Décomposition de graphes en composantes 3-connexes

David Coudert
david.coudert@inria.fr

31 janvier 2017

Nombre d'étudiants souhaités : 1 (RIF)

Description du sujet

Un graphe $G = (V, E)$ est connexe si pour toute paire de sommets u, v de V, il existe un chemin (suite d'arêtes) permettant d'aller de u à v. Le graphe est k-connexe si il reste connexe après la suppression de n'importe quel sous-ensemble de $k-1$ de ses sommets, et il est k-arête-connexe si il reste connexe après la suppression de n'importe quel sous-ensemble de $k-1$ arêtes. Le plus petit ensemble de sommets (ou d'arêtes) dont la suppression déconnecte le graphe est une coupe minimale.

Savoir tester si un graphe est k-connexe, identifier les composantes k-connexes et les coupes associées sont des briques de base essentielles dans la conception d'algorithmes de graphes et la résolution de problèmes variés d'optimisation. Par exemple, pour le problème du cycle Hamiltonien (existe-t-il un cycle passant une et une seule fois par chacun des sommets du graphe ?), la connaissance d'une coupe à 2 arêtes séparant des composantes 3-connexes permet de séparer le problème et deux sous-problèmes sur des graphes plus petits. En effet, les arêtes de la coupe sont nécessairement sur le cycle. C'est une des astuces que nous avons utilisé pour gagner le challenge FHCP (`http://fhcp.edu.au/fhcpcs`).

Le problème d'identifier les composantes k-connexes d'un graphe se résout en temps linéaire pour $k \leq 3$ [1,2] et il existe de nombreuses implémentations pour $k \leq 2$. Par contre, pour $k = 3$ les implémentations sont rares et cette méthode est donc peu utilisée.

Le premier objectif de ce TER est de réaliser une implémentation efficace de l'algorithme de décomposition d'un graphe en composantes 3-connexes (sommets et arêtes), ainsi que de méthodes permettant de certifier la correction des résultats [3,4]. En plus d'une implémentation indépendante en C/C++ pouvant être librement partagée, il faudra proposer une implémentation pour inclusion dans le logiciel open-source Sagemath (`http://www.sagemath.org`). Nous chercherons ensuite à exploiter cette implémentation pour accélérer la résolution de problèmes d'optimisation sur les graphes dans Sagemath.

Lieu

Inria Sophia Antipolis

Prérequis

Algorithmique ; Théorie des graphes ; Langages Python, C/C++

Informations complémentaires

1. C. Gutwenger and P. Mutzel. A Linear Time Implementation of SPQR-Trees. In : Proc. of the 8th International Symposium on Graph Drawing (GD'00), pp. 77–90, 2001
2. J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. SIAM J. Comput., 2(3) :135–158, 1973.
3. K. Mehlhorn, A. Neumann and J. M. Schmidt. Certifying 3-Edge-Connectivity. Algorithmica, 77(2) :309–335, 2017.
4. J. M. Schmidt. A simple test on 2-vertex- and 2-edge-connectivity. Inf. Process. Lett. 113(7) : 241–244, 2013.