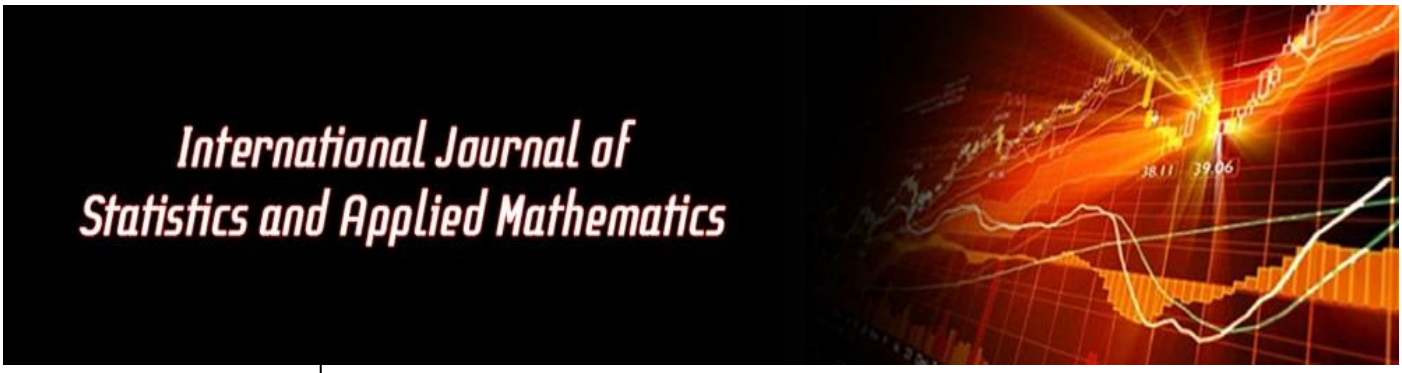
International Journal of Statistics and Applied Mathematics 2018; 3(1): 177-181



ISSN: 2456-1452
Maths 2018; 3(1): 177-181

www.mathsjournal.com
Received: 24-11-2017
Accepted: 25-12-2017

Y Anitha Kumari
Lecturer in Mathematics,
J.K.C. College, Guntur,
Andhra Pradesh, India

K Padmavathi
Lecturer in Mathematics,
P.A.S. College, Pedanandipadu,
Andhra Pradesh, India

Correspondence
Y Anitha Kumari
Lecturer in Mathematics,
J.K.C. College, Guntur,
Andhra Pradesh, India

Passage linear based expansion of connectivity algorithms

Y Anitha Kumari and K Padmavathi

Abstract
This paper presents notions of 1- and 2-connectivity. It starts with 1-connectivity of directed graphs, and it then examines 2-connectivity of undirected graphs. Depth-first search is the method of choice to calculate low order connectivity information. The algorithms which are designed for connectivity properties are originally due to Tarjan [1]. This paper follows the path-based development of [2], which simplifies the algorithms to eliminate the depth-first spanning tree.

Keywords: Connectivity, strong component, directed graph

Introduction

Strong Components of a Directed Graph

In this paper, G = (V, E) is a directed graph.

Definitions

- For two vertices a and v, *a uv-path* is a path starting at a and ending at v.
- A directed graph G = (V, E) is *strongly connected* if for every two distinct vertices a and v, there is a uv-path and a vu-path.
- In general, a directed graph will not be strongly connected. But the vertices can be partitioned into blocks that are strongly connected, according to this definition: two vertices u v are in the same *strong component (SC)* if and only if they can reach each other, i.e., there is a uv-path and a vu-path. This defines a partition of V since it is an equivalence relation.
- For any directed graph *G*, contracting each SC to a vertex gives the *strong component graph* or *condensation* of *G*.
- A *tournament* is a directed graph *G* such that each pair of vertices is joined by exactly one edge. This models a round robin tournament, where edge (x, y) represents the inference that player x beat player y.

Inference

- Let C be a cycle in a graph G. All vertices of C are in the same SC. Contracting the vertices of cycle C to a single vertex yields a graph with the same SC graph as G.
- The SC graph is always a Directed Acyclic Graph.
- A topological numbering of the SC graph of a tournament gives a ranking of the players. To see why, note that if player x is in an SC with lower topological number than y, then the tournament contains the edge (x, y) not (y, x). Thus SC number I contains the players that are unequivocally in the top tier they all beat all other players. SC number 2 contains the 2nd tier players they all beat all other players except those in tier 1, etc.
- All the vertices on a cycle belong to the same SC. In inference the SC graph is formed by repeatedly contracting cycles, until no cycle remains.
- A sink *s* is a vertex of the SC graph. In inference the SC's are {s} and the SC's of *G s.*
- A high-level algorithm for finding the SC graph is given below. It repeatedly contracts a cycle or deletes a sink.

- Next we present a linear-time depth-first search algorithm for finding the strong components and the SC graph of a given directed graph.

Algorithm1: Strong Components
Input: directed graph *G = (V, E); Output:* strong components of *G*
repeat until G has no vertices: grow a dfs path P until a sink or a cycle is found
sink *s:* mark *{s}* as an SC & delete *s* from *P* & *G cycle c:contract the vertices of C*

- Each iteration grows P by starting with the previous P and extending it, if possible.
- The algorithm has a low-level implementation that finds the SC graph in linear time [11]. Sinks are deleted and Cycles are contracted using a stack to represent P and another stack to give the boundaries of contracted vertices in P.
- The algorithm discovers each SC as a sink of the SC graph. So the SC's can be numbered in topological order by the method of topological order Algorithm.

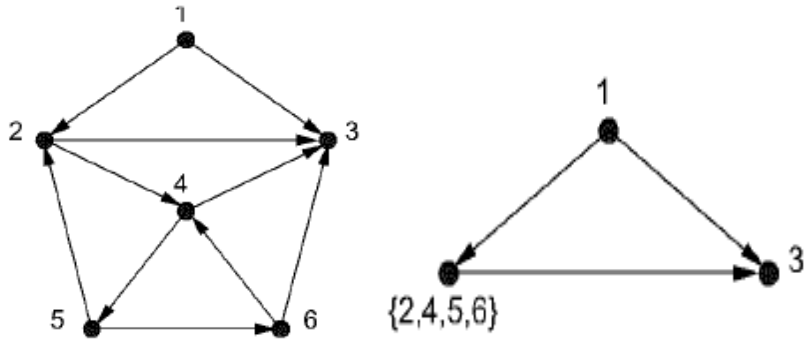


Fig 1: Strong components of a directed graph.

An elementary misperception is that a strongly connected graph has a Hamiltonian cycle. The component {2, 4, 5, 6} illustrates that this is not always true.

- Each vertex is labeled by its preorder number followed by its low point value.

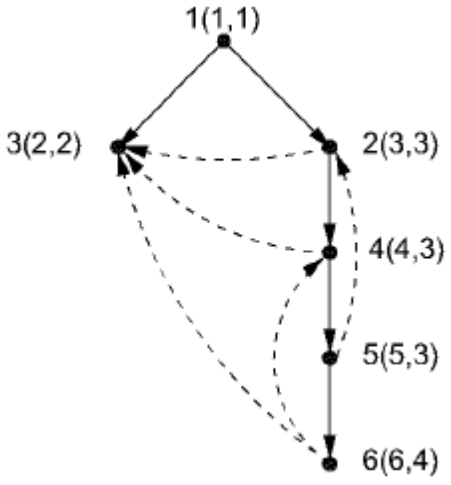


Fig 2: Execution of strong component algorithm.

- EX 1**:** Suppose we number the vertices of an arbitrary directed graph by topologically numbering the SC graph, and then listing first the vertices in SC number 1, then the vertices in SC number 2, etc. The adjacency matrix of the graph with new vertex numbers is upper block triangular.
- The first linear-time algorithm for strong components is due to Tarjan [1]. It computes a value called *lowpoint(v)* for each vertex *v. lowpoint(v)* is the lowest-numbered vertex (in preorder) in v's SC that is reachable from v by a path of (0 or more) tree edges followed by a back or cross edge *(lowpoint(v)* equals *v* if no smaller numbered vertex can be reached). The vertices with *lowpoint(v) = v* are the "roots" of the strong components.
- A third linear-time strong component algorithm is due to Sharir [3] and Kosaraju (unpublished; see also [4]). It does a depth-first search, followed by a second depth-first search on the reverse graph. This makes good sense
the first search discovers which vertices can reach which others, and the second search discovers which vertices can be reached by which others.

Examples
Figure 1 shows a directed graph, its three strong components, and its SC graph. Each strong component is strongly connected.

This is because no edge goes from a higher numbered SC to a lower numbered SC It is upper triangular except for the block corresponding to SC {b, d, e}.

	a	b	d	e	c
a	0	1	1	1	1
b	0	0	0	1	1
d	0	1	0	0	1
e	0	0	1	0	1
c	0	0	0	0	0

Fig 3: Upper block triangular adjacency matrix.

- EX 2: Example 1 shows how the SC graph is used to speed up operations on sparse matrices like Gaussian elimination, matrix inversion, finding eigenvalues, etc. The given matrix *M* is interpreted as a directed graph, with *mzj* corresponding to edge (i, j). The adjacency matrix of Example1 is constructed, and the 1 for each edge (i, j) is replaced by the value *m2j*. The resulting block upper triangular matrix has less fill-in for Gaussian elimination and nice properties for other matrix operations [7].
- Figure 4 below illustrates the execution of the algorithm on the graph.

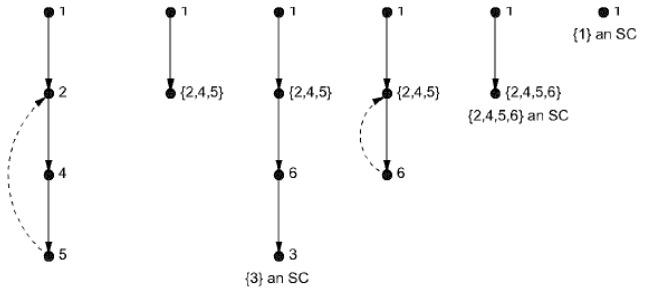


Fig 4: Execution of strong component algorithm.

- Figure 5 below shows a tournament and its SC graph. Player a is first, players *b, d, e* are in the 2nd tier, and player c is last
- A Markov chain is *irreducible* if the graph of its (nonzero) transition probabilities is strongly connected.

Observation

- The algorithm is very simple to code and is covered in many textbooks. It can be appreciably slower than the other two algorithms, because it makes two passes over the graph and has larger memory requirement.

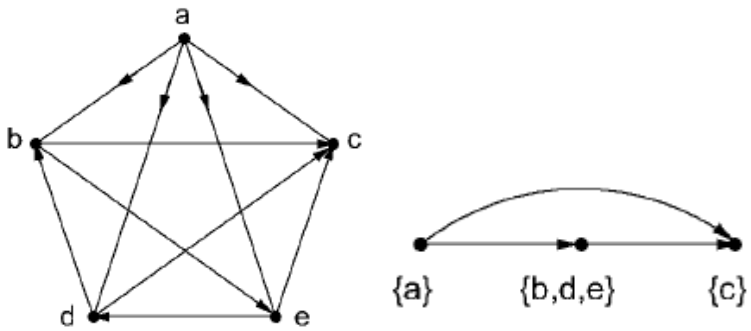


Fig 5: Tournament and its SC graph.

Bridges and Cut points of an Undirected Graph

In this paper *G = (V, E)* is a connected undirected graph.

Definitions

- A vertex *v* is an *cutpoint or articulation point,* if *G - v* is not connected. A graph is *biconnected* if it has no cutpoint.
- *A biconnected component* is a maximal subgraph that has no cutpoint.
- An edge e is *a bridge* if *G - e* is not connected. An edge is a bridge if and only if it's not in any cycle. A graph is *bridgeless* if it has no bridges.
- Let *B* be the set of all bridges of *G.* The *bridge components (BCs)* of *G* are the connected components of *G - B.* Equivalently a BC is the induced subgraph on a maximal set of vertices, any of which can reach any other without crossing a bridge.
- Contracting each BC to a vertex gives a tree, the *bridge tree.*
- An *orientation* of an undirected graph assigns a unique direction to each edge.
- *A perfect matching* of an undirected graph *G* is a spanning subgraph in which every vertex has degree exactly 1.

Examples

- Figure 6 shows a graph with 3 bridges, 6 cut points, and 7 biconnected components. It illustrates that an end of a bridge is a cut point unless it has degree one. However, a cut point need not be the end of a bridge.

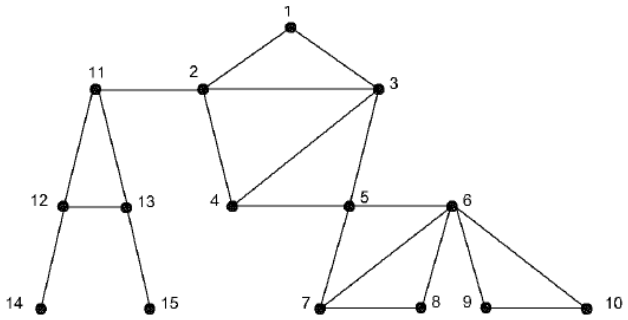


Fig 6: Undirected graph with bridges and cutpoints.

- If a communications network (e.g., Internet) has a bridge, that link's failure disables communication, i.e., there are sites that cannot send messages to each other. If the network has an articulation point, that site's failure also disables communication.

Inference

- All vertices on a cycle are in the same BC. In inference the bridge tree is formed by repeatedly contracting cycles.
- A vertex x of degree < 1 is a vertex of the bridge tree. In inference the BC's are {x} and the BC's of G - x.
- The following is a high level algorithm for finding the bridges and bridge tree. It has a linear-time implementation almost identical to the strong component algorithm. We call the last vertex x of a dfs path a dead end if x has degree < 1.

Algorithm 3**:** Bridges
Input: connected undirected graph *G = (V, E)*
Output: bridge components and bridges of *G*
repeat until G has no vertices:
grow a dfs path P until a cycle is found or a dead end is reached cycle C: contract the vertices of C
dead end x: mark {x} as a BC
if x has degree 1, then mark its edge as a bridge of G

- A similar linear-time algorithm finds the cutpoints and biconnected components of an undirected graph [2].
- The original linear-time dfs algorithm of Hopcroft and Tarjan for cutpoints and biconnected components [1] is based on the idea of lowpoints.

Start with a dfs tree T. Assume that the vertices are numbered in discovery order and that each vertex is identified with its discovery number. Define
lowpoint(v) = min{v} U {w : some back edge goes from a descendant of *v* to *w}* Hopcroft and Tarjan proved that G is biconnected if and only if
vertex 1 has exactly one child (which must be vertex 2);
lowpoint(2) = 1;
each vertex *w* > 2 has *lowpoint(w)* < v, where *v* is the parent of *w.*
The cutpoints have a similar characterization.
Lowpoint is easy to compute in a bottom-up pass over T, since
lowpoint(v) = min{v} U *{lowpoint(w) : w* a child of *v}* U *{w : (v, w)* a back edge}

More Examples
Figure 7 below illustrates the execution of the Bridges algorithm on the graph

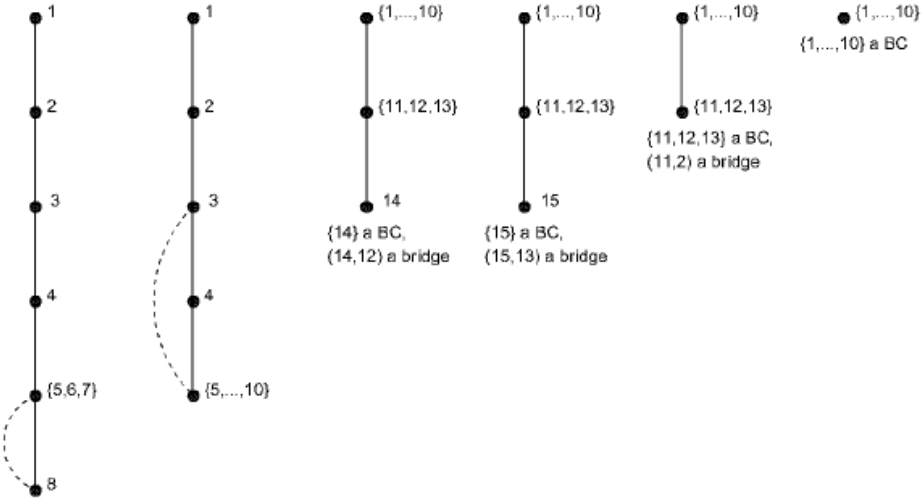


Fig 7: Execution of bridge algorithm.

- Figure 8 below illustrates Robbins's Theorem that a connected undirected graph has a strongly connected orientation if and only if it is bridgeless [5]. If one of the horizontal edges is deleted, making the other a bridge, then the graph has no strongly connected orientation.

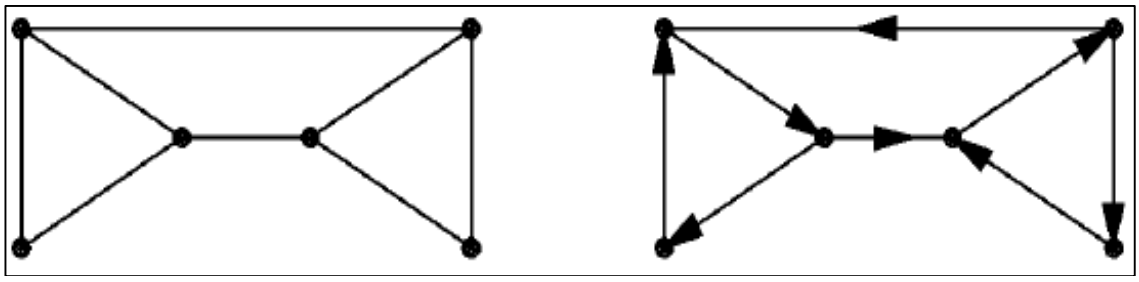


Fig 8: Undirected graph and strongly connected orientation.

EX: Kotzig's Theorem states that a unique perfect matching must contain a bridge of G. Figure 9 shows a graph with a unique perfect matching matched edges are drawn heavy. Note that deleting the bridge of the matching gives another graph with a unique perfect matching. This idea can be used to efficiently find a unique perfect matching or show it does not exist.

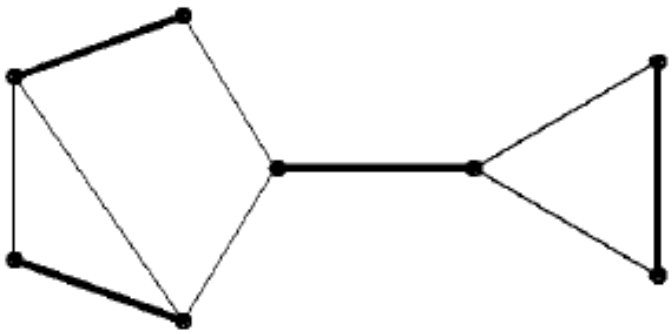


Fig 9: Graph with a unique perfect matching.

- Whitney's Flipping Theorem asserts that a graph is planar if and only if each biconnected component is planar [6].

References

1. Tarjan RE. Depth-first search and linear graph algorithms, SIAM J. Comput. 1972; 1:146-160.
2. Gabow HN. Path-based depth-first search for strong and biconnected components, Inf. Proc. Letters. 2000; 74:107-114.
3. Sharir M. A strong-connectivity algorithm and its application in data flow analysis, Comp. and Math. With Applications. 1981; 7:67-72.
4. Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms, Second Edition, McGraw-Hill, 2001.
5. Robbins HE. A theorem on graphs, with an application to a problem in traffic control, Amer. Math. Monthly. 1939; 46:281-283.
6. Whitney H. Non-separable and planar graphs, Trans. Amer. Math. Soc. 1932; 34:339-362.
7. Harary F. Graph Theory, Addison-Wesley, Reading MA, 1969.
8. Aho AV, Sethi R, Sharir M. A strong-connectivity algorithm and its application in data flow analysis, Comp. and Math. With Applications. 1981; 7:67-72.
9. Alstrup S, Harel D, Lauridsen PW, Thorup M. Dominators in linear time. SIAM J. Comput. 1999; 28:2117-2132.
10. Auslander L, Parter SV. On imbedding graphs in the plane, J. Math. and Mech. 1961; 10:517-523.
11. Gabow HN. An ear decomposition approach to approximating the smallest 3-edge connected spanning subgraph of a multigraph, Proc. 13th Annual ACM-SIAM Symp. on Disc. Algorithms, 2002, 84-93.