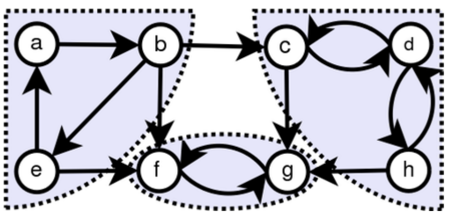
WIKIPEDIA

Strongly connected component

In the mathematical theory of directed graphs, a graph is said to be **strongly connected** or **diconnected** if every vertex is reachable from every other vertex. The **strongly connected components** or **diconnected components** of an arbitrary directed graph form a partition into subgraphs that are themselves strongly connected. It is possible to test the strong connectivity of a graph, or to find its strongly connected components, in linear time (that is, Θ(V+E)).



Graph with strongly connected components marked

Contents

Definitions

A directed graph is called **strongly connected** if there is a path in each direction between each pair of vertices of the graph. In a directed graph *G* that may not itself be strongly connected, a pair of vertices *u* and *v* are said to be strongly connected to each other if there is a path in each direction between them.

The binary relation of being strongly connected is an equivalence relation, and the induced subgraphs of its equivalence classes are called **strongly connected components**. Equivalently, a **strongly connected component** of a directed graph *G* is a subgraph that is strongly connected, and is maximal with this property: no additional edges or vertices from *G* can be included in the subgraph without breaking its property of being strongly connected. The collection of strongly connected components forms a partition of the set of vertices of *G*.

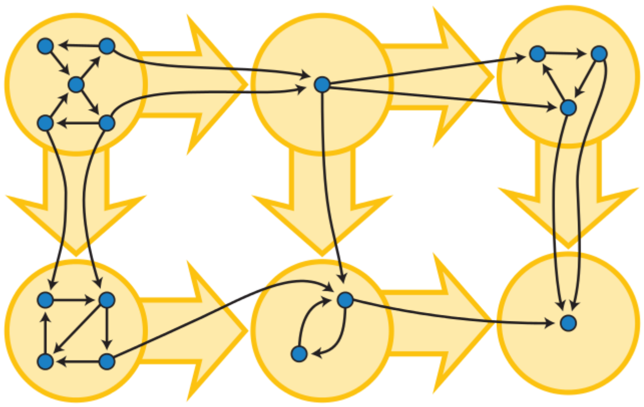
If each strongly connected component is contracted to a single vertex, the resulting graph is a directed acyclic graph, the **condensation** of *G*. A directed graph is acyclic if and only if it has no strongly connected subgraphs with more than one vertex, because a directed cycle is strongly connected and every nontrivial strongly connected component contains at least one directed cycle.

Algorithms

DFS-based linear-time algorithms

Several algorithms based on depth first search compute strongly connected components in linear time.

- Kosaraju's algorithm uses two passes of depth first search. The first, in the original graph, is used to choose the order in which the outer loop of the second depth first search tests vertices for having been visited already and recursively explores them if not. The second depth first search is on the transpose graph of the original graph, and each recursive exploration finds a single new strongly connected component.[1] It is named after S. Rao Kosaraju, who described it (but did not publish his results) in 1978; Micha Sharir later published it in 1981.[2]
- Tarjan's strongly connected components algorithm, published by Robert Tarjan in 1972,[3] performs a single pass of depth first search. It maintains a stack of vertices that have been explored by the search but not yet assigned to a component, and calculates "low numbers" of each vertex (an index number of the highest ancestor reachable in one step from a descendant of the vertex) which it uses to determine when a set of vertices should be popped off the stack into a new component.


The yellow directed acyclic graph is the condensation of the blue directed graph. It is formed by contracting each strongly connected component of the blue graph into a single yellow vertex.

- The path-based strong component algorithm uses a depth first search, like Tarjan's algorithm, but with two stacks. One of the stacks is used to keep track of the vertices not yet assigned to components, while the other keeps track of the current path in the depth first search tree. The first linear time version of this algorithm was published by Edsger W. Dijkstra in 1976.[4]

Although Kosaraju's algorithm is conceptually simple, Tarjan's and the path-based algorithm require only one depth-first search rather than two.

Reachability-based Algorithms

Previous linear-time algorithms are based on depth-first search which is generally considered hard to parallelize. Fleischer et al.[5] in 2000 proposed a divide-and-conquer approach based on reachability queries, and such algorithms are usually called reachability-based SCC algorithms. The idea of this approach is to pick a random pivot vertex and apply forward and backward reachability queries from this vertex. The two queries partition the vertex set into 4 subsets: vertices reached by both, either one, or none of the searches. One can show that a strongly connected component has to be contained in one of the subsets. The vertex subset reached by both searches forms a strongly connected components, and the algorithm then recurses on the other 3 subsets.

The expected sequential running time of this algorithm is shown to be $O(n \log n)$, a factor of $O(\log n)$ more than the classic algorithms. The parallelism comes from: (1) the reachability queries can be parallelized more easily (e.g. by a BFS, and it can be fast if the diameter of the graph is small); and (2) the independence between the subtasks in the divide-and-conquer process. This algorithm performs well on real-world graphs,[6] but does not have theoretical guarantee on the parallelism (consider if a graph has no edges, the algorithm requires $O(n)$ levels of recursions).

Blelloch et al.[7] in 2016 shows that if the reachability queries are applied in a random order, the cost bound of $O(n \log n)$ still holds. Furthermore, the queries then can be batched in a prefix-doubling manner (i.e. 1, 2, 4, 8 queries) and run simultaneously in one round. The overall span of this algorithm is $\log_2 n$ reachability queries, which is probably the optimal parallelism that can be achieved using the reachability-based approach.

Applications

Algorithms for finding strongly connected components may be used to solve 2-satisfiability problems (systems of Boolean variables with constraints on the values of pairs of variables): as Aspvall, Plass & Tarjan (1979) showed, a 2-satisfiability instance is unsatisfiable if and only if there is a variable v such that v and its complement are both contained in the same strongly connected component of the implication graph of the instance.[8]

Strongly connected components are also used to compute the Dulmage–Mendelsohn decomposition, a classification of the edges of a bipartite graph, according to whether or not they can be part of a perfect matching in the graph.[9]

Related results

A directed graph is strongly connected if and only if it has an ear decomposition, a partition of the edges into a sequence of directed paths and cycles such that the first subgraph in the sequence is a cycle, and each subsequent subgraph is either a cycle sharing one vertex with previous subgraphs, or a path sharing its two endpoints with previous subgraphs.

According to Robbins' theorem, an undirected graph may be oriented in such a way that it becomes strongly connected, if and only if it is 2-edge-connected. One way to prove this result is to find an ear decomposition of the underlying undirected graph and then orient each ear consistently.[10]

See also

- Clique
- Connected component
- Modular decomposition

References

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 22.5, pp. 552–557.
2. Micha Sharir. A strong connectivity algorithm and its applications to data flow analysis. *Computers and Mathematics with Applications* 7(1):67–72, 1981.
3. Tarjan, R. E. (1972), "Depth-first search and linear graph algorithms", *SIAM Journal on Computing*, **1** (2): 146–160, doi:10.1137/0201010 (https://doi.org/10.1137%2F0201010)
4. Dijkstra, Edsger (1976), *A Discipline of Programming*, NJ: Prentice Hall, Ch. 25.
5. Fleischer, Lisa K; Hendrickson, Bruce; and Pinar, Ali. On identifying strongly connected components in parallel (http://www.sandia.gov/~apinar/papers/irreg00.pdf). IPDPS 2000.
6. Hong, Sungpack; Rodia, Nicole C; and Olukotun, Kunle. On fast parallel detection of strongly connected components (SCC) in small-world graphs (https://ppl.stanford.edu/papers/sc13-hong.pdf). SC 2013.
7. Blelloch, Guy; Gu, Yan; Shun, Julian; and Sun, Yihan. Parallelism in Randomized Incremental Algorithms (http://www.cs.cmu.edu/~ygu1/paper/SPAA16/Incremental.pdf). SPAA 2016. doi:10.1145/2935764.2935766.
8. Aspvall, Bengt; Plass, Michael F.; Tarjan, Robert E. (1979), "A linear-time algorithm for testing the truth of certain quantified boolean formulas", *Information Processing Letters*, **8** (3): 121–123, doi:10.1016/0020-0190(79)90002-4 (https://doi.org/10.1016%2F0020-0190%2879%2990002-4).
9. Dulmage, A. L. & Mendelsohn, N. S. (1958), "Coverings of bipartite graphs", *Can. J. Math.*, **10**: 517–534, doi:10.4153/cjm-1958-052-0 (https://doi.org/10.4153%2Fcjm-1958-052-0).
10. Robbins, H. E. (1939), "A theorem on graphs, with an application to a problem on traffic control", *American Mathematical Monthly*, **46**: 281–283, doi:10.2307/2303897 (https://doi.org/10.2307%2F2303897), JSTOR 2303897 (https://www.jstor.org/stable/2303897).

External links

- Java implementation for computation of strongly connected components (http://code.google.com/p/jbpt/) in the jBPT library (see StronglyConnectedComponents class).
- C++ implementation of Strongly Connected Components (http://www.geeksforgeeks.org/tarjan-algorithm-find-strongly-connected-components/)