Projet MOGPL

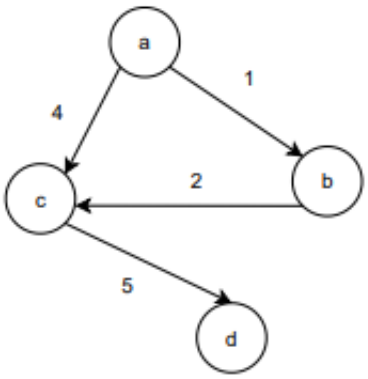
Antoine TOULLALAN Noureddine YAKHOU

December 5, 2021

Question 1 :

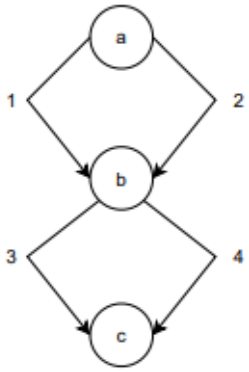
Assertion 1

Un sous-chemin préfixe d'un chemin d'arrivée au plus tôt peut ne pas être un chemin d'arrivée au plus tôt.
Si on prend le chemin de a à d { a-(4)↦c-(5)↦d} qui est un chemin d'arrivée au plus tôt le sous chemin de a à c { a-(4)↦c } n'est pas un chemin d'arrivée au plus tôt car pour aller du sommet a au sommet c le chemin d'arrivée au plus tôt est le chemin { a-(1)↦b-(2)↦c } vue que celui -ci nous permet d'arriver à c le jour 3 et que le chemin direct nous permet d'arriver le jour 5.



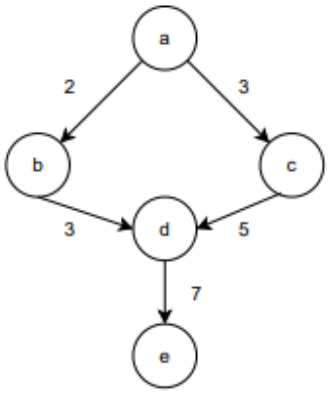
Assertion 2

Un sous-chemin postfixe d'un chemin de départ au plus tard peut ne pas être un chemin de départ au plus tard.
Si on prend le chemin de départ au plus tard de a à c qui est { a-(2) ↦ b-(3)↦c } son sous chemin de b à c n'est pas un chemin de départ au plus tard car pour aller du sommet b au sommet c le chemin de départ au plus tard est{ b-(4) ↦c }



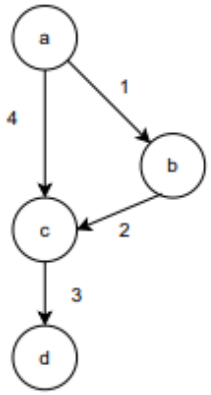
Assertion 3

Un sous chemin d'un chemin le plus rapide peut ne pas être le chemin le plus rapide.
Si on prend le chemin le plus rapide dans notre cas qui est { a-(3)-d-(5)-e } son sous chemin pour aller de a à d n'est pas le chemin le plus rapide car pour aller de a à d le chemin{ a-(2)↦b-(3)↦d } est le chemin de plus rapide vue qu'il nous permet d'arriver à d le jour 4 tant dis que notre sous chemin de départ nous permet d'arriver à d le jour 6.



Assertion 4

: un sous chemin d'un plus court chemin peut ne pas être un plus court chemin.
Dans notre cas le chemin le plus court pour aller de a à l est {a-(1)↦b-(2)↦c-(3)↦d } vue que le chemin {a-(4)↦c-(3)↦d } n'est pas possible ,le sous chemin de a à c n'est pas le plus court chemin vue qu'il s'étend sur 2 arc et que le chemin { a-(4)↦c } prend seulement un arc ce qui fait que ce dernier est le plus court chemin.



Question 2 :

Calcul du chemin de type I

Pour calculer le chemin d'arrivée au plus tôt P dans G tel que $P \in P(x, y, [td, tf])$, on construit $\tilde{G} = (\tilde{V},\tilde{E})$. Dans $\tilde{G}$, on prend x' = (x,tmin) le sommet de départ avec tmin = min(t $\parallel$ (x,t) $\in \tilde{E}$, $t \geq td$) et y' = (y,tmax) le sommet d'arrivée avec tmax = max(t $\parallel$ (y,t) $\in \tilde{E}$, $t \leq tf$).
On modifie $\tilde{G}$: pour tous les sommets s **sauf y**, pour chaque arcs (s, t_i) $\rightarrow$ (s, t_{i+1}), on met un poids égale à $t_{i+1} - t_i$ à cet arc.C'est à dire qu'on ajoute les temps d'attente pour chaque sommet de G.

On calcule ensuite avec l'algorithme de Djikstra le chemin de moindre coût de x' à y'. On a ainsi un chemin P' dans $\tilde{G}$ (v0,v1,v2,...,vn). Pour avoir un chemin P dans G, on prend les sommets appartenant à G qui sont visités dans P' dans le même ordre. Par exemple, si dans l'exemple de graphe $\tilde{G}$ de l'énoncé, on a le chemin de a à f avec notre algorithme: P'=((a,1),(a,2),(b,3),(b,5),(f,6),(f,7)),
le chemin associé dans P sera P= (a,b,f) avec l'arête de a-b =(a,b,2,1) et l'arête b-f =(b,f,5,1) avec une arrivée dans f à t=6 qui est bien l'arrivée la plus tôt.

On a ainsi un chemin dans P qui est le chemin d'arrivée au plus tôt car dans $\tilde{G}$, chaque arête est pondéré par le temps d'attente/trajet depuis sauf pour les arêtes entre les sommets y où le poids est 0. Le chemin de moindre cout dans $\tilde{G}$ est donc le chemin qui "arrive" après le moins de temps d'attente/trajet possible à un sommet (y,t), càd qui permet d'avoir le moins de temps entre le sommet de départ et le sommet d'arrivée.
On a donc bien un chemin qui est celui de l'arrivée la plus tôt de x à y.

Exemple du graphe G modifié dans l'annexe.

Calcul du chemin de type II

Pour le calcul du chemin de départ P le plus tard dans G tel que $P \in P(x, y, [td, tf])$, on construit $\tilde{G} = (\tilde{V},\tilde{E})$. Comme dans le calcul du chemin I, dans $\tilde{G}$, on prend x' = (x,tmin) le sommet de départ avec tmin = min(t $\parallel$ (x,t) $\in \tilde{E}$, $t \geq td$) et y' = (y,tmax) le sommet d'arrivée avec tmax = max(t $\parallel$ (y,t) $\in \tilde{E}$, $t \leq tf$).
On modifie $\tilde{G}$: pour tous les sommets s **sauf x**, pour chaque arcs (s, t_i) $\rightarrow$ (s, t_{i+1}), on met un poids égale à $t_{i+1} - t_i$ à cet arc.

On calcule ensuite avec l'algorithme de Djikstra le chemin de moindre coût de x' à y'. On a donc un chemin P' dans $\tilde{G}$.

De la même manière que dans le calcul du chemin I on passe d'un chemin P' dans $\tilde{G}$ à un chemin P dans G.

P est le chemin de départ le plus tard car dans $\tilde{G}$, chaque arête est pondéré par le temps d'attente/trajet depuis sauf pour les arêtes entre les sommets x où le poids est 0. Le chemin de moindre coût de x' à y' dans $\tilde{G}$ est donc le chemin qui "part" après le plus de temps d'attente possible d'un sommet (x,t) (x est le sommet de départ) car le chemin emprunte alors des arêtes de coût nuls, càd dont le coût est inférieurs aux autres arêtes qui "part" après moins de temps d'un sommet (x,t), c'est donc le sous-chemin que va emprunter le chemin de moindre cout.

On a donc bien un chemin qui est celui de départ le plus tard de x à y.
Exemple du graphe G modifié dans l'annexe.

Calcul du chemin de type III

Pour le calcul du chemin le plus rapide P dans G tel que $P \in P(x, y, [td, tf])$, on construit $\tilde{G} = (\tilde{V},\tilde{E})$. Comme dans le calcul du chemin I et II, dans $\tilde{G}$, on prend x' = (x,tmin) le sommet de départ avec tmin = min(t $\parallel$ (x,t) $\in \tilde{E}$, $t \geq td$) et y' = (y,tmax) le sommet d'arrivée avec tmax = max(t $\parallel$ (y,t) $\in \tilde{E}$, $t \leq tf$).
On modifie $\tilde{G}$: pour tous les sommets s **sauf x et y**, pour chaque arcs (s, t_i) $\rightarrow$ (s, t_{i+1}), on met un poids égale à $t_{i+1} - t_i$ à cet arc.

On calcule ensuite avec l'algorithme de Djikstra le chemin de moindre coût de x' à y'. On a donc un chemin P' dans $\tilde{G}$.

De la même manière que dans le calcul du chemin I on passe d'un chemin P' dans $\tilde{G}$ à un chemin P dans G.

P est le chemin le plus rapide car les arêtes dans $\tilde{G}$ connectant les sommets (x,t) entre eux sont nulles, idem pour les les arêtes connectant les sommet (y,t).Ainsi, dans le chemin de moindre coût on maximise les temps d'"attente" dans les sommets de départ et d'arrivée, donc on minimise les temps de trajet/attente entre les sommets de départ et les sommets d'arrivée dans P'. On a donc un chemin avec un temps de trajet/attente minimal dans P.

On a donc bien un chemin qui est le plus rapide de x à y.
Exemple du graphe G modifié dans l'annexe.

Calcul du chemin de type IV

Pour le calcul du chemin le plus court P dans G tel que $P \in P(x, y, [td, tf])$, on construit $\tilde{G} = (\tilde{V},\tilde{E})$. Contrairement aux parties précédentes, on ne modifie pas $\tilde{G}$.
On prend x' = (x,tmin) le sommet de départ avec tmin = min(t $\parallel$ (x,t) $\in \tilde{E}$, $t \geq td$) et y' = (y,tmax) le sommet d'arrivée avec tmax = max(t $\parallel$ (y,t) $\in \tilde{E}$, $t \leq tf$).

On calcule ensuite avec l'algorithme de Djikstra le chemin de moindre coût de x' à y'. On a donc un chemin P' dans $\tilde{G}$.De la même manière que dans le calcul du chemin I on passe d'un chemin P' dans $\tilde{G}$ à un chemin P dans G.

Ce chemin P est le plus court dans G car comme P' est de moindre cout dans $\tilde{G}$, on change d'aéroport le moins souvent possible.

On a donc bien un chemin P qui est le plus court de x à y.
Exemple du graphe G modifié dans l'annexe.

Question 3 :

Pour les 4 chemins, on calcule avec un algorithme de Djikstra le chemin de moindre coût sur un graphe qui a le même nombre de sommets et d'arêtes. Donc les 4 algorithmes ont la même complexité.
Soit le sommet G avec n sommets et m arêtes, on suppose qu'on calcule les chemins I,II,III,IV dans P(x,y,[td,tf]) et que $td \geq tf$.

On a pour chaque sommet de G, au plus tf-td sommets associés dans $\tilde{G}$ (on a tf-td temps différents) et aussi au plus m sommets différents pour chaque sommet de G dans $\tilde{G}$ (car chaque arête représente un trajet de durée 1, donc comme chaque sommet est l'extrémité d'au plus m arêtes représentant un trajet, dans $\tilde{G}$, chaque sommet est associé à au plus m sommets). Donc pour chaque sommet dans G, on a au plus min((tf-td),m) associé dans $\tilde{G}$.
On note tm=min((tf-td),m).

Pour chaque sommet de G, on a au plus tm sommets associés dans $\tilde{G}$ et pour si un sommet de G donne k ($k \leq tm$) "nouveaux" sommets, ces sommets sont connectés par k-1 arêtes. Donc pour chaque sommet de G, on a dans $\tilde{G}$ au plus (tm-1)*n arêtes supplémentaires.

Donc dans $\tilde{G}$, soit n' le nombre de sommet et m' le nombre d'arêtes, on a $n' \leq n * tm$ et $m' \leq m + (tm - 1) * n$. Or lors des calculs des chemins I,II,III,IV, on applique un algorithme de Djikstra, donc la complexité des algorithmes est: O(n'^2) donc O($n^2 * tm^2*$).

Donc la complexité des algorithmes des chemins I,II,II,IV est O($n^2 * tm^2*$).

Question 5 :

Supposons qu'on ait un multigraphe G sans cycle. Pour calculer le plus court chemin du chemin x à y, on calcule le graphe $\tilde{G}$ associé avec n sommets (qu'on note s1,s2...sn) et m arêtes (qu'on note a1,a2..am). A partir de $\tilde{G}$, on crée le programme linéaire P avec m+1 contrainte et n variables:

On ajoute une borne supérieure aux variables de P, car dans le cas où un sommet n'est pas accessible depuis le sommet de départ, la variable associée à ce sommet ne sera pas bornée (avec une borne supérieure). Il faut que cette borne soit supérieure aux variables des sommets qui ont comme voisin le sommet d'arrivée. Donc ,si le calcul du chemin IV se fait dans P(x,y,[td,tf], on prend comme borne supérieure tf+1.

Soit s le sommet (x,t) dans $\tilde{G}$ avec t=min $\{t \parallel (x,t) \in \tilde{V}\}$

Max s1+s2+...+sn

s=0
ou s$\leq$0 Pour chaque arête ai=(si,sj,val), on crée la contrainte
sj-si$\leq$ val

s1,s2,...,sn$\geq$ 0
La résolution de P donne une valeur à chaque sommet car chaque variable représente un sommet dans $\tilde{G}$. Pour chaque sommet si, on a donc la valeur vi qui est la longueur du plus court chemin de x à si.
Pour connaitre le plus court chemin de x à y, on se place au sommet y et on prend ,parmi les sommets qui ont comme voisins y, celui qui a la valeur associée la plus petite, à partir de ce sommet s1 on prend, parmi les sommets qui ont comme voisins s1, celui qui a la valeur associée la plus petite,... on itère jusqu'a être au noeud x. On a ainsi le chemin le plus court de x à y.

En annexe, on montre le programme linéaire pour le multigraphe G de l'énoncé.

Question 6 :

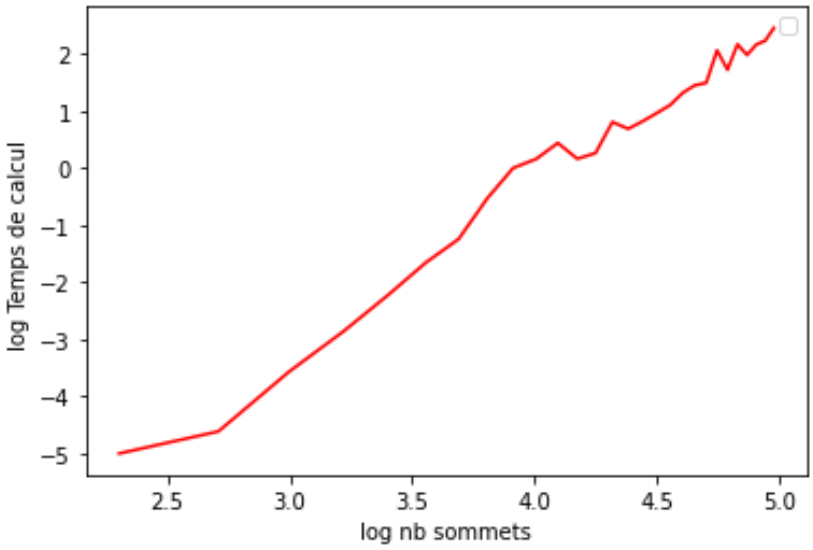
Pour observer les temps de calcul de notre méthode en fonction de la taille de l'entrée, on a crée une foncion qui génère des multigraphes aléatoirement. Cette fonction prend en entrée le nombre de sommets, la probabilité de jonction de deux sommets, tdebut et tfin.
Pour éviter les cycles dans notre graphe, on divise les sommets en sous-ensemble qu'on nomme niveau1,niveau2,... et on peut placer des arêtes qu'entre les sommets d'un niveau i aux sommets

du niveau i+1. Si on joint 2 sommets (avec une probabilité p), on lie ces 2 sommets avec arbitrairement 1,2 ou 3 arêtes avec des temps choisient aléatoirement entre tdebut et tfin-1.

On observe ainsi les temps de calcul de la méthode utilisant gurobi en fonction du nombre de sommets du graphe, et du nombre d'arêtes.
Pour influer sur le nombre d'arêtes du multigraphe, on modifie la probabilité p de jonction de 2 sommets par 1,2 ou 3 arêtes (de manière aléatoire).

Pour chacun de ces 2 paramètres, on en fait varier un en fixant l'autre, on a ainsi les graphes suivants en échelles logarithmique:

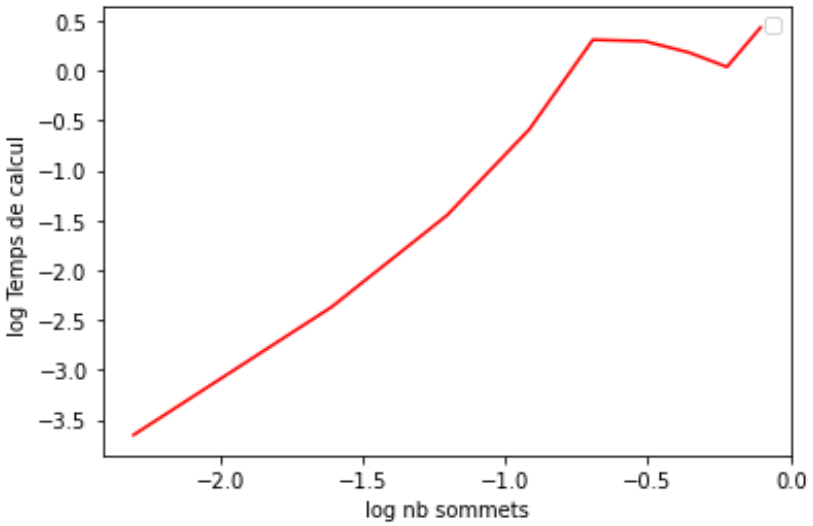
Log Nombre de sommets en fonction du log de temps de calcul



On voit qu'on a une forme de droite qui a un coefficient directeur d'environ 3 (2.7). On en déduit que le temps de calcul de notre méthode évolue de manière polynomiale en fonction du nombre de sommets avec un coefficient de 3

.

Log de la probabilité de jonction de sommets en fonction du log de temps de calcul



On a ici aussi une forme de droite avec un coefficient directeur d'environ 2. On en déduit que le temps de calcul évolue de manière polynomiale en fonction du nombre d'arêtes avec un coefficient de 2.

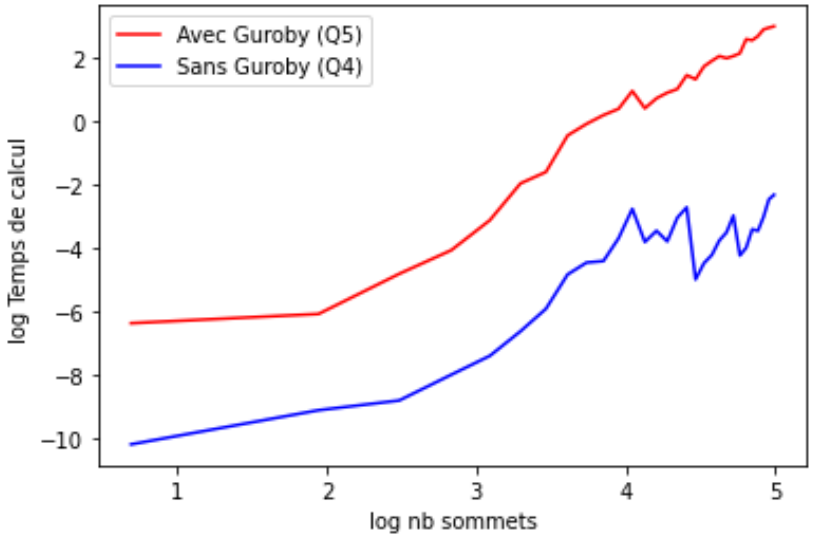
Lorsqu'on fait varier tfin (avec tdebut fixé pour faire varier tfin-tdebut), on voit que à partir d'une certaine valeur de tfin, le temps de calcul stagne. En effet, lorsque le temps de calcul stagne, cela

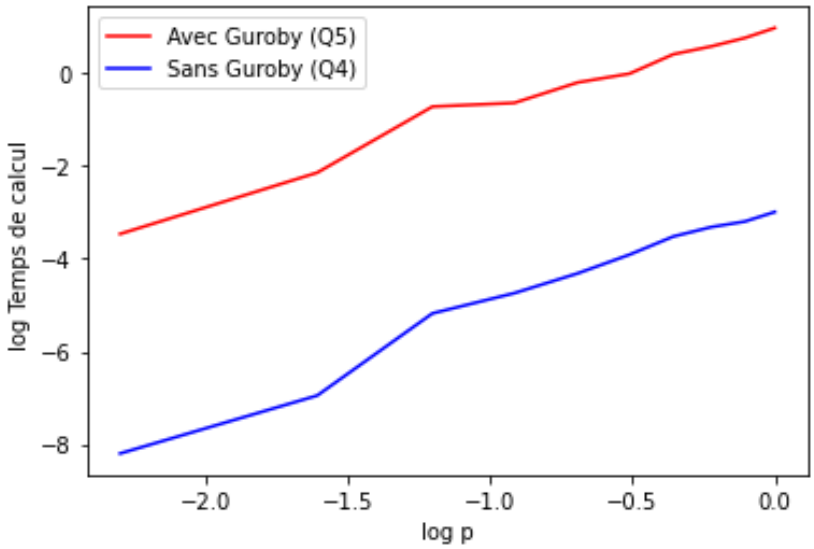
correspond à tfin qui est supérieur au nombre maximum de sommets possibles dans $\tilde{G}$ pour chaque sommet de G.

On peut en déduire que la complexité de notre algorithme utilisant gurobi est $O(n^3 * p^2)$ pour un graphe avec n sommets et m arêtes et un intervalle de temps [td,tf].

Question 7 :

On compare les temps d'exécution des algorithmes de la question 4 et 5 par rapport à la taille de l'entrée en échelle logarithmique:





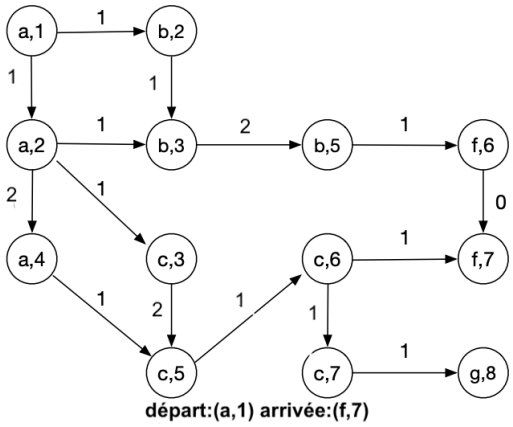
On voit que l'algorithme avec gurobi a un temps de calcul très supérieur a celui de l'algorithme sans gurobi. On voit aussi que l'évolution des temps de calcul de ces deux algorithmes en fonction p est similaire, car les 2 courbes associées sont parallèles.
Comme la complexité de l'algorithme avec gurobi est $O(n^3 * p^2)$, on en déduit que dans l'algorithme sans gurobi, le temps de calcul augmente proportionnellement à p^2.
Dans l'évolution des temps de calcul en fonction du log du nombre de sommets, les deux courbes sont différentes: avec l'algorithme sans gurobi, le temps de calcul stagne à partir de x=4, alors que l'algorithme avec gurobi a son temps de calcul qui continue d'augmenter.
Donc le temps de l'algorithme sans gurobi n'est pas proportionnel à n^3 contrairement à l'algorithme avec gurobi. Cela peut s'expliquer par le fait que l'algorithme sans gurobi est proportionnel à $min(m, (tf-$

$td))^2$, or lorsqu'on augmente le nombre de sommet, on augmente aussi le nombre d'arêtes avec notre fonction génératrice de multigraphe, or dans notre test, td et tf sont fixes, donc tf-td est fixe. Pour un graphe de petite taille (m petit et $m \leq (tf - td)$), le temps de calcul est proportionnel à m^2, et lorsque on a des graphes tels que $m \geq (tf - td)$, le temps de calcul est proportionnel à $(tf - td)^2$ qui est fixe, donc le temps de calcul stagne.
On a bien vérifié que dans l'algorithme de la question 4, le temps de calcul est proportionnel à $min(m, (tf - td))^2$ ce qui est bien cohérent avec la complexité que l'on a calculé qui est en $O(n^2 * min(m, (tf - td))^2)$.

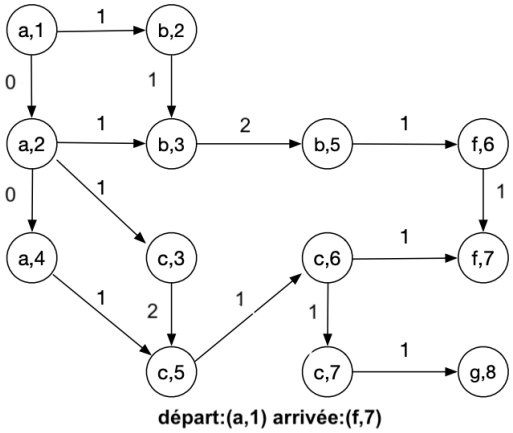
ANNEXE :

On prend les graphes modifiés à partir du graphe G de l'énoncé.
On a départ=a, arrivée=f, et [td,tf]=[1,8].

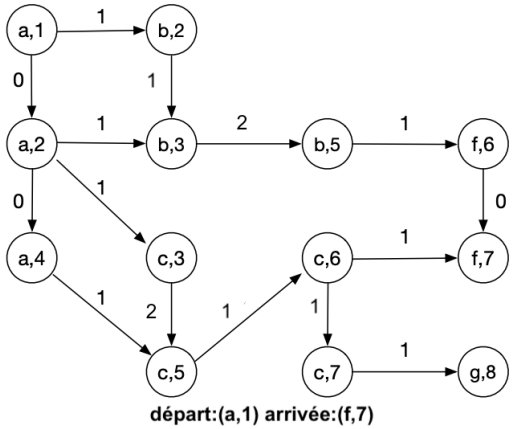
Chemin I



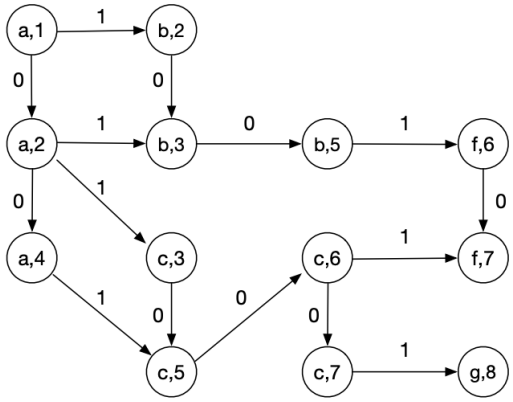
Chemin II



Chemin III



Chemin IV



Programme linéaire pour le chemin le plus court de a à f

Max a1,a2,a4,b2,b3,b5,c3,c5,c6,c7,f6,f7,g8

a1≤0
b2-a1≤1
a2-a1≤0
a4-a2≤0
b3-a2≤1
b3-b2≤0
b5-b3≤0
c3-a2≤1
c5-a4≤1
c5-c3≤0
c6-c5≤0
c7-c6≤0
g8-c7≤1
f7-c6≤1
f6-b5≤1
f7-f6≤0

0≤ a1,a2,a4,b2,b3,b5,c3,c5,c§,c7,f6,f7,g8 ≤ 9