Dossier Optimisation et Recherche Opérationnelle

Thomas Duvignau / Chloé Boissavy

Université Lyon 2

M1 Informatique 2018 / 2019

Table des matières

[I. Introduction 2](#_Toc530761329)

[II. Algorithme 1 : Coloration d’un graphe non orienté avec Welsh-Powell 2](#_Toc530761330)

[1. Objet de l’algorithme 2](#_Toc530761331)

[2. Pseudo-code de l’algorithme 2](#_Toc530761332)

[3. Code R de l’algorithme 3](#_Toc530761333)

[4. Illustration sur un exemple 4](#_Toc530761334)

[III. Algorithme 2 4](#_Toc530761335)

[IV. Algorithme 3 4](#_Toc530761336)

[V. Conclusion 4](#_Toc530761337)

# Introduction

Nous avons écrit ce dossier afin de répondre à la demande d’une évaluation en contrôle continu de nos connaissances sur le cours d’Optimisation et de Recherche Opérationnelle, supervisé par le professeur Julien Ah-Pine au premier semestre de notre M1 Informatique.

Vous pourrez trouver ci-dessous différents algorithmes avec leurs utilités, leurs buts, leurs codes ainsi que des explications détaillées sur leurs fonctionnement et sur pourquoi ils ont été implémenter ainsi.

Nous nous sommes intéressés en premier sur l’algorithme de Welsh-Powell, c’est-à-dire la coloration d’un graphe non orienté. Puis nous avons effectué la détermination d’un flot maximal dans un réseau avec capacités par Ford-Fulkerson. Et pour finir, nous avons fait des chemins de longueur p avec fermeture transitive et connexité.

# Algorithme 1 : Coloration d’un graphe non orienté avec Welsh-Powell

## Objet de l’algorithme

Ici, nous traitons des graphes non orienté et non pondéré. Le but de cet algorithme est d’attribuer différentes couleurs aux sommets du graphe tout en respectant la règle qui stipule que deux sommets liés par une arrête ne peuvent avoir la même couleur. Il est tout de même préférable d’avoir le moins de couleurs possible, ainsi attribuer une couleur à chaque sommet est trop simple et peu utile si des solutions meilleures sont disponibles.

Ce genre d’algorithme est très utilisé dans le domaine de la télécommunication ou pour résoudre des problèmes d’incompatibilités. Par exemple, si vous avez des produits chimiques, vous devez les ranger dans une armoire à plusieurs étages. Sachant que certain de ces produits chimiques peuvent exploser s’ils sont à côté sur la même étagère. Il suffit de faire un graphe avec les incompatibilités de chacun puis d’utiliser Welsh-Powell afin de trouver une solution rapide et efficace qui permettra de ne rien faire exploser.

Il faut savoir tout de même que l’algorithme de Welsh-Powell ne permet pas nécessairement d’obtenir le nombre de coloration minimale d’un graphe G = [X, U]. On appelle cela, un algorithme heuristique.

## Pseudo-code de l’algorithme

Voici ci-dessous l’algorithme que l’on a étudié en cours :

**Input :** A (matrice d’adjacence de G)

1. Ranger les sommets par ordre de degrés non croissant
2. k = 0
3. B = liste des sommets rangés par ordre de degrés non croissant
4. Tant que toutes les lignes de B ne sont pas colorées faire
5. k = k + 1
6. Tant que B ≠ ∅ faire
7. Colorer dans A par la couleur ck la 1ère ligne non colorée dans B ainsi que la colonne correspondante
8. B = liste des lignes non colorées ayant un zéro dans toutes les colonnes de A de couleur ck
9. Fin Tant que
10. B = liste des sommets non colorés rangés par ordre de degrés non croissant
11. Fin Tant que
12. **Output :** k-coloration de G

## Code R de l’algorithme

## Illustration sur un exemple

Pour illustrer cet exemple, nous allons utiliser la matrice d’adjacence :

**A =**  1 2 3 4 5 6 7

1. 0 1 1 0 1 0 1
2. 1 0 1 1 0 1 0
3. 1 1 0 1 1 0 1
4. 0 1 1 0 1 1 1
5. 1 0 1 1 0 1 1
6. 0 1 0 1 1 0 1
7. 1 0 1 1 1 1 0

Et on obtient comme résultat :

* X = c(1,2,3,4,5,6,7)
* A = cbind( c(0,1,1,0,1,0,1), c(1,0,1,1,0,1,0), c(1,1,0,1,1,0,1), c(0,1,1,0,1,1,1), c(1,0,1,1,0,1,1), c(0,1,0,1,1,0,1), c(1,0,1,1,1,1,0))
* Welsh\_Powell2(X,A)

[[1]]

[1] 3 6

[[2]]

[1] 4 1

[[3]]

[1] 5 2

[[4]]

[1] 7

On a donc 4 couleurs pour le graphe, la couleur 1 est pour les sommets 3 et 6, la couleur 2 pour les sommets 4 et 1, la couleur 3 pour les sommets 5 et 2 et la couleur 4 pour le sommet 7.

1. Détermination d’un flot maximal dans un réseau avec capacités par Ford-Fulkerson

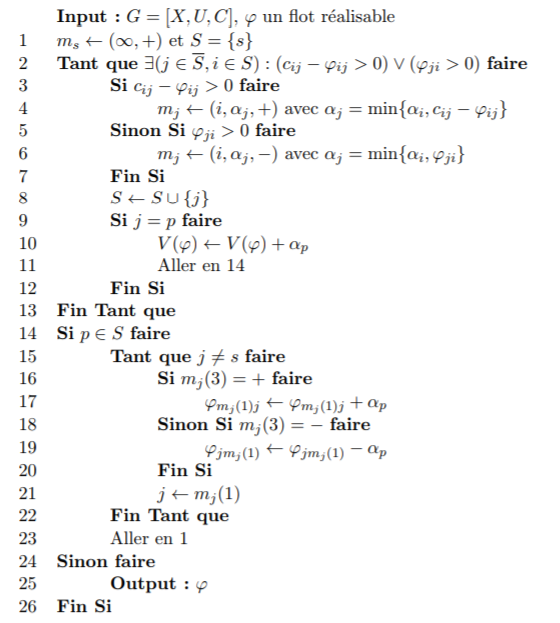
1. Objet de l’algorithme

Les problèmes de flots constituent un très important domaine de la théorie des graphes. Sous leur forme la plus simple, ils consistent à organiser de façon optimale, sous diverses contraintes, les mouvements de certaines quantité d’un bien dans un réseau.

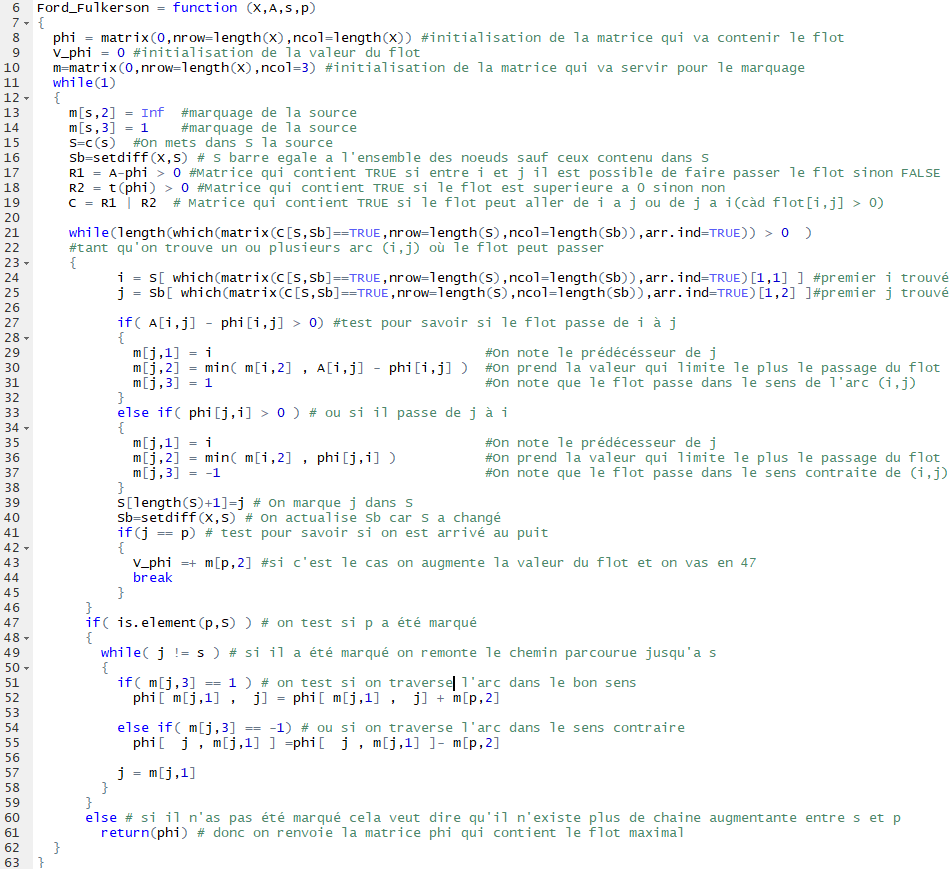
L’algorithme prend en paramètre la matrice de capacité du graphe (A), l’ensemble des sommets du graphe (X), la source et le puits (s et p). Et il retourne une matrice (phi) qui contient la valeur du flot maximal.

2. Pseudo-code de l’algorithme

Voici ci-dessous le pseudo code de l’algorithme que l’on a étudié en cours :

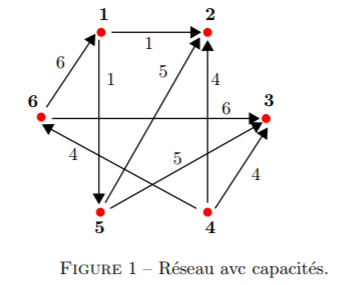


3. Code R de l’algorithme

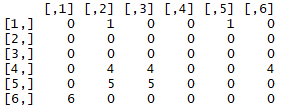


4. Illustration sur un exemple

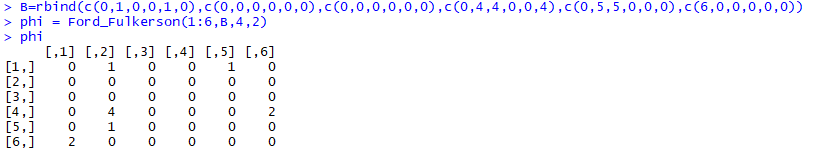
Le graphe :



Sa matrice de capacité :



L’exécutions du code R présenté dans la sous-section précédente donne le résultat suivant :



1. Chemins de longueur p, fermeture transitive et connexité

1. Objet de l’algorithme

Cette partie ce compose de deux algorithme, le premier qui permet de calculer la puissance booléenne d’une matrice et le second qui permet de trouver la fermeture transitive d’un graphe a partir de sa matrice d’adjacences.

Le calcul de la puissance booléenne d’une matrice reviens finalement à calculer sa puissance normal et à remplacer les éléments de la matrice supérieure à 1 par un 1.

En effet si l’on prend la définition de la somme booléenne tel quel :

Alors on remarque x+y est égale a un nombre différent de 0 si x et y sont différent de zéro, donc il si le résultat de leur somme est supérieure a 1 il suffit de le ramener à 1 pour avoir le résultat de la somme booléenne.

De même si l’on prend la définition du produit booléen :

On remarque que x\*y est égale à nombre différent de zéro si x et y sont différent de zéro. Donc si le résultat de leur produit est supérieure à 1 il suffit de le ramener à 1 pour avoir le résultat du produit booléen.

Donc dans les algorithmes suivant il y a la présence de boucles for imbriqués pour tester chaque éléments de la matrice et les mettre à 1 si ils sont supérieure à 1.

2. Pseudo-code de l’algorithme

Pseudo code de l’algorithme de la puissance :

**Input**: A

1. result ← A

2. **Pour** p **de** 1 **à** (p-1)

3. result ← result \* A

4. **Fin Pour**

5. **Pour** i **de** 1 **a** N

6. **Pour** j **de** 1 **à** N

7. **Si** result[i][j] > 1

8. result[i][j] ← 1

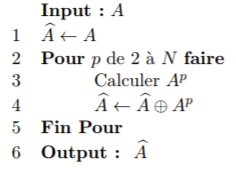
9. **Fin Si**

10. **Fin Pour**

11. **Fin Pour**

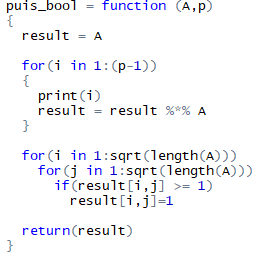
12. **Output** : result

Pseudo code de l’algorithme de la fermeture transitive :

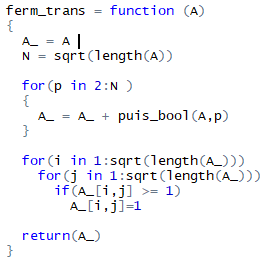


3. Code R de l’algorithme

code de l’algorithme de la puissance :

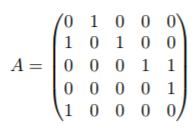


code de l’algorithme de la fermeture transitive :

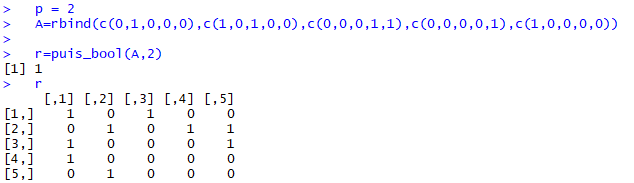
****

4. Illustration sur un exemple

Matrice des adjacences de A :



Application de notre fonction puissance :



Donc d’apres ce résult il existe des chemins de longueur deux entre les couples de nœuds suivants : (1,1) (1,3) (2,2) (2,4) (2,5) (3,1) (3,5) (4,1) (5,2)

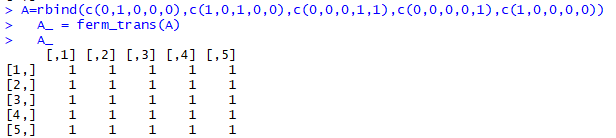
Si on reprend la matrice des adjacences de A on peut facilement retrouver ces chemins :

(1,1) = 1→2→1 (1,3) = 1→2→3 (2,2) = 2→1→2 (2,4) = 2→3→4

(2,5) = 2→3→5 (3,1) = 3→5→1 (3,5) = 3→4→5 (4,1) = 4→5→1

(5,2) = 5→1→2

Application de la fonction fermeture transitive sur A :



On remarque que la matrice de la fermeture transitive est composé uniquement de 1. Cela veut dire qu’il existe un chemin entre 1 et 1, 1 et 2, etc. Et donc cela veut dire qu’à partir de n’importe quel nœud du graphe on peut se rendre sur un autre nœud du graphe.

// Nom de la propriété j’ai pas réussit a trouver

# Conclusion

# 