International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 4 | ISSN : 2456-3307

# Algorithm for 2-Vertex Connectivity in Directed Graphs

**T Manohar Reddy**
P.G. Student, Department of Computer Engineering, JNTU College of Engineering, Anantapur, Andhra Pradesh, India

## ABSTRACT

Graph theory is very important in solving many problems efficiently. Each graph is made up of vertices and edges. Common notation used to represent a graph is G = (V, E) where V is a set of vertices and E is a set of edges. The dynamics of graphs were understood and explored well in the literature while there is insufficient research related to directed graphs. There are many problems to be solved with respect to graphs. For instance, 2-vertex connectivity is an important problem to be addressed. Recently Georgiadis et al. focused on this problem and found that two vertices v and w are 2-vertex connected when there are two internally vertex -disjoint paths coming from v to w and two internally vertex disjoint paths from w to v. Many kinds of combinations were explored in their work. However, it is useful to have more investigation on this. In this paper we reviewed directed graphs with 2-vertex connectivity with some algorithms. Our study reviewed that the approaches can be used effectively to solve select problems in the real world.

**Keywords :** Graph Theory, Directed Graphs, 2-Edge Connectivity

## I. INTRODUCTION

There are many applications that are based on graphs. For instance network related things are generally solved in the form of graphs. Of late there is cloud computing technology where thousands of physical machines and more than that of virtual machines are running. The mapping of virtual machines to physical machines effectively can be done by employing graph theory. Thus graph theory plays vital role in the real world. Another example is model transformations in software engineering. A model can be represented as a graph and it can be subjected to transformation that can again be represented in the form of another graph.Accordingly graph theory is commonly used in exceptional purposes.Customary notation used to symbolize a graph is G = (V, E) where V is a set of vertices and E is a set of edges. The dynamics of graphs were understood and explored well in the literature while there is insufficient research related to directed graphs. There are many problems to be solved with respect to graphs. Graphs are of two type known as directed graphs and undirected graphs. The directed graphs are shown in Figure 1 while the undirected graph is shown in Figure 2.

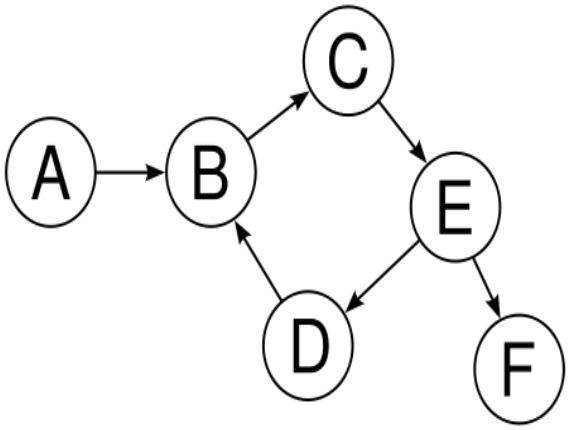



**Figure 1:** Illustrates directed graph

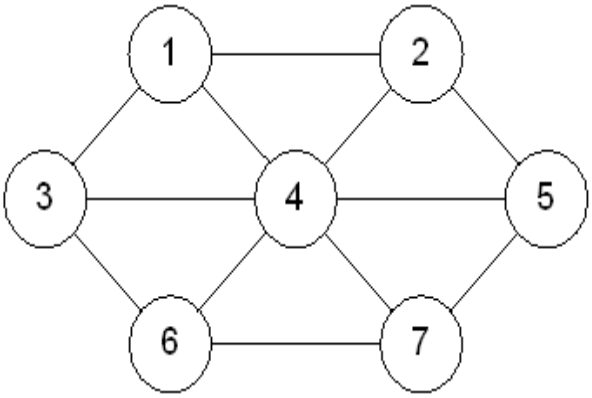



**Figure 2:** Illustrates undirected graph

As shown in Figure 1, the directed graph contains directions from one vertex to another using some arrow head which is missing in undirected graph shown in Figure 2. In this paper, the focus is on the directed

graphs. Especially it throws light into 2-Vertex connectivity in directed graphs. Our contributions in this paper include the algorithm that can be used to have 2-vertex connectivity graphs and the descriptions that can help in understanding the utility of such graphs in the real world. The structure of the paper is as follows. Section 2 threw light into literature review related to graph theory. Section 3 presents proposed algorithms related to directed graphs with 2-vertex connectivity. Section 4 presents the directed graphs with variants. Section 5 concludes the paper.

**2-connectivity**

Given an undirected graph G = (V, E), an edge is a bridge if its removal increases the number of connected components of G. Graph G is 2-edge-connected if it has no bridges. The 2 edge-connected components of G are its maximal 2-edge-connected subgraphs. Two vertices v and w are 2-edge-connected if there are two edge-disjoint paths between v and w: we denote this relation by $v \leftrightarrow_e w$. Equivalently, by Menger's Theorem [28], v and w are 2-edge-connected if the removal of any edge leaves them in the same connected component. Analogous definitions can be given for 2-vertex connectivity. In unique, a vertex is an articulation point if its removal increases the quantity of connected components of G. A graph G is 2-vertex-connected if it has minimum three vertices and no articulation points. The 2-vertex-connected components of G are its maximal 2-vertex-connected subgraphs.Note that the condition on the minimum number of vertices in a 2-vertex-connected graph disallows degenerate 2-vertex-connected components consisting of one single edge. Two vertices v and w are 2-vertex-connected if there are two internally vertex-disjoint paths between v and w. we denote this relation by $v \leftrightarrow_{2v} w$. If v and w are 2-vertex-connected then Menger's Theorem implies that the removal of any vertex different from v and w leaves them in the same connected component. The converse does not necessarily hold, since v and w may be adjacent but not 2-vertex-connected. It is easy to show that $v \leftrightarrow 2e\ w$ (resp., $v \leftrightarrow 2v\ w$) if and only if v and w are in a same 2-edge-connected (resp., 2-vertex- connected) component. All bridges, articulation points, 2-edge- and 2-vertex-connected components of undirected graphs can be computed in linear time essentially by the same Algorithm[16].

## II. RELATED WORK

This section provides review of literature related to directed graphs and related aspects including 2-Vertex connectivity. Paudel et al. [1] used directed graphs for solving problems related to node detection. They proposed a linear time algorithm in order to have a heuristic approach in making use of directed graphs. Such graphs are subjected to the case of node detection problem which is considered to be NP-hard. Ghaffari and Su [2] studied distributed graph problems. They focused on degree splitting or in order words portioning the edges. They used undirected degree splitting for edge-colouring and other applications in the real world. Merker [3] on the other hand focused on decomposing highly edge-connected graphs in order to convert them into fixed tree. They named the method as Tree Decomposition Conjecture (TDC). In the process of decomposition, sub graphs are generated and with certain constraints. Esperet et al. [4] studied yet different aspects of graphs. They investigated on flows and additive bases in graphs. With that they generated many highly edge-connected graphs.

Florini et al. [5] proposed an approximation algorithm for tree augmentation. Towards this end, they used Chvatal Gmory cuts. It was the problem related to network design. Fundamentally the tree augmentation is used in networking studies. Henginzer et al. [6] focused on conditional hardness with respect to sensitivity problems. Conditional lower bounds are used for setting sensitivity and solve many real world problems. They used a conjecture known as Boolean Matrix Multiplication (BMM) which is one of the combinatorial algorithms used for solving problems. He et al. [7] explored on matching covered graphs for the purpose of problems related to perfect matching. They used them as matching-covered graphs where edges are removable. Characterization of such graphs hasutility in the real world applications.

Gutin et al. [8] focused on several problems that can be solved using graphs. They investigated on connectivity preservation, edge deletions and path-contractions. They used biconnectivity deletion, strong connectivity, path-contraction with connectivity preservation constraints with appropriate parameter settings. Baswana et al. [10] studied depth first search (DFS) algorithms for solving problems that are pertaining to graphs. They focused on incremental DFS based on the

observations made from existing algorithms. They studied the dynamics of insertion, deletion and update of graphs dynamically and randomly to evaluate performance of their method. Georgiadis et al. [9] investigated on decremental data structures that have special utility in graph problems. They focussed on the decremental data structures that support sensitivity queries. They used directed graphs with connectivity and dominators. Maintaining a dominor tree is an important application of such data structures.

Horvath et al. [11] focused on finite graphs and maximal sub graphs in order to investigate problems in the real world. Flow semi group introduced by John Rhodes played vital role in graph applications. The rationale behind this is that it was able to provide many variants of graphs to deal with different problems. Zhang et al. [12] studied problems related to graphs. Especially they threw light into flip-distance between alpha-orientations of graphs. They formulated necessary conditions needed to achieve this. Archdeacon et al. [13] proposed digraphs with 2-regular planar. More details needed to understand them include embedding process and directed means of doing it. Such graphs are widely used for solving problems where natural analogue in parameter setting is required. Holm et al. [14] proposed a data structure to have planar graphs with edge contractions. They proposed algorithms in order to improve running times and used computations such as MST. Georgiadis et al. [15] proposed mechanisms to understand and use digraphs with 2-vertex connectivity. In this paper we explore same algorithm in detailed manner related to digraphs that reflect 2-vertex connectivity.

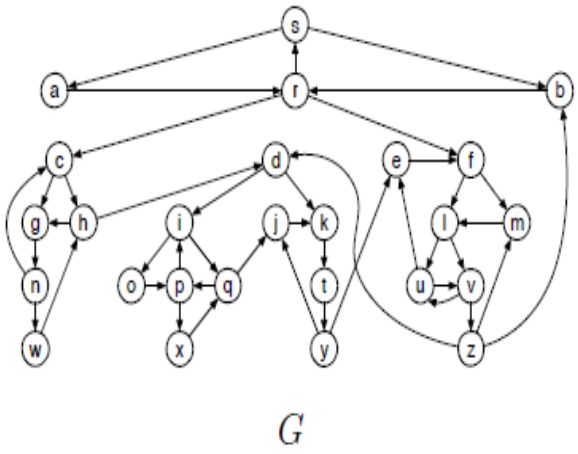
## III. ALGORITHM FOR DIRECTED GRAPHS

We discussed algorithm to have diversified 2-vertex connectivity graphs and exploring them in a better way. There are many terms related to graphs. They are strongly connected digraph, 2-vertex connected components, 2-vertex connected blocks, 2-edge connected components, and 2-edge connected blocks. More details on this can be found in [17].

**Algorithm 1: Linear-time computation of the vertex-resilient blocks**
**of a strongly connected digraph G = (V, E)**

**Step 1:** Choose an arbitrary vertex s ∈V as a start vertex. Compute the dominator tree D(s).
For any vertex v, let $\hat{C}(v)$ be the set containing v and the childrenof v in D(s).
Initialize the block forest F by associating block$\hat{C}(v)$ with everyvertex w ∈$\hat{C}(v)$,
for all vertices v that are not a leaves in D(s).
**Step 2:** Compute the auxiliary graphs $G_r$ for all vertices r that are not leaves in D(s).
**Step 3:** Process the vertices of D(s) in bottom-up order. For each auxiliary graphH = $G_r$ with
r not a leaf in D(s) do:
**Step 3.1:** Compute the dominator tree T = $D^R_H(r)$.
**Step 3.2:** $(x, S, T) \coloneqq$ s-t max-flow$(G, s, t)$.
**Step 3.3:**$(x', T', S') \coloneqq$ s-t max-flow$(G, t, s)$.
**If** $x' < x$
**Step3.4:**$x \coloneqq x', S \coloneqq S', T \coloneqq T'$
**Step3.5:**Add edge $(s, t)$ with weight $x$ to $A$
**Step3.6:**Construction$(G, s, N \cap S)$
Construction$(G, t, N \cap T)$

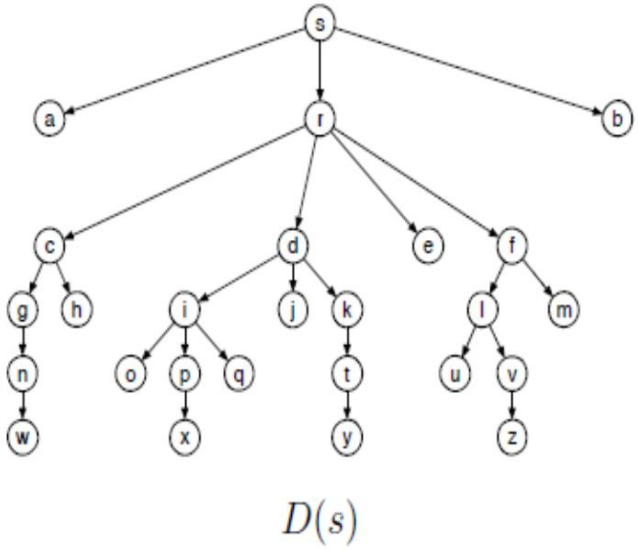
Algorithm 1 for computing 2-vertex connectivity




$G$

**Figure 3:** An input graph

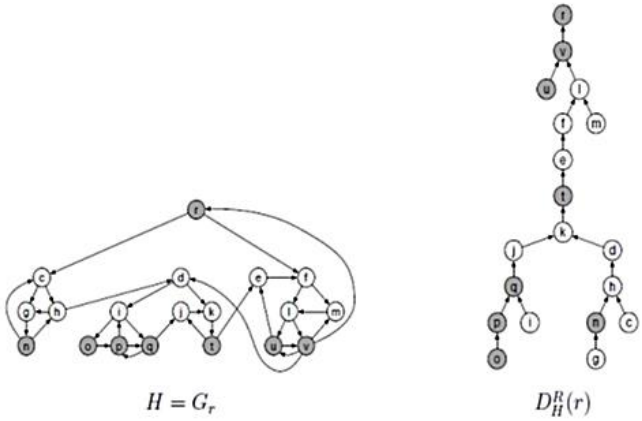
As shown in Figure 3, it is evident that he input graph is a directed graph. It shows various edges and vertices. By giving this graph as input, the algorithm 1, tries to have recursive approach to generated 2-vertex connectivity graphs with different variants.




$D(s)$

**Figure 4:**the dominator tree D(s) of flow graph G(s)

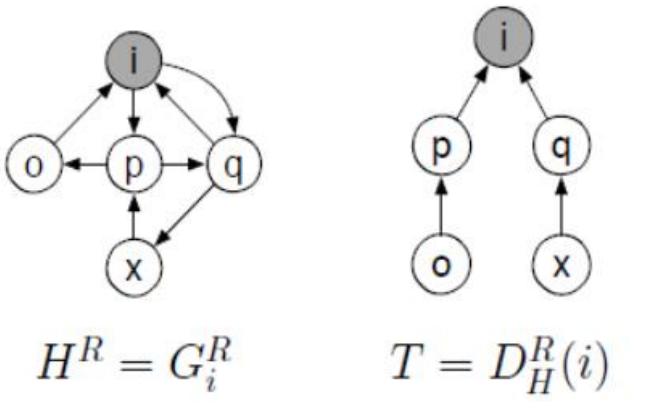
The result of two recursive calls generated this kind of graph. It has many vertices and edges with the features pertaining to 2-vertex connectivity.



$H = G_r$ $\quad$ $D_H^R(r)$

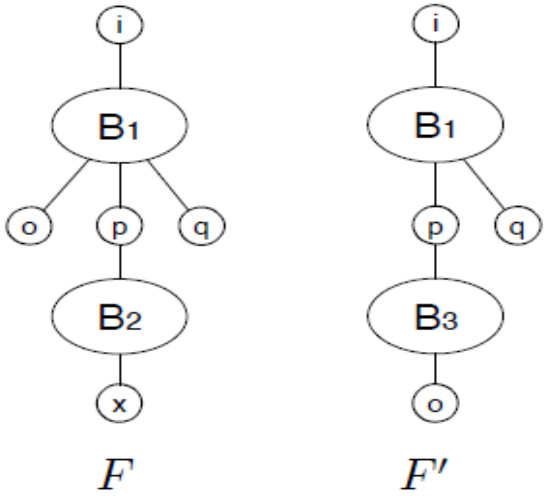
**Figure 5:** Result of first two recursive class when input is given from digraph. The auxiliary graph H = $G_r$ and the dominator tree $D^R_H(r)$ of the flow graph $H^R(r)$.

As shown in Figure 5, it is evident that the recursive calls generated another directed graph that shows 2-vertex connectivity.




$H^R = G_i^R$ $\quad$ $T = D_H^R(i)$

**Figure 6**: The reverse auxiliary graph $H^R = G^R_i$of the flow graph G(s). Result of the Algorithm 1 evident that the recursive call at Step 3.




$F$ $\quad$ $F'$

**Figure 7 :** F and F' are, respectively, the block forest before andafter the execution of split(B, T). Only the affected portion of the block forest is shown.

**Lemma 1**.Algorithm 1 runs in O(m) time.

Proof. We account for the total time spent on each step that Algorithm 1 executes.Step 1 takes O(m) time by [3], and Step 2 takes O(m) time.We have that the total number of vertices and the total number of edges in all auxiliarygraphs H of G are O(n) and O(m) respectively. The totalsize (number of vertices and edges) of all auxiliary graphs $H^R_q$for all H, computed inStep 3.4, is still O(m) and they are also computed in O(m) total time. Now consider the split operations. All these operations that occur during Step 3.3 for a specific auxiliary graph Gr operate on the same tree T, which can be preprocessed once,for all split operations. Therefore, the total preprocessing time for all split operations is O(n). Excluding the preprocessing time for T, a split(B, T) operationtakes time proportional to the number of vertices in B. Therefore all split operations takeO(n) time in total. In Step 3.5.1 we examine the adjacency listsof the ordinary vertices v $\in H^R_q$and find the corresponding blocks that contain at leastsuch two ordinary vertices. Then we examine the adjacency lists of each such block. So,the adjacency lists of each vertex v and each block that contains v can be examined atmost three times. Hence, Step 3.5.1 takes O(n) time in total. Finally, Steps 3.5.2 and3.5.3 take O(m) time in total by [16].

**Theorem.** Let G be a digraph with n vertices and m edges. We can compute the2-vertex-connected blocks of G in O(m + n) time and store them in a data structure ofO(n) space. Given this data structure, we can test in

O(1) time if any two vertices are2-vertex-connected. Moreover, if the two vertices are not 2-vertex-connected, then we canreport in O(1) time a strong articulation point or a strong bridge that separates them.

## IV.CONCLUSIONS AND FUTURE WORK

In this paper we studied graph theory that is widely used to solve many real world problems. When the complexity of any problem is more, it can be understood and solved easily using graph theory. This theory supports both directed and undirected graphs. Common notation used to represent a graph is G = (V, E) where V is a set of vertices and E is a set of edges. The dynamics of graphs were understood and explored well in the literature while there is insufficient research related to directed graphs. There are many problems to be solved with respect to graphs. For instance, 2-vertex connectivity is an important problem to be addressed. Recently Georgiadis et al. focused on this problem and found that two vertices v and w are 2-vertex connected when there are two vertex-disjoint paths coming from v to w and two vertex- disjoint paths from w to v. In this paper we explored 2-vertex connectivity with algorithmthat can support different variations in directed graphs. We presented the resultant graphs in the paper. In future we focus on more algorithms to have diversified graphs that support 2-vertex connectivity with respect to directed graphs.

## V. REFERENCES


[1]. NilakanthaPaudel, LoukasGeorgiadis and Giuseppe F. Italiano. (2017). Computing Critical Nodes in Directed Graphs. SIAM. P43-57.

[2]. Mohsen Ghaffari and Hsin-HaoSu .Distributed Degree Splitting, Edge Coloring, and Orientations.p2505-2523.

[3]. Martin Merker. (2017). Decomposing highly edge-connected graphs into homomorphic copies of a fixed tree. ELSEVIER.122 , P91–108.

[4]. louisesperet, remi de joannis de verclos, tien-nam le, ' and stephanthomass ' e. (2017). additive bases and flows in graphs, p1-11.

[5]. Samuel Fiorini Martin GroßJochenKonemann and Laura Sanita .(2017). A 3 2 - Approximation Algorithm for Tree Augmentation via Chvatal-Gomory Cuts, p1-21.

[6]. Monika Henzinger , Andrea Lincoln , Stefan Neumann , and Virginia Vassilevska Williams . (2017). Conditional Hardness for Sensitivity Problems, p1-33.

[7]. JinghuaHe ,Erling Wei , Dong Ye and ShaohuiZhai. (2017). On Perfect Matchings in Matching Covered Graphs, p1-10.

[8]. Gregory Gutin , M. S. Ramanujan , Felix Reidl and Magnus Wahlstrom. (2017). Path-contractions, edge deletions and connectivity preservation, p1-10.

[9]. LoukasGeorgiadis,ThomasDueholmHansen,Giuseppe F. Italiano,SebastianKrinninger and Nikos Parotsidis. (2017). Decremental Data Structures for Connectivity and Dominators in Directed Graphs, p1-39.

[10]. SurenderBaswanaAyushGoel and Shahbaz Khan. (2017). Incremental DFS algorithms: a theoretical and experimental study, p1-31.

[11]. Gaborhorvath, chrystopher l. nehaniv, and karolypodoski. (2017). the maximal subgroups and the complexity of the flow semigroup of finite (di)graphs, p1-23.

[12]. WeijuanZhang ,JianguoQian and Fuji Zhang. (2017). Flip-distance between α-orientations of graphs embedded on plane and sphere, p1-15

[13]. Dan Archdeacon,MattDeVos,StefanHannie and BojanMohar. (2017). Whitney's Theorem for 2-Regular Planar Digraphs, p1-8.

[14]. Jacob Holm, Giuseppe F. Italiano , Adam Karczmarz , JakubŁącki, Eva Rotenberg , and PiotrSankowski. (2017). Contracting a Planar Graph Efficiently, p1-21.

[15]. LoukasGeorgiadis, Giuseppe F. Italiano, Luigi Laura,NikosParotsidis. (2017). 2-Edge Connectivity in Directed Graphs. SIAM, p1988-2005.

[16]. R. E. Tarjan. Depth-_rst search and linear graph algorithms. SIAM Journal on Computing, 1(2):146_160, 1972.

[17]. L. Georgiadis, G. F. Italiano, L. Laura, and N. Parotsidis.2-vertex connectivity in directed graphs.In Proceedings of the 42th International Colloquium on automata, Languages, and Programming, 2015.