International Journal of Maps in Mathematics

*Volume **8**, Issue **2**, 2025, Pages:**791-815***

E-ISSN: 2636-7467

www.simadp.com/journalmim

THE EXISTENCE AND IDENTIFICATION OF STRONGLY CONNECTED EDGE DIRECTION ASSIGNMENTS IN BRIDGELESS GRAPHS AND MULTIGRAPHS

ANDREW CHARLES RADICKER, LIANYONG XUE, AND TÜRKAY YOLCU *

Abstract. The aim of this paper is two-fold. First, we will provide clarity on a result concerning the strong connectivity, a concept whose usefulness is readily apparent in several fields of study including social networking and transport networks, of a bridgeless connected graph achieved through the depth-first search (DFS) technique. To this end, we will demonstrate two rigorous mathematical proofs of this robust and well-known result. One proof takes the approach of seeking a contradiction by investigating the relationship between directed paths and maximal strongly connected subgraphs after the application of DFS. The other proof features a direct approach that demonstrates that for each tree edge $\{U, V\}$, there is a directed path from V to U by utilizing the fact that each edge in a connected multigraph on at least two vertices is either a bridge or is included in some cycle. Second, for a multigraph without a bridge, we provide two different proofs ensuring the existence of an assignment of edge directions that induces strong connectivity. One of these proofs utilizes the previous fact, whereas the second proof is independent of it and features a technique that focuses on collapsing entire connected multigraphs into a single vertex.

Keywords: Graph theory, Bridgeless multigraphs, Depth-first search, Strong connectivity.

2020 Mathematics Subject Classification: 68R10, 94C15, 05C40, 05C20, 05C85.

Received: 2025.03.03 *Accepted: 2025.08.29*

* Corresponding author

Andrew Charles Radicker ⋄ Andrew.Radicker@uky.edu ⋄ https://orcid.org/0009-0006-8059-9448

Lianyong Xue ⋄ lxue@bradley.edu ⋄ https://orcid.org/0000-0001-5896-9234

Türkay Yolcu ⋄ tyolcu@bradley.edu ⋄ https://orcid.org/0009-0002-5435-9417.

1. Introduction

Graph theory has rich history dating back to 1736, when Leonhard Euler published a paper using this versatile branch of mathematics to approach a problem concerning the seven bridges of Königsberg [8, 17]. Today, graph theory has garnered interest and found applications in other branches of mathematics as well as in several disciplines within academia. Specifically, some applications of graph theory include additive number theory [1], cryptography [28], molecular topology [5], Alzheimer's Disease [12], algebra [29], the study of DNA and biological networks [27], spectroscopy and quantum chemistry [7], chemistry [6, 16], social media and social networking [10, 24, 2, 4], blockchain technologies [22], social trust models [31], maze solving [18, 23, 25] and GPS networks [19]. In particular, we would like to examine the applications of graph theory in computer science as well as the inherent mathematical beauty therein.

In the realm of graph theory and the computational sciences, various algorithms, such as Depth-First Search (DFS), Breadth-First Search, Dijkstra's Algorithm, and Floyd-Warshall's Algorithm [11, 17, 15] play important roles in understanding graph structures. Each of these algorithms have a variety of applications. For more information pertaining to some applications of these algorithms, see [11, 21] and the references therein. Particularly, the Depth-First Search Algorithm serves as a means of graph traversal for the sake of identifying vertices and their relationships to underlying structures embedded within a given graph and an associated directed graph. This algorithm commences its journey from the *root*, an arbitrarily selected vertex from the given graph, and thoroughly explores each and every vertex as far as possible before traversing its moves backward.

Ever since a version of DFS was introduced as a means of solving mazes [18], it has been widely regarded as a versatile tool for approaching problems in both theory and practice pertaining to, for example, finding strongly connected components in a directed graph [11] and topological sorting [11]. With regard to applications in the computational sciences, DFS sees use in various areas including, but not limited to, image recognition [3] and computing search trees [30]. For further information concerning the implementation and execution of DFS in the computational sciences and the associated data structures, see [18]. Before we continue, we will revisit some graph theoretic definitions that will assist us in this article. Additionally, we will provide some motivating examples for some of the concepts introduced.

In this paper, a graph is a simple graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} \neq \emptyset$ is a nonempty set of vertices and $\mathcal{E}$ is a set of edges. A *multigraph* is a graph that can have *parallel edges* as well as *loops*, neither of which are possessed by (simple) graphs. Two vertices U and V are said to be *adjacent* if there is an edge between them. The *degree* of a vertex $V \in \mathcal{V}$ is the number of vertices to which V is adjacent and is denoted $\deg(V)$. Sometimes, we wish to assign directions to the edges of a graph. To see an example of this, let us recall the *Collatz Conjecture*. That is, let us define the function $C : \mathbb{N} \to \mathbb{N}$ given by

$$C(n) = \begin{cases} 3n + 1, \text{ if } n \text{ is odd} \\ n/2, \text{ if } n \text{ is even.} \end{cases}$$

The Collatz Conjecture states that for any $n \in \mathbb{N}$, repeated applications of C will eventually result in 1. For example, let $n = 6$. Observe that

$$C^8(6) = C^7(3) = C^6(10) = C^5(5) = C^4(16) = C^3(8) = C^2(4) = C(2) = 1.$$

Now, observe that we can represent this repeated application of C as a graph whose edges represent the notion that adjacent numbers have the property that one of the numbers is the result of applying C to the other. To indicate which number is obtained from applying C to another number, we can use arrows as is done in Figure 1

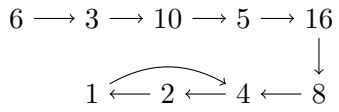

FIGURE 1. A graphical visualization of applying C to $n = 6$.

Above is an example of a directed graph. A *directed graph* (or *digraph*) is a graph whose edges are each assigned a direction. A graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ is called a *subgraph* of another graph $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ if $\mathcal{V}_1 \subseteq \mathcal{V}_2$ and $\mathcal{E}_1 \subseteq \mathcal{E}_2$. If this is the case, then we can write $\mathcal{G}_1 \subseteq \mathcal{G}_2$. If we have a graph $\mathcal{G}$ along with a nonempty set $X \subseteq \mathcal{V}(\mathcal{G})$, then the subgraph of $\mathcal{G}$ *induced* by X is the graph with vertex set X and edge set consisting of all edges $\{U, V\}$ with both U and V elements of X.

Let us again consider the graph depicted in Figure 1. Suppose that each number denotes a particular building in a city. Suppose further that if two buildings are joined by an edge, then there exists a one-way, road, whose direction is dictated by the direction of the edge, connecting these buildings. For example, one could travel from building 10 to building 4

through the roads creating the sequence $10 \to 5 \to 16 \to 8 \to 4$. However, in the way that these roads are directed, one would be unable to travel from building 4 to building 10. Let us make the graph-theoretic phenomenon in this example more precise.

The *connectivity* of an undirected graph depends on its vertices' capacity to reach one another along the edges of the graph. A *walk* is a sequence of adjacent vertices and can be thought of as moving between the vertices of a graph along the edges. A *closed walk* is a walk that begins and ends on the same vertex. A *path* is a walk that does not repeat vertices. A *cycle* is a closed path of the form $\{V_1, \ldots, V_k, V_{k+1} = V_1\}$. A *chord* is an edge that is not contained within a cycle, but joins two vertices within said cycle. It is said that two vertices are *connected* if there exists a path between them. It is said that a graph $\mathcal{G}$ is *connected* if for any two vertices $u, v \in \mathcal{V}(\mathcal{G})$, u and v are connected. Furthermore, a graph is called *disconnected* if it is not connected. For a connected graph $\mathcal{G}$, an edge $e \in \mathcal{E}(\mathcal{G})$ is called a *bridge* if the graph obtained by removing the edge e from the graph G is disconnected. On the other hand, the concept of strong connectivity is applicable solely to directed graphs. In the context of a directed graph, a graph $\mathcal{G}$ is *strongly connected* if and only if there is a directed path from a vertex V to a vertex U, as well as a directed path from U to V, for each pair of vertices U and V in $\mathcal{V}(\mathcal{G})$. The study of strongly connected graphs has various applications as well as additional routes for further inquiry in, for example, disciplines concerned with the reduction of complexity in certain problems [9, 11, 13]. A *strongly connected component* of a directed graph is a subgraph that is maximal with respect to the property of being strongly connected [14]. For an example of the study of strongly connected graphs and how it pertains to social networking, see [14]. For an example of how the study of strongly connected graphs can be considered in the study of public transport networks as well as their efficiency, see [26].

Recalling the Collatz Conjecture, let us consider a directed graph whose vertices consist of all elements of $\mathbb{N}$ and whose edges are constructed in the following way. Suppose $m, n \in \mathbb{N}$ are such that $C(m) = n$. Then we construct a directed edge from m to n. Let us refer to this graph as the *Collatz Graph*. Then observe that disproving the Collatz Conjecture could be simplified to locating a directed cycle in the Collatz Graph other than the cycle $\{4, 2, 1, 4\}$. Using our new terminology, we can again consider the graph in Figure 1 as a network of buildings in a city. Recall that there exists a path from 10 to 4 but that there does not exist a path from 4 to 10. From this, we can conclude that this graph is not strongly connected. Why is this? In fact, there is a direct relationship between the existence of bridges in a graph

and the existence of an edge direction assignment that makes the resulting directed graph strongly connected. Observe that in the graph in Figure 1, every edge is a bridge other than the edges of the cycle $\{4, 2, 1, 4\}$. However, if we simply add the directed edge $\{1, 6\}$, we obtain the following graph.

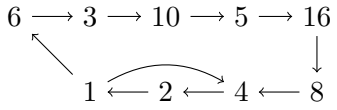

FIGURE 2. The graph in Figure 1 along with the directed edge $\{1, 6\}$.

Notice that in this graph, for each pair of vertices, there exists a directed path from one to the other. As such, we can conclude that this graph is strongly connected. Observe further that there does not exist a bridge in this graph. That is, the removal of any edge will not leave the resulting graph disconnected. It may, however, leave the resulting directed graph no longer strongly connected. In this paper, we will further explore the connection between the lack of bridges in a connected graph, and that graph's potential to have a strongly connected edge direction assignment. Thus, notice that if we think of the graph in Figure 2 as a transportation network with the directed edges representing one-way roads and the vertices representing buildings, one could travel between any two buildings. Let us wrap up our discussion of graph-theoretic terminology with a brief discussion about trees.

A *tree* is a connected graph containing no cycles. It can be observed that the number of vertices in a tree is one more than the number of edges. The converse is not necessarily true. For $j = i, \ldots, k-1$, V_j is called the parent of V_{j+1} in the DFS tree, and the edge (V_k, V_i) is called a *back-edge*. See [17, 20] for the properties of trees and rooted trees. Observe that every edge of a tree is a bridge. As an example, referring back to Figure 1, if we think of the cycle $\{4, 2, 1, 4\}$ as a single vertex $\boxed{4}$. Then the resulting graph would be as follows.

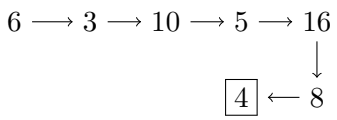

FIGURE 3. A graphical visualization of collapsing the cycle $\{4, 2, 1, 4\}$ into a single vertex $\boxed{4}$.

Observe that this graph is a tree, and that, indeed, each edge of this graph is a bridge and that this graph is not strongly connected. We will utilize this concept of collapsing subgraphs into a single vertex in a later proof. For this, see our second proof of Theorem 6.1.

This paper is organized in the following manner. In Section 2, we provide a motivating example demonstrating the usefulness of strong connectivity in network survivability and how it relates to trust networks. In Section 3, after recalling the Depth-First Search Algorithm, we discuss Theorem 3.1, concerning a technique for constructing a strongly connected edge direction assignment in a connected bridgeless graph, by presenting an example demonstrating its applicability. In Section 4, after presenting important facts concerning strong connectivity and cycles, we provide a proof of Theorem 3.1 by contradiction. In Section 5, we provide a second proof of Theorem 3.1 after proving Lemma 5.1, which states that each edge in a connected multigraph on at least two vertices is either a bridge or is included in a cycle. In Section 6, we provide two proofs of Theorem 6.1, which states that in a connected, bridgeless multigraph, there exists an edge direction assignment that makes the resulting directed graph strongly connected. One of these proofs invokes Lemma 5.1 and the other does not. Finally, we present concluding remarks in Section 7.

2. A Motivating Example of Strong Connectivity

The study of graph and multigraph connectivity can be used in a variety of real-world applications including the study of *network survivability* [20]. Let $\mathcal{G}$ be a simple, undirected, connected graph. Let $\kappa_v(\mathcal{G})$ denote the *vertex connectivity* of $\mathcal{G}$, or the smallest number of vertices whose removal from $\mathcal{G}$ can disconnect $\mathcal{G}$ or turn it into the trivial graph on a single vertex [20]. Similarly, let $\kappa_e(\mathcal{G})$ denote the *edge connectivity* of $\mathcal{G}$, or the smallest number of edges whose removal from $\mathcal{G}$ can disconnect $\mathcal{G}$ [20]. Both $\kappa_v(\mathcal{G})$ and $\kappa_e(\mathcal{G})$ are used to assess the network survivability of a network, or "the capacity of a network to retain connections among its nodes after some edges or nodes are removed" [20]. To further explore the applications of graph and multigraph connectivity, we can consider the concept of a *fault-tolerant* communications network, which is a communications network that "has at least two alternative paths between each pair of vertices" [20]. For additional details regarding network survivability and fault-tolerant communications networks, see Chapter 5 of [20].

Let us consider the following motivating example for the sake of demonstrating the applicability of network survivability.

Example 2.1. *Suppose we have a simple, undirected, connected graph $\mathcal{G}$ whose vertices represent people and whose edges represent a friendship between two people. It would be reasonable to infer, that if two vertices are connected, then the two corresponding people can have information transferred between them. Suppose $\mathcal{G}$ is the graph represented in Figure 4.*

$$A \text{ — } B \text{ — } C \text{ — } D$$

FIGURE 4. A group of people and their corresponding friendships as a graph $\mathcal{G}$.

Observe that this is a quite fragile network, as removing B or C from the group would preclude A and D from sharing information. That is, $\kappa_v(\mathcal{G}) = 1$. Similarly, if any two people choose to end their friendship with one another, then the graph will become disconnected, and thus, there will be at least two people who cannot share information. As such, $\kappa_e(\mathcal{G}) = 1$. However, if the people in this group acknowledge this fact about their network survivability, they can act to strengthen their friend group's ability to share information by suggesting that the people represented by vertices A and D form a friendship, as shown in Figure 5.

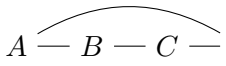

FIGURE 5. A stronger friendship network than in Figure 4.

In this graph, which we will call $\mathcal{G}'$, there is no vertex, nor edge, that can be removed to cause the resulting graph to be disconnected. Removing two vertices or edges, however, will cause the graph to be disconnected. As such, $\kappa_v(\mathcal{G}') = \kappa_e(\mathcal{G}') = 2$. We can further observe that the network represented by the graph in Figure 5 is fault-tolerant. Since there is no single edge in $\mathcal{G}'$ that disconnects the graph upon removal, we acknowledge that $\mathcal{G}'$ (and the friendship network represented therein) has stronger network survivability than $\mathcal{G}$ since no bridge exists in $\mathcal{G}'$. In fact, we observe that for a simple, undirected, connected graph $\mathcal{G}$, if $\kappa_e(\mathcal{G}) > 1$, then $\mathcal{G}$ does not contain a bridge.

Continuing with our supposition that the vertices of a graph $\mathcal{G}$ represent people and edges of $\mathcal{G}$ represent a friendship between two people, let us further assume that the friend group represented by $\mathcal{G}$ has a method of communication dependent on the trust one person has in another. That is, suppose that for any two vertices $v_1, v_2 \in \mathcal{V}(\mathcal{G})$, either the person represented by v_1 can receive information given by the person represented by v_2 or they can give information to the person represented by v_2. In this case, the graph becomes a

directed graph. Now, the friend group may be inclined to ponder whether or not they can share information throughout the network if they were to impose the trust-dependent structure described above. Asking this question would be equivalent to determining whether the graph representing the network could be made strongly connected. If $\kappa_e(\mathcal{G}) > 1$, then the answer to this question is "yes."

3. DFS and its Relation to Strong Connectivity

One purpose of this paper is to provide two different mathematically rigorous proofs of Theorem 3.1, a well-known result, which appears in [17]. Although the result is fairly intuitive, the demonstration of this fact is quite intricate. Moving forward, we will include several figures to supplement the reader's understanding of the theorems, proofs, and applications discussed.

Before we begin, let us recall the Depth-First Search Algorithm which inductively operates on a graph with n vertices in the following manner.

(1) Start by picking any vertex from the graph. Label that vertex as V_1.
(2) Visit an adjacent vertex of the vertex labeled V_1 and label it V_2.
(3) Visit an adjacent unlabeled vertex of the vertex labeled V_2 and label it V_3.
(4) Continue this process until vertices have been labeled V_1, V_2, ..., V_r and the vertex labeled V_r is not adjacent to an unlabeled vertex for $1 \leq r \leq n$.
(5) If $r < n$, select the largest i, $1 \leq i \leq r$, such that V_i is adjacent to an unlabeled vertex. Assign the label V_{r+1} to that vertex and return to step (4). Otherwise, if $r = n$, we are done.

Now, we can state the theorem, as referenced in [17], Theorem 5.8 on page 259.

Theorem 3.1. [17] *Suppose we apply depth-first search to a connected, bridgeless graph. If we assign directions to tree edges from lower depth-first search label to higher and to back edges from higher label to the lower, then the resulting directed graph is strongly connected.*

In order to comprehend this result, let us commence by examining an illustrative example that highlights the potency of Theorem 3.1.

Example 3.1. *Let us consider the connected graph $\mathcal{G}$ depicted in Figure 6, which does not possess a bridge.*

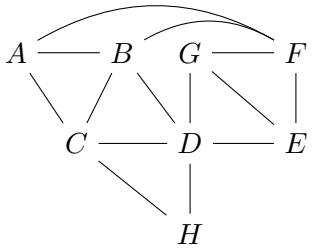

FIGURE 6. A connected graph $\mathcal{G}$ without a bridge.

Let us demonstrate the flexibility and versatility of DFS by exemplifying three distinct applications and the resulting strongly connected graphs. In any application of DFS, the designation of the root is arbitrary. As such, we could begin the algorithm on vertex A, B, C, D, or any other vertex. For the sake of simplicity and readability, we will start with A as our root in the following applications of DFS. Let us again consider the friendship and trust network as in Example 2.1. We see that through this application of DFS, we can determine a trust structure that would allow for strong connectivity, or in the example, a flow of information between all members of the friend group.

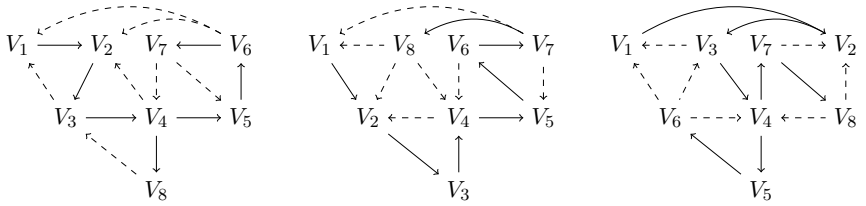

FIGURE 7. Three applications of DFS to $\mathcal{G}$ in Figure 6 with dashed back-edges.

Having applied DFS to $\mathcal{G}$, we can display the resulting strongly connected graphs using the directions assigned in Figure 7.

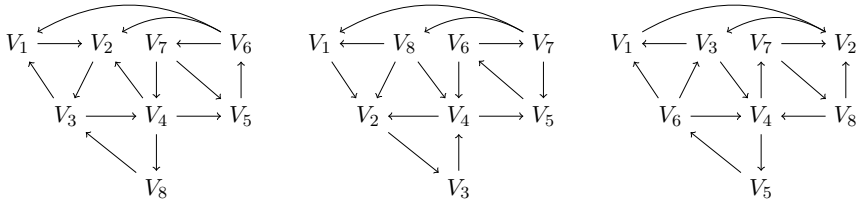

FIGURE 8. Three distinct strongly connected directed representations of $\mathcal{G}$ as in Figure 6.

We encourage the reader to ensure the strong connectivity of each graph presented in Figure 8 by confirming the existence of closed walks containing all of the vertices of $\mathcal{G}$ therein.

4. First Proof: A Contradiction

Prior to commencing our proof, we shall first revisit certain preliminary and relatively-easy-to-verify facts about strong connectivity of finite directed graphs. These simple facts are crucial for understanding the proofs we present in this article; therefore, we provide statements and straightforward examples for each fact for the sake of completeness. We encourage the reader to verify these facts for themself.

(I) A directed cycle is strongly connected.

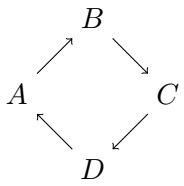

Figure 9. An illustration of Fact (I).

(II) A graph consisting solely of two directed cycles that share a common vertex (an "8" or "∞" shape) is strongly connected.

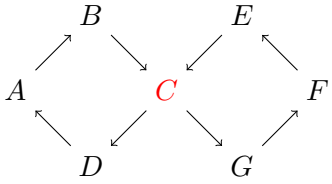

Figure 10. An illustration of Fact (II) with common vertex C.

(III) A graph consisting solely of two directed cycles that share a common directed edge is strongly connected.

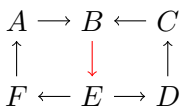

Figure 11. An illustration of Fact (III) with common directed edge $\{B, E\}$.

(IV) A graph containing two strongly connected directed subgraphs with some common vertex, or some common directed edge, is strongly connected.

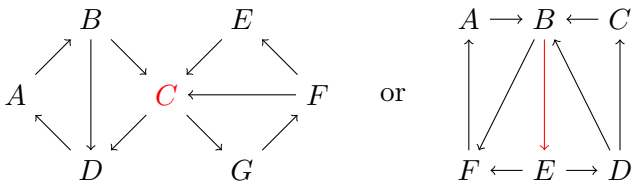

FIGURE 12. Two illustrations of Fact (IV).

Now that we have equipped ourselves with the appropriate mathematical instruments, let us discuss our first proof of Theorem 3.1.

Proof of Theorem 3.1. Let $V_1, V_2, \ldots, V_n$ be the labels of the depth-first search tree $\mathcal{T}$ where i designates the label number of vertex V_i.

Now, we will follow these steps to execute the proof.

(1) $\{V_1, V_2\}$ is a tree edge, i.e., a directed path from V_1 to V_2.

(2) Let $\{V_1, V_2, \ldots, V_{n_1-1}, V_{n_1}\}$ be a longest directed path with consecutive labels starting from V_1 in $\mathcal{T}$ as in Figure 13.

$$V_1 \longrightarrow V_2 \longrightarrow \cdots \longrightarrow V_{n_1-1} \longrightarrow V_{n_1}$$

FIGURE 13. A longest directed path in a depth-first search.

Then, we first make the following observations:

(a) $\deg(V_{n_1}) = 1$ in $\mathcal{T}$. Indeed, if $\deg(V_{n_1}) > 1$ in $\mathcal{T}$, then

$$\{V_1, V_2, \ldots, V_{n_1-1}, V_{n_1}\}$$

would not be a longest directed path in a depth-first search. Consequently, V_{n_1} is not adjacent to any vertex in $\{V_{n_1+1}, V_{n_1+2}, \ldots, V_{n-1}, V_n\}$ according to the depth-first search.

(b) $\deg(V_{n_1}) > 1$ in $\mathcal{G}$. This must be the case because otherwise, $\{V_{n_1-1}, V_{n_1}\}$ would be a bridge.

(c) Hence, by (a) and (b), there is a vertex V_{b_1} in $\{V_1, V_2, \ldots, V_{n_1-1}\}$ such that $\{V_{n_1}, V_{b_1}\}$ is a back edge, and so $\{V_{b_1}, V_{b_1+1}, \ldots, V_{n_1-1}, V_{n_1}\}$ is a directed cycle as in Figure 14. Therefore, the directed cycle

$$\{V_{b_1}, V_{b_1+1}, \ldots, V_{n_1-1}, V_{n_1}\}$$

is a strongly connected subgraph of $\mathcal{G}$.

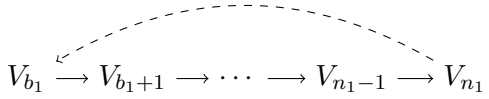

FIGURE 14. A directed cycle of $\mathcal{G}$.

(d) Let s_1 be the smallest number less than or equal to b_1 and m_1 be the maximum number greater than or equal to n_1 such that
   (i) $\{V_1, V_2, \ldots, V_{s_1-1}, V_{s_1}\}$ is a directed path in $\mathcal{T}$ and
   (ii) $\mathcal{G}_1 = \{V_{s_1}, V_{s_1+1}, \ldots, V_{m_1-1}, V_{m_1}\}$ is a strongly connected subgraph of $\mathcal{G}$.

$$V_1 \longrightarrow V_2 \longrightarrow \cdots \longrightarrow V_{s_1-1} \longrightarrow \{V_{s_1}, V_{s_1+1}, \ldots, V_{m_1-1}, V_{m_1}\} = \mathcal{G}_1$$

FIGURE 15. A directed path to a maximal strongly connected subgraph $\mathcal{G}_1$ of $\mathcal{G}$.

(3) We shall prove that $s_1 = 1$ and $m_1 = n$, and so the graph induced by the vertices $\{V_1, V_2, \ldots, V_n\}$ is strongly connected.

Suppose that $m_1 < n$. Then there is an integer, denoted as i_1, which represents the largest label less than or equal to m_1, such that $\{V_{i_1}, V_{m_1+1}\}$ is a tree edge. By the depth-first search, there is no vertex in $\{V_{i_1+1}, V_{i_1+2}, \ldots, V_{m_1}\}$ adjacent to any vertex in $\{V_{m_1+1}, V_{m_1+2}, \ldots, V_n\}$.

First, we will demonstrate that $i_1 \geq s_1$. That is to say, $V_{i_1} \in \mathcal{G}_1$. In fact, if $i_1 < s_1$, then no vertex in $\mathcal{G}_1$ is adjacent to any vertex in $\{V_{m_1+1}, V_{m_1+2}, \ldots, V_n\}$. Hence there is a vertex V_m in $\mathcal{G}_1$ which is adjacent to a vertex V_s in $\{V_1, V_2, \ldots V_{s_1-2}\}$. This must be the case because otherwise, $\{V_{s_1-1}, V_{s_1}\}$ would be a bridge. Using this back edge $\{V_m, V_s\}$, we have a directed path $\{V_m, V_s, V_{s+1}, \ldots, V_{s_1-1}, V_{s_1}\}$.

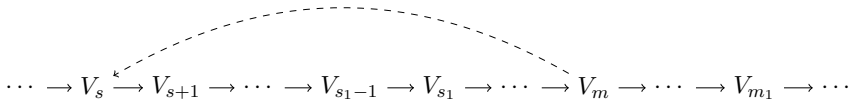

FIGURE 16. A larger directed cycle of $\mathcal{G}$ when $i_1 < s_1$.

Since $\mathcal{G}_1$, induced by $\{V_{s_1}, V_{s_1+1}, V_{s_1+2}, \ldots, V_{m_1}\}$, is a strongly connected subgraph of $\mathcal{G}$, there is a directed path from V_{s_1} to V_m, and so we have a directed cycle from V_m to V_s to V_{s_1} to V_m as shown in Figure 16. Therefore, if we had $i_1 < s_1$, then

the subgraph induced by $\{V_s, V_{s+1}, \ldots, V_{s_1-1}, V_{s_1}, V_{s_1+1}, \ldots, V_{m_1}\}$ would be strongly connected by Fact (IV). This contradicts the selection of the minimum value s_1 and the maximum value m_1 in (2d). Therefore, we know that $i_1 \geq s_1$. Let

$$\{V_{i_1}, V_{m_1+1}, V_{m_1+2}, \ldots, V_{n_2}\}$$

be a longest directed path beginning with the tree edge $\{V_{i_1}, V_{m_1+1}\}$ and with consecutive labels $m_1+1, m_1+2, \ldots, n_2$ in $\mathcal{T}$.

$$\begin{array}{ccccccccc} V_1 & \longrightarrow & V_2 & \longrightarrow & \cdots & \longrightarrow & V_{s_1-1} & \longrightarrow & \{V_{s_1}, \ldots, V_{i_1}, \ldots, V_{m_1}\} \\ & & & & & & & & \downarrow \\ & & & & & & & & V_{m_1+1} \\ & & & & & & & & \downarrow \\ V_{n_2} & \longleftarrow & \cdots & \longleftarrow & V_{m_1+2} & & & & \end{array}$$

FIGURE 17. A directed path from V_{i_1} to the vertex V_{n_2}.

Then

(a) $\deg(V_{n_2}) = 1$ in $\mathcal{T}$ by the longest property. Hence, V_{n_2} is not adjacent to any vertex in $\{V_{n_2+1}, V_{n_2+2}, \ldots, V_n\}$ by the depth-first search.

(b) $\deg(V_{n_2}) > 1$ in $\mathcal{G}$. This must be the case because otherwise, $\{V_{n_2-1}, V_{n_2}\}$ would be a bridge.

(c) Hence, there is a vertex V_{b_2} in $\{V_1, V_2, \ldots V_{n_2-1}\}$ such that $\{V_{n_2}, V_{b_2}\}$ is a back edge.

Now we consider 3 cases: $b_2 < s_1$, $s_1 \leq b_2 \leq m_1$, and $b_2 > m_1$.

Case 1: $b_2 < s_1$. If this is the case, then

$$\{V_{i_1}, V_{m_1+1}, V_{m_1+2}, \ldots V_{n_2}, V_{b_2}, V_{b_2+1}, \ldots, V_{s_1}\}$$

is a directed path. Since $\mathcal{G}_1$, the graph induced by

$$\{V_{s_1}, V_{s_1+1}, V_{s_1+2}, \ldots, V_{m_1}\}$$

is a strongly connected subgraph, there is a directed path from V_{s_1} to V_{i_1}, and so we have a directed cycle from V_{i_1} to V_{n_2} to V_{b_2} to V_{s_1} to V_{i_1}. Hence, the graph induced by

$$\{V_{b_2}, V_{b_2+1}, \ldots, V_{s_1}, \ldots, V_{i_i}, \ldots, V_{m_1}, V_{m_1+1}, V_{m_1+2}, \ldots, V_{n_2}\}$$

is a strongly connected subgraph of $\mathcal{G}$ by Fact (IV). This contradicts the selection of s_1 and m_1.

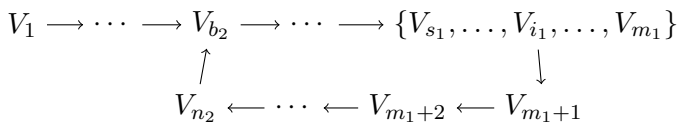

FIGURE 18. A directed path from V_{i_1} through V_{m_1+1}, V_{n_2}, V_{b_2} to V_{s_1}.

Case 2: $s_1 \leq b_2 \leq m_1$. If this is the case, then similar to Case 1, we will have a directed cycle from V_{i_1} to V_{n_2} to V_{b_2} to V_{i_1}. Hence, the graph induced by

$$\{V_{s_1}, V_{s_1+1}, V_{s_1+2}, \ldots, V_{m_1}, V_{m_1+1}, V_{m_1+2}, \ldots, V_{n_2}\}$$

is a strongly connected subgraph of $\mathcal{G}$ by Fact (IV). This contradicts the selection of s_1 and m_1.

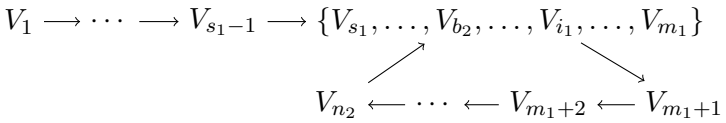

FIGURE 19. A directed cycle from V_{i_1} through V_{m_1+1}, V_{n_2}, V_{b_2} to V_{i_1}.

Case 3: $b_2 > m_1$. In this case, $\{V_{b_2}, V_{b_2+1}, \ldots, V_{n_2}, V_{b_2}\}$ becomes a directed cycle. Let s_2 be the smallest number less than or equal to b_2 and m_2 be the maximum number greater than or equal to n_2 such that $\mathcal{G}_2$, the graph induced by

$$\{V_{s_2}, V_{s_2+1}, \ldots, V_{b_2}, V_{b_2+1}, \ldots, V_{n_2}, \ldots, V_{m_2}\},$$

is a strongly connected subgraph with consecutive labels. Note that, by the selection of $\mathcal{G}_1$, $m_1 < s_2$. Also,

$$\{V_{i_1}, V_{m_1+1}, V_{m_1+2}, \ldots, V_{s_2}\}$$

is a directed path from $\mathcal{G}_1$ to $\mathcal{G}_2$. Moreover, there is no back edge from a vertex in $\mathcal{G}_2$ to any vertex with label less than s_2 by the "maximum" property. By repeating this finitely many, say r, times, we will have disjoint "maximal" strongly connected subgraphs $\mathcal{G}_j$, induced by

$$\{V_{s_j}, V_{s_j+1}, V_{s_j+2}, \ldots, V_{i_j}, \ldots, V_{m_j}\},$$

for $1 \leq j \leq r$, such that $\{V_1, V_2, \ldots, V_{s_1}\}$ is a directed path in the DFS tree $\mathcal{T}$, $\{V_{i_j}, V_{m_j+1}, V_{m_j+2}, \ldots, V_{s_{j+1}}\}$ is a directed path from $\mathcal{G}_j$ to $\mathcal{G}_{j+1}$, and $V_{m_r} = V_n$.

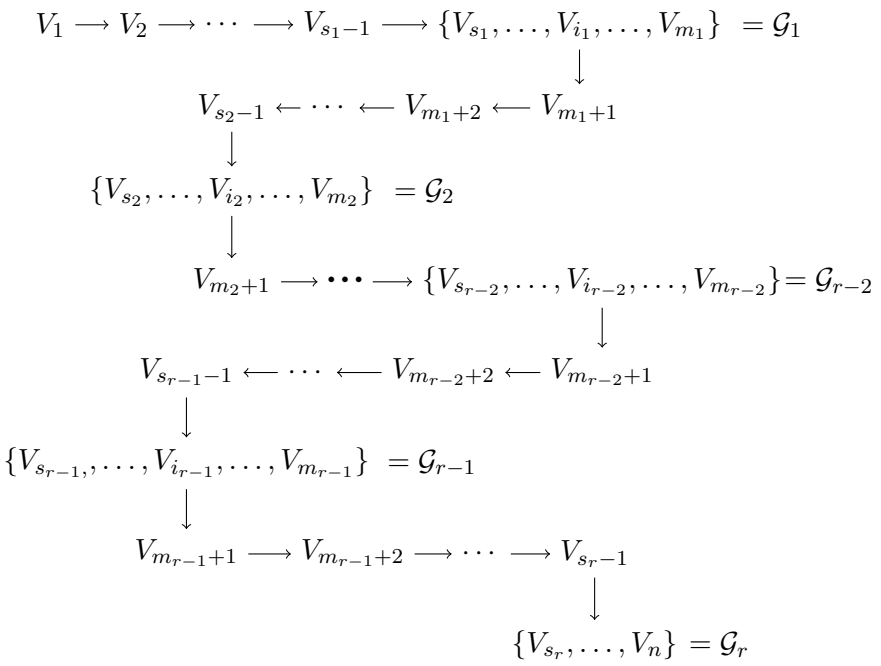

FIGURE 20. A structure of disjoint, maximal, and strongly connected subgraphs.

Now, if no vertex in $\{V_1, V_2, \ldots, V_{m_{r-1}}\}$ is adjacent to any vertex in $\mathcal{G}_r$, then the first edge $\{V_{i_{r-1}}, V_{m_{r-1}+1}\}$ in the directed path from $\mathcal{G}_{r-1}$ to $\mathcal{G}_r$ would be a bridge, which is a contradiction. If there is a vertex V_s in $\{V_1, V_2, \ldots, V_{m_{r-1}}\}$ which is adjacent to a vertex V_m in $\mathcal{G}_r$, then $\{V_s, V_m\}$ is a back edge, which would contradict the "maximal" property of $\mathcal{G}_i s$. Hence r must be equal to 1 and $m_1 = n$.

Furthermore, we will prove (by contradiction) that s_1 must be equal to 1. In fact, if $s_1 > 1$, then there is a vertex V_s in $\{V_1, V_2, \ldots, V_{s_1-1}\}$ and a vertex V_m in

$$\mathcal{G}_1 = \{V_{s_1}, V_{s_1+1}, \ldots, V_n\}$$

such that $\{V_m, V_s\}$ is a back edge. This must be the case because otherwise, $\{V_{s_1-1}, V_{s_1}\}$ would be a bridge.

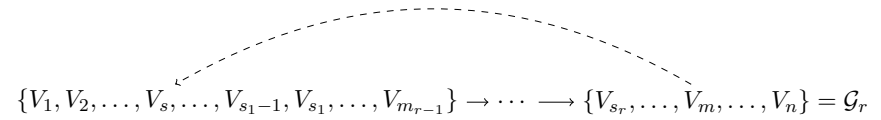

FIGURE 21. A critical back edge $\{V_m, V_s\}$.

Hence, the graph induced by

$$\{V_s, V_{s+1}, \ldots, V_{s_1}, V_{s_1+1}, \ldots, V_n\}$$

is a strongly connected subgraph of $\mathcal{G}$ by Fact (IV). This contradicts the selection of s_1. Therefore, s_1 must be equal to 1.

This completes the proof. □

5. Second Proof: A Direct Method

We will now focus our efforts on proving the following lemma, which will assist us in our second proof of Theorem 3.1.

Lemma 5.1. *Let $\mathcal{G}$ be a connected multigraph with $n > 1$ vertices. Then each edge in $\mathcal{G}$ is either a bridge or is included in a cycle.*

Proof. Let $\{U, V\}$ be an edge in $\mathcal{G}$. Now, we perform a DFS algorithm beginning with $U = V_1$ and $V = V_2$ and let $V_1, V_2, \ldots, V_n$ be the labels in the resulting tree $\mathcal{T}$. By assigning a direction for each edge in $\mathcal{T}$ from lower label to higher label, $\mathcal{T}$ becomes a rooted tree with vertex V_1 as its root and there is a (unique) directed path P_i from V_1 to any vertex V_i. Now, we divide all the V_is into two parts:

$$P_U = \{V_i \mid P_i \text{ does not include } V_2\} \quad \text{and} \quad P_V = \{V_j \mid P_j \text{ includes } V_2\}. \tag{5.1}$$

Then P_U and P_V are disjoint and $\{U, V\}$ is the only edge in $\mathcal{T}$ joining P_U and P_V. If there is no other edge in $\mathcal{G}$ connecting P_U and P_V, then $\{U, V\}$ is a bridge. If there is some edge $\{V_i, V_j\}$ in $\mathcal{G}$ where $V_i \in P_U$ and $V_j \in P_V$, then $\{U, V\}$ is included in the (undirected) cycle

$$\{V_i, \ldots, V_1 = U, V_2 = V, \ldots, V_j, V_i\}.$$

Thus, we can conclude that each edge in $\mathcal{G}$ is either a bridge or is included in a cycle. □

For an illustration of the sets P_U and P_V as used in Equation (5.1) within Lemma 5.1 as well as the cases contemplated in the Lemma, let us consider the following examples.

Example 5.1. (*a*) *Let $\mathcal{G}_1$ be the graph on the left depicted in Figure 22. To its right is an application of DFS to $\mathcal{G}_1$.*

FIGURE 22. An illustration of Lemma 5.1 with $\{U, V\}$ as a bridge.

In this case, we observe that $P_U = \{V_1, V_5, V_6\}$ *and* $P_V = \{V_2, V_3, V_4\}$ *and thus, we have that* $P_U \cap P_V = \emptyset$*. Furthermore, we observe that* $\{U, V\}$ *is the only edge of* $\mathcal{G}_1$ *connecting* P_U *and* P_V*. Hence,* $\{U, V\}$ *is a bridge of* $\mathcal{G}_1$*.*

(*b*) *Let* $\mathcal{G}_2$ *be the graph on the left depicted in Figure 23. To its right is an application of DFS.*

FIGURE 23. An illustration of Lemma 5.1 with $\{U, V\}$ as an edge of a cycle.

In this case, we observe that $P_U = \{V_1\}$ *and* $P_V = \{V_2, V_3, V_4, V_5, V_6\}$ *and thus, we have that* $P_U \cap P_V = \emptyset$*. In this case,* $\{U, V\}, \{U, C\}$*, and* $\{U, D\}$ *all join* P_U *with* P_V *in* $\mathcal{G}_2$*, and as such,* $\{U, V\}$ *is not the only edge connecting the sets. Moreover, we see that* $\{U, V\}$ *is not a bridge of* $\mathcal{G}_2$*, and thus, is included in a cycle of* $\mathcal{G}_2$*.*

We will now provide another proof of Theorem 3.1 using the machinery we have established in Lemma 5.1.

Proof of Theorem 3.1. Suppose depth-first search is applied to a connected graph $\mathcal{G}$ without a bridge. Let us assign a tree edge from lower to higher depth-first search number and assign a back edge from higher to lower depth-first search number.

We will prove that the resulting directed graph is strongly connected. To this end, let us assign $V_1, V_2, \ldots, V_n$ as the labels of the depth-first search tree $\mathcal{T}$ with i the label number of V_i. Then $\mathcal{T}$ is a rooted tree with V_1 as the root and there is a directed path from V_1 to any vertex V_i by DFS.

We shall prove that for any vertex V_i, there is a directed path from V_i to V_1. Doing so will ensure that $\mathcal{G}$ is strongly connected. To this end, it suffices to show that there is a directed path from V to U for each tree edge $\{U, V\}$.

Let $\{U, V\}$ be an arbitrary tree edge. Then, by Lemma 5.1, since $\mathcal{G}$ is bridgeless, the edge $\{U, V\}$ is included in some cycle $\mathcal{C}$. Let m be the lowest label among the vertices in $\mathcal{C}$ and $\mathcal{T}_{V_m}$ be the subtree of $\mathcal{T}$ formed by V_m and its descendants. To illustrate this, consider the graph on the left of Figure 24, and the designation of $\mathcal{T}_{V_m}$ after the application of DFS.

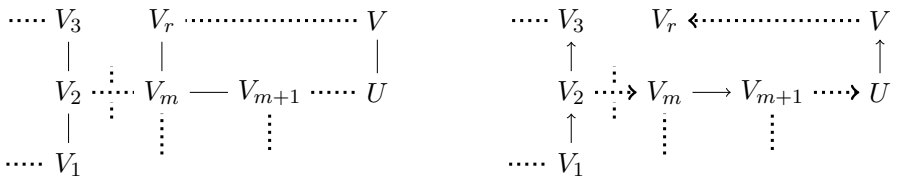

FIGURE 24. An example graph $\mathcal{G}$ and the corresponding $\mathcal{T}_{V_m}$ identified.

Then we have that V_m is the root of $\mathcal{T}_{V_m}$. If we denote the (undirected) cycle $\mathcal{C}$ by $\{U_1 = V_m, U_2, \ldots, U_s, U_1\}$ such that $U = U_i$ and $V = U_{i+1}$ for some i, then $\{U_1, U_2, \ldots, U_s\}$ is a directed path in $\mathcal{T}_{V_m}$ and (U_s, U_1) is a back edge. As such, $\{U_1, U_2, \ldots, U_s, U_1\}$ becomes a directed cycle. Moreover, (U, V) is included in the directed cycle $\mathcal{C} = \{U_1, U_2, \ldots, U_s, U_1\}$. Hence, there is a directed path from V to U. To further illustrate this, included in Figure 25 is the graph from Figure 24 with the cycle in question highlighted in red on the left as well as assigned directions such that there exists both a directed path from $U = U_i$ to $V = U_{i+1}$ and from $V = U_{i+1}$ to $U = U_i$ on the right.

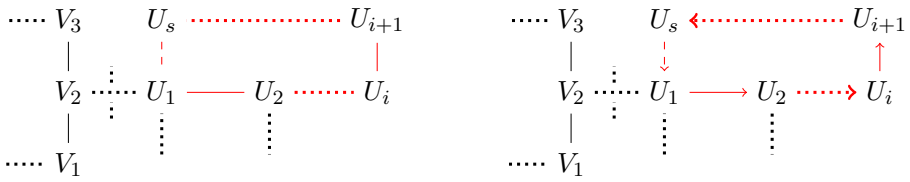

FIGURE 25. An undirected cycle $\mathcal{C}$ and directed paths from $U = U_i$ to $V = U_{i+1}$ and from $V = U_{i+1}$ to $U = U_i$.

Since $\{U, V\}$ was selected arbitrarily, this process guarantees that performing DFS in the way provided will ensure the existence of a directed path from V to U for every tree edge $\{U, V\}$. Thus, $\mathcal{G}$ becomes strongly connected. □

6. Strong Connectivity of a Bridgeless Multigraph

Let us consider and acknowledge the fact that given a bridgeless multigraph on at least two vertices, we can ensure the existence of a strongly connected graph as a result of some assignment of directions to the edges of said graph.

Theorem 6.1. *Let $\mathcal{G}$ be a connected multigraph with $n > 1$ vertices without a bridge. Then we can make $\mathcal{G}$ a strongly connected multigraph by assigning a direction to each edge in $\mathcal{G}$.*

We will prove this theorem in two ways. One of these ways will rely on Lemma 5.1, whereas the other will not.

Proof of Theorem 6.1 (Using Lemma 5.1). Pick up an edge $\{U, V\}$ in $\mathcal{G}$. Then by Lemma 5.1, $\{U, V\}$ is contained in some cycle C_1. We can assign a direction to each edge in the cycle C_1 to make it a directed cycle.

If C_1 contains all the vertices in $\mathcal{G}$, then $\mathcal{G}$ is strongly connected regardless of the directions of edges not in the cycle C_1.

If C_1 does not contain all the vertices in $\mathcal{G}$. Let $\mathcal{G}_1$ be the subgraph induced by C_1. Then there is an edge $\{V_1, V_2\}$ such that V_1 is in $\mathcal{G}_1$ and V_2 is not in $\mathcal{G}_1$. By Lemma 5.1, $\{V_1, V_2\}$ is contained in some cycle C_2. This leads us to contemplate the two following cases.

(a) If V_1 is the only common vertex of $\mathcal{G}_1$ and C_2, then we can assign a direction to each edge in the cycle C_2 to make it a directed cycle, and so the subgraph $\mathcal{G}_2$ induced by $\mathcal{G}_1 \cup C_2$ is strongly connected regardless the directions of edges not in $\mathcal{G}_1$ and C_2.

(b) If there is more than one common vertex between $\mathcal{G}_1$ and C_2, then let V_1' be the first other common vertex in the portion of C_2 formed by the directed path $\{V_1, V_2, \ldots, V_1'\}$. Then along the directed path from V_1' to V_1 in $\mathcal{G}_1$, we can assign a direction to each edge in $V_1, V_2, \ldots, V_1'$ to form a directed cycle C_2', and so $\mathcal{G}_2$, the graph induced by the vertices of $C_1 \cup C_2'$, is strongly connected regardless of the directions of edges not in the cycles C_1 and C_2'. Since $\mathcal{G}$ has finitely many vertices, after r iterations of this procedure, we will have $\mathcal{G}_r$, the graph induced by the vertices of $C_1 \cup C_2' \cup \cdots \cup C_r'$, which contains all the n vertices in $\mathcal{G}$, and so $\mathcal{G}_r = \mathcal{G}$ is strongly connected.

This concludes the proof. □

Below is a simple illustration of the central concepts of the above proof.

Example 6.1. *First, let us consider Case (a). That is, below are two cycles that share exactly one common vertex. Here, we consider the cycles*

$$C_1 = \{V_1, U, V, V_1\} \quad \text{and} \quad C_2 = \{V_1, V_2, V_3, V_4, V_1\}.$$

Observe that, in Figure 26, the graph induced by $C_1 \cup C_2$ is strongly connected, since it contains the closed walk

$$W := \{V_1, U, V, V_1, V_2, V_3, V_4, V_1\}.$$

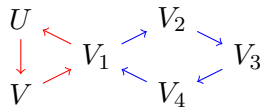

FIGURE 26. Case (a) with C_1 and C_2.

Furthermore, below are simple illustrations of an instance where Case (b) is in effect. Let $\mathcal{G}$ be the graph in Figure 27 on the left. First, we acknowledge $\mathcal{G}_1$, the graph induced by

$$C_1 = \{U, V, V_1, V_5, V_1', U\},$$

and assign directions to the edges.

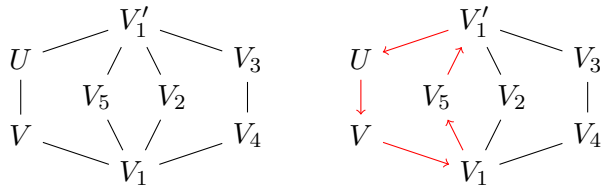

FIGURE 27. An example of Case (b) with C_1 designated.

Notice that $\mathcal{G}_1$ is strongly connected. Now, we observe that the vertices V_1, V_2, V_1', V_3, and V_4 induce another cycle. We will call this cycle C_2. Then, we identify the path $\{V_1, V_2, V_1'\}$ as the portion of C_2 that shares two common vertices with C_1 and construct the cycle $C_2' = \{V_1, V_2, V_1', U, V, V_1\}$. Thus, we have assigned directions to the edges in $\mathcal{G}_2$, the graph induced by the vertices in $C_1 \cup C_2'$. Observe that $\mathcal{G}_2$ is strongly connected. Now, we observe the cycle

$$C_3 = \{V_1, V_4, V_3, V_1', U, V, V_1\}.$$

Next, we identify the path $\{V_1, V_4, V_3, V_1'\}$ as the portion of C_3 that shares two common vertices with $\mathcal{G}_2$ and construct the cycle $C_3' = \{V_1, V_4, V_3, V_1', U, V, V_1\}$.

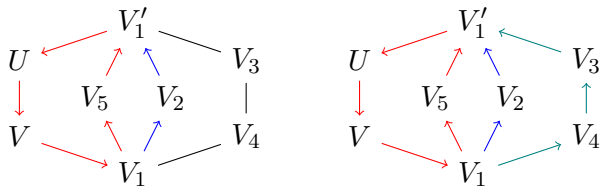

FIGURE 28. An example of Case (b) with $C_1 \cup C_2' \cup C_3'$ designated.

Now, after appending C_3', we have that each edge of the graph has been assigned a direction. Moreover, $\mathcal{G}_3$, the graph induced by $C_1 \cup C_2' \cup C_3'$, contains all of the vertices and edges of $\mathcal{G}$. Furthermore, we have that $\mathcal{G}_3$ is strongly connected and that $\mathcal{G}_3 = \mathcal{G}$. Therefore, $\mathcal{G}$ is strongly connected with the assigned directions.

Proof of Theorem 6.1 (Without using Lemma 5.1). First, we know that since $\mathcal{G}$ has no bridges, $\mathcal{G}$ is not a tree. Hence $\mathcal{G}$ contains a cycle $C_1 = \{V_1, V_2, \ldots, V_k, V_1\}$. Assigning directions

$$(V_1, V_2), \ldots, (V_{k-1}, V_k), (V_k, V_1),$$

we obtain a directed cycle C_1 and consequently achieve a strongly connected subgraph of $\mathcal{G}$. If there are any chords in C_1, we arbitrarily assign a direction to the chords in question. Moving forward, we have a strongly connected submultigraph $\mathcal{C}_1^*$ of $\mathcal{G}$ consisting of all vertices in C_1 and all the edges of $\mathcal{G}$ which have both end vertices in C_1. Now, we will employ $\mathcal{C}_1^*$ to construct a new connected multigraph $\mathcal{G}_1$ without a bridge. To this end, we will consider $\mathcal{C}_1^*$ as a single vertex, say $\overline{V}$, and all the vertices in $\mathcal{G}$ but not in $\mathcal{C}_1^*$ as the other vertices in a new graph $\mathcal{G}_1$, with vertex set $\overline{V} \cup (V(\mathcal{G}) - V(\mathcal{C}_1^*))$, and keep all the edges in $\mathcal{G}$ while collapsing all the edges in $\mathcal{C}_1^*$ into the vertex $\overline{V}$. Then, we are left with the following cases.

(a) If $\mathcal{G}_1$ contains only one vertex $\overline{V}$, then $\mathcal{C}_1^* = \mathcal{G}$, and so we are done.

(b) If $\mathcal{G}_1$ contains more than one vertex, then $\mathcal{G}_1$ is a connected multigraph with $n_1 > 1$ vertices without a bridge and $n_1 < n$. Since n is a finite number, after finitely many, say r, steps, we will have that $\mathcal{G}_r$ contains a single vertex. As a result, the entire graph $\mathcal{G}$ becomes a strongly connected multigraph.

This concludes our proof. □

Let us illustrate the above proof.

Example 6.2. *First, we will consider Case (a). Observe that the graph in Figure 29 is a cycle with a single chord, and so we can assign directions to the edges of the graph in such a*

way that the resulting graph is strongly connected. Furthermore, this allows us to immediately collapse the entire directed graph into $\overline{V}$*, thus completing our procedure.*

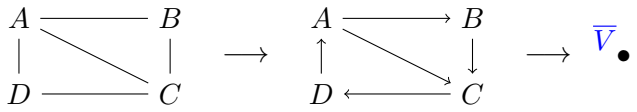

FIGURE 29. An example of Case (a).

Note that we could have assigned the cycle to be counter-clockwise and the chord in the other direction.

Now, we will contemplate an instance of Case (b). Consider the graph in Figure 30.

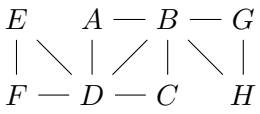

FIGURE 30. An example of a graph satisfying Case (b).

Next, we will identify cycles, direct each corresponding cycle, and collapse them one by one until we are left with a single "vertex" representing the entire graph as a strongly connected portion. First, we consider C_1*.*

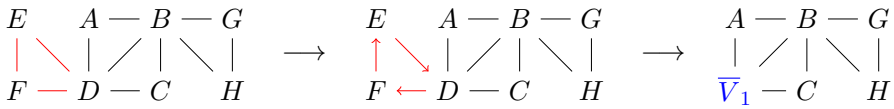

FIGURE 31. The identification of C_1, direction of $\mathcal{C}_1^*$, and collapse of $\mathcal{C}_1^*$ into $\overline{V}_1$.

Next, we turn our attention to C_2*.*

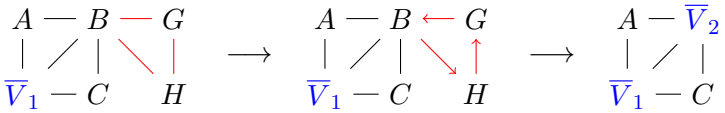

FIGURE 32. The identification of C_2, direction of $\mathcal{C}_2^*$, and collapse of $\mathcal{C}_2^*$ into $\overline{V}_2$.

Finally, we will acknowledge C_3*.*

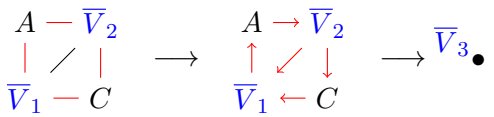

FIGURE 33. The identification of C_3, direction of $\mathcal{C}_3^*$, and collapse of $\mathcal{C}_3^*$ into $\overline{V}_3$.

Now, since each $\overline{V}_i$ is strongly connected and we have simplified the given graph to $\overline{V}_3$, we are done and have a strongly connected graph.

7. Conclusion

As mentioned in the introduction, studying strong connectivity in graphs is instrumental in various real-world applications of graph theory, including social networks and transportation systems. The ability to determine whether a graph's edges can be oriented to produce a strongly connected directed graph is particularly valuable for analyzing graph structures and the relationships they model. Furthermore, understanding exactly how to construct such an edge direction assignment could also prove to be a very useful endeavor. Within our paper, we have presented two proofs of a known result regarding the ability of the Depth-First Search Algorithm to generate a strongly connected graph from a bridgeless graph as well as two proofs of the existence of a strongly connected assignment of edge directions for a bridgeless multigraph.

For a potential application of the content of this manuscript, note that the algorithms that we provided in the proofs can be implemented into software for the purpose of utilizing computers to extrapolate information concerning computationally complex strongly connected graphs, multigraphs, and components that could potentially pertain to real-world applications.

Acknowledgments. The authors would like to thank the referee for some useful comments and their helpful suggestions that have improved the quality of this paper.

References

[1] Alon, N., & Erdős, P. (1985). An application of graph theory to additive number theory. *European Journal of Combinatorics, 6*(3), 201–203. https://doi.org/10.1016/S0195-6698(85)80027-5

[2] Xu, L., Jiang, C., Wang, J., Yuan, J., & Ren, Y. (2014). Information security in big data: Privacy and data mining. *IEEE Access, 2*, 1149–1176. https://doi.org/10.1109/ACCESS.2014.2362522

[3] Wu, J., & Tsai, Y. J. (2005). Real-time speed limit sign recognition based on locally adaptive thresholding and depth-first-search. *American Society for Photogrammetry and Remote Sensing, 71*(4), 405–414. https://doi.org/10.14358/PERS.71.4.405

[4] Zhang, L., & Li, K. (2021). Influence maximization based on backward reasoning in online social networks. *Mathematics, 9*(24), 3189. https://doi.org/10.3390/math9243189

[5] Amigó, J. M., Gálvez, J., & Villar, V. M. (2009). A review on molecular topology: Applying graph theory to drug discovery and design. *Naturwissenschaften, 96*, 749–761. https://doi.org/10.1007/s00114-009-0536-7

[6] Balaban, A. T. (1985). Applications of graph theory in chemistry. *Journal of Chemical Information and Computer Sciences, 25*, 334–343.

[7] Balasubramanian, K. (1985). Applications of combinatorics and graph theory to spectroscopy and quantum chemistry. *Chemistry Review, 85*, 599–618.

[8] Biggs, N. L., Lloyd, E. K., & Wilson, R. J. (1999). *Graph Theory 1736–1936* (1st ed.). Oxford, England: Clarendon Press.

[9] Biró, C., & Kusper, G. (2018). Equivalence of strongly connected graphs and black-and-white 2-SAT problems. *Miskolc Mathematical Notes, 19*(2), 755–768. https://zbmath.org/1424.94109

[10] Chakraborty, A., Dutta, T., Mondal, S., & Nath, A. (2018). Application of graph theory in social media. *International Journal of Computer Sciences and Engineering, 6*(10), 722–729.

[11] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms* (4th ed.). MA, USA: The MIT Press.

[12] delEtoile, J., & Adeli, H. (2017). Graph theory and brain connectivity in Alzheimer's disease. *The Neuroscientist, 23*(6), 616–626. https://doi.org/10.1177/1073858417702621

[13] He, H., Xu, T., Chen, J., Cui, Y., & Song, J. (2024). A granulation strategy-based algorithm for computing strongly connected components in parallel. *Mathematics, 12*(11), 1723. https://doi.org/10.3390/math12111723

[14] Dhingra, S., Dodwad, P. S., & Madan, M. (2016). Finding strongly connected components in a social network graph. *International Journal of Computer Applications, 136*(7), 1–5. https://doi.org/10.5120/ijca2016908481

[15] Ray, S. S. (2012). *Graph Theory with Algorithms and its Applications: In Applied Science and Technology.* Springer.

[16] Hansen, P. J., & Jurs, P. C. (1988). Chemical applications of graph theory. Part I. Fundamentals and topological indices. *Journal of Chemical Education, 65*(7), 574–580. https://doi.org/10.1021/ed065p574

[17] Dossey, J. A., Otto, A. D., Spence, L. E., & Vanden Eynden, C. (2018). *Discrete Mathematics* (5th ed.). NY, USA: Pearson.

[18] Even, S. (2005). *Graph Algorithms* (2nd ed.). Cambridge, England: Cambridge University Press.

[19] Even-Tzur, G. (2001). Graph theory applications to GPS networks. *GPS Solutions, 5*(1), 31–38. https://doi.org/10.1007/PL00012874

[20] Gross, J. L., Yellen, J., & Anderson, M. (2018). *Graph Theory and Its Applications* (3rd ed.). FL, USA: Chapman and Hall/CRC.

[21] Husain, Z., Al Zaabi, A., Hildmann, H., Saffre, F., Ruta, D., & Isakovic, A. F. (2022). Search and rescue in a maze-like environment with ant and Dijkstra algorithms. *Drones, 6*(10), 273. https://doi.org/10.3390/drones6100273

[22] Jayabalasamy, G., Pujol, C., & Latha Bhaskaran, K. (2024). Application of graph theory for blockchain technologies. *Mathematics, 12*(8), 1133. https://doi.org/10.3390/math12081133

[23] Kumar, N., & Kaur, S. (2019). A review of various maze solving algorithms based on graph theory. *International Journal for Scientific Research and Development, 6*, 431–434.

[24] Majeed, A., & Rauf, I. (2020). Graph theory: A comprehensive survey about graph theory applications in computer science and social networks. *Inventions, 5*(1), 10. https://doi.org/10.3390/inventions5010010

[25] Martín-Nieto, M., Castaño, D., Horta Muñoz, S., & Ruiz, D. (2024). Solving mazes: A new approach based on spectral graph theory. *Mathematics, 12*(15), 2305. https://doi.org/10.3390/math12152305

[26] Mouronte-López, M. L. (2021). Modeling the public transport networks: A study of their efficiency. *Complexity*. https://doi.org/10.1155/2021/3280777

[27] Muhiuddin, G., Samanta, S., Aljohani, A. F., & Alkhaibari, A. M. (2023). A study on graph centrality measures of different diseases due to DNA sequencing. *Mathematics, 11*(14), 3166. https://doi.org/10.3390/math11143166

[28] Nandhini, R., Maheswari, V., & Balaji, V. (2018). A graph theory approach on cryptography. *Journal of Computational Mathematica, 2*(1), 97–104. https://doi.org/10.26524/32

[29] Swan, R. G. (1963). An application of graph theory to algebra. *Proceedings of the American Mathematical Society, 14*(3), 367–373. https://doi.org/10.2307/2033801

[30] Thornton, C. J., & Boulay, B. du. (1992). *Artificial Intelligence Through Search* (1st ed.). Springer.

[31] Zhao, J., Jiang, N., Pei, K., Wen, J., Zhan, H., & Tu, Z. (2024). TPoison: Data-poisoning attack against GNN-based social trust model. *Mathematics, 12*(12), 1813. https://doi.org/10.3390/math12121813

Department of Mathematics, University of Kentucky, 101 Main Building, Lexington, 40506, KY, USA

Department of Mathematics, Bradley University, 1501 W Bradley Ave, Peoria, 61625, IL, USA

Department of Mathematics, Bradley University, 1501 W Bradley Ave, Peoria, 61625, IL, USA