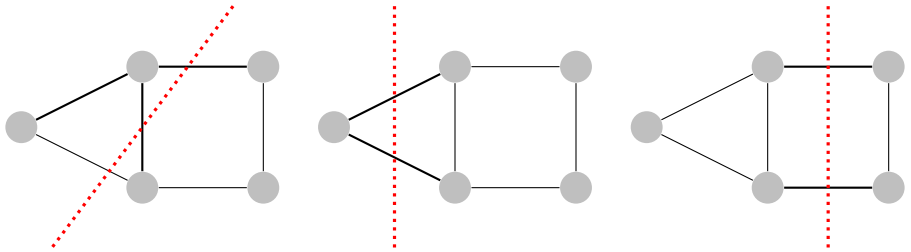
COMPLEX - TME Projet

novembre-décembre 2022

Énoncé

En théorie des graphes et en informatique théorique, une coupe minimum d'un graphe est une coupe contenant un nombre minimal d'arêtes. Cette notion apparaît notamment dans le théorème flot-max/coupe-min et peut être utilisée dans différents contextes, notamment en vision artificielle.



(a) (b) (c)

FIGURE 1 – Exemples de coupe dans un graphe non orienté. La coupe (a) est de cardinal 3 alors que les coupes (b) et (c) sont de cardinal 2 et sont minimales.

Nous avons vu en cours l'algorithme de Karger qui est un algorithme probabiliste simple de type Monte Carlo pour trouver une coupe minimum dans un graphe. Nous avons également abordé rapidement une variante (également probabiliste) fondée sur l'approche « diviser pour régner » et appelée algorithme de Karger-Stein.

Le but de ce projet est d'implanter ces algorithmes, de choisir et d'analyser des structures de données adaptées et d'effectuer des tests d'efficacités sur différentes familles de graphe.

Organisation du travail

- Le choix du langage de programmation pour implanter les différents algorithmes est libre.
- Le travail est à effectuer en binôme.
- Les projets doivent être rendus au plus tard le
  - mercredi 7 décembre 2022 pour le groupe du lundi ;
  - vendredi 9 décembre 2020 pour le groupe du mercredi et du jeudi ;

  par mail à votre chargé de TD ; le sujet de l'email doit être de la forme,

  `[CPLX] Projet, NOMbinôme1-NOMbinôme2`

  Votre livraison sera constituée d'une archive tar.gz qui doit comporter un rapport décrivant vos choix d'implémentation et votre code ainsi que la description des tests de validation et les réponses aux questions.
- Une soutenance est prévue lors des TME lors de la semaine du 12 décembre 2022.

Travail à réaliser

Exercice 1 : Algorithme de Karger

Rappelons tout d'abord l'algorithme de Karger vu en cours :

Algorithme 1 : Algorithme de Karger

Entrée : $G = (V, E)$ un graphe non orienté avec $n = \#V$ et $m = \#E$
Sortie : S une coupe de G
tant que $\#V > 2$ **faire**
  $e \xleftarrow{\text{⚂⚄}} E$
  $G \leftarrow G/e$
$\{v_1, v_2\} \leftarrow V$ ▷ Il reste deux sommets dans V
retourner $S = \{$ sommets qui « apparaissent » dans v_1 $\}$

Au cours de l'exécution de cet algorithme, le graphe G donné en entrée peut être transformé en un multi-graphe et il est donc nécessaire de pouvoir exécuter cet algorithme sur un multi-graphe dans votre implantation.

1.a] Dans le cours, nous avons vu que l'opération de contraction d'arête « $G \leftarrow G/e$ » de la boucle **tant que** prend un temps $O(n)$ en utilisant une représentation du multi-graphe G par matrice d'adjacence.
Programmer une structure de donnée permettant de représenter un multi-graphe par matrice d'adjacence et une fonction réalisant l'opération de contraction (en prenant en entrée un (multi)-graphe et une arête).

1.b] Tester votre fonction sur différentes familles de graphe de votre choix (par exemple, mais sans s'y limiter, des cycles contenant n sommets, des graphes complets à n sommets, des graphes bipartis complets à $n = 2k$ sommets, des graphes aléatoires à n sommets où une arête est ajoutée à E avec une probabilité $0 < p < 1$, ...).
Présenter et analyser la complexité expérimentale de votre implantation sur ces familles de graphe.

1.c] Une fois que la première opération de contraction a eu lieu dans l'algorithme de Karger, le graphe G est possiblement devenu un multi-graphe avec plusieurs arêtes entre deux sommets. Dans notre analyse théorique de la complexité de l'algorithme de Karger, nous avons supposé que l'arête e de l'opération de sélection aléatoire « $e \xleftarrow{\text{⚂⚄}} E$ » dans la boucle **tant que** est tirée uniformement aléatoirement parmi toutes les arêtes possibles. Expliquer en détail comment effectuer ce tirage aléatoire lorsque le multi-graphe G est représenté par une matrice d'adjacence Implanter cette fonction.

1.d] Utiliser les fonctions des questions précédentes pour terminer une première implantation de l'algorithme de Karger utilisant une représentation par matrice d'adjacence. Présenter et analyser la complexité expérimentale obtenue pour les familles de graphe de la question **1.b]**. Comparer avec l'analyse théorique vue en cours.

L'objectif suivant est de proposer (au moins) une deuxième implantation de l'algorithme de Karger en utilisant une autre représentation des multi-graphes. Il est possible par exemple d'utiliser une liste d'adjacence ou une structure de type *union-find* comme utilisée dans l'algorithme de Kruskal.

1.e] Programmer la structure de donnée choisie pour représenter les multi-graphes et une fonction réalisant l'opération de contraction.

1.f] Expliquer comment effectuer le tirage aléatoire d'une arête avec votre structure de données et implanter une fonction probabiliste effectuant cette sélection aléatoire.

1.g] Analyser la complexité théorique de l'algorithme obtenu en utilisant cette structure de données.

1.h] Présenter et analyser la complexité expérimentale obtenue pour les graphes de la question **1.b]**. Comparer les résultats obtenus avec ceux de l'implantation utilisant les matrices d'adjacence.

1.i] **[optionnel]** Il est possible d'adapter l'algorithme de Karger à un graphe pondéré (c.-à-d. où chaque arête du graphe possède un poids dans $\mathbb{R}_+$). Expliquer comment l'algorithme de Karger s'adapte pour trouver une coupe de poids globale minimal dans un graphe pondéré non orienté et modifier vos implantations pour traiter ce nouveau problème.

Exercice 2 : Amplification du succès de l'algorithme

Comme nous l'avons vu en cours, cet algorithme probabiliste est de type Monte-Carlo et peut retourner une coupe du graphe qui n'est pas minimale.

2.a] Effectuer une étude expérimentale de la probabilité de succès de vos implantations de l'exercice précédent sur les familles de graphes de la question **1.b]** pour lesquelles vous connaissez la taille d'une coupe minimale. Comparer avec l'analyse théorique vue en cours.

Nous avons vu que pour augmenter la probabilité de succès, il est utile de répéter plusieurs fois l'algorithme de Karger. Nous obtenons ainsi l'algorithme de Karger itéré qui prend en entrée un entier T et exécute l'algorithme précédent $T \geq 1$ fois

Algorithme 2 : Algorithme de Karger itéré

Entrée : $G = (V, E)$ un graphe non orienté avec $n = \#V$ et $m = \#E$, T
Sortie : S une coupe de G
$m^* \leftarrow +\infty$
pour i *de* 1 *à* T **faire**
  tant que $\#V > 2$ **faire**
    $e \xleftarrow{\text{🎲}} E$
    $G \leftarrow G/e$
  $\{v_1, v_2\} \leftarrow V$ ▷ Il reste deux sommets dans V
  $S =$ { sommets qui « apparaissent » dans v_1 }
  $m \leftarrow$ cardinal de la coupe $(S, V \setminus S)$
  si $m < m^*$ **alors**
    $S^* \leftarrow S$
    $m^* \leftarrow m$
retourner S^*

2.b] Adapter vos implantations de l'exercice précédent pour obtenir des implantations de l'algorithme de Karger itéré.

2.c] Effectuer une étude expérimentale de ces implantations sur les familles de graphes de la question **1.b]** pour lesquelles vous connaissez la taille d'une coupe minimale. (en fonction de T et du nombre de sommets du graphe). Comparer avec l'analyse théorique vue en cours.

2.d] **[optionnel]** Si vous avez traité la question **1.i]**, vous pouvez analyser et discuter les performances de l'algorithme de Karger modifié pour les graphes pondérés non orientés.

Exercice 3 : Algorithme de Karger-Stein

Nous avons vu en cours dans l'analyse de la probabilité de succès de l'algorithme de Karger itéré que la probabilité d'erreur augmente lorsque le nombre de sommets diminue et devient proche de $1/2$ lorsque le nombre de sommets restants est de l'ordre de $n/\sqrt{2}$ (*cf.* Figure 2).

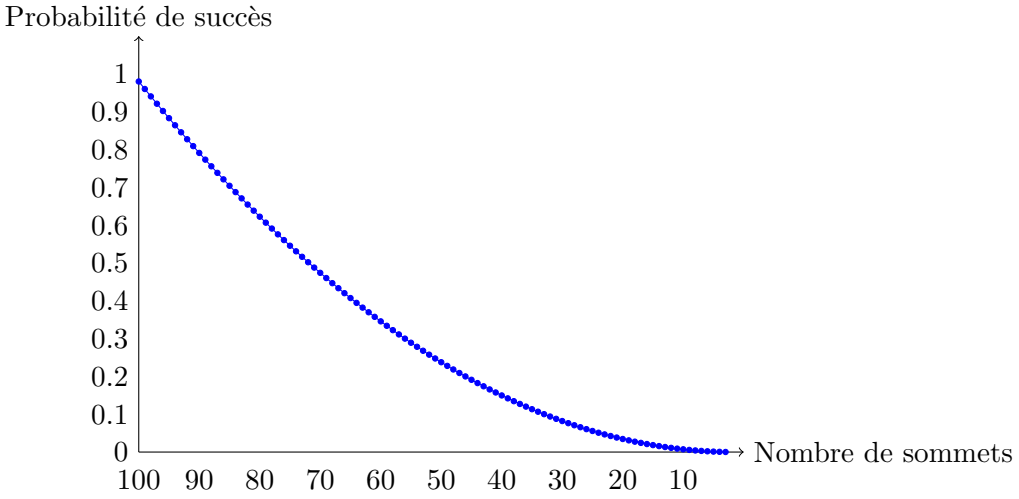


FIGURE 2 – Probabilité de succès de l'algorithme de Karger en fonction du nombre de sommets restants

D. Karger et Stein ont suggéré une approche « diviser pour régner » pour améliorer l'algorithme de Karger où l'algorithme effectue deux contractions partielles indépendantes du graphe jusqu'à ce qu'il reste environ $n/\sqrt{2}$ sommets et cherche récursivement une coupe minimale dans les deux multi-graphes obtenus (et retourne la coupe obtenue de plus .petit cardinal).

Algorithme 3 : Algorithme de contraction partielle ContractionPartielle(G, t)

Entrée : $G = (V, E)$ un graphe non orienté avec $n = \#V$ et $m = \#E$; $t \leq n \in \mathbb{N}$
Sortie : G' un multi-graphe à t sommets
tant que $\#V > t$ **faire**
  $e \xleftarrow{\text{🎲}} E$
  $G \leftarrow G/e$
  ▷ Il reste t sommets dans le graphe
retourner G

Algorithme 4 : Algorithme de Karger-Stein KargerStein(G)

Entrée : $G = (V, E)$ un graphe non orienté avec $n = \#V$ et $m = \#E$, T
Sortie : S une coupe de G
$m^* \leftarrow +\infty$
si $\#V \leq 6$ **alors**
  retourner une coupe minimale de G ▷ par recherche exhaustive
sinon
  $t \leftarrow \lceil 1 + \#V/\sqrt{2} \rceil$
  $G_1 \leftarrow$ ContractionPartielle(G, t)
  $S_1 \leftarrow$ KargerStein(G_1) ; $m_1 \leftarrow$ cardinal de la coupe $(S_1, V \setminus S_1)$
  $G_2 \leftarrow$ ContractionPartielle(G, t)
  $S_2 \leftarrow$ KargerStein(G_2) ; $m_2 \leftarrow$ cardinal de la coupe $(S_2, V \setminus S_2)$
  si $m_1 < m_2$ **alors**
    retourner S_1
  sinon
    retourner S_2

3.a] Montrer que le temps d'exécution $T(n)$ de l'algorithme de Karger-Stein sur un graphe à n sommets vérifie la relation

$$T(n) = 2T\left(\left\lceil 1 + \frac{n}{\sqrt{2}} \right\rceil\right) + O(n^2)$$

et en déduire que $T(n) = O(n^2 \log n)T(n)$.

3.b] Montrer que la probabilité de succès $P(n)$ de l'algorithme de Karger-Stein sur un graphe à n sommets vérifie la relation

$$P(n) \geq 1 - \left(1 - \frac{1}{2}P\left(\frac{n}{\sqrt{2}} + 1\right)\right)^2.$$

3.c] **[optionnel et difficile]** Montrer que $P(n) = \Omega(1/\log n)$.

3.d] Adapter les implantations des exercices **1** et **2** pour obtenir des implantations de l'algorithme de Karger-Stein.

3.e] Effectuer une étude expérimentale de ces implantations sur les familles de graphes de la question **1.b]** pour lesquelles vous connaissez la taille d'une coupe minimale.
Comparer expérimentalement avec l'algorithme de Karger itéré et avec la borne théorique de la question **3.c]**.