Fully Dynamic Strong Connectivity and Reachability in Digraphs

Gregory Morse and Tamás Kozsik

ELTE Eötvös Loránd University, Budapest, Hungary, Faculty of Informatics, 3in Research Group, Martonvásár, Hungary

morse@inf.elte.hu kto@elte.hu

Although computing all-pairs reachability and strongly connected components (SCCs) have well-known algorithms [1, 2], less attention has been given to their incremental and decremental variants, and in particular to fully dynamic approaches that avoid data structures specialized exclusively for edge insertions or deletions.

In the context of compilers and static analysis tools, the control flow graph (CFG) often needs to make reachability queries or identify SCCs when evaluating, refactoring or optimizing code. We present a deterministic fully dynamic algorithm for maintaining strong connectivity and reachability in directed graphs, designed for compiler control-flow graphs and static analysis workloads.

An attempt at moving towards an efficient, deterministic and precise fully dynamic algorithm is thus made based on utilizing ideas of well-known non-incremental algorithms [3, 4]. In this paper, efficient data structures for storing SCCs are used [5] but transitive closure is represented in a manner that requires propagation. For dense graphs and sparse graphs which are expected to contain many SCCs or large SCCs, it can be expected to have good performance. As expected, edge insertions are handled more efficiently than deletions: insertions primarily merge SCCs using Union-Find [5], whereas deletions may require rediscovery of SCC structure, analogous to decremental Even-Shiloach (ES) tree behavior [6]. Intuitively, adding edges can merge SCCs into a larger one and thus the removal of an edge has to effectively perform a discovery process.

Although this algorithm does not achieve state-of-the-art bounds on the incremental [7] or decremental [8] cases for SCCs, for digraphs, to the best of our knowledge, no prior work addresses fully dynamic strong connectivity and all-pairs reachability simultaneously in a deterministic and precise setting suitable for compiler CFGs. For all-pairs reachability, many algorithms are known but only in this isolated context. As well the best known algorithms are probabilistic or randomized without decrease of their time complexity and not precise and deterministic as is needed in the given context of CFGs which tend to be sparse and contain a relatively low number of vertices.

The algorithm is based on an offline SCC-based reachability method that maintains SCCs in $\mathcal{O}(n)$ space and $\mathcal{O}(n+m)$ time, with an additional $\mathcal{O}(n^2)$ space and $\mathcal{O}(mn+n^2)$ worst-case time for reachability propagation where n and m are the number of vertices and edges in the graph respectively. For total update time on insertion and deletion, the presented algorithm has the same storage space and worst case time complexity, yet in practice the cases in which the worst case would occur are rare.

Empirical evaluation on random dense and sparse graphs, as well as real-world CFGs extracted from Windows and Linux binaries, shows that the proposed approach significantly outperforms repeated offline recomputation in practice. While worst-case behavior remains quadratic, such cases are rare in CFG-like graphs, where SCC structure changes incrementally. Consider the example of random graphs with $\frac{n^2}{2}$ edges inserted, a single offline computation performed, and finally all of the edges being deleted. As the number of vertices (and thereby edges) in the graph increases, the incremental case for a sequence of $\frac{n^2}{2}$ edge insertions is only slower by approximately a constant factor a_1 from the final single offline computation. Likewise, the decremental case for a sequence of $\frac{n^2}{2}$ edge deletions will have an increase in time of about $a_2 n$ where a_2 is a constant. This comparison is in fact generous to the offline algorithm as a graph with this many edges will possibly and very probabilistically contain only one large component. Without considering pathological cases, it will perform much faster than repeated offline computations after each edge modification in the graph.

Testing on random sparse graphs including some with certain property constraints as well as actual CFGs taken from selected Windows and Linux programs was performed to evaluate the algorithm.

References

[1] R. Tarjan, "Depth first search and linear graph algorithms," *SIAM JOURNAL ON COMPUTING*, vol. 1, no. 2, 1972.

[2] R. W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, vol. 5, p. 345, June 1962.

[3] P. Purdom, "A transitive closure algorithm," *BIT Numerical Mathematics*, vol. 10, pp. 76–94, Mar 1970.

[4] E. Nuutila, "An efficient transitive closure algorithm for cyclic digraphs," *Information Processing Letters*, vol. 52, 1994.

[5] R. E. Tarjan, *Data Structures and Network Algorithms, 2. Disjoint Sets*, pp. 23–31. Society for Industrial and Applied Mathematics, 1983.

[6] Y. Shiloach and S. Even, "An on-line edge-deletion problem," *J. ACM*, vol. 28, p. 1–4, Jan. 1981.

[7] B. Haeupler, T. Kavitha, R. Mathew, S. Sen, and R. E. Tarjan, "Incremental cycle detection, topological ordering, and strong component maintenance," *ACM Trans. Algorithms*, vol. 8, Jan. 2012.

[8] A. Bernstein, M. Probst, and C. Wulff-Nilsen, "Decremental strongly-connected components and single-source reachability in near-linear time," in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, (New York, NY, USA), p. 365–376, Association for Computing Machinery, 2019.

Acknowledgement

The research has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013, Thematic Fundamental Research Collaborations Grounding Innovation in Informatics and Infocommunications).