Maintaining Connectivity Classes in A Graph

Reza Zamani Nasab

1 Problem

Consider the following problem:

Problem: Design a data structure which can maintain the k-edge-connectivity classes of vertices of a graph, while we can insert edges and vertices in the graph. In other words, the data structure have to support these three operations on the represented graph G:

1. `InsertVertex`(u)
2. `InsertEdge`(u,v)
3. `InSame-`k`-Class`(u,v) : returns back a boolean, indicating whether vertices u and v are in the same k-edge-connectivity class.

2 Previous Work

- In [1] a solution for case $k = 1, 2$ is presented. Achieved time bounds are $O(\log^2 n)$ for $k = 1$ and $O(\log^4 n)$ for $k = 2$ (we have seen ideas involved for the case $k = 1$ in class).
- In [2] a solution to case $k = 3$ is presented. The main idea is maintaining connectivity structure among cycles of the graph (a *tree of cycles*).
- In [3] a solution to case $k = 4$ is presented. Assuming q query operations, m edge insertions, and n vertex insertions, total time $O(q + m + n \log n)$ will be spent (while worst case query time is constant). Again the idea is to maintain a simplified graph structure called *cactus tree*. This also presents a solution for the case of general k, while we know the original graph is $(k - 1)$-edge-connected.
- In [4] a solution to case $k = 5$ is presented. With the above settings, it needs time $O(q + m + n \log^2 n)$ (again with worst case query time of $O(1)$).
- In [5] a $(1 \pm o(1))$-approximation solution for $k = O(\log^{O(1)} n)$ is presented, with amortized time bound $O(\sqrt{n})$. The idea comes out of a combinatorial result which maintains the minimum cut using a *greedy tree packing*, a family of spanning trees with certain properties.

3 Possible Improvements

- [Thorup] Is it possible to get rid of the $(1 \pm o(1))$ approximation factor?
- [Thorup] Can we repeat the bound mentioned in [1] for higher values of k ?
- [Thorup] Is there a way to have time bounds in [5] in worst case sense? For this one, most probably we need a more careful proof of the main combinatorial lemma in [5].
- [Thorup] Can we get similar result (or anything interesting) for vertex connectivity?
- [Jeff] In all of these schemes there is a solution with near linear update time-bound (up to logarithmic factors) and constant time-bound for queries. What is the maximum k for which such a scheme can be found?
- What about randomized paradigms? Is there a simpler but randomized solution for the existing solved cases which answers correctly most of the time?

References

[1] Jacob Helm et al, *Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity*, http://citeseer.ist.psu.edu/holm97polylogarithmic.html

[2] Galil, Z. and Italiano, G. F.: *Maintaining the 3-edge-connected classes of a graph on-line*, SIAM J. Comput. 22 (1993), no.1, 11-28, http://citeseer.ist.psu.edu/context/2003670/0

[3] Yefim Dinitz, Jeffery Westbrook, *Maintaining the classes of 4-edge-connectivity in a graph online*, Algorithmica 1995, http://citeseer.ist.psu.edu/dinitz95maintaining.html

[4] Yefim Dinitz, Ronit Nossenson, *Incremental Maintenance of the 5-Edge-Connectivity Classes of a Graph*, Lecture Notes In Computer Science; Vol. 1851, http://portal.acm.org/citation.cfm?id=672606

[5] Mikkel Thorup, *Fully-dynamic min-cut*, Annual ACM Symposium on Theory of Computing, 2001, pp 224-230, http://portal.acm.org/citation.cfm?id=380804&coll=portal&dl=ACM