



**INSTITUT AFRICAIN D'INFORMATIQUE**

**Etablissement Inter-états d'Enseignement Supérieur**

**B.P : 2263 Libreville (Gabon) Tél. (241) 07 70 55 00 – 07 70 56 00**

**Site web: [www.iaisiege.net](http://www.iaisiege.net) – Email: [contact@iaisiege.net](mailto:contact@iaisiege.net)**

# **RAPPORT DE TRAVAUX PRATIQUES**

## **COURS DE THEORIE DES GRAPHS ET**

### **RECHERCHE OPERATIONNELLE**

## **Thème**

# **Implémentation de l'algorithme de Ford-Fulkerson (sous Matlab)**

Réalisé par:

KABORE Yabyouré Eric

ING1

Sous la supervision de :

Pr. Ousmane BALIRA KONFE

## Table des matières

Table des illustrations.....	3
Introduction .....	4
I. Présentation du problème et de l'algorithme de Ford Fulkerson.....	5
1. Présentation du problème .....	5
2. Traduction mathématique.....	5
3. Algorithme de Ford et Fulkerson .....	5
II. Implémentation de l'algorithme .....	6
1. L'environnement Matlab .....	6
2. Aperçu des fonctions écrites.....	6
III. Mise en marche de l'application.....	10
1. Importation du projet sous Matlab 2016 .....	10
2. Exécution du fichier Ford_Fulkerson.exe .....	13
3. Utilisation de l'application .....	13
4. Remarque.....	15
Conclusion.....	17

## Table des illustrations

FIGURE 1:FONCTION FULKERSON .....	7
FIGURE 2:FONCTION FLUX_MAX .....	7
FIGURE 3:FONCTION RESIDUEL .....	8
FIGURE 4:FONCTION CONSTRUIRE_FLUX.....	8
FIGURE 5;FONCTION CHERCHER_CHEMIN .....	9
FIGURE 6:FONCTION AJOUTER_ARC .....	9
FIGURE 7:FONCTION FINDEGE2 .....	10
FIGURE 8: FICHIER A OUVRIR AVEC MATLAB .....	10
FIGURE 9:CHANGEMENT DU REPERTOIRE COURANT DE MATLAB .....	11
FIGURE 10: COMMANDE DE LANCEMENT .....	11
FIGURE 11:APERÇU DE L'APPLICATION .....	12
FIGURE 12:FICHIER .EXE A DOUBLE-CLIQUER.....	13
FIGURE 13:AJOUT D'ARC DANS LE GRAPHE.....	13
FIGURE 14:GRAPHE DU COURS A TESTER.....	14
FIGURE 15: BOUTON DE LANCEMENT .....	14
FIGURE 16:RESULTAT DE L'APPLICATION .....	15
FIGURE 17: COMMANDES NATIVES POUR LE FLUX MAXIMAL .....	16
FIGURE 18:RESULTAT AVEC LA COMMANDE MAXFLOW.....	16

## Introduction

Un ingénieur est très souvent confronté dans le cadre de son travail à des problèmes d'optimisation. La théorie des graphes offre d'innombrables outils très puissants pour résoudre ces problèmes. C'est ainsi que dans le cadre de notre formation d'ingénieur, il nous a été donné de pouvoir recevoir des connaissances dans ce domaine à travers le cours de Théorie des Graphe et Recherche Opérationnelle. Dans le cadre de ce cours il nous a été demandé de réaliser une implémentation de l'algorithme de Ford-Fulkerson pour la recherche de flux dans un graphe orienté, sous l'environnement Matlab ou Maple. Le présent document est donc le rapport de ce que nous avons réalisé, de la manière dont nous l'avons conçu, et surtout de comment l'utiliser.

## I. Présentation du problème et de l'algorithme de Ford Fulkerson

### 1. Présentation du problème

On considère un réseau d'adduction d'eau d'une compagnie entre plusieurs localités. Les localités sont liées par des tuyaux n'ayant pas tous des capacités d'adduction égale du fait soit du coût soit en fonction du nombre d'habitant. Chaque tuyau a donc une capacité minimale et maximale. Le problème de flux maximal entre deux localités dont l'une est la source et l'autre la destination (voire puits) est donc de chercher la quantité optimale sur chaque tuyau pour que la quantité totale d'eau sortante de la source soit égale à la quantité totale entrante dans la destination et que cette quantité soit maximale.

### 2. Traduction mathématique

Le problème expliqué ci-dessus peut être traduit en mathématiques grâce à la théorie des graphes de la manière suivant.

Le réseau d'adduction sera appelé réseau de transport où graphe orienté anti-symétrique sans boucle. Concrètement, s'il existe un tuyau allant d'une ville A à une ville B, et de B à A alors les deux villes sont donc la seule et même ville.

Le flux d'un réseau de transport  $G = (X, A, C)$  de transport est maximal, si le réseau résiduel du réseau de transport ne possède pas de chemin de la source vers la destination. Le réseau résiduel étant défini par :

$$\bar{G}(\phi^k) = (X, \bar{A}, \bar{C}):$$

- si  $a \in A$  et  $\phi(a) < C(a)$  alors  $a \in \bar{A}$  et  $\bar{C}(a) = C(a) - \phi(a)$
- si  $a = (x, y) \in A$  et  $\phi(a) > 0$  alors  $a^{-1} = (y, x) \in \bar{A}$  et  $\bar{C}(a^{-1}) = \phi(a)$ .

L'algorithme de Ford-Fulkerson permet donc de résoudre le problème en construisant le graphe résiduel, en y cherchant un chemin entre la source et la destination et ce, de façon itérative jusqu'à ce qu'il n'y ait plus de chemin dans le graphe résiduel.

### 3. Algorithme de Ford et Fulkerson

L'algorithme ci-dessous sous-entend que les capacités minimales du graphe sont toutes égales à 0. En effet pour des capacités minimales différentes de zéro, il faudra juste remplacer le flux initial par les capacités minimales.

(a) - Itération  $k = 0$

Partir d'un flux initial  $\phi^0$  (compatible avec les contraintes de capacité), par exemple  $\phi^0 = (0, 0, \dots, 0)$ .

(b) - A l'itération  $k$ , soit  $\phi^k$  le flot courant. Rechercher un chemin  $\mu^k$  de  $X_1$  à  $X_n$  dans le graphe résiduel  $\bar{G}(\phi^k)$ . S'il n'en existe pas, alors FIN : le flux  $\phi^k$  est maximal.

(c) - Soit  $\varepsilon_k$  la capacité résiduelle du chemin  $\mu^k$  (minimum des capacités résiduelles des arcs du chemin). Définir le flux  $\phi^{k+1}$  par :

$$\begin{cases} (\phi^{k+1}) a = (\phi^k) a + \varepsilon^k k \text{ si } a \in \mu \text{ et si } a \text{ est orientée dans le sens de } \mu \\ (\phi^{k+1}) a = (\phi^k) a - \varepsilon^k k \text{ si } a \in \mu \text{ et si } a \text{ est orientée dans le sens contraire de } \mu \end{cases}$$

Faire  $k \leftarrow k + 1$  et retourner en (b).

## II. Implémentation de l'algorithme

### 1. L'environnement Matlab

Pour réaliser une implémentation de cet algorithme, nous disposons de plusieurs outils dont Matlab et Maple. Nous avons préféré utiliser Matlab à Maple pour la simple raison qu'étant plus à l'aise sur celui-ci. De plus la version de Matlab, la 2016b que nous avons utilisé possède en son sein une structure de graphe et pleines de fonction portants sur les graphes dont l'algorithme de Ford-Fulkerson, celui-là même que nous allons implémenter nous-même.

Comme dit ci-dessus la version de Matlab (2016b) que nous avons utilisé pour implémenter l'algorithme de Ford Fulkerson possède déjà une fonction qui résout le problème. Nous avons préféré ré-implémenter notre propre fonction par souci d'apprentissage mais aussi et surtout parce que cette fonction résout le problème de flux maximal sans que l'on ne puisse voir les modifications effectuées progressivement sur le graphe résiduel.

Nous nous sommes donc basé sur des fonctions intéressante de Matlab pour produire notre propre application, tel que :

- Addedge : pour l'ajout d'un arc dans le graphe.
- Rmeddge : pour la suppression d'un arc dans un graphe.
- Shortestpath : pour la recherche du plus courts chemin.
- Findedge : Qui permet de tester si un arc appartient à un graphe.

### 2. Aperçu des fonctions écrites

Ces fonctions citées ci-dessus nous ont permis d'écrire nos propres fonction avec une grande souplesse moyennant quelque bugs de notre code dû à notre méconnaissance de certaines spécifications du langage Matlab :

- La fonction Ford\_Fulkerson : Elle prend en entrée un graphe, une source et une destination et nous retourne le flux maximal, sous forme de vecteur, de dimension égale au nombre d'arcs. Ce vecteur contient en fait les capacités optimales sur chaque permettant arrête permettant d'obtenir le flux maximal. En Matlab il y'a une fonction principale par fichier script et le nom du script doit être égale au nom de cette fonction. Notre fonction Ford\_Fulkerson est la fonction principale du fichier script Ford\_Fulkerson.m

```

function fi = Fulkerson(G, s, t)
    fi = zeros(1, G.numedges);
    vide=[0 0];
    Gc = G;
    Gres = residuel(Gc, fi);

    chemin = chercher_chemin(Gres, s, t);
    fi = construire_flux(Gc, chemin, fi);
    while ( size(chemin) ~= vide)
        Gres = residuel(Gc, fi);
        chemin = chercher_chemin(Gres, s, t);
        subplot(2,2,1), plot(Gc, 'EdgeLabel', G.Edges.Weight), title('Graphe initial');
        pause(1);
        fi = construire_flux(Gc, chemin, fi);
    end
    G.Edges.Weight=fi;
    max=flux_max(G,s,fi);
    disp(fi);
    disp(max);
    text= strcat('Graphe final ayant pour flux max = ', num2str(max));
    subplot(2,2,[3,4]), p2=plot(G, 'EdgeLabel', G.Edges.Weight); highlight(p2,G, 'NodeColor',
end

```

Figure 1: fonction fulkerson

- La fonction flux\_max : Elle prend en entrée le vecteur flux renvoyé par la fonction Ford\_Fulkerson et de renvoyer le flux maximal selon le théorème de Kirchhoff, c'est-à-dire la somme des capacités des arcs sortant de la source. Cette fonction est la fonction principale du fichier script flux\_max.m

```

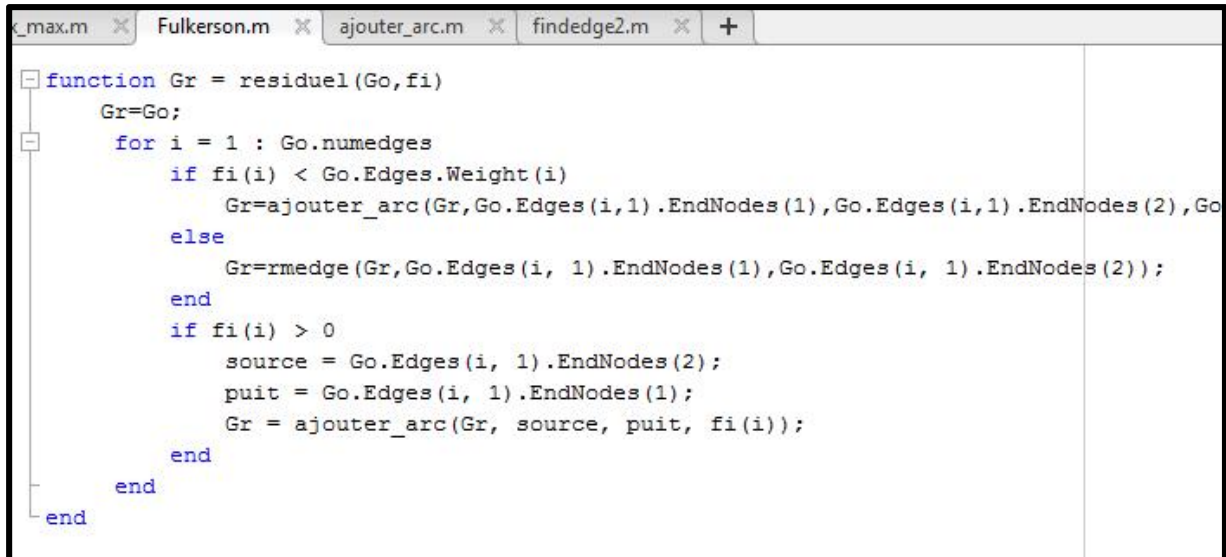
function f=flux_max(G,s,fi)
    f=0;
    for i=1:G.numedges

        a=char(G.Edges(i,1).EndNodes(1));
        if s== a
            f=f+fi(i);
        end
    end
end

```

Figure 2: fonction flux\_max

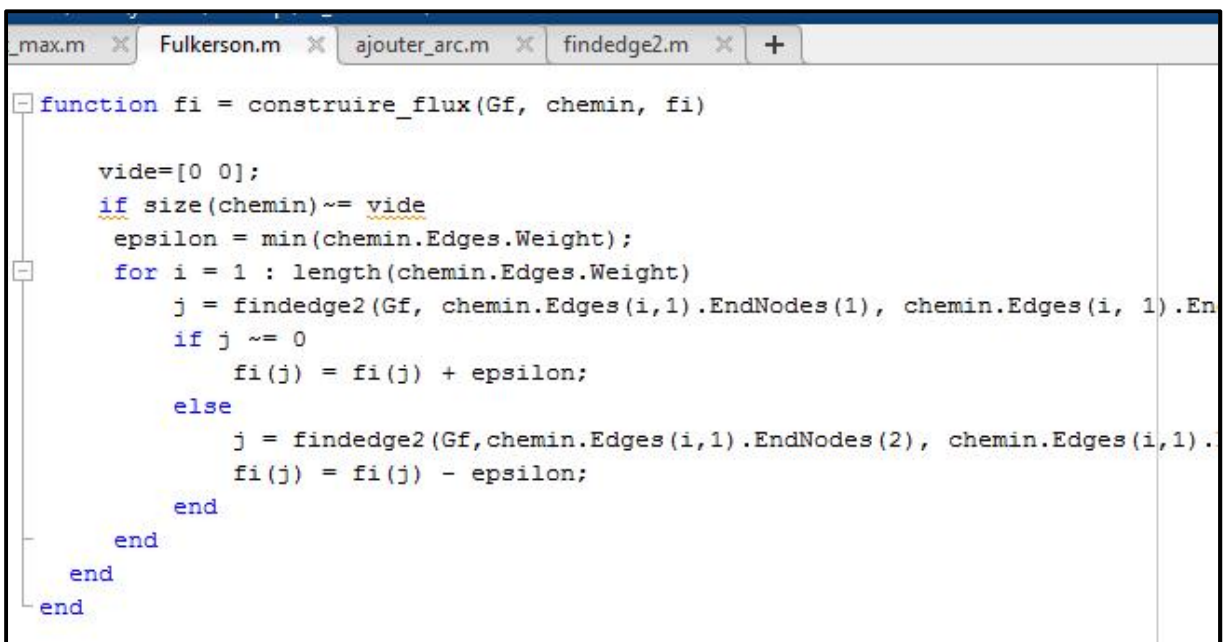
- La fonction Residuel : Elle prend en entrée une copie du graphe résiduel ainsi que le flux à la kième itération pour produire le graphe résiduel. C'est fonction secondaire du fichier script Ford\_Fulkerson.m.



```
function Gr = residuel(Go,fi)
Gr=Go;
for i = 1 : Go.numedges
    if fi(i) < Go.Edges.Weight(i)
        Gr=ajouter_arc(Gr,Go.Edges(i,1).EndNodes(1),Go.Edges(i,1).EndNodes(2),Go.Edges(i,1).Weight(i)-fi(i));
    else
        Gr=rmedge(Gr,Go.Edges(i,1).EndNodes(1),Go.Edges(i,1).EndNodes(2),fi(i));
    end
    if fi(i) > 0
        source = Go.Edges(i,1).EndNodes(2);
        puit = Go.Edges(i,1).EndNodes(1);
        Gr = ajouter_arc(Gr, source, puit, fi(i));
    end
end
end
```

Figure 3:fonction residuel

- La fonction construire\_flux : Elle prend en entrée une copie du graphe initial, le chemin entre la source et la destination dans le graphe résiduel et le flux précédent. C'est une fonction secondaire du fichier script Ford\_Fulkerson.m



```
function fi = construire_flux(Gf, chemin, fi)
vide=[0 0];
if size(chemin)~=vide
    epsilon = min(chemin.Edges.Weight);
    for i = 1 : length(chemin.Edges.Weight)
        j = findedge2(Gf, chemin.Edges(i,1).EndNodes(1), chemin.Edges(i,1).EndNodes(2));
        if j ~= 0
            fi(j) = fi(j) + epsilon;
        else
            j = findedge2(Gf, chemin.Edges(i,1).EndNodes(2), chemin.Edges(i,1).EndNodes(1));
            fi(j) = fi(j) - epsilon;
        end
    end
end
end
```

Figure 4:fonction construire\_flux

- La fonction chercher\_chemin : Elle prend en entrée le graphe résiduel, la source et la destination pour donner un chemin entre les deux. L'implémentation de cette fonction a été rendue facile par l'utilisation d'une fonction native de Matlab 2016b, la fonction shortestpath qui cherche un plus court chemin entre deux destinations dans un graphe. Nous l'avons utilisé parce que si nous cherchons simplement un chemin entre deux localités et qu'une personne nous propose le plus court chemin entre les deux localités, cela nous arrange aussi. C'est une fonction secondaire du fichier script Ford\_Fulkerson.m.



```

function chemin = chercher_chemin(G, s, t)
    chemin = shortestpath(G, s, t);
    subplot(2,2,2),p1=plot(G, 'EdgeLabel',G.Edges.Weight);highlight(p1,chemin, 'EdgeColor','r'),
    vide=[0 0];
    if size(chemin) == vide
        graph_chem = digraph();
    else
        graph_chem = digraph();
        for i = 1 : length(chemin) - 1
            graph_chem = addedge(graph_chem, chemin(i), chemin(i+1), G.Edges.Weight(findedge2
        end
        chemin = graph_chem ;
    end
end

```

Figure 5:fonction chercher\_chemin

- La fonction ajouter\_arc. Elle prend en entrée un graphe, une source, une destination et la capacité et renvoie le graphe augmentée de l'arc défini par la source la destination et du poids. Si l'arc existe, il est modifié sinon il est ajouté. Cette fonction aurait pu ne pas être écrite car il existe une autre fonction qui permet d'ajouter, la fonction addedge, mais qui ne permet pas de modifier, d'où la nécessité de l'avoir écrite. Cette fonction est la fonction principale du fichier script ajouter\_arc.m.

```

function G = ajouter_arc(G, a, b, c)
    d=findedge2(G, a, b);
    if d ~= 0
        G = rmedge(G, a, b);
        G = addedge(G, a, b, c);
    else
        G = addedge(G, a, b, c);
    end
end

```

Figure 6:fonction ajouter\_arc

- La fonction findedge2 : Elle prend en entrée un graphe, l'arc défini par sa source, sa destination et sa capacité et renvoie l'indice de l'arc, s'il existe dans le graphe. Cette fonction aurait aussi pu ne pas être écrite, car il existe une autre fonction, la fonction findedge qui permet de faire le même test mais si et seulement si les sommets en question appartiennent au graphe, d'où la nécessité de l'avoir écrite. Cette fonction est la fonction principale du fichier script findedge2.m.

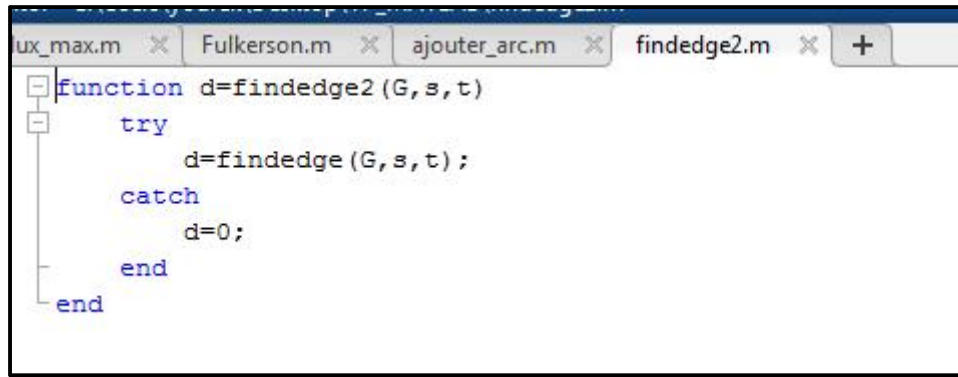


Figure 7: fonction findedge2

### III. Mise en marche de l'application

Nous avons établi deux procédures permettant d'utiliser notre application. Toutefois dans les deux cas il est indispensable d'avoir une version de Matlab postérieure à la version 2015a. La version 2016b que nous avons utilisée serait l'idéale.

#### 1. Importation du projet sous Matlab 2016

Pour pouvoir utiliser convenablement l'application à travers l'importation du projet, la procédure la plus sûre est la suivante :

- Créer un dossier et y mettre tous les fichiers contenu dans l'archive.
- Ouvrir le fichier Ford\_Fulkerson.prj avec Matlab

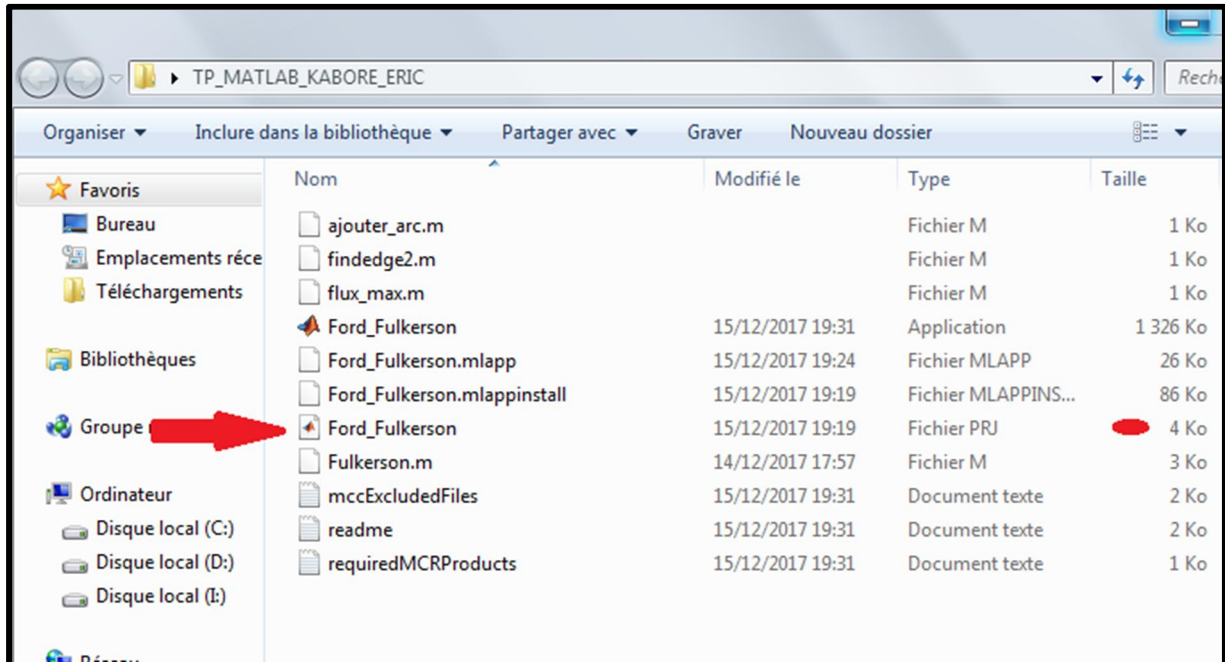


Figure 8: fichier à ouvrir avec Matlab

- Transformer le repertoire courant de Matlab par le dossier qui contient les fichiers du projet

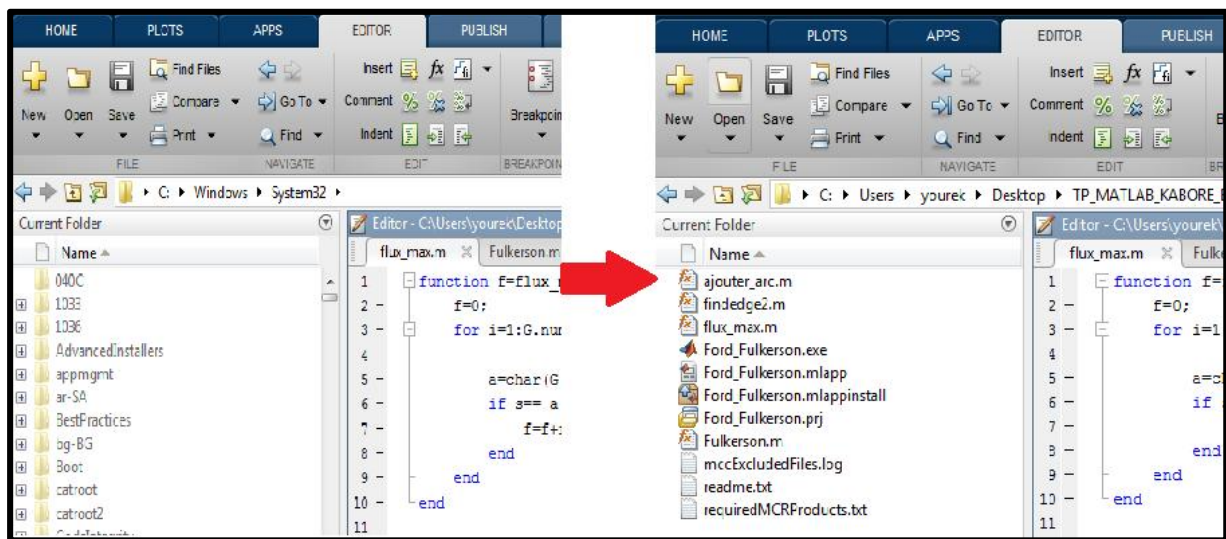


Figure 9: changement du répertoire courant de Matlab

- Lancer l'application en entrant dans l'invite de commande la commande suivante :  
« Ford\_Fulkerson »

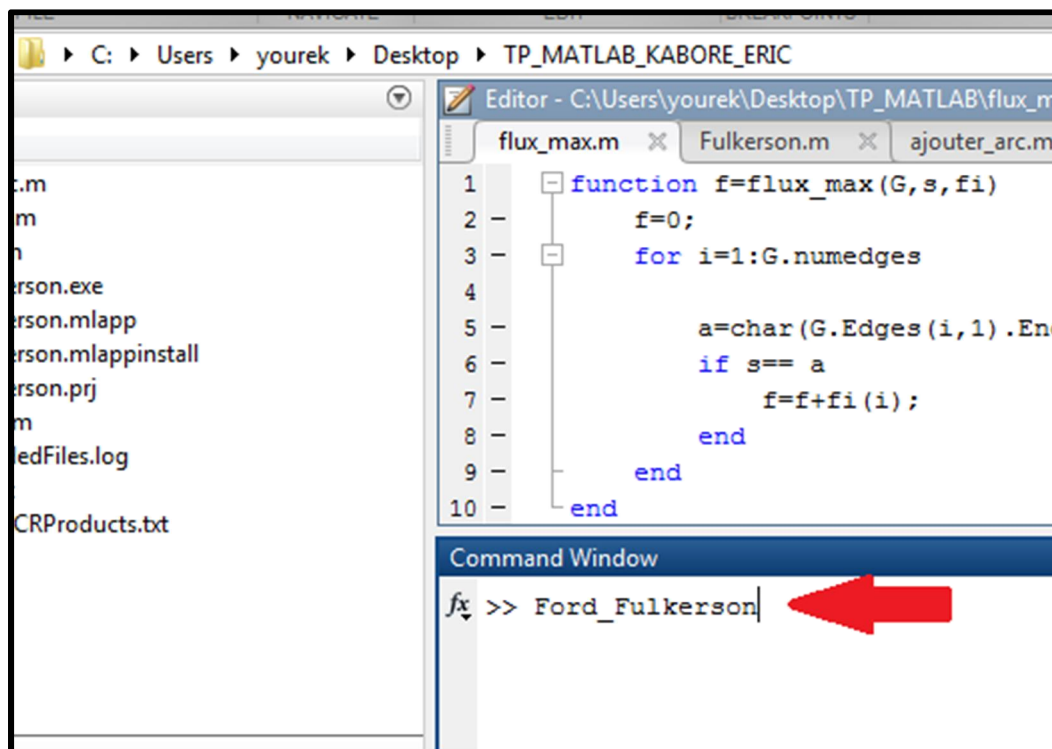


Figure 10: Commande de lancement

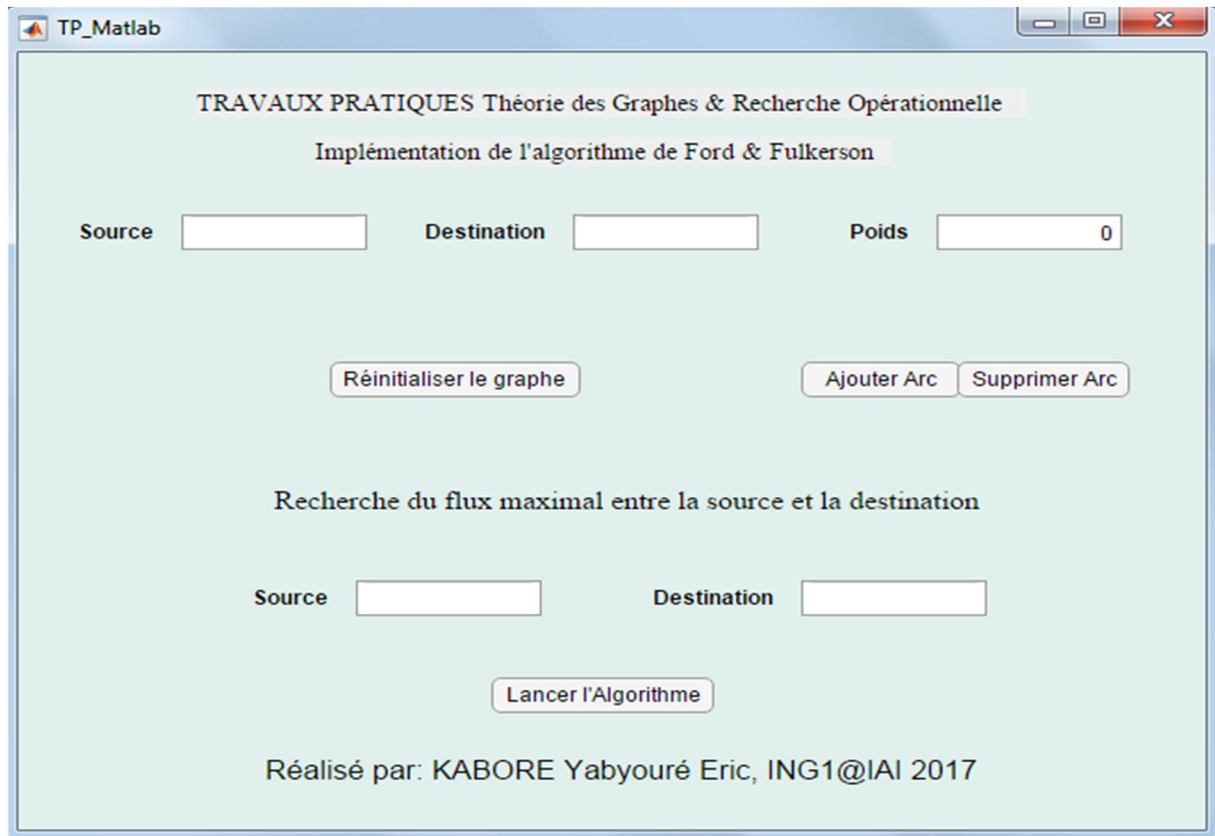


Figure 11:Aperçu de l'application

## 2. Exécution du fichier Ford\_Fulkerson.exe

Cette procédure est beaucoup plus simple que la précédente, il suffit juste double cliquer sur le fichier Ford\_Fulkerson.exe.

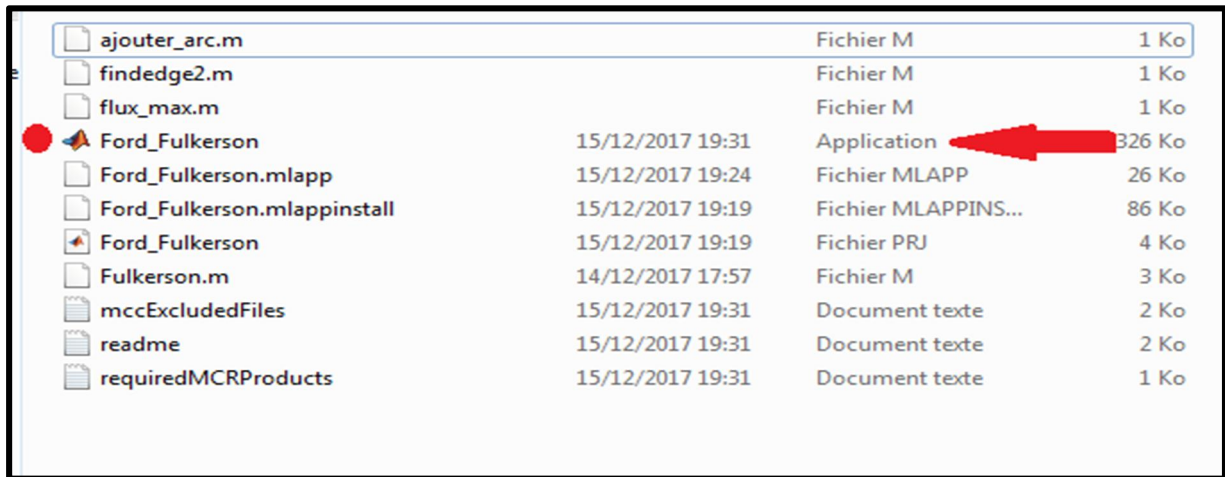


Figure 12: fichier .exe à double-cliquer

## 3. Utilisation de l'application

Pour entrer un graphe, on procède arc par arc. Vous remplissez les valeurs pour un arc et vous cliquer sur "ajouter arc". La fenêtre d'affichage apparaît automatiquement, il faut juste la déplacer et la mettre à côté de la fenêtre de l'application.

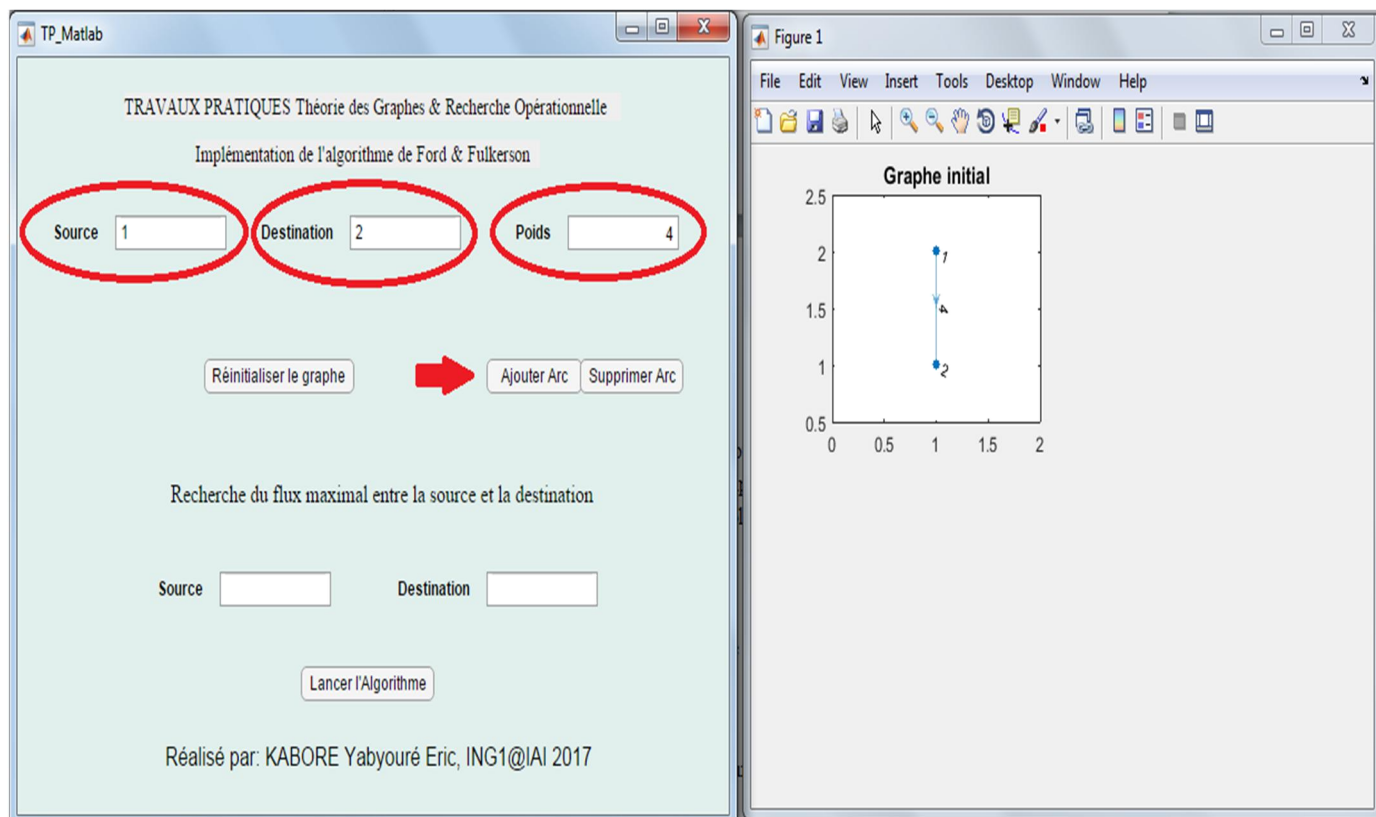


Figure 13: ajout d'arc dans le graphe

- ✚ Une fois qu'on a fini d'entrer le graphe, on entre la source et la destination pour la recherche de flux maximal.
- ✚ On peut passer maintenant au test en cliquant sur lancer l'algorithme pour démarrer la recherche du flux.

Nous prendrons le graphe vu en cours pour réaliser un test. Il s'agit du graphe suivant.

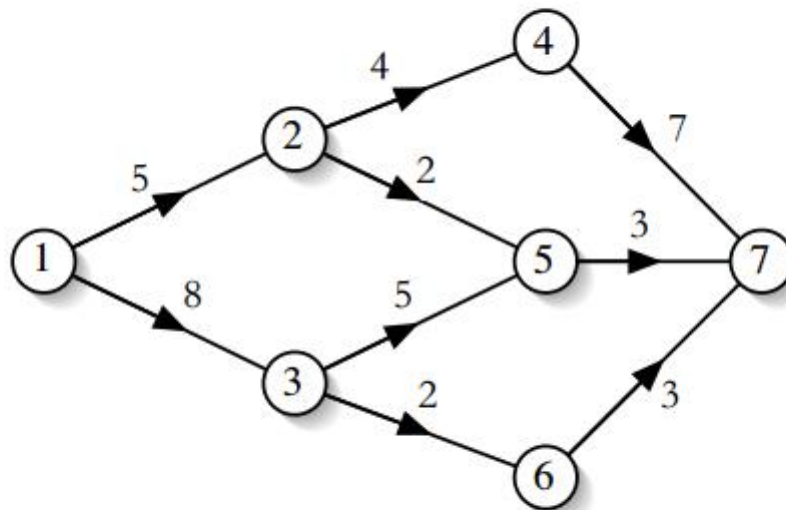



FIG. 11 – Graphe orienté élémentaire

Figure 14: graphe du cours à tester

Recherche du flux maximal entre la source et la destination

Source  Destination



Réalisé par: KABORE Yabyouré Eric, ING1@IAI 2017

Figure 15: bouton de lancement



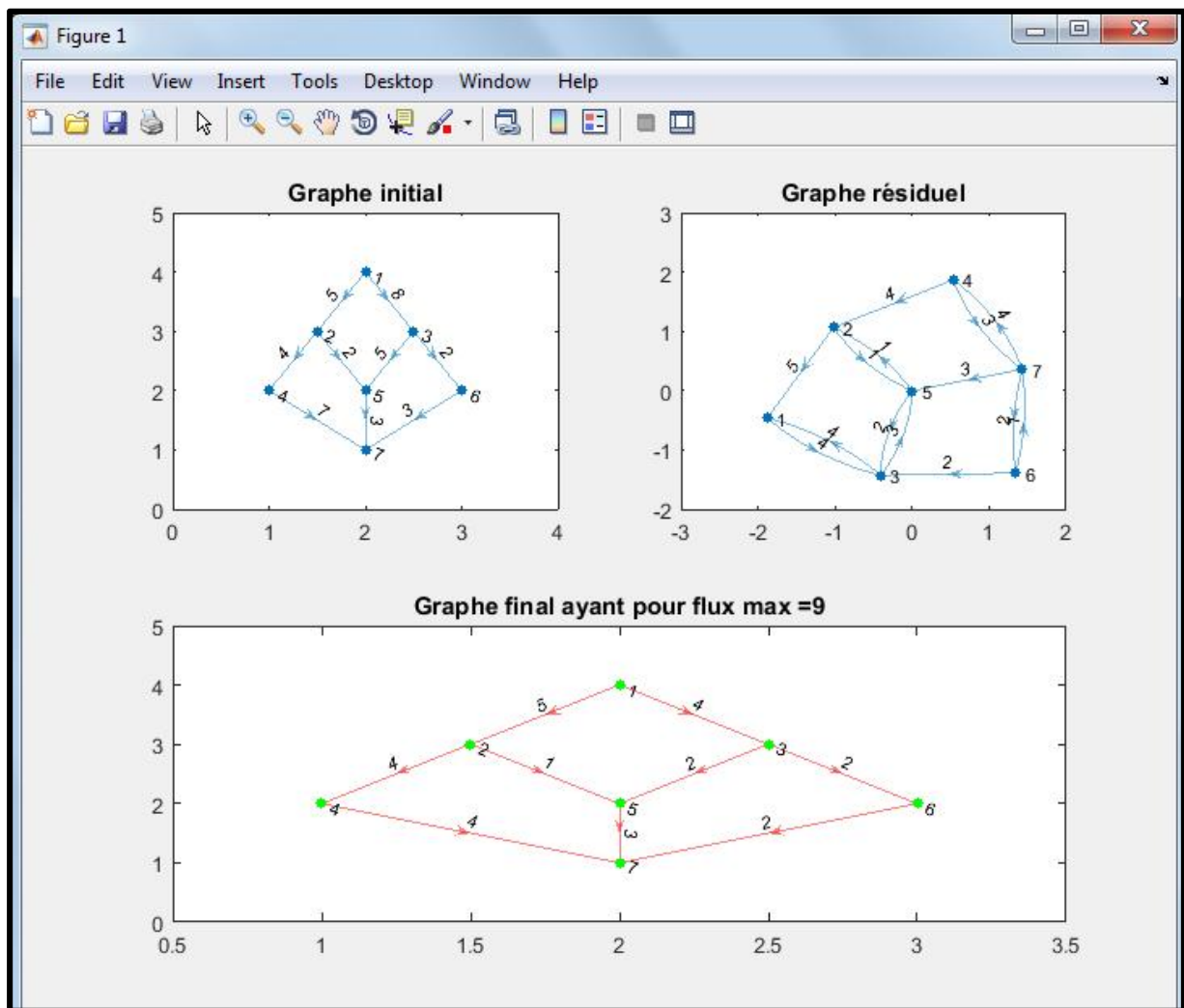


Figure 16: résultat de l'application

#### 4. Remarque

Nous avons évoqué au début que Matlab possède déjà une fonction permettant de résoudre le problème. Il s'agit de la fonction `maxflow`. Elle prend en entrée un graphe, la source, la destination et l'algorithme de recherche du flux. Pour le flux maximal, cet algorithme correspond au mot clé « `augmentpath` ». Nous allons donc résoudre le même problème du graphe ci-dessus mais cette fois avec la fonction native de Matlab.

```

Command Window
fx >> G=digraph([1 2 4 2 5 1 3 6 3],[2 4 7 5 7 3 6 7 5],[5 4 7 2 3 8 2 3 5]);
subplot(1,2,1),H=plot(G,'EdgeLabel',G.Edges.Weight),title('Graphe initial');
[mf,GF] = maxflow(G,1,7,'augmentpath');H.EdgeLabel = {};
text=strcat('Graphe final ayant pour flux max = ',num2str(mf));
subplot(1,2,2),H=plot(G,'EdgeLabel',G.Edges.Weight),title(text);
highlight(H,GF,'EdgeColor','r','LineWidth',2);
st = GF.Edges.EndNodes;
labeledge(H,st(:,1),st(:,2),GF.Edges.Weight);

```

Figure 17: Commandes natives pour le flux maximal

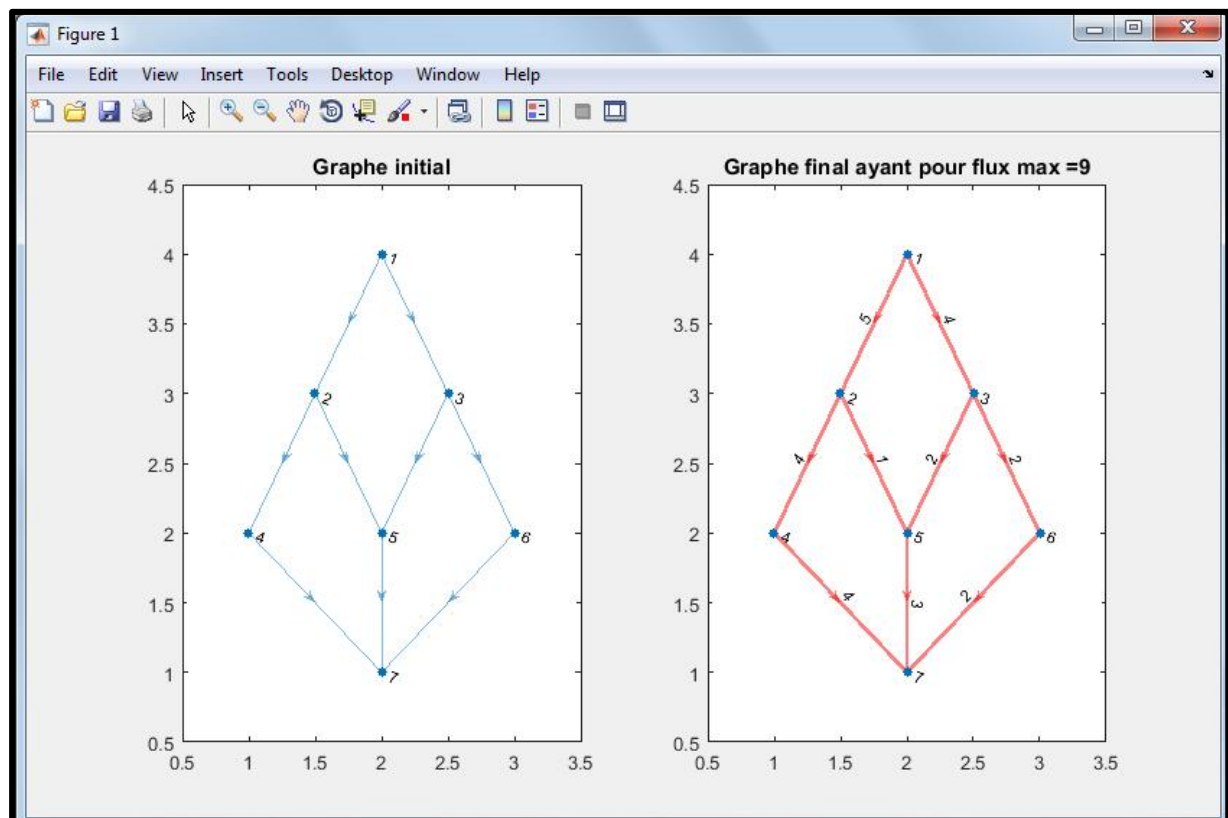


Figure 18: Résultat avec la commande maxflow



## Conclusion

En comparant les deux résultats nous constatons que nous avons le même flux. Cependant notre application a l'avantage de présenter la phase de recherche de chemin dans le graphe résiduel. L'implémentation de cet algorithme n'a pas été facile à cause d'une méconnaissance de certaines spécificités du langage et cela engendraient des erreurs et des bugs. Cependant nous avons relevé le défi et cela nous a énormément été utile dans la mesure où cela nous a permis de vraiment bien comprendre l'algorithme de Ford-Fulkerson, qui était encore très flou dans notre tête jusqu'au jour où nous avons commencé à l'implémenter. Au de-là de cette appréhension, cela a surtout accru nos compétences dans l'environnement Matlab et ce de façons considérable et nous ne pouvons conclure ce rapport que par « Merci Professeur ».