EL MISSIRY Tarek, AMRANI Wassim
BOUKHAMSA Ilies, AISSANI Yanis

Rapport : Analyse des Algorithmes de Flot Maximaux

1. Introduction

Les algorithmes de flot maximal, tels que **Ford-Fulkerson** et **Pousser-Réétiqueter (Push-Relabel)**, sont essentiels dans des domaines comme les réseaux de transport et l'optimisation des flux. Ce rapport analyse leurs performances sur des graphes variés, en comparant les résultats expérimentaux aux prédictions théoriques.

L'objectif est d'identifier les forces et limites de chaque algorithme dans différents contextes, tout en éclairant les conditions où ils excellent. Cette étude fournit une vision claire de leurs applications pratiques et de leur efficacité.

2. Méthodologie

1. Description des algorithmes :
- Ford-Fulkerson : Utilise un chemin augmentant pour incrémenter le flot jusqu'à atteindre la capacité maximale. Cet algorithme est itératif et s'appuie sur la recherche de chemins augmentants successifs dans le graphe résiduel. Sa complexité temporelle est $O(E \cdot |f^*|)$, où E est le nombre d'arêtes et $|f^*|$ est la valeur du flot maximal.
- Pousser-Réétiqueter (Push-Relabel) : Cet algorithme repose sur des opérations locales au niveau des sommets (« poussées » et « réétiquetages »), visant à redistribuer le flot dans le graphe jusqu'à atteindre l'optimalité. Il a une complexité temporelle de $O(V^2 \cdot E)$, où V est le nombre de sommets et E le nombre d'arêtes.

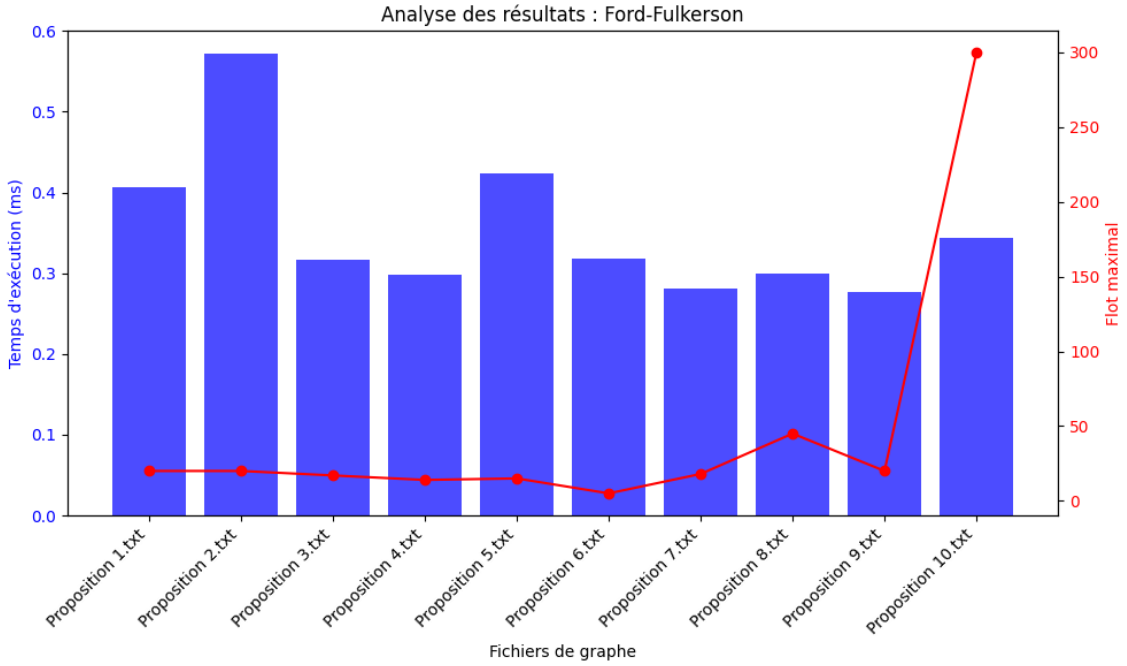
2. Données et outils :
- Les tests ont été effectués sur 10 fichiers (« Proposition 1.txt » à « Proposition 10.txt »), chacun représentant un graphe orienté avec des capacités définies sur les arêtes.
- Python et NetworkX ont été utilisés pour implémenter et exécuter les algorithmes.
- Chaque algorithme a été exécuté 1000 fois pour chaque graphe afin de calculer un temps d'exécution moyen fiable.

3. Critères d'évaluation :
- Temps d'exécution : Mesuré en millisecondes.
- Flot maximal calculé : Mesure de l'efficacité des algorithmes à atteindre la capacité optimale du graphe.

3. Résultats

Ford-Fulkerson

- Proposition 2 montre un temps d'exécution plus long que Proposition 10, bien que le graphe soit plus grand. Cela s'explique par une structure clairsemée, avec de nombreux chemins augmentants nécessitant un plus grand nombre d'itérations.
- Proposition 10, malgré un flot maximal élevé (300), reste rapide. Cela est attribué à une structure dense et à un graphe qui favorise des chemins directs entre la source et le puits.

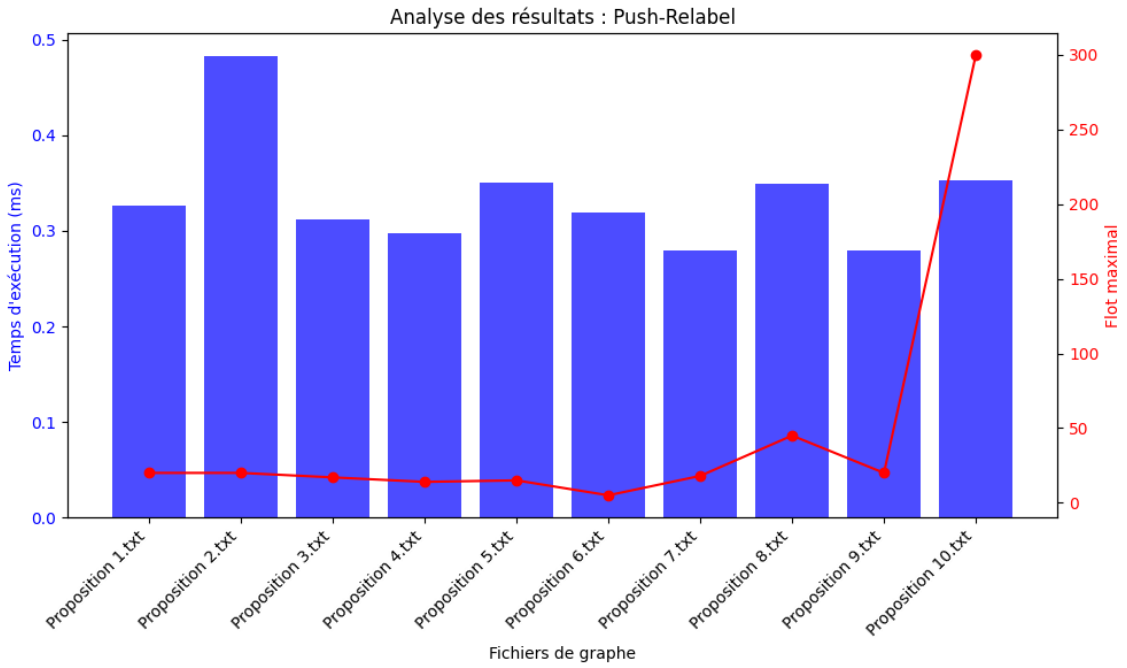



```
Algorithm: Ford-Fulkerson, File: Proposition 1.txt, Nodes: 8, Edges: 13, Max Flow: 20, Time: 0.000406s
Algorithm: Ford-Fulkerson, File: Proposition 2.txt, Nodes: 10, Edges: 19, Max Flow: 20, Time: 0.000572s
Algorithm: Ford-Fulkerson, File: Proposition 3.txt, Nodes: 8, Edges: 12, Max Flow: 17, Time: 0.000316s
Algorithm: Ford-Fulkerson, File: Proposition 4.txt, Nodes: 8, Edges: 12, Max Flow: 14, Time: 0.000297s
Algorithm: Ford-Fulkerson, File: Proposition 5.txt, Nodes: 8, Edges: 13, Max Flow: 15, Time: 0.000423s
Algorithm: Ford-Fulkerson, File: Proposition 6.txt, Nodes: 6, Edges: 9, Max Flow: 5, Time: 0.000318s
Algorithm: Ford-Fulkerson, File: Proposition 7.txt, Nodes: 6, Edges: 10, Max Flow: 18, Time: 0.000281s
Algorithm: Ford-Fulkerson, File: Proposition 8.txt, Nodes: 7, Edges: 12, Max Flow: 45, Time: 0.000299s
Algorithm: Ford-Fulkerson, File: Proposition 9.txt, Nodes: 6, Edges: 11, Max Flow: 20, Time: 0.000276s
Algorithm: Ford-Fulkerson, File: Proposition 10.txt, Nodes: 7, Edges: 11, Max Flow: 300, Time: 0.000344s
```

Moyenne de temps : 0.3532ms

Pousser-Réétiqueter (Push-Relabel)

- Push-Relabel présente des temps d'exécution plus stables que Ford-Fulkerson, car il n'est pas directement influencé par la valeur du flot maximal $|f^*|$.
- Proposition 2 reste plus lente, probablement en raison d'une densité plus faible et d'un traitement local plus long pour redistribuer le flot.
- Proposition 10 met en avant la robustesse de Push-Relabel sur des graphes denses, avec des temps très proches de ceux de Ford-Fulkerson malgré la complexité théorique supérieure.




```
Algorithm: Push-Relabel, File: Proposition 1.txt, Nodes: 8, Edges: 13, Max Flow: 20, Time: 0.000327s
Algorithm: Push-Relabel, File: Proposition 2.txt, Nodes: 10, Edges: 19, Max Flow: 20, Time: 0.000483s
Algorithm: Push-Relabel, File: Proposition 3.txt, Nodes: 8, Edges: 12, Max Flow: 17, Time: 0.000312s
Algorithm: Push-Relabel, File: Proposition 4.txt, Nodes: 8, Edges: 12, Max Flow: 14, Time: 0.000298s
Algorithm: Push-Relabel, File: Proposition 5.txt, Nodes: 8, Edges: 13, Max Flow: 15, Time: 0.000350s
Algorithm: Push-Relabel, File: Proposition 6.txt, Nodes: 6, Edges: 9, Max Flow: 5, Time: 0.000319s
Algorithm: Push-Relabel, File: Proposition 7.txt, Nodes: 6, Edges: 10, Max Flow: 18, Time: 0.000279s
Algorithm: Push-Relabel, File: Proposition 8.txt, Nodes: 7, Edges: 12, Max Flow: 45, Time: 0.000349s
Algorithm: Push-Relabel, File: Proposition 9.txt, Nodes: 6, Edges: 11, Max Flow: 20, Time: 0.000280s
Algorithm: Push-Relabel, File: Proposition 10.txt, Nodes: 7, Edges: 11, Max Flow: 300, Time: 0.000353s
```

Moyenne de temps : 0.335ms

4. Discussion

1. Observations principales :

Ford-Fulkerson :

- Son temps d'exécution est fortement dépendant de la structure du graphe et du nombre de chemins augmentants.
- Les graphes plus simples (dense et avec des capacités élevées) favorisent des temps d'exécution courts.

Push-Relabel :

- Présente une meilleure stabilité des temps d'exécution, indépendamment de la valeur du flot maximal.
- Adapté aux graphes denses ou complexes, où il offre une gestion efficace des flots locaux.

2. Analyse de la complexité :

Ford-Fulkerson :

Les résultats montrent que Ford-Fulkerson est fortement influencé par le flot maximal $|f^*|$. Par exemple, Proposition 2, avec ses nombreux chemins augmentants, nécessite plus d'itérations pour atteindre le flot maximal, ce qui prolonge le temps d'exécution. En revanche, pour Proposition 10, la structure dense et les capacités élevées permettent à l'algorithme d'exploiter des chemins directs, réduisant ainsi le temps d'exécution.

Push-Relabel :

Bien que théoriquement plus coûteux avec $O(V^2 \cdot E)$, Push-Relabel montre une robustesse pratique, notamment sur des graphes denses comme Proposition 10. Cette stabilité est due à sa gestion locale des flots, qui distribue efficacement les excès entre les sommets. En revanche, pour des graphes plus clairsemés comme Proposition 2, les calculs liés au réétiquetage local peuvent légèrement allonger les temps d'exécution, mais cette différence reste modérée.

3. Limites des tests :

- Les graphes testés (6 à 10 sommets) ne permettent pas de capturer pleinement les différences asymptotiques entre les deux algorithmes.
- Les variations de structure influencent davantage Ford-Fulkerson, ce qui complique les comparaisons directes.

4. Améliorations potentielles :

- Tester les algorithmes sur des graphes artificiellement plus grands (à partir de 50 nœuds).
- Explorer des graphes extrêmes : à densité minimale ou maximale.

5. Conclusion

Cette étude a comparé les performances de Ford-Fulkerson et Pousser-Réétiqueter sur des graphes variés, mettant en lumière leurs forces et limites respectives. Ford-Fulkerson s'est révélé efficace sur des graphes simples et denses, mais ses performances peuvent être ralenties par des configurations avec de nombreux chemins augmentants, comme observé dans Proposition 2. Sa dépendance au flot maximal $|f^*|$ correspond bien à sa complexité théorique $O(E \cdot |f^*|)$.

En revanche, Pousser-Réétiqueter a démontré une stabilité remarquable, particulièrement sur des graphes denses et complexes, grâce à sa gestion locale des flots. Malgré une complexité théorique $O(V^2 \cdot E)$ plus élevée, il offre des performances constantes, comme observé pour Proposition 10.

Ces résultats confirment que le choix de l'algorithme dépend de la structure du graphe. Ford-Fulkerson est adapté aux graphes clairsemés et simples, tandis que Push-Relabel excelle sur des graphes denses ou avec des flots maximaux élevés. Pour approfondir cette analyse, des tests sur des graphes plus grands seraient nécessaires pour observer des différences asymptotiques plus marquées.