Practical Algorithm for Cluster Deletion on Cographs

Let G be a cograph with discriminating cotree (T, t). That is, $G = G[\rho]$ is recursively determined as [1]

$$G[u] = \begin{cases} \bigcup\limits_{v \in \mathrm{child}(u)} G[v] & \text{if } t(u) = 0 \\ \bigtriangledown\limits_{v \in \mathrm{child}(u)} G[v] & \text{if } t(u) = 1 \\ \{x\} & \text{if } x \text{ is a leaf} \end{cases} \tag{1}$$

Here $\cup$ and $\bigtriangledown$ denote the disjoint union and the join of graphs, respectively.

As an immediate consequence, the size $\kappa(G)$ of a maximal clique of G can also be computed recursively [1]

$$\kappa(G[u]) = \begin{cases} \max\limits_{v \in \mathrm{child}(u)} \kappa(G[v]) & \text{if } t(u) = 0 \\ \sum\limits_{v \in \mathrm{child}(u)} \kappa(G[v]) & \text{if } t(u) = 1 \\ 1 & \text{if } x \text{ is a leaf} \end{cases} \tag{2}$$

The Cluster Deletion problem ask for a partition of $V(G)$ such that each class $V_i = V(G_i)$ is clique, i.e., a complete subgraph G_i of G, such that the total number of edge, $\sum |E(G_i)|$ is maximal [3]. For cographs, Cluster Deletion has a very simple solution: As shown in [2], every optimal solution is obtained by iteratively finding a maximal clique G_1 in G and then deleting G_1 from G. As a consequence, a simple modification of eq.(2) also pertains to Cluster Deletion.

Denote by $\mathcal{P}(u)$ the optimal clique partition of the cograph implies by the subtree $T(u)$ of the discriminating cotree (T, t). We think of $\mathcal{P}(u) := [Q_1(u), Q_2(u), \dots]$ as an ordered list, such that $|Q_i(u)| \geq |Q_j(u)|$ if $i < j$. It will be convenient assume that the list contains an arbitary number of empty sets. The $\mathcal{P}(u)$ satisfy the following recursion:

$$\mathcal{P}(u) = \begin{cases} \bigcup\limits_{v \in \mathrm{child}(u)} \mathcal{P}(v) & \text{if } t(u) = 0 \\ \left[\bigcup\limits_{v \in \mathrm{child}(u)} Q_i(v) \mid i = 1, 2, \dots \right] & \text{if } t(u) = 1 \\ [\{x\}, \emptyset, \dots] & \text{if } x \text{ is a leaf} \end{cases} \tag{3}$$

To see that this is correct it suffices to observe: (i) the largest clique in a disjoint union of graphs is the largest clique in any of its components. The optimal clique partition of a disjoint union of graphs thus is the union of the optimal clique partitions of the constitutent connected components. (ii) For a join of two or more graphs G_i, each maximum size clique Q is the join of a maximum size clique of each constituent. The next largest clique disjoint from $Q = \bigtriangledown_i Q_i$

thus is the join of a largest cliques disjoint from Q_i in each constituent graph G_i. Thus a greedy clique partition of G is obtained by size ordering the clique partitions of G_i and joining the the k-largest cliques from each. If the list of cliques in a G_i is exhausted, it simply does not contribute anymore, which can be represented as join with an empty clique $Q_i = \emptyset$. (iii) Trivally, the optimal clique partition of a single vertex graph consists only of that vertex.

The dynamic programming recursion (3) operates directly on the discriminating cotree (T, t) of the cograph G. For each node u, the effort is proportional to $|L(T(u))| \log(\deg(u))$ for the $\deg(u)$-wise merge sort step if $t(u) = 0$ proportional to $|L(T(u))|$ for the merging of the k-th largest clusters for $t(u) = 1$. Using $\sum_u \deg(u)|L(T(u))| \leq |L(T)| \sum_u \deg(u) \leq 2|E(T)||L(T)|^2 \leq 2|L(T)|^2$. Since $|V(T)| \leq 2|L(T)| - 1$ we obtain a quadratic upper bound on running time.

References

[1] D. G. Corneil, H. Lerchs, and L. Steward Burlingham. Complement reducible graphs. *Discr. Appl. Math.*, 3:163–174, 1981.

[2] Y. Gao, D. R. Hare, and J. Nastos. The cluster deletion problem for cographs. *Discrete Math.*, 313(23):2763–2771, 2013.

[3] R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Appl. Math.*, 144(1-2):173–182, 2004.