Projet COMPLEX

Algorithmes de Karger et Karger-Stein

Amélie Sun
Jules Bouton

Sorbonne Université
Master ANDROIDE
MU4IN900 Groupe 3
Année universitaire 2022-2023

Table des matières

1 Algorithme de Karger

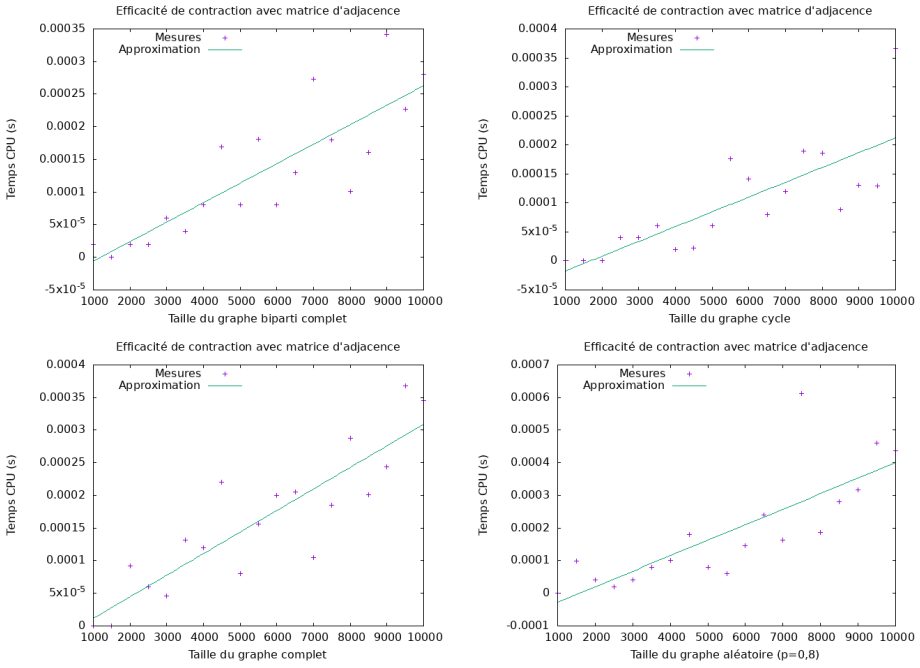
1.a Structure de donnée pour le multi-graphe

Nous avons choisi de représenter un multi-graphe avec une classe qui a comme variables une matrice d'adjacence (variable d'entrée) et une liste d'ensembles s représentant les sommets.
Pour la fonction réalisant l'opération de contraction, elle prend en entrée les indices des deux sommets u et v à contracter. On ajoute d'abord la ligne de v à celle de u. Puis par symétrie on définit la colonne u. Et enfin on met à zéro la ligne et la colonne v sans oublier de mettre la case d'indice (u, u) à zéro également. On ajoute le multi-sommet de l'indice v dans s au multi-sommet $s[u]$ et on remplace $s[v]$ par l'ensemble vide. Ainsi avec l'arête (u, v) contractée on un multi-sommet de moins.

1.b Complexité expérimentale de contraction

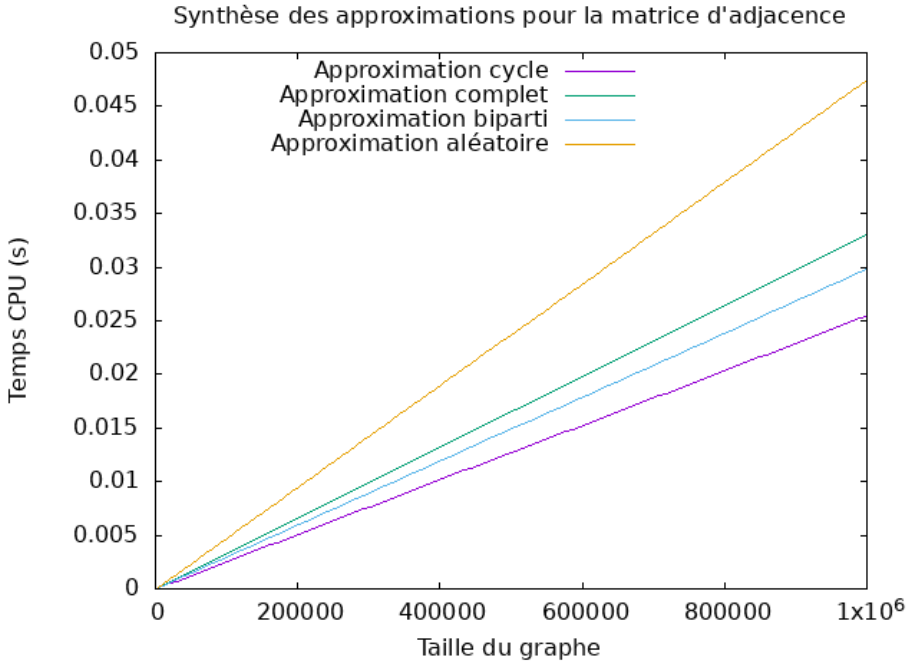
Afin de mesurer la complexité expérimentale présentée dans la figure 1, nous avons fait le choix de prendre la moyenne de 50 opérations de contraction.

Figure 1: Mesures des temps de calcul de la fonction contraction implémentée avec matrice d'adjacence



Elles sont toutes réalisées sur une instance de graphe qui n'a pas encore été contractée et pour chacune des 50 contractions on tire l'arête aléatoirement. On a résumé les différentes approximations linéaires dans un seul graphe présenté dans la figure 2 afin de faire une comparaison sur les différentes familles. Nous pouvons remarquer que sur toutes les familles, nous sommes dans le même ordre de grandeur quelle que soit la taille du graphe. En effet, on effectue les mêmes opérations sur une matrice d'adjacence de taille n^2 ne dépendant donc pas du nombre d'arêtes.

Figure 2: Comparaison des approximations des temps de calcul de contraction implémenté avec matrice d'adjacence



1.c Tirage uniforme d'arête dans une matrice d'adjacence

Pour le tirage aléatoire d'une arête parmi toutes les arêtes possibles, nous allons procéder au tirage d'un numéro entre 0 et le nombre d'arêtes du graphe. On obtient celui-ci en sommant la matrice et en la divisant par 2. Ensuite, nous parcourons la matrice au dessus de sa diagonale et nous comptons le nombre d'arêtes rencontrées. Pour finir, nous sélectionnons l'arête correspondant au numéro tiré dans le parcours comme arête à contracter.

1.d Implantation de Karger et analyse expérimentale

Pour l'algorithme de Karger, nous répétons l'opération de tirage-contraction jusqu'à ce que le nombre de sommets soit égal à 2 dans le multi-graphe. Pour connaître le cardinal de la coupe retournée, il suffit de compter les arêtes du multi-graphe en fin d'algorithme. Pour mesurer le temps de calcul, nous avons fait la moyenne sur 5 exécutions de l'algorithme. Cette procédure nous donne ainsi la figure 3. Nous avons réalisé une approximation quadratique de nos mesures qui semble bien modéliser le temps de calcul de notre algorithme. De plus, la complexité théorique de l'algorithme de Karger vu en cours est en $O(n^2)$ et on se rapproche avec notre implantation. Nous avons pu ainsi réaliser la comparaison de son efficacité sur les différentes familles de graphes, voir figure 4. Nous pouvons faire les mêmes observations que pour l'efficacité de contraction, la forme du graphe n'impacte pas l'ordre de grandeur du temps de calcul.

Figure 3: Mesures des temps de calcul de Karger implémenté avec matrice d'adjacence

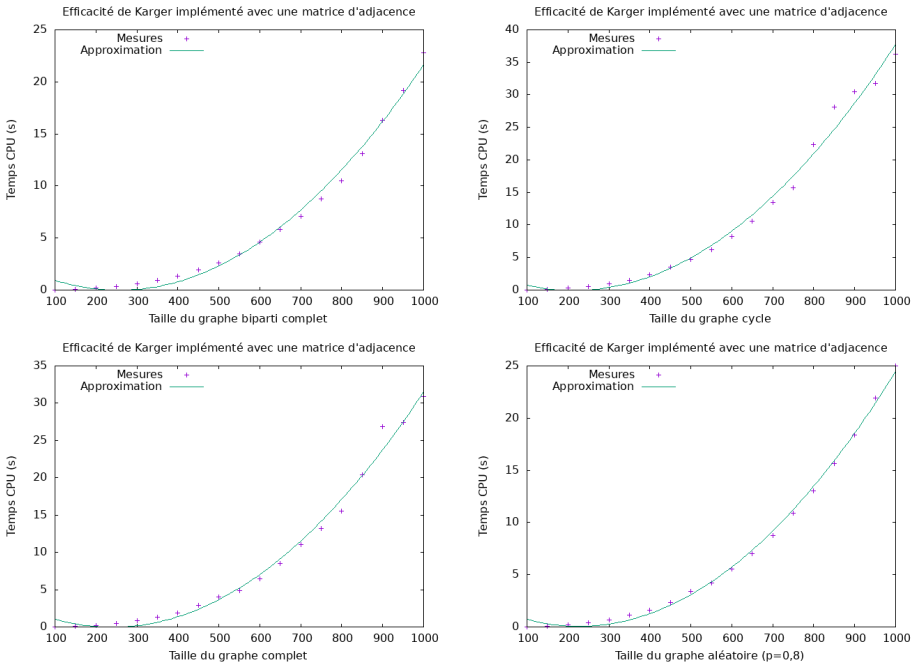
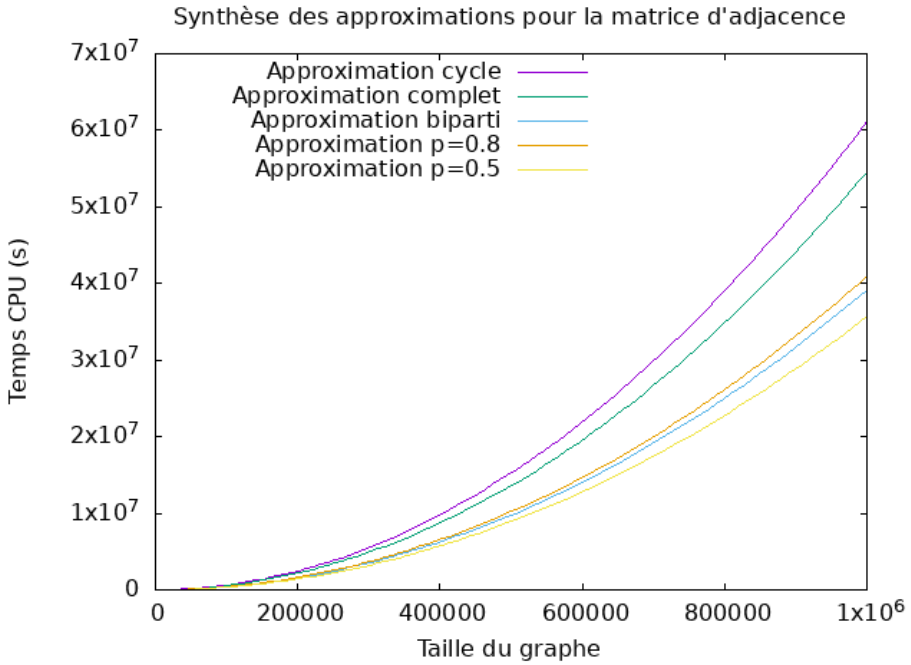


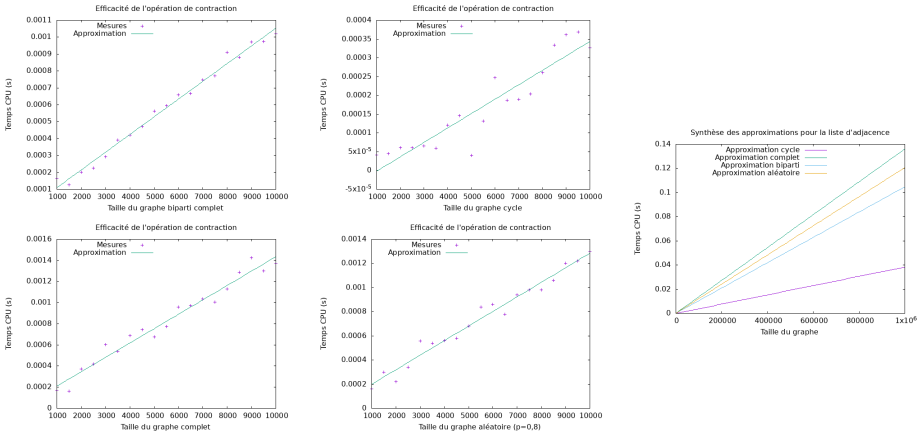
Figure 4: Comparaison des approximations des temps de calcul de Karger implémenté avec matrice d'adjacence



1.e Multi-graphe par liste d'adjacence

Pour la deuxième implantation, nous avons choisi de stocker le graphe dans une liste d'adjacence. Nous avons donc une nouvelle classe, MultigraphList, qui a comme attribut un dictionnaire de liste dont les clés sont les sommets et à chaque clé on associe la liste de ses voisins. La contraction se fera en concaténant à la liste des voisins de u, la liste des voisins de v privée de u (u pouvant être un ensemble de sommets). Une fois l'opération réalisée on supprime l'entrée v et on met à jour le nom de u dans s en lui ajoutant les sommets de v, le deuxième attribut de la structure représentant la liste des sommets. Enfin, nous mettons les sommets de v à vide dans s. Nous avons réalisé la même analyse expérimentale pour vérifier la linéarité de la complexité de l'opération, dont on présente les résultats sur la figure 5. On peut remarquer que sur cette deuxième implantation, l'implantation du graphe impacte sur la complexité puisque les listes d'adjacence de taille $n + m$, des graphes peu denses seront traités plus rapidement qu'un graphe dense. Cela s'observe avec les mesures sur un graphe cyclique, qui donnent des résultats significativement en dessous des autres.

Figure 5: Mesures et comparaison des temps de calcul de contraction implémenté avec une liste d'adjacence



1.f Tirage uniforme d'arête dans une liste d'adjacence

En se basant sur le même principe que la matrice d'adjacence, on arrive à élaborer une procédure de tirage uniforme. En effet, il suffit de remplacer le parcours des cases de la matrice par le parcours des listes de chaque sommets. Puisque chaque arête y est représentée on peut simplement tirer un numéro et prendre l'arête y correspondant dans le parcours. Ici, le nombre d'arêtes est donné par la moitié de la somme des longueurs de chaque liste.

1.g Complexité théorique

Premièrement, notre implantation de tirage parcours les sommets et vérifie si le numéro tombe dans la liste de ce sommet. Nous sommes donc en $O(n)$. Deuxièmement, notre opération de contraction commence par trouver les ensembles de sommets correspondant dans s, ce qui se fait en $O(n)$. Ensuite, le calcul des sommets à ajouter est en $O(m)$ tout comme enlever toutes les arêtes (u, v) de la liste v. La concaténation est en temps constant pareil pour la suppression. Enfin, la mise à jour est en $O(n)$ puisque qu'il faut tester l'intersection. On a donc une complexité de contraction en $O(m + n)$. Il faut maintenant prendre en compte le fait que n diminue de 1 à chaque contraction puisqu'on supprime

le sommet du dictionnaire. La complexité théorique de la boucle est donc en :

$$
\begin{aligned}
O(\sum_{i=0}^{n-3} m+n-i) &= O((n-3)m + n^2 - 3n - \frac{(n-3)(n-4)}{2}) \\
&= O(mn + n^2 - n - n^2) \\
&= O(mn)
\end{aligned}
$$

On est bien en complexité quadratique par rapport à la taille de l'entrée si le nombre d'arête est en $O(n)$, sinon on est en $O(n^3)$.

1.h Analyse expérimentale des listes d'adjacences et comparaison avec les matrices

En suivant la même procédure que pour l'analyse avec des matrices d'adjacences, on obtient les courbes de la figure 6. On peut constater l'impact du nombre d'arêtes sur le temps de calculs en observant la courbe obtenu sur des graphes cycliques. On peut également comparer avec notre première implantation (figure 7). Nous pouvons ainsi conclure que dans un graphe suffisamment dense les deux approches nous donne une efficacité similaire. En revanche pour des graphes peu denses la liste d'adjacence est bien plus rapide que la matrice.

Figure 6: Mesures et comparaison des temps de calcul de contraction implémenté avec une liste d'adjacence

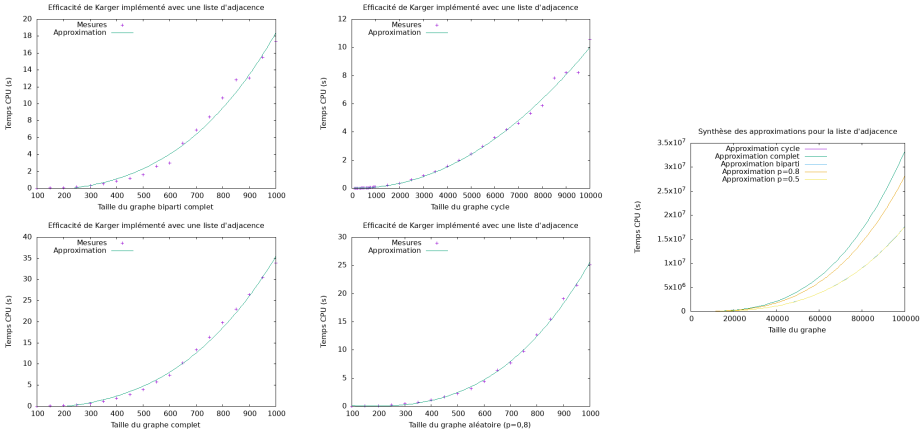
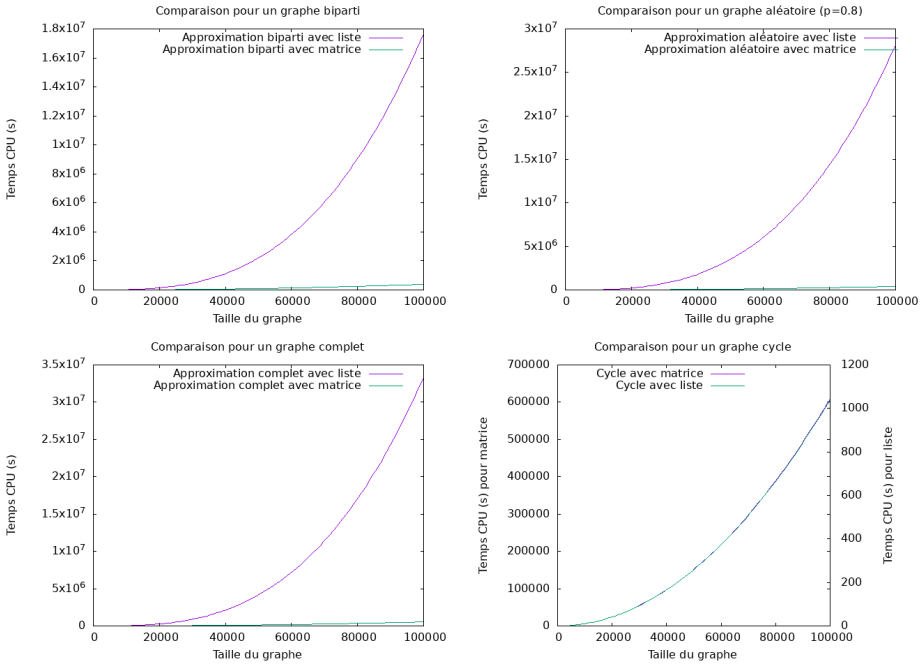


Figure 7: Comparaison des deux implantation



2 Amplification du succès de l'algorithme

2.a Étude expérimentale de la probabilité de succès de Karger

Pour l'étude expérimentale de la probabilité de succès de notre algorithme de Karger, nous avons testé sur les familles de graphes dont nous connaissons la taille de la coupe minimale (figure 8). Pour cet étude, nous avons rajouté les graphes bicomplets qui sont des graphes de taille n comportant deux graphes complets $G1$ et $G2$ de taille $\frac{n}{2}$ reliés par une arête multiple dont un sommet appartient à $G1$ et l'autre à $G2$. Cette famille de graphes va nous permettre d'avoir des résultats plus exploitables, sachant que les autres familles ont des probabilités de succès très élevés. Nous avons itéré la fonction karger 1000 fois sur des familles de graphes de taille 100 (figure 9). Premièrement, nous pouvons remarquer que les probabilités de succès pour les deux structures sont presque identiques, ceux des listes sont quand même inférieure à ceux des matrices. De plus, nous pouvons voir que pour les cycles, notre fonction karger retourne toujours la coupe minimale, toutes les coupes étant minimales. Par conséquent, pour la suite des questions, nous ne traiterons pas les cycles. Pour les graphes complets et bipartis complet, la probabilité est très proche de 1. Cependant, pour les graphes bicomplets la probabilité de succès est très faible par rapport aux autres familles. D'après l'analyse théorique vu en cours, la probabilité de succès de l'algorithme de Karger est supérieure à $\frac{2}{n(n-1)}$. Ici, avec $n = 100$, nous devrions avoir une probabilité de succès supérieure à $\frac{1}{4950}$, ce qui est le cas. Cependant, pour certaines familles, cette borne est absurde car la probabilité de succès est plus proche de 1 que de 0. Par conséquent, nous avons introduit la famille de graphes bicomplets qui a des résultats plus proches de la borne vue en cours.

	cycle	complet	biparti	bicomplet
cmin	2	$n-1$	$n/2$	$\frac{n}{2}-2$

Figure 8: Tableau du nombres d'arêtes de la coupe minimale pour chaque famille de graphes

structure \ graphe	cycle	complet	biparti	bicomplet
matrice	1	0,966	0,971	0,015
liste	1	0,958	0,959	0,014

Figure 9: Tableau des probabilités de succès de la fonction Karger sur les différentes familles de graphes

2.b Karger itéré

Pour la fonction de Karger itéré, comme pour Karger, nous avons qu'une seule implantation pour les deux structures. On initialise m^* et S^* en appelant la fonction Karger. Ensuite, on itére T-1 fois Karger en comparant le nombre d'arête de la coupe actuelle m avec m^*, si m^* est supérieure on affecte m à m^* et S à S^*. On retourne l'ensemble S^* et le nombre d'arête m^* de la coupe minimale trouvée.

2.c Étude expérimentale de la probabilité de succès de Karger itéré

Nous avons réalisé l'étude expérimentale pour Karger itéré :

- **Complet et biparti** Pour ces deux familles puisque la probabilité de Karger simple est déjà très élevée, on a un taux de succès supérieur à $1 - 1/n$ pour $T = 2$ pour toutes les tailles sur lesquelles nous avons fait les tests.
- **Bicomplet** On peut résumer les résultats dans le tableau suivant :

Taille	6	16	26	36	46	56
T=20	1	0.77	0.6	0.47	0.4	0.34

Figure 10: Tableau des taux de succès de la pour T=20

Ici on a fait le choix de fixer T à 20 pour des raisons de temps d'exécutions. Nous avons fait varier T et nous nous sommes arrêtés après un temps raisonnable. Une procédure plus efficace aurait été de mesurer la coupe min évoluée au fur et à mesure des itérations pour mesurer le taux de succès pour tout les T en une seule exécution. On constate que quand la taille augmente la probabilité de succès diminue rapidement pour $T = 20$, comme prévu par le nombre d'itération théorique pour atteindre une probabilité inférieure à $\frac{1}{n}$. En effet, on est bien en dessous de $O(n^2 \log n)$ pour $T = 20$. Un autre problème rencontré est celui de la mesure statistique. En effet, pour avoir des résultats significatifs il nous a fallu mesurer sur un grand nombre d'instances, ici 500, mais raisonnable par rapport au temps d'exécution. C'est ce qui explique probablement le résultat de 1 pour $n = 6$ qui devrait être un peu plus faible étant donné que le T idéal serait 26.

3 Algorithme de Karger-Stein

3.a Complexité de Karger-Stein

Montrons que le temps d'exécution $T(n)$ de l'algorithme de Karger-Stein sur un graphe à n sommets vérifie la relation :

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

On procède à l'analyse d'un tour de boucle. Une opération de contraction partielle effectue $n - \left\lceil 1 + \frac{n}{\sqrt{2}} \right\rceil$ contractions. Puisqu'une opération de contraction se fait en $O(n)$, la complexité d'une contraction partielle est en

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

De plus la taille d'un graphe résultant d'une contraction partielle est $\left\lceil 1 + \frac{n}{\sqrt{2}} \right\rceil$. Étant donné que l'on fait deux appels récursifs, le temps d'exécution $T(n)$ vérifie bien la relation 1. $T(n)$ s'exprime donc sous la forme $T(n) = aT(\frac{n}{b}) + O(n^d)$ avec $a = 2$, $b = \sqrt{2}$, $d = 2$,et $log_b(a) = d$. En appliquant le théorème maître on obtient :

$$T(n) = O(n^2 \log n) \tag{2}$$

3.b Probabilité de succès de Karger-Stein

Montrons que la probabilité de succès $P(n)$ de l'algorithme de Karger-Stein sur un graphe à n sommets vérifie la relation :

$$P(n) \geq 1 - \left(1 - \frac{1}{2}P\left(\left\lceil \frac{n}{\sqrt{2}} + 1 \right\rceil\right)\right)^2 \tag{3}$$

La probabilité que l'algorithme échoue à renvoyer une coupe minimale d'un graphe de taille n correspond à la probabilité que l'algorithme échoue sur les contractions partielles. Les deux contractions partielles étant tirées indépendamment et de taille $\left\lceil 1 + \frac{n}{\sqrt{2}} \right\rceil$, on peut poser:

$$\begin{aligned} P(n) &= 1 - \overline{P(n)} \\ &= 1 - \left(1 - P\left(\left\lceil \frac{n}{\sqrt{2}} + 1 \right\rceil\right)\right)^2 \\ &\geq 1 - \left(1 - \frac{1}{2}P\left(\left\lceil \frac{n}{\sqrt{2}} + 1 \right\rceil\right)\right)^2 \end{aligned}$$

En utilisant $\left(1 - \frac{1}{2}P(n)\right)^2 \geq (1 - P(n))^2$ du fait que $P(n)$ soit une probabilité.

3.c Borne de la probabilité de succès

On veut faire apparaître $\log n$ dans le dénominateur de la borne inférieur. Dans la suite de cette question tout les log sont en base 2. Montrons par récurrence forte la propriété :

$$E(n) : P(n) \geq \frac{1}{\log n + 1} \tag{4}$$

- **Base** Pour $n = 1, \ldots, 6$, $P(n) = 1$ et puisque $1 \geq \frac{1}{\log n+1}$, on vérifie bien la propriété en 4
- **Hérédité** On pose l'hypothèse suivante pour $n_0 > 6$:

$$\forall n,\ n < n_0\ :\ E(n) \tag{5}$$

On sait par 3 que :

$$P(n_0) \geq 1 - \left(1 - \frac{1}{2} P\left(\left\lceil \frac{n_0}{\sqrt{2}} + 1 \right\rceil\right)\right)^2$$

Pour $n_0 > 6$, $\left\lceil \frac{n_0}{\sqrt{2}} + 1 \right\rceil < n_0$, donc en appliquant 5 :

$$\begin{aligned}
P(n_0) &\geq 1 - \left(1 - \frac{1}{2} \cdot \frac{1}{\log\left\lceil \frac{n_0}{\sqrt{2}} + 1 \right\rceil + 1}\right)^2 \\
&\geq \frac{1}{\log\left\lceil \frac{n_0}{\sqrt{2}} + 1 \right\rceil + 1} - \frac{1}{4\left(\log\left\lceil \frac{n_0}{\sqrt{2}} + 1 \right\rceil + 1\right)^2} \\
&\geq \frac{1}{\log\left\lceil \frac{n_0}{\sqrt{2}} + 1 \right\rceil + 1} - \frac{1}{\left(\log\left\lceil \frac{n_0}{\sqrt{2}} + 1 \right\rceil + 1\right)\left(\log\left\lceil \frac{n_0}{\sqrt{2}} + 1 \right\rceil + 2\right)} \\
&\geq \frac{1}{\log\left\lceil \frac{n_0}{\sqrt{2}} + 1 \right\rceil + 2} \\
&\geq \frac{1}{\log n_0 + 1}
\end{aligned}$$

En utilisant que $4(x + 1) \geq x + 2$ pour $x \geq -\frac{2}{3}$ et $\log\left\lceil \frac{n}{\sqrt{2}} + 1 \right\rceil \leq \log n$ pour $n > 6$. On en conclut que (5) $\implies E(n_0)$.

On a prouvé la base et l'hérédité, on a donc bien montré la propriété 4 pour tout n. On a donc :

$$\boxed{P(n) = \Omega\left(\frac{1}{\log n}\right)} \tag{6}$$

3.d Adaptation à Karger-Stein

Afin d'implémenter Karger-Stein, on a besoin d'une fonction de contraction partielle et une fonction de recherche exhaustive. Pour la contraction partielle, nous avons simplement répété l'opération de contraction que nous avions déjà implémenté. Pour la recherche exhaustive, on évalue le cardinal de toutes les coupes possibles, en générant tout les ensembles de sommets possibles. On peut ainsi sélectionner une coupe minimale parmi toutes celles existantes.

3.e Étude expérimentale de la probabilité de succès de Karger-Stein

Comme précédemment, nous allons nous concentrer sur les graphes complets, bipartis complets et "bicomplets".

- **Complet** Pour les graphes complets, l'algorithme de Karger-Stein que nous avons codé semble avoir une très haute probabilité de succès, semblant s'approcher de 1 sur les graphes de taille 10 à 100 que nous avons testé.
- **Biparti** Idem pour les graphes bipartis complets, ce qui s'explique comme pour les graphes complets avec l'existence de n coupes min.
- **Bicomplet** Sur notre famille de graphes, qui pour rappel consiste en deux graphes complets de taille $n/2$ une arête multiple qui va d'un sommet de sous-graphe à un sommet de l'autre sous-graphe, on observe les taux de succès suivant avec une coupe min de cardinal $n/2 - 2$:

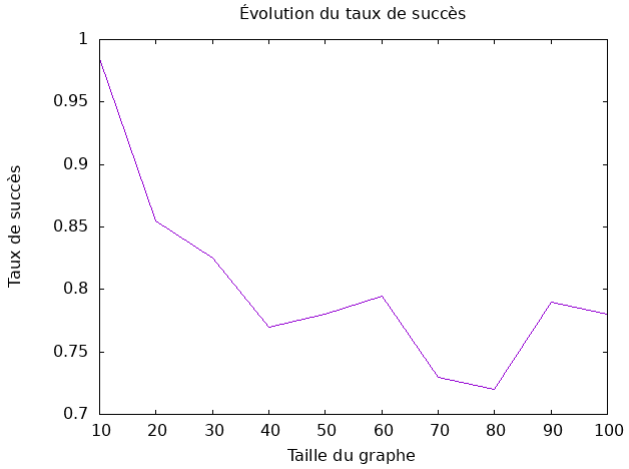


Figure 11: Taux de succès pour les graphes bicomplets