Faster Algorithms for Maximal 2-connected Subgraphs in Digraphs

based on Henzinger et al. ICALP 2015 [1] and Chechik et al. SODA 2017 [2]

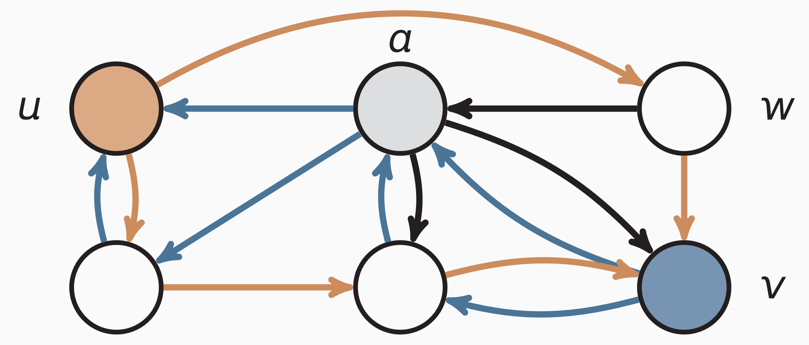
Shiri Chechik[1], Thomas D. Hansen[2], Monika Henzinger[3], Giuseppe F. Italiano[4], Sebastian Krinninger[3], *Veronika Loitzenbauer*[5], and Nikos Parotsidis[4]

[1]Tel Aviv University [2]Aarhus University [3]University of Vienna [4]Università di Roma "Tor Vergata" [5]Bar-Ilan University

Highlights of Algorithms, June 2017, Berlin, Germany

Definitions

Input: Directed graph G with n vertices V and m edges E



2-edge-connected u and v are strongly connected after any single edge removed, i.e., there are 2 edge-disjoint paths from u to v and from v to u *as in example*

2-vertex-connected u and v are strongly connected after any single vertex removed—*not the case in this example*

bridge an edge whose removal increases the number of strongly connected components, find all in $O(m)$ time [3]—*in the example the edge from u to w is a bridge*

articulation point a vertex whose removal increases the number of strongly connected components, find all in $O(m)$ time [3]—*the vertex a is an articulation point*

2-edge-connected graph all pairs of vertices are 2-edge-connected

2-vertex-connected graph all pairs of vertices are 2-vertex-connected and $n \geq 3$

2-edge-connected block set of vertices that are pairwise 2-edge-connected, paths might use edges outside of the block, computable in $O(m)$ time [4]

2-vertex-connected block set of vertices that are pairwise 2-vertex-connected, paths might use vertices outside of the block, in $O(m)$ time [5]

2-edge-connected subgraphs maximal 2-edge-connected (induced) subgraphs

2-vertex-connected subgraphs maximal 2-vertex-connected (induced) subgraphs

In *undirected* graphs 2-edge/vertex-connected blocks and subgraphs coincide and can be computed in $O(m)$ time [6].

Results

The **2-edge-connected** and the **2-vertex-connected subgraphs** of a directed graph can be computed in $\mathbf{O(\min\{n^2, m^{3/2}\})}$ **time**.

For constant k the **k-edge-connected subgraphs** can be computed in $O(\min\{n^2, m^{3/2}\} \log n)$ time and the **k-vertex-connected subgraphs** in $O(\min\{n^2, m^{3/2}\} \cdot n)$ time. Extends with slightly better dependence on m to undirected graphs.

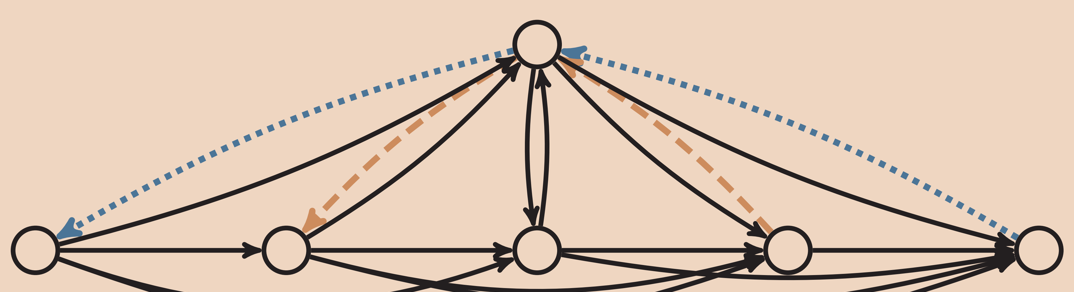
This poster: 2-edge-connected subgraphs in $O(n^2)$ [1] and $O(m^{3/2})$ [2] time

Basic Algorithm for 2-Edge-Connected Subgraphs

while the graph G contains a bridge **do**
  delete bridges from G in time $O(m)$
output the strongly connected components of G

Running time [7, 3]: $O(mn)$

Example with $\Theta(n)$ Iterations



The blue, dotted edges are bridges in the first iteration. After their deletion the orange, dashed edges become bridges. This can happen $\Theta(n)$ times.

1-Edge-Out Set

Idea: Find directed edge cut of size ≤ 1 that separates "small" vertex set S from $V \setminus S$ in time proportional to size of S

1-edge-out set of u minimal vertex set with $u \in S$ and ≤ 1 edge to $V \setminus S$

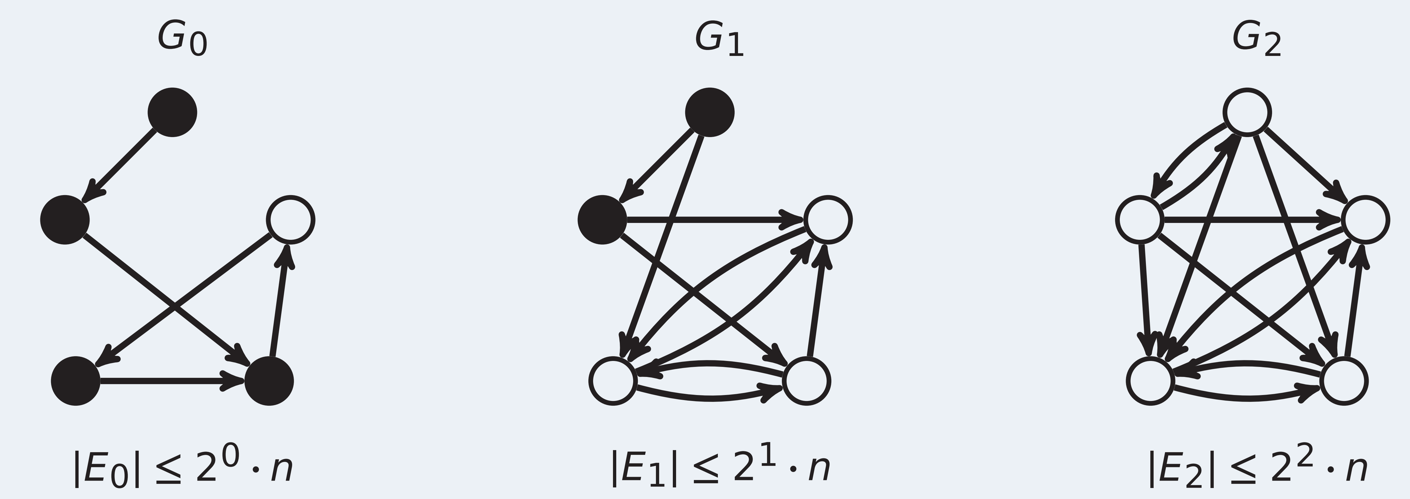
- G is 2-edge-connected if and only if G does not contain any 1-edge-out set
- Every 2-edge-connected subgraph is either completely contained in any 1-edge-out set or does not intersect with it

Dense Graphs: $O(n^2)$ Time Algorithm [1]

if the graph G contains a 1-edge-out set S **then**
  recurse on $G[S]$ and $G[V \setminus S]$
else
  output G as a 2-edge-connected subgraph

Observation: vertices of 1-edge-out set S have outdegree at most $|S|$
$\Rightarrow$ for $|S| \leq 2^i$ search in graph G_i that contains first 2^i outgoing edges of each vertex

Search for 1-Edge-Out Set in Sparse Subgraph



Hierarchical Graph Decomposition [8]
- White vertices have outdegree $\leq 2^i$
- 1-edge-out set S with $|S| \leq 2^i \Rightarrow S$ is white
- 1-edge-out set induced by white vertices in $G_i \Rightarrow$ 1-edge-out set in G

Running Time
- Use linear time algorithm based on [6, 7] on G_i, from $i = 0$ to $\log n$ till set found
- Identifies 1-edge-out S in time $O(n \cdot \min\{|S|, |V \setminus S|\})$
- $O(n)$ time per vertex, $O(n^2)$ in total

Sparse Graphs: $O(m^{3/2})$ Time Algorithm [2]

Observation: Any new 1-edge-out set has lost outgoing edge
$\Rightarrow$ $O(m)$ searches from vertices that lost outgoing edges in total are enough to find all

while the graph G contains bridges **do**
  delete bridges and recompute strongly connected components
  mark vertices that lost adjacent edges
  while exists marked vertex u **do**
    search for 1-edge-out or 1-edge-in set S of u with $\leq \sqrt{m}$ edges
    if found, remove edges between S and $V \setminus S$
    unmark u and mark vertices that lost adjacent edges
output the strongly connected components of G

Local Search for 1-Edge-Out Sets with Depth-First Search

Observation: If we find path from u that ends outside 1-edge-out set of u, reverse edges of path and discover the 1-edge-out set with graph traversal from u

Finding a path with DFS:
- Assume 1-edge-out set S of u with $\leq \Delta$ edges and outgoing edge (x, y) exists
- Run DFS for $2\Delta + 1$ edges
- DFS leaves S exactly once, using the edge (x, y)
- Assign each vertex v as weight the number of edges in DFS-subtraversal from v

Property 1: Vertices with weight $> \Delta$ form a path from u in DFS tree
Property 2: More than Δ weight can be picked up only with edges outside of S
$\Rightarrow$ Path of vertices with weight $> \Delta$ goes from u to a vertex outside of S

Running Time
- Identify 1-edge-out or 1-edge-in set S with $\leq \Delta$ edges in time $O(\Delta)$ starting from endpoints of deleted edges $\Rightarrow$ total time $O(m \cdot \Delta)$
- If none of with $\leq \Delta$ exists, do iteration of basic algorithm in time $O(m)$; this can happen at most m/Δ times $\Rightarrow$ total time $O(m^2/\Delta)$
- $\Delta = \sqrt{m} \Rightarrow$ total time $O(m\sqrt{m})$

References

[1] M. Henzinger, S. Krinninger, and V. Loitzenbauer. *Finding 2-Edge and 2-Vertex Strongly Connected Components in Quadratic Time*, ICALP 2015, p. 713–724.

[2] S. Chechik, T. D. Hansen, G. F. Italiano, V. Loitzenbauer, and N. Parotsidis. *Faster Algorithms for Computing Maximal 2-Connected Subgraphs in Sparse Directed Graphs*, SODA 2017, p. 1900–1918.

[3] G. F. Italiano and L. Laura and F. Santaroni. *Finding strong bridges and strong articulation points in linear time*, Theoretical Computer Science 447, 74–84, 2012.

[4] L. Georgiadis, G. F. Italiano, L. Laura, and N. Parotsidis. *2-Edge Connectivity in Directed Graphs*, ACM Transactions on Algorithms 13(1), 9:1–9:24, 2016, announced at SODA'15.

[5] L. Georgiadis, G. F. Italiano, L. Laura, and N. Parotsidis. *2-Vertex Connectivity in Directed Graphs*, ICALP 2015, p. 605–616.

[6] R. E. Tarjan. *Depth-First Search and Linear Graph Algorithms*, SIAM Journal on Computing 1(2), 146–160, 1972.

[7] R. E. Tarjan. *Edge-Disjoint Spanning Trees and Depth-First Search*, Acta Informatica 6(2), 171–185, 1976.

[8] K. Chatterjee and M. Henzinger. *Efficient and Dynamic Algorithms for Alternating Büchi Games and Maximal End-component Decomposition*, Journal of the ACM 61(3), 15:1–15:40 (2014), announced at SODA'11 and SODA'12.