# Two algorithmic results for the traveling salesman problem d'Alexander I. Barvinok

Vincent DESPRE

18 novembre 2012

## Table des matières

# Introduction

L'objectif de cet article est de démontré le théorème suivant :

**Théorème 1.** *Soient $d \in \mathbb{N}$, une norme $\|\cdot\|$ de $\mathbb{R}^d$, $\epsilon > 0$, $n \in \mathbb{N}$ et $P_1, \cdots, P_n \in \mathbb{R}^d$. Il existe un algorithme polynomial qui calcule une $\epsilon -$ approximation de la longueur maximale d'un circuit hamiltonien passant par $P_1, \cdots, P_n$.*

Le degré du polynôme qui donne la complexité de l'algorithme dépend de d, de la norme choisie et particulièrement, de la précision $\epsilon$ demandée. Au cours de la démonstration, on démontre aussi le résultat suivant :

**Théorème 2.** *Soient $n \in \mathbb{N}$ et $\vec{G}$ un graphe orienté de taille n. Il existe un algorithme qui donne le nombre de cycles hamiltoniens de $\vec{G}$ en $O(n^4 2^n)$. On peut construire un tel circuit en $O(n^6 2^n)$.*

Il est à noter que les algorithmes du théorème 2 ont une complexité spatiale en $O(n^2)$.

# 1 Idée de la démonstration.

Pour obtenir le résultat, on doit utiliser le fait que la dimension d de l'espace ambiant est fixée. On va utiliser la matrice d'adjacence W du graphe (coefficientée par les poids des arrêtes) et on va calculer le polynôme hamiltonien de W.

## 1.1 Structure matricielle.

Pour comprendre les conséquences de la constante d sur la structure de la matrice W, on est amené à considérer la définition suivante :

**Définition 1.** *Soit $W = (w_i j)_{i,j \in [\![1;n]\!]}$ une matrice. On appelle rang combinatoire de W le minimum des r tels que :*
*$\exists$ r vecteurs $h^s = (h_1^s, \cdots, h_n^s)$ et r vecteurs $v^s = (v_1^s, \cdots, v_n^s)$ tels que :*

$$\forall i, j \in [\![1, n]\!], w_i j = max\{h_i^s + v_j^s, s \in [\![1, r]\!]\}$$

*On note comr $W = r$.*

On dit que cette représentation des $w_i j$ est correctement centrée si

$$max\{h_i^s + v_j^s, s \in [\![1, r]\!]\} = max\{|h_i^s + v_j^s|, s \in [\![1, r]\!]\}$$

Le rang combinatoire d'une matrice est difficile à calculer exactement, cependant, il suffit d'avoir *comr* $W \leq r$. Le comportement du rang combinatoire est comparable à celui du rang habituel, par exemple, *comr* $W \leq n$.

### 1.2 Les différentes normes sur $\mathbb{R}^d$.

On s'intéresse d'abord aux normes dont la boule unité B est un polytope tel que la $\|\cdot\|_\infty$ ou $\|\cdot\|_1$. On a alors le résultat suivant :

**Lemme 3.** *Soit $\|\cdot\|$ une telle norme et $P_1, \cdots, P_n \in \mathbb{R}^d$. Alors le rang combinatoire de la matrice $W = (\|P_i - P_j\|)_{i,j\in[\![1,n]\!]}$ est borné par le nombre de faces du polytope B.*

*Démonstration.* On utilise le fait que B peut s'écrire sous la forme :

$$B = \{x \in \mathbb{R}^d / < b^s, x > \leq 1, s \in [\![1, r]\!]\}$$

Les $b^s$ sont sur les faces de B et on peut choisir les $b^s$ opposés 2 à 2. On prend : $h_i^s =< P_i, b^s >$ et $v_i^s = - < P_i, b^s >$ dans la définition 1. Dans ces conditions, la représentation obtenue pour W est correctement centrée et *comr* $W \leq r$. □

Par exemple, on obtient des bornes 2d pour $\|\cdot\|_\infty$ et $2^d$ pour $\|\cdot\|_1$. En général, on va construire un algorithme dans le cas d'une norme de ce type puis, pour une norme quelconque, on pourra construire un polytope suffisamment proche de B pour avoir une bonne approximation du résultat.

## 2 Etude du polynôme hamiltonien.

### 2.1 Une nouvelle structure euclidienne pour $\mathbb{R}[x_1, \cdots, x_n]$.

On note les exposants $\alpha = (\alpha_1, \cdots, \alpha_n)$. On a alors $P = \sum_\alpha p_\alpha x^\alpha$ et $Q = \sum_\alpha q_\alpha x^\alpha$. On a alors un nouveau produit scalaire :

**Définition 2.**

$$< P, Q >= \sum_\alpha \alpha_1! \cdots \alpha_n! \cdot p_\alpha \cdot q_\alpha$$

**Définition 3.** *Notation :*

$$P\left(\frac{\partial}{\partial_{x_1}}, \cdots, \frac{\partial}{\partial_{x_n}}\right) = \sum_\alpha p_\alpha \frac{\partial^{\alpha_1+\cdots+\alpha_n}}{\partial_{x_1}^{\alpha_1} \cdots \partial_{x_n}^{\alpha^n}}$$

On obtient :

$$< P, Q >= P\left(\frac{\partial}{\partial_{x_1}}, \cdots, \frac{\partial}{\partial_{x_n}}\right) Q\left(\frac{\partial}{\partial_{y_1}}, \cdots, \frac{\partial}{\partial_{y_n}}\right) exp(< x, y >)|_{x=y=0}$$

où $< x, y >$ représente le produit scalaire canonique de $\mathbb{R}^n$.
Si P est homogène et si $Q(x) = x_1 \cdots x_n$ (Q représentera de nouveau ce polynôme par la suite) alors on a :

$$< P, Q >= (-1)^n \cdot \sum_{\omega} (-1)^{|\omega|} P(x_\omega)$$

où $\omega$ parcourt les parties de $[\![1, n]\!]$ et $x_\omega \in \mathbb{R}^n$ est un vecteur dont la i-ième coordonnée est 1 si $i \in \omega$ et 0 sinon.

## 2.2 Calcul du polynôme hamiltonien.

On note hamp A le polynôme hamiltonien de la matrice A. On a alors le résultat suivant avec les définitions 2 et 3, et en notant $D(x) = diag\{x_1, \cdots, x_n\}$ et $P(x) = Tr(D(x)A)^n$ :

**Lemme 4.**

$$hamp\ A = \frac{1}{n} < P, Q >$$

En remarquant que P est homogène et en développant les calculs, on démontre facilement ce résultat. On définit alors l'algorithme suivant :

**Algorithme 1.** *Calcul du polynôme hamiltonien.*
*Entrée : A.*
*Sortie : hamp A.*

1. *hamp A :=0.*
2. *Pour tout $\omega \subset [\![1, n]\!]$ faire*
   *A($\omega$) :=$a_{ij}(\omega)$ est la matrice A avec $a_{ij}(\omega) = 0$ si $i \notin \omega$.*
   *Calculer $A(\omega)^n$ puis affecter :*

$$hamp\ A := hamp\ A + (-1)^{|\omega|} Tr\ A(\omega)^n$$

3. *Finalement hamp $A := \frac{(-1)^n}{n} hamp\ A$.*

On va alors démontrer le théorème 2 :

*Démonstration.* Tout d'abord, démontrons que la complexité de l'algorithme 1 est en $O(n^4 2^n)$ :
- chaque tour de boucle demande de calculer la puissance n-ième d'un polynôme de taille n, opération facilement exécutée en $O(n^4)$.
- On a $2^n$ choix pour $\omega$.
On considère A la matrice d'adjacence du graphe $\vec{G}$. Alors hamp A donne le nombre de cycles hamiltoniens dans $\vec{G}$. On va maintenant donner un algorithme qui permet de trouver un cycle hamiltonien :

**Algorithme 2.** *Calcul d'un cycle hamiltonien.*
*Entrée : A la matrice d'adjacence de* $\vec{G}$.
*Sortie : Un cycle hamiltonien de* $\vec{G}$ *s'il existe.*

1. *Vérifier que G a au moins un cycle hamiltonien grâce à l'algorithme 1.*
2. *Affecter E :=l'ensemble des arêtes de* $\vec{G}$.
3. *Pour toute arête e de* $\vec{G}$ *faire*
   *Appliquer l'algorithme 1 à* $\vec{G}$ *avec uniquement les arêtes de* $E \setminus \{e\}$.
   *Si* $\vec{G}$ *contient toujours un cycle hamiltonien, affecter* $E := E \setminus \{e\}$.
4. *Renvoyer E.*

$\vec{G}$ a, au plus, $n^2$ arêtes, l'algorithme 2 contient donc, au plus, $n^2$ tours de boucle. On obtient $O(n^6 2^n)$.
Les algorithmes 1 et 2 ne nécessitent de mémoriser qu'un nombre fini de matrices de taille n. Il nécessitent donc un volume en $O(n^2)$. □

La complexité en mémoire semble excellente puisque la simple donnée du graphe $\vec{G}$ est de taille $O(n^2)$. On remarquera que ces algorithmes gardent une complexité importante même dans les cas dégénérés car on garde toujours les $2^n$ tours de boucle de l'algorithme 1. On va maintenant traiter le cas où le rang de A est borné.

## 2.3 Calcul du polynôme hamiltonien quand la matrice a un rang borné.

On considère les matrices A dont le rang est borné par un entier r. Pour transformer la matrice A, on construit une matrice U inversible de taille n telle que les n-r dernières colonnes de AU soient nulles. On va alors s'intéresser à la matrice :

$$S(x) = U^{-1} D(x) A U$$

Ainsi, on a les résultats suivants :

$$S(x)^n = U^{-1}(D(x)A)^n U \text{ et donc } Tr(S(x)^n) = Tr(D(x)A^n)$$

On obtient alors un algorithme grâce aux lemmes précédents qui permet de calculer hamp A en $O(n^{r^2})$ :

**Algorithme 3.** *Calcul du polynôme hamiltonien paramétré par Rang(A).*
*Entrée : A et r avec* $rang(A) \leq r$.
*Sortie : hamp A.*

# 3 Adaptation des méthodes précédentes à $\mathbb{R}^d$.

Pour une matrice W d'éléments positifs, on définit :

$$c(W) = max\left\{\sum_{i=1}^{n} w_{i,\sigma(i)}, \sigma : permutation\ cyclique\ de\ taille\ n\right\}$$

On note $H_n$ l'ensemble de ces permutations.

## 3.1 Lien entre c(W) et les polynômes hamiltoniens.

On va supposer que W est donné par une représentation correctement centrée de taille r. On définit alors une famille de matrices paramétrées par $t \in \mathbb{R}$ :

$$A(t) = \left(\sum_{s=1}^{r} exp(t \cdot h_i^s) exp(t \cdot v_j^s)\right)_{i,j \in [\![1,n]\!]}$$

Pour un paramètre m, on considère la quantité :

$$c_m = \left(\frac{1}{r^n (n-1)!} \frac{d^m}{dt^m} hamp\ A(t)|_{t=0}\right)^{\frac{1}{m}}$$

On a le résultat de convergence suivant :

$$\lim_{m \to +\infty} c_m = c(W)$$

On contrôle l'erreur grâce à la formule :

$$(r^n (n-1)!)^{\frac{1}{m}} c_m \geq c(W) \geq c_m$$

Pour un certain $\epsilon$, on peut choisir m pour obtenir $(r^n (n-1)!)^{\frac{1}{m}} \leq 1 + \epsilon$, on résout cet équation pour obtenir m :

$$exp(\frac{1}{m} ln(r^n (n-1)!)) \leq 1 + \epsilon$$

$$\Leftrightarrow \frac{1}{m} ln(r^n (n-1)!) \leq ln(1+\epsilon)$$

$$\Leftrightarrow \frac{1}{m} \leq \frac{ln(1+\epsilon)}{ln(r^n (n-1)!)}$$

$$\Leftrightarrow m \geq \frac{n \cdot ln(r) + ln((n-1)!)}{ln(1+\epsilon)}$$

On prendra un m qui satisfait cette condition pour obtenir une approximation.

### 3.2 Approximation de c(W).

Pour un t fixé, on va calculer une approximation de A(t), on tronquant le développement en série entière de $exp(\cdot)$ :

$$\tilde{A}(t) = \left( \sum_{s=1}^{r} \left( \sum_{k=0}^{m} \frac{(t \cdot h_i^s)^k}{k!} \right) \cdot \left( \sum_{k=0}^{m} \frac{(t \cdot v_j^s)^k}{k!} \right) \right)_{i,j \in [\![1,n]\!]}$$

On remarque que : $\forall t \in \mathbb{R}, rang(\tilde{A}(t)) \leq r)$ et que *hamp* $\tilde{A}(t)$ est un polynôme en t de degré au plus 2mn. On peut alors réaliser l'algorithme suivant :

**Algorithme 4.** *Approximation de c(W).*
*Entrée : W de rang combinatoire au plus r donné par une représentation correctement centrée et* $\epsilon > 0$.
*Sortie : Une* $\epsilon -$ *approximation de c(W).*

1. *Pour tout* $t \in [\![0, 2mn]\!]$ *calculer les coefficients des matrices* $\tilde{A}(t)$.
2. *Pour toutes les matrices obtenue en 1, appliquer l'algorithme 3.*
3. *Grâce aux données obtenues, calculer la valeur du coefficient* $\alpha_m$ *du monôme de degré m du polynôme hamp* $\tilde{A}(t)$.
4. *Calculer puis renvoyer :* $c = \left( \frac{m!}{r^n (n-1)!} \cdot \alpha_m \right)^{\frac{1}{m}}$.

La complexité de cet algorithme est $O(2mn \cdot n^{r^2})$. On peut alors obtenir un autre algorithme avec le même raisonnement que le passage de l'algorithme 1 à l'algorithme 2, on obtient :

**Algorithme 5.** *Calcul d'un cycle hamiltonien qui approche c(W).*
*Entrée : W de rang combinatoire au plus r donné par une représentation correctement centrée et* $\epsilon > 0$.
*Sortie : Un cycle hamiltonien C tel que :* $poids(C) \leq (1 + \epsilon)c(W)$.

### 3.3 Démonstration du théorème 1.

Soit $\| \cdot \|$ une norme quelconque et un $\epsilon > 0$ quelconque.

1. On peut approcher cette norme avec une erreur $\epsilon_1 > 0$ aussi petite que l'on veut par une norme $\| \cdot \|_{bis}$ dont la boule unité est un polytope.
2. On considère la matrice $W = (\|P_i - P_j\|_{bis})_{i,j \in [\![1,n]\!]}$. Le rang combinatoire de cette matrice est bornée (Lemme 3), on peut donc appliquer l'algorithme 4 et on obtient un erreur $\epsilon_2 > 0$ aussi petite que l'on veut.
3. Pour le problème de base, il ne reste plus qu'à choisir $\epsilon_1$ et $\epsilon_2$ suffisamment petits pour obtenir une erreur globale inférieur à $\epsilon$.

## Bilan

Le fait de plonger le graphe dans $\mathbb{R}^d$ nous permet d'obtenir un algorithme polynomial pour calculer une approximation de c(W) à $\epsilon$ fixé. Cependant, on peut se demander comment se comporte la complexité de l'algorithme quand $\epsilon$ tend vers 0. L'algorithme central a une complexité en $O(m \cdot n^{r^2+1})$. m peut être choisit de taille $O(\frac{n(ln(n)+ln(r))}{ln(1+\epsilon)})$ ce qui est équivalent pour $\epsilon$ petit à $O(n(ln(n) + ln(r))\frac{1}{\epsilon})$. On a donc une partie en $\frac{1}{\epsilon}$ et une partie en $n^{r^2}$. Si la première partie est satisfaisante et facile à évaluer, la seconde est beaucoup plus délicate. Effectivement, le rang combinatoire r de la matrice dépend du nombre de faces du polytope de la norme d'approximation. Si le cas général ne semble par pouvoir donner de borne simple pour r, on peut cependant remarquer que notre algorithme a une complexité facile à évaluer dans le cas où la norme considérée n'a pas à être approchée par une autre norme. En particulier, cette méthode fonctionne très bien avec $\| \cdot \|_\infty$ qui donne $r = 2d$ !