Exercice 1. Cheminement. (5 points)
1. On considère le graphe G=(X,U) donné ci-dessous. Sur chaque arc u nous avons associé sa valeur v(u).

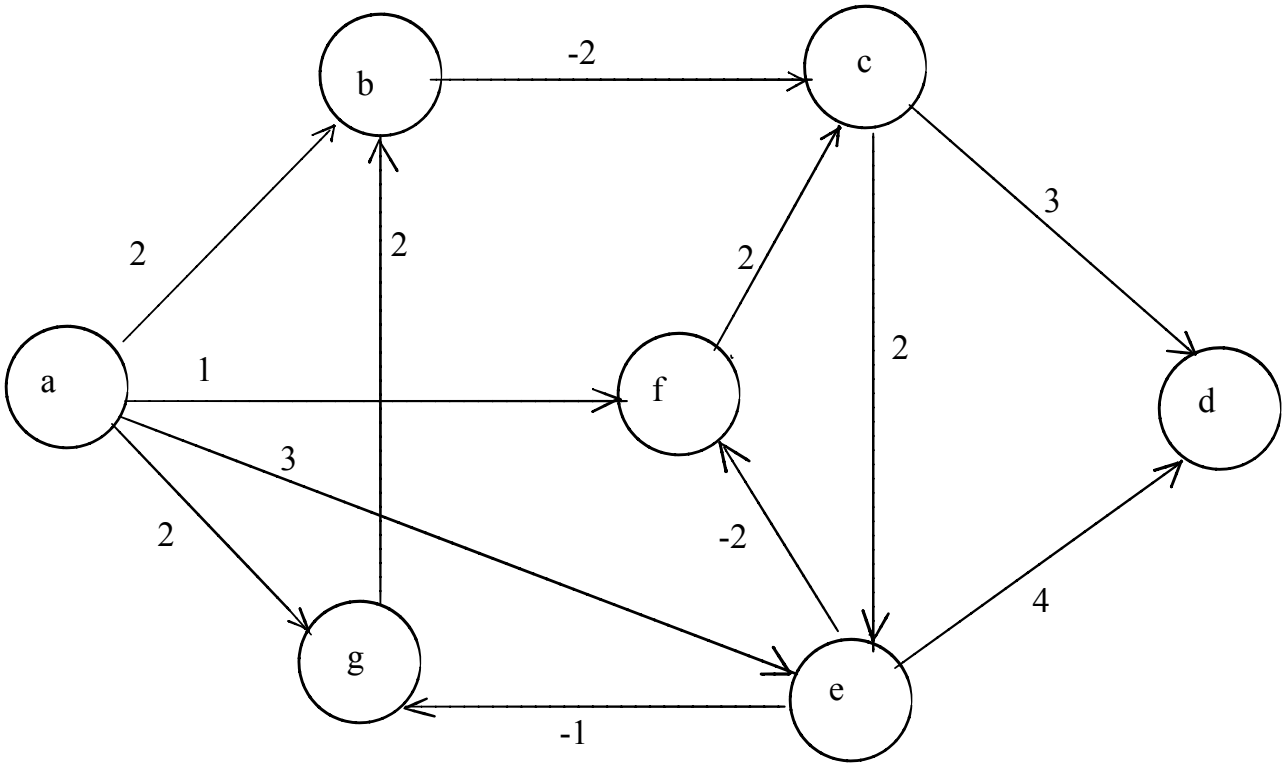


Fig. 1. Graphe G=(X,U)

Q1. Démontrer que le graphe admet des chemins de plus petite valeur du sommet a à tout autre sommet. Indication : utilisez le résultat d'un théorème vu en cours. Quel algorithme proposez vous pour calculer les chemins de valuation minimale du sommet a à tout autre sommet. Il n'est pas demandé d'appliquer l'algorithme. Indication : il y a deux circuits élémentaires.

Q2. Dans le tableau ci-dessous les valeurs Lambda(x) donnent les valeur des chemins de plus petite valuation de a à tout autre sommet dans le graphe G. Utiliser cette information pour déduire les arcs appartenant à l'arbre des chemins de plus petite valuation de a à tout autre sommet. Donner ces arcs. Justifier votre réponse.

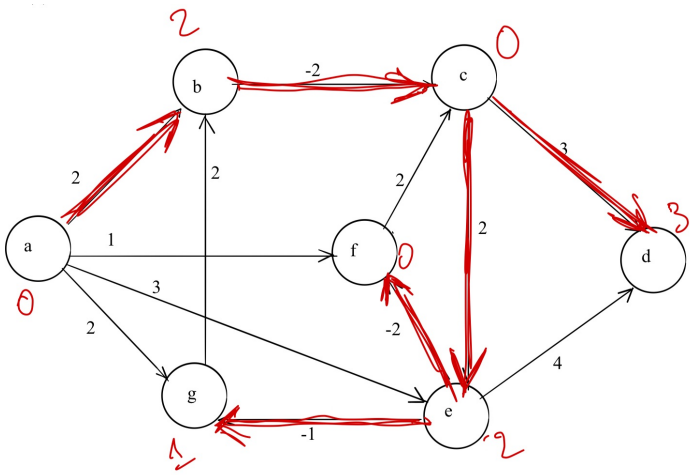
X	a	b	c	d	e	f	g
Lambda(X)	0	2	0	3	2	0	1

Q3. Y-a t il unicité des chemins ? Justifier.

Réponse Q1: Selon le Lemme 2 (page 55), l'existence des chemins est garantie en cas de non présence de circuits absorbants. Dans notre exemple il existe seulement deux circuits mais aucun des deux n'est pas absorbants. On ne peut utiliser que l'algorithme de Ford à cause de l'existence des circuits, qui disqualifie l'algorithme de Bellman, et des arcs de valeurs negatives qui interdit l'algorithme de Dijkstra.

Reponse Q2: Les arcs de l'arbre de PCC: (a,b), (b,c), (cd), (c,e), (e,f), (eg). Ces arcs sont calculés comme correspondant à ceux tel que lambda(j)-lambda(i) = v(i,j) (voir corollaire du théorème 1).

reponse Q3: Il y a unicité car les arcs appartenant à l'arbre ce sont les seuls ayant la propriété lambda(j)-lambda(i) = v(i,j) et ils ne permettent pas de construire plus qu'un chemin.



Exercice 2. Algorithme de Roy sur un probleme de matching. (3 points)
Considérons un problème d'affectation comme celui étudié au TD 13. Il y a trois programmeurs A, B et C qui doivent réaliser trois programmes 1, 2 et 3. Chacun doit exécuter un seul programme et tout programme doit être exécuté par un seul programmeur. La Fig. 2 donne les coûts de réalisation pour chaque affectation possible de chaque programmeur à un programme et qui est représenté par un arc.
On souhaite utiliser l'algorithme de Roy pour résoudre ce problème.

Question 1. Donner les arcs et les capacités qu'il faudra rajouter sur la figure suivante pour obtenir un réseau de transport permettant de calculer une affectation de coût minimal. Appliquer l'algorithme de Roy. Rapporter les itérations successives, le couplage de coût minimum obtenu ainsi que le coût de ce couplage.

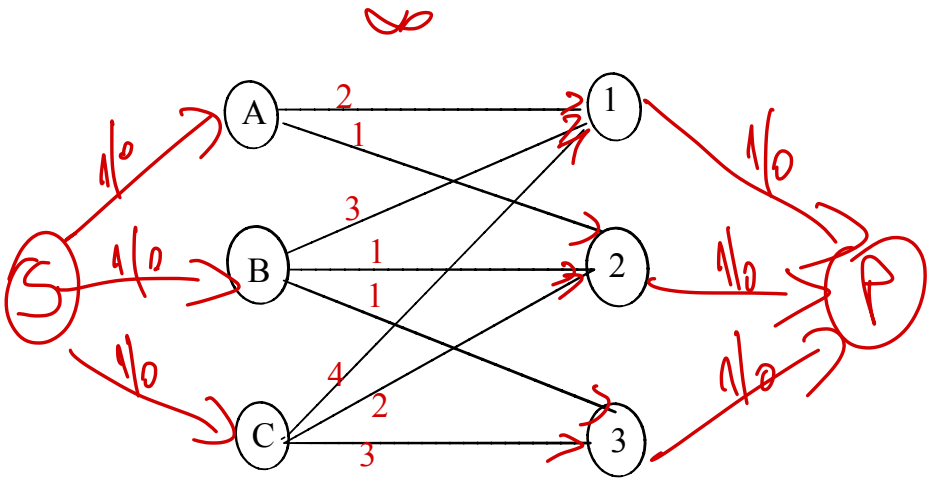


Fig. 2. Coûts des affectations

Solution :
Il faudra d'abord rajouter les sommets S et P et ensuite des arcs de capacités 1 depuis S vers A, B et C et ensuite des arcs de capacité 1 depuis 1,2, 3 vers P.
Tous ces arcs seraient de cout nul.
Ensuite, on peut appliquer l'algorithme de ROY:
Itération 1 : SA2P (flot 1, cout 1)
Itération 2 : SB3P (flot 1, cout 1),
Itération 3: SC2A1P (flot 1, cout 3)
Cout global d'affectation 5.
Affectation: A1, B3, C2.

Exercice 3. Graphe d'écart et flots canalisés (5 points)
On considère le réseau de transport ci-dessous. Soit F0 un flot compatible et Ge(F0) le graphe d'écart correspondant donné ci-dessous avec seulement les notations des coûts associés.

Question 1. Identifier les valeur de F0 sur chaque arc. (2 points)

Question 2. Est-ce que F0 est de coût minimum ? Justifier votre réponse. Si besoin calculer un flot compatible de coût minimum. On n'oubliera pas de prendre en compte l'arc de retour.

Reponse : Valeur des flots: sx1=4, sx2=1, sx3=1, x1x3=2, x1p=2, x2x3=1, x2x4=0, x3x4=2, x3p=2, x4p=2, ps=6.

Notation : Borne(i,j), Capacite(i,j) /(Coût (ij))

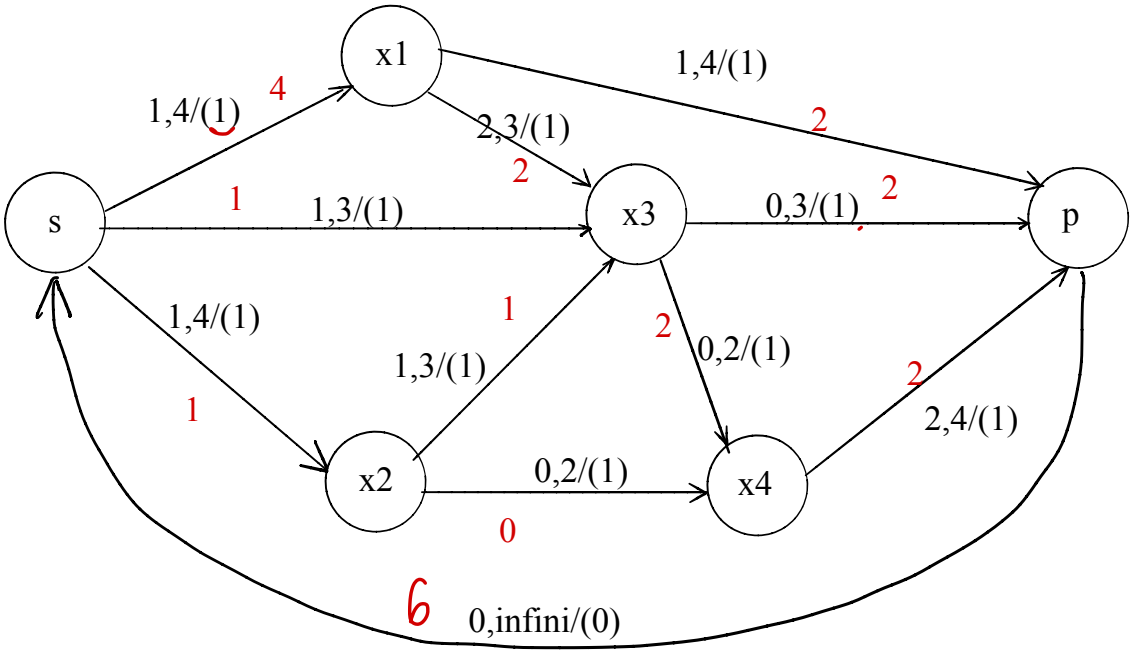
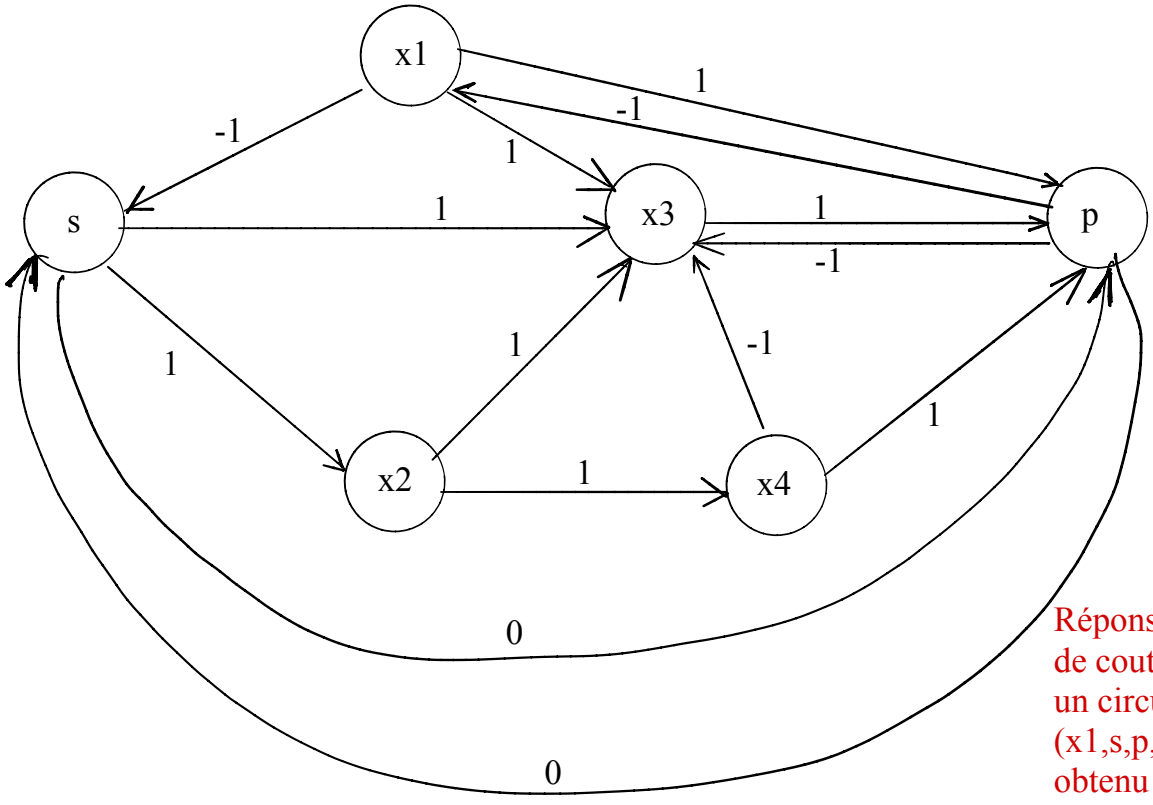


Fig. 3. Réseau de transport

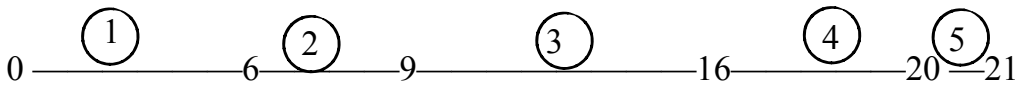


Réponse: le flot F0 n'est pas de cout minimum. Il existe un circuit de cout négatif : (x1,s,p,x1). Ensuite le flot obtenu est de cout min.

Fig. 4. Graphe d'écart Ge(F0) (valuation correspondant aux coûts)

Ordonnancement

Partie I. On considère le problème d'ordonnancement dans lequel il faudra déterminer un ordre d'exécution des tâches sur une machine en optimisant un critère. Les tâches s'exécutent alors dans cet ordre et consécutivement. Par exemple, étant donné un ensemble de n=5 tâches, de durées P1=6, P2=3, P3=7, P4=4, P5=1, et supposant que ces tâches s'exécutent dans l'ordre 1,2, 3, 4, 5, on a l'ordonnancement suivant:



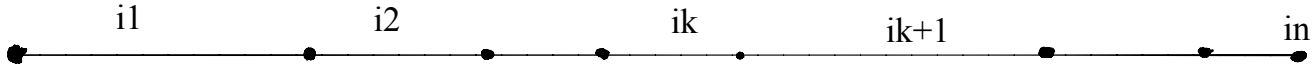
Si on note avec Ci la date d'achèvement de la tâche i, C1=6, C2=9, C3=16, C4=20, C5=21. Le critère que nous considérons ici est celui de minimisation de la somme des dates d'achèvement.

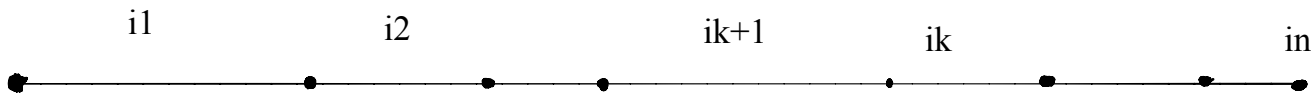
Plus généralement, si on a n tâches de durées Pi1, Pi2,...,Pin qui s'exécutent dans l'ordre i1,i2, ..., in, on a l'ordonnacement avec Cik= Pi1+ Pi2+,...,+Pik. Le critère est : minimiser la somme des Cik.

Q1. Justifier brièvement : Ci1 +Ci2 +...+Cin = n*Pi1+(n-1)*Pi2+...+Pin. Montrer qu'on obtient un ordonnancement optimal en appliquant la règle : « ordonner les tâches dans le sens des Pi croissants ». (2 points)
Réponse: la premiere partie est facile : il suffit de developper les valeurs des Ci en somme des Pi et de faire la somme. Ensuite, pour la deuxième partie une demonstration rigoureuse serait par l'absurde : tout autre ordonnancement qui ne respecte pas l'ordre croissant des pi, et donc contient Pi>Pj et i exécuté avant j est améliorable en les permutant.

Q2. Donner l'ordonnancement optimal pour l'exemple précédent et la valeur de la somme des Ci. Comparez au coût initial que vous calculerez. (1 point)
Réponse : P5, P2, P4, P1, P3; somme des Ci=48 versus 72.

Partie II. Minimisation du plus grand retard.
On associe en outre aux n tâches des dates d'échéances D1, D2, ..., Dn. On dit que la tâche i est en retard si Ci > Di, son retard Ti est alors Ci - Di. Par définition si Ci<= Di, alors son retard est nul. Donc, Ti = max(0, Ci-Di).
On veut minimiser Tmax = max Ti (i=1..n).
Q1. On considère un ordre quelconque i1, i2, ..., in et deux tâches consecutives ik, ik+1 dans cet ordre. On se propose de comparer cet ordre O et celui O' obtenu en permutant les deux taches consecutives ik et ik+1. On note avec T'i le retard de la tâche i dans l'ordre O'.





Comparer Ti et T'i pour i different de ik et ik+1. (1 point) réponse : les valeurs ne changent pas

Q2. On suppose que Dik > Dik+1. Comparer max(Tik, Tik+1) à max(T'ik, T'ik+1). (1 point)
Reponse : max(Tik, Tik+1) >= max(T'ik, T'ik+1) car max(Tik, Tik+1) = Tik+1>= max(T'ik, T'ik+1).
Q3. Déduire une règle générale et la justifier. (1 point)
Reponse: suivant la réponse à la question précédente, on déduit que la meilleure stratégie serait d'ordonner les tâches selon l'ordre croissant des Di.
Q4. Appliquer cette règle à l'exemple de la Partie I avec D1=12, D2=7, D3=21, D4=5 et D5=10.
Ca donne les taches ordonnées : 4, 2, 5, 1, 3, et un Tmax=2.