Xavier FRACCARO
Clint Henry NGUEFACK MOMO
Thomas PICOU
Romain PREVOT

Projet de Recherche opérationnelle

Département de Mathématiques Efrei Paris

Année 2024/2025 S6

I1

INTRODUCTION

Dans le cadre de notre cours de Recherche Opérationnelle, nous avons dû étudier et implémenter des algorithmes de résolution de problèmes de flot. Ces problèmes sont fondamentaux en optimisation et trouvent des applications concrètes dans des domaines variés comme le transport, la logistique, les réseaux informatiques ou encore la gestion de ressources par exemple.

L'objectif de ce projet était premièrement de développer des programmes capables de résoudre des problèmes de flot maximal en utilisant les algorithmes de Ford-Fulkerson et Pousser-Réétiqueter, ainsi que de flot de coût minimal pour une valeur donnée de flot. Il nous a fallu ensuite réaliser une étude de la complexité de ces algorithmes en analysant leur comportement selon la taille du réseau et les données d'entrée.

Notre travail s'est articulé autour de plusieurs étapes clés : la compréhension du problème, le codage des algorithmes, les tests sur des jeux de données fournis, la génération de problèmes aléatoires, et enfin, l'analyse des performances algorithmiques.

Ce rapport a pour but de détailler notre travail effectué sur la complexité des algorithmes.

Les algorithmes

Algorithmes pour résolution de flot maximal

Ford-Fulkerson

L'algorithme de Ford-Fulkerson est une méthode utilisée pour résoudre le problème de flux maximal dans un réseau de transport. L'algorithme cherche à augmenter le flux entre une source (point de départ) et un puits (point d'arrivée) tant qu'il existe un chemin augmentant (chemin avec une capacité résiduelle supérieure à 0).

La complexité dépend de la méthode utilisée pour calculer. Dans notre cas, le parcours en largeur (BFS).

La complexité est donc en $O(VE^2)$. V est le nombre de sommets et E le nombre d'arrêtes. Chaque recherche de chemin augmentant avec BFS prend O(E) temps. On peut faire au plus O(VE) augmentations. Chaque fois qu'un chemin augmentant est trouvé, le niveau du puits (sa distance en BFS) augmente ou reste le même. Chaque arête ne peut provoquer que O(V) augmentations avant de devenir inutilisable.

Pousser-réétiqueter

L'algorithme de Pousser Réétiqueter est une méthode utilisée, comme Ford-Fulkerson, pour résoudre le problème de flux maximal dans un réseau de transport. Mais Contrairement à Ford-Fulkerson, qui cherche des chemins de la source au puits, Pousser-Réétiqueter travaille localement, il pousse du flux en excès depuis les sommets, tout en ajustant des étiquettes de hauteur pour guider le flux vers le puits.

La complexité est en $O(V^2E)$ dans sa version simple. V est le nombre de sommets et E le nombre d'arrêtes. Pour Réétiqueter, chaque sommet (sauf source et puits) peut voir sa hauteur augmenter au plus 2V - 1 fois. Il y a donc $O(V^2)$ réétiquetages au total.

Algorithmes pour résolution de coût minimal

Bellman

L'algorithme de Bellman-Ford est une méthode utilisée pour trouver les plus courts chemins depuis une source vers tous les sommets d'un graphe, en prenant en compte même si certaines arêtes ont des poids négatifs.

La complexité en temps est en O(VE) (Toujours avec V le nombre de sommets et E le nombre d'arrêtes.) Chaque itération parcourt toutes les arêtes soit O(E), et on fait V - 1
La mesure du temps

Pour étudier la complexité nous avons générer 100 graphes aléatoires sans boucles par valeurs de n et avons testé nos 3 algorithmes sur chacun de ces graphes. Nous avons pris des mesures de temps entre le début de chaque itération des algorithmes de flow et leurs fins. Nous avons pu faire les constats suivants :

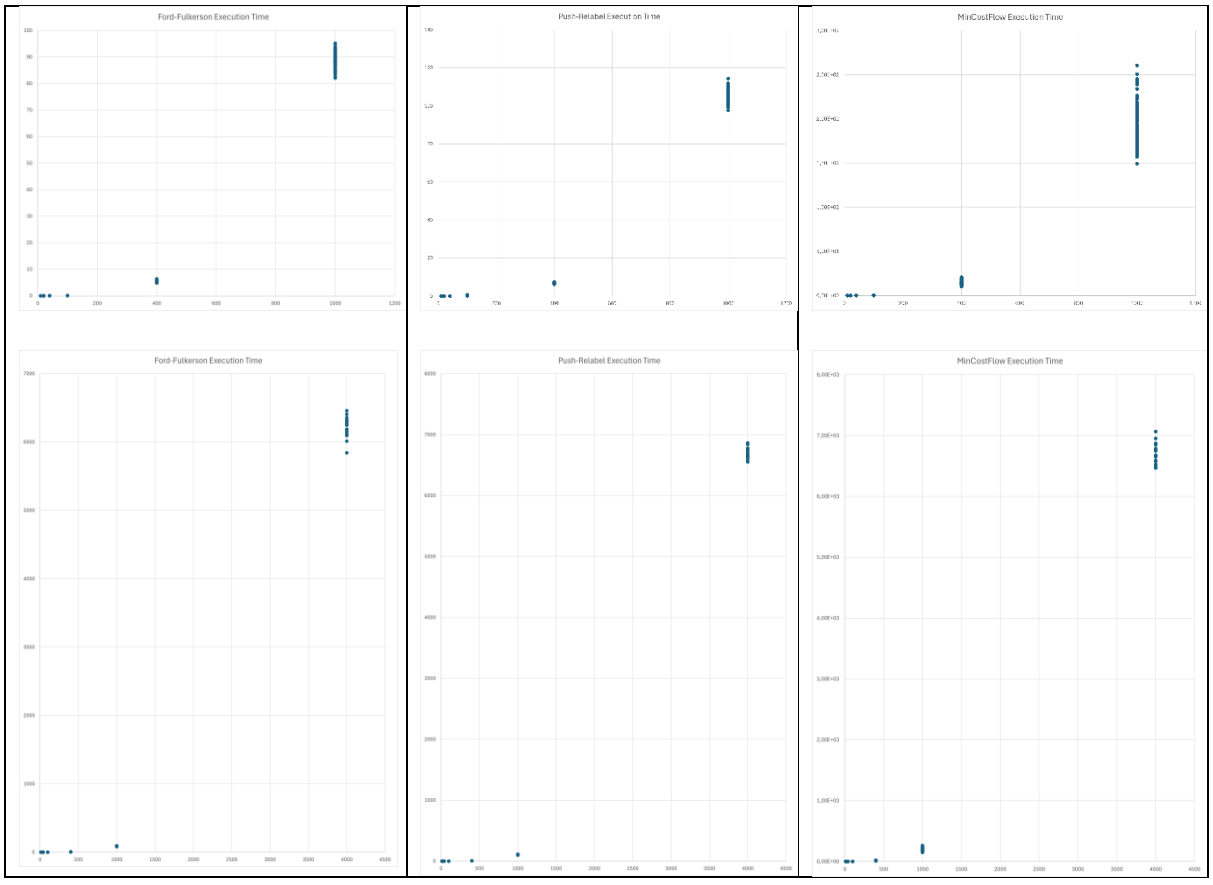
Pour une itération de n = 4000 le temps d'exécution a été de d'environ 11 heures et 40 minutes, si l'on prend une moyenne de 11h30 d'exécution par itération, on serait à environ à 1150 heures soit 1 mois et 18 jours pour tout exécuter.

Pour n = 10000, après 34 heures d'exécution, le premier algorithme Ford-Fulkerson n'a pas fini de s'exécuter pour la première itération. Itérations. Le total est donc O((V - 1) × E) Soit O(VE).

Etude de la complexité

Il nous était demandé de mesurer la complexité pour chacun des algorithmes présentés ci-dessus.

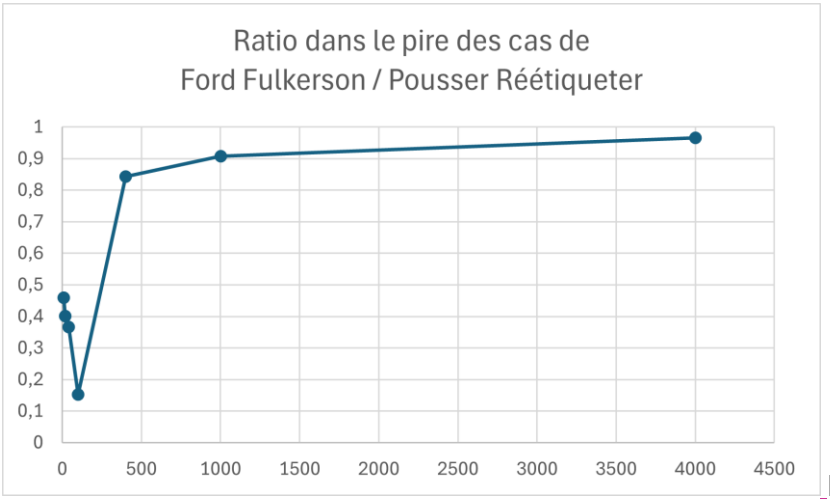
Le nuage de points



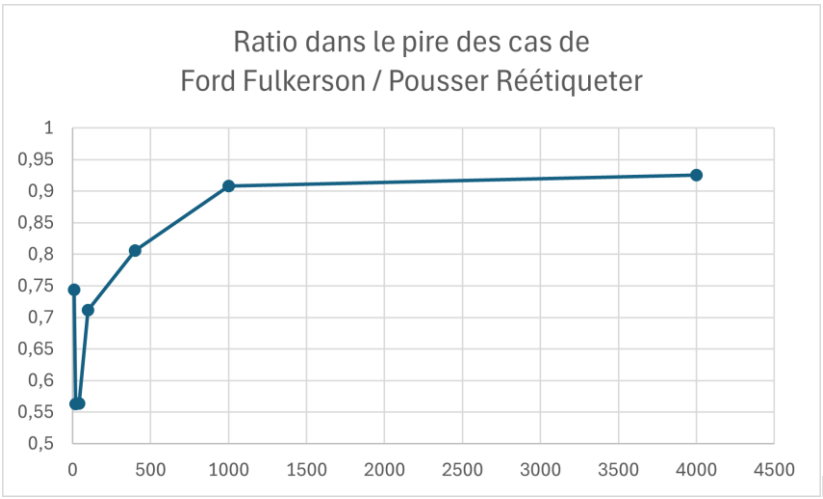
On remarque bien sur les nuages de points que pour chaque n plus grand, les valeurs sont plus dispersées et que la moyenne de ces valeurs augmente. On note aussi que la cette moyenne semble augmenter en suivant une courbe exponentielle. On peut donc formuler l'hypothèse que les algorithmes sont de complexité O(k^n).

Comparaison de la complexité dans le pire des cas

En utilisant les valeurs moyennes et les valeurs max des temps d'itérations pour chaque valeur de n, nous avons obtenu les graphes suivants :



MAX



MOYENNE

Ce résultat nous semble logique car bien que les complexités des algorithmes soit semblable ($O(V^2E)$ et $O(VE^2)$). Pour de grandes valeurs de n, la différence est bien plus forte. De plus, dans la majorité des cas, le nombre d'arêtes est bien supérieur au nombre de sommet rendant l'algorithme Pousser-réétiqueter plus efficace que Ford Fulkerson.

Ainsi, on constate que plus la valeur de n augmente, plus le ratio augmente.