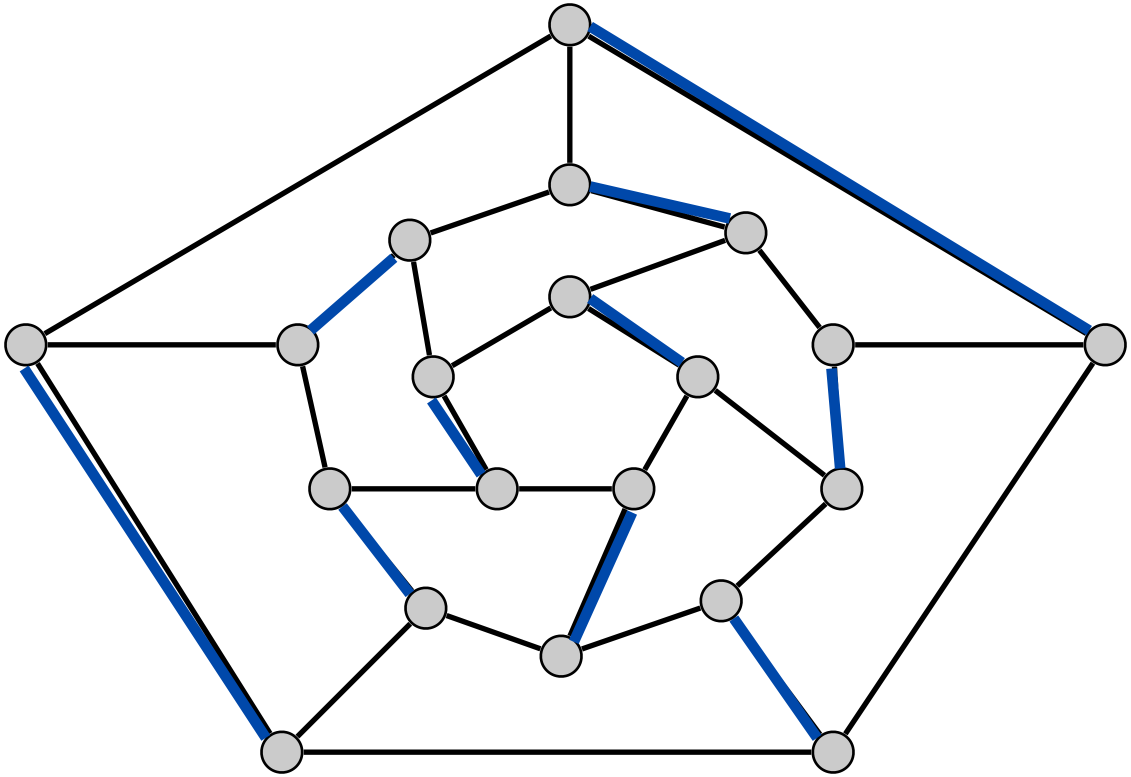
Le programme

1 et 2. Rappels sur les graphes
Plus courts chemins
Programmation dynamique
(2. et) 3. Problèmes de flots
4. Problèmes d'affectation et de transport
5. Compléments et révisions

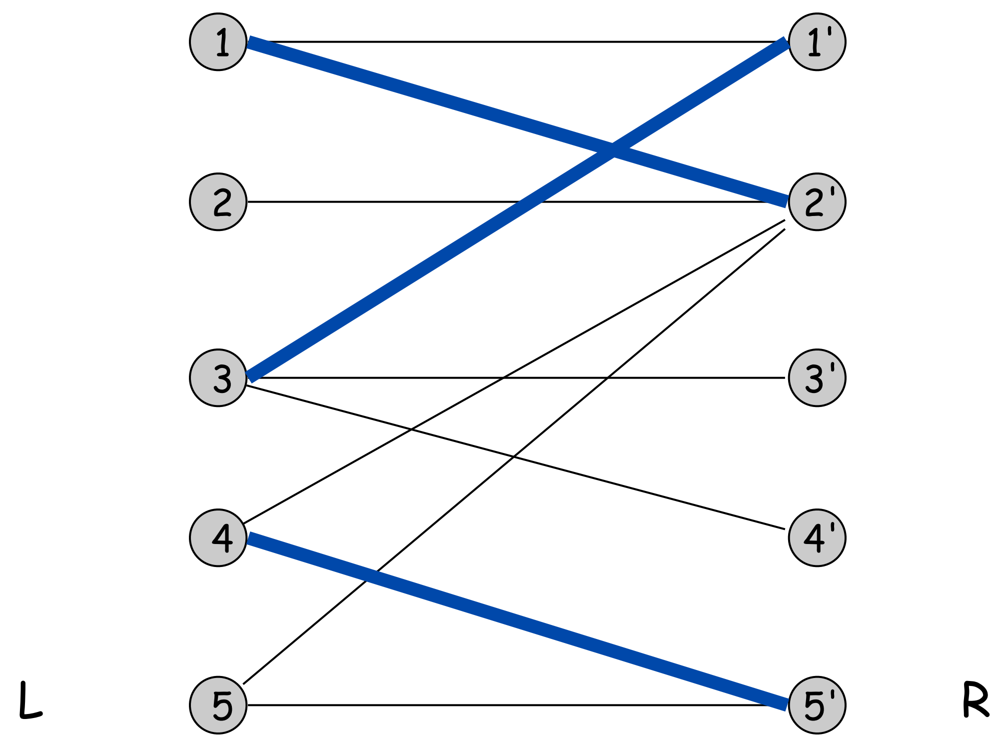
Couplages

- Entrée : graphe non-orienté $G = (V, E)$.
- $M \subseteq E$ est un couplage si chaque sommet apparaît à une arête au plus de M.
- Couplage Max : un couplage de cardinalité maximum.



Couplage biparti

- Entrée : graphe biparti non-orienté $G = (L \cup R, E)$.
- $M \subseteq E$ est un couplage si chaque sommet apparaît à au plus une arête de M.
- Couplage Max : un couplage de cardinalité maximum.

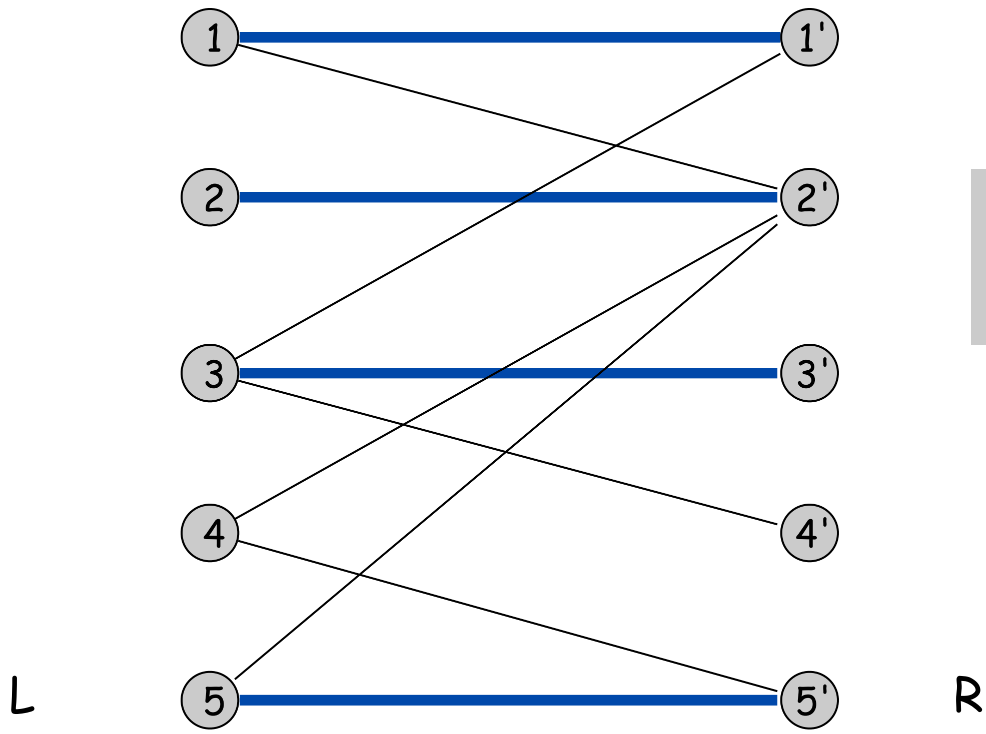


couplage

1-2', 3-1', 4-5'

Couplage biparti

- Entrée : graphe biparti non-orienté $G = (L \cup R, E)$.
- $M \subseteq E$ est un couplage si chaque sommet apparaît à une arête au plus de M.
- Couplage Max : déterminer un couplage de cardinalité maximum.



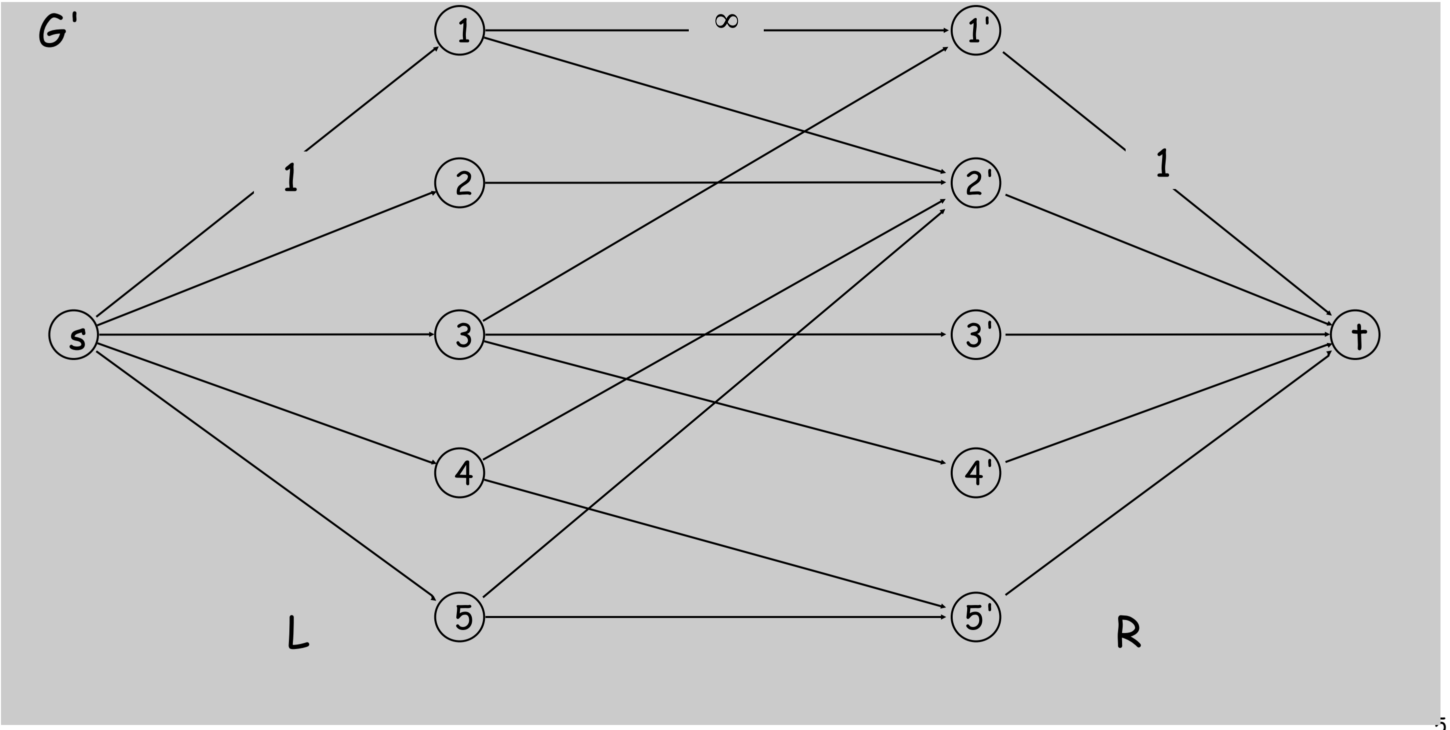
couplage max

1-1', 2-2', 3-3', 5-5'

Couplage biparti

Formulation Flot Max.

- Créer un graphe orienté $G' = (L \cup R \cup \{s, t\}, E')$.
- Orienter toutes les arêtes de L à R, et affecter une capacité infinie (ou unitaire).
- Ajouter s (source), et des arêtes de capacité 1 de s vers tout sommet de L.
- Ajouter t (puits), et des arêtes de capacité 1 de tout sommet de R vers t.

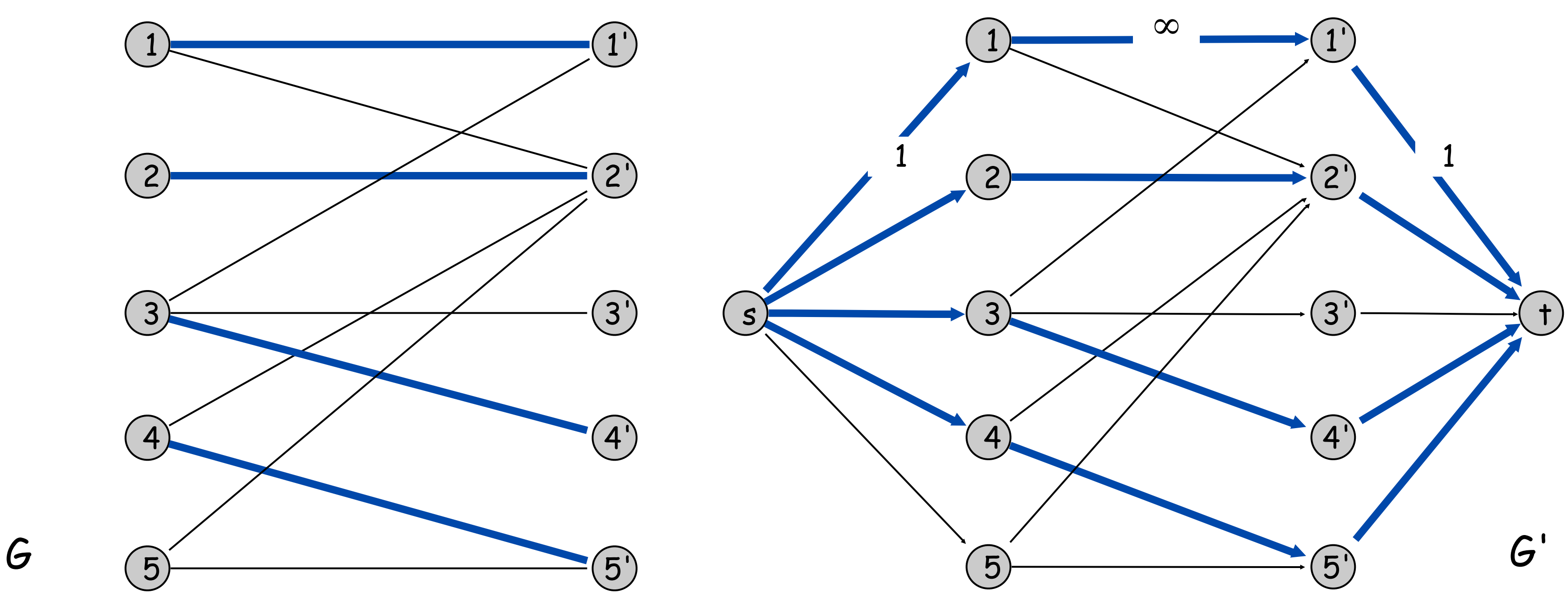


Couplage biparti

Théorème. Couplage de cardinalité max de G = valeur de flot max dans G'.

Idée de la preuve. ≤

- Etant donné un couplage max M de cardinalité k.
- Considérons un flot f qui envoie une 1 unité le long de chacun de k chemins.
- f est un flot, dont la valeur est k.



Couplage parfait

Déf. Un couplage $M \subseteq E$ est parfait si chaque sommet apparaît à exactement une arête de M.

Q. Quand un graphe biparti a un couplage parfait ?

Structure de graphes bipartis avec des couplages parfaits.

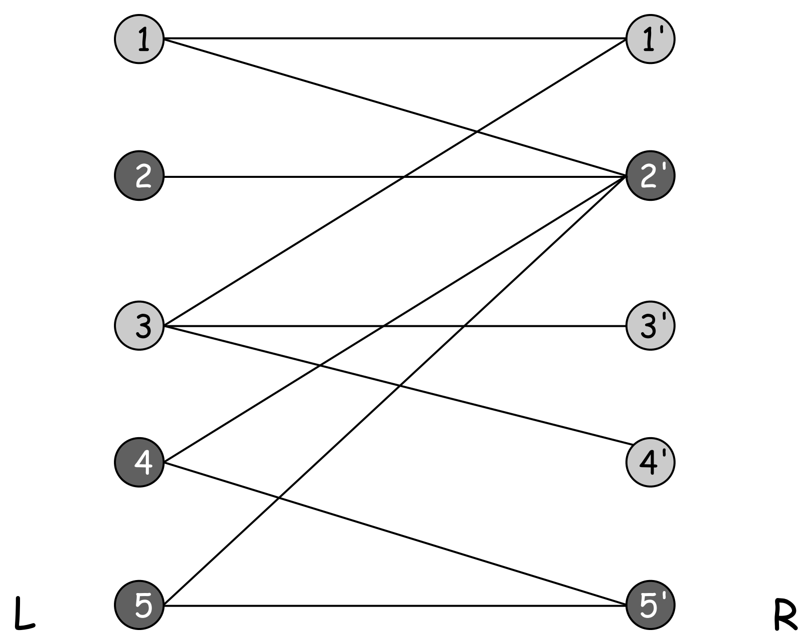
- $|L| = |R|$.
- Quelles autres conditions sont nécessaires ?
- Quelles conditions sont suffisantes ?

Couplage parfait

Notation. Soit S un sous-ensemble de sommets, et soit N(S) l'ensemble de sommets qui sont voisins de ceux de S.

Observation. Si un graphe biparti $G = (L \cup R, E)$ a un couplage parfait, alors $|N(S)| \geq |S|$ pour tous les sous-ensembles $S \subseteq L$.

Preuve. Chaque sommet de S doit être couplé avec un sommet différent de N(S).



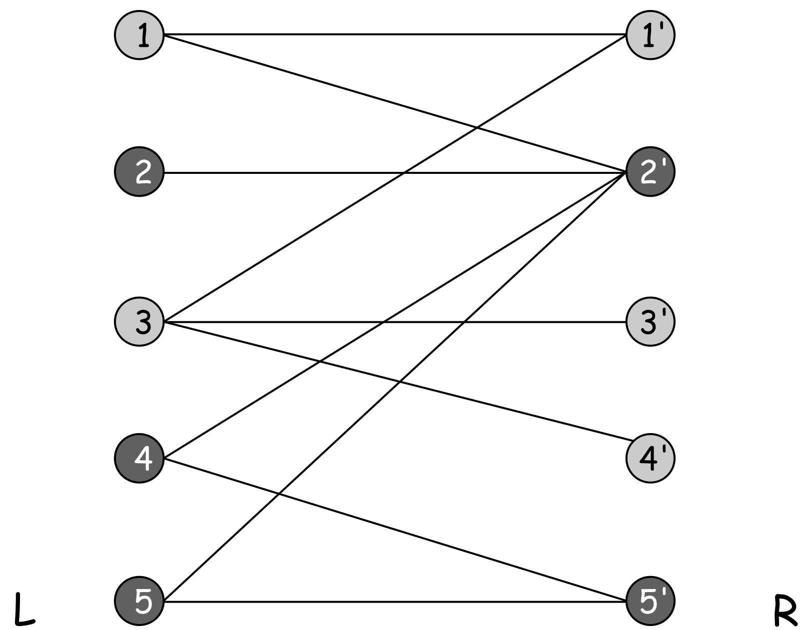
Pas de couplage parfait:

$S = \{ 2, 4, 5 \}$

$N(S) = \{ 2', 5' \}$.

Théorème du mariage. [Frobenius 1917, Hall 1935]

Soit $G = (L \cup R, E)$ un graphe biparti avec $|L| = |R|$. Alors, G a un couplage parfait ssi $|N(S)| \geq |S|$ pour tous les sous-ensembles $S \subseteq L$.



<u>Pas de couplage parfait :</u>

$S = \{ 2, 4, 5 \}$

$N(S) = \{ 2', 5' \}$.

Affectation et transport

I. Problème d'affectation
 1. Définition
 2. Méthode hongroise

II. Problème de transport
 1. Définition
 2. Méthode hongroise généralisée
 3. Algorithme de Busacker et Gowen

1. Définition

- Exemple : 5 tâches à réaliser, 5 agents. Chaque agent doit réaliser une et une seule tâche. L'agent j peut réaliser la tâche i en une durée d[i,j].
- But : affecter les tâches de manière à minimiser la durée totale

	1'	2'	3'	4'	5'
1	3	8	9	15	10
2	4	10	7	16	14
3	9	13	11	19	10
4	8	13	12	20	13
5	1	7	5	11	9

<u>Solution</u>

M = { 1-2' , 2-3' , 3-5' , 4-1' , 5-4' }

coût(M) = 8 + 7 + 10 + 8 + 11 = 44

Formulation par les graphes :
Entrée : graphe biparti complet pondéré, G = (L U R, E) avec |L| = |R|.
Objectif : déterminer un couplage parfait de poids minimum.

Applications

Applications naturelles.

- Coupler tâches et machines.
- Coupler personnel et tâches.
- ...

D'autres applications.

- Routage de véhicules.
- Traitement du signal.
- ...

2. Méthode hongroise

a. Cas simple.

0	6	0	10	15
4	0	7	6	11
0	20	2	16	17
7	5	11	0	8
12	9	12	0	0

- Coûts positifs ou nuls
- Il existe une affectation de coût 0

→ Cette affectation est optimale.

→ Deux questions

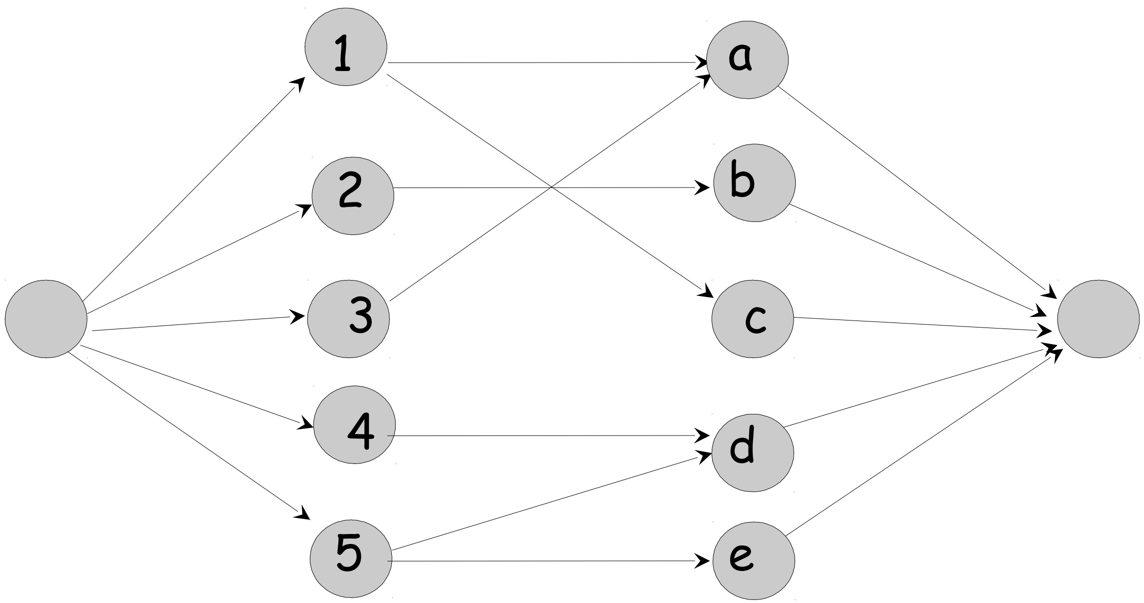
Q1 - Comment détecter ce cas « simple » ?

Q2 - Peut-on s'y ramener ?

2. Méthode hongroise

b. Question 1.

	a	b	c	d	e
1	0	6	0	10	15
2	4	0	7	6	11
3	0	20	2	16	17
4	7	5	11	0	8
5	12	9	12	0	0



Graphe (biparti) : arête (i,j) si la paire (i,j) est de coût 0
→ Cas simple si et seulement s'il existe un couplage parfait dans ce graphe

→ Résolution par un flot maximum

2. Méthode hongroise

c. Question 2.

c(x, y)

2	8	4	12	18
7	3	12	9	15
0	20	4	16	18
8	6	14	1	10
15	12	17	3	4
			1	

soustraire 1 de la colonne 4

→

2	8	4	11	18
7	3	12	8	15
0	20	4	15	18
8	6	14	0	10
15	12	17	2	4

Propriété : ajouter (ou soustraire) une constante à chaque entrée d'une ligne x ou d'une colonne y ne change pas le(s) couplage(s) parfait(s) de coût minimum.

→ Idée : créer des 0 dans les lignes/colonnes avec cette opération

2. Méthode hongroise

c. Question 2.

→ Etape 1 :

- A partir de C construire une matrice C' en soustrayant à chaque ligne son élément minimum
- A partir de C' construire une matrice C'' en soustrayant à chaque colonne son élément minimum

2	8	4	12	18	-2
7	3	12	9	15	-3
0	20	4	16	18	-0
8	6	14	1	10	-1
15	12	17	3	4	-3

0	6	2	10	16
4	0	9	6	12
0	20	4	16	18
7	5	13	0	9
12	9	14	0	1
		-2		-1

0	6	0	10	15
4	0	7	6	11
0	20	2	16	17
7	5	11	0	8
12	9	12	0	0

lignes

colonnes

2. Méthode hongroise

c. Question 2.

→ Etape 1 :

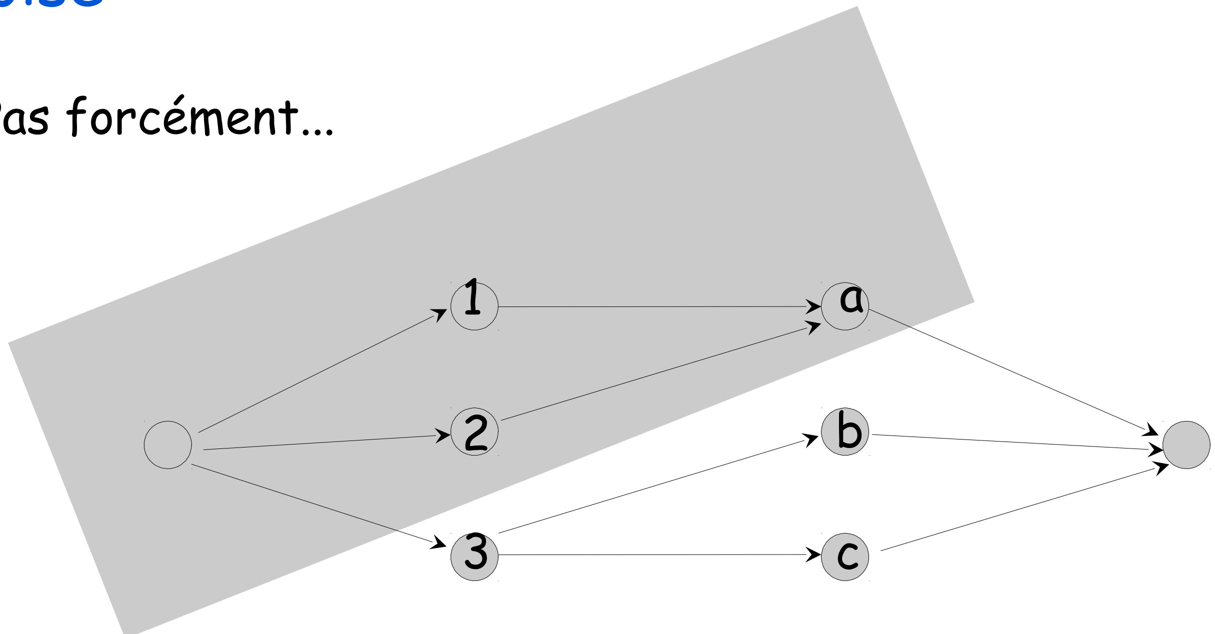
- A partir de C construire une matrice C' en soustrayant à chaque ligne son élément minimum
- A partir de C' construire une matrice C'' en soustrayant à chaque colonne son élément minimum

2. Méthode hongroise

c. Question 2.

→ Est-ce suffisant ? Pas forcément...

	a	b	c
1	0	8	4
2	0	3	12
3	1	0	0



→ Besoin d'ajouter un arc sortant de la coupe

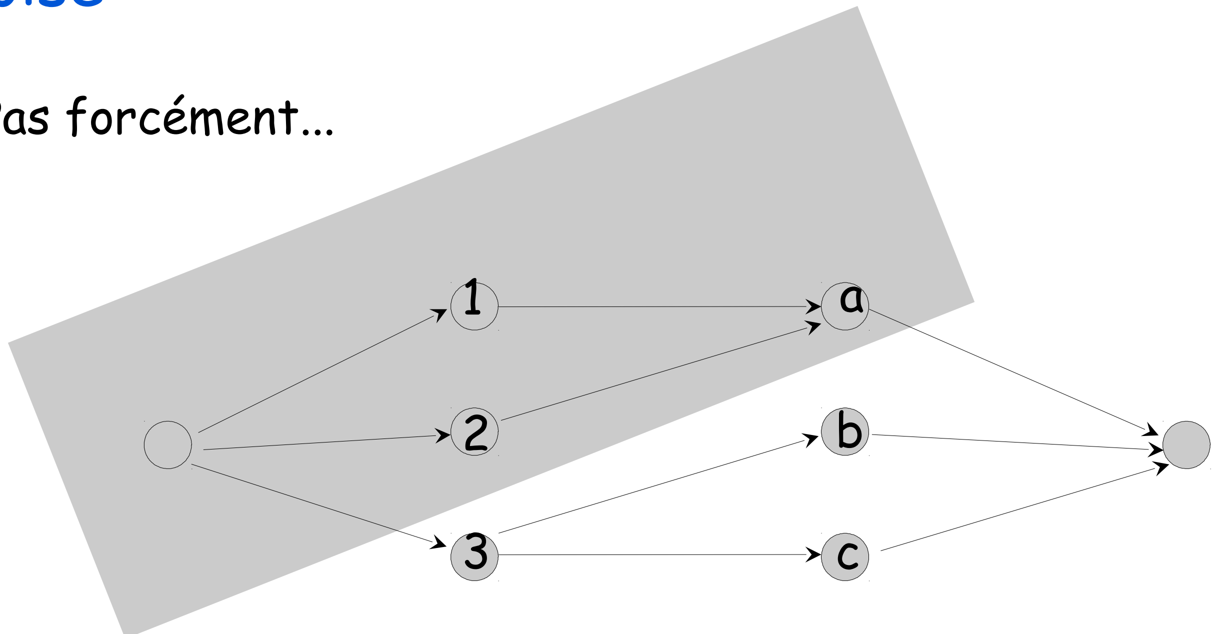
→ Un 0 dans le carré correspondant

2. Méthode hongroise

c. Question 2.

→ Est-ce suffisant ? Pas forcément...

	a	b	c
1	0	5	1
2	0	0	9
3	4	0	0



→ Besoin d'ajouter un arc sortant de la coupe

→ Un 0 dans le carré correspondant

→ Baisser de 3 dans le carré … sans créer d'éléments négatifs

→ - 3 sur les lignes marquées (1,2)
+ 3 sur les colonnes marquées (a)

→ Solution de coût 0 : (1,a), (2,b), (3,c)

2. Méthode hongroise
d. Algorithme

1. Faire l'étape 1 pour faire apparaître des 0

2. Chercher un flot maximum dans le « graphe des paires nulles » par l'algorithme de Ford et Fulkerson à partir du flot de l'itération précédente.
 - si flot max = n : le flot donne un couplage parfait, qui est une affectation optimale
 - sinon :
    - soit S_L et S_C les ensembles des sommets-lignes et sommets-colonnes marqués à la dernière étape
    - soit D le minimum des éléments à l'intersection d'une ligne marquée et d'une colonne non marquée
    - Faire -D sur les lignes de S_L
    - Faire +D sur les colonnes de S_C
    - Mettre à jour le graphe et itérer.

Exemple

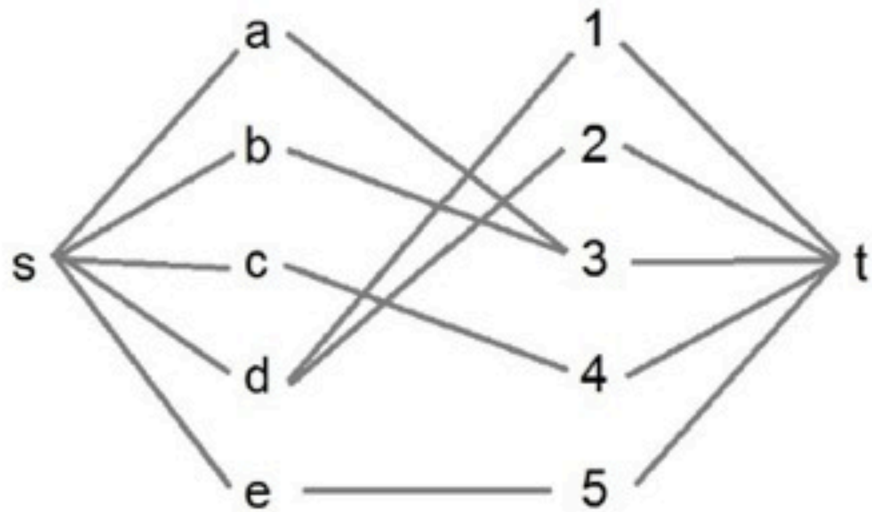
Un problème d'affectation
de 5 tâches $\{a, b, c, d, e\}$ à 5 machines $\{1, 2, 3, 4, 5\}$

t_{ij} temps pour effectuer la tâche i par la machine j

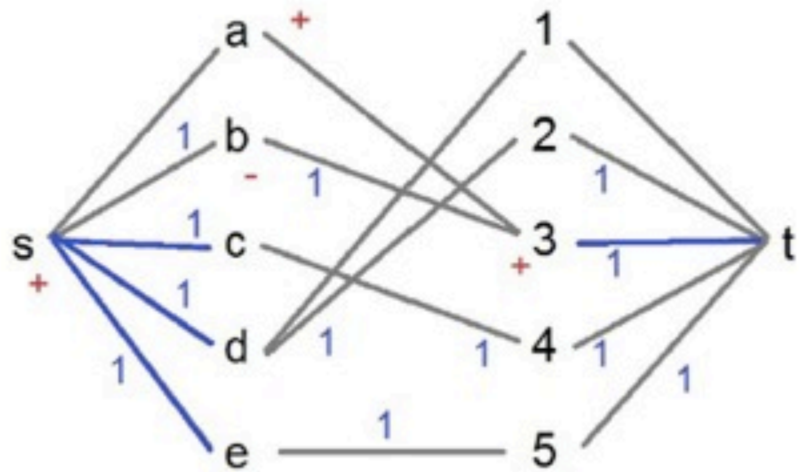
Durée minimale de fabrication qui nécessite d'effectuer les 5 tâches en série.

$$\begin{array}{c}\begin{pmatrix} 15 & 40 & 5 & 20 & 20 \\ 22 & 33 & 9 & 16 & 20 \\ 40 & 6 & 28 & 0 & 26 \\ 8 & 0 & 7 & 25 & 60 \\ 10 & 10 & 60 & 15 & 5 \end{pmatrix} \\ \begin{array}{ccccc} -8 & & -5 & & -5 \end{array}\end{array} \Rightarrow \begin{pmatrix} 7 & 40 & 0 & 20 & 15 \\ 14 & 33 & 4 & 16 & 15 \\ 32 & 6 & 23 & 0 & 21 \\ 0 & 0 & 2 & 25 & 55 \\ 2 & 10 & 55 & 15 & 0 \end{pmatrix}$$

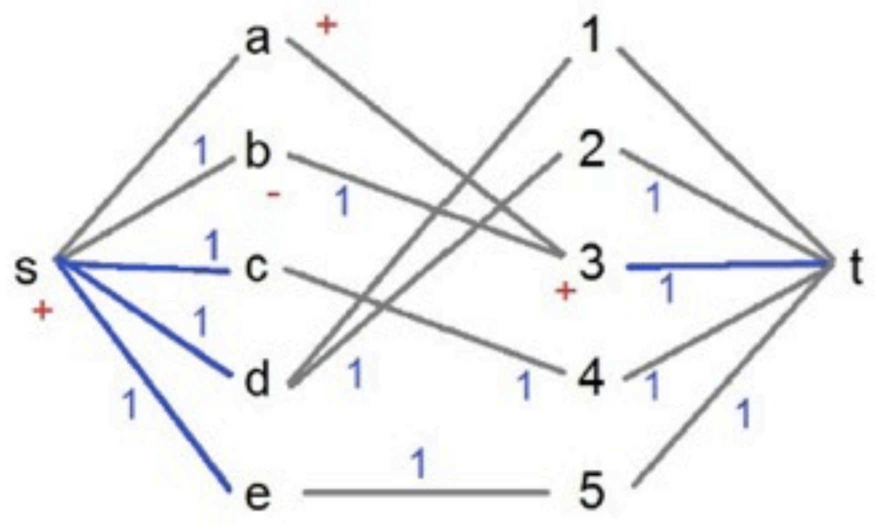
$$\begin{pmatrix} 7 & 40 & 0 & 20 & 15 \\ 14 & 33 & 4 & 16 & 15 \\ 32 & 6 & 23 & 0 & 21 \\ 0 & 0 & 2 & 25 & 55 \\ 2 & 10 & 55 & 15 & 0 \end{pmatrix} -4 \quad \Rightarrow \quad \begin{pmatrix} 7 & 40 & 0 & 20 & 15 \\ 10 & 29 & 0 & 12 & 11 \\ 32 & 6 & 23 & 0 & 21 \\ 0 & 0 & 2 & 25 & 55 \\ 2 & 10 & 55 & 15 & 0 \end{pmatrix}$$



$$\begin{pmatrix} 7 & 40 & 0 & 20 & 15 \\ 14 & 33 & 4 & 16 & 15 \\ 32 & 6 & 23 & 0 & 21 \\ 0 & 0 & 2 & 25 & 55 \\ 2 & 10 & 55 & 15 & 0 \end{pmatrix} -4 \quad \Rightarrow \quad \begin{pmatrix} 7 & 40 & 0 & 20 & 15 \\ 10 & 29 & 0 & 12 & 11 \\ 32 & 6 & 23 & 0 & 21 \\ 0 & 0 & 2 & 25 & 55 \\ 2 & 10 & 55 & 15 & 0 \end{pmatrix}$$



$$V(\varphi) = 4 < 5$$



- La coupe minimum donne une couverture minimale des 0 de la matrice, ici par les sommets $\{c, d, e, 3\}$
- Pour augmenter le flot, on cherche augmenter la capacité de la coupe en insérant de nouvelles liaisons parmi celles qui lient un sommet (tâche) marqué à un sommet (machine) non-marqué.
- Dans la matrice on raye donc les lignes des sommets non-marqués $\{c, d, e\}$ et colonnes des sommets marqués $\{3\}$.
- Ce faisant, on raye tous les 0 du tableau (qui sont justement couverts par les lignes $\{c, d, e\}$ et la colonne 3) et il ne reste que des quantités strictement positives.

$$\begin{array}{cc} \begin{matrix} & & +7 & & \end{matrix} & \\ \begin{pmatrix} 7 & 40 & 0 & 20 & 15 \\ 10 & 29 & 0 & 12 & 11 \\ 32 & 6 & 23 & 0 & 21 \\ 0 & 0 & 2 & 25 & 55 \\ 2 & 10 & 55 & 15 & 0 \end{pmatrix} & \begin{matrix} -7 \\ -7 \\ \\ \\ \\ \end{matrix} \end{array}$$

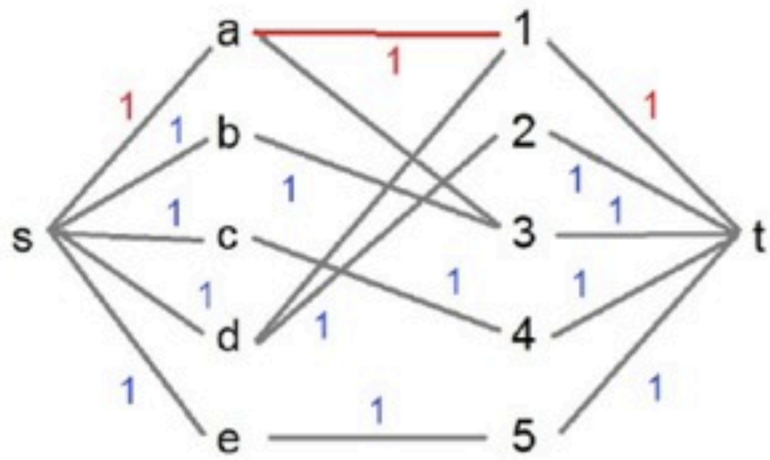
On retranche 7 à tout le tableau non rayé pour faire apparaître un zéro sans en faire disparaître. Pour cela on supprime 7 dans toute le tableau et on l'ajoute à toute rangée (ligne ou colonne) de la couverture minimale (pour préserver les 0 des étapes précédentes)

$$\begin{pmatrix} \mathbf{0} & 33 & 0 & 13 & 8 \\ 3 & 22 & 0 & 5 & 4 \\ 32 & 6 & 30 & 0 & 21 \\ 0 & 0 & 9 & 25 & 55 \\ 2 & 10 & 62 & 15 & 0 \end{pmatrix}$$

Il apparaît au moins un nouveau 0

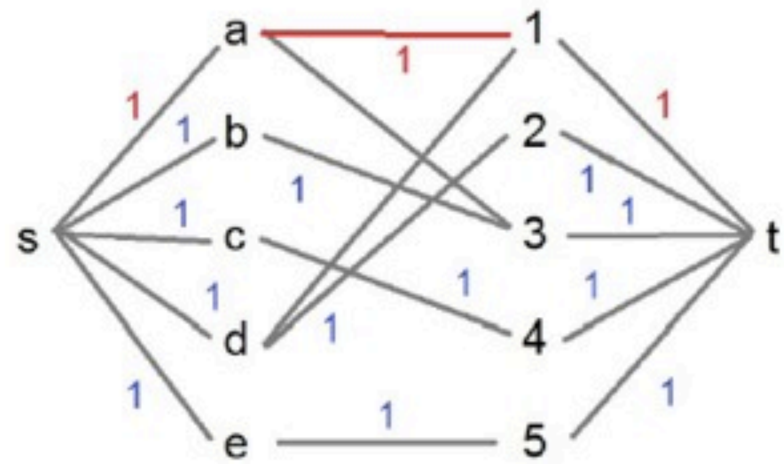
$$\begin{pmatrix} 0 & 33 & 0 & 13 & 8 \\ 3 & 22 & 0 & 5 & 4 \\ 32 & 6 & 30 & 0 & 21 \\ 0 & 0 & 9 & 25 & 55 \\ 2 & 10 & 62 & 15 & 0 \end{pmatrix}$$

Le nouveau 0 nous permet d'ajouter un arc. On obtient le réseau suivant :



$$\begin{pmatrix} \boxed{0} & 33 & 0 & 13 & 8 \\ 3 & 22 & \boxed{0} & 5 & 4 \\ 32 & 6 & 30 & \boxed{0} & 21 \\ 0 & \boxed{0} & 9 & 25 & 55 \\ 2 & 10 & 62 & 15 & \boxed{0} \end{pmatrix}$$

Le nouveau 0 nous permet d'ajouter un arc. On obtient le réseau suivant :



Solution : $a \to 1, b \to 3, c \to 4, d \to 2 e \to 5$

2. Méthode hongroise

UE MOGPL.
M1 Informatique.

Examen du 19 janvier 2018. Durée : 2 heures

Une feuille recto-verso est autorisée, tout autre document est interdit.
Téléphones portables éteints et rangés dans vos sacs.

Le barème est indicatif et est susceptible d'être modifié.

Exercice 1 (6 points)

Question 1 (4/6) — On considère le problème d'affectation (à coût minimum) suivant.

	a	b	c	d	e
A	4	1	7	0	2
B	8	6	3	0	4
C	7	0	7	4	2
D	9	0	7	6	0
E	9	4	8	6	0

Appliquer l'algorithme hongrois pour résoudre le problème. Vous spécifierez à chaque étape : la matrice, un flot maximum (sans spécifier comment vous l'avez trouvé) et les sommets lignes et colonnes marqués, ainsi que les opérations sur les lignes et les colonnes.

2. Méthode hongroise

d. Algorithme

Théorème : l'algorithme hongrois termine en $O(n^2)$ itérations.

Affectation et transport

I. Problème d'affectation
 1. Définition
 2. Méthode hongroise

II. Problème de transport
 1. Définition
 2. Méthode hongroise généralisée
 3. Algorithme de Busacker et Gowen

1. Définition

Généralisation de l'affectation au cas non unitaire

		Demandes		
		a 15	b 15	C 10
Ressources	A :10	3	4	10
	B :10	2	6	6
	C :20	7	6	4

Matrice : coût d'acheminement d'une unité de la ressource i vers la demande j
But : acheminement les ressources en satisfaisant les demandes et en minimisant le coût total d'acheminement

	a 15	b 15	C 10
A :10	0	10	0
B :10	5	5	0
C :20	10	0	10

Solution de valeur
10*4+5*2+5*6+10*7+10*4=190

1. Définition

Hypothèse simplificatrice : somme des ressources = somme des demandes

Demandes

Ressources	a 15	b 15	c 10
A :10	3	4	10
B :10	2	6	6
C :20	7	6	4

→ Il est toujours possible de satisfaire les demandes
→ Dans toute solution réalisable : toutes les ressources sont utilisées (et les demandes satisfaites à l'égalité)

	a 15	b 15	c 10
A :10	0	10	0
B :10	5	5	0
C :20	10	0	10

1. Définition

Formulation par les graphes : cela revient à chercher un flot maximum de **coût total minimum**

Résolution : *généralisation de l'algorithme hongrois*

2. Algorithme hongrois généralisé

Validité de l'étape 1 ? (addition/soustraction par ligne/colonne)

Hypothèse simplificatrice → Dans toute solution réalisable : toutes les ressources sont utilisées (et les demandes satisfaites à l'égalité)

		Demandes			
		a 15	b 15	c 10	
Ressources	A :10	3	4	10	-3
	B :10	2	6	6	-2
	C :20	7	6	4	-4

0	1	7
0	4	4
3	2	0
	-1	

0	0	7
0	3	4
3	1	0

2. Algorithme hongrois généralisé

Etape 2 : calculer un flot maximum dans le « réseau des coûts 0 ».

→ Si on a un flot de valeur = somme des demandes : on a une solution de coût 0, donc optimale.

→ Sinon : on fait exactement comme dans le cas de l'affectation pour faire apparaître un 0 supplémentaire dans une 'coupe min'.

Demandes

Ressources		a 15	b 15	C 10	
	A :10	0	0	7	
	B :10	0	3	4	
	C :20	3	1	0	-1
				+1	

0	0	8
0	3	5
2	0	0

Solution optimale (coût 155)

5	5	0
10	0	0
0	10	10

3. Algorithme de Busacker et Gowen

Idée : reproduire l'algorithme de Ford et Fulkerson, mais en prenant à chaque étape le chemin augmentant **de coût minimum**
→ On achemine une (ou plusieurs) unité(s) de flot avec un coût unitaire minimum.
→ On itère jusqu'à obtenir un flot maximum. On peut montrer que le coût total est minimum (validité de l'algorithme).

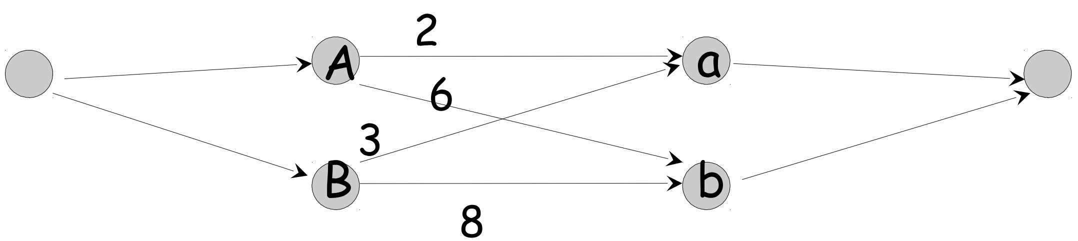
Demandes

Ressources	a 15	b 15
A :10	2	6
B :20	3	8

« Flot » initial

0	0
0	0

Graphe écart avec coûts



3. Algorithme de Busacker et Gowen

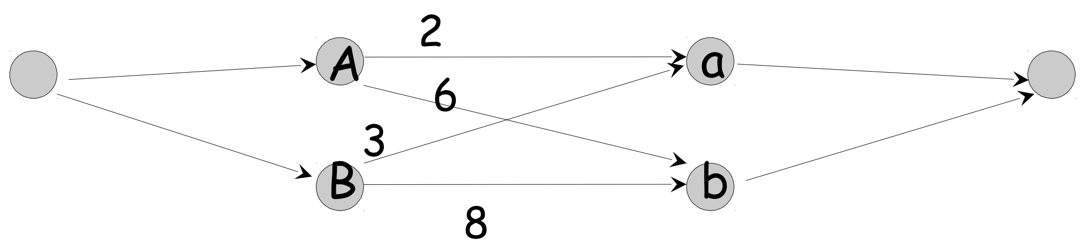
Demandes

Ressources	a 15	b 15
A :10	2	6
B :20	3	8

« Flot » initial

0	0
0	0

Graphe écart avec coûts

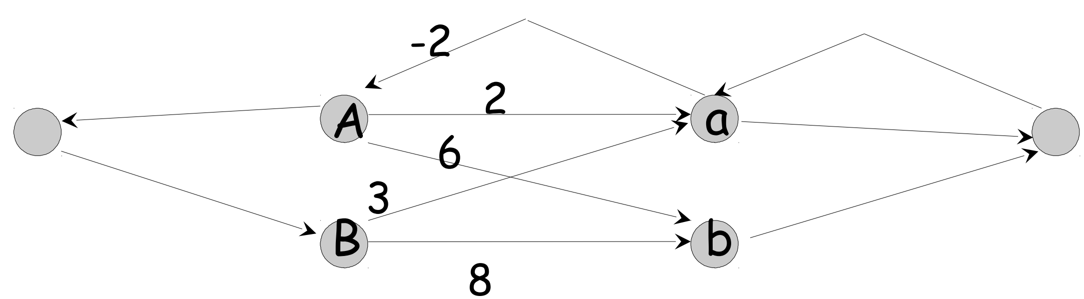


Chemin coût min : (s,A,a,t). Coût 2, cap 10

« Flot »

10	0
0	0

Graphe écart avec coûts



Chemin coût min : (s,B,a,t). Coût 3, cap 5. [...]

Marquage de Ford & Fulkerson

ou comment déterminer un chemin augmentant sans utiliser le graphe d'écart

marquer s **d'un** $+$

    répéter

        si il existe $e = (u, v)$**:** u **marqué,** v **non marqué et** $f(e) < c(e)$

            alors marquer v **d'un** $+$; **père**$(v) \leftarrow u$

        sinon

            si il existe $e = (u, v)$**:** v **marqué,** u **non marqué et** $0 < f(e)$

            alors marquer u **d'un** $-$; **père**$(u) \leftarrow v$

    jusqu'à ce que ($\nexists$ **plus de marquage possible) ou (**t **est marqué)**

Marquage de Ford & Fulkerson

ou comment déterminer un chemin augmentant sans utiliser le graphe d'écart

marquer s **d'un** $+$

répéter

si il existe $e = (u, v)$**:** u **marqué,** v **non marqué et** $f(e) < c(e)$

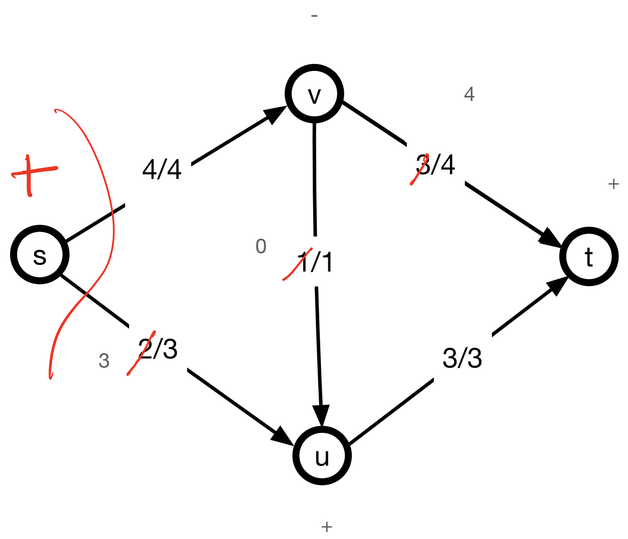
alors marquer v **d'un** $+$; **père**$(v) \leftarrow u$

sinon

si il existe $e = (u, v)$**:** v **marqué,** u **non marqué et** $0 < f(e)$

alors marquer u **d'un** $-$; **père**$(u) \leftarrow v$

jusqu'à ce que ($\nexists$ **plus de marquage possible) ou (**t **est marqué)**



Graphe

$s \underset{1}{-} u \underset{1}{-} v \underset{1}{-} t$

$\min\{3-2, 1, 4-3\} = 1$