Two algorithmic results for the Travelling Salesman Problem

Rapport par Vors Guillaume

24 novembre 2011

Résumé

Ce rapport met en valeur les résultats exposés dans [1]. Il s'agit notamment d'exposer les motivations et d'expliquer l'apport d'un tel article. Les auteurs s'intéressent à un problème largement étudié mais l'aborde depuis un angle différent.

1 Introduction

1.1 Principes généraux

Le problème du voyageur de commerce est un des problèmes les plus connus de recherche opérationnelle. Il est connu pour être NP-complet, de même que le problème du Cycle Hamiltonien. L'article met en avant deux nouveaux résultats algorithmiques sur ces problèmes. Pour rappel, on peut énoncer le problème du Cycle Hamiltonien comme suit :
Etant donné un graphe orienté G, donné par sa matrice d'adjacence, existe-t-il un cycle hamiltonien dans G, et si oui, le construire.
Le problème du Voyageur de Commerce est une version dans laquelle les arêtes ont des poids du problème ci-dessus. Concernant la complexité du problème, voici quelques résultats. La complexité temporelle considère le nombre d'opérations arithmétiques (addition, soustraction, multiplication, division et comparaison à un réel). La recherche exhaustive peut être accompli avec une complexité polynomiale en l'entrée, mais sa complexité temporelle est en O(n!). Dans [3], la méthode dynamique offre une complexité temporelle en $O(n^2.2^n)$ mais une complexité spatiale en $O(2^n)$. Dans [2], "HPA3" offre une complexité spatiale en $O(n)$ mais une complexité temporelle en $2^{2n}.n^{O(1)}$.

Dans l'article, les auteurs s'intéressent à trois aspects, tout d'abord trouver un algorithme offrant un meilleur compromis entre complexité spatiale et complexité temporelle pour le problème du Cycle Hamiltonien. Ils cherchent aussi à donner une condition sur le graphe permettant d'obtenir une complexité polynomiale. Ensuite ils essayent d'extrapoler ce résultat pour le problème du Voyageur de Commerce, avec une erreur relative ϵ. Et ensuite étudier le problème du Voyageur de Commerce dans un espace euclidien (le poids d'une arête est la distance entre 2 sommets). Ici, on se concentrera sur le premier aspect, ainsi que les outils permettant d'extrapoler depuis Circuit Hamiltonien vers Voyageur de Commerce. L'article a vraiment un objectif algorithmique, il s'agit ici de donner des

algorithmes meilleurs que ceux connus. Il a donc peu de références à d'autres travaux, si ce n'est pour donner l'état de l'art.

1.2 Notations

Dans toute la suite on notera S_n l'ensemble des permutations de $\{1, ..., n\}$, et H_n le sous ensemble de S_n des permutations ne contenant qu'un seul cycle.

Définition 1.2.1. *Pour un graphe pondéré G, avec n sommets et la matrice de poids $W = (w_{ij})$, le poids du cycle hamiltonien le plus lourd peut s'écrire :*

$$c(W) = max\{\sum_{i=1}^{n} w_{i,g(i)} \ : \ g \in H_n\}$$

Définition 1.2.2. *Soit $A = (a_{ij})$ une matrice carré réelle n x n, on appele permanent de A l'expression :*

$$per\ A = \sum_{g \in S_n} \prod_{i=1}^{n} a_{i,g(i)}$$

Pour cette même matrice A, on peut aussi définir le permanent hamiltonien :

$$hamp\ A = \sum_{g \in H_n} \prod_{i=1}^{n} a_{i,g(i)}$$

On remarque que si A est la matrice d'adjacence d'un graphe orienté, hamp A calcule le nombre de cycle hamiltonien du graphe. Le calcul du permanent et celui du permanent hamiltonien sont des problèmes #P-dur.

2 Outils mathématiques

Nous avons vu en cours que l'une des analogies de la théorie de la complexité algébrique est de ramener le calcul du permanent au calcul d'un polynôme via un circuit et inversement. Ici, les auteurs partent de ce constat mais essayent de trouver une forme calculatoire à cette analogie. Afin de construire des algorithmes permettant de calculer le permanent en calculant des polynômes, ils définissent un produit scalaire sur P_n l'anneau des polynômes à n variables réelles. C'est à partir de ce produit scalaire qu'ils calculent le permanent.

2.1 Produit scalaire de polynômes

Soit $P \in P_n$, on considère l'ensemble de ses arguments $x = (x_1, ..., x_n)$ comme un vecteur de R^n. On désigne par $\langle ., . \rangle$ le produit scalaire classique sur R^n Pour un ensemble d'indice $\alpha = (\alpha_1, ..., \alpha_n)$ où les α_i sont des entiers positifs, on désigne par x^α le monôme $x^\alpha = x_1^{\alpha_1} ... x_n^{\alpha_n}$.

$$Pour \quad P = \sum_{\alpha} p_\alpha . x^\alpha \ \ et \ \ Q = \sum_{\alpha} q_\alpha . x^\alpha$$

on défini le produit scalaire $\langle ., . \rangle$ par :

$$\langle P, Q \rangle = \sum_{\alpha} \alpha_1! ... \alpha_n! q_\alpha p_\alpha$$

Il s'agit d'un produit scalaire qui se c̈omporte bienv̈is-à-vis des transformations linéaire sur les variables $x_1, ...x_n$. Plus précisément, si on considère G un opérateur linéaire sur R^n, et G^* son conjugué, i.e. $\langle x, Gy \rangle = \langle G^* x, y \rangle$. Pour un polynôme $P \in P_n$, on défini le polynôme $G(P) \in P_n$ par :

$$G(P)(x) = P(G^*(x))$$

On a alors :

Théorème 2.1. *Pour tout $P, Q \in P_n$ et pour tout opérateur G sur R^n :*

$$\langle G(P), Q \rangle = \langle P, G^*(Q) \rangle$$

2.2 Quelques résultats

Tout d'abord, un premier résultat qui est utilisé dans les preuves de la suite de l'article.

Lemme 2.1. *Soit $P \in P_n$ un polynôme homogène de degré n et $Q(x) = x_1 ... x_n$. Pour chaque sous-ensemble $\omega \subset \{1,...,n\}$ notons $x_\omega \in R^n$ le vecteur dont la i-ème coordonnée est égale à 1 si $i \in \omega$ et 0 sinon. Alors :*

$$\langle GP, Q \rangle = (-1)^n . \sum_{\omega} (-1)^{|\omega|} P(x_\omega)$$

où la somme est faite sur tous les sous-ensembles ω possibles.

Pour un polynôme comprenant un nombre $O(n^r)$ termes, il faut $O(n^r)$ opérations arithmétiques pour calculer son expansion en somme de monômes. Si on suppose r fixé, alors le calcul du produit scalaire de deux polynômes contenant $O(n^r)$ monômes se fait en temps polynomial. Il est donc intéressant de de se ramener à un tel cas. L'intuition des auteurs a été de se ramener à calculer le permanent et le hamiltonien sur des matrices de rang r fixé, et voir si l'on pouvait exploiter cette restriction.

3 Algorithmes

3.1 Permanent et Hamiltonien

On cherche à calculer le permanent d'une matrice n x n, de rang inférieur à r, où r est fixé. Pour cela les auteurs proposent un algorithme fonctionnant en 3 parties :
- Tout d'abord calculer une matrice n x n $B = (b_{ij})$ telle que $b_{ij} = 0$ si i>0, et une matrice G telle que GB = A. - Définir deux polynômes L et R de degré n sur r variables $x_1, ..., x_n$ par :

$$L(x_1, ..., x_n) = \prod_{j=1}^{n} \sum_{i=1}^{r} b_{ij} . x_i \ , \ R(x_1, ..., x_n) = \prod_{j=1}^{n} \sum_{i=1}^{r} g_{ij} . x_i$$

Puis calculer leur forme étendue

$$L = \sum_{\alpha} \lambda_\alpha . x^\alpha \ , \ R = \sum_{\alpha} \rho_\alpha . x^\alpha$$

- Calculer

$$per\ A = \sum_{\alpha} \alpha_1 ! ... \alpha_n ! \lambda_\alpha \rho_\alpha$$

Cela nous donne un algorithme effectuant un nombre polynomial en n d'opérations arithmétiques. En effet, le calcul de G et B revient à résoudre un problème d'algèbre linéaire, ce que l'on peut faire en $O(n^3)$ opérations. r étant fixé, le calcul de L et R, ainsi que leur expansion et le calcul de leur produit scalaire se fait aussi en temps polynomial, dont le degré dépend de r. Montrons maintenant que le permanent est bien le produit scalaire de L et R. Pour cela on pose $P(x) = \prod_{j=1}^{n} \sum_{i=1}^{n} a_{ij} . x_i$ et Q(x) = $x_1 ... x_n$, alors par définition on a

$$per\ A = \langle P, Q \rangle$$

Or G(L) = P, donc à l'aide du Théorème 2.1, on obtient

$$per\ A = \langle G(L), Q \rangle = \langle L, G^*(Q) \rangle$$

$G^*(Q) = \prod_{j=1}^{n} \sum_{i=1}^{n} g_{ij} . x_i$, mais L est polynôme sur seulement les r premières variables donc

$$per\ A = \langle L, G^*(Q) \rangle = \langle L, R \rangle$$

De la même manière que pour le permanent les auteurs propose un algorithme polynomial pour calculer hamp A lorsque A est rang inférieur à r, avec r fixé. La méthode est similaire à celle vu au dessus, seules les matrices changent. Mais une fois les matrices obtenus, on en obtient 2 polynômes, que l'on étend puis on calcule le produit scalaire.

3.2 Problème du Cycle Hamiltonien

Les auteurs se concentre tout d'abord à démontré le théorème suivant :

Théorème 3.1. *Il existe un algorithme qui pour tout $n \in N$, et pour tout graphe orienté G avec n sommets, calcule le nombre de cycles hamiltoniens du graphe avec une complexité temporelle $O(n^4.2^n)$ et une complexité spatiale $O(n^2)$. Si G contient un cycle hamiltonien, alors l'algorithme construit un cycle en complexité temporelle $O(n^6.2^n)$ et complexité spatiale $O(n^2)$.*

Un tel algorithme serait le plus efficace pour un algorithme ayant une complexité spatiale polynomial en l'entrée.

Pour la preuve ils créent un algorithme qui calcule le hamiltonien d'une matrice avec la complexité souhaitée. Une fois cet algorithme trouvé, la suite vient naturellement. En calculant le hamiltonien de la matrice d'adjacence de G, on calcule le nombre de cycles hamiltonien. Reste alors à en construire un, pour cela, on utilise la méthode diviser pour régner. Pour cela, on choisi une arête e, on la supprime, et on vérifie si le nouveau graphe contient encore un cycle hamiltonien, si non on la remet dans le graphe. On recommence

de la même manière pour chacune des arêtes et il ne restera que les arêtes d'un cycle hamiltonien. On va donc répéter n^2 fois l'algorithme de vérification, ce qui nous donne bien les complexités souhaitées.

Pour démontrer l'existence, et construire, de l'algorithme de vérification, les auteurs se basent sur le lemme suivant :

Lemme 3.1. *Pour $x = (x_1, ..., x_n) \in R^n$ soit la matrice n x n $D(x) = diag\{x_1, ..., X_n\}$. Pour toute matrice A, on défini :*

$$P(x_1, ..., x_n) = Tr(D(x)A)^n$$

où Tr désigne la trace d'une matrice. P est un polynôme homogène de degré n, on choisi $Q(x) = x_1...x_n$, on a alors

$$hamp\ A = \frac{1}{n}\langle P, Q\rangle$$

La preuve de ce lemme se base sur $T_s(x)$, le s-ième élément de la diagonale de $Tr(D(x)A)^n$. On a

$$T_s(x) = \sum_I \prod_{j=1}^{n} x_{ij} a_{i_j, i_{j+1}}$$

où la somme est faite sur toutes les séquences I = $(1 \leq i_1, ..., i_n \leq n)$ avec $i_1 = s$ et $i_{n+1} = s$. Après quelques calculs, on s'aperçoit que

$$\langle T_s, Q\rangle = hamp\ A$$

ce qui nous donne le lemme.

La construction de l'algorithme à partir du lemme se fait naturellement en utilisant le lemme 2.1 pour se ramener à calculer la puissance et la trace de matrice plus simples.

4 Extrapolation

4.1 Rang combinatoire

On a vu que lorsqu'une matrice est de rang inférieur à un certain r, il est alors possible d'avoir un algorithme éfficace pour calculer per A et hamp A. Les auteurs se posent alors la question de savoir quelles seraient les restrictions permettant d'avoir un algorithme polynomial qui répond au problème suivant :

Etant donné un graphe orienté complet G pondéré par W, avec n sommets, on souhaite calculer c(W).

Pour commencer ils comparent :

$$c(W) = max\{\sum_{i=1}^{n} w_{i,g(i)} \ : \ g \in H_n\}$$

$$hamp\ A = \sum_{g \in H_n} \prod_{i=1}^{n} a_{i,g(i)}$$

On s'aperçoit qu'il y a certaines similarités, il suffit de remplacer $\sum$ par max, et $\prod$ par $\sum$ dans hamp pour obtenir c(W). On va effectuer ce changement dans la notion de rang, on obtient alors la définition suivante.

Définition 4.1.1. *Soit $W = (w_{ij})$ une matrice n x n. On dit que le rang combinatoire de W ne dépasse pas r si il existe r vecteurs de réels $h^s = (h_1^s, ..., h_n^s$ et r vecteurs de réels $v^s = (v_1^s, ..., v_n^s$ pour s de 1 à r, tels que*

$$w_{ij} = max\{h_i^s + v_j^s,\ s = 1, ..., r\}$$

On dit que la résentation ci-dessus est bien-centrée si

$$max\{h_i^s + v_j^s,\ s = 1, ..., r\} = max\{|h_i^s + v_j^s|,\ s = 1, ..., r\} \quad \forall\ 1 \leq i, j \leq n$$

4.2 Algorithme

On considère maintenant que la matrice de poids du graphe est bien-centrée et qu'elle est donnée par les 2r vecteurs h^s et v^s. A partir de là, l'objectif des auteurs est le suivant : Prouver que pour tout $r \in N$, il existe un algorithme en temps polynomial qui pour toute matrice de poids W de rang combinatoire inférieur à r, donnée par sa représentation bien centrée, et pour tout $\epsilon > 0$, calcule le cycle hamiltonien dont le poids approxime, avec une erreur relative inférieure à ϵ.

Pour cela, ils procèdent en 2 étapes, tout d'abord calculer approximativement le poids maximal c(W), puis calculer un cycle qui convient. On s'intérèsse à la première partie, pour cela ils utilisent le lemme

Lemme 4.1. *On suppose que W est une matrice de poids positifs, donnée par sa représentation bien centrée. Etant donné un paramètre réel t, on défini la matrice A(t) par*

$$a_{ij}(t) = \sum_{s=1}^{r} exp\{t.h_i^s\}.exp\{t.v_j^s\}$$

pour $1 \leq i, j \leq n$. Pour un $\epsilon > 0$ donné, on choisi un entier pair m tel que

$$m > \frac{log(n-1)! + n.logr}{log(1+\epsilon)}$$

alors si

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

c_m approxime le plus gros poids c(W) avec une erreur relative inferieure à ϵ.

A l'aide de ce lemme, on calcule une approximation de c(W), en calculant A (on remplace exp par les m premiers termes de son développement de Taylor). Puis on calcule hamp A, et on obtient c_m, ce qui nous donne une bonne approximation de c(W).

5 Conclusion

Dans ce rapport, j'ai choisi de ne pas m'attarder sur les calculs présent dans les preuves où dans les algorithmes, mêmes si ils représentent une grande partie de l'article. J'ai fait ce choix car je considère que l'essentiel de l'intérêt de l'article est sur la forme et non le fond. Les auteurs donnent une restriction permettant d'obtenir un algorithme polynomial pour les problèmes du Cycle Hamiltonien et du Voyageur de Commerce. Ils donnent aussi un algorithme plus efficace que ceux déjà connus pour le Cycle Hamiltonien. Et tous leurs résultats se basent sur l'intuition de calculer ces problèmes à l'aide de polynômes, alors que normalement, c'est le cheminement inverse que l'on fait. La définition d'un produit scalaire original sur les polynômes leur a donné la piste de recherche pour les algorithmes. Mais il y a certains points noirs, notamment pour le Voyageur de Commerce. Dans cette partie ils supposent tout du long que les matrices de poids sont donnés par leurs représentation bien centrée, mais il s'agit d'une supposition très forte, car on ne sait pas quelle est la complexité pour obtenir cette représentation. Ni même si l'on peut efficacement savoir si une matrice est de rang combinée inférieur à r ou non. Dans la dernière partie ils exposent des résultats sur les espaces euclidiens. On se rend compte que la caractérisation du poids sur les espaces euclidiens est particulière et permet certains résultats que l'on a pas dans le cas général.

Références

[1] Alexander I. Barvinok. Two algorithmic results for the travelling salesman problem. jan 2001.

[2] Yu. Gurevich and S. Shelah. Expected computation time for hamiltonian path problem. *SIAM J. Comput.*, (16) :486–502, 1985.

[3] E. Lawler. The travelling salesman problem : a guided tour of combinatorial optimization. 1985.