Langages formels (11X003) - Automne 2025

6. Algorithmes d'approximation (II)

Enseignant: Arnaud Casteigts (et F. Raynaud) *Assistant: Brian Pulfer*
Moniteurs: T. von Düring & A. Maendly

Dans ce cours, nous allons voir des exemples de problèmes pour chaque catégorie d'approximation. En fait, il s'agit de plusieurs versions d'un problème que vous connaissez bien, le voyageur de commerce (TSP). Nous avons déjà parlé de ce problème pour illustrer l'approche gloutonne et la programmation dynamique. Voyons à quel degré plusieurs versions du problème sont approximables.

6.1 Trois versions du TSP

Pour rappel, une instance du TSP peut être représentée par un graphe pondéré $G = (V, E, w)$ dont les sommets V correspondent aux n villes à visiter, les arêtes E représentent les transitions possibles d'une ville à l'autre et la fonction de poids $w : E \to \mathbb{R}$ indique le coût de chacune de ces transitions (quand elles existent). Le problème consiste à trouver une tournée optimale – techniquement, un cycle de poids total minimum – qui visite chaque ville exactement une fois et revient au point de départ.

- TSP général (General TSP) : Le graphe d'entrée et les poids sont quelconques.
- TSP métrique (Metric TSP) : Il y a plusieurs définitions équivalentes. On utilisera celle où le graphe d'entrée est un graphe complet dont les coût satisfont l'inégalité triangulaire : pour tout $x, y, z \in V$, on a la garantie que $w(x, z) \leq w(x, y) + w(y, z)$. Autrement dit, il n'est jamais plus cher d'aller quelque part directement que d'y aller en passant par une ville intermédiaire, ce qui est souvent une hypothèse réaliste.
- TSP euclidien (Euclidean TSP) : Le graphe d'entrée est un graphe complet dont les coûts correspondent aux distances euclidiennes entre les villes, que l'on suppose placées dans le plan (deux dimensions).

Le TSP euclidien est un cas particulier de TSP métrique, qui est un cas particulier de TSP général. Les trois versions sont NP-difficiles, mais nous allons voir dans ce cours qu'elles s'approximent à différents niveaux.

General TSP	inapproximable
Metric TSP	1.5-approximable
Euclidean TSP	$(1+\epsilon)$-approximable

6.2 TSP général

Dans le cas général, le TSP est inapproximable : il n'existe pas de constante k telle que le problème est k-approximable (sauf si $\mathsf{P} = \mathsf{NP}$). On peut le montrer par l'absurde, en utilisant une réduction depuis le problème Hamiltonian Cycle, qui est NP-difficile. Pour rappel, ce problème consiste à décider si un graphe donné admet un cycle hamiltonien (cycle qui passe exactement une fois par chaque sommet). Voici le raisonnement :

Supposons qu'il existe un algorithme de k-approximation pour General TSP, pour un certain k. Étant donné un graphe $H = (V, E_H)$ dont on veut savoir s'il a un cycle hamiltonien, on créé un graphe complet et pondéré $G = (V, E, w)$ tel que pour toute arête $e \in E_H$, $w(e) = 1$ et pour toute arête $e \in E \setminus E_H$, $w(e) > kn$ (n'importe quelle valeur convient). On utilise alors notre algorithme pour trouver une solution S de coût $\leq k \cdot OPT$.

Observons maintenant que si un cycle hamiltonien existe, alors $OPT = n$ (il y a n arêtes dans un cycle hamiltonien et chacune a un poids égal à 1). Sinon, $OPT > kn$. On a donc la propriété que $cost(S) \leq kn$ si et seulement si un cycle hamiltonien existe. Autrement dit, en examinant $cost(S)$, on peut déterminer si H admet un cycle hamiltonien. En supposant $\mathsf{P} \neq \mathsf{NP}$, un tel algorithme ne peut donc pas exister.

6.3 TSP métrique

Le TSP métrique admet une 1.5-approximation en temps polynomial. Avant de la présenter, voyons d'abord une 2-approximation plus simple (que l'on étendra ensuite pour obtenir 1.5).

6.3.1 Algorithme de 2-approximation

1. Calculer un arbre couvrant T de poids minimum (MST)
2. Effectuer un parcours en profondeur de T en sautant les villes déjà rencontrées

Appelons S la solution ainsi obtenue. Clairement, S est bien une tournée et son calcul prend un temps polynomial (par exemple, en utilisant l'algorithme Kruskal pour le MST, c.f. Cours 6). Il nous reste donc à vérifier qu'il s'agit bien d'une 2-approximation. Quel est le coût total de S ? Un parcours en profondeur traverse chaque arête de l'arbre deux fois et les raccourcis que nous prenons ne peuvent pas détériorer ce coût (grâce à l'inégalité triangulaire). Nous avons donc :

$$cost(S) \leq 2 \cdot cost(MST) \tag{1}$$

Par ailleurs, on peut montrer que le poids total de l'arbre (en comptant une seule fois chaque arête) ne dépasse jamais le coût OPT d'une tournée optimale.

Lemme 6.1. *$cost(MST) \leq OPT$ (un arbre couvrant de poids minimum ne pèse jamais plus qu'une tournée)*

Preuve. Soit T un MST. Par l'absurde, s'il existe une tournée dont le poids est strictement inférieur à T, alors on peut enlever une arête à cette tournée et obtenir un nouvel arbre T' qui est encore moins cher que T, contredisant l'optimalité de T. □

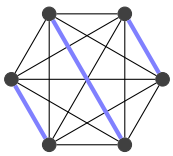
En combinant (1) et le lemme 6.1, on obtient bien :

$$S \leq 2 \cdot cost(MST) \leq 2 \cdot OPT$$

6.3.2 Algorithme de 1.5-approximation (Christofides, 1976)

L'algorithme suit la même stratégie que précédemment, mais en ajoutant des étapes qui utilisent les concepts additionnels en théorie des graphes : les couplages parfaits minimums et les cycles eulériens. Commençons par discuter de ces deux notions.

Couplage parfait minimum. Un couplage (ou matching, déjà vu au cours précédent) est un ensemble d'arêtes qui ne se touchent pas. Un couplage est parfait si il touche tous les sommets. L'existence de couplages parfaits n'est pas garantie en général, mais elle est garantie dans les graphes complets qui ont un nombre pair de sommets (voir ci-dessous). Par ailleurs, on peut trouver en temps polynomial un couplage parfait de poids total minimum (non décrit ici).



Cycle eulérien. Un cycle eulérien est un cycle qui passe exactement une fois par chaque arête du graphe. Attention, ce n'est pas pareil qu'un cycle hamiltonien (qui passe exactement une fois par chaque sommet). Le problème du cycle hamiltonien est difficile, mais le problème du cycle eulérien est facile : un tel cycle existe si et seulement si le graphe est connexe et tous les sommets ont un degré pair (nombre de voisins). Si ces conditions sont vérifiées, il est également facile de trouver un tel cycle (non décrit ici).

L'algorithme de Christofides utilise ces deux notions dans certains sous-graphes particuliers du graphe d'entrée.

L'algorithme de Christofides

1. Calculer un arbre couvrant T de coût minimum (MST)
2. Le nombre de sommets de degré impair dans T doit être pair. On peut donc calculer un couplage parfait minimum M entre ces sommets dans le graphe d'origine.
3. Faire l'union de M et T (ayant potentiellement deux fois certaines arêtes)
4. Trouver un cycle eulérien dans cette union, qui n'a que des degrés pairs
5. Parcourir le cycle obtenu en sautant les sommets déjà visités.

Explications. Dans tout graphe (et à fortiori, dans T), le nombre de sommets de degré impair est forcément pair (lemme des "poignées de main"). Le sous-graphe induit par ces sommets dans le graphe d'origine est donc un graphe complet ayant un nombre pair de sommets, on peut donc trouver un couplage parfait minimum M entre ces sommets. Si on fait l'union $T \cup M$, tous les sommets auront alors un degré pair dans cette union (soit ils avaient déjà un degré pair dans T, soit leur degré est devenu pair grâce à l'ajout du couplage). Il existe donc forcément un cycle eulérien dans $T \cup M$.

Facteur d'approximation de 1.5. La tournée finale S consiste à parcourir le cycle eulérien trouvé dans $T \cup M$, en sautant les sommets déjà visités. Grâce à l'inégalité triangulaire, ces raccourcis ne peuvent pas augmenter le coût total. On a donc :

$$cost(S) \leq cost(T \cup M) \leq cost(T) + cost(M) \tag{2}$$

On sait déjà que $cost(T) \leq OPT$ (Lemme 6.1). Il suffit donc pour conclure de montrer que le coût de M est inférieur à $0.5 \cdot OPT$. Cela se fait en deux temps. Tout d'abord, on considère le sous-graphe complet G' induit par les sommets impliqués dans M. Une tournée optimale dans ce graphe aurait un coût $OPT' \leq OPT$ (grâce à l'inégalité triangulaire, une tournée visitant seulement un sous-ensemble de villes ne peut pas être plus coûteuse qu'une tournée visitant toutes les villes). On montre ensuite :

Lemme 6.2. $cost(M) \leq 0.5 \cdot OPT'$

Preuve. Toute tournée optimale définit elle-même un couplage parfait (pas forcément minimum), en prenant une arête sur deux le long de la tournée. Elle en définit un autre : les autres arêtes de la tournée. La somme des coûts de ces deux couplages parfaits est exactement égale à OPT', donc l'un des deux ne dépasse pas la moitié, et à fortiori, un couplage parfait minimum entre ces sommets (qui ne peut pas être plus coûteux) non plus. □

En résumé, on a :

$$cost(S) \leq cost(T \cup M) \leq cost(T) + cost(M) \leq OPT + 0.5 \cdot OPT' \leq 1.5 \cdot OPT \tag{3}$$

Mais que c'est beau ! :-D

6.4 TSP euclidien

Le TSP euclidien admet une $(1+\epsilon)$-approximation, ce qui signifie qu'on peut s'approcher autant qu'on veut de la solution optimale, en payant un temps de calcul qui reste polynomial en n, mais qui augmente en fonction de l'erreur ϵ souhaitée. Un exemple classique est l'algorithme d'Arora (1996), qui permet cela en temps $n^{O(1/\epsilon)}$ pour des instances en deux dimensions (l'algorithme se généralise en payant un peu plus en dimensions supérieures).

L'idée générale est de décomposer l'espace récursivement en petites sections résolues séparément, puis recollées par de la programmation dynamique. Cet algorithme n'est pas au programme, mais c'est un classique des algorithmes d'approximation. Vous pouvez le trouver dans le cours de Michel Goemans : `https://share.google/rtk6WC5GUqRjB8gyQ`