L3-INFO Algorithmique

Université Joseph Fourier GRENOBLE UFR IM²AG

ALGO5 : Travaux dirigés, séance 4
Algorithme randomisé

L'objectif de cette séance est d'étudier un algorithme randomisé résolvant le problème Coupe-Min (MinCut). On dispose d'un graphe non orienté, connexe $G = (S, A)$, avec $|S| = n$ le cardinal de S.

On désire enlever le nombre minimal d'arêtes de G pour obtenir un nouveau graphe dans lequel au moins une paire de noeuds ne sont pas relié par un chemin. On appelle ceci *une coupe*. Une *coupe* dans G est ainsi un ensemble des arêtes telles que leur effacement produit la coupure de G en 2 ou plus graphes séparés.

Plus formellement une coupe $C \subset A$ est un ensemble d'arêtes de G telle que le graphe partiel $G' = (S, A \backslash C)$ ait 2 composantes connexes.

Une solution naïve consiste à énumerer les parties de A et pour chacune regarder si la suppression des arêtes la composant est une coupe. Mais combien y-a-t-il de parties de l'ensemble A ? Pour ne pas "exploser" en temps de calcul on explore une technique se basant sur l'utilisation d'un générateur aléatoire (*random*). On obtient un algorithme dit *randomisé*.

1 Algorithme de génération d'une coupe aléatoire

L'idée est la suivante : choisir une arête de façon aléatoire, la supprimer en contractant le graphe. Répéter l'opération $n - 2$ fois. Les arêtes entre les deux sommets restant forment une coupe du graphe initial.

```
Genere_Coupe(Multigraphe G)
   pour i=1 a n-2
       Choisir une arete a aleatoirement dans G
       G  = contract(G,a)
   retourner G { G contient 2 sommets, les aretes entre ces sommets
                 forment une coupe de G initial }
```

La contraction d'un graphe par une arête rassemble en un seul sommet les deux sommets extrêmités de l'arête et garde les arêtes connectées aux sommets supprimés. La figure 1 montre la contraction du graphe en supprimant l'arête 3. On va ainsi travailler avec un graphe pouvant comporter plusieurs arêtes entre deux sommets ; on parle de multigraphe. De façon à distinguer les arêtes on les identifie avec des entiers. Les sommets sont identifiés avec des lettres majuscules.

2 Application de l'algorithme à un exemple

Soit le graphe de la figure 2. Appliquer l'algorithme précédent après avoir choisi de façon aléatoire 4 arêtes. Quelle coupe obtenez-vous ? Comparer sa taille avec celle obtenue par votre voisin.

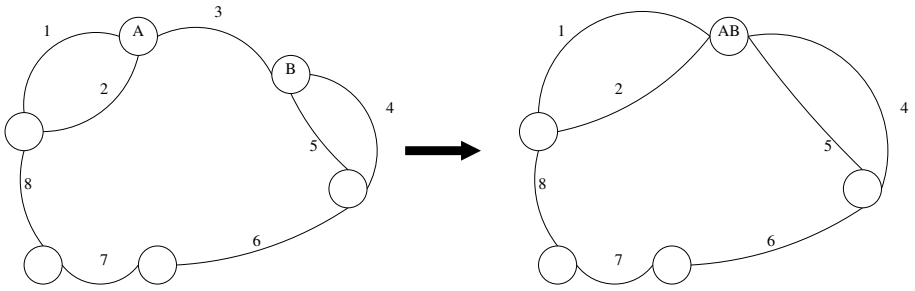

FIGURE 1 – Contraction du multigraphe par l'arête 3 reliant A et B

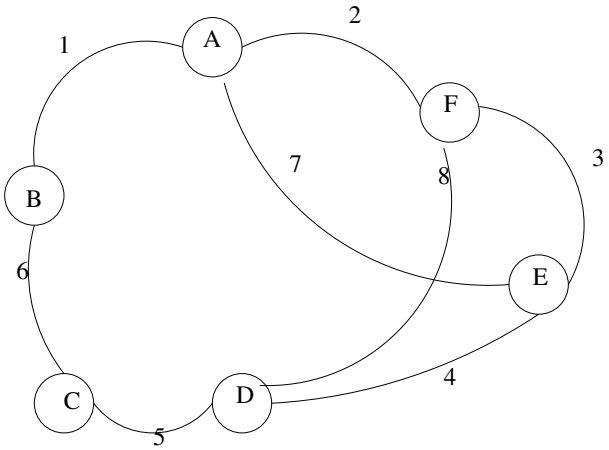

FIGURE 2 – Graphe $G = (S, A)$, $S = \{A, B, C, D, E, F\}$, A={1,2,3,4,5,6,7,8}

3 Analyse de l'algorithme

L'exercice précédent nous a (peut être) donné l'intuition que la coupe générée ne doit pas être trop mauvaise.

Pour analyser cet algorithme on cherche à estimer la probabilité que cet algorithme fournisse bien une coupe minimale. En fait, comme le calcul exact est difficile on va minorer cette probabilité et montrer qu'elle est suffisante pour avoir un algorithme *efficace*.

Notations

- on suppose que la taille de la coupe minimale est k
- on considère un ensemble C d'arêtes qui donne une des coupes minimales
- on note p la probabilité que l'algorithme génère la coupe C

Propriétés du graphe

1. Quelle est la relation entre les degrés des noeuds (nombre d'arêtes d'extrêmité ce noeud) et le nombre d'arêtes total du graphe, soit $|A|$.
2. La coupe min étant de taille k, donner un minorant du nombre d'arêtes du graphe G.
3. Quelle est la probabilité de tirer une arête parmi les k de C à la première itération ? En déduire un minorant de la probabilité de ne pas tomber sur une arête de C au premier tirage aléatoire.

4. Donner un minorant du nombre d'arêtes du graphe G' après la première contraction. Minorer la probabilité de ne pas tomber sur une des arêtes de C au second tirage.
5. Donner un minorant de la probabilité de ne pas tomber sur une des arêtes de C au i-ème tirage.
6. Calculer un minorant de la probabilité que l'algorithme proposé génère la coupe minimale C.

4 A la recherche de la coupe min

Comme on ne génère pas "la bonne coupe" à tous les coups, on répète le tirage et on garde le meilleur résultat. Cela revient à générer un échantillon de taille m.

```
Repete_genere_Coupe (multigraphe G; entier Taille_echantillon)
   C_min = aretes de G
   pour i=1 a Taille_echantillon
      C_courant = Genere_Coupe (G)
      si |C_courant| < |C_min| alors C_min = C_courant
   retourner (C_min) { renvoie la meilleure coupe obtenue }
```

Calcul de la taille de l'échantillon

1. Minorer la probabilité que ce deuxième algorithme fournisse le bon résultat en fonction de m.
2. Montrer que pour une taille d'échantillon de l'ordre de $m = n^2$ la probabilité d'obtenir une coupe minimale est minorée par $\frac{1}{e}$,
3. On se donne une probabilité α et un graphe de taille n ; donner la taille d'échantillon telle que la probabilité d'avoir généré une coupe minimale soit supérieure à α. Calculer la valeur de la taille de léchantillon pour $\alpha = 0.99$ et $n = 100$.