

# TER Sur Le Problème Des Préflots

DUVILLIE Guillaume  
Ould Mohamed Abdellahi Cheikh Mehdi

Université Montpellier2  
Master1-Informatique  
Spécialité-MOCA

6 mai 2012

# Sommaire

- 1 Introduction : le flot et ses algorithmes
  - Cadre du TER
  - Motivation
  - Les algorithmes de chaînes améliorantes
- 2 Le préflot et ses algorithmes
  - Définitions
  - Principe
  - Les procédures
  - Exemple
  - Les algorithmes dérivés
- 3 Les tests
  - Modus Operandi
  - Retour d'expériences
- 4 Conclusion

# Plan

- 1 Introduction : le flot et ses algorithmes
  - Cadre du TER
  - Motivation
  - Les algorithmes de chaînes améliorantes
- 2 Le préflot et ses algorithmes
  - Définitions
  - Principe
  - Les procédures
  - Exemple
  - Les algorithmes dérivés
- 3 Les tests
  - Modus Operandi
  - Retour d'expériences
- 4 Conclusion

# Cadre et objectifs

- Ce TER se situe dans le cadre de Problème de flot maximum
- Il existe différents types d'algorithmes répondant à ce problème, certains basés sur la recherche de chaînes améliorantes (Edmonds-Karp, Ford Fulkerson, ...), d'autres sur les fonctions de préflots. Nous nous intéresserons à la seconde catégorie.

# Utilité

Considérons une entreprise, devant acheminer sa production depuis l'usine de création au lieu de stockage.

Usine de Création

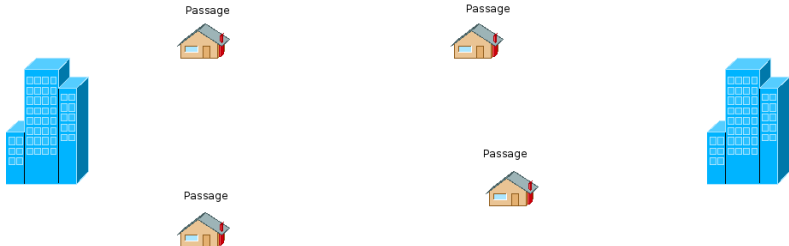


Entrepôt



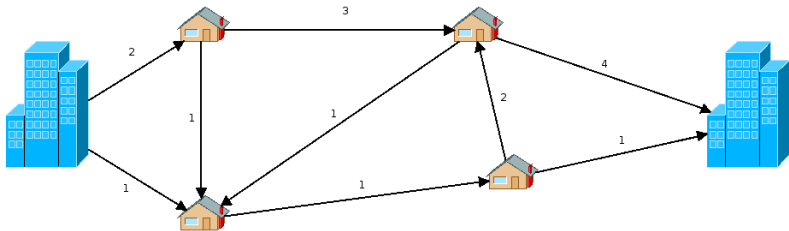
# Utilité

Il existe différents points, par lesquels peut transiter la production.



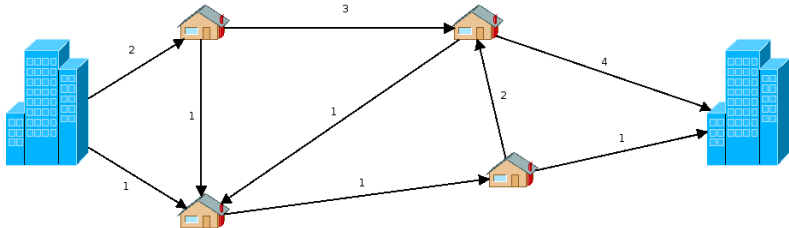
# Utilité

Chaque passage est reliés aux autres à l'aide de routes permettant un transit plus ou moins facile jusqu'à destination.



# Utilité

Chaque passage est reliés aux autres à l'aide de routes permettant un transit plus ou moins facile jusqu'à destination.



Bingo !

Il s'agit là d'un problème de flot maximum.



# Définitions

Nous allons définir quelques notions qui seront utilisées par la suite pour le déroulement de l'algorithme de préflot et ses dérivés.

# La notion de flot

Un flot est une fonction de graphes vérifiant certaines propriétés, qui sont :

# La notion de flot

Un flot est une fonction de graphes vérifiant certaines propriétés, qui sont :

le respect de la capacité de l'arc

La valeur du flot sur un arc ne peut être supérieure à la capacité de l'arc et ne peut être négative

# La notion de flot

Un flot est une fonction de graphes vérifiant certaines propriétés, qui sont :

le respect de la capacité de l'arc

$$\forall (i,j) \in A, \quad 0 \leq x(i,j) \leq c(i,j)$$

le respect de la loi de Kirchoff

Pour chaque noeud différent de la source et du puits, la quantité de flot entrant dans le noeud est égale à la quantité de flot sortant de ce dernier.

# La notion de flot

Un flot est une fonction de graphes vérifiant certaines propriétés, qui sont :

le respect de la capacité de l'arc

$$\forall (i, j) \in A, \quad 0 \leq x(i, j) \leq c(i, j)$$

le respect de la loi de Kirchoff

$$\forall i \in S - \{s, t\}, \quad \sum_{j \in A^+(i)} x(i, j) = \sum_{k \in A^-(i)} x(k, i)$$

Le respect de la contrainte de symétrie

Pour chaque arc du graphe, il n'y a aucun phénomène de perte entre le sommet de départ et le sommet d'arrivée.

# La notion de flot

Un flot est une fonction de graphes vérifiant certaines propriétés, qui sont :

le respect de la capacité de l'arc

$$\forall (i,j) \in A, \quad 0 \leq x(i,j) \leq c(i,j)$$

le respect de la loi de Kirchoff

$$\forall i \in S - \{s, t\}, \quad \sum_{j \in A^+(i)} x(i,j) = \sum_{k \in A^-(i)} x(k,i)$$

Le respect de la contrainte de symétrie

$$f = \sum_{j \in A^+(s)} x(s,j) = \sum_{k \in A^-(t)} x(k,t)$$



## Le réseau résiduel

Soit un graphe  $G(S, A)$ , un flot  $f$  et  $i, j$  deux sommets de  $G$ . On dira que l'arête  $(i, j)$  appartiendra au réseau résiduel  $A_f$  si et seulement si la capacité résiduelle est non nulle.

Autrement dit

$$r(i, j) = c(i, j) - x(i, j) > 0$$

On appelle alors  $r(i, j)$  la capacité résiduelle de l'arête  $(i, j)$ .

# Le réseau résiduel

Soit un graphe  $G(S, A)$ , un flot  $f$  et  $i, j$  deux sommets de  $G$ . On dira que l'arête  $(i, j)$  appartiendra au réseau résiduel  $A_f$  si et seulement si la capacité résiduelle est non nulle.

Autrement dit

$$r(i, j) = c(i, j) - x(i, j) > 0$$

On appelle alors  $r(i, j)$  la capacité résiduelle de l'arête  $(i, j)$ .

Note

Si  $x(i, j) > 0$  alors  $(j, i)$  appartient au réseau résiduel avec une capacité résiduelle  $r(j, i) = x(i, j)$ .



## Le réseau résiduel

Soit un graphe  $G(S, A)$ , un flot  $f$  et  $i, j$  deux sommets de  $G$ . On dira que l'arête  $(i, j)$  appartiendra au réseau résiduel  $A_f$  si et seulement si la capacité résiduelle est non nulle.

Autrement dit

$$r(i, j) = c(i, j) - x(i, j) > 0$$

On appelle alors  $r(i, j)$  la capacité résiduelle de l'arête  $(i, j)$ .

**Théorème**

S'il existe un chemin de la source au puits dans le réseau résiduel, alors on peut augmenter la valeur du flot dans le graphe. Ce chemin est appelé chaîne améliorante.

# Le réseau résiduel

Pour le flot suivant, ...

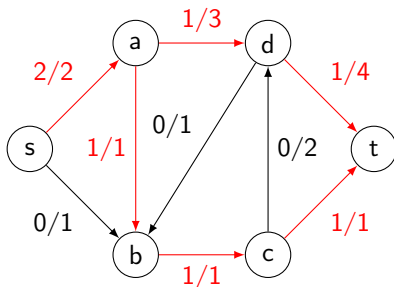


Figure: Un exemple de flot

# Le réseau résiduel

... on a le réseau résiduel suivant :

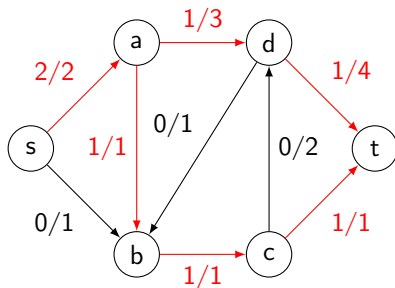


Figure: Un exemple de flot

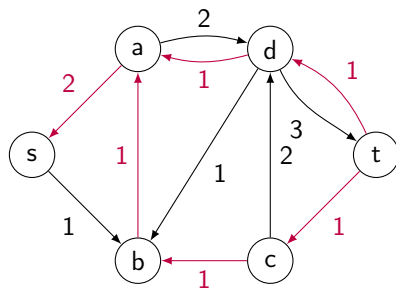


Figure: Le réseau résiduel associé

# Le réseau résiduel

Il existe une chaîne améliorante dans le graphe d'écart :

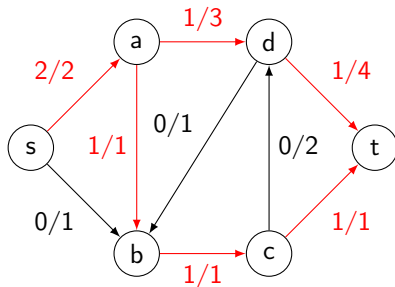


Figure: Un exemple de flot

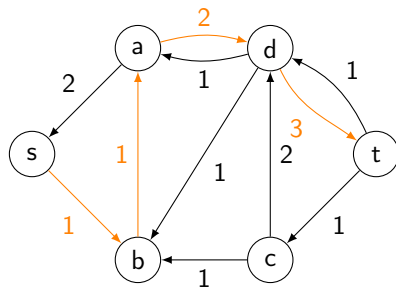


Figure: Le réseau résiduel associé

# Le réseau résiduel

Donc une augmentation possible du flot :

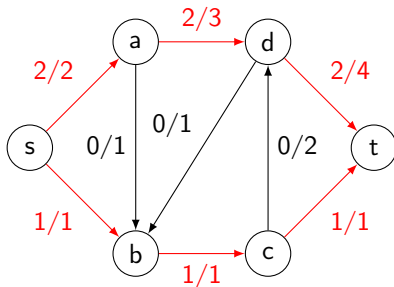


Figure: Un exemple de flot

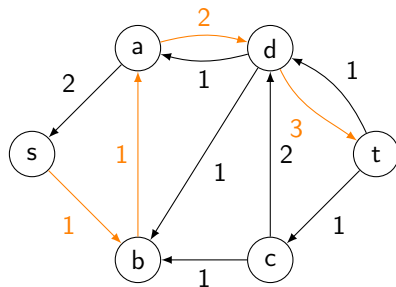


Figure: Le réseau résiduel associé

# Les algorithmes

Varient sur le choix des chaînes améliorantes :

# Les algorithmes

Variante sur le choix des chaînes améliorantes :

- Ford-Fulkerson : choix aléatoire

# Les algorithmes

Varient sur le choix des chaînes améliorantes :

- Ford-Fulkerson : choix aléatoire
- Edmonds-Karp : la plus petite chaîne en nombre d'arcs



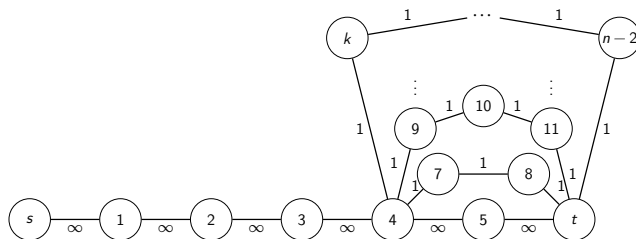
# Les algorithmes

Variante sur le choix des chaînes améliorantes :

- Ford-Fulkerson : choix aléatoire
- Edmonds-Karp : la plus petite chaîne en nombre d'arcs
- Dinic : l'ensemble des plus petites chaînes en nombre d'arcs

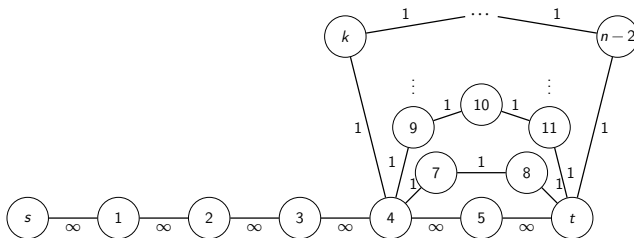
# Les limites de ces algorithmes

Considérons le graphe suivant :



# Introduction aux opérations

Considérons le graphe suivant :

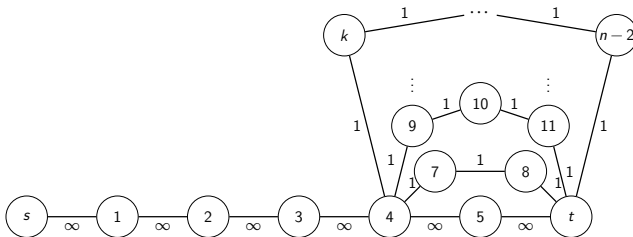


## Constat

Il met en défaut les algorithmes de recherche de chaînes améliorantes.

# Introduction aux opérations

Considérons le graphe suivant :



## Intuition

Agir de façon locale au noeud 4, afin d'envoyer du flot à tous ses voisins en une seule opération.

# Plan

- 1 Introduction : le flot et ses algorithmes
  - Cadre du TER
  - Motivation
  - Les algorithmes de chaînes améliorantes
- 2 Le préflot et ses algorithmes
  - Définitions
  - Principe
  - Les procédures
  - Exemple
  - Les algorithmes dérivés
- 3 Les tests
  - Modus Operandi
  - Retour d'expériences
- 4 Conclusion

# Intuitivement...

Le graphe est assimilé à un réseau de tuyaux de capacité donnée.

# Intuitivement...

Le graphe est assimilé à un réseau de tuyaux de capacité donnée.

Chaque noeud possède un réservoir de capacité infinie.

# Intuitivement...

Le graphe est assimilé à un réseau de tuyaux de capacité donnée.

Chaque noeud possède un réservoir de capacité infinie.

Seule la source possède un réservoir plein.



# Intuitivement...

Le graphe est assimilé à un réseau de tuyaux de capacité donnée.

Chaque noeud possède un réservoir de capacité infinie.

Seule la source possède un réservoir plein.

À chaque noeud est associée une hauteur.

# Intuitivement...

Le graphe est assimilé à un réseau de tuyaux de capacité donnée.

Chaque noeud possède un réservoir de capacité infinie.

Seule la source possède un réservoir plein.

À chaque noeud est associée une hauteur.

Deux actions possibles :

- 1 vidange du réservoir

# Intuitivement...

Le graphe est assimilé à un réseau de tuyaux de capacité donnée.

Chaque noeud possède un réservoir de capacité infinie.

Seule la source possède un réservoir plein.

À chaque noeud est associée une hauteur.

Deux actions possibles :

- 1 vidange du réservoir
- 2 augmentation de la hauteur du réservoir

# Intuitivement...

Le graphe est assimilé à un réseau de tuyaux de capacité donnée.

Chaque noeud possède un réservoir de capacité infinie.

Seule la source possède un réservoir plein.

À chaque noeud est associée une hauteur.

Deux actions possibles :

- 1 vidange du réservoir
- 2 augmentation de la hauteur du réservoir

On commence par vidanger la source, puis on cherche à vidanger les noeuds dont le réservoir n'est pas vide.

# Intuitivement...

Le graphe est assimilé à un réseau de tuyaux de capacité donnée.

Chaque noeud possède un réservoir de capacité infinie.

Seule la source possède un réservoir plein.

À chaque noeud est associée une hauteur.

Deux actions possibles :

- 1 vidange du réservoir
- 2 augmentation de la hauteur du réservoir

On commence par vidanger la source, puis on cherche à vidanger les noeuds dont le réservoir n'est pas vide.

Si on ne peut vidanger aucun réservoir et qu'il reste des réservoirs non vides, on augmente la hauteur de ces derniers.

# La notion de préflot

Un préflot est une fonction obtenue à partir d'un flot par relaxation de la loi de Kirchoff : la quantité de préflot sortant d'un noeud doit être inférieure ou égale à la quantité de préflot entrant dans celui-ci.

# La notion de préflot

Un préflot est une fonction obtenue à partir d'un flot par relaxation de la loi de Kirchoff : la quantité de préflot sortant d'un noeud doit être inférieure ou égale à la quantité de préflot entrant dans celui-ci.

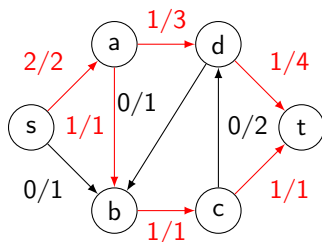


Figure: Un exemple de flot

# La notion de préflot

Un préflot est une fonction obtenue à partir d'un flot par relaxation de la loi de Kirchoff : la quantité de préflot sortant d'un noeud doit être inférieure ou égale à la quantité de préflot entrant dans celui-ci.

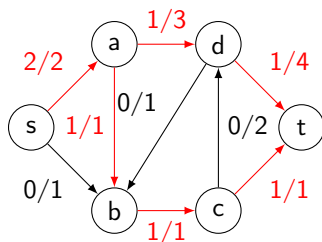


Figure: Un exemple de flot

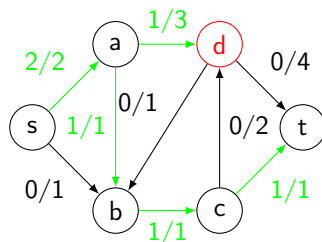


Figure: Un exemple de préflot



# La notion de préflot

Un préflot est une fonction obtenue à partir d'un flot par relaxation de la loi de Kirchoff : la quantité de préflot sortant d'un noeud doit être inférieure ou égale à la quantité de préflot entrant dans celui-ci.

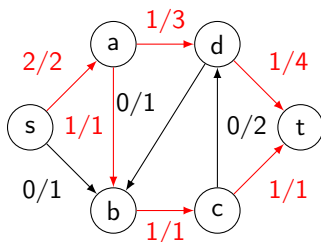


Figure: Un exemple de flot

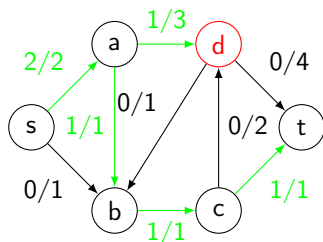


Figure: Un exemple de préflot

## Définition

$e(d) = f_{in} - f_{out} = 1$ .  $d$  est appelé noeud actif et  $e(d)$ , l'excédent de  $d$ .

# La fonction de distance

Une fonction de distance est une fonction représentant le plus court chemin en nombre d'arcs d'un noeud au puits, vérifiant les propriétés suivantes :

# La fonction de distance

Une fonction de distance est une fonction représentant le plus court chemin en nombre d'arcs d'un noeud au puits, vérifiant les propriétés suivantes :

## Distance de la source

la distance de la source est égale au nombre de sommets du graphe

# La fonction de distance

Une fonction de distance est une fonction représentant le plus court chemin en nombre d'arcs d'un noeud au puits, vérifiant les propriétés suivantes :

## Propriétés

$$d(s) = |S|$$

## Distance du puits

la distance du puits est égale à zéro

# La fonction de distance

Une fonction de distance est une fonction représentant la distance d'un noeud au puits, vérifiant les propriétés suivantes :

## Propriétés

$$d(s) = |S|$$

$$d(t) = 0$$

## Distances des sommets

si  $(i, j)$  appartient au réseau résiduel, la distance du sommet  $i$  est inférieure ou égale à la distance du sommet  $j$  plus un

# La fonction de distance

Une fonction de distance est une fonction représentant la distance d'un noeud au puits, vérifiant les propriétés suivantes :

## Propriétés

$$d(s) = |S|$$

$$d(t) = 0$$

$$\forall (i, j) \in A_f, \quad d(i) \leq d(j) + 1$$

# Poussage

**Poussage** : L'action de faire transiter le préflot d'un noeud  $i$  à un noeud  $j$  est appelée opération de poussage.

# Poussage

**Poussage** : L'action de faire transiter le préflot d'un noeud  $i$  à un noeud  $j$  est appelée opération de poussage.

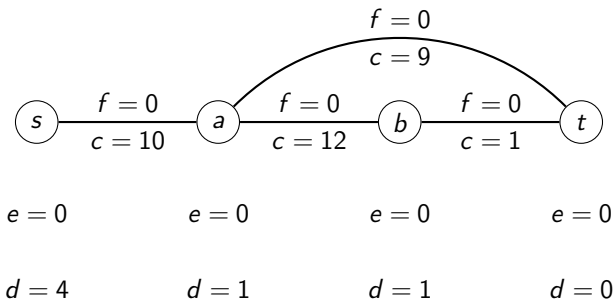
## Condition

On ne peut pas appliquer un poussage sur l'arête  $(i, j)$  que si le sommet  $i$  est actif,  $C_f(i, j) > 0$  et  $d(i) = d(j) + 1$ . L'arête  $(i, j)$  sera alors appelée arête acceptable.



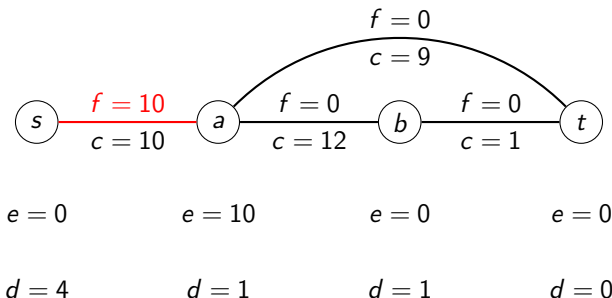
# Insuffisance des poussages

Soit le graphe :



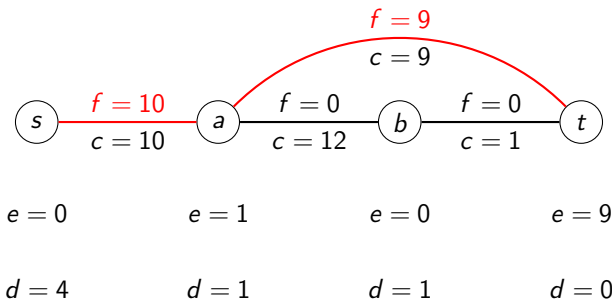
# Insuffisance des poussages

Soit le graphe :



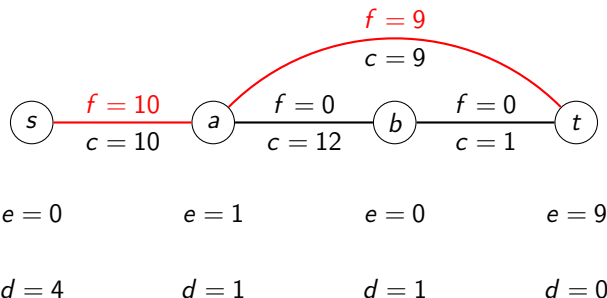
# Insuffisance des poussages

Soit le graphe :



# Insuffisance des poussages

Soit le graphe :



Et maintenant ...

Que faire ?

# Réétiquetage

**Réétiquetage** : Le ré-étiquetage consiste à réévaluer la distance du noeud actif au noeud puits afin qu'il soit possible, après cette opération, d'effectuer une opération de poussage depuis ce noeud.

# Réétiquetage

**Réétiquetage** : Le ré-étiquetage consiste à réévaluer la distance du noeud actif au noeud puits afin qu'il soit possible, après cette opération, d'effectuer une opération de poussage depuis ce noeud.

## Condition

On ne peut pas réétiqueter que si  $i$  est actif et, pour tout  $j \in S$  tel que  $(i, j) \in A_f$ . On a  $d(i) \leq d(j)$ .

# Approche formelle

L'algorithme examine les noeuds actifs.

# Approche formelle

L'algorithme examine les noeuds actifs.

Il cherche à réduire l'excédent de flot en l'envoyant vers un noeud voisin dans le graphe résiduel (Poussage).



# Approche formelle

L'algorithme examine les noeuds actifs.

Il cherche à réduire l'excédent de flot en l'envoyant vers un noeud voisin dans le graphe résiduel (Poussage).

S'il n'existe aucun voisin, on augmente la distance du noeud au puits (Ré-étiquetage).

# Approche formelle

L'algorithme examine les noeuds actifs.

Il cherche à réduire l'excédent de flot en l'envoyant vers un noeud voisin dans le graphe résiduel (Poussage).

S'il n'existe aucun voisin, on augmente la distance du noeud au puits (Ré-étiquetage).

Lorsque plus aucun noeud est actif, le préflot restant est un flot maximum.

# Initialisation

Consiste à calculer les distances de chacun des noeuds de façon à obtenir une fonction de distance valide.

# Initialisation

Consiste à calculer les distances de chacun des noeuds de façon à obtenir une fonction de distance valide.

Procédure d'initialisation :

- 1: **for**  $i \in S$  **do**
- 2:     Calculer la distance  $d(i)$  en nombre d'arêtes de  $i$  à  $t$
- 3: **end for**
- 4: **for**  $a \in A(s)$  **do**
- 5:      $x(a) \leftarrow c(a)$
- 6: **end for**
- 7:  $d(s) \leftarrow |S|$

# Examen des noeuds

Consiste à sélectionner un noeud actif et à effectuer un poussage si possible, ou un ré-étiquetage sinon.

# Examen des noeuds

Consiste à sélectionner un noeud actif et à effectuer un poussage si possible, ou un ré-étiquetage sinon.

Procédure Pousser-Réétiqueter :

- 1: **if**  $e(i) > 0$  **then**
- 2:   **if**  $\exists j$  tel que  $c(i,j) > 0$  **and**  $d(i) = d(j) + 1$  **then**
- 3:      $\delta \leftarrow \min(e(i), c(i,j))$
- 4:      $f(i,j) \leftarrow f(i,j) + \delta$
- 5:      $e(i) \leftarrow e(i) - \delta$
- 6:      $e(j) \leftarrow e(j) + \delta$
- 7:   **else**
- 8:      $d(i) \leftarrow 1 + \min\{d(j) / (i,j) \in A_f\}$
- 9:   **end if**
- 10: **end if**

# La procédure principale

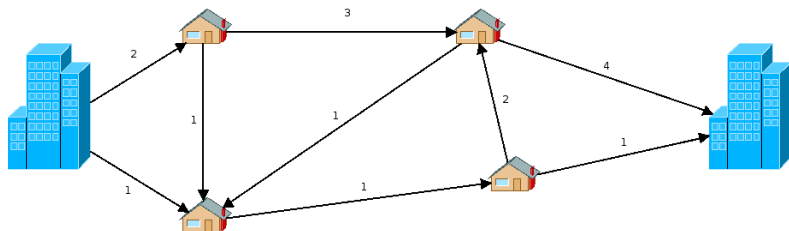
Consiste à initialiser le problème, et parcourir les noeuds pour l'application du poussage et ré-étiquetage.

Algorithme Générique :

- 1: Initialisation()
- 2: **while** Il existe un noeud actif  $i$  **do**
- 3:   Pousser-Réétiqueter( $i$ )
- 4: **end while**

# Sur le graphe de départ

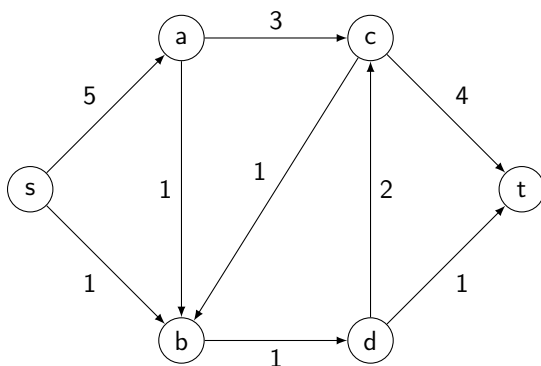
Reprenons notre entreprise ...



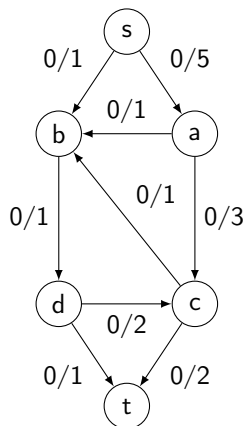


## Sur le graphe de départ

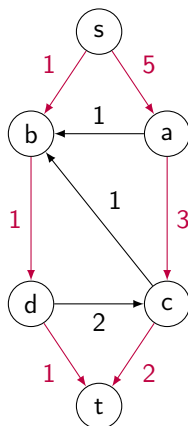
... et associons lui un graphe.



# Initialisation



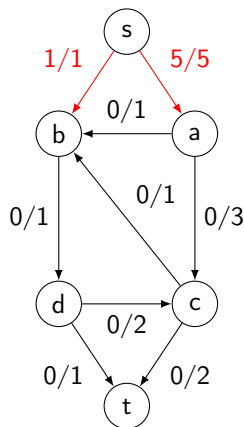
Graphe de préflot



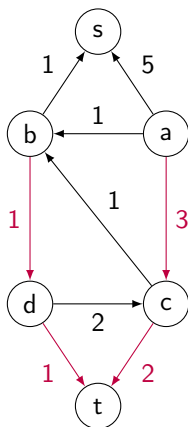
Réseau résiduel

$i$	$e(i)$	$d(i)$
s	$\infty$	$3^*$
a	0	2
b	0	2
c	0	1
d	0	1
t	0	0

# Phase 1 - Poussage



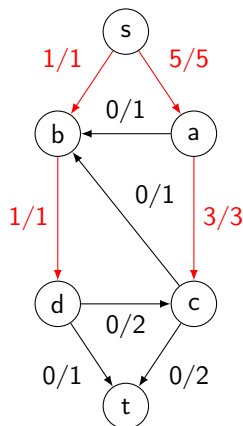
Graphe de préflot



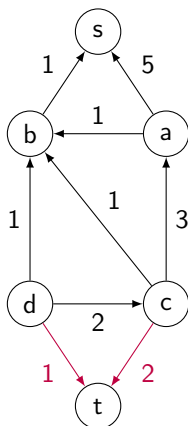
Réseau résiduel

$i$	$e(i)$	$d(i)$
s	$\infty$	6*
a	5	2
b	1	2
c	0	1
d	0	1
t	0	0

# Phase 1 - Poussage



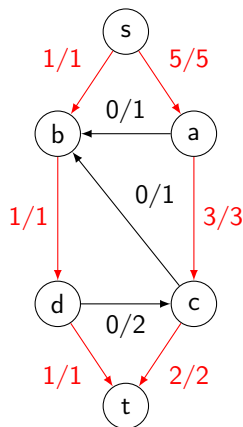
Graphe de préflot



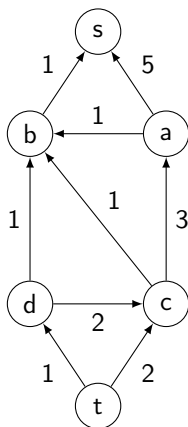
Réseau résiduel

$i$	$e(i)$	$d(i)$
s	$\infty$	6*
a	2	2
b	0	2
c	3	1
d	1	1
t	0	0

# Phase 1 - Poussage



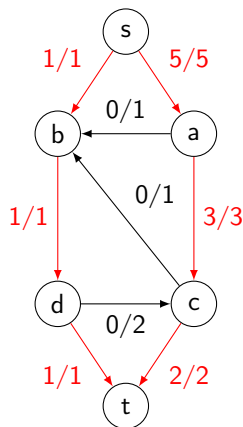
Graphe de préflot



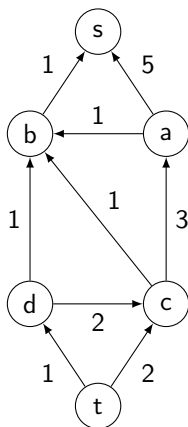
Réseau résiduel

$i$	$e(i)$	$d(i)$
s	$\infty$	6*
a	2	2
b	0	2
c	1	1
d	0	1
t	3	0

# Phase 1 - Réétiquetage



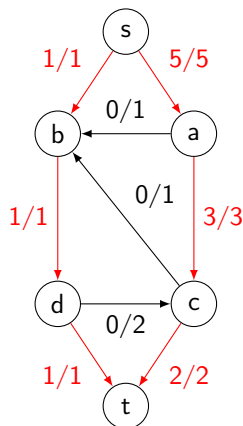
Graphe de préflot



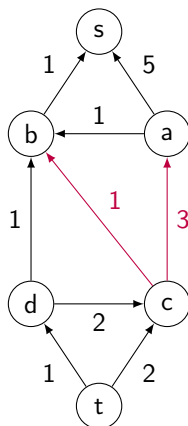
Réseau résiduel

$i$	$e(i)$	$d(i)$
s	$\infty$	6*
a	2	2
b	0	2
c	1	3
d	0	1
t	3	0

# Phase 1 - Réétiquetage



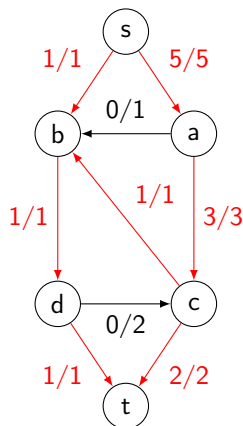
Graphe de préflot



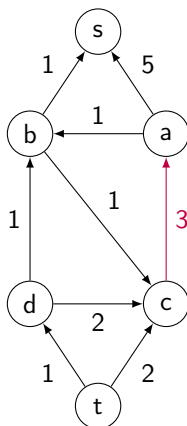
Réseau résiduel

$i$	$e(i)$	$d(i)$
s	$\infty$	6*
a	2	2
b	0	2
c	1	3
d	0	1
t	3	0

## Phase 2 - Poussage



Graphe de préflot

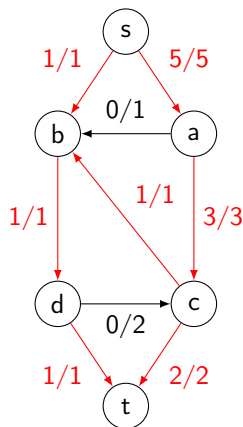


Réseau résiduel

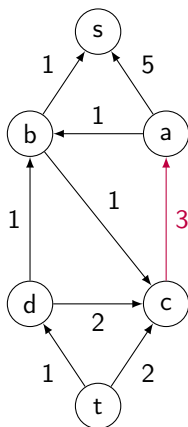
$i$	$e(i)$	$d(i)$
s	$\infty$	6*
a	2	2
b	1	2
c	0	3
d	0	1
t	3	0



## Phase 2 - Réétiquetage



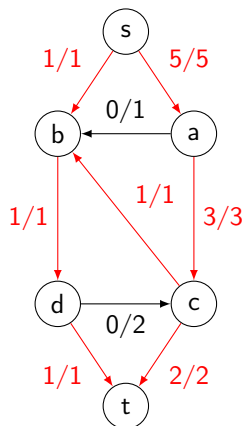
Graphe de préflot



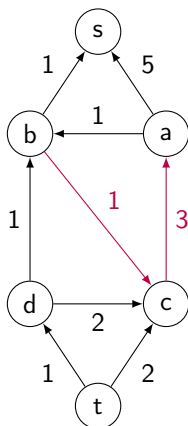
Réseau résiduel

$i$	$e(i)$	$d(i)$
s	$\infty$	6*
a	2	2
b	1	4
c	0	3
d	0	1
t	3	0

## Phase 2 - Réétiquetage



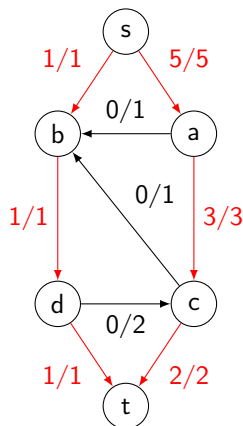
Graphe de préflot



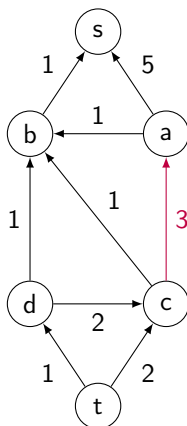
Réseau résiduel

$i$	$e(i)$	$d(i)$
s	$\infty$	6*
a	2	2
b	1	4
c	0	3
d	0	1
t	3	0

## Phase 3 - Poussage



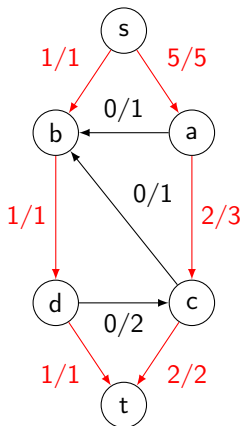
Graphe de préflot



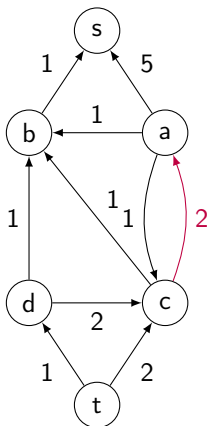
Réseau résiduel

$i$	$e(i)$	$d(i)$
s	$\infty$	6*
a	2	2
b	0	4
c	1	3
d	0	1
t	3	0

## Phase 3 - Poussage



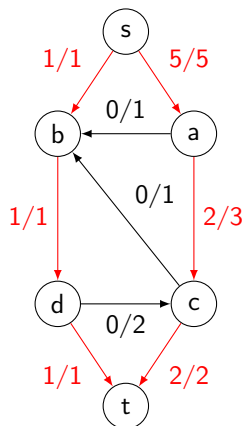
Graphe de préflot



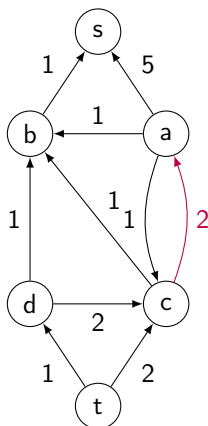
Réseau résiduel

$i$	$e(i)$	$d(i)$
s	$\infty$	6*
a	3	2
b	0	4
c	0	3
d	0	1
t	3	0

## Phase 3 - Réétiquetage



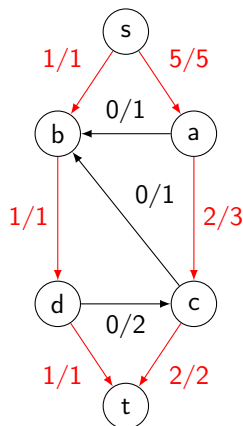
Graphe de préflot



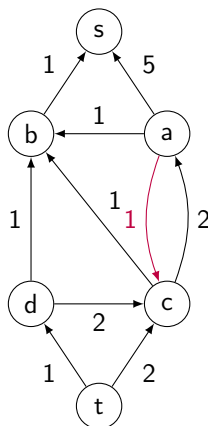
Réseau résiduel

$i$	$e(i)$	$d(i)$
s	$\infty$	6*
a	3	4
b	0	4
c	0	3
d	0	1
t	3	0

## Phase 3 - Réétiquetage



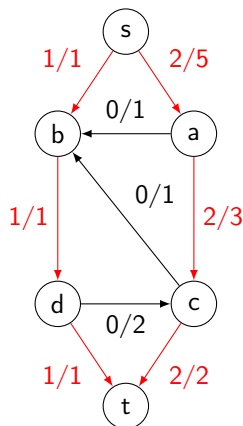
Graphe de préflot



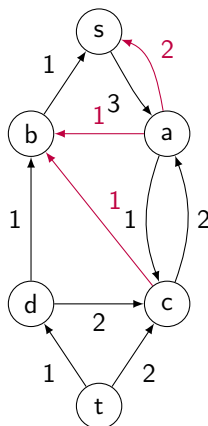
Réseau résiduel

$i$	$e(i)$	$d(i)$
s	$\infty$	6*
a	3	4
b	0	4
c	0	3
d	0	1
t	3	0

# Quelques tours d'algorithme plus tard...



Graphe de préflot



Réseau résiduel

$i$	$e(i)$	$d(i)$
s	$\infty$	6*
a	0	7
b	0	6
c	0	7
d	0	1
t	3	0

# Une question de choix

Les algorithmes dérivés sont basés sur un choix plus judicieux des sommets actifs à traiter.



# Une question de choix

Les algorithmes dérivés sont basés sur un choix plus judicieux des sommets actifs à traiter.

**L'algorithme FIFO** : dès qu'un noeud devient actif, on le place dans une file. On traite alors les sommets de la file dans l'ordre *First In First Out*

# Une question de choix

Les algorithmes dérivés sont basés sur un choix plus judicieux des sommets actifs à traiter.

**L'algorithme FIFO** : dès qu'un noeud devient actif, on le place dans une file. On traite alors les sommets de la file dans l'ordre *First In First Out*

**L'algorithme High Label** : on choisit de préférence, le noeud actif ayant la plus grande distance au puits

# Complexités

	Algorithme	Complexité
Chaînes améliorantes	Edmonds-Karp	$O(SA^2)$
	Dinic	$O(S^2A)$
Poussage Réétiquetage	Générique	$O(S^2A)$
	FIFO	$O(S^3)$
	High Label	$O(S^2\sqrt{A})$

# Plan

- 1 Introduction : le flot et ses algorithmes
  - Cadre du TER
  - Motivation
  - Les algorithmes de chaînes améliorantes
- 2 Le préflot et ses algorithmes
  - Définitions
  - Principe
  - Les procédures
  - Exemple
  - Les algorithmes dérivés
- 3 Les tests
  - Modus Operandi
  - Retour d'expériences
- 4 Conclusion

# Les tests

- Définit la densité :  $r = \frac{m}{n}$

# Les tests

- Définit la densité :  $r = \frac{m}{n}$
- Fait varier le nombre de noeuds

# Les tests

- Définit la densité :  $r = \frac{m}{n}$
- Fait varier le nombre de noeuds
- Réalise les trois algorithmes sur 100 graphes aléatoires différents

# Les tests

- Définit la densité :  $r = \frac{m}{n}$
- Fait varier le nombre de noeuds
- Réalise les trois algorithmes sur 100 graphes aléatoires différents
- Trace les courbes



# Les résultats attendus

Pour une densité  $r = O(n^2)$  :

# Les résultats attendus

Pour une densité  $r = O(n^2)$  :

- Une complexité en  $O(n^4)$  pour Dinic

# Les résultats attendus

Pour une densité  $r = O(n^2)$  :

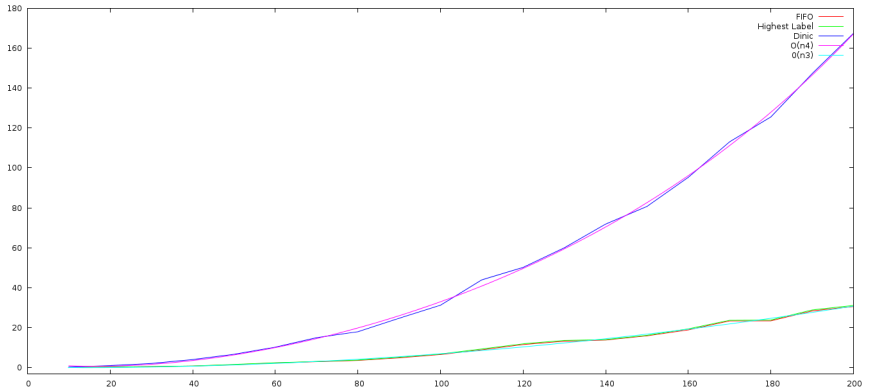
- Une complexité en  $O(n^4)$  pour Dinic
- Une complexité en  $O(n^3)$  pour les algorithmes FIFO et High Label

# Les résultats attendus

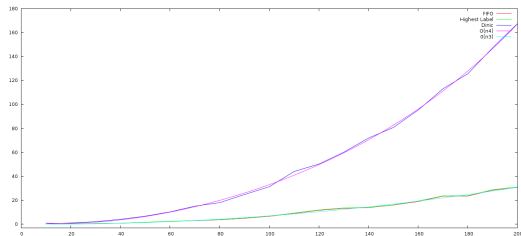
Pour une densité  $r = O(n^2)$  :

- Une complexité en  $O(n^4)$  pour Dinic
- Une complexité en  $O(n^3)$  pour les algorithmes FIFO et High Label
- Une exécution plus rapide pour FIFO et High Label

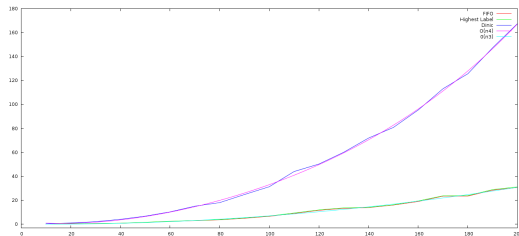
# Les résultats obtenus



# Les résultats obtenus

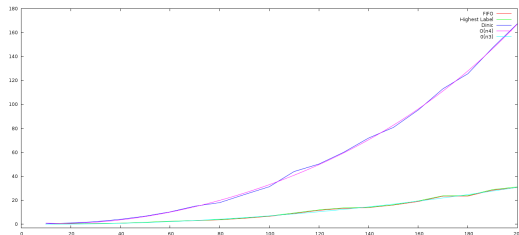


# Les résultats obtenus



•  $O(n^4)$  pour Dinic

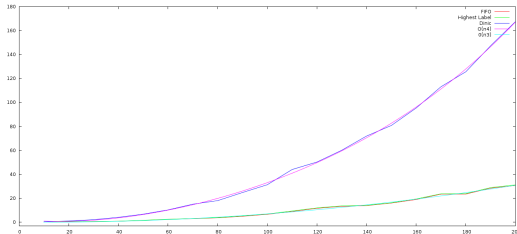
# Les résultats obtenus



- $O(n^4)$  pour Dinic
- $O(n^3)$  pour les algorithmes de préflots



## Les résultats obtenus



- $O(n^4)$  pour Dinic
- $O(n^3)$  pour les algorithmes de préflots
- Un temps égal pour les algorithmes de préflots

# Plan

- 1 Introduction : le flot et ses algorithmes
  - Cadre du TER
  - Motivation
  - Les algorithmes de chaînes améliorantes
- 2 Le préflot et ses algorithmes
  - Définitions
  - Principe
  - Les procédures
  - Exemple
  - Les algorithmes dérivés
- 3 Les tests
  - Modus Operandi
  - Retour d'expériences
- 4 Conclusion

# Conclusion

- Algorithmes intéressants (complexités, principes, fonctions de préflots, ...)

# Conclusion

- Algorithmes intéressants (complexités, principes, fonctions de préflots, ...)
- Parmi les plus rapides en pratique et en théorie