eFrei
PARIS PANTHÉON-ASSAS UNIVERSITÉ

Projet de Recherche opérationnelle

Groupe B6

Problème de transport et minimisation de coûts

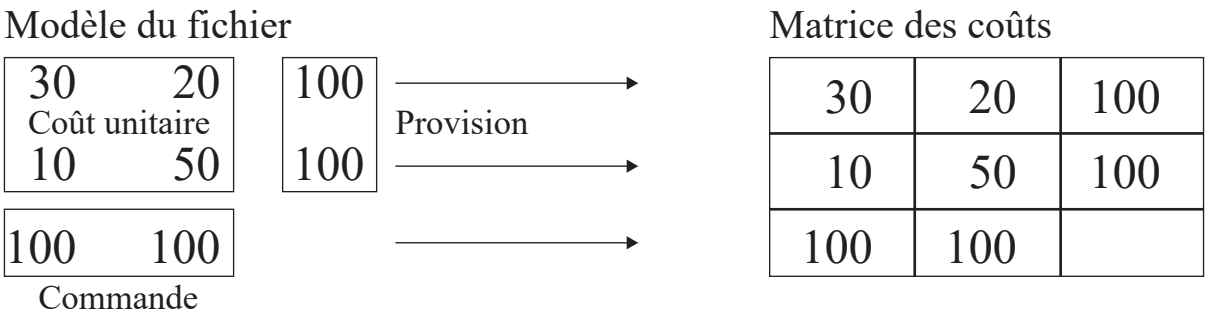
Pei LIU, Panying SONG, Nikola POPOVIC, Matthieu RAMANANTSALAMA

Plan

- Traitement de fichier
- Nord - Ouest
- Balas - Hammer
- Détéction de circuit
- Suppression de circuit
- Connexité
- Coûts potentiels | Coûts marginaux
  Détéction de valeur négative (Maximisation)

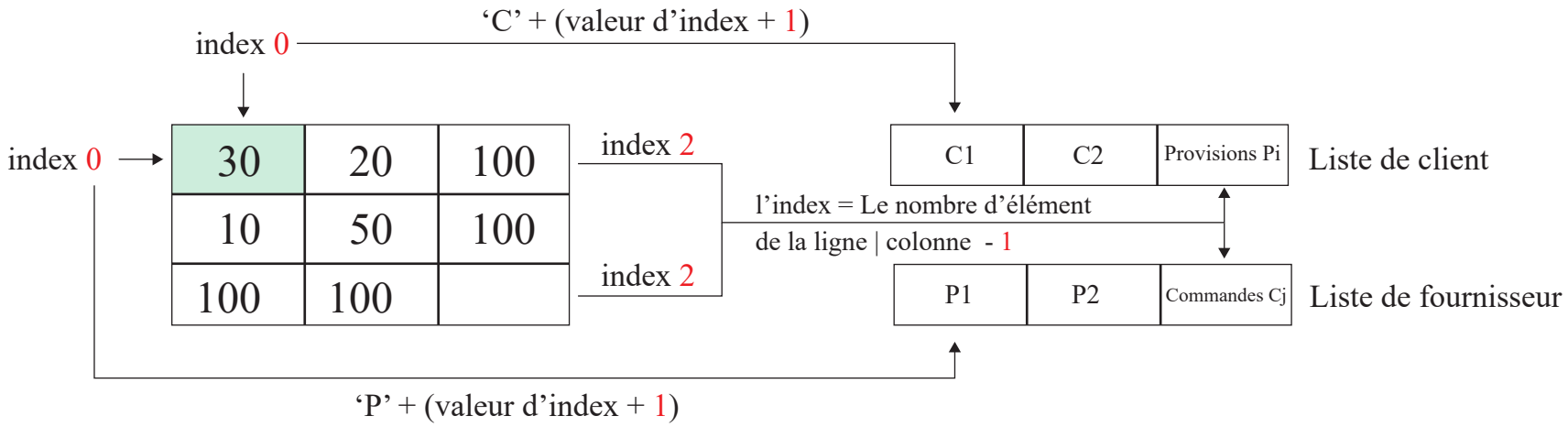
Lecture de fichier et regroupement des données

1. Lecture du fichier ligne par ligne



2. Créer la liste des fournisseurs et celle des clients à partir de la matrice des coûts

Parcourir la matrice de gauche à droite et de haut en bas



Nord - Ouest

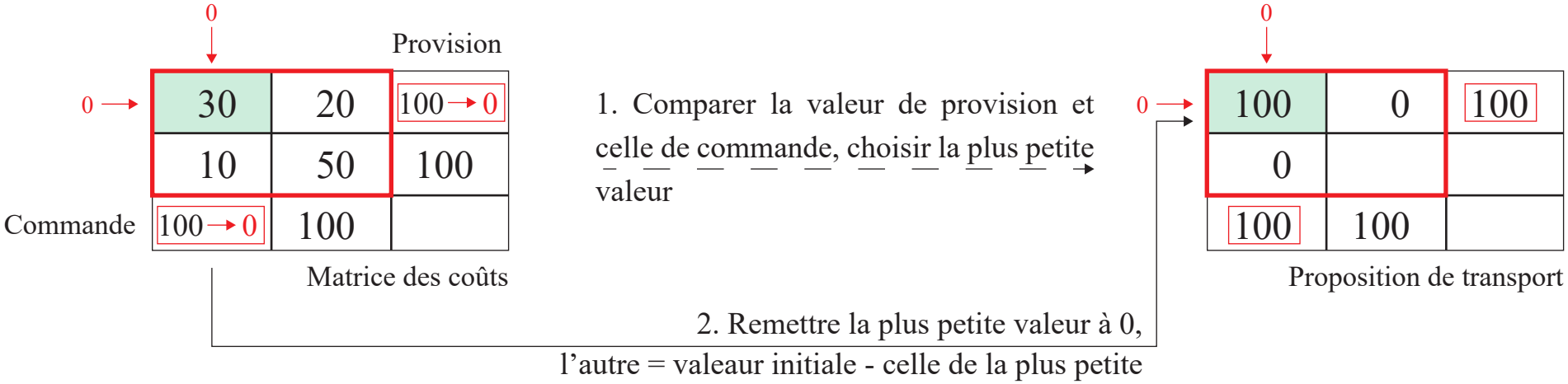
Nombre de coûts = (nombre de ligne - 1) * (nombre de colonne - 1)

Index de départ de linge = 0

Index de départ de colonne = 0

Case traitée = 0

1ère itération



3. Si provision > commande, parcourir la colonne pour mettre toutes les cases à 0, au contraire, mettre toute la ligne à 0. En cas d'égalité, cela sera les deux

Index de départ de linge + 1 (On fait un déplacement à droite pour le prochain départ)

Index de départ de colonne + 1 (On fait un déplacement en bas pour le prochain départ)

Case traitée = 1 + 1 + 1 = 3

Case traitée = Nombre de coûts Stop !

Balas - Hammer

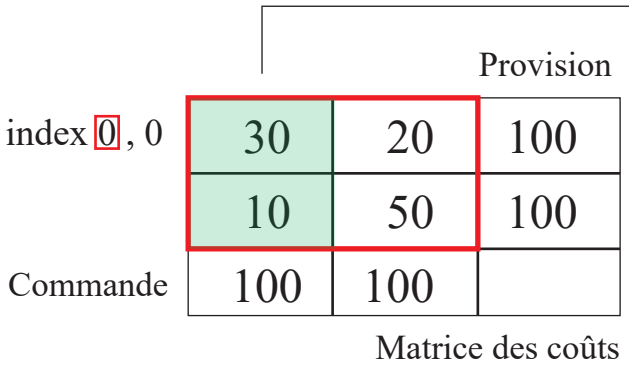
1. Trouver la plus grande pénalité dans l'ensemble des colonnes

temp = [ ] (La liste pour stocker temporairement toutes les valeurs de chaque colonne)

max_val_colonne = 0 (La plus grande pénalité dans l'ensemble des colonnes)

index_colonne = 0 (L'index pour localiser la colonne qui a la plus grande pénalité)

1.1 Parcourir toutes les colonnes de la matrice



1.2 Mettre toutes les valeurs de la colonne parcourue dans la liste

1.3 Remettre la liste en ordre croissant

Si max_val_colonne < temp[1] - temp[0], max_val_colonne = temp[1] - temp[0]

index_colonne = index de colonne parcourue

Sinon max_val_colonne et index_colonne ne changent pas

1.4 Vider la liste temp, et parcourir la prochaine colonne

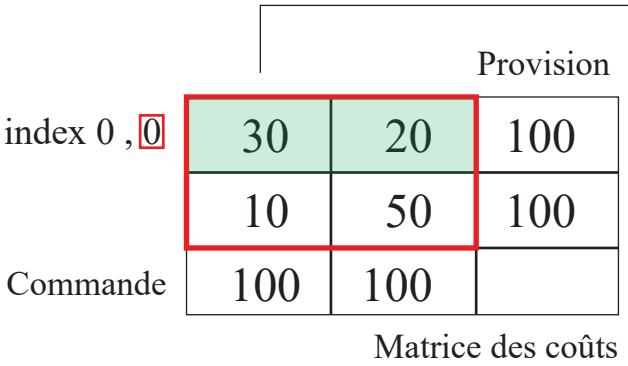
2. Trouver la plus grande pénalité dans l'ensemble des lignes

temp = [ ] (La liste pour stocker temporairement toutes les valeurs de chaque ligne)

max_val_ligne = 0 (La plus grande pénalité dans l'ensemble des ligne)

index_ligne = 0 (L'index pour localiser la ligne qui a la plus grande pénalité)

2.1 Parcourir toutes les lignes de la matrice



2.2 Mettre toutes les valeurs de la ligne parcourue dans la liste

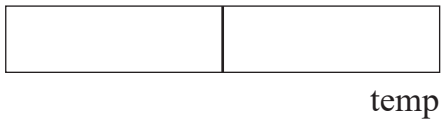
2.3 Remettre la liste en ordre croissant

Si max_val_ligne < temp[1] - temp[0], max_val_ligne = temp[1] - temp[0]

index_ligne = index de ligne parcourue

Sinon max_val_ligne et index_ligne ne changent pas

2.4 Vider la liste temp, et parcourir la prochaine ligne

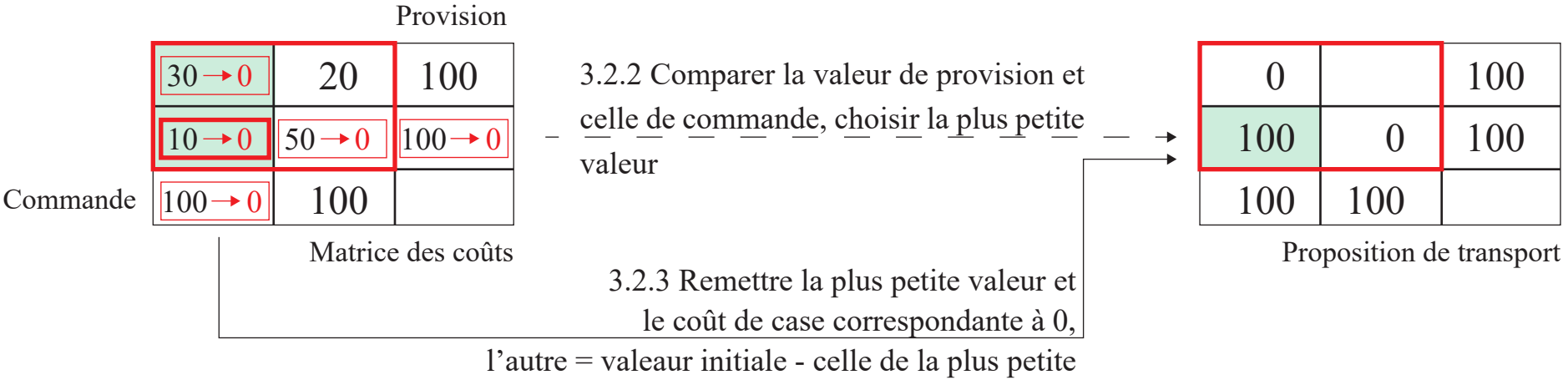


3. Trouver la proposition de transport

3.1 Comparer la plus grande pénalité de colonne avec celle de ligne

• Supporsons max_val_colonne >= max_val_ligne

3.2.1 Parcourir la colonne correspondante avec index_colonne, et trouver la plus petite valeur



3.2.4 Si provision > commande, parcourir la colonne pour mettre toutes les cases à 0,
au contraire, mettre toute la ligne à 0. En cas d'égalité, cela sera les deux

• Supposons max_val_colonne < max_val_ligne

Cela sera les même démarches, mais sur la ligne

- — -

Case traitée = Nombre de coûts Stop !

Marche - Pied

1. Transformer la matrice en graphe

1.1 Définir la structure de sommet

Sommet

Nom de sommet
Valeur de sommet
Liste des arêtes

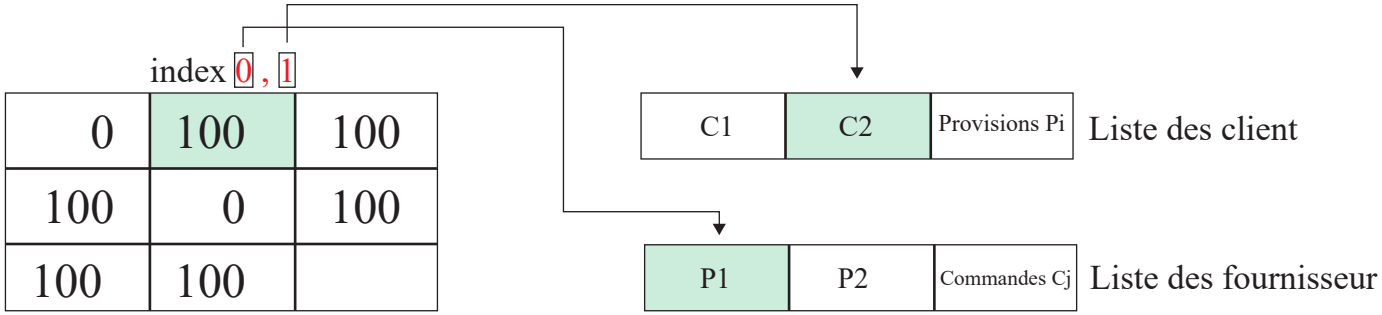
1.2 Constituer le graphe à partir de la martrice

1.2.1 Définir un hashMap pour stocker tous les sommets dans le graphe →

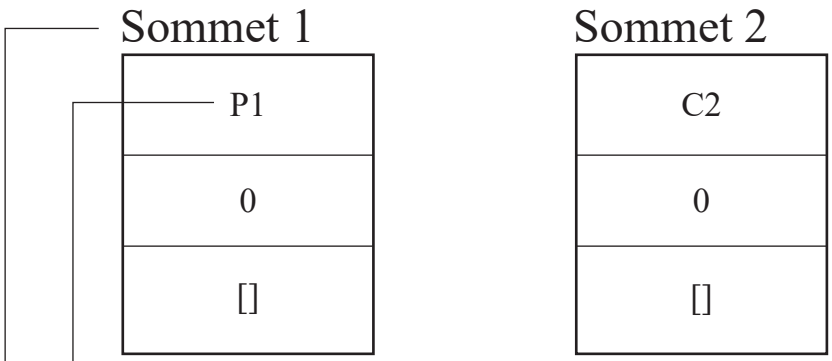
Clé	Valeur	...	...

1.2.2 Parcourir la matrice de proposition de transport (Balas-Hammer)

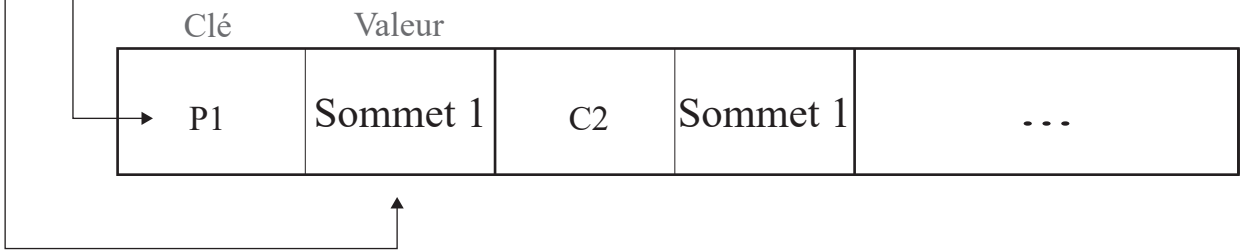
• Si la valeur de case != 0:



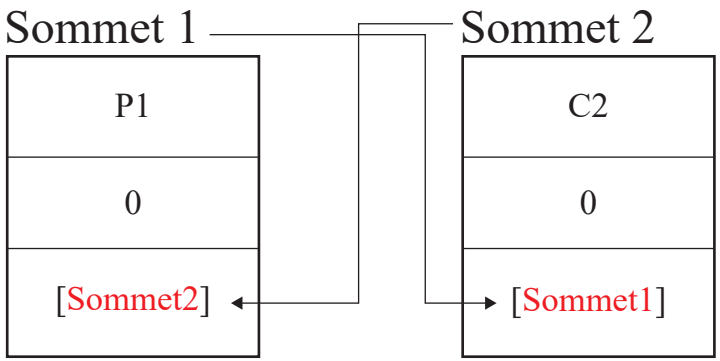
1. Créer les sommets



2. Intégrer les deux sommets dans le graphe, s'ils ne sont pas encore dans le hashMap

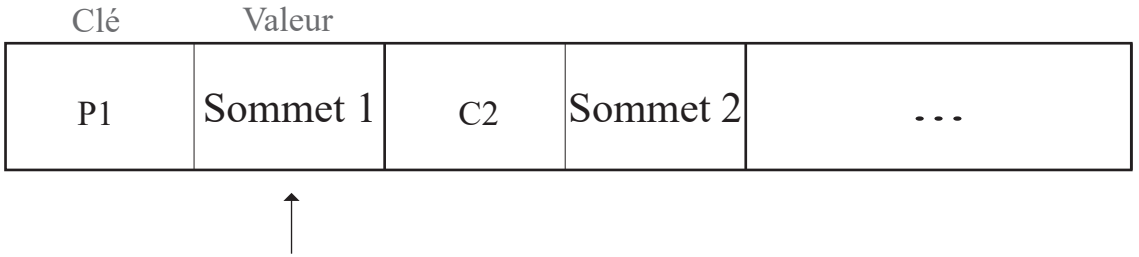


3. Créer l'arête entre les deux sommets

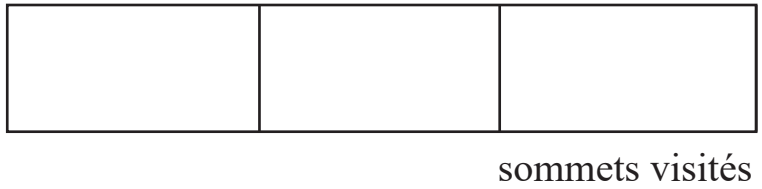


2. Détécter le circuit

2.1 Parcourir tous les sommets dans le graphe(hashMap)



2.1.2 Définir un Set pour stocker tous les sommets visités



2.1.3 Définir une liste pour stocker tous les sommets qui forment le circuit

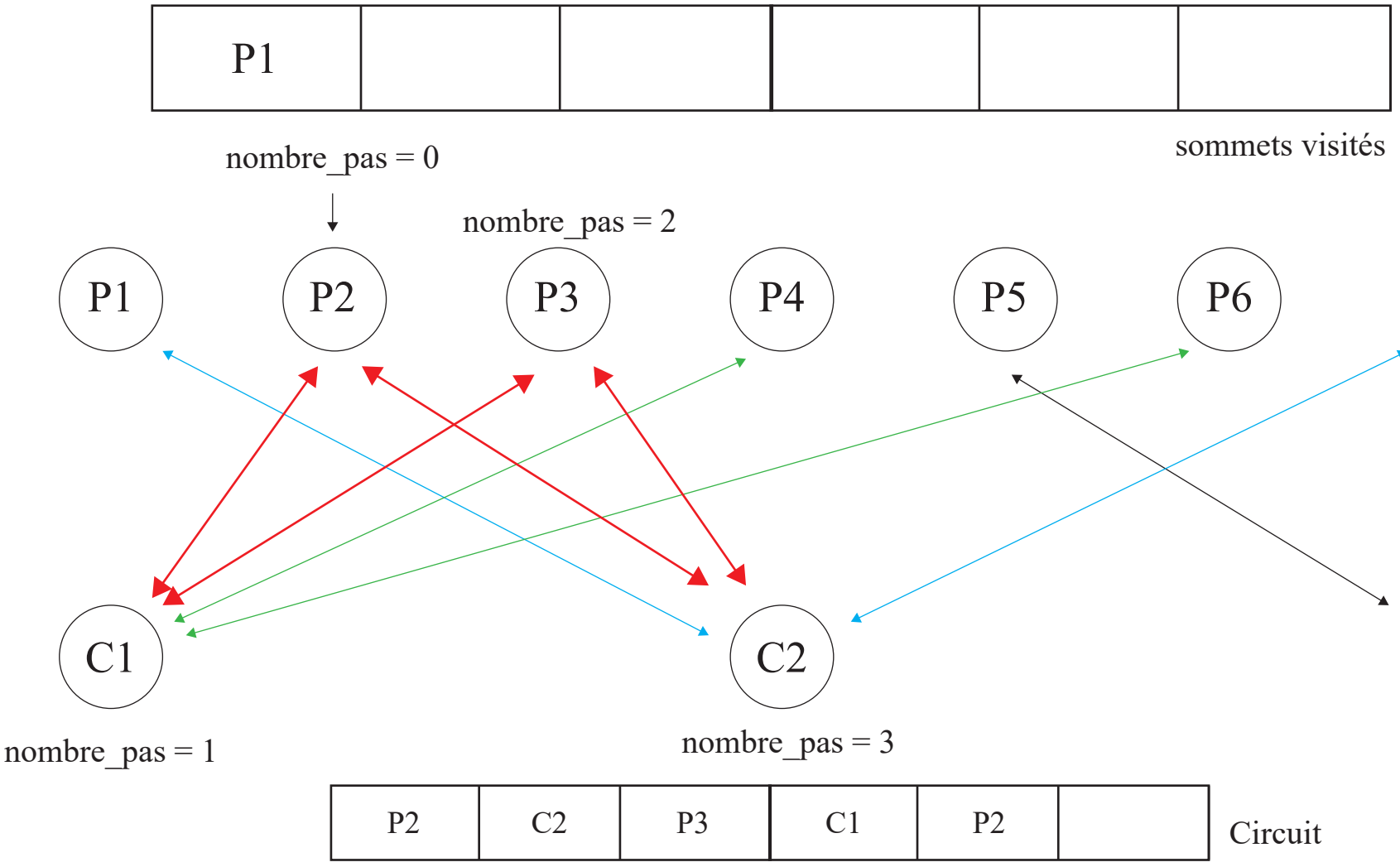
2.1.4 Définir avoirCircuit(sommet, parent, sommet_départ, nombre_pas)

variable : sommet : le sommet courant

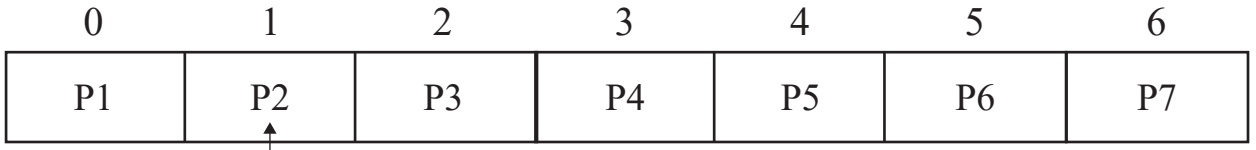
parent: le sommet qui constitue l'arête avec le sommet courant

sommet_départ: le premier sommet

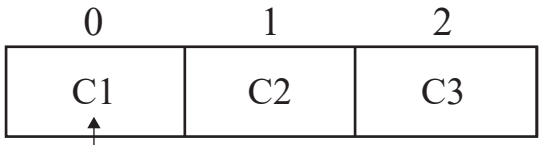
nombre_pas: le nombre de sommet parcouru



3. Suppression de circuit

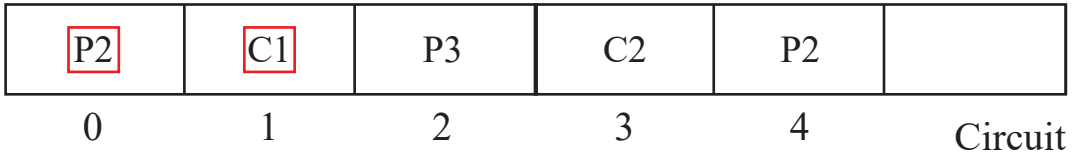


Liste des fournisseurs



Liste des clients

	C1	C2	C3	Provisions Pi
P1	0	100	0	100
P2	50	50	0	100
P3	50	50	0	100
P4	100	0	0	100
P5	0	0	100	100
P6	100	0	0	100
P7	0	100	0	100
Commandes Cj	400	200	100	



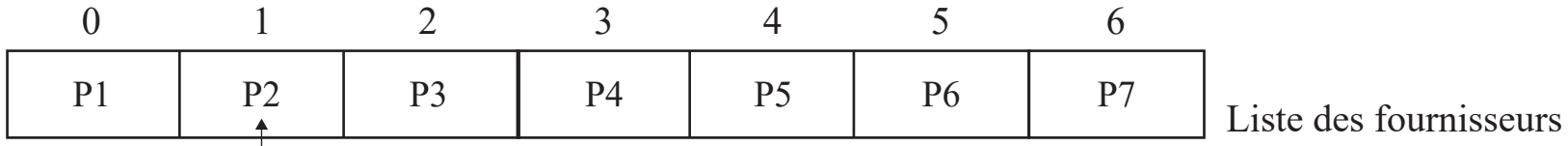
Circuit

3.1 Parcourir la liste du circuit,

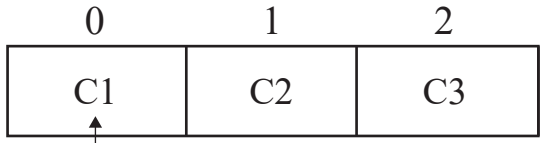
On va prendre circuit[index] et circuit[index+1]

3.2 Utiliser les valeurs trouvées dans la liste de circuit chercher les index correspondants dans la liste des fournisseurs et celle des clients

3.3 Avec ces index, on se refére à la matrice de transport pour trouver la plus petite valeur dans le circuit

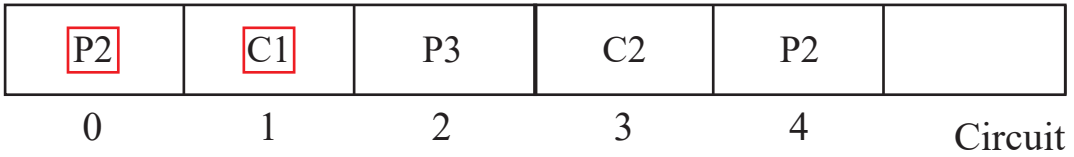


Liste des fournisseurs



Liste des clients

	C1	C2	C3	Provisions Pi
P1	0	100	0	100
P2	50->100	50->0	0	100
P3	50->0	50->100	0	100
P4	100	0	0	100
P5	0	0	100	100
P6	100	0	0	100
P7	0	100	0	100
Commandes Cj	400	200	100	



3.4 Parcourir la liste du circuit,

On va prendre circuit[index] et circuit[index+1]

3.5 Utiliser les valeurs trouvées dans la liste de circuit,
puis chercher les index correspondants dans la liste
des fournisseurs et celle des clients

3.6 Avec la valeur minimale, on procéde à la maximisation
l'index du circuit % 2 ==0,
valeur initiale = valeur initiale + valeur minimale
Sinon:
valeur initiale = valeur initiale - valeur minimale

4. Vérifier la connexité

4.1 Parcourir tous les sommets dans le graphe(hashMap)

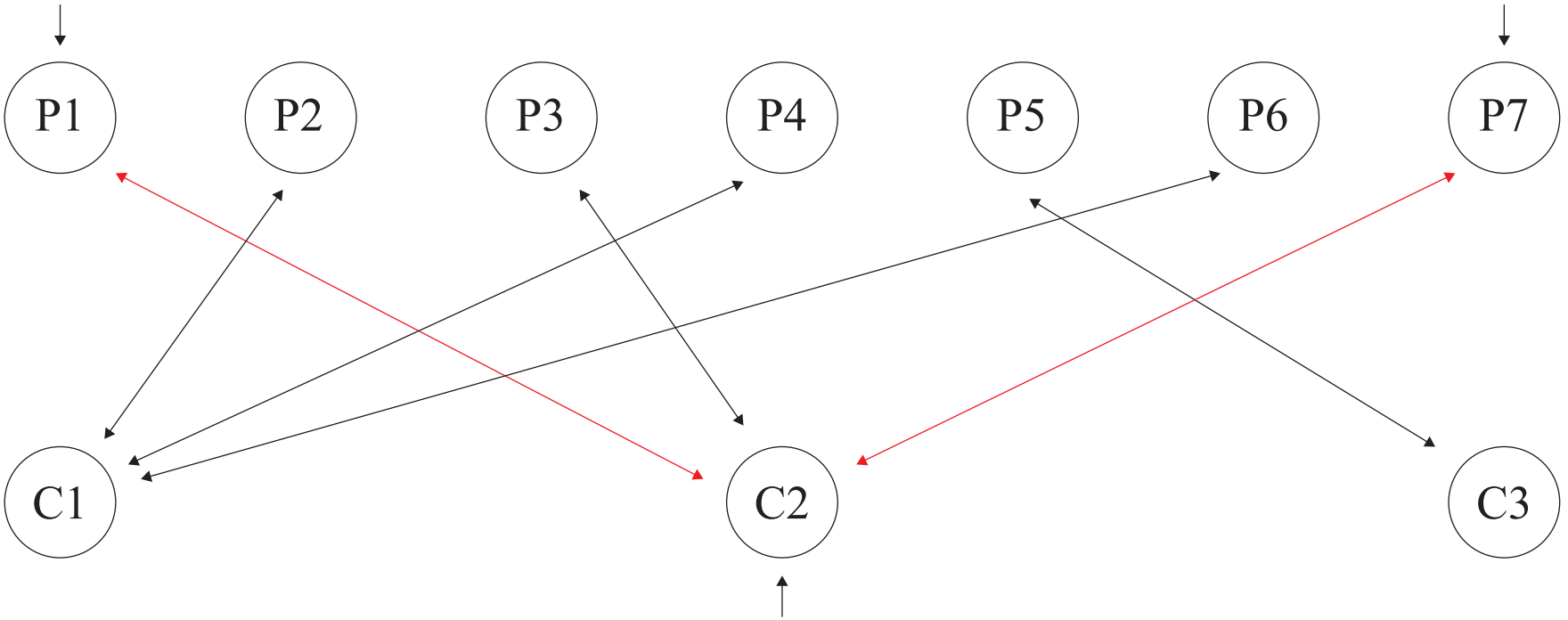
Clé	Valeur			
P1	Sommet 1	C2	Sommet 2	...

4.1.2 Définir un Set pour stocker tous les sommets visités

P1	C2	P7			

sommets visités

4.1.3 Définir une liste pour stocker l'ensemble des sous-graphes



4.1.4 Retourner le Set des sommets visités

4.1.5 Vérifier la connexité ou stocker les sous-graphes

Si length(Set) == length(graphe):

Le graphe est connexe

Sinon:

Parcourir la liste qui contient l'ensemble des sous-graphes(Si elle n'est pas vide)

Si chaque sous-liste n'a pas d'intersection avec le Set:

Rajouter le Set dans la liste

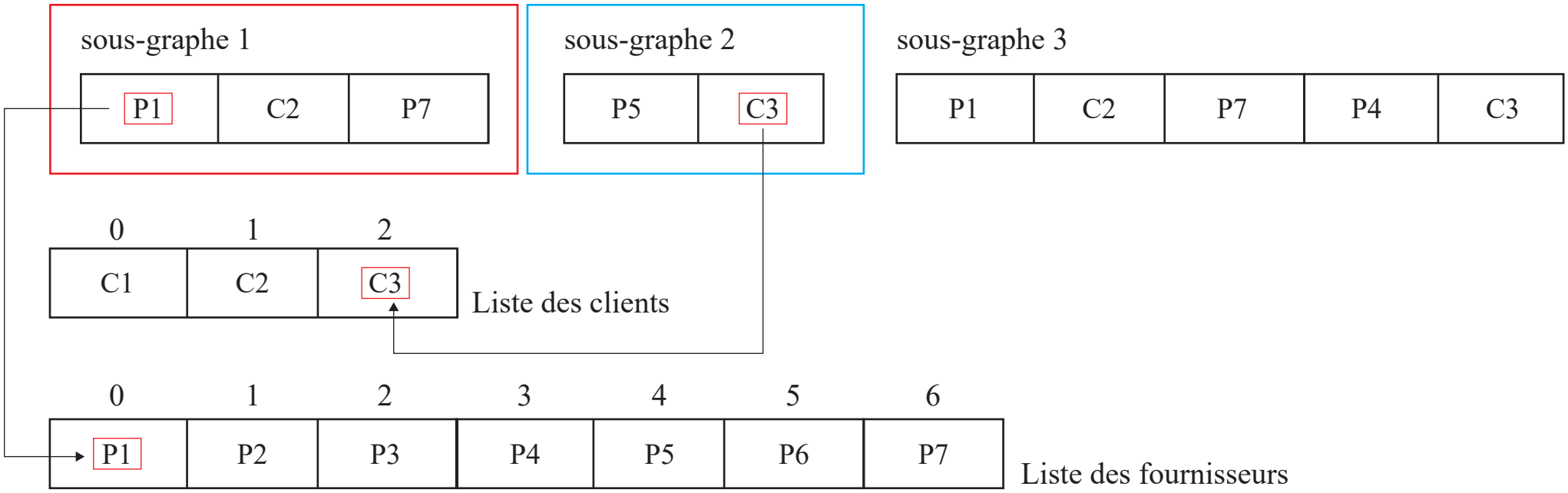
Sinon: on repasse à l'étape 4.1 pour trouver le nouveau sous-graphe

Losque l'on a parcouru tous les sommets, on retourne la liste de sous-graphes

5. Rajouter des arêtes pour que le graphe soit connexe

5.1 Définir trois variables: arête_ajouter: liste, index_arête_ajouter: liste, minVal: entité

5.2 Parcourir la liste qui contient l'ensemble des sous-graphes



	C1	C2	C3	Provisions Pi
P1	30	20	15	100
P2	10	50	2	100
P3	9	10	30	100
P4	6	2	29	100
P5	50	40	3	100
P6	5	38	27	100
P7	50	4	22	100
Commandes Cj	400	200	100	

Matrice des coûts

5.2.1 Si la valeur parcourue contient ‘P’:

- Dans la même boucle, parcourir les autres sous-graphes

Si la valeur parcourue contient ‘C’:

- Avec ces deux valeurs,On releve les index correspondants dans la liste des fournisseurs et celle des client
- A partir des index, on trouve le coût correspondant dans la matrice.

Si coût < minVal:

minVal = coût

arête_ajouter = [”P1”,”C3”]

index_arête_ajouter = [0,2]

5.2.2 Ajouter l'arête

arête_ajouter = ["P4","C2"] | index_arête_ajouter = [3,1]

Clé	Valeur							
P1	Sommet 1	C2	Sommet 2	- - -	P4	Sommet 3	C3	Sommet 4

• Avec les noms dans arête_ajouter, on cherche les sommets correspondants dans le hashMap, et rajouter une arête entre ces deux sommets

Sommet 2

P4
0
[sommet 3]

Sommet 3

C2
0
[sommet 2]

5.2.3 Fusionner les deux sous-graphes

sous-graphe 1

P1	C2	P7	P1	C2	P7	P4	C3

sous-graphe 2

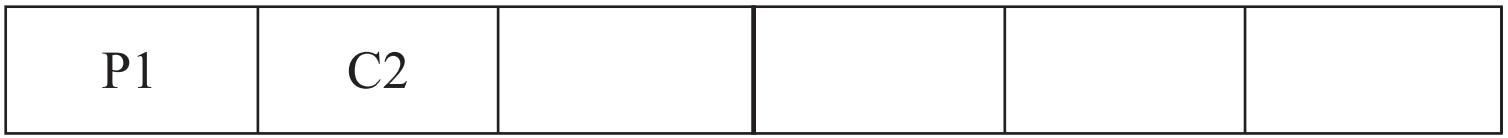
P5	C3

5.2.4 Retourner à l'étape 5.2

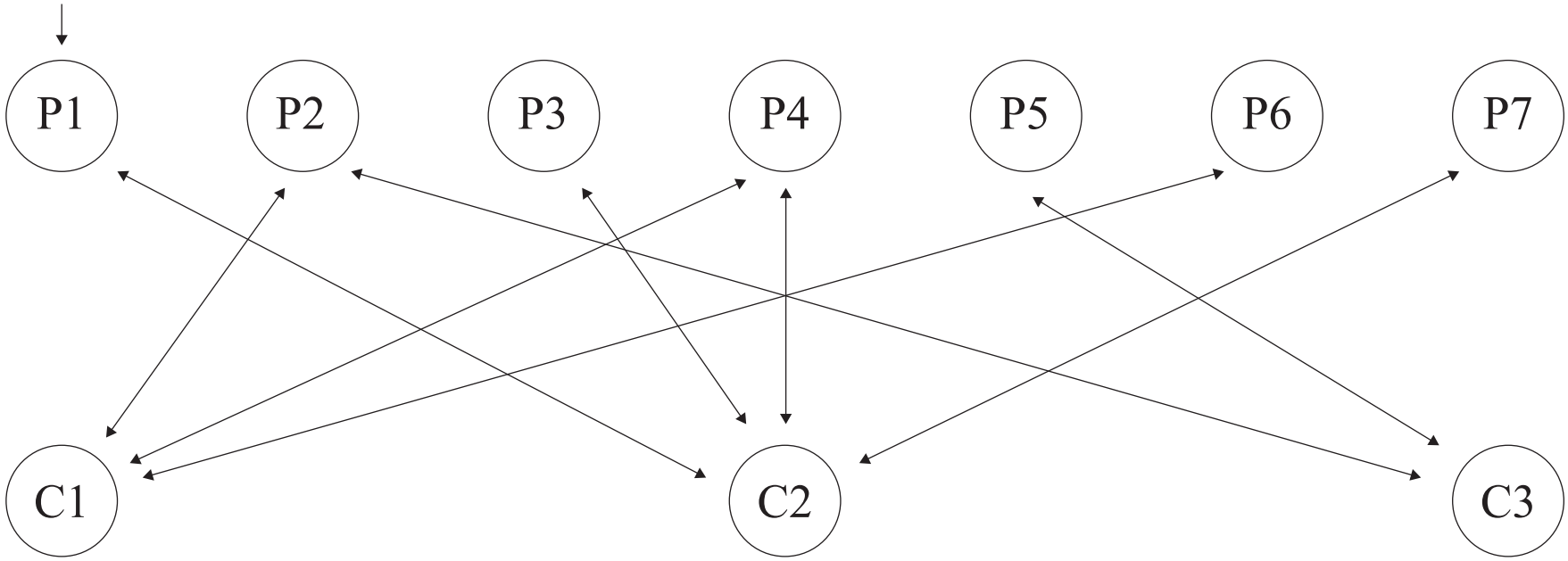
Lorsqu'il y a un seul sous-graphe Stop !

6. Calculer la valeur de chaque sommet

6.1 Définir un Set pour stocker tous les sommets visités

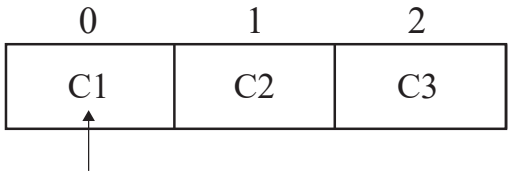


sommets visités

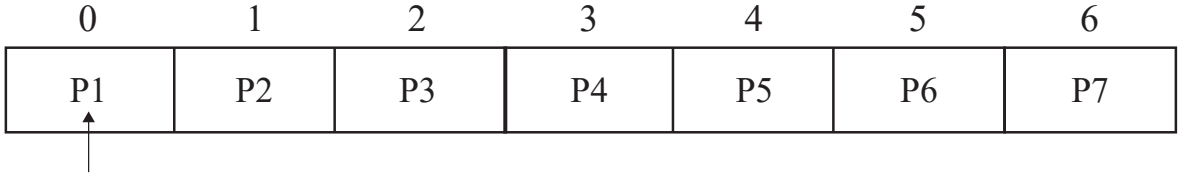


- Lorsque le premier sommet et le premier pas: sommet.valeur = coût correspondant
- Lorsque le sommet.nom contient ‘P’ : sa fille.valeur = sommet.valeur - coût correspondant
- Lorsque le sommet.nom contient ‘C’ : sa fille.valeur = coût correspondant + sommet.valeur

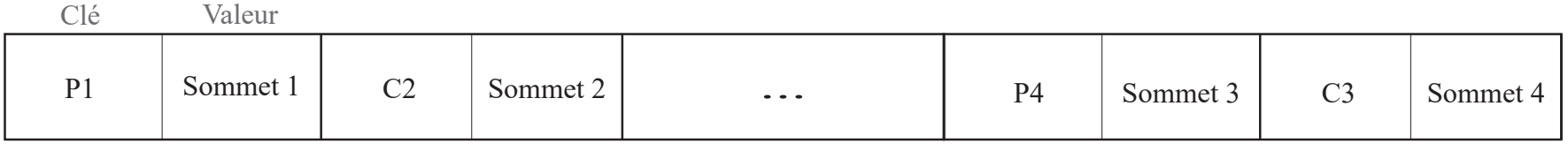
7. Calculer le coût potentiel



Liste des clients



Liste des fournisseurs



	C1	C2	C3	Provisions Pi
P1	24	20	16	100
P2	10	6	2	100
P3	9	5	1	100
P4	6	2	-2	100
P5	11	7	3	100
P6	5	1	-3	100
P7	8	4	0	100
Commandes Cj	400	200	100	

Coûts potentiels

7.1.1 Parcourir la liste des fournisseurs

- Dans la même boucle, parcourir la liste des clients
- Avec les deux noms parcourus, on va trouver les valeurs de ces deux sommets dans le graphe
- Calculer le coût potentiel entre ces deux sommets, et designer ce coût à la matrice

	C1	C2	C3	Provisions Pi
P1	30	20	15	100
P2	10	50	2	100
P3	9	10	30	100
P4	6	2	29	100
P5	50	40	3	100
P6	5	38	27	100
P7	50	4	22	100
Commandes Cj	400	200	100	

Coûts initiaux

-

	C1	C2	C3	Provisions Pi
P1	24	20	16	100
P2	10	6	2	100
P3	9	5	1	100
P4	6	2	-2	100
P5	11	7	3	100
P6	5	1	-3	100
P7	8	4	0	100
Commandes Cj	400	200	100	

Coûts potentiels

=

	C1	C2	C3	Provisions Pi
P1	6	0	-1	100
P2	0	44	0	100
P3	0	5	29	100
P4	0	0	31	100
P5	39	33	0	100
P6	0	37	30	100
P7	42	0	22	100
Commandes Cj	400	200	100	

Coûts marginaux

8. Calculer le coût marginal

9. Détécter la valeur négative et ajouter la nouvelle arête

	C1	C2	C3	Provisions Pi
P1	6	0	-1	100
P2	0	44	0	100
P3	0	5	29	100
P4	0	0	31	100
P5	39	33	0	100
P6	0	37	30	100
P7	42	0	22	100
Commandes Cj	400	200	100	

Coûts marginaux

9.1 Parcourir la matrice

- S'il n'y a pas de valeur négative:
  - Le coût de transport est minimisé
- Sinon:
  - Rajouter l'arête entre les deux sommets
  - Détécter le circuit
    - Supprimer le circuit(Maximisation)
  - Repasser à l'étape 1 de marche-pied

10. Amélioration de la maximisation

Lors de la maximisation, le delta = 0 ou il s'agit d'une égalité, on ne peux pas supprimer le circuit

1. Récupérer la liste des rajouts dans la partie de connexité

2. A la fois supprimer les arêtes dans la liste des rajouts, on garde l'arête ajouté selon la valeur négative

3. Lorsque l'on repasse à l'étape de connexité, on se refére à la liste des rajoutes pour les nouvelles arêtes à ajouter soient différentes que ceux qui sont dans la liste


Merci !