Algorithmes d'approximation

François Schwarzentruber

10 mars 2021

Définition 1 Étant donné un problème d'optimisation, un algorithme d'approximation calcule une solution quasi optimale.

Définition 2 Un algorithme d'approximation est de ratio ρ si, sur toute entrée de taille n,

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

si problème de minimisation — si problème de maximisation

1 Classification

Définition 3 (PTAS) Un *polynomial time approximation scheme* est un algorithme d'approximation qui prend en entrée une instance I et un nombre ϵ, et qui calcule une solution de ratio $\begin{cases} 1+\epsilon & \text{pour pb de min} \\ 1-\epsilon & \text{pour pb de max} \end{cases}$, et tel qu'en fixant ϵ, l'algorithme est en temps $poly(|I|)$.

Définition 4 (FPTAS) Un *fully polynomial time approximation scheme* est un PTAS en temps $poly(|I|, \frac{1}{\epsilon})$.

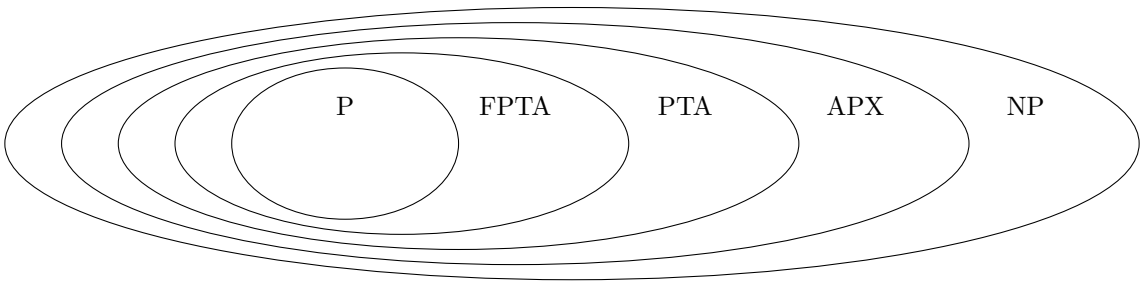
Remarque 5
- Avec PTAS, on peut avoir une complexité $2^{1/\epsilon}n^{1/\epsilon}$.
- Avec FPTAS, on ne peut pas mais on peut avoir $n^2\frac{1}{\epsilon^3}$.

Définition 6 (Classes de complexité)

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

PTA = problèmes d'optimisation admettant un PTAS

FPTA = problème d'optimisation admettant un FPTAS

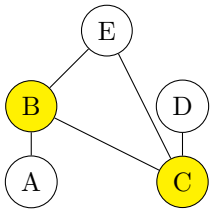


2 Couverture de sommets

Application 7 Placer le minimum de gardiens pour surveiller tous les couloirs (= arêtes).

Définition 8 (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 $v \in C$.

Exemple 9

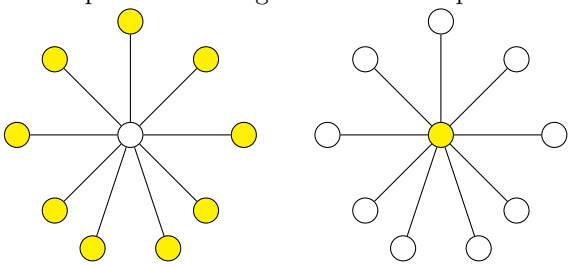


Définition 10 (problème d'optimisation) **VERTEX COVER**

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

2.1 Algorithme naïf

Résultat possible de l'algo | Solution optimale



```
Algo naïf
    Ajouter des sommets
        jusqu'à obtenir une couverture
```

2.2 Couplage

Définition 11 (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 12 $|M| \leq |C|$ pour tout couplage M, pour toute couverture C.

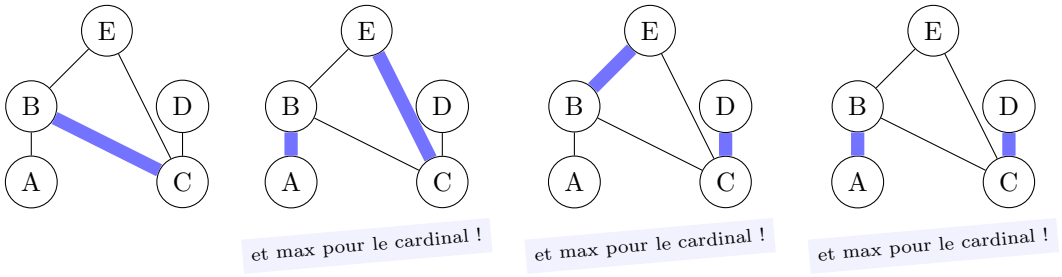
Démonstration. Chaque couloir de M est vu par un gardien (forcément différent). Plus formellement, soit f une fonction de M dans C qui associe à toute arête $e \in M$ une des extrémités de e, qui est dans C. Comme M est un couplage, on conclut car f est injective. ■

2.3 Couplage maximal pour l'inclusion (couplage maximal)

Définition 13 (couplage maximum) Un couplage M est maximum si pour tout couplage M' on a $|M'| \leq |M|$.

Définition 14 (couplage maximal pour l'inclusion) 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 15 (couplages maximaux pour l'inclusion)

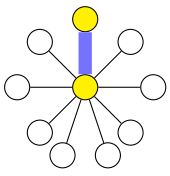


Proposition 16 Un couplage maximum pour le cardinal est maximal pour l'inclusion.

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

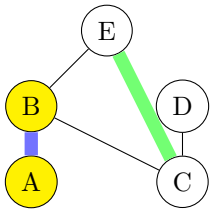
2.4 Algorithme d'approximation pour VERTEX COVER

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



Proposition 17 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. ■

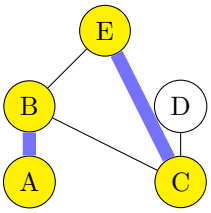
Proposition 18 VERTEX COVER admet un algorithme d'approximation est de ratio 2.

Démonstration.

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

où C^* est une couverture minimale. ■

Exemple 1 Le ratio 2 est atteint par cet algorithme, comme par exemple sur les graphes ci-dessous :



2.5 La dualité est affaiblie

Définition 19 (couplage maximum pour le cardinal) Un couplage M est maximum si $|M|$ maximal.

Proposition 20 Dans un graphe biparti, cardinal min d'une couverture = cardinal max d'un couplage.

Proposition 21 Dans un graphe qcq, card couplage maximum $\leq$ card min couverture $\leq 2\times$ card couplage maximum.

Aller plus loin

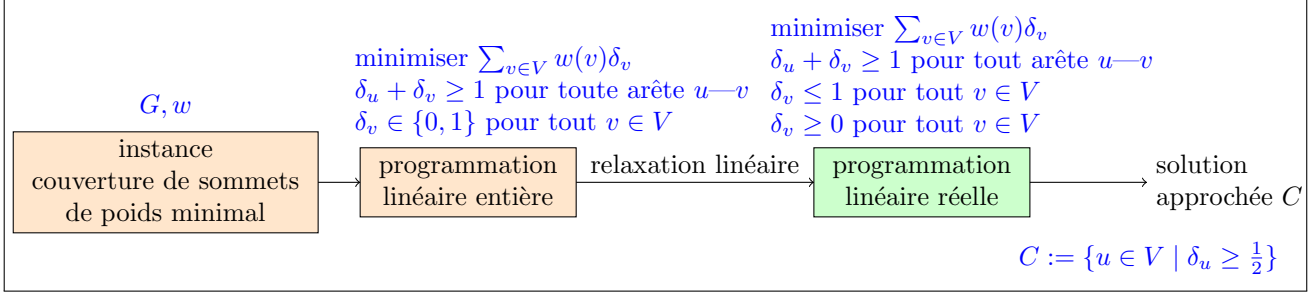
— algorithme de ratio $2 - \Theta\left(\frac{1}{\sqrt{\log |V|}}\right)$ [?].
— Si $P \neq NP$, alors pas d'algo polynomial avec un facteur ≤ 1.3606. [?]

3 Couverture de sommets pondérée et relaxation linéaire

Définition 22 **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 1 Donner un exemple pertinent.

Proposition 23 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 24 WEIGHTED VERTEX COVER admet un algorithme 2-approximant.

Démonstration.

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

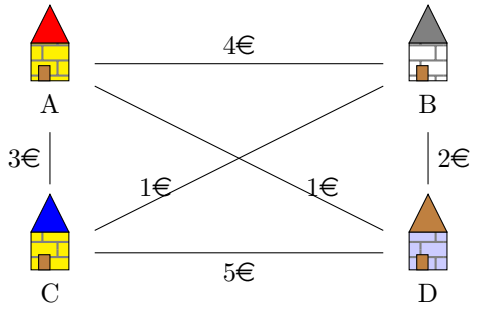
où C^* est une couverture de sommets de poids minimal. ■

4 Inapproximabilité du voyageur de commerce (TSP)

Définition 25 (problème d'optimisation) **TSP**

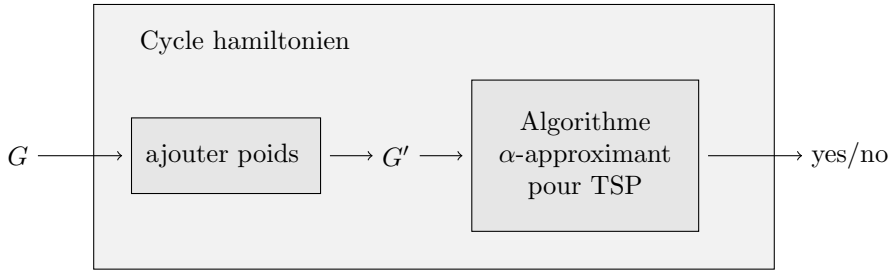
entrée : un graphe non orienté complet pondéré $G = (V, d)$ où $d : V \times V \to \mathbb{N}$ avec $d(i, j) = d(j, i)$
sortie : un tour dans G de poids minimal

Exemple 26



Théorème 27 Si P $\neq$ NP, alors TSP n'admet pas d'algo d'approximation en temps poly de ratio constant.

Démonstration. Soit α le ratio. 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 $(u, v) \notin E$.

Fait 28 G admet un cycle hamiltonien ssi G' admet un tour optimal de coût $|S|$.

Démonstration. Si G admet un cycle hamiltonien, alors ce même cycle est un tour de coût $1+1+...1 = |S|$. Réciproquement, si on a un tour de coût $|S|$, alors nous ne sommes passé que par des arêtes de E (sinon le coût serait supérieur strictement à $|S|$). ■

Fait 29 Si G' admet un tour de coût $\leq \alpha|S|$, alors il admet un tour de coût $|S|$.

Démonstration. Si un tour optimale est de coût $\leq \alpha|S|$, alors nous avons pris que des arêtes de E et donc le tour est de coût $|S|$. ■

Conclusion :

— Si G admet un cycle hamitonien, alors G' admet un tour optimal de coût $|S|$. L'algorithme produit un tour de coût $\alpha|S|$.
Réciproquement, si l'algo produit un tour de coût $\alpha|S|$, alors il y a un tour de coût $|S|$ par le fait ci-dessus, et donc G admet un cycle hamiltonien.

■

5 Voyageur de commerce avec inégalité triangulaire

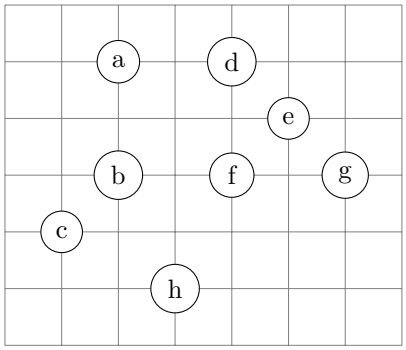
Applications : Transport en bus, faire des trous dans un morceau de bois.

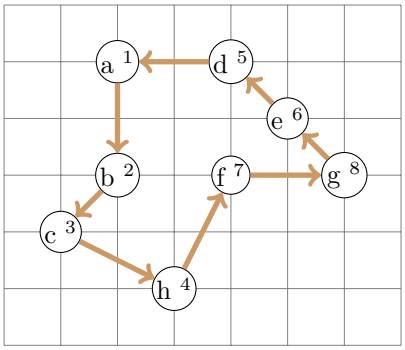
Définition 30 Un graphe non orienté complet pondéré $G = (V, d)$ vérifie l'inégalité triangulaire si $d(u, w) \leq d(u, v) + d(v, w)$ pour tous les sommets $u, v, w \in V$.

Définition 31 **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 32

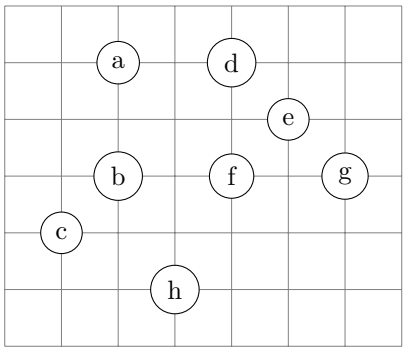




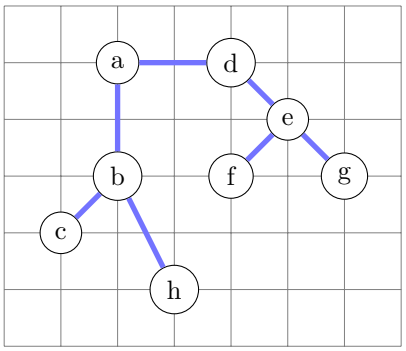
Théorème 33 METRIC TSP admet un algorithme 2-approximant.

Exemple 34

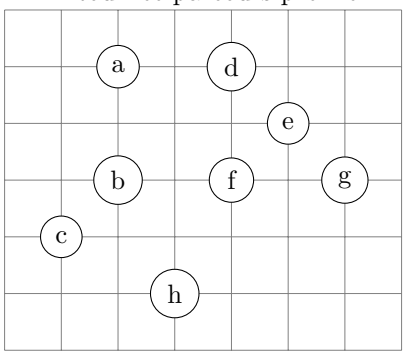
Instance G



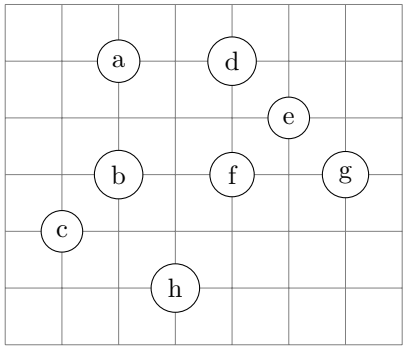
Arbre couvrant min



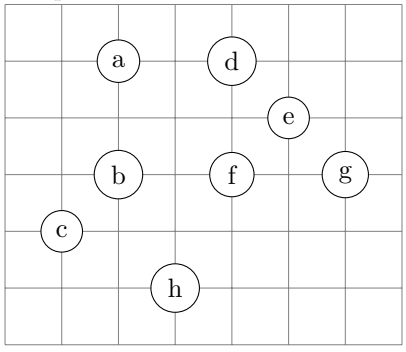
tour approximant
= tournée parcours préfixe



Arbre couvrant min où on a doublé les arêtes



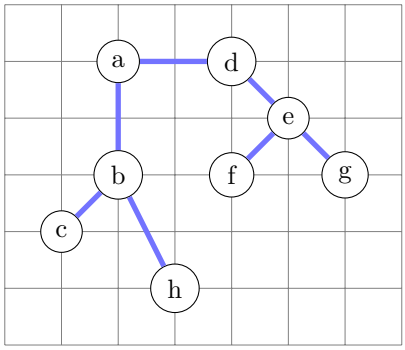
tour optimal où on a enlevé une arête



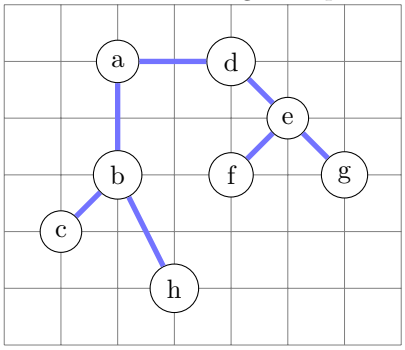
Théorème 35 (algorithme de Christofides) [Vaz04] METRIC TSP admet un algorithme $\frac{3}{2}$-approximant.

Exemple 36

Ensemble des sommets de degrés impairs



Couplage parfait de coût min des sommets de degrés impairs

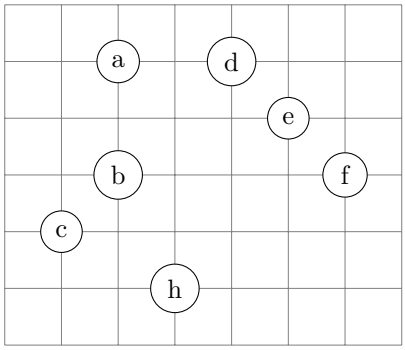


Exercice 2 Montrer que le problème du voyageur de commerce avec inégalité triangulaire est toujours NP-complet.

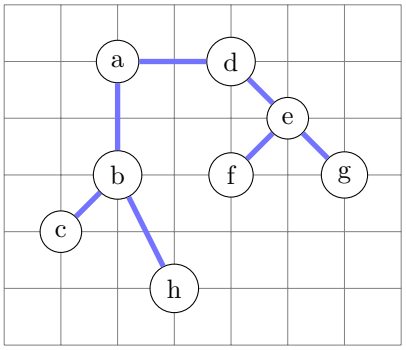
fonction algoApprox(G)
construire T un arbre couvrant minimum de G
réaliser un parcours préfixe de T
renvoyer la tournée correspondant au parcours préfixe

Exemple 2

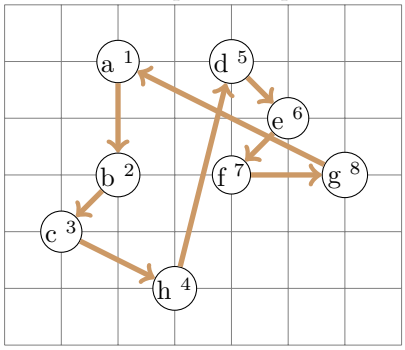
Instance G



Arbre couvrant min

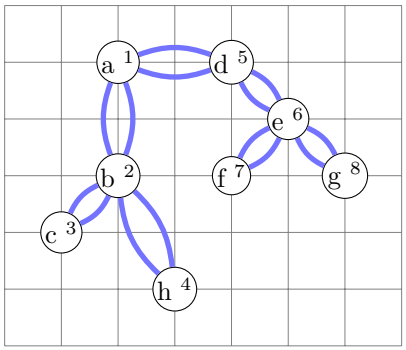


tour approximant
= tournée parcours préfixe

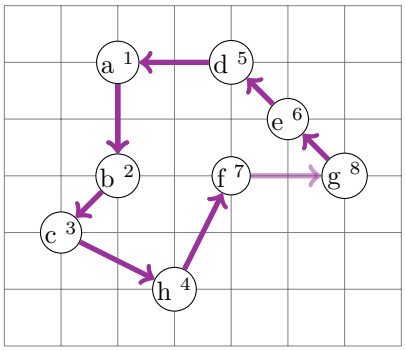


Proposition 37 Poids du tour approximant $\leq 2 \times$ poids d'un tour optimal.

Démonstration.



+ inégalité triangulaire



Tour optimal où on a supprimé une arête
= arbre couvrant

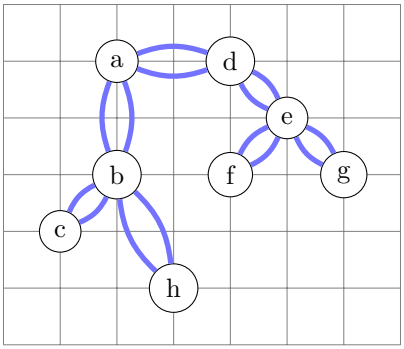
$$\frac{\dfrac{}{w(\text{tour approx}) \leq 2w(T)} \qquad \dfrac{}{w(T) \leq w(\text{tour opt})}}{w(\text{tour approx}) \leq 2w(\text{tour opt})}$$

■

5.1 Première version de l'algorithme de Christofides

fonction première version de l'algorithme de Christofides
  calculer un arbre couvrant minimum T
  $T' =$ le multi-graphe obtenu depuis T en doublant chaque arête
  $L :=$ tour eulérien dans T'
  renvoyer tour qui visite les sommets de G en ne gardant que les premières occurrences dans L

Exemple 38 Voir l'arbre T où on a doublé chaque arête :



Proposition 39 Le ratio de l'algo est de 2.

DÉMONSTRATION. $cout(\text{tour approx}) \leq 2cout(T) \leq 2cout(\text{tour opt})$. ■

5.2 Algorithme de Christofides

Au lieu de doubler toutes les arêtes de l'arbre couvrant minimum T, nous considérons sa partie "problématique" : l'ensemble I des sommets de degrés impairs en T. Calculons alors un couplage parfait à coût minimum sur I.

Définition 40 (couplage parfait) Un *couplage parfait* sur I est un couplage $M \subseteq I \times I$ tel que pour tout sommet $u \in I$, il y a exactement une arête $(a, b) \in M$ qui touche u.

Définition 41 (poids d'un couplage parfait) Le poids d'un couplage parfait est la somme des poids des arêtes du couplage.

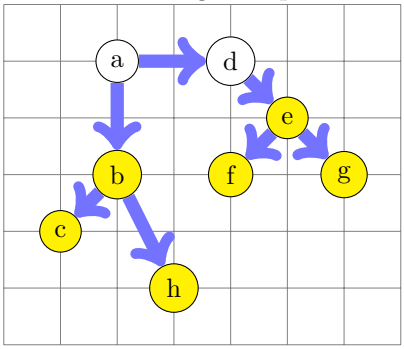
Définition 42 (couplage parfait minimum) Un couplage parfait minimum est un couplage parfait de poids minimum.

Projet 43 Algorithme d'Edmonds pour calculer un couplage parfait minimum

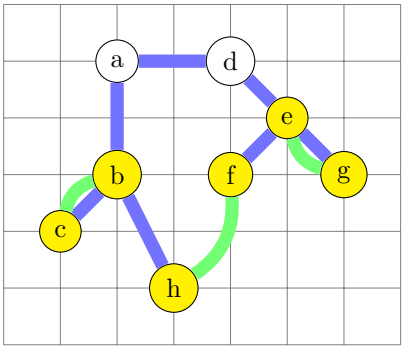
fonction algorithme de Christofides
  $T :=$ un arbre couvrant minimum
  $I :=$ sommets de degrés impairs dans T
  $M :=$ couplage parfait de coût min dans I
  $TM :=$ arêtes de $T \quad \cup \quad$ arêtes de M
  $L :=$ tour eulérien dans TM
  renvoyer tour qui visite les sommets de G en ne gardant que les premières occurrences dans L

Exemple 44

En jaune, l'ensemble I
des sommets de degrés impairs dans T



En vert, un couplage parfait
de coût min dans I



Le coût du tour retourné par l'algo est $\leq$ coût d'un tour eulérien dans TM (à cause de l'inégalité triangulaire). Le coût d'un tour eulérien dans $TM = cout(T) + cout(M) \leq cout(touroptimal) + cout(M)$ par la proposition **??**.

Exemple 45 Voici un tour eulérien dans TM :

$$a, b, c, b, h, f, e, g, e, d, a$$

On ne garde que les premières occurrences et on obtient le tour retourné par l'algo :

$$a, b, c, \not{b}, h, f, e, g, \not{e}, d, a \qquad\qquad a, b, c, h, f, e, g, d, a.$$

The cost of T is smaller than *opt* the length a minimum salesman tour. We will have proven that

Théorème 46 The cost of the computed tour is bounded by $\frac{3}{2}opt$.

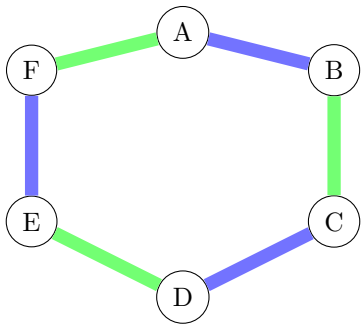
if we prove :

Lemme 47 $cout(M) \leq \frac{1}{2}opt$.

Démonstration. Considérons un tour optimal τ. Soit τ' obtenu à partir de τ en supprimant les sommets hors de I. L'inégalité triangulaire donne :

$$cost(\tau') \leq cost(\tau).$$

Le chemin τ' est un cycle sur I. Comme le nombre de sommets dans I est pair, τ' est l'union de deux couplages parfait sur I :



L'un des deux couplages est de coût $\leq \frac{cost(\tau')}{2} \leq \frac{cost(\tau)}{2}$. Comme M est un couplage parfait min sur I, $cout(M) \leq \frac{cost(\tau)}{2}$.

■

Il existe des algos avec meilleurs ratio que $\frac{3}{2}$ [**?**]. Si le problème de voyageur de commerce est dans le plan, il y a un algo avec un ratio aussi bon que l'on veut [Aro98], plus précisément il s'agit d'un PTAS.

Définition 48 Un *polynomial time approximation scheme* (PTAS) est un algorithme d'approximation qui calcule une solution de ratio $(1-\epsilon)$ (ou $(1+\epsilon)$ pour un problème de minimisation), qui polynomial en la taille de l'entrée.

Remarque 49 La complexité peut être $O(n^{3/\epsilon})$ par exemple.

6 Voyage de commerce euclidien

Définition 50 **problème du voyageur de commerce euclidien**

entrée : un ensemble P de n points de $\mathbb{R}^d$ où le poids entre points est donné par la distance euclidienne ;
sortie : un tour sur P de poids minimal.

Théorème 51 (admis) EUCLIDEAN TSP admet un PTAS [Aro98].

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

Application 52 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 53 **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 3 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.

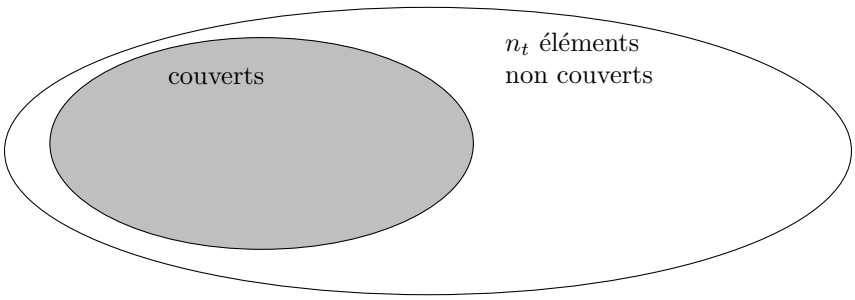
```
fonction approxSetCover(B, S_1, ··· , S_n)
    J := ensemble vide
    C := ∅
    tant que C ≠ B
        choisir j ∈ {1, ..., n} tel que |S_j ∩ B\C| soit maximal.
        si S_j ⊆ C alors
            renvoyer impossible
        C := C ∪ S_j
        ajouter j à J
    renvoyer J
```

Théorème 54 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.

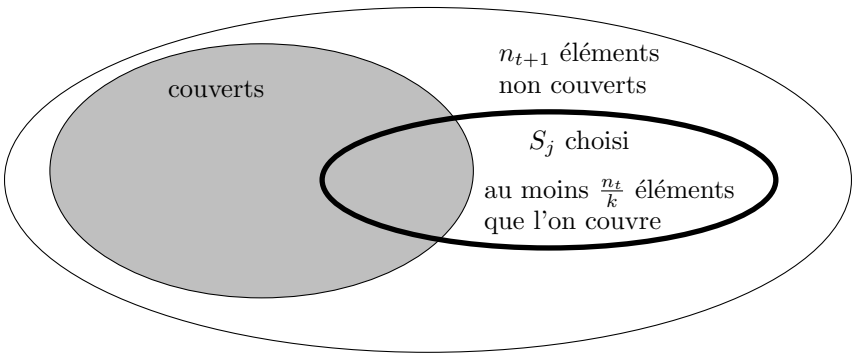
Démonstration. Supposons qu'il y a une couverture. Considérons une couverture optimale $\mathcal{S} = (S_j)_{j \in J}$ avec $|J| = k$. On note n_t le nombre d'éléments non couverts au t-ème tour de boucle. Pour $t = 0$, on a $n_0 = n$.

Lemme 55 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. ■

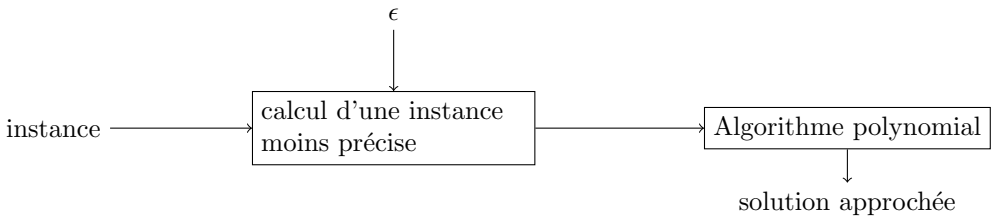
Remarque 56 Si $P \neq NP$, Raz et Safra ont montré qu'on ne faire mieux qu'un ratio de $c \log n$ où $c > 0$ [**?**].

8 Sac à dos

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 $J \subseteq \{1, \dots, n\}$ tel que $\sum_{i\in J} w_i \leq W$ et $\sum_{i\in J} v_i$ maximal.

Théorème 57 SAC A DOS admet un FPTAS.



8.1 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 58 (sous-problèmes) Pour tout $k \in \{1, \dots, n\}$, tout $V \in \{1, \dots, V^*\}$,

$$\begin{aligned} opt(k, V) &= \text{le poids minimal d'un sac contenant des objets de } \{1, \dots, k\} \text{ avec une valeur totale} \geq V \\ &= \min_{J\subseteq\{1,\dots,k\} | \sum_{i\in J} v_i \geq V} \sum_{i\in J} w_i. \\ &\qquad \text{ou } +\infty \text{ quand il n'y a pas de } J \subseteq \{1, \dots, k\} \mid \sum_{i\in J} v_i \geq V. \end{aligned}$$

Proposition 59 La solution cherchée est la valeur V maximale telle que $opt(n, V) \leq W$.

Exercice 60 Concevoir un algorithme qui calcule une solution à sac à dos en temps $O(nV^*)$.
$opt(k, 0) = 0$
$opt(0, V) = +\infty$ pour tout $V \geq 1$
$opt(k, V) = min(opt(k-1, V), w(k) + opt(k-1, V - v(k)))$

8.2 Approximation

fonction *sacadosApprox*$((w_i, v_i)_{i=1..n}, W, \epsilon)$
  $v_{max} = \max(v_1, \dots, v_n)$
  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 61 *sacadosApprox* est en temps $O(\frac{n^3}{\epsilon})$.

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

Théorème 62 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.

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

■

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.

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

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