Information and Computation 261 (2018) 248–264

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

2-vertex connectivity in directed graphs

Loukas Georgiadis [a], Giuseppe F. Italiano [b,1], Luigi Laura [c], Nikos Parotsidis [b,*]

[a] Department of Computer Science & Engineering, University of Ioannina, Greece
[b] Dipartimento di Ingegneria Civile e Ingegneria Informatica, Università di Roma "Tor Vergata", Italy
[c] Dipartimento di Ingegneria Informatica, Automatica e Gestionale, "Sapienza" Università di Roma, Italy

ARTICLE INFO

Article history:
Received 31 May 2015
Available online 8 February 2018

Keywords:
Connectivity
Directed graph
Dominators
Flow graph
Strong articulation points

ABSTRACT

Given a directed graph, two vertices v and w are 2-*vertex-connected* if there are two internally vertex-disjoint paths from v to w and two internally vertex-disjoint paths from w to v. In this paper, we show how to compute this relation in $O(m+n)$ time, where n is the number of vertices and m is the number of edges of the graph. As a side result, we show how to build in linear time an $O(n)$-space data structure, which can answer in constant time queries on whether any two vertices are 2-vertex-connected. Additionally, when two query vertices v and w are not 2-vertex-connected, our data structure can produce in constant time a "witness" of this property, by exhibiting a vertex or an edge that is contained in all paths from v to w or in all paths from w to v.

1. Introduction

Let $G=(V,E)$ be a directed graph (digraph), with m edges and n vertices. Digraph G is *strongly connected* if there is a directed path from each vertex to every other vertex. The *strongly connected components* of G are its maximal strongly connected subgraphs. Two vertices $u, v \in V$ are *strongly connected* if they belong to the same strongly connected component of G. A vertex (resp., an edge) of G is a *strong articulation point* (resp., a *strong bridge*) if its removal increases the number of strongly connected components. A digraph G is 2-*vertex-connected* if it has at least three vertices and no strong articulation points; G is 2-*edge-connected* if it has no strong bridges. The 2-*vertex-* (resp., 2-*edge-*) *connected components* of G are its maximal 2-vertex- (resp., 2-edge-) connected subgraphs.

Differently from undirected graphs, in digraphs 2-vertex and 2-edge connectivity have a much richer and more complicated structure. To see an example of this, let v and w be two distinct vertices and consider the following natural 2-vertex and 2-edge connectivity relations, defined in [7,11,17]. Vertices v and w are said to be 2-*vertex-connected* (resp., 2-*edge-connected*), and we denote this relation by $v \leftrightarrow_{2\mathrm{v}} w$ (resp., $v \leftrightarrow_{2\mathrm{e}} w$), if there are two internally vertex-disjoint (resp., two edge-disjoint) directed paths from v to w and two internally vertex-disjoint (resp., two edge-disjoint) directed paths from w to v (note that a path from v to w and a path from w to v need not be edge- or vertex-disjoint). A 2-*vertex-connected block* (resp., 2-*edge-connected block*) of a digraph $G=(V,E)$ is defined as a maximal subset $B \subseteq V$ such that $u \leftrightarrow_{2\mathrm{v}} v$ (resp., $u \leftrightarrow_{2\mathrm{e}} v$) for all $u, v \in B$. In undirected graphs, the 2-vertex- (resp., 2-edge-) connected blocks are identical to the 2-vertex-

* Corresponding author.
E-mail addresses: loukas@cs.uoi.gr (L. Georgiadis), giuseppe.italiano@uniroma2.it (G.F. Italiano), laura@dis.uniroma1.it (L. Laura), nikos.parotsidis@uniroma2.it (N. Parotsidis).

[1] Partially supported by MIUR, the Italian Ministry of Education, University and Research, under Project AMANDA (Algorithmics for MAssive and Networked DAta).

https://doi.org/10.1016/j.ic.2018.02.007

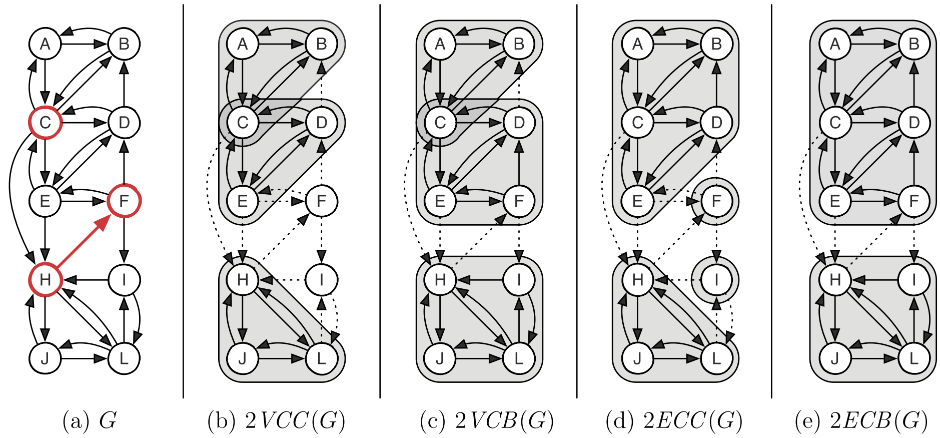

Fig. 1. (a) A strongly connected digraph G, with strong articulation points and strong bridges shown in red (better viewed in color in the web version of this article). (b) The 2-vertex-connected components of G. (c) The 2-vertex-connected blocks of G. (d) The 2-edge-connected components of G. (e) The 2-edge-connected blocks of G. Note that vertices E and F are in the same 2-vertex- (resp., 2-edge-) connected block of G since there are two internally vertex-disjoint (resp., edge-disjoint) paths from E to F and from F to E. However, E and F are not in the same 2-vertex (resp., 2-edge-) connected component of G.

(resp., 2-edge-) connected components. As shown in Fig. 1, this is not the case for digraphs. Indeed, two vertices v and w of a directed graph are in the same 2-vertex- (resp., 2-edge-) connected component if and only if (i) $v \leftrightarrow_{2v} w$ (resp., $v \leftrightarrow_{2e} w$), i.e., they are in the same 2-vertex- (resp., 2-edge-) connected block, and (ii) the internal vertex-disjoint (resp., edge-disjoint) paths are contained entirely inside the 2-vertex- (resp., 2-edge-) connected component. Put in other words, differently from the undirected case, in digraphs 2-vertex- (resp., 2-edge-) connected components do not encompass the notion of pairwise 2-vertex (resp., 2-edge) connectivity among its vertices: two vertices may be 2-vertex- (resp., 2-edge-) connected but not necessarily in the same 2-vertex- (resp., 2-edge-) connected component (see Fig. 1). We note that pairwise 2-connectivity (given by 2-vertex- and 2-edge-connected blocks) may be relevant in several applications, where one is often interested in local properties, e.g., checking whether two vertices are 2-connected, rather than in global properties.

Due to their more complicated structure, it is not surprising that 2-connectivity problems on directed graphs appear to be more difficult than on undirected graphs. For undirected graphs it has been known for over 40 years how to compute all bridges, articulation points, 2-edge- and 2-vertex-connected components in linear time, by simply using depth first search [18]. In the case of digraphs, however, the very same problems have been much more challenging. Indeed, it has been shown only few years ago that all strong bridges and strong articulation points of a digraph can be computed in linear time [10]. Furthermore, the best current bound for computing the 2-edge- and the 2-vertex-connected components in digraphs is not even linear, but $O(n^2)$, and was achieved only very recently by Henzinger et al. [9], improving previous $O(mn)$ time bounds [12,16]. Finally, it was shown also very recently how to compute the 2-edge-connected blocks of digraphs in linear time [7].

In this paper, we complete the picture on 2-connectivity for digraphs by presenting the first algorithm for computing the 2-vertex-connected blocks in $O(m+n)$ time. Our bound is asymptotically optimal and it improves sharply over a previous $O(mn)$ time bound by Jaberi [11]. As a side result, our algorithm constructs an $O(n)$-space data structure that reports in constant time if two vertices are 2-vertex-connected. Additionally, when two query vertices v and w are not 2-vertex-connected, our data structure can produce, in constant time, a "witness" by exhibiting a vertex (i.e., a strong articulation point) or an edge (i.e., a strong bridge) that separates them. We are also able to compute in linear time a sparse certificate for 2-vertex connectivity, i.e., a subgraph of the input graph that has $O(n)$ edges and maintains the same 2-vertex connectivity properties. Our algorithm follows the high-level approach of [7] for computing the 2-edge-connected blocks. However, the algorithm for computing the 2-vertex-connected blocks is much more involved and requires several novel ideas and non-trivial techniques to achieve the claimed bounds. In particular, the main technical difficulties that need to be tackled when following the approach of [7] are the following:

- First, the algorithm in [7] maintains a partition of the vertices into approximate blocks, and refines this partition as the algorithm progresses. Unlike 2-edge-connected blocks, however, 2-vertex-connected blocks do not partition the vertices of a digraph, and therefore it is harder to maintain approximate blocks throughout the algorithm's execution. To cope with this problem, we show that these blocks can be maintained using a more complicated forest representation, and we define a set of suitable operations on this representation in order to refine and split blocks. We believe that our forest representation of the 2-vertex-connected blocks of a digraph can be of independent interest.
- Second, in [7] we used a *canonical decomposition* of the input digraph G, in order to obtain smaller *auxiliary* digraphs (not necessarily subgraphs of G) that maintain the original 2-edge-connected blocks of G. A key property of this decomposition was the fact that any vertex in an auxiliary graph G_r is reachable from a vertex outside G_r only through