Projet MOGPL

Samuel Lalam
Santhos Arichandra

Décembre 2021

Question 1

- Assertion 1

Nous avons dans l'exemple le chemin de a à k (notée C1). Le chemin de type I pour le chemin C1 est de 7, en passant par les arcs :

1. a1 : (a,b,2)
2. a2 : (b,g,3)
3. a3 : (g,k,2)

A présent prenons le sous-chemin de a à b (notée SC1). Le chemin de type I pour SC1 est de 1 avec comme arc (a,b,1).
Donc nous avons bien un sous-chemin préfixe d'arrivée au plus tôt, qui n'est pas le chemin d'arrivée au plus tôt.

Donc l'assertion est vraie

- Assertion 2

Nous avons dans l'exemple le chemin de a à l (notée C2). Le chemin de type II pour le chemin C2 est de 4, avec comme arcs :

1. a1 : (a,c,4)
2. a2 : (c,h,6)
3. a3 : (h,i,7)
4. a4 : (i,l,8)

A présent prenons le sous-chemin i à l (notée SC2). Le début au plus tard (chemin de type II) de SC2 est de 9, or dans C2 nous avions 8 pour l'arc i à l.

Donc l'assertion est vraie.

- Assertion 3

Nous avons dans l'exemple le sous-chemin a à k (notée C3). Le chemin le plus rapide est de 4 (7+1-4), en passant par les arcs :

1. a1 : (a,c,4)
2. a2 : (c,h,6)
3. a3 : (h,k,7)

A présent prenons le sous-chemin a à h (notée SC3), on constate que le chemin le plus rapide est (valeur : 2):

1. a1 : (a,b,1)
2. a2 : (b,h,3)

Ce qui ne correspond pas aux a1 et a2 du chemin C3.

Donc l'assertion est vraie.

- Assertion 4

Nous avons dans l'exemple le chemin de a à l (notée C4). Le chemin de le plus court (chemin de type IV) a pour valeur 3 en passant par les arcs:

1. a1 : (a,f,3)
2. a2 : (f,i,5)
3. a3 : (i,l,8)

A présent prenons le sous-chemin de a à i (notée SC4), on constate que le chemin le plus court n'est pas a1 puis a2, mais l'arc (a,i,10), ce qui fait une valeur de 1.

Donc l'assertion est vraie.

Question 2

- Algorithme du chemin de type I

Entree : $\tilde{G}$ = (S,A), init, dest
Pour cet algorithme on va faire une transformation en plus de celle du $\tilde{G}$.
Voici la transformation réalisée :
$\tilde{G}$ = (S,A) vers newG = (S,newA)
∀ ((av,valAv),t,(ap,valAp)) ∈ A:
    si t==0 and av==dest:
        add (av,valAv),t,(ap,valAp) dans newA
    sinon:
        add (av,valAv),valAp-valAv,(ap,valAp)) dans newA

Suite à cette transformation, on applique l'algorithme de Dijkstra sur newG.
Sortie : Chemin d'arrivée au plus tôt de init à dest.

- Algorithme du chemin de type II

Entree : $\tilde{G}$ = (S,A), init, dest
Premièrement, on récupère tous les sommets de S égale à init (ListeInit).
On va trier la liste ListeInit de manière décroissante par rapport à la valeur de départ. (Donc la première valeur de la liste ListeInit est le sommet init le plus tard).

∀ i ∈ ListInit:
    res = BFS avec $\tilde{G}$, i, dest
    si res != [] (il existe un chemin)
        return res
return []
Sortie : Chemin de départ au plus tard de init à dest.

- Algorithme du chemin de type III

Entree : $\tilde{G}$ = (S,A), init, dest
Premièrement, on récupère tous les sommets de S égale à init (ListeInit), de même pour dest (ListeDest).
∀ (ilabelSommet, iValDepart) ∈ ListInit:
    ∀ (dlabelSommet, dValDepart) ∈ ListeDest:
        ajouté à ways (ilabelSommet, dlabelSommet, dValDepart-iValDepart)

Trier ways de manière croissante par rapport à la troisième valeur de ways

∀ (iSommet,dSommet,time) ∈ ways:
    res = BFS avec $\tilde{G}$, iSommet, dSommet
    si res != [] (il existe un chemin)

return res
return []
Sortie : Chemin le plus rapide de init à dest.

- Algorithme du chemin de type IV

Entree : $\tilde{G}$ = (S,A), init, dest

On lance l'algorithme de Dijkstra avec $\tilde{G}$, init, dest
Sortie : Chemin le plus court d'init à dest.

Question 3

- Complexité de l'algorithme du chemin de type I

Soit $\tilde{G}$ = (S,A)
Dans cet algorithme nous réalisons une transformation qui est de complexité $O(|A|)$.
Suite à cette transformation, on applique l'algorithme de Dijkstra, qui est de complexité $O(|A| + |S| \log |S|)$

Donc la complexité de l'algorithme est de $O(2 * |A| + |S| \log |S|)$

- Complexité de l'algorithme du chemin de type II

Soit $\tilde{G}$ = (S,A)
On cherche le chemin de a a k, avec le départ au plus tard.
Dans cet algorithme, on doit d'abord récupérer tous les sommets de a du graphe G, ce qui correspond a une complexité de $O(|S|)$
Puis on lance BFS sur tous les sommets a, donc au pire on a une complexité $O(|S| * (|S| + |A|))$.

La complexité de l'algorithme est donc de $O(|S| * (|S| + |A|) + |S|)$

- Complexité de l'algorithme du chemin de type III

Le chemin le plus rapide de init a dest

Soit $\tilde{G}$ = (S,A)
Soit nbInits égale au nombre de sommets de S ayant pour label init
Soit nbDests égale au nombre de sommets de S ayant pour label dest

$nbCouples = nbInits * nbDests$

La complexité est de $O(nbCouples + nbCouples * (|S| + |A|))$

- Complexité de l'algorithme du chemin de type IV

Soit $\tilde{G}$ = (S,A)
Dans cet algorithme, on utilise directement l'algorithme de Dijkstra sur $\tilde{G}$.

Donc la complexité de l'algorithme est de $O(|A| + |S| \log |S|)$

Question 5

Le programme linéaire permettant de trouver le chemin le plus court entre init et dest est le suivant:

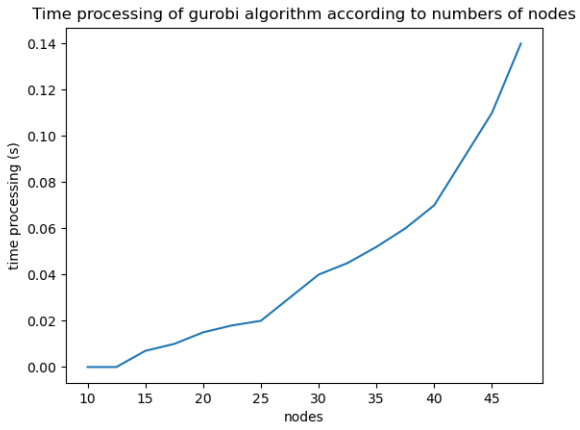
Minimiser $\sum_{ij \in A} w(i,j) x_{ij}$

sous contraintes:

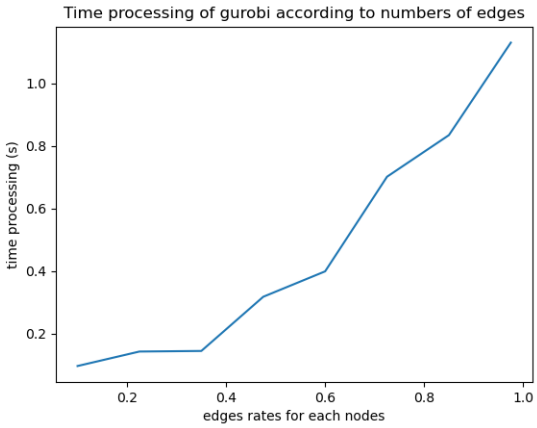
$$\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1 & \text{si } i = init \\ -1 & \text{si } i = dest \\ 0 & \text{sinon} \end{cases}$$

avec $\forall$i,j $x_{ij} \geq 0$

Question 6

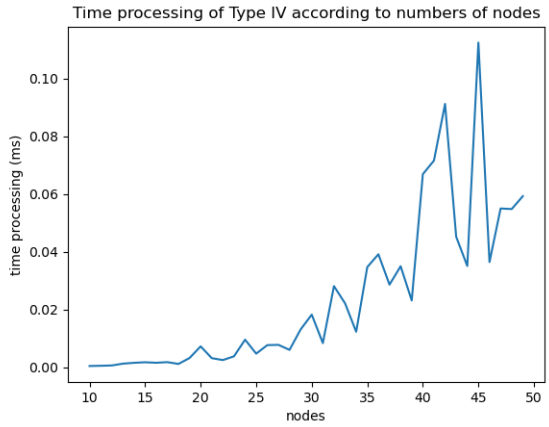


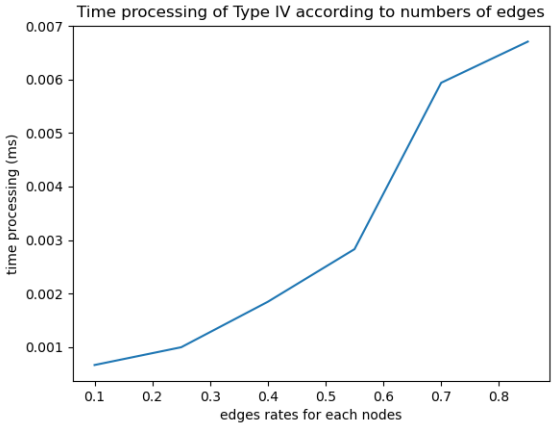
On constate qu'en faisant varier le nombre de sommets, le temps d'exécution de l'algorithme Gurobi augmente exponentiellement.



On constate qu'en faisant varier le nombre d'arêtes pour un nombre de sommets fixes, le temps d'exécution de l'algorithme Gurobi augmente de manière linéaire.

Question 7





On s'aperçoit très clairement que l'algorithme de la question 4 concernant le chemin de type IV est bien plus rapide que celui de Gurobi on parle ici d'une différence de grandeur de 10^3. On peut aussi conclure que le nombre d'arêtes, fait varier le temps de manière linéaire, que ce soit pour la question 4 ou Gurobi. Pour finir, il n'y aucun intérêt à utiliser Gurobi, au vu des résultats.