Liens entre problèmes de plus courts chemins, de fermeture transitive et de multiplication de matrices.

Bertrand Marc

22 mai 2007

Table des matières

Introduction

Je vais essayer de présenter brièvement les problèmes que nous allons étudier. Le plus intuitif est de classer ces problèmes en trois familles :

1. Les problèmes de plus court chemins
2. Les problèmes de fermeture transitive

3. Les problèmes de multiplication de matrices

Ces problèmes se regroupent naturellement de cette manière. Nous allons en effet voir que ces familles correspondent aux réductions les plus simples et les plus intuitives. Nous verrons ensuite dans la dernière partie les liens entre ces différentes familles de problèmes.

1 Algorithmes de plus courts chemins

Fixons quelques conventions : nous nous plaçons dans un graphe orienté $G = (V, E)$ (on pose $n = |V|$ et $m = |E|$), muni d'une fonction de pondération $w : E \to \mathbb{R}_+^*$ qui associe à chaque arc un poids à valeur réelle. On lui associe une distance : $\delta(u, v) = \begin{cases} \min\{w(p) : u \rightsquigarrow v\} & \text{s'il existe un chemin p de u à v} \\ \infty & \text{sinon} \end{cases}$

1.1 Plus courts chemins à origine unique

Il s'agit ici de trouver tous les plus courts chemins entre un sommet $s \in V$ (la source) et le reste du graphe. L'algorithme le plus utilisé dans ce cas est l'algorithme de Dijkstra :

Algorithme 1 Dijkstra(G,s,w)

```
pour chaque sommet v ∈ V faire
    d[v] ← ∞
    π[v] ← NIL
fin pour
d[s] ← 0
E ← ∅
F ← V
tantque F ≠ ∅ faire
    u ← min F
    E ← E ∪ {u}
    pour chaque sommet v ∈ Adj[u] faire
        si d[v] > d[u] + w(u, v) alors
            d[v] ← d[u] + w(u, v)
            π[v] ← u
        finsi
    fin pour
fin tantque
```

Pour étudier correctement la complexité de cet alogorithme, il faut préciser la ligne $u \leftarrow \min F$. Il s'agit en fait de trouver le sommet u qui minimise $d[v]$, $v \in F$. Et selon comment on met en oeuvre la file de priorité qui permet de calculer le minimum (en fait selon comment on implante F). Avec un simple tableau, on obtient une complexité en $O(n^2)$. Avec des tas de Fibonacci, on peut descendre à $O(n \ln n + m)$

1.2 Plus court chemin entre deux sommets u et v

Si on résoud le problème à origine unique pour le sommet origine u, on résoud également ce problème de manière triviale. Par ailleurs on ne connait aucun algorithme qui soit meilleur asymptotiquement que les algorithmes pour le problème à origine unique.

Ainsi, si je sais résoudre le problème de plus court chemin à origine unique en temps $O(T(n, m))$, alors je sais aussi résoudre ce problème en temps $O(T(n, m))$. En pratique, on a donc une solution en $O(n \ln n + m)$.

1.3 Plus courts chemins pour tout couple de sommets

Ce problème peut être résolu en éxecutant un algorithme à origine unique à partir de chaque sommet. Ainsi, si je sais résoudre le problème de plus court chemin à origine unique en temps $O(T(n, m))$, alors je sais aussi résoudre ce problème en temps $O(n \times T(n, m))$ (et de même l'espace utilisé est multiplié par n). En pratique on obtient alors une complexité en $O(n^2 \ln n + nm)$.

Il est à noter qu'il existe aussi des algorithmes calculant directement tous les plus courts chemins, comme l'algorithme de *Floyd-Warshall* qui s'execute en $O(n^3)$ et l'algorithme de *Johnson* qui s'execute en $O(n^2 \ln n + nm)$.

1.4 Cas non pondéré

On dispose ici d'un graphe orienté sans fonction de pondération des arêtes et l'on veut résoudre les différents problèmes de plus courts chemins. Il est important de préciser que la distance utilisée ici est :

$$\delta'(u, v) = \begin{cases} \min\{longueur(p) : u \rightsquigarrow v\} & \text{s'il existe un chemin p de u à v} \\ \infty & \text{sinon} \end{cases}.$$

Il est clair que l'on peut munir le graphe d'une fonction de pondération constante w qui renvoie 1 sur toute arête de E. Et dans ce cas, il est évident que $\delta = \delta'$. A partir de là, on peut dire que le cas non pondéré est un cas particulier du cas pondéré et donc on obtient les mêmes complexités.

Mais on peut aussi modifier l'algorithme de Dijkstra. On peut en effet remarquer qu'il est très proche d'un parcours en largeur. Et en appliquant un

parcours en largeur à notre graphe, on peut calculer les plus courts chemins à partir d'une source fixe. On peut ainsi résoudre le problème des plus courts chemins à origine unique en $O(n+m)$. Et grâce à la réduction qui précède, on a une solution du problème de plus courts chemins pour tout couple de sommets en $O(n^2+nm)$.

2 Calcul de fermeture transitive

2.1 Fermeture transitive d'un graphe orienté

La fermeture transitive d'un graphe $G=(V,E)$ est définie par le graphe $G^*=(V,E^*)$ où $E^*=\{(i,j) : \text{il existe un chemin reliant } i \text{ à } j \text{ dans } G\}$.

Montrons que l'on peut calculer la fermeture transitive d'un graphe à partir du calcul des plus courts chemins pour tout couples de sommets. Si D est la matrice résultant de l'algorithme des plus courts chemins ($d_{ij} = \min\{longueur(p) : i \rightsquigarrow j\}$ en supposant que les sommets sont numérotés de 1 à n). Alors on a clairement l'équivalence $d_{ij}=\infty \Leftrightarrow$ il n'y a pas de chemin de i à j. On construit alors la matrice C de la manière suivante : $c_{ij}=0 \Leftrightarrow d_{ij}=\infty$ et $c_{ij}=1 \Leftrightarrow d_{ij}<\infty$. La matrice C est bien évidemment la matrice d'adjacence de G^*.

Ainsi si l'on sait résoudre le problème des plus courts chemins pour tout couple de sommets en $O(T(n,m))$, on sait calculer la fermeture transitive en $O(T(n,m)+n^2)$. En pratique, cela nous donne une complexité en $O(n^2+nm)$.

2.2 Fermeture transitive d'une matrice booléenne

La fermeture transitive d'une matrice booléenne M est défine par $M^* = \bigvee_{n\geq 0} M^n$. On peut remarquer qu'une matrice booléenne M est aussi la matrice d'adjacence d'un graphe G. Mais que signifie M^k en terme de matrice d'adjacence ? Montrons par récurrence que $(M^k)_{ij}=1$ si et seulement si il existe un chemin de longueur k entre i et j dans G.

Pour $k=1$, cela découle trivialement de la définition d'une matrice d'adjacence.

Supposons que c'est vrai pour un certain k, et montrons que c'est vrai pour $k+1$.

$$(M^{k+1})_{ij} = \bigvee_{p=1}^{n} (M^k)_{ip} m_{pj}$$

Celà signifie que $(M^{k+1})=1 \Leftrightarrow \exists p\ (M^k)_{ip}=1$ et $m_{pj}=1$. Ceci est équivalent à : il existe un chemin de longueur k entre i et p, et $(p,j)\in E$.

C'est à dire il existe un chemin de longueur $k+1$ entre i et j. Ainsi $(M^*)_{ij} = 1$ si et seulement si il existe un chemin dans G entre i et j. Donc M^* est la matrice d'adjacence de G^*.

Ainsi on peut calculer la fermeture transitive d'un graphe ou d'une matrice booléenne indiférement : c'est la même chose, et donc c'est la même complexité. En pratique on peut donc calculer la fermeture transitive d'une matrice booléenne en $O(n^2 + nm)$. où m est le nombre de 1 contenu dans M.

3 Multiplication de matrices

3.1 Mutiplication de matrices réelles

Il existe plusieurs algorithmes de multiplication de matrices réelles : l'algorihtme naïf fonctionne en $O(n^3)$. Il existe des algorithmes plus efficaces, comme Strassen (en $O(n^{2,81})$) ou Coppersmith et Winograd (en $O(n^{2,376})$). Je ne vais pas présenter ces algorithmes ici, mais je vais plutôt essayer de faire le lien avec les problèmes qui nous intéresse.

3.2 Multiplication de matrices booléennes

On pourrait penser que multiplier des matrices booléennes est un cas particulier des matrices réelles, mais ce n'est pas tout à fait aussi simple. Il faut travailler dans un anneau pour pouvoir appliquer les algorithmes de la famille de Strassen. Mais ce n'est pas un réel problème. Pour calculer $C = AB$, Il suffit de multiplier les matrices dans $\mathbb{Z}_{n+1}$ (qui est un anneau) : $D = AB$ dans $\mathbb{Z}_{n+1}$. On remarque que $c_{ij} = 0 \Leftrightarrow d_{ij} = 0$ et $1 \leq d_{ij} \leq n \Leftrightarrow c_{ij} = 1$. Ainsi si on sait multiplier des matrices réelles avec une complexité en $M(n)$, on sait multiplier des matrices booléennes en $O(M(n) + n^2)$. En pratique on peut donc multiplier les matrices booléennes avec une complexité en $O(M(n))$.

4 Réduction entre ces différentes catégories

4.1 Matrices booléennes

Théorème 1 *Le temps $T(n)$ nécessaire pour calculer la fermeture transitive d'une matrice booléenne est du même ordre que le temps $M(n)$ nécessaire pour calculer le produit de deux matrices booléennes.*

Preuve 1 *Soit X une matrice $n \times n$. On supposera pour l'instant que $n = 2^k$. On partitionne X en quatre matrices égales : $X = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$. Si on suppose que X représente un graphe $G = (V, E)$ où les sommets sont séparés en deux parties égales V_1 et V_2, alors la matrice A représente les arêtes entre les sommets de V_1, la matrice D représente les arêtes entre les sommets de V_2, la matrice B représente les arêtes entre V_1 et V_2 et la matrice B représente les arêtes entre V_2 et V_1.*

Calculons X^*, on peut commencer par le coin supérieur gauche : un chemin entre deux sommets de V_1 a deux possibilités :

- *ce chemin reste dans V_1*
- *ce chemin passe par V_2 avant de revenir.(il peut donc être représenté par BD^*C)*

Ainsi ces chemins sont représentés par $E = (A+BD^*C)^*$. De même, on peut calculer les autres éléments de X^* : $X^* = \begin{pmatrix} E & EBD^* \\ D^*CE & D^* + D^*CEBD^* \end{pmatrix} = \begin{pmatrix} E & F \\ G & H \end{pmatrix}$ (on pourrait bien sûr trouver des écritures plus simples, mais elles induiraient plus de calculs par la suite). On peut calculer ces quatres matrices grâce à la séquence suivante :

$$\begin{aligned} T_1 &= D^* \\ T_2 &= BT_1 \\ E &= (A + T_2C)^* \\ F &= ET_2 \\ T_3 &= T_1C \\ G &= T_3E \\ H &= T_1 + GT_2 \end{aligned}$$

On a donc calculé 2 fermetures transitives, 6 multiplications et 2 additions sur des matrices de tailles moitié : $T(2^k) \leq 2T(2^{k-1}) + 6M(2^{k-1}) + 2 \times 2^{2k-2}$. On obtient par récurrence : $T(2^k) \leq cM(2^k)$.

Pour le cas où n n'est pas une puissance de 2, il suffit de compléter la matrice X de la manière suivante pour arriver à une puissance de 2 : $\begin{pmatrix} X & 0 \\ 0 & I \end{pmatrix}$, où I représente l'identité.

4.2 Les Semi-anneaux fermés

Théorème 2 *Si on sait calculer le produit de deux matrices en temps $O(M(n))$, alors on peut calculer les plus courts chemins pour tout couple de sommets*

en $O(M(n)\ln n)$

Preuve 2 *On va calculer les plus courts chemins pour tout couple de sommets en se basant sur la procédure suivante. En entrée on donne une matrice* $D^{(m)}$ *telle que* $d_{ij}^{(m)}$ *est la longueur minimale d'un chemin d'au plus* m *arcs du sommet* i *au sommet* j*, et la matrice* W *des poids des arêtes de* E*. En sortie on récupère la matrice* $D^{(m+1)}$

Algorithme 2 Extensions-plus-courts-chemins(D,W)

Soit D' une matrice $n \times n$
pour $i = 1$ à n **faire**
  pour $j = 1$ à n **faire**
    $d'_{ij} \leftarrow \infty$
    pour $k = 1$ à n **faire**
      $d'_{ij} \leftarrow \min(d'_{ij}, d_{ik} + w_{kj})$
    fin pour
  fin pour
fin pour
retourner D'

On peut remarquer que cet algorithme est exactement le produit de deux matrices, à ceci près que l'on a remplacé l'opération $+$ *par* min *et l'opération* $\times$ *par* $+$*. Ainsi cette procédure peut être effectuée comme un produit de matrices, et donc avec la même complexité. Et on peut aussi utiliser l'associativité de ce "produit", ce qui justifie l'exponentiation rapide ci-dessous.*

Comme on se place dans un graphe sans circuit de poids négatif, tous les plus courts chemins sont de longueur $\leq n-1$*. Donc tous les plus courts chemins se trouvent dans la matrice* $D^{(n-1)}$*. Pour la calculer, on va utiliser un algorithme d'exponentiation rapide :*

Algorithme 3 Plus-courts-chemins-tout-couples(W)

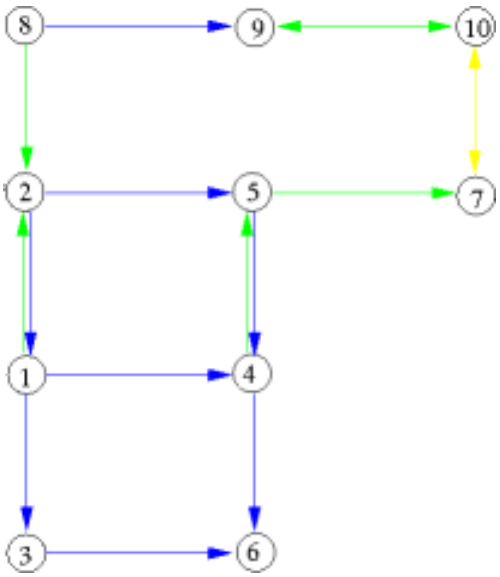
$D^{(1)} \leftarrow W$
$m \leftarrow 1$
tantque $m < n-1$ **faire**
  $D^{(2m)} \leftarrow$ Extensions-plus-courts-chemins$(D^{(m)}, D^{(m)})$
  $m \leftarrow 2m$
fin tantque
retourner $D^{(m)}$

En fait, on a utilisé une structure algébrique particulière qui permet de mettre en parallèle le produit de matrice et la procédure Extensions-plus-courts-chemins. Pour être rigoureux, on aurait dû montré que $S_1 = (\mathbb{R}, \min, +, \infty, 0)$ et $S_2 = (\mathbb{R}, +, \times, 0, 1)$ sont des *semi-anneaux fermés* et il aurait ensuite fallu montré les propriétés pricipales des semi-anneaux fermés.

Conclusion

Comme on a pu le voir, tous ces problèmes sont liés. On obtient assez facilement des réductions pertinentes entre ces problèmes polynomiaux. On aurait même pu en rajouter d'autres, comme l'utilisation du calculs de plus courts chemins (avec des poids négatifs) dans le cadre de programmation linéaire (dans le cas particulier des contraintes de potentiel).

Voici le schéma bilan des réductions présentées ici. Les flèches jaunes marquent les équivalences, les bleues pointent les problèmes que l'on peut déduire directement (ou les cas particuliers), et enfin les vertes marquent les problèmes que l'on peut résoudre en utilisant le problème source. Les numéros de problèmes sont indiqués juste en dessous avec la meilleure complexité au vu de ce qui a été démontré ici.



1. Plus courts chemins à partir d'un sommet source, dans un graphe orienté dont les arcs ont des poids positifs en $O(n \ln n + m)$
2. Plus courts chemins entre tous les sommets, dans un graphe orienté dont les arcs ont des poids positifs en $O(n^2 \ln n + nm)$

3. Plus court chemin entre deux sommets, dans un graphe orienté dont les arcs ont des poids positifs en $O(n \ln n + m)$
4. Plus courts chemins à partir d'un sommet source, dans un graphe orienté sans poids en $O(n + m)$
5. Plus courts chemins entre tous les sommets, dans un graphe orienté sans poids en $O(n^2 + nm)$
6. Plus court chemin entre deux sommets, dans un graphe orienté sans poids en $O(n + m)$
7. Fermeture transitive d'un graphe orienté en $O(n^2 + nm)$
8. Multiplication de deux matrices sur un anneau en $O(n^{2,81})$
9. Multiplication de deux matrices booléennes en $O(n^{2,81})$
10. Fermeture transitive d'une matrice booléenne en $O(n^2 + nm)$

Bibliographie

- *Introduction à l'algorithmique*, livre de T. Cormen, C. Leiserson, R. Rivest (et C. Stein dans la dernière version), Dunod (1994)
- *The design and analysis of computer algorithms*, livre de Aho, Hocroft et Ullman, Addison-Wesley (1974)