Projet MOGPL MU4IN200

Rapport

Amélioration de Bellman Ford

Étudiants :
Anyes TAFOUGHALT
Racha Nadine DJEGHALI

Encadré par :
Patrice PERNY

9 décembre 2023

Table des matières

1 Introduction

L'algorithme de Bellman-Ford joue un rôle central dans la résolution des problèmes de recherche de plus courts chemins au sein de graphes orientés pondérés, capable de traiter des poids d'arêtes aussi bien positifs que négatifs. La performance de cet algorithme est étroitement liée à l'ordre dans lequel les sommets sont choisis pour les mises à jour itératives de la valeur de leur plus court chemin.

Dans le cadre de ce projet, notre objectif principal est d'optimiser le temps d'exécution de l'algorithme de Bellman-Ford, en particulier dans des scénarios où les instances ne sont pas parmi les plus défavorables.

Afin d'atteindre cette optimisation, nous introduisons une phase de prétraitement dans l'algorithme de Bellman-Ford. Cette phase, basée sur une collection d'instances typiques, vise à déterminer un ordre optimal pour l'examen des sommets. Nous autorisons ainsi un coût de prétraitement plus important, tout en restant polynomial, afin de réduire le temps d'exécution spécifique à chaque instance.

Le problème étudié concerne un graphe orienté pondéré sans circuits de poids négatif. Nous démontrons que le choix stratégique de l'ordre des sommets peut considérablement réduire le nombre d'itérations nécessaires à la convergence de l'algorithme de Bellman-Ford. Pour résoudre ce problème, nous employons l'algorithme GloutonFas, décrit dans le sujet du projet, une approche gloutonne visant à minimiser la violation d'un ordre donné. Notre étude s'appuie sur des exemples de graphes avec des poids d'arcs issus de distributions variées. De plus, nous explorons l'applicabilité de notre approche à travers des expériences et des comparaisons avec des méthodes aléatoires, évaluant ainsi l'efficacité de notre prétraitement.

2 Implémentation

2.1 L'algorithme de Bellman-Ford

— La fonction **"bellman_ford(G , source, ordre)"** implémente l'algorithme de Bellman-Ford pour trouver les plus courts chemins depuis un nœud source dans un graphe orienté pondéré. Elle permet de spécifier l'ordre de traitement des nœuds, ajoutant ainsi la possibilité de personnaliser la séquence dans laquelle les nœuds sont examinés à chaque itération. L'algorithme itère jusqu'à convergence ou jusqu'à ce qu'un nombre maximal d'étapes, égal au nombre de nœuds dans le graphe, soit atteint. Notre méthode renvoie une liste de paires qui indique pour chaque nœud la distance du chemin le plus court associé à ce nœud et son prédécesseur dans ce chemin.

2.2 L'algorithme GloutonFas

— La fonction **"GloutonFas(G)"** a été implémentée en se basant sur le pseudocode fourni dans l'énoncé du projet. Elle utilise un graphe G pour déterminer un ordre optimal de traitement, accélérant ainsi la convergence de l'algorithme de Bellman-Ford. La méthode identifie successivement les sources et les puits du graphe, puis sélectionne un sommet ayant la plus grande différence entre ses poids entrants et sortants. En retirant itérativement ces sommets du graphe, la fonction génère un

ordre de traitement qui contribue à accélérer le processus de convergence de l'algorithme de Bellman-Ford, conformément aux indications fournies dans l'énoncé du projet.

2.3 Génération des graphes de test :

Afin de tester l'efficacité de la méthode étudiée dans l'accélération de la convergence de l'algorithme de Bellman-Ford, des graphes pondérés G1, G2, G3, et H sont générés avec des fonctions de poids aléatoires, permettant ainsi d'évaluer les performances sur des cas variés sans circuits négatifs.

2.3.1 Fonction d'attribution de poids :

— **"ajout_poids_graphe(G)"** attribue des poids aléatoires à chaque arête de G et supprime éventuellement les cycles négatifs qui pourraient être créés. Elle utilise une boucle infinie pour itérer jusqu'à ce qu'aucun cycle négatif ne soit détecté. À chaque itération, la fonction exécute l'algorithme de Bellman-Ford pour calculer les plus courts chemins depuis chaque nœud, puis vérifie s'il existe un cycle négatif. Si un cycle négatif est détecté, la fonction ajuste le poids de l'arête correspondante pour éliminer le cycle négatif. Ce processus se répète jusqu'à ce qu'aucun cycle négatif ne soit présent dans le graphe pondéré résultant, qui est ensuite renvoyé.

3 Phase de tests

3.1 Test sur un seule graphe G

Dans la phase de test, nous avons suivi les étapes suivantes :

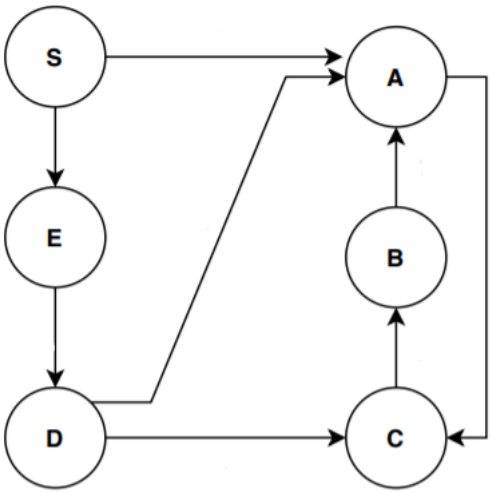


FIGURE 1 – Le graphe G utilisé pour la première phase de test

1. Nous avons génèré un graphe G illustré dans la figure 1, et choisit comme source le sommet s.

2. Après avoir généré les trois graphes pondérés G1, G2, G3 (comme décrit dans la Question 3) et en utilisant la fonction **ajout_poids_graphe(G)**, nous avons appliqué l'algorithme de Bellman-Ford à chacun de ces graphes pour déterminer l'union de leurs arborescences des plus courts chemins, appelée T.
3. Ensuite, dans la Question 5, nous avons utilisé l'algorithme GloutonFas avec T comme entrée, obtenant ainsi un ordre $<_{tot}$.
4. Par la suite, dans la Question 6, pour le graphe H généré à la Question 3 et qui est illustré dans la figure 2, nous avons appliqué l'algorithme de Bellman-Ford en utilisant l'ordre $<_{tot}$.
5. Dans la Question 7, toujours pour le graphe H, nous avons appliqué l'algorithme de Bellman-Ford en utilisant un ordre tiré aléatoirement de manière uniforme.

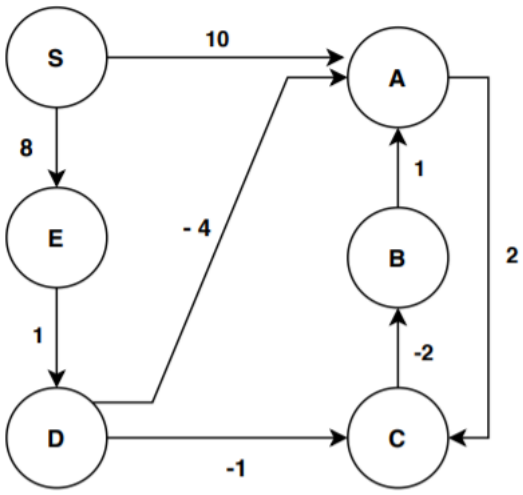


FIGURE 2 – Le graphe H génèré à partir du graphe G

6. En observant les résultats, nous avons noté une amélioration dans le nombre d'itérations de l'algorithme de Bellman-Ford lorsque l'ordre $<_{tot}$ a été utilisé, comparé à l'utilisation d'un ordre tiré aléatoirement.

Conclusion : Suite à la comparaison entre les résultats obtenus dans les Questions 6 et 7, où l'algorithme de Bellman-Ford a été appliqué sur le graphe H en utilisant respectivement l'ordre $<_{tot}$ et un ordre aléatoire, il a été observé que l'utilisation de l'ordre $<_{tot}$ a conduit à une amélioration du nombre d'itérations nécessaires pour la convergence de l'algorithme en passant de 4 itérations pour l'odre aléatoire **[4, 2, 0, 5, 1, 3]** a 1 seule étape avec ce nouvel ordre optimale **[0, 1, 2, 3, 5, 4]** retourné par l'algo **GloutonFas** .

3.2 Analyse comparative du nombre d'itérations renvoyé par l'algorithme de Bellman-Ford sur diverses instances de graphes

Afin de répondre à la question 9, nous avons généré plusieurs instances de graphes de manière aléatoire, en faisant varier le nombre de sommets, n, de 3 à 50. Pour chaque valeur de n, nous avons exécuté l'algorithme de Bellman-Ford à plusieurs reprises, avec et sans la phase de prétraitement. Nous avons enregistré le nombre d'itérations retourné par Bellman-Ford à chaque exécution, permettant ainsi le calcul de la moyenne pour chaque

configuration. Ce processus itératif a abouti à la construction de deux courbes distinctes représentées dans la figure 3 , détaillant la performance de l'algorithme de Bellman-Ford avec et sans la phase de prétraitement, en fonction de la variation du nombre de sommets dans les graphes.

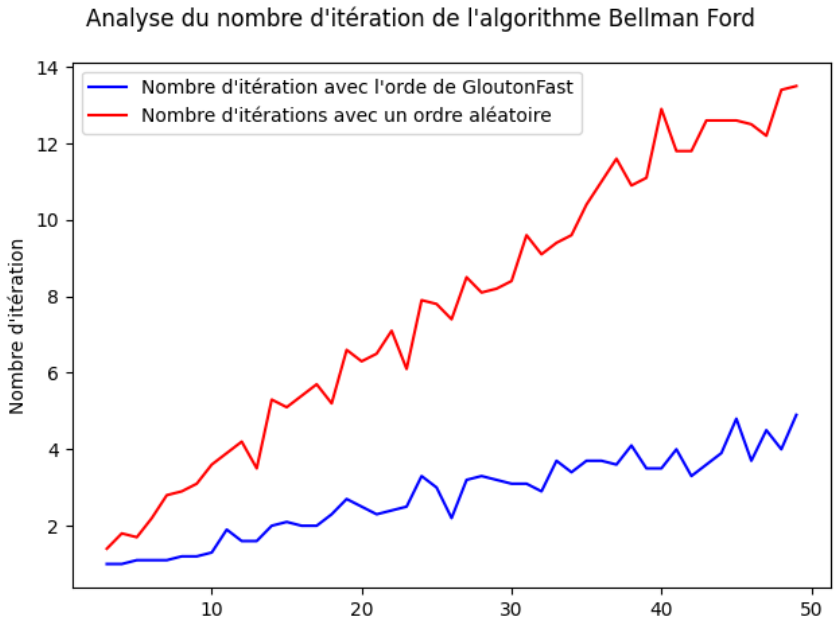


FIGURE 3 – Comparaison du nombre d'itérations retournées par Bellman-Ford avec et sans la phase de prétraitement

Les résultats déduits des deux courbes présentées dans la figure 3 révèlent une tendance intéressante. Lorsque nous mettons en œuvre l'algorithme de Bellman-Ford en suivant l'ordre déterminé par l'algorithme de GloutonFas après la phase de prétraitement, nous observons une diminution notable du nombre d'itérations requises. En d'autres termes, cette approche semble améliorer la complexité de Bellman-Ford.

D'un autre côté, lorsque nous adoptons un ordre aléatoire pour le traitement des sommets, Bellman-Ford semble requérir davantage d'itérations pour parvenir à la solution.

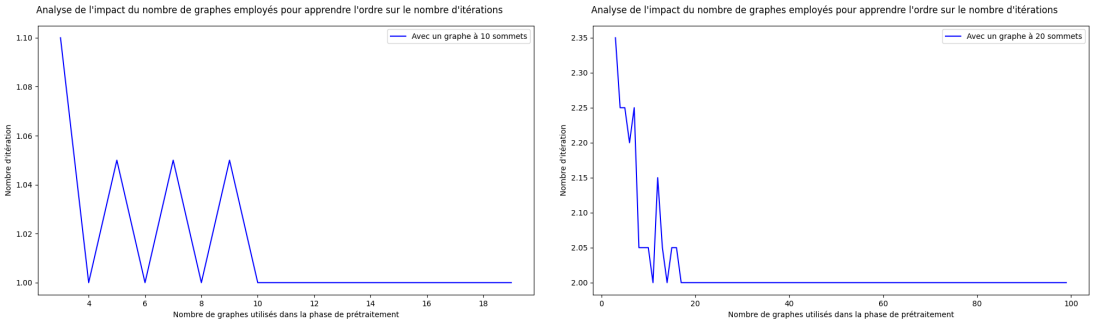
Ces observations suggèrent que l'utilisation de l'ordre déterminé par GloutonFas pourrait jouer un rôle crucial dans l'efficacité de l'algorithme de Bellman-Ford, en réduisant la complexité du calcul et en améliorant sa performance globale. Ces résultats soulignent l'importance de la phase de prétraitement dans le contexte de l'algorithme de Bellman-Ford et offrent des pistes prometteuses pour l'optimisation de ses performances.

3.3 l'impact du nombre de graphes utilisés lors de la phase de prétraitement sur le nombre d'itérations de Bellman-Ford

Afin d'étudier l'impact du nombre de graphes utilisés pour apprendre l'ordre sur le nombre d'itérations, nous avons mené notre analyse en utilisant quatre types de graphes, chacun avec 10 et 20 sommets respectivement. Pour chaque catégorie de graphes, nous avons effectué des itérations sur le nombre de graphes employés durant la phase de prétraitement (de 1 à 30 pour 10 sommets et 1 à 100 pour 20 sommets). Nous avons ensuite

calculé la moyenne du nombre d'itérations retourné par l'algorithme de Bellman-Ford à chaque itération, permettant ainsi une comparaison significative.

L'objectif de cette approche était d'observer comment la variation du nombre de graphes utilisés influençait la performance de l'algorithme.



(a) Graphe avec 10 sommets (b) Graphe avec 20 sommets

FIGURE 4 – Analyse de l'impact du nombre de graphes employés pour apprendre l'ordre sur le nombre d'itérations

Il est notable, en examinant ce graphe composé de 20 sommets de la **figure 4a**, qu'avec un ensemble de 20 graphes de prétraitement, l'algorithme GloutonFas retourne systématiquement le même ordre optimal. En conséquence, l'algorithme de Bellman-Ford converge de manière constante en seulement deux étapes lorsqu'il utilise cet ordre spécifique. Cette régularité souligne la capacité de la méthode GloutonFas à identifier et à appliquer un ordre de traitement stable, conduisant à une convergence rapide de l'algorithme de Bellman-Ford pour ce même ordre optimal.

D'après les diagrammes précédemment présentés, on observe en moyenne que la quantité de graphes utilisés pendant la phase de prétraitement, dans le but d'apprendre l'ordre à appliquer lors de l'exécution de Bellman-Ford, a effectivement un impact sur la complexité de Bellman-Ford. En d'autres termes, nous avons constaté que, à partir d'un nombre suffisant de graphes, l'algorithme parvient toujours à converger vers le même ordre de traitement optimal, ce qui lui permet d'effectuer un nombre constant d'itérations pour atteindre la convergence. Cela explique la régularité observée sur les quatre figures.

3.4 Les graphes par niveaux

La dernière série de tests que nous avons réalisée portait sur des graphes organisés en niveaux, plus spécifiquement sur des graphes comprenant 4 sommets par niveau et 2500 niveaux, où les sommets du niveau j précédaient tous les sommets du niveau $j+1$. Chaque arc était associé à un poids tiré de manière uniforme et aléatoire parmi les entiers dans l'intervalle $[-10, 10]$. Pour générer ces graphes, nous avons mis en place la fonction **generation_graphe_niveaux(nb_sommets_par_niv, nb_niveaux, p)**.

Nous avons généré 20 graphes de cette famille, calculant à chaque fois le nombre d'itérations effectuées par l'algorithme de Bellman-Ford, avec et sans la phase de prétraitement. À chaque itération, nous avons constaté que l'algorithme de Bellman-Ford convergeait en **une seule itération** lorsqu'il utilisait l'ordre optimal pour le traitement des nœuds,

tandis qu'il nécessitait plus de 1200 itérations avec un ordre aléatoire. Ces résultats sont clairement illustrés dans le graphique de la figure 5.

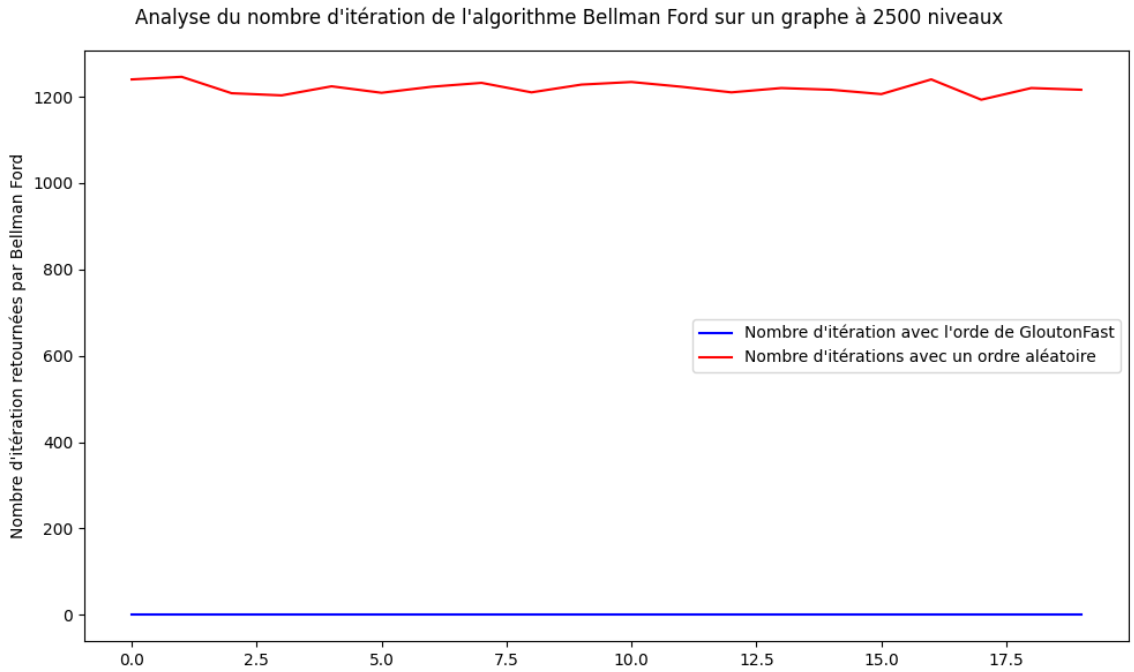


FIGURE 5 – Analyse du nombre d'itérations de l'algorithme Bellman Ford sur des graphes à 2500 niveaux

La méthode avec prétraitement semble adéquate pour la famille d'instances décrite. En effet, dans ces graphes par niveau où chaque niveau précède le niveau suivant et les poids des arcs sont choisis de manière aléatoire dans l'intervalle $[-10, 10]$, la structure particulière des graphes assure l'absence de cycles négatifs. Dans un tel contexte, l'ordre optimal de traitement des nœuds serait celui qui permet de parcourir tous les nœuds en une seule itération, trouvant ainsi les plus courts chemins dès la première mise à jour.

Les résultats obtenus à partir de 20 graphes générés de cette famille, où l'algorithme de Bellman-Ford converge en une seule itération avec l'utilisation de l'ordre optimal, justifient l'efficacité de la méthode avec prétraitement. Dans ce scénario spécifique, l'approche préalable permet de déterminer un ordre de traitement qui prend avantage de la structure des graphes, réduisant ainsi le nombre d'itérations nécessaires à la convergence de l'algorithme de Bellman-Ford.

4 Conclusion

En conclusion, l'analyse des résultats suggère que les graphes appris par l'algorithme GloutonFas permettent de déterminer un ordre optimal qui contribue significativement à l'accélération du processus de convergence de l'algorithme de Bellman-Ford. La formation de cet ordre, basé sur l'union des arborescences des plus courts chemins des différents graphes pondérés , a démontré un impact positif sur l'efficacité globale. Cette approche se révèle particulièrement avantageuse dans des contextes où la réduction du nombre d'itérations est cruciale, soulignant ainsi l'efficacité de l'utilisation de l'ordre $<$tot comme une stratégie bénéfique pour améliorer les performances de l'algorithme de Bellman-Ford dans le contexte spécifique de cette étude.