naïf

Algo naïf
Ajouter des sommets
jusqu'à obtenir une couverture



1.2 Couplage

Définition 7 (couplage) Soit $G = (V, E)$ un graphe non orienté. Un ensemble $M \subseteq E$ est un couplage si pour tout sommet $u \in V$, il existe au plus une arête $(a, b) \in M$ avec $a = u$ ou $b = u$.



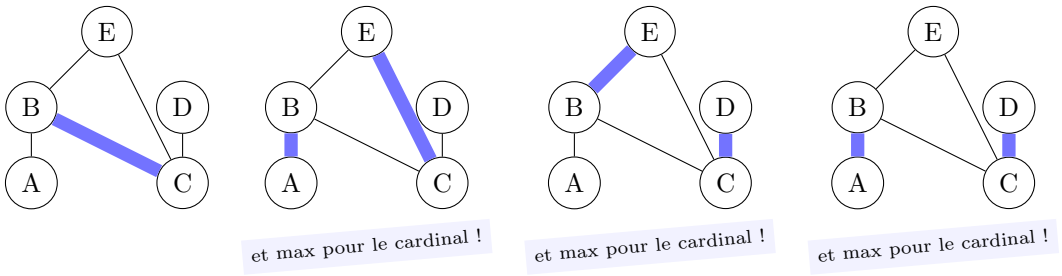
Proposition 8 $|M| \leq |C|$ pour tout couplage M, pour toute couverture C.

Démonstration. Chaque couloir de M est vu par un gardien (forcément différent). Plus formellement, soit f une fonction de M dans C qui associe à toute arête $e \in M$ une des extrémités de e, qui est dans C. Comme M est un couplage, on conclut car f est injective. ■

1.3 Couplage maximal pour l'inclusion (couplage maximal)

Définition 9 Un couplage maximal pour l'inclusion est un couplage M tel que tout pour tout $M' \subseteq E$, $M \subsetneq M'$ implique M' n'est pas un couplage.

Exemple 10 (couplages maximaux pour l'inclusion)

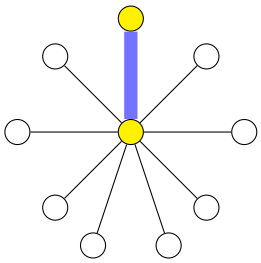


Proposition 11 Un couplage maximum pour le cardinal est maximal pour l'inclusion.

Algo pour calculer un couplage maximal : ajouter des arêtes jusqu'à ne plus pouvoir

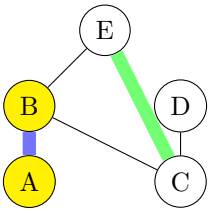
1.4 Algorithme d'approximation pour VERTEX COVER

Algo d'approximation : $M :=$ un couplage max pour l'inclusion **renvoyer** $C :=$ l'ensemble des extrémités des arêtes de M



Proposition 12 C est une couverture de sommets.

Démonstration. Par l'absurde, supposons que C ne soit pas une couverture de sommets, i.e. il existe une arête e qui ne touche aucun sommet de C.



Mais alors $M \cup \{e\}$ est aussi un couplage, contredisant la maximalité pour l'inclusion de M. ■

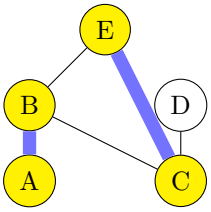
Proposition 13 Le ratio d'approximation est 2.

Démonstration.

$$\frac{\dfrac{C = \bigsqcup_{e \in M} extremites(e)}{|C| = 2|M|} \qquad \dfrac{\text{Proposition 8}}{|M| \leq |C^*|}}{|C| \leq 2|C^*|}$$

où C^* est une couverture minimale. ■

Exemple 14 Le ratio 2 est atteint par cet algorithme, comme par exemple sur les graphes ci-dessous :



1.5 La dualité est affaiblie

Définition 15 (couplage maximum pour le cardinal) Un couplage M est maximum si $|M|$ maximal.

Proposition 16 Dans un graphe biparti, cardinal min d'une couverture = cardinal max d'un couplage.

Proposition 17 Dans un graphe qcq, card couplage maximum $\leq$ card min couverture $\leq 2\times$ card couplage maximum.

Aller plus loin

— algorithme de ratio $2 - \Theta\left(\frac{1}{\sqrt{\log |V|}}\right)$ [Kar09].
— Si $P \neq NP$, alors pas d'algo polynomial avec un facteur ≤ 1.3606. [DS05]

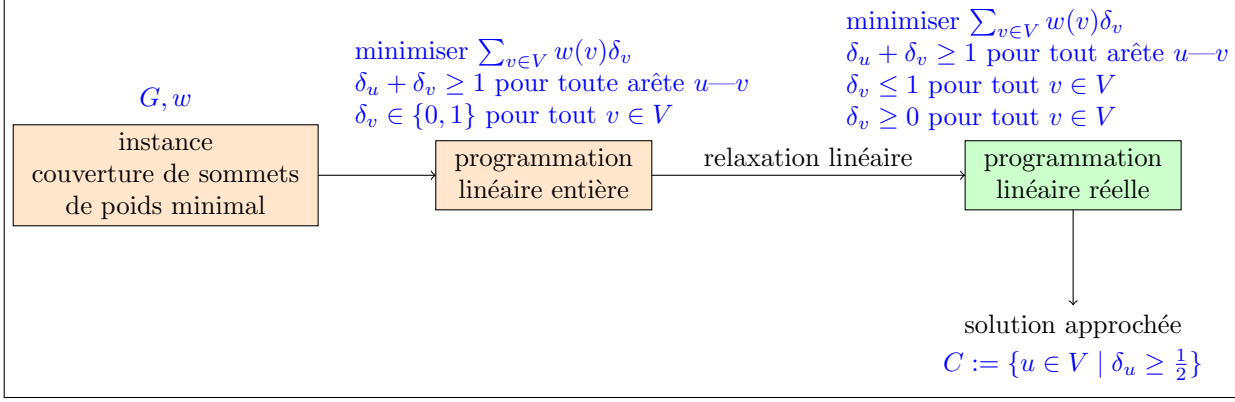
2 Couverture de sommets pondérée et relaxation linéaire

Généralisons le problème de la couverture de sommets avec des sommets pondérés.

Définition 18 **problème de la couverture de sommets pondérée**

entrée : Un graphe non orienté $G = (V, E)$, des poids positifs $w(u)$ à chaque sommet $u \in V$;
sortie : une couverture de sommets $C^* \subseteq V$ tel que son poids $\sum_{v \in C^*} w(v)$ soit minimal.

Algorithme d'approximation



Exercice 19 Donner un exemple pertinent.

Proposition 20 C est une couverture de sommets.

Démonstration. La contrainte $\delta_u + \delta_v \geq 1$ pour tout arête u—v, force que $\delta_u \geq \frac{1}{2}$ ou $\delta_v \geq \frac{1}{2}$. ■

Théorème 21 L'algorithme est 2-approximant.

Démonstration.

$$\dfrac{\dfrac{\text{Définition du poids}}{w(C) = \sum_{u\in C} w(u)} \quad \dfrac{\dfrac{\text{Définition de } C}{\sum_{u\in C} w(u) \leq \sum_{u\in V} 2w(u)\delta_u}}{\sum_{u\in C} w(u) \leq 2\sum_{u\in V} w(u)\delta_u} \quad \dfrac{\text{fonction caractéristique de } C^* \text{ solution du programme linéaire réel}}{\sum_{u\in V} w(u)\delta_u \leq w(C^*)}}{w(C) \leq 2w(C^*)}$$

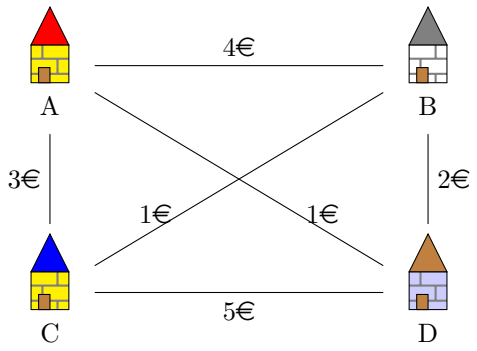
où C^* est une couverture de sommets de poids minimal. ■

3 Inapproximabilité du voyageur de commerce (TSP)

Définition 22 (problème d'optimisation) **TSP**

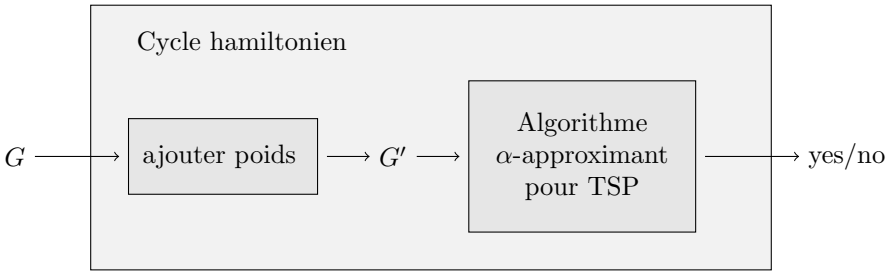
entrée : un graphe non orienté complet pondéré $G = (V, d)$ où $d : V \times V \to \mathbb{N}$ avec $d(i, j) = d(j, i)$
sortie : un tour dans G de poids minimal

Exemple 23



Théorème 24 Si P $\neq$ NP, alors TSP n'admet pas d'algo d'approximation en temps poly de ratio constant α.

Démonstration. Sans perte de généralité, α est un entier. Par l'absurde, supposons qu'un tel algo existe. Construisons un algorithme polynomial décidant cycle hamiltonien.



Ajout de poids. Soit $G=(V, E)$ un graphe non orienté. On crée le graphe complet pondéré $G' = (V, d)$ par :
— $d(u, v) = 1$ si $(u, v) \in E$;
— et $d(u, v) = \alpha|S| + 1$ si $(u, v) \notin E$.

Exercice 25

1. Montrer que G admet un cycle hamiltonien ssi G' admet un tour optimal de coût $|S|$.
2. Montrer que si G' n'admet pas de tour de coût $|S|$, alors un tour optimal est de coût $> \alpha|S|$.
3. Conclure.

■

4 Voyageur de commerce avec inégalité triangulaire

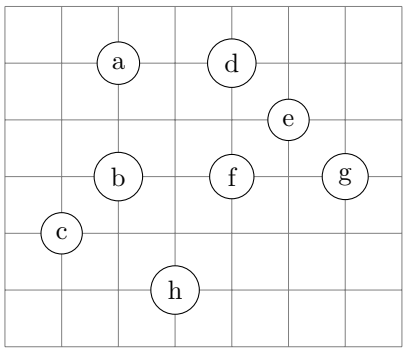
Applications : Transport en bus, faire des trous dans un morceau de bois.

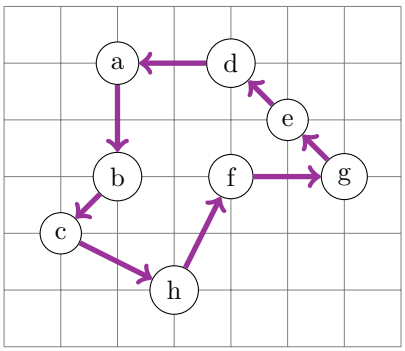
Définition 26 Un graphe non orienté complet pondéré $G = (V, d)$ vérifie l'inégalité triangulaire si $d(u, w) \leq d(u, v) + d(v, w)$ pour tous les sommets $u, v, w \in V$.

Définition 27 **problème du voyageur de commerce avec inégalité triangulaire**

entrée : un graphe non orienté complet pondéré $G = (V, d)$ avec inégalité triangulaire ;
sortie : un tour de poids minimal.

Exemple 28 (Graphe où les sommets sont des points du plan)



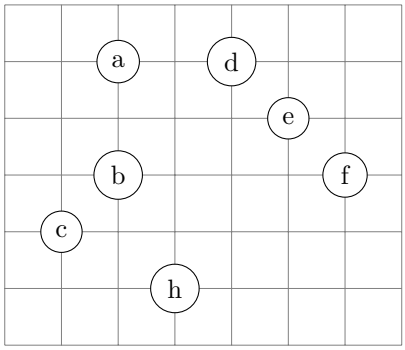


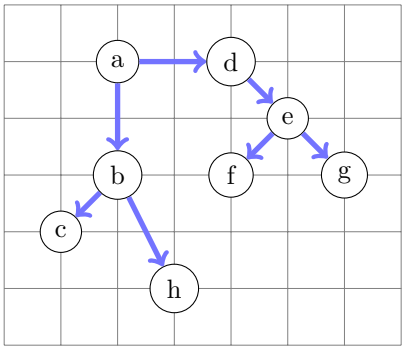
Exercice 29 Montrer que le problème du voyageur de commerce avec inégalité triangulaire est toujours NP-complet.

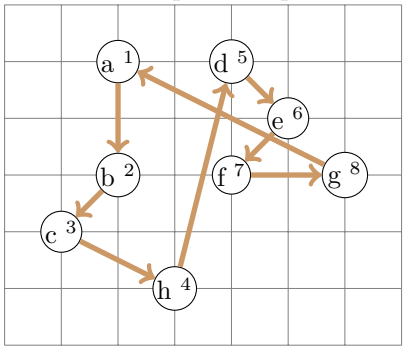
fonction algoApprox(G)
construire T un arbre couvrant minimum de G
réaliser un parcours préfixe de T
renvoyer la tournée correspondant au parcours préfixe

Exemple 30

Instance G | Arbre couvrant min | tour approximant = tournée parcours préfixe

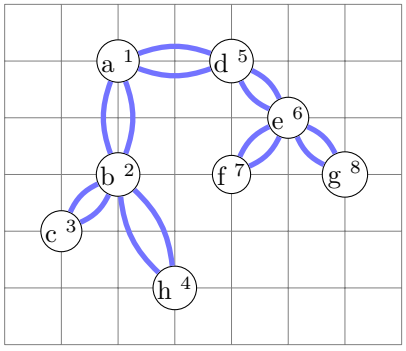




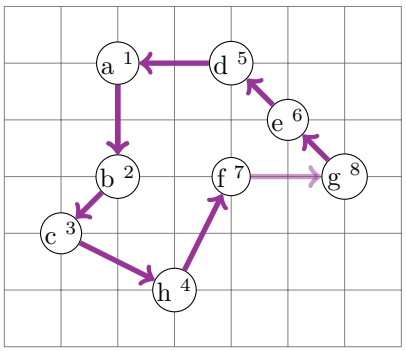


Proposition 31 Poids du tour approximant $\leq 2 \times$ poids d'un tour optimal.

Démonstration.



+ inégalité triangulaire



Tour optimal où on a supprimé une arête = arbre couvrant

$$\frac{\dfrac{}{w(\text{tour approx}) \leq 2w(T)} \qquad \dfrac{}{w(T) \leq w(\text{tour opt})}}{w(\text{tour approx}) \leq 2w(\text{tour opt})}$$

■

Aller plus loin

— Algorithme de Christofides de facteur 3/2 [Vaz04]
— Cas plus restrictif : TSP euclidien où la distance est la distance entre points, comme dans l'exemple 28 qui admet un PTAS, voir plus loin [Aro98]

5 Set cover : approximation avec un ratio $O(\log n)$

Application 32 On a un ensemble de n villes et on veut placer un nombre minimum d'écoles avec deux contraintes : chaque école est dans une ville et il doit y avoir une école à moins de 10km de chaque village.

Définition 33 **SET COVER**

entrée : B un ensemble fini et des sous-ensembles $S_1, \cdots, S_n$ inclus dans B ;
sortie : Un sous-ensemble $J \subseteq \{1, \ldots, n\}$ tel que $\bigcup_{j \in J} S_j = B$ et $|J|$ est minimal (s'il en existe).

Exemple 34 B est un ensemble des n villes et chaque S_i est l'ensemble des villes à moins de 10km de la ville numéro i.

Si on utilise une stratégie gloutonne, on a un algorithme approximatif mais efficace. À chaque fois, on choisit un ensemble S_j avec le maximum d'éléments non couverts.

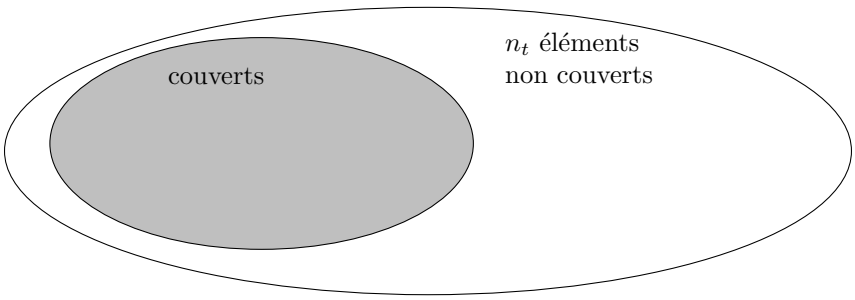
```
fonction approxSetCover(B, S_1, ··· , S_n)
    J := ensemble vide
    C := ∅
    tant que C ≠ B
        choisir j ∈ {1, ..., n} tel que |S_j ∩ B\C| soit maximal.
        si S_j ⊆ C alors
            renvoyer impossible
        C := C ∪ S_j
        ajouter j à J
    renvoyer J
```

Théorème 35 Si B contient n éléments et que la couverture optimale en contient k, alors l'algorithme glouton renvoie au plus $k\lceil \ln(n) \rceil$ sous-ensembles.

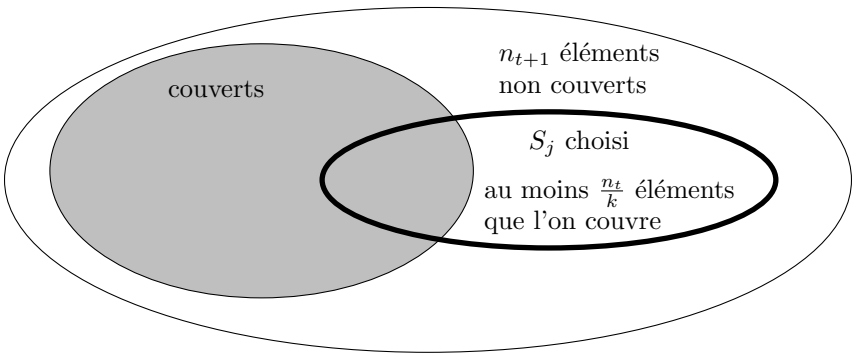
Démonstration. Supposons qu'il y a une couverture. Considérons une couverture optimale $\mathcal{S} = (S_j)_{j \in J}$ avec $|J| = k$. On note n_t le nombre d'éléments non couverts au t-ème tour de boucle. Pour $t = 0$, on a $n_0 = n$.

Lemme 36 Pour tout $t \geq 1$, $n_t \leqslant n\left(1 - \frac{1}{k}\right)^t$.

Démonstration. Plaçons nous au t-ème tour de boucle.



Comme $\mathcal{S}$ couvre les éléments n_t éléments non couverts, il y a un ensemble S_j avec au moins $\frac{n_t}{k}$ éléments non couverts. Ainsi, à l'étape t l'algorithme va choisir un ensemble S_j et couvrir au moins $\frac{n_t}{k}$ éléments supplémentaires.



Ainsi, $n_{t+1} \leqslant n_t - \frac{n_t}{k} = n_t\left(1 - \frac{1}{k}\right)$. Par récurrence sur $t \in \mathbb{N}$, on montre que $n_t \leqslant n\left(1 - \frac{1}{k}\right)^t$.
■

L'inégalité de convexité $1 - x \leq e^{-x}$ pour tout x réel, avec égalité uniquement si $x = 0$, donne $n_t < ne^{-\frac{t}{k}}$. Maintenant, à partir de quelle étape t, tous les éléments sont couverts à coup sûr ? Pour $t \geqslant k\lceil \ln(n) \rceil$, on a $ne^{-\frac{t}{k}} \leq ne^{-\lceil \ln(n) \rceil} < 1$; et donc tous les éléments sont couverts avec $k\lceil \ln(n) \rceil$ ensembles. ■

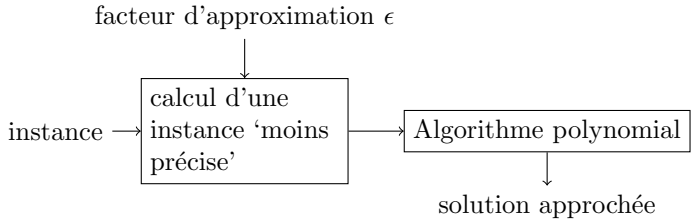
Remarque 37 Si $P \neq NP$, Raz et Safra ont montré qu'on ne faire mieux qu'un ratio de $c \log n$ où $c > 0$ [RS97].

6 Sac à dos : compromis entre précision et temps de calcul

Ref : [TK06][11.8, p. 644], [DPV06][p. 283], [Vaz04][p. 68]

SAC A DOS
entrée : n objets $((w_i, v_i))_{i=1..n}$, un poids W
sortie : un sous-ensemble $J \subseteq \{1, \ldots, n\}$ tel que $\sum_{i\in J} w_i \leq W$ et $\sum_{i\in J} v_i$ maximal.

Compromis entre précision et temps de calcul



6.1 Sous-routine basée sur la programmation dynamique

En ALGO1, nous avons donné un algorithme de programmation dynamique en temps $O(nW)$ où W est le poids total. Ici, nous donnons un algorithme en temps $O(nV^*)$ où V^* est la somme des valeurs.

Définition 38 (sous-problèmes) Pour tout $k \in \{1, \ldots, n\}$, tout $V \in \{1, \ldots, V^*\}$,

$$\begin{aligned} opt(k, V) &= \text{le poids minimal d'un sac contenant des objets de } \{1, \ldots, k\} \text{ avec une valeur totale} \geq V \\ &= \min_{J\subseteq\{1,\ldots,k\} | \sum_{i\in J} v_i \geq V} \sum_{i\in J} w_i. \end{aligned}$$

ou $+\infty$ quand il n'y a pas de $J \subseteq \{1, \ldots, k\} \mid \sum_{i\in J} v_i \geq V$.

Proposition 39 La solution cherchée est la valeur V maximale telle que $opt(n, V) \leq W$.

Exercice 40 Concevoir un algorithme qui calcule une solution à sac à dos en temps $O(nV^*)$.

6.2 Approximation

fonction *sacadosApprox*$((w_i, v_i)_{i=1..n}, W, \epsilon)$
  $v_{max} = \max(v_1, \ldots, v_n)$
  pour $i = 1..n$ **faire**
    $\widetilde{v}_i := \lfloor \frac{n v_i}{\epsilon v_{max}} \rfloor$
  renvoyer *sacados*$((w_i, \widetilde{v}_i)_i, W)$

Proposition 41 *sacadosApprox* est en temps $O(\frac{n^3}{\epsilon})$.

Démonstration.
Les nouvelles valeurs $\widetilde{v}_k$ sont dans $\{0, \ldots, \lfloor \frac{n}{\epsilon} \rfloor\}$. Ainsi $V^* = \Theta(\frac{n^2}{\epsilon})$, et la complexité est en $O(n \times \frac{n^2}{\epsilon})$. ■

Théorème 42 Si S^{ap} est la solution trouvé par l'algorithme d'approximation, alors pour toute solution S^* avec $\sum_{i\in S^*} w_i \leq W$ on a $\sum_{i\in S^{ap}} v_i \geq \sum_{i\in S^*} v_i(1-\epsilon)$.

Démonstration.

$$\begin{aligned} \sum_{i\in S^{ap}} v_i &\geq \sum_{i\in S^{ap}} \frac{\epsilon v_{max} \widetilde{v}_i}{n} && \text{par définition de } \widetilde{v}_i \\ &\geq \sum_{i\in S^*} \widetilde{v}_i \frac{\epsilon v_{max}}{n} && \text{car l'algo est correct par rapport aux valeurs } \widetilde{v}_i \\ &\geq \sum_{i\in S^*} \left(\frac{n}{\epsilon} \frac{v_i}{v_{max}} - 1\right) \frac{\epsilon}{n} v_{max} && \text{par définition de } \widetilde{v}_i \\ &\geq \left(\sum_{i\in S^*} v_i\right) - \left(\frac{n\epsilon v_{max}}{n}\right) \\ &\geq \left(\sum_{i\in S^*} v_i\right) - \epsilon \times v_{max} \\ &\geq \left(\sum_{i\in S^*} v_i\right)(1-\epsilon) && \text{car } v_{max} \leq \left(\sum_{i\in S^*} v_i\right) \end{aligned}$$

■

6.3 PTAS, EPTAS, FPTAS

On donne les définitions pour les problèmes de maximisation. Pour les problèmes de minimisation, remplacer $1-\epsilon$ par $1+\epsilon$.

Définition 43 Un *polynomial time approximation scheme* (PTAS) est un algorithme d'approximation qui calcule une solution de ratio $(1-\epsilon)$, et tel qu'en fixant ϵ, l'algorithme est en temps polynomial en la taille de l'entrée.

Définition 44 Un *fully polynomial time approximation scheme* (FPTAS) est un PTAS, qui s'exécute en temps polynomial en la taille de l'entrée et en $\frac{1}{\epsilon}$.

Remarque 45
- Avec PTAS, on peut avoir une complexité $2^{1/\epsilon}n^{1/\epsilon}$.
- Avec FPTAS, on ne peut pas mais on peut avoir $n^2\frac{1}{\epsilon^3}$.

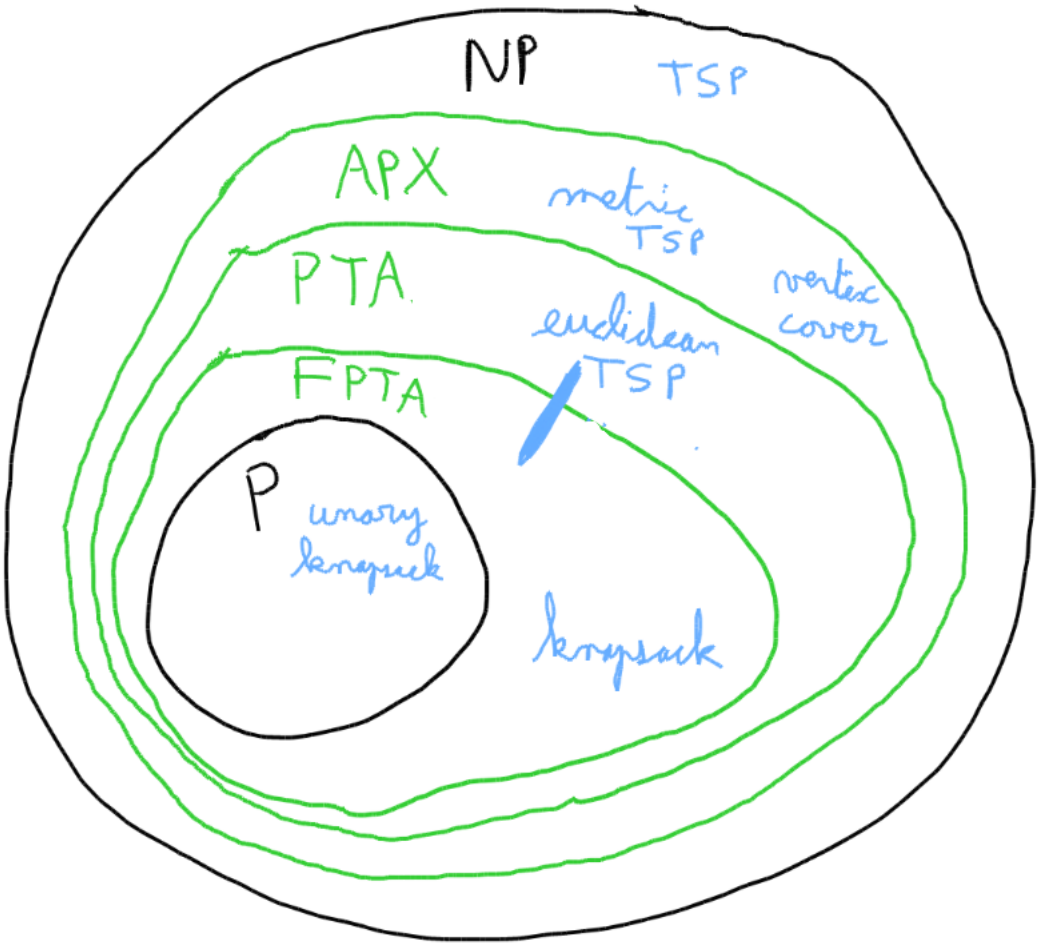
7 Classes de complexité de problèmes d'optimisation

Définition 46 (APX, PTA, FPTA)

APX = problèmes admettant un algo d'approximation poly avec ratio constant

PTA = problèmes admettant un PTAS

FPTA = problème admettant un FPTAS



Références

[Aro98] Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of the ACM (JACM)*, 45(5) :753–782, 1998.

[DPV06] S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani. Algorithms. 2006.

[DS05] I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, pages 439–485, 2005.

[Kar09] G. Karakostas. A better approximation ratio for the vertex cover problem. *ACM Transactions on Algorithms (TALG)*, 5(4) :41, 2009.

[RS97] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 475–484. ACM, 1997.

[TK06] É. Tardos and J. Kleinberg. Algorithm design, 2006.

[Vaz04] V.V. Vazirani. *Approximation algorithms*. springer, 2004.