INFO-F-310
Projet Algorithmique et recherche opérationnelle
22-23

Nicolas Melaerts 496111
Manu Mathey-Prévot 499399

May 22, 2023

Index

1 Machine utilisée

L'ensemble de nos tests et des temps de nos algorithmes récoltés ont été faits sur un Mac avec MacOS Ventura 13.3.1 et un Intel Core I5 quatre coeurs 2GHz et 16 Go de ram.

2 Résolution du problème de flot maximum à l'aide d'un programme linéaire et du solver glpk

L'implémentation qui génère un programme linéaire en nombre entiers d'une instance au format CPLEX LP se trouve dans generate_model.py.

Dans make_objective, on crée la fontion objectif du problème qui vise à maximiser le flot sortant du noeud source. Cette méthode itère sur les noeuds du graphe et ajoute les variables de flot sortant du noeud source à cette fonction objectif.

Dans make_constraints, on construit les contraintes de conservation du flot. Ca itère sur les noeuds du graphe, mais pas sur le noeud source et le noeud puits, et crée les contraintes de conservation du flot pour chaque noeud. On se base sur la somme des flots sortants et des flots entrants.

Dans le code on représente les variables du probème par les flot f_i_j où i et j sont les indices des noeuds du graphe. Dans la méthode make_bounds, on génére les variables f_i_j pour chaque arc du graphe ayant une capacité supérieure à 0. Comme ça nous avons toutes les variables du problème correctement définies et associées aux arcs du graphe.

Dans make_integer, on itére sur tous les arcs du graphe pour voir si la capacité de celle-ci est supérieure à 0. Si c'est le cas, la variable de flot f_i_j est ajoutée pour spécifier qu'elle doit être entière dans la formulation LP.

La méthode générate_lp fait la formulation complète du problème du flot maximal en format LP en utilisant ce que donnent les méthodes make_bounds, make_objective et make_constraints. Avec les variables, les contraintes et la fonction objectif du problème, on peut résoudre le problème et déterminer les valeurs optimales des variables du flot.

3 Description de la méthode des chemins augmentants implémentée

Nous avons choisi d'implémenter l'algorithme de Ford et Fulkerson présenté dans le cours.

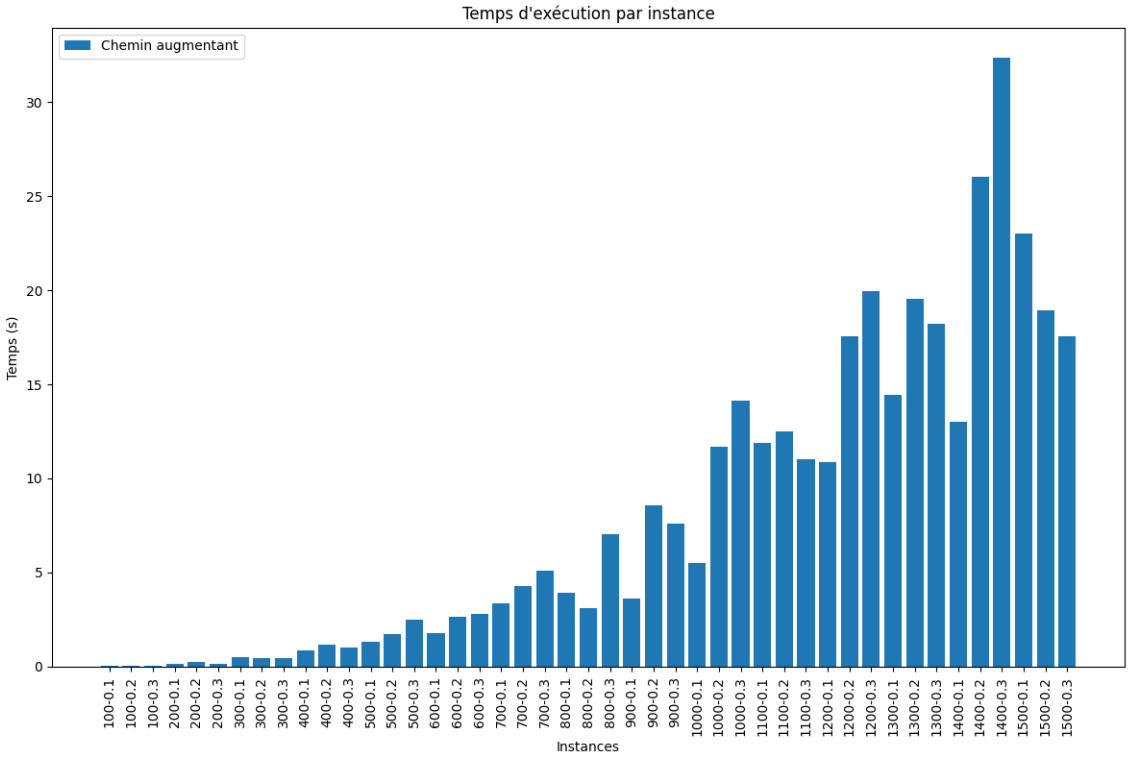
L'algorithme alterne entre une phase de marquage et une phase d'augmentation, qui sont répétées tant qu'il existe encore un chemin augmentant la valeur du flot.

Lors de la phase de marquage, on recherche un chemin augmentant en marquant les noeuds avec la quantité de flot encore possible de faire passer depuis la source jusqu'à ce noeud.

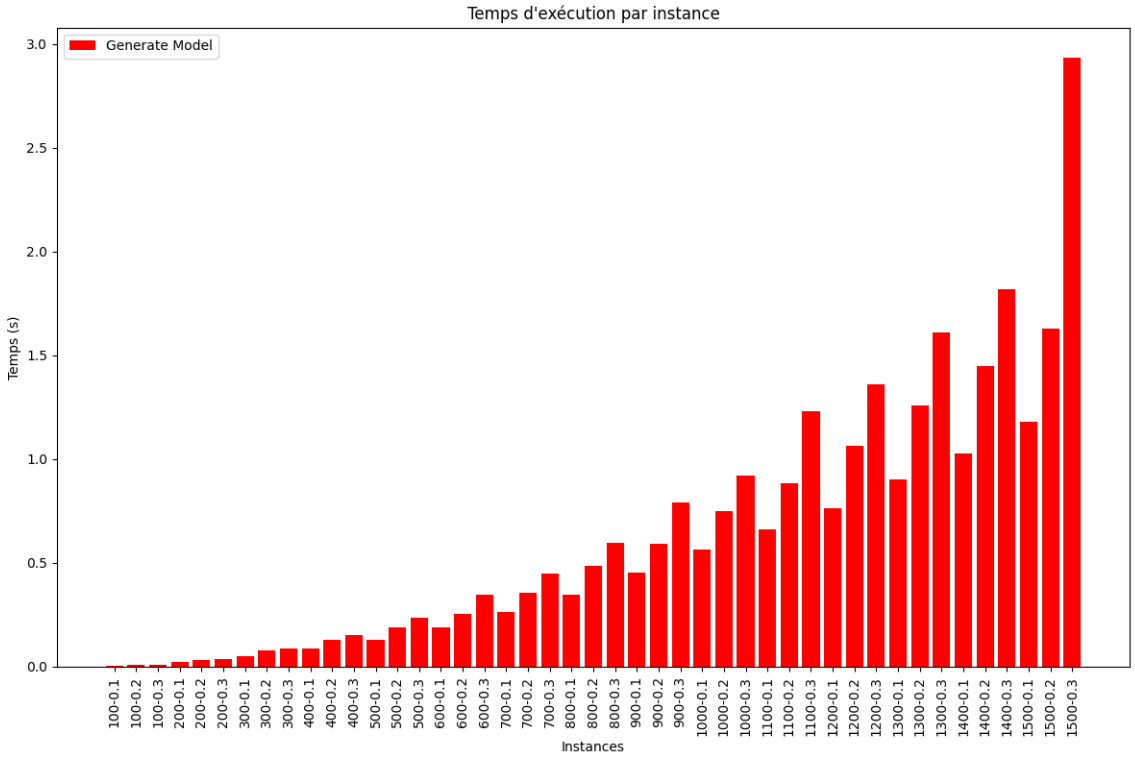
Lors de la phase d'augmentation, on augmente le flot sur le chemin trouvé en phase de marquage.

4 Analyse des résultats lors de la résolution des différentes instances

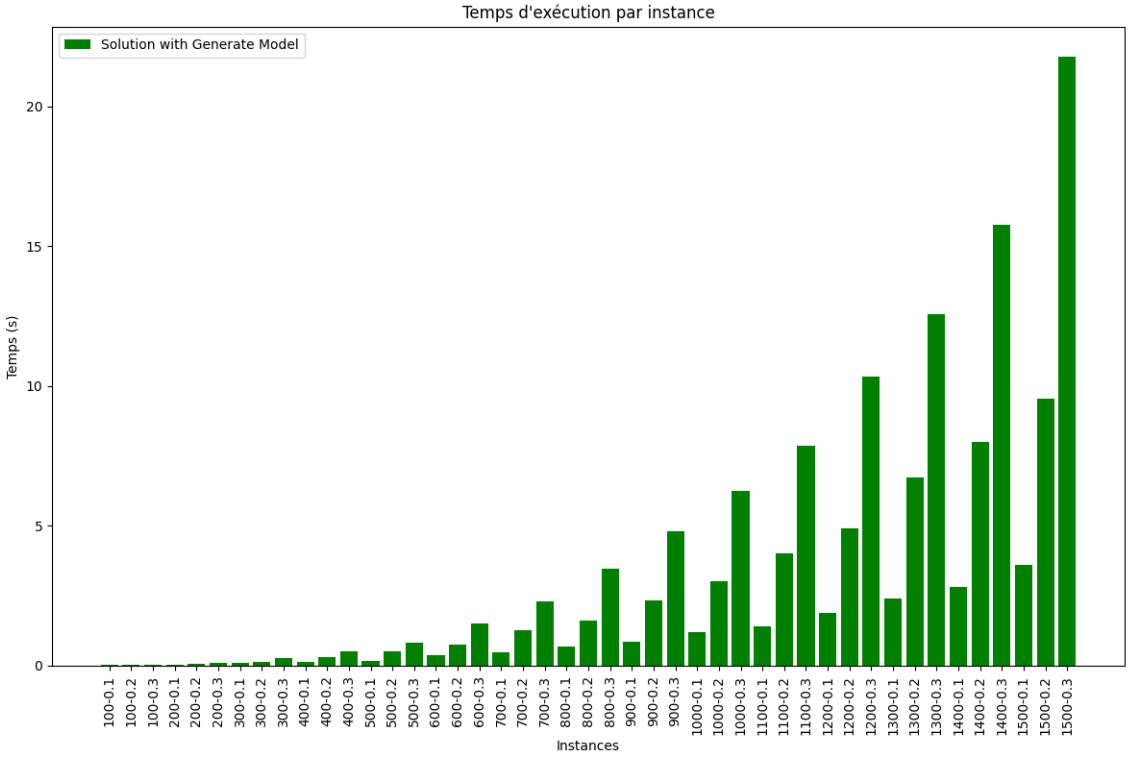
Nous avons, pour chaque instance, calculé le temps d'exécution de notre méthode augmenting_paths et voici un graphique en batonnet. On se rend bien compte ici que plus l'instance est grande et complète, plus ça prend du temps pour trouver le flot maximum.



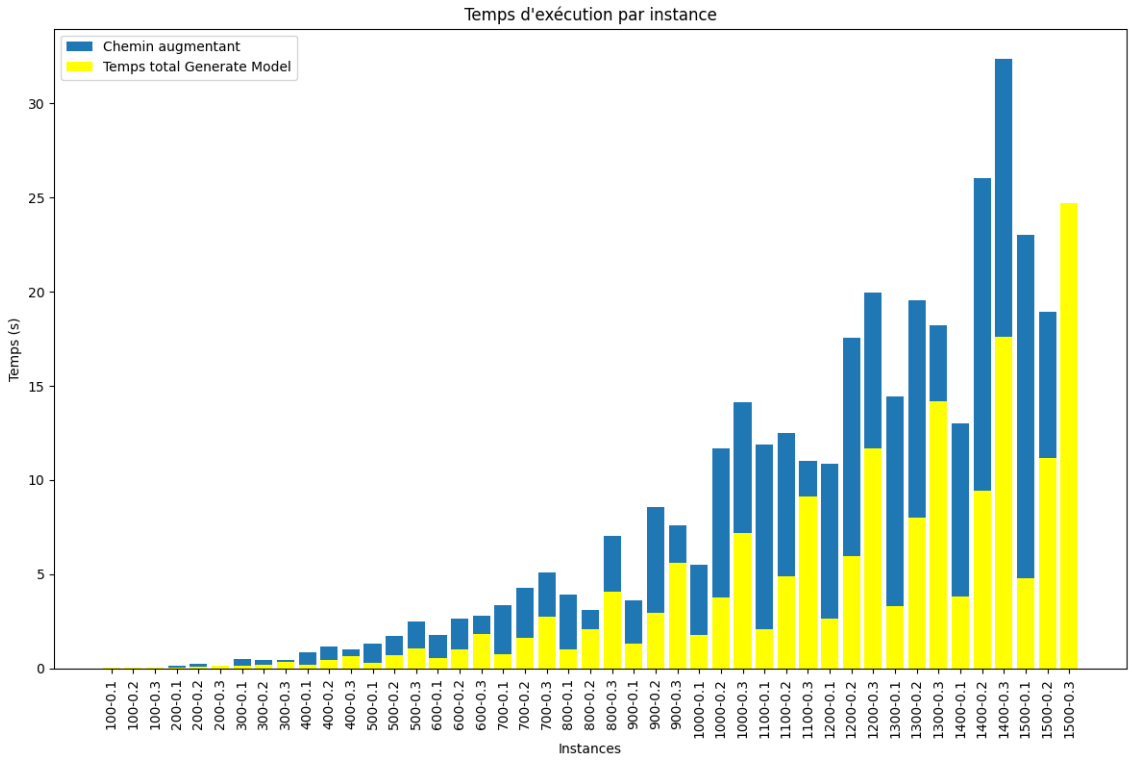
Ensuite nous avons calculé pour chaque instance le temps d'exécution pour générer le model (donc le .lp). On constate que c'est assez rapide, pas plus de 3 secondes pour la dernière instance.



Et pour générer la solution à partir du model .lp. On constate que ça ne dépasse pas les 22 secondes pour la dernière instance.



Dans ce graphique, on a la somme des temps pour générer le model (.lp) et la solution (.sol) en jaune, et en bleu le temps pour les chemins augmentants pour chaque instance. Donc on constate ici clairement qu'en général l'algorithme des chemins augmentants est moins rapide pour trouver le flot maximum que la méthode avec generate model. Sauf par exemple pour la dernière instance, où le temps total de generate model est plus long que le chemin augmentant qui doit sûrement être dû à la façon dont le graphe est formé.



On constate donc que la méthode generate model sera plus appropriée en général, même si dépendemment de l'instance, elle peut parfois être moins efficace comme observé pour la dernière instance.

5 Bonus : Vérification que la solution est optimale

Comme mentionné par mail, il est possible de vérifier si la solution trouvée par l'algorithme des chemins augmentant est optimal en trouvant la coupe minimum qui a la même valeur que le flot trouvé. Pour ce faire, on se base sur le dernier marquage de la méthode des chemins aumentants.

Notre méthode find_minimum_cut va parcourir les noeuds du graphe et vérifier si le noeud i à été marqué lors de la dernière itération dans marking. Si c'est le cas, le noeud i appartient à la coupe minimale. On refait donc un parcours des noeuds du graphe et si le noeud j n'a pas été marqué on peut ajouter la capacité de l'arc. Donc on renvoie la coupe minimale qui est la somme des capacités de chaque noeud marqué lors de la dernière itération et chaque autre noeud non marqué.