CHEN Camille
LIN Benoit
TANNOUS Gabriel
SU Henri
LAOUITI Kais

Rapport SM602 - RECHERCHE OPERATIONNELLE

Intro

Dans le cadre de notre projet de recherche opérationnelle et d'algorithmique, nous avons étudié et implémenté trois algorithmes classiques de flot dans les graphes : Ford-Fulkerson, Pousser-Réétiqueter et Bellman-Ford. Ces algorithmes visent à faire circuler un flot entre une source et un puits, en respectant les contraintes de capacité des arêtes, et, dans le cas du flot à coût minimal, en optimisant également un coût associé à chaque passage.

Après avoir validé leur bon fonctionnement sur des cas simples, nous avons mené une analyse expérimentale de leur complexité, en nous concentrant sur l'évolution du temps d'exécution en fonction de la taille du graphe (nombre de sommets n). Pour cela, nous avons généré aléatoirement plusieurs graphes pour différentes valeurs de n : 10, 20, 40 et 100. Pour chaque taille, 100 mesures ont été effectuées, ce qui nous a permis de tracer des nuages de points et d'extraire les temps maximaux observés, afin d'étudier la complexité dans le pire des cas.

Ce rapport présente donc la méthodologie suivie, les résultats expérimentaux, ainsi qu'une comparaison détaillée des performances des trois algorithmes testés.

Complexité

La complexité d'un algorithme désigne la quantité de ressources (généralement le temps ou la mémoire) qu'il consomme pour résoudre un problème en fonction de la taille de l'entrée.

Dans notre projet, on s'intéresse surtout à la complexité en temps, c'est-à-dire au temps d'exécution nécessaire pour que l'algorithme calcule un flot maximal ou un flot à coût minimal, en fonction du nombre de sommets n.

Méthodologie

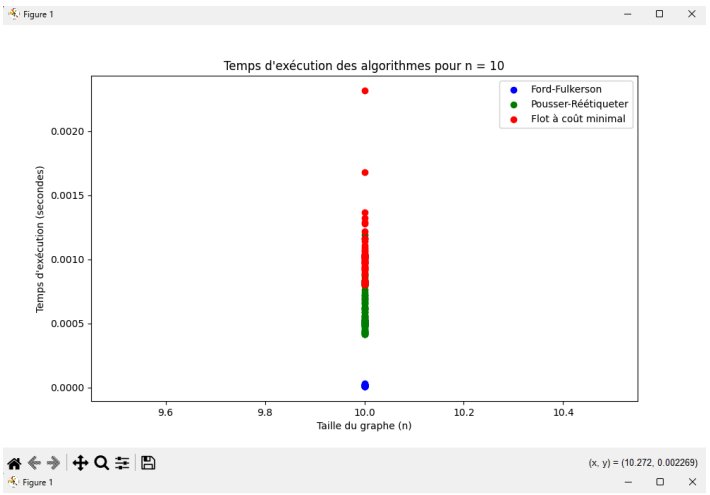
Nous avons cherché à analyser comment la complexité des algorithmes évolue lorsque la taille du graphe augmente. Pour cela, nous avons fait varier le nombre de sommets n et généré avec la fonction *generate_random_graphe(n)*, pour chaque valeur de n, des graphes aléatoires. Nous avons ensuite mesuré les temps d'exécution (avec les fonctions *time_execution)* des trois algorithmes sur ces graphes, puis comparé les résultats obtenus aux complexités théoriques attendues. Cette approche nous a permis de confronter les performances observées en pratique avec les bornes théoriques connues dans la littérature algorithmique.

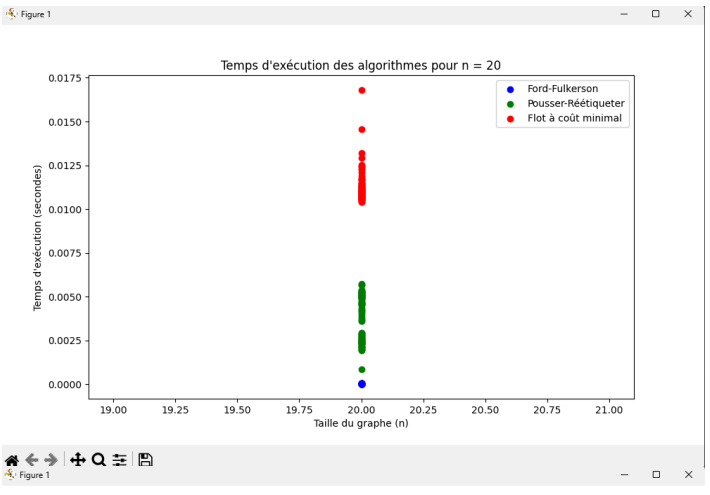
CHEN Camille
LIN Benoit
TANNOUS Gabriel
SU Henri
LAOUITI Kais

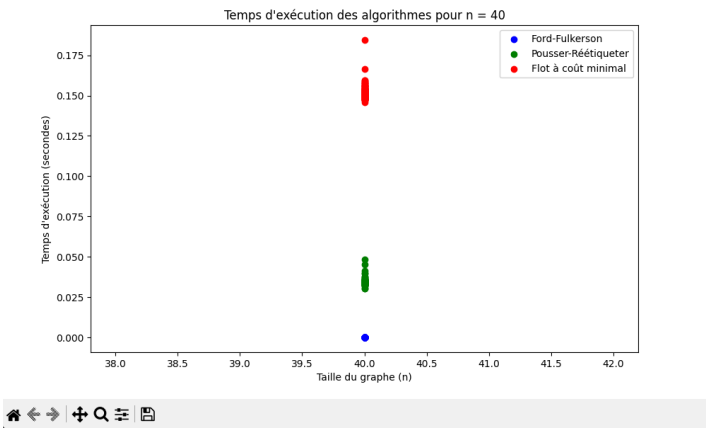
Pour chaque taille de graphe n = 10, 20, 40, 100, 400, 1000 (nous nous sommes arrêté à n = 1000), nous avons effectué 100 tests afin d'obtenir une mesure représentative, que nous avons visualisée sous forme de nuages de points.

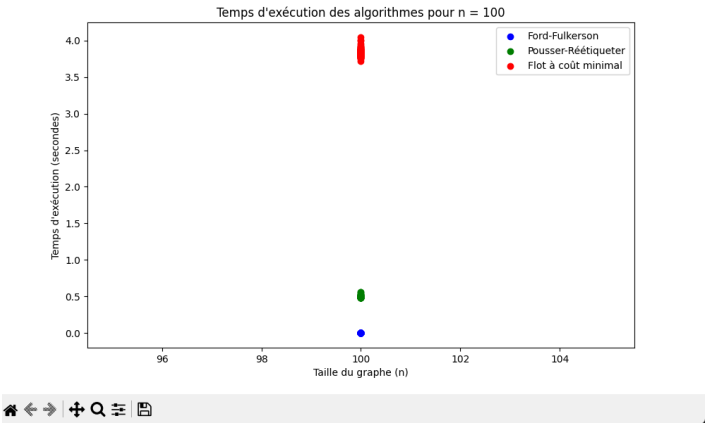
Analyse de données

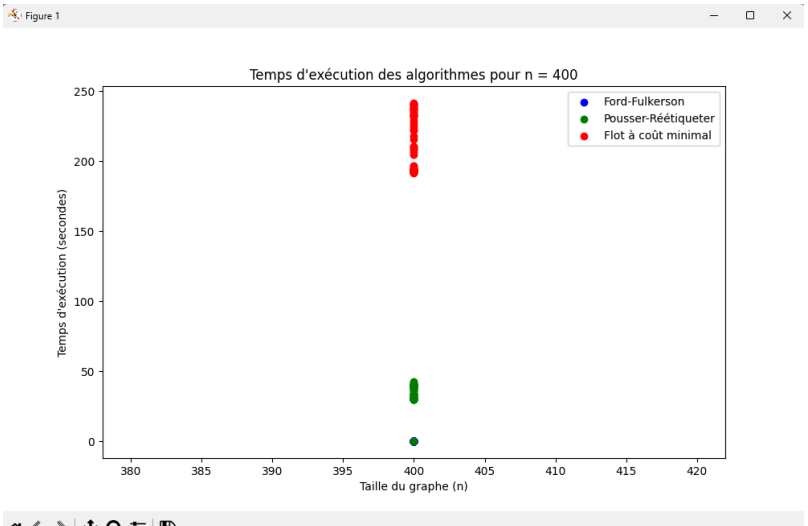
1. Les nuages de points







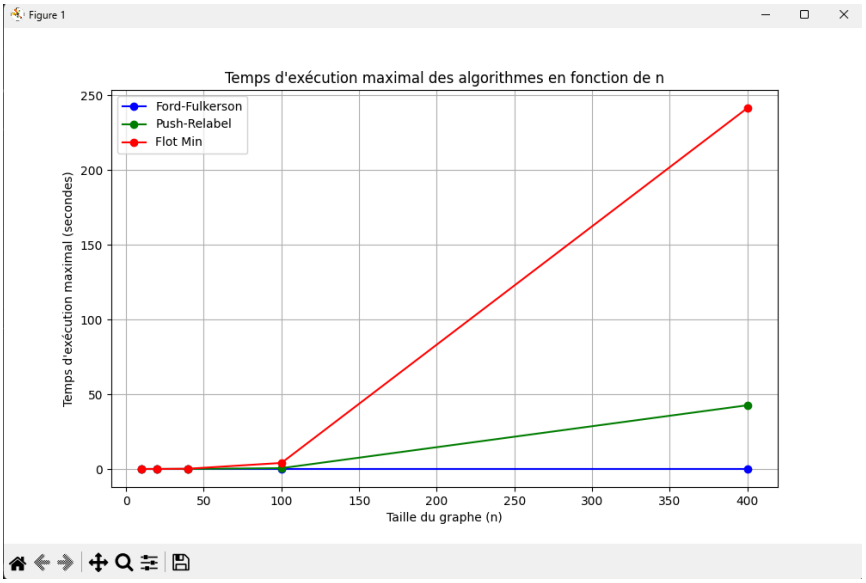




CHEN Camille
LIN Benoit
TANNOUS Gabriel
SU Henri
LAOUITI Kais

Voici les nuages de points pour chaque n jusqu'à 400. La simulation n'a pas pu être faite pour n = 1 000, n = 4 000 et n = 10 000 car le programme a planté. En effet, le traitement pour de tailles aussi grandes demandait trop de calculs pour nos ordinateurs.
Pourtant, on peut très bien analyser la tendance du temps d'exécution grâce à ces nuages de points : plus la taille du graphe est grande, plus le temps d'exécution des algorithmes est grand. On peut cependant remarquer que l'algorithme Ford-Fulkerson est le seul qui a tendance à tendre vers le 0. Il s'agit de l'algorithme le plus rapide. L'algorithme Pousser-Réétiqueter est lui aussi plutôt rapide, avec moins de 50 secondes pour un graphe de taille 400. Par contre, on voit que l'algorithme de Bellman-Ford est le plus long. Avec un temps d'exécution de 250 secondes pour n = 400, on comprend très vite que c'est lui qui demandait le plus de calcul et à cause duquel nous n'avons pas pu continuer à simuler pour les différents n supérieurs à 1 000.

2. La complexité dans le pire des cas par algorithme



Le graphique ci-dessus montre le temps d'exécution maximal des algorithmes en fonction de la taille du graphe n. On observe que :

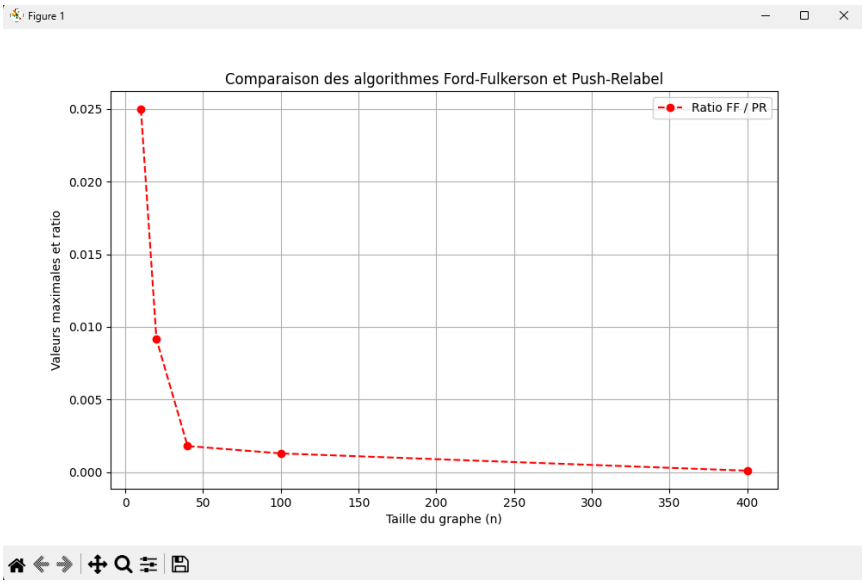
- Ford-Fulkerson a un temps d'exécution très faible, presque constant très proche de 0 jusqu'à n = 400. Pourtant, sa complexité dans le pire des cas étant polynomiale O(nk) / exponentielle O(2n). Donc son comportement dans le graphique est trompeur, puisque son temps dépend fortement du choix des chemins augmentants : il devient inefficace sur des grands graphes.

- Push-Relabel présente une croissance modérée et régulière, ce qui correspond à une complexité quadratique O(n²). Cela correspond bien à sa complexité théorique dans le pire des cas qui indique quadratique. Il est l'algorithme qui s'est révélé le plus

rapide et le plus stable.

- Bellman-Ford a une croissance très rapide du temps d'exécution : il passe de quelques secondes à près de 250 secondes pour n = 400. Ce comportement indique une complexité polynomiale O(nk).

3. Comparaison de la complexité dans le pire des cas



Le rapport entre les deux algorithmes est élevé pour les petites tailles de graphe. Cela montre que Ford-Fulkerson a un temps d'exécution assez similaire à Push-Relabel. À mesure que la taille du graphe augmente, le rapport diminue de manière significative, se rapprochant de plus en plus de 0. On peut observer que Push-Relabel devient de moins en moins efficace par rapport à Ford-Fulkerson pour les graphes de plus grande taille. On peut donc en conclure que, pour des graphes de petites tailles (allant jusqu'à n=100 environ), les deux algorithmes seront aussi efficaces l'un que l'autre. Cependant, pour des graphes plus grands, il est préférable de choisir Ford-Fulkerson pour un temps d'exécution plus rapide.

Conclusion

Au terme de notre étude, nous pouvons retenir que chaque algorithme présente des avantages spécifiques selon le type de graphe traité et la taille de l'entrée. Ford-Fulkerson, bien que très rapide dans nos simulations, peut devenir inefficace dans certains cas à cause de sa complexité exponentielle théorique. Push-Relabel a montré une bonne stabilité et des performances régulières, ce qui en fait un bon choix pour des graphes de taille moyenne. Enfin, Bellman-Ford s'est révélé le plus lent, avec un temps d'exécution qui explose rapidement, ce qui le rend peu adapté pour les grands graphes dans un contexte de flot.