MOGPL Automne 2023

Projet

L'algorithme de Bellman-Ford est une méthode de base pour calculer les plus courts chemins à source unique dans les graphes avec des poids d'arête à la fois positifs et négatifs. Sa durée d'exécution dépend de l'ordre dans lequel l'algorithme sélectionne les sommets pour les mises à jour itératives de la valeur de leur plus court chemin.

Notre objectif principal est d'améliorer le temps d'exécution de l'algorithme de Bellman-Ford dans le cas où les exemples ne sont pas les pires. Pour ce faire, nous introduisons une étape de prétraitement à l'algorithme de Bellman-Ford qui, étant donné une collection d'instances typiques, sélectionne un bon ordre pour examiner les sommets. Ce faisant, nous autorisons un coût de prétraitement plus important (mais toujours polynomial) afin de réduire le temps d'exécution par instance.

Le problème : Soit $G = (V, E)$ un graphe orienté pondéré avec $|V| = n$ sommets et $|E| = m$ arcs. Pour un arc $e \in E$, nous notons w_e le poids de e. Certains poids peuvent être négatifs, mais le graphe ne contient aucun circuit de poids négatif. Étant donné un sommet particulier $s \in V$, l'algorithme de Bellman-Ford calcule le plus court chemin de s à chaque sommet $v \in V$. L'algorithme maintient une borne supérieure, d_v, sur la plus courte distance de s à v. Il commence par fixer $d_v = \infty$ (sauf $d_s = 0$) et procède en itérant sur tous les sommets et en considérant tous les arcs entrants, c'est-à-dire, pour un sommet v, en fixant

$$d_v = \min_{y:(y,v)\in E} d_y + w(y, v)$$

Il est connu que si G n'a aucun circuit de poids négatif, l'algorithme Bellman-Ford converge après au plus $(n-1)$ itérations à travers l'ordre complet des sommets.

Bien que le nombre d'itérations soit $(n-1)$ dans le pire des cas, nous pouvons être plus précis sur le nombre d'itérations en considérant un ordre spécifique des sommets. Etant donné un graphe G et deux sommets s et v, soit $P_v = (s, x_1, x_2, \ldots, v)$ le plus court chemin de s à v. Soit $<_{tot}$ un ordre sur V où, pour deux sommets u, v, nous avons $u <_{tot} v$ si u apparaît avant v dans cet ordre. Pour un chemin P, notons $dist(<_{tot}, P)$ le nombre d'arcs dans P inversés par $<_{tot}$, c'est-à-dire, $dist(<_{tot}, P) = |\{(u, v) \in P \mid v <_{tot} u\}|$.

Le résultat suivant motive l'approche proposée :

Proposition. Etant donné un graphe G, le temps d'exécution de l'algorithme de Bellman-Ford en utilisant l'ordre $<_{tot}$ pour traiter les sommets est en

$$O(m \max_{v\in V} dist(<_{tot}, P_v))$$

Ce résultat met formellement en évidence l'avantage d'avoir des sommets dans un bon ordre.

Supposons que le graphe G soit fixe, mais que les poids sur les arêtes soient tirés d'une distribution inconnue. Quel est le meilleur ordre à utiliser ?

Par la suite, on fixe le graphe $G = (V, E)$ et on considère un ensemble des fonctions des poids des arcs $w^1, w^2, \ldots, w^k$. Pour chaque $i \in [k]$, nous définissons P_v^i comme un plus court chemin de la source s à $v \in V$ dans le cas où les poids sur les arcs de G sont w^i. On appellera $G_i = (V, E, w^i)$, pour $i \in [k]$. Enfin, le graphe H représentera le graphe test où nous évaluerons la performance de cette approche. La question qui nous intéresse est de savoir si plus G_i et H sont similaires, plus l'amélioration en termes de diminution du nombre d'itérations de l'algorithme Bellman-Ford est importante.

Pour décider de l'ordre que nous utiliserons comme entrée de l'algorithme Bellman-Ford, nous devons résoudre le problème MVP (Minimum Violation Permutation) qui est défini comme suit :

Définition de MVP : Étant donné un ensemble de sommets V et un ensemble de r chemins $P_1, P_2, \ldots, P_r$, où chaque $P_i \subseteq V$, trouver un ordre total $<_{tot}$ sur V qui minimise l'objectif

$$\max_{i \in [r]} dist(<_{tot}, P_i).$$

Relation de MVP avec le problème du plus court chemin. Considérons une solution $<_{tot}$ de MVP pour l'ensemble des chemins

$$\cup_{i \in [k]} \cup_{v \in V} P_v^i.$$

Cette solution réduit le temps d'exécution dans le pire cas de l'algorithme de Bellman-Ford dans tous les graphes G. Le nombre maximum d'itérations sur n'importe quelle instance parmi ces k instances est au plus $\max_{i \in [k], v \in V} dist(<_{tot}, P_v^i)$ qui est égale à la valeur de la solution $<_{tot}$ de l'instance de MVP.

Résolution de MVP. Pour résoudre MVP, nous allons considèrer une méthode gloutonne que l'on appelera *GloutonFas*. Elle prend en entrée un graphe correspondant à l'union des plus courts chemins et sélectionne de manière gloutonne les sommets pour construire un ordre linéaire des sommets. L'intuition qui sous-tend la méthode est de déplacer tous les sommets ayant un faible degré entrant[1] au début de la liste et ceux ayant un faible degré sortant[2] à la fin. L'algorithme maintient deux listes, s_1 et s_2, qui sont initialisées comme étant vides. À chaque itération, toutes les sources (sommets avec un degré entrant 0) sont d'abord supprimées l'une après l'autre et ajoutées à la fin de la liste s_1, et tous les puits (sommets avec un degré sortant 0) sont supprimés et ajoutés à la tête de la liste s_2. Enfin quand il n'y a plus de source ou de puit, un sommet u, avec la différence maximale entre son degré sortant et son degré entrant, est supprimé et ajouté à la fin de la liste s_1. Cette boucle se répète jusqu'à ce que tous les sommets du graphe se trouvent soit dans s_1, soit dans s_2, et les deux listes sont alors concaténées. Une description plus précise est donnée par le pseudocode suivant :

Exemple. A titre d'exemple, considérons l'exécution de GloutonFas sur le graphe de la figure

[1]Le degré entrant d'un sommet v, noté $d^-(v)$, est le nombre d'arcs ayant comme extrémité finale v.

[2]Le degré sortant d'un sommet v, noté $d^+(v)$, est le nombre d'arcs ayant comme extrémité initiale v.

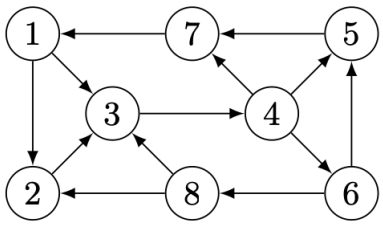
Algorithm 1 Algorithme GloutonFas(G)

Entrée : un graphe orienté G

Sortie : une permutation des sommets de G

$s_1 \leftarrow \emptyset$, $s_2 \leftarrow \emptyset$
while G n'est pas vide **do**
  while G contient une source u **do**
    $s_1 \leftarrow s_1 u$
    retirer u de G
  end while
  while G contient un puits u **do**
    $s_2 \leftarrow u s_2$
    retirer u de G
  end while
  Choisir le sommet u qui maximise $\delta(u) = d^+(u) - d^-(u)$
  $s_1 \leftarrow s_1 u$
  Retirer u de G
end while
Retourner $s = s_1 s_2$

suivante.



Initialement, il n'y a pas de puits ou de source, nous supprimons donc le sommet u pour lequel $\delta(u) = d^+(u) - d^-(u)$ est un maximum : le sommet 4. Le sommet et ses arcs sont retirés de G et le sommet 4 est ajouté à s_1. Après avoir retiré le sommet 4 de G, une source est créée le sommet 6 qui est retiré de G et ajouté à la fin de s_1. Le processus se poursuit en supprimant les sommets 5,8,7,1,2,3 qui sont tous ajoutés à s_1. L'ordre résultant est $s = s_1 s_2 = [4, 6, 5, 8, 7, 1, 2, 3]$.

Question 1 Coder l'algorithme Bellman-Ford comme ennoncé plus haut. L'algorithme renverra l'arborescence des plus courts chemins ainsi que le nombre d'itérations qui ont été nécessaires avant que l'algorithme converge.

Question 2 Coder l'algorithme GloutonFas qui prend comme entrée un graphe et renverra un arrangement linéaire de ses sommets.

Question 3 Afin de tester l'efficacité de la méthode, générer un graphe orienté $G = (V, E)$ et choisir un sommet source s parmi les sommets de G. Pour éviter des cas triviaux choisir comme

source un sommet à partir duquel on peut atteindre au moins $|V|/2$ sommets. A partir de G construire 3 graphes ponderés G_1, G_2, G_3 ainsi que le graphe test H. Ce qui différencie ces 4 graphes ce sont les fonctions de poids w^1, w^2, w^3, w^4. Pour chaque fonction de poids w^i on associe à chaque arc un poids tiré de manière uniforme et aléatoire parmi les entiers d'un intervalle, p.ex. $[-10, 10]$. On supposera que les graphes G_1, G_2, G_3 et H ne contiennent pas de circuit négatif.

Question 4 Etant donnés les 3 graphes G_1, G_2, G_3 (générés dans la question 3), appliquer l'algorithme Bellman-Ford dans chacun de ces graphes et déterminer l'union de leurs arborescences des plus courts chemins que l'on appelera T.

Question 5 Appliquer l'algorithme GloutonFas avec comme entrée T et retourner un ordre $<_{tot}$.

Question 6 Pour le graphe H généré à la question 3 appliquer l'algorithm Bellman-Ford en utilisant l'ordre $<_{tot}$.

Question 7 Pour le graphe H appliquer l'algorithm Bellman-Ford en utilisant un ordre tiré aléatoirement (de manière uniforme).

Question 8 Comparer les résultats (nombre d'itérations) obtenus dans les questions 6 et 7.

Question 9 Générer des instances aléatoires pour tester la méthode avec prétraitement par rapport à l'application de l'algorithme de Bellman-Ford basée simplement sur un ordre tiré de manière aléatoire.

Question 10 Comment le nombre de graphes que l'on utilise pour apprendre l'ordre influence le nombre d'itérations ?

Question 11 Soit la famille d'instances suivantes : un graphe par niveau avec 4 sommets par niveau et 2500 niveaux où les sommets du niveau j précédent tous les sommets du niveau $j+1$ et où pour chaque arc on a un poids tiré de manière uniforme et aléatoire parmi les entiers dans l'intervalle $[-10, 10]$. Est-ce que la méthode avec prétraitement est adéquate ? Justifier votre réponse.

Fonctionnement, échéances et délivrables

Le projet est à réaliser en binôme, le binôme devant être composé de deux personnes inscrites dans le même groupe en MOGPL. Les binômes devront remettre un document pdf où seront rédigées les réponses aux questions du projet. On mettra en annexe de ce document pdf une description des modèles et tests réalisés ainsi qu'une archive zip concernant les programmes réalisés. Ces documents seront remis au chargé de TD de votre groupe (un par binôme) via une soumission Moodle au plus tard le 8 décembre 2023. Les soutenances de projet auront lieu lors des créneaux habituels de TD la semaine du 11 décembre 2023.