WEIGHTED CHAINED GRAPHS AND SOME APPLICATIONS*

C. FENU†, L. REICHEL‡, G. RODRIGUEZ*, AND Y. ZHANG§

Abstract. This paper introduces weighted chained graphs, as well as minimal broadcasting and receiving sets, and investigates their properties. Both directed and undirected graphs are considered. Applications to the analysis of bu and airline networks are presented.

Key words. network analysis, weighted chained graph, central vertex

1. Introduction. Many complex systems can be modeled as networks. A network is a set of objects, referred to as *nodes* or *vertices*, that are connected by *edges*. The nature of the nodes and edges depends on the application. Network are represented by graphs. These graphs leave out many details of the system they model. Nevertheless, they are often able to capture much of the complexity of the original system, and their relative simplicity makes them amenable to analysis. For instance, network models typically allow us to assess which nodes and edges are particularly important in a network. Networks models are employed in a wide range of areas including telecommunication, transportation, epidemiology, and biology; see e.g., Estrada [4] and Newman [5] for discussions of networks and these as well as many other applications.

We consider networks that can be represented by a weighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$, where $\mathcal{V} = \{v_i\}_{i=1}^n$ denotes a set of vertices or nodes, $\mathcal{E} = \{e_i\}_{i=1}^m$ is the set of edges that connects the vertices, and $\mathcal{W} = \{w_i\}_{i=1}^m$ are edges weights, i.e., the edge e_i is equipped with the weight w_i. Each weight w_i is a non-negative scalar that indicates the strength (or importance) of the connection between the vertices that are connected by the edge e_i. A graph is said to be *unweighted* if all positive weights w_i equal 1. We will consider simple graphs, i.e., graphs without multiple edges and self-loops. A *directed edge* e_k that points from vertex v_i to v_j can be identified with the ordered pair $e_k = (v_i, v_j)$. A directed edge may be considered a "one-way street." An *undirected edge* e_k between the vertices v_i and v_j is identified with the set $e_k = \{v_i, v_j\}$. Thus, an undirected edge may be regarded as a "two-way street." If all the edges of a graph are undirected, then the graph is said to be *undirected*; otherwise the graph is directed.

A *walk* with $k+1$ vertices is a sequence of vertices $v_{i_1}, v_{i_2}, \ldots, v_{i_{k+1}}$ and an associated sequence of edges $e_{i_1}, e_{i_2}, \ldots, e_{i_k}$ such that the edge e_{i_j} in this walk points from vertex v_{i_j} to vertex $v_{i_{j+1}}$ for $j = 1, 2, \ldots, k$. An undirected edge e_i in this walk is said to be between the vertices v_{i_j} and $v_{i_{j+1}}$. The length of the walk determined by the vertices $v_{i_1}, v_{i_2}, \ldots, v_{i_{k+1}}$ and edges $e_{i_1}, e_{i_2}, \ldots, e_{i_k}$ is defined as the sum of the weights of the edges in the walk, i.e., the length is given by $\sum_{j=1}^k w_{i_j}$. In particular, for an unweighted graph, the length of a walk is the number of edges that make up

[0]Version May 22, 2026

*This paper has been published on *Appl. Numer. Math.* 208:232–245, 2025, DOI:10.1016/j.apnum.2023.12.017.

†Dipartimento di Matematica e Informatica, Università di Cagliari, via Ospedale 72, 09124 Cagliari, Italy. E-mail: kate.fenu@unica.it, rodriguez@unica.it

‡Department of Mathematical Sciences, Kent State University, Kent, OH 44242, USA. E-mail: reichel@math.kent.edu

§School of Data Science and Artificial Intelligence, Dongbei University of Finance and Economics, Dalian, 116025, China. E-mail: yunzizhang@yahoo.com

the walk. Vertices and edges of a walk may be repeated. A *path* is a walk in which no vertex is repeated. Assume that there is a path from node v_i to node v_j. Then the *distance* $d(v_i, v_j)$ from node v_i to node v_j is the length of the shortest path from node v_i to node v_j. If the graph is unweighted, then $d(v_i, v_j)$ is the number of edges in the shortest path from v_i to v_j. Note that $d(v_i, v_j)$ may differ from $d(v_j, v_i)$; in fact, some distances might not be defined.

Recently, Concas et al. [1, 2] introduced the notions of chained directed and chained undirected graphs and some generalizations. For an undirected graph, the notion of chained graph generalizes bipartivity and allows the determination of *central nodes* of the graph; see [2]. The analysis is based on the use of spanning trees for the graph. A generalization to directed graphs is described in [1]. Under suitable conditions, the chained structure can be uncovered by using spanning trees for directed graphs. When applicable, this analysis allows the definition of central nodes, and has been used to shed light on the structure of graphs that arise in a variety of applications [1]. However, some directed graphs do not have a directed spanning tree, and then the approach to define central nodes of an undirected graph proposed in [1] cannot be applied.

It is the purpose of this paper to generalize the approach to chained structure for directed graphs presented in [1] in several ways: We base our analysis on spanning forests instead of on spanning trees. This allows us to identify a chained structure, if present, for a general directed graph. Moreover, we allow edge weights different from one. This generalizes results both in [1, 2] and allows us to define weighted chained structures both for directed and undirected graphs, as well as to investigate how sensitive the computed results are to perturbations of the graphs to changes in the weights.

This paper is organized as follows. Section 2 generalizes results on chained structure for unweighted undirected graphs described in [2] to weighted directed graphs. Our approach to define chained structure is based on the application of directed forests. This allows the definition of chained structure for a larger set of undirected graphs than the approach in [1]. Moreover, the graphs are allowed to have edge weights different from unity. Section 3 defines the notions of node centrality for directed and weighted graphs. Broadcasting and receiving sets are introduced. This allows the definitions of out-centrality and in-centrality of nodes. The sensitivity of these centrality measures to perturbations in the weights is also discussed. Section 4 is concerned with the special case of weighted undirected graphs and generalizes the discussion in [2] by allowing weights different from unity. Undirected graphs are considered in Section 3. Several applications are presented in Section 5, and concluding remarks can be found in Section 6.

2. Chained structure of weighted directed graphs. The following definition extends the notion of chained graphs introduced in [1].

DEFINITION 2.1. *A weighted directed graph (in short, "digraph") $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$ is said to be weighted* ℓ-chained *with initial vertex set $\mathcal{V}_1$ if to each edge $e \in \mathcal{E}$ is associated a positive weight $w \in \mathcal{W}$ and the set of vertices can be partitioned into ℓ disjoint non-empty subsets*

$$\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_\ell \tag{2.1}$$

such that all edges connect vertices in the set $\mathcal{V}_j$ to vertices in the set $\mathcal{V}_{j+1}$, for $j = 2, 3, \ldots, \ell - 1$. The chain length ℓ is the largest number of vertex subsets $\mathcal{V}_j$. The vertex sets $\mathcal{V}_j$ and $\mathcal{V}_{j+1}$, for $j = 1, \ldots, \ell - 1$, are said to be consecutive.

An undirected edge in a digraph may be regarded as a pair of directed edges of opposite orientations. Many real-world weighted networks may not have an ℓ-chained structure. We therefore also consider more general networks, as defined by Definition 2.1, which allows edges between non-consecutive vertex subsets.

DEFINITION 2.2. *A weighted digraph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$ is said to be weighted (ℓ, k)-chained with initial vertex set $\mathcal{V}_1$ if it has the chained structure described in Definition 2.1 with the extension that edges may connect vertices belonging to non-consecutive vertex subsets. The* lower bandwidth *k is defined as the largest integer for which there exists an edge from a vertex in the subset $\mathcal{V}_j$ and a vertex in the subset $\mathcal{V}_{j-k}$.*

REMARK 2.3. *The bandwidth k is always larger than or equal to -1 and smaller than ℓ. If $k = -1$, then Definition 2.2 agrees with Definition 2.1. If an edge connects a node v_1 in $\mathcal{V}_j$ to a node v_2 in $\mathcal{V}_{j+k}$, with $k > 1$, then node v_2 is to be moved to set $\mathcal{V}_{j+1}$.*

A digraph is said to be *strongly connected* if there exist directed paths connecting each vertex pair (v_i, v_j) in both directions. It is *semi-connected* if for some vertex pair such a connection exists only in one direction. A digraph is *weakly connected* if there is an undirected path that connects any vertex pair (v_i, v_j), i.e., a path obtained by replacing all directed edges by undirected ones. An undirected graph is said to be *connected* if each vertex pair is connected by a path; see [4, 5].

THEOREM 2.4. *Every weakly connected digraph is (ℓ, k)-chained.*

Proof. Let the node set $\mathcal{V}_1$ initially only contain the node v_1, and consider undirected unweighted paths for all vertex pairs (v_1, v_i), $i = 2, \ldots, n$. Let one of these paths, of length $s - 1$, be $(v_1, v_{i_2}, \ldots, v_{i_{s-1}}, v_{i_s})$. If there is a directed edge that connects v_1 to v_{i_2}, then $v_{i_2} \in \mathcal{V}_2$; otherwise $v_{i_2} \in \mathcal{V}_0$. Proceed similarly for all other nodes in the path. If node v_{i_r} in the path already has been assigned to a set $\mathcal{V}_j$, then the node $v_{i_{r+1}}$ must be attributed either to the set $\mathcal{V}_{j-1}$ or to the set $\mathcal{V}_{j+1}$, depending on the orientation of the edge that connects the two vertices. The proof is the same for undirected weighted graphs.

Since the graph is weakly connected, every node will be assigned to a set $\mathcal{V}_j$ and, given the way the sets have been constructed, every node in $\mathcal{V}_j$ will be connected to a node in $\mathcal{V}_{j+1}$. When the process ends, the node sets must be renumbered as $\mathcal{V}_1, \ldots, \mathcal{V}_\ell$, and the missing edges must be added to the new graph. If the initial graph is ℓ-chained, we will end up with the chained structure of the graph. Otherwise, there will be some connections going from nodes in $\mathcal{V}_j$ to $\mathcal{V}_{j-k}$. If some k is smaller than -1, then the corresponding node must be relocated so that $k \geq -1$. In the end, the largest value of k will identify the underlying (ℓ, k)-chained structure. ∎

The following example explains how to detect an (ℓ, k)-chained structure of a weakly connected graph.

EXAMPLE 2.1. *Consider the weakly connected digraph shown in Figure 2.1(a). To detect its (ℓ, k)-chained structure, let $v_1 \in \mathcal{V}_1$ be the initial node. Since there is a directed edge e_1 that points from v_1 to v_2, node v_2 should be placed in vertex set $\mathcal{V}_2$. Edge e_2 provides a connection from v_3 to v_2. Therefore, node v_3 belongs to vertex set $\mathcal{V}_1$. Continuing the process, nodes v_4 and v_5 are assigned to $\mathcal{V}_2$. Then, for the last node v_6, we have two possible assignments. We may assign v_6 to the vertex set $\mathcal{V}_1$ according to the direction of edge e_5. The edge e_6, which points from v_4 to v_6, indicates that $k = 1$ and a $(2, 1)$-chained graph is achieved; see Figure 2.1(b).*

Alternatively, we may consider the direction of edge e_6 and assign v_6 to $\mathcal{V}_3$, as in Figure 2.1(c). However, this configuration is not permitted, as there is a "long" forward connection from v_3 to v_6. Hence, v_6 has to be moved to the set $\mathcal{V}_2$. Then, edges e_5 and e_6 indicate that $k = 0$, and we obtain the $(2, 0)$-chained graph of Figure 2.1(d). Considering the edge e_7 would result in the same node sets.

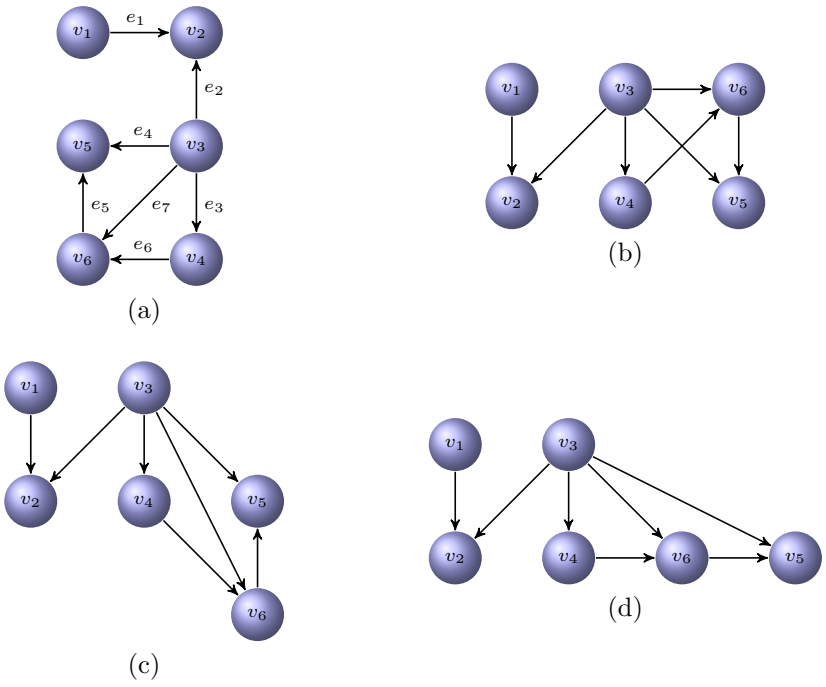

FIG. 2.1. *A weakly connected digraph (a) and two (ℓ, k)-chained structures (b and d) of it.*

THEOREM 2.5. *Every digraph is (ℓ, k)-chained.*

Proof. Since any digraph is the union of weakly connected components, Theorem 2.4 can be applied to every one of these components. The chained structures of these components can be merged. For instance, let two components produce the chained structures $(\mathcal{V}_1, \ldots, \mathcal{V}_\ell)$ and $(\mathcal{V}'_1, \ldots, \mathcal{V}'_{\ell'})$, with $\ell' < \ell$. Then the first structure can be updated, without increasing the chain length, by setting $\mathcal{V}_i = \mathcal{V}_i \cup \mathcal{V}'_{i-j}$, $i = j + 1, \ldots, j + \ell'$, for any $j = 0, \ldots, \ell - \ell'$. ∎

The chained structure of a directed graph can be identified independently of the weights. For the sake of simplicity, we drop the weights in the following, when they are not relevant.

DEFINITION 2.6. *A digraph is said to be an* out-tree *if it is acyclic (i.e., it does not contain any directed or undirected cycle) and if it has only one vertex v with zero in-degree. The vertex v is called the* root *of the out-tree. When the graph is undirected, the in-degree and out-degree of a node are the same, and are referred to as the degree of the node.*

Sometimes, an out-tree is referred to as an *arborescence*, or a *branching*; see [3]. Figure 2.2 shows an example of an arborescence.

DEFINITION 2.7. *A digraph is said to be an* in-tree *if it is acyclic and only has one vertex v with zero out-degree. This vertex is referred to as the* root *of the in-tree.*

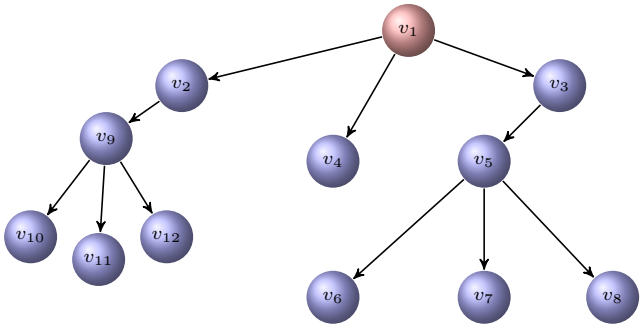

FIG. 2.2. *An arborescence rooted at node* v_1.

DEFINITION 2.8. *A* spanning subgraph $\mathcal{G}' = \{\mathcal{V}, \mathcal{E}'\}$ *of a weakly connected digraph* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ *is a graph with the same vertex set* $\mathcal{V}$ *but a possibly smaller edge set* $\mathcal{E}' \subset \mathcal{E}$. *The definitions of spanning out-trees and spanning in-trees are immediate.*

Figure 2.3 shows an example of a digraph which admits a spanning out-tree rooted at node v_1.

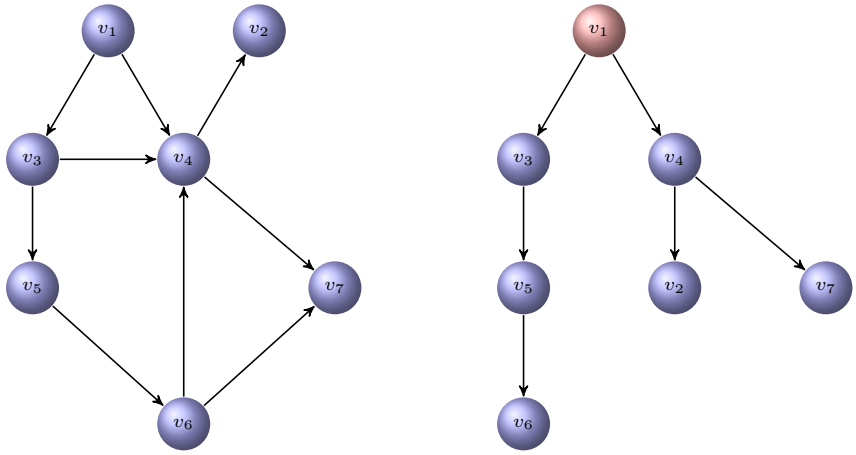

FIG. 2.3. *A digraph (left) and one of its spanning arborescences (right).*

Not all digraphs admit a spanning out-tree or in-tree. These digraphs can be studied with the aid of spanning forests.

DEFINITION 2.9. *A* spanning forest $\mathcal{G}' = \{\mathcal{V}, \mathcal{E}'\}$ *for a digraph* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ *is an acyclic spanning subgraph, that may consist of disconnected components. Every connected component of a spanning forest is either an out-tree or an in-tree.*

Every spanning forest is a weighted ℓ-chained graph with the vertex set $\mathcal{V}_1$ containing the roots of each arborescence and the subset $\mathcal{V}_i$ containing the children of the vertices in $\mathcal{V}_{i-1}$, for $i = 2, \ldots, \ell$. The roots of each out-tree may be placed in node sets different from $\mathcal{V}_1$, if this is useful.

A chained digraph is not necessarily a forest, but any (ℓ, k)-chained digraph has a spanning forest. Figure 2.4 shows an (ℓ, k)-chained graph which is not a forest and does not admit a spanning out-tree, together with a spanning forest for it. This digraph is $(4, 2)$-chained with $\mathcal{V}_1 = \{v_1, v_2\}$, $\mathcal{V}_2 = \{v_3, v_4\}$, $\mathcal{V}_3 = \{v_5, v_7\}$, and $\mathcal{V}_4 = \{v_6\}$; the bandwidth is 2 because there is a connection from $v_6 \in \mathcal{V}_4$ to $v_4 \in \mathcal{V}_2$. Figure 2.5

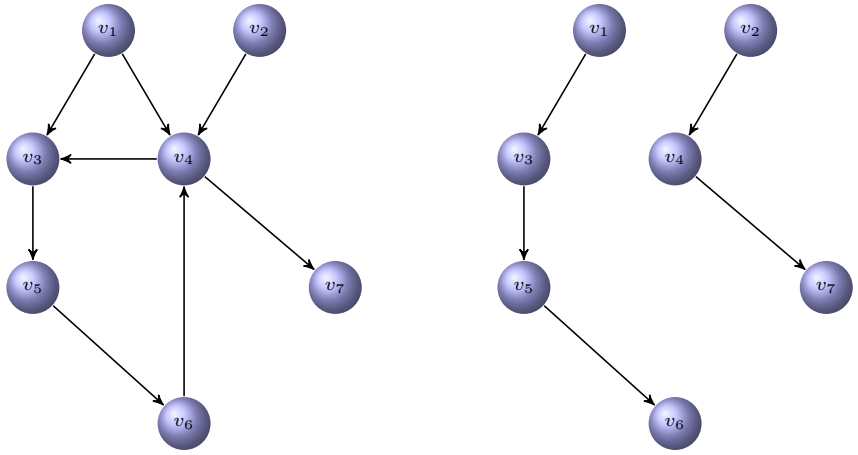

FIG. 2.4. *On the left, a digraph which does not admit a spanning arborescence; on the right, a spanning forest for the graph.*

displays another spanning forests starting from node v_1, and a spanning forest starting from v_2.

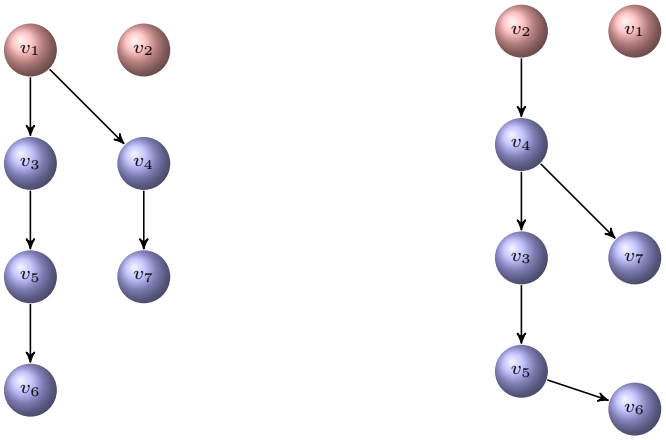

FIG. 2.5. *Spanning forests of the digraph in Figure 2.4.*

The (ℓ, k)-chained structure of a weakly connected graph is closely related to a spanning forest for the graph, in the sense that the latter can be deduced from the former. On the contrary, a spanning forest alone does not allow the determination of the chained structure for the graph that it spans, because the spanning forest lacks information about which sets $\mathcal{V}_j$ contain the roots.

Motivated by the above discussion, we show in the following that spanning trees and forests are useful tools for detecting the chained structure of a digraph. Algorithm 1 constructs a forest containing a subset of the nodes either made of out-trees or of in-trees, depending on the value of the third input argument. The result is simply a tree if a single node is given as input. If an incomplete chained structure for a network is given as input, then Algorithm 1 extends this structure by constructing either all the out-trees by starting from nodes in $\mathcal{V}_1$ or all in-trees ending at nodes in $\mathcal{V}_\ell$.

The algorithm starts by scanning the input chained structure by considering all the available node sets $\mathcal{V}_j$, $j = 1, \ldots, \ell$, or initializing a new set as empty; see lines 5–9. If an out-forest is sought, then the algorithm selects all the nodes pointed to by vertices in $\mathcal{V}_{j-1}$, removing those already present in another set (lines 10–18), and adds them to the set $\mathcal{V}_j$. A slight modification of the algorithm allows to handle an in-forest. The iteration is interrupted when all the nodes have been added (in this case a spanning forest is obtained) or when no new nodes are available. The output is the updated chained structure and the corresponding forest.

Algorithm 1 Construction of an out-forest or a in-forest (function IOFOREST).

Require: Adjacency matrix $A = [a_{ij}]_{i,j=1}^{n} \in \mathbb{R}^{n \times n}$, incomplete directed chained structure $\mathcal{V}$, inflag = 'in' or 'out'

Ensure: Out/in forest $\mathcal{F} = \{\mathcal{T}_1, \ldots, \mathcal{T}_\tau\}$ starting/ending at the first/last set of $\mathcal{V}$, updated chained structure $\mathcal{V}$, set $\mathcal{S}$ of the nodes involved

1: **if** inflag = 'in' **then** reverse the chained structure $\mathcal{V}$
2: $\mathcal{S} = \cup \mathcal{V}_i$, $\ell = \text{length}(\mathcal{V})$, $j = 1$
3: **while** cardinality$(\mathcal{S}) < n$ **do**
4: $j = j + 1$
5: **if** $j \leq \ell$ **then**
6: $V = \mathcal{V}_j$
7: **else**
8: $V = \emptyset$
9: **end if**
10: **for** $v \in \mathcal{V}_{j-1}$ **do**
11: **if** inflag = 'in' **then**
12: $\Omega = \{\text{nodes pointing to } v\}$
13: **else**
14: $\Omega = \{\text{nodes pointed by } v\}$
15: **end if**
16: $\Omega = \Omega \setminus (\Omega \cap \mathcal{S})$ (remove from Ω the nodes already contained in $\mathcal{S}$)
17: $V = V \cup \Omega$, $\mathcal{S} = \mathcal{S} \cup \Omega$
18: **end for**
19: **if** $V = \emptyset$ **and** $j > \ell$ **then**
20: $j = j - 1$
21: exit the **while** loop
22: **end if**
23: $\mathcal{V}_j = V$
24: **end while**
25: Extract an out/in forest $\mathcal{T}$ from $\mathcal{V}$
26: **if** inflag = 'in' **then** reverse the chained structure $\mathcal{V}$

Algorithm 2 identifies the chained structure of a graph, returning the node sets $\mathcal{V}_i$, $i = 1, \ldots, \ell$, and the corresponding spanning forest. It starts by constructing, in line 1, the out-tree starting at a chosen node v. This produces an initial partial chained structure $\mathcal{V}$. Then the algorithm extends this structure (see lines 4–9) by iteratively constructing the in-forest ending at $\mathcal{V}_\ell$ and the out-forest starting at $\mathcal{V}_1$. This is done by calling Algorithm 1. Algorithm 2 terminates when all the nodes have been included in the chained structure. If this fails, then the graph is not weakly connected; this follows from Theorem 2.4. If needed, then the chained structure

obtained can be extended by initializing Algorithm 2 with a node that is not in the connected component just found. The chained structures so determined may be joined by identifying the node sets with the same index, but different couplings are possible; see the proof of Theorem 2.5.

Algorithm 2 Identification of a directed (ℓ, k)-chained graph.

Require: Adjacency matrix $A = [a_{ij}]_{i,j=1}^{n} \in \mathbb{R}^{n\times n}$, initial node v
Ensure: Directed chained structure $\mathcal{V} = \{\mathcal{V}_1, \ldots, \mathcal{V}_\ell\}$ and spanning forest $\mathcal{F} = \{\mathcal{T}_1, \ldots, \mathcal{T}_\tau\}$, if they exist
1: $[\mathcal{F}, \mathcal{V}, \mathcal{S}] = \text{IOFOREST}(A, v, \text{'out'})$ *% determine an out-forest $\mathcal{F}$, with root at node v, and the corresponding chained structure $\mathcal{V}$ involving the nodes in $\mathcal{S}$*
2: $N_{\text{old}} = -1$
3: $N = \text{length}(\mathcal{S})$
4: **while** $N < n$ and $N \neq N_{\text{old}}$ **do**
5: $N_{\text{old}} = N$
6: $[\mathcal{F}, \mathcal{V}, \mathcal{S}] = \text{IOFOREST}(A, \mathcal{V}, \text{'in'})$ *% determine an in-forest ending at $\mathcal{V}_\ell$*
7: $[\mathcal{F}, \mathcal{V}, \mathcal{S}] = \text{IOFOREST}(A, \mathcal{V}, \text{'out'})$ *% determine an out-forest starting at $\mathcal{V}_1$*
8: $N = \text{length}(\mathcal{S})$
9: **end while**
10: **if** $N < n$ **then**
11: The graph has not a chained structure, so it is not weakly connected.
12: **end if**

3. Central nodes and sensitivity analysis for directed graphs. It important in some applications to be able to determine a small set of nodes that can spread information to all the network at minimal cost in a digraph, where the cost of each edge coincides with its weight. We will denote such a set as the *minimum broadcasting set*. Chained structures and spanning forests are helpful for determining such a set.

DEFINITION 3.1. *A* minimum spanning forest *is a spanning forest whose sum of weights is the minimum possible.*

DEFINITION 3.2. *A* broadcasting set *for a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$ is a subset of nodes $\mathcal{B} \subset \mathcal{V}$ such that*

1. *$\mathcal{B}$ is connected to every other node in $\mathcal{V}$, in the sense that for any $v_j \notin \mathcal{B}$ there is a node $v_i \in \mathcal{B}$ connected by a path to v_j;*
2. *$\mathcal{B}$ is "essential", that is, by removing any node from $\mathcal{B}$ property 1 is lost.*

A minimum broadcasting set *is such that the distance between $\mathcal{B}$ and any other vertex in the network*

$$\epsilon^{\text{out}}(\mathcal{B}) = \max_{v_j \notin \mathcal{B}} \min_{v_i \in \mathcal{B}} d(v_i, v_j)$$

is minimal.

Similarly, a receiving set *for a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$ is an "essential" subset $\mathcal{R} \subset \mathcal{V}$ such that there is a path from every node in $\mathcal{V}$ ending in $\mathcal{R}$. A* minimum receiving set *is such that the distance between the vertices in the network and $\mathcal{R}$,*

$$\epsilon^{\text{in}}(\mathcal{R}) = \max_{v_j \notin \mathcal{R}} \min_{v_i \in \mathcal{R}} d(v_j, v_i),$$

is minimal.

We remark that if a broadcasting set is considered as a macro-node, then $\epsilon^{\text{out}}(\mathcal{B})$ coincides with the *eccentricity* of the graph $\mathcal{G}$. By minimizing it, the radius of $\mathcal{G}$ is obtained. This corresponds to the minimum broadcasting set. We will refer to $\epsilon^{\text{out}}(\mathcal{B})$ and $\epsilon^{\text{in}}(\mathcal{R})$ as the out-eccentricity of $\mathcal{B}$ and in-eccentricity of $\mathcal{R}$, respectively.

If a network is ℓ-chained, then there is only one broadcasting set. It coincides with $\mathcal{V}_1$ plus any other node without an incoming edge.

THEOREM 3.3. *A broadcasting set for a general weakly connected graph is always the union of a subset of the set $\mathcal{V}_1$ of an (ℓ, k)-chained structure for the graph, and the set of nodes with zero in-degree.*

Proof. The subset of $\mathcal{V}_1$ is obtained by removing nodes with incoming edges only. ∎

An analogous statement can be made for the receiving set $\mathcal{R}$ and the set $\mathcal{V}_\ell$. In this case, nodes with zero out-degree must be added to $\mathcal{R}$.

In an (ℓ, k)-chained network, different chained structures can be determined. It is therefore meaningful to consider the minimum broadcasting set.

EXAMPLE 3.1. *The graph in Figure 2.1(a) admits two chained structures ((b) and (d) in the same figure). The set $\mathcal{V}_1$ for (b) is not essential, because v_3 is connected to v_6. We obtain the broadcasting set $\mathcal{B} = \{v_1, v_3\}$ by moving v_6 to $\mathcal{V}_2$. The structure (d) produces the same $\mathcal{B}$. From (d), a receiving set is obtained by moving nodes v_4 and v_6 to $\mathcal{V}_1$, yielding $\mathcal{R} = \{v_2, v_5\}$, which is essential.*

To better illustrate the connections between a broadcasting or receiving set and the rest of the vertices in a digraph, we introduce the concept of centrality to describe the spread of information from the broadcasting set or the reception of information by the receiving set.

DEFINITION 3.4. *The* out-centrality *of the broadcasting set $\mathcal{B}_i$ is defined as*

$$P_p^{\text{out}}(\mathcal{B}_i) = \left(\sum_{v_j \notin \mathcal{B}_i} \min_{v_i \in \mathcal{B}_i} d(v_i, v_j)^p \right)^{1/p},$$

where $d(v_i, v_j)$ denotes the distance from node $v_i \in \mathcal{B}_i$ to node $v_j \in \mathcal{V} \setminus \mathcal{B}_i$, as defined in Section 1, and $p \in \mathbb{R}$.

Similarly, the in-centrality *of the receiving set $\mathcal{R}_i \in \mathcal{V}$ is given by*

$$P_p^{\text{in}}(\mathcal{R}_i) = \left(\sum_{v_j \notin \mathcal{R}_i} \min_{v_i \in \mathcal{R}_i} d(v_j, v_i)^p \right)^{1/p},$$

where $p \in \mathbb{R}$.

EXAMPLE 3.2. *Let us again consider the graph in Figure 2.1(a). We identified in Example 3.1 the broadcasting set $\mathcal{B} = \{v_1, v_3\}$ and the receiving set $\mathcal{R} = \{v_2, v_5\}$. After assigning weights to each edge in the graph as shown in Figure 3.1, we let $p = 2$ and evaluate the out-centrality of the broadcasting set $\mathcal{B}$ as*

$$P_2^{\text{out}}(\mathcal{B}) = (3^2 + 3^2 + 6^2 + 8^2)^{1/2} \approx 10.863.$$

Similarly, the in-centrality of $\mathcal{R}$ with $p = 2$ is

$$P_2^{\text{in}}(\mathcal{R}) = (3^2 + 5^2 + 6^2 + 2^2)^{1/2} \approx 8.602.$$

If we let $p = 1$, then we obtain $P_1^{\text{out}}(\mathcal{B}) = 20$ and $P_1^{\text{in}}(\mathcal{R}) = 16$.

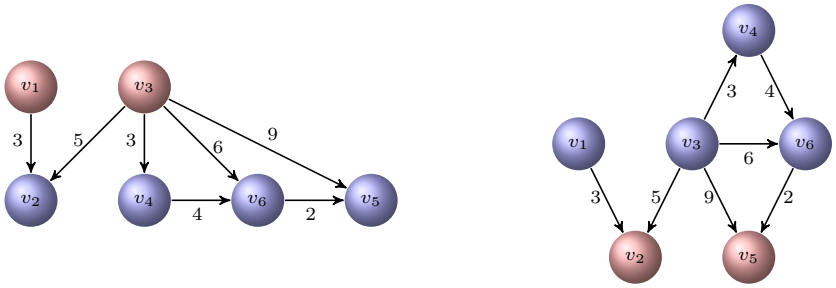

FIG. 3.1. *A weighted weakly connected digraph with the broadcasting set* $\mathcal{B} = \{v_1, v_3\}$ *on the left and the receiving set* $\mathcal{R} = \{v_2, v_5\}$ *on the right.*

It is interesting to investigate how sensitive the in- and out-centralities are to changes in the edge weights w_k. We therefore consider, for $p \neq 0$, the quantities

$$\frac{\partial P_p^{\text{out}}(\mathcal{B}_i)}{\partial w_k} = \left(P_p^{\text{out}}(\mathcal{B}_i)\right)^{1-p} \sum_{v_j \notin \mathcal{B}_i} \min_{v_i \in \mathcal{B}_i} d(v_i, v_j)^{p-1} \frac{\partial}{\partial w_k} \min_{v_i \in \mathcal{B}_i} d(v_i, v_j),$$

$$\frac{\partial P_p^{\text{in}}(\mathcal{R}_i)}{\partial w_k} = \left(P_p^{\text{in}}(\mathcal{R}_i)\right)^{1-p} \sum_{v_j \notin \mathcal{R}_i} \min_{v_i \in \mathcal{R}_i} d(v_j, v_i)^{p-1} \frac{\partial}{\partial w_k} \min_{v_i \in \mathcal{R}_i} d(v_j, v_i).$$

We remark that the if the path from d_i to d_j measured by $d(v_i, v_j)$ contains the edge e_k with weight w_k, then $\frac{\partial}{\partial w_k} \min_{v_i \in \mathcal{B}_i} d(v_i, v_j) = 1$; otherwise the derivative vanishes.

4. Central nodes and sensitivity analysis for undirected graphs. This section discusses how to determine the chained structure of weighted undirected graphs, and how to identify their center node(s). We consider graphs that are undirected, connected, simple, and weighted. The chained structure of these graphs extends the notion of chained graphs in [2].

The vertices v_i and v_j are said to be *adjacent* if there is an edge that connects these nodes.

DEFINITION 4.1. *A graph* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$ *is said to be weighted and* ℓ-chained *with initial vertex* v_i *if each edge* $e \in \mathcal{E}$ *is associated with a positive weight* $w \in \mathcal{W}$ *and the set of vertices* $\mathcal{V}$ *can be partitioned into* ℓ *disjoint non-empty subsets*

$$\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_\ell, \tag{4.1}$$

such that $v_i \in \mathcal{V}_1$ *and all the vertices in the set* $\mathcal{V}_j$ *are adjacent only to vertices in the sets* $\mathcal{V}_{j-1}$ *or* $\mathcal{V}_{j+1}$*, for* $j = 2, 3, \ldots, \ell - 1$*. The* chain length ℓ *of the graph* $\mathcal{G}$ *is the largest number of vertex subsets* $\mathcal{V}_j$ *with this property. It typically depends on the choice of initial vertex* v_i*. The vertex sets* $\mathcal{V}_j$ *and* $\mathcal{V}_{j+1}$ *are said to be consecutive for* $j = 1, 2, \ldots, \ell - 1$*.*

DEFINITION 4.2. *The graph* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$ *is said to be weighted* ℓ-semi-chained *with initial vertex* v_i *if it has the chained structure described in Definition 4.1 with the extension that connections are allowed between vertices belonging to the same vertex subset.*

We remark that every weighted undirected graph has an ℓ-(semi-)chained structure. The determination of the chained structure of weighted undirected graphs is

independent of the weights. The notion of a *tree* can be helpful for determining the chained structure of a graph.

DEFINITION 4.3. *A* tree *is a connected undirected graph in which any two vertices are connected by exactly one path. Any vertex of a tree may be designated as the* root. *Vertices with degree one, except for the root, are referred to as* leaves.

DEFINITION 4.4. *A* spanning tree *for a weighted undirected graph* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$ *is a subgraph* $\mathcal{T} = \{\mathcal{V}, \mathcal{E}', \mathcal{W}'\}$ *that is a tree and contains all the vertices of* $\mathcal{G}$.

DEFINITION 4.5. *A* shortest-path spanning tree *is a spanning tree such that the path distance from the root to any other vertex is the smallest possible.*

We remark that the shortest-path spanning tree is not necessarily unique. The following example illustrates the process of determining the chained structure of a weighted undirected graph by identifying a shortest-path spanning tree.

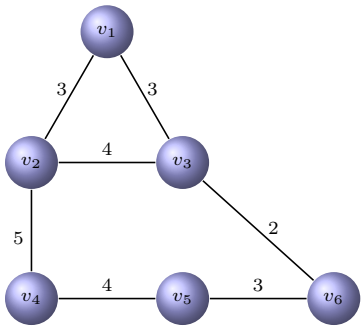

FIG. 4.1. *A weighted undirected graph.*

EXAMPLE 4.1. Figure 4.2 shows two shortest-path spanning trees rooted at vertex v_5 for the weighted undirected graph $\mathcal{G}$ in Figure 4.1. Notice that from vertex v_5 to v_2 there are two shortest paths of length 9. One path contains the edge from v_5 to v_4 and the edge from v_4 to v_2. This can be seen in the shortest-path spanning tree $\mathcal{T}_1^5$. The other path displays the shortest-path spanning tree $\mathcal{T}_2^5$, which connects nodes from v_5 to v_6, from v_6 to v_3, and from v_3 to v_2.

The chained structures of the shortest-path trees $\mathcal{T}_1^5$ and $\mathcal{T}_2^5$ also can be identified from Figure 4.2. The partition of nodes of $\mathcal{T}_1^5$ is $\mathcal{V}_1 = \{v_5\}$, $\mathcal{V}_2 = \{v_4, v_6\}$, $\mathcal{V}_3 = \{v_2, v_3\}$, and $\mathcal{V}_4 = \{v_1\}$. For the tree $\mathcal{T}_2^5$, the partition is defined by $\mathcal{V}_1 = \{v_5\}$, $\mathcal{V}_2 = \{v_4, v_6\}$, $\mathcal{V}_3 = \{v_3\}$, and $\mathcal{V}_4 = \{v_1, v_2\}$.

To determine the chained structure of the graph $\mathcal{G}$, we add the edges in $\mathcal{G}$, but not in $\mathcal{T}_1^5$, to the spanning tree $\mathcal{T}_1^5$ as shown on the left of Figure 4.3. The graph $\mathcal{G}$ is identified as a 4-semi-chained graph. Similarly, the graph on the right in Figure 4.3 is constructed by adding the missing edges in $\mathcal{G}$, but not in $\mathcal{T}_2^5$, to the spanning tree. With this node partitioning, the structure is not chained since the adjacent nodes v_4 and v_2 neither belong to two consecutive node subsets nor to the same node subset.

For weighted undirected graphs, it is interesting to determine nodes that can spread information to all the other nodes in the graph in the shortest amount of time, or for the least cost, depending on the meaning attributed to the weights of the edges. Such nodes are referred to as center nodes. The determination of center nodes can be easily achieved if the chained structure and the associated shortest-path spanning tree of a weighted undirected graph are known.

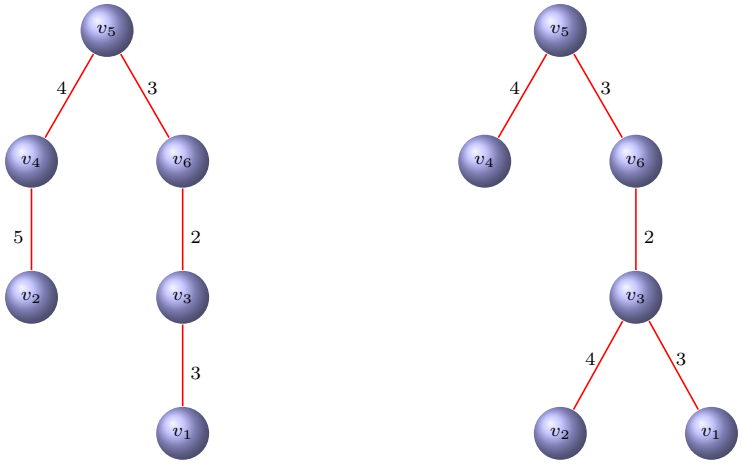

FIG. 4.2. *The shortest-path trees $\mathcal{T}_1^5$ (left) and $\mathcal{T}_2^5$ (right).*

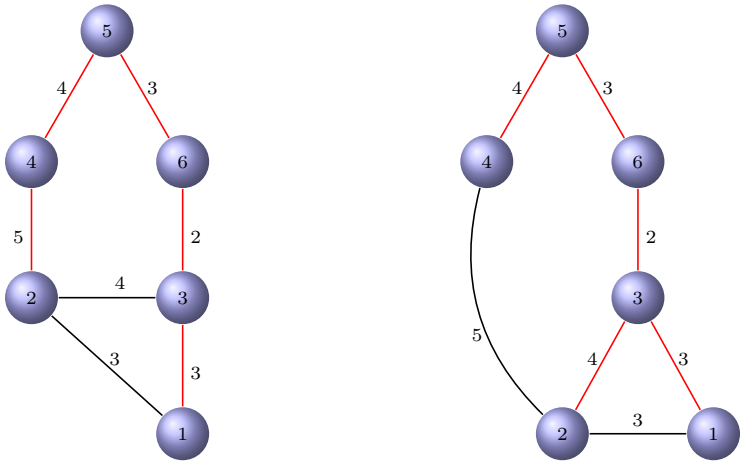

FIG. 4.3. *The missing edges (in black) are added to the trees in Figure 4.2 respectively.*

DEFINITION 4.6. *Let $\mathcal{T}$ be a shortest-path spanning tree of the weighted undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$ with chained structure*

$$\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_\ell \tag{4.2}$$

starting at vertex $v_i \in \mathcal{V}_1$. The p-position centrality of vertex v_i, for $p \geq 0$, is defined as

$$P_p(v_i) = \left(\sum_{j=1}^{n} d(v_i, v_j)^p \right)^{1/p}.$$

This centrality measure coincides with the p-norm of the vector of the distances between the vertex v_i and any other vertex in the graph. We refer to a vertex with the smallest p-position centrality as a p-center vertex.

We remark that for unweighted undirected graphs, the definition of position centrality with $p = 1$ coincides with the one given in [2] for undirected chained graphs.

EXAMPLE 4.2. *Consider the graph in Figure 4.1. The spanning trees rooted at each vertex v_i, for $i = 1, 2, \ldots, 6$, with its computed position centrality for $p = 2$ are displayed in Figure 4.4. The 2-center node is the node v_1.*

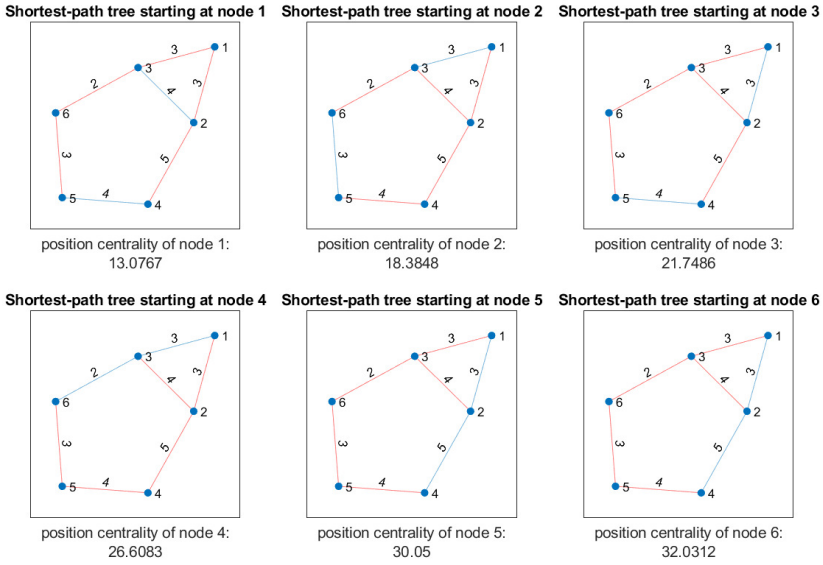

FIG. 4.4. *Shortest-path spanning trees rooted at each node of the weighted graph $\mathcal{G}$ in Figure 4.1, each with its computed position centrality for $p = 2$.*

5. Some examples. This section illustrates how the broadcasting and receiving sets and its associated chained structure can be determined in a real world transportation network. We investigate the impact of weight changes on the out- and in-centralities of the broadcasting and receiving sets.

We also consider the airline data set reported by the Bureau of Transportation Statistics of the U.S. Department of Transportation. It describes the airline routes between 129 cities in the 48 contiguous states in the U.S. for the first three quarters of 2019. From this airline data set, we obtain two graphs $\mathcal{G}_1 = \{\mathcal{V}, \mathcal{E}, \mathcal{W}_1\}$ and $\mathcal{G}_2 = \{\mathcal{V}, \mathcal{E}, \mathcal{W}_2\}$, where the node set $\mathcal{V}$ and the edge set $\mathcal{E}$ are represented by cities and airline routes, respectively. The weight $w_{i,j}$ in the first graph equals the number of passengers in the flight from city i to city j, for $i, j = 1, \ldots, 129$. If there are no flights between these cities, the weight is set to zero. Since the number of passengers of the flights are reported for each quarter, we take the average numbers as the weights. For the second graph, the weight, denoted by $\alpha_{i,j}$, represents the number of flights between two cities. Since the flights between some of the paired cities are one-way, the networks $\mathcal{G}_1$ and $\mathcal{G}_2$ are directed and weighted.

To determine the broadcasting and receiving sets, we only need to consider the graph $\mathcal{G}_1$, because both graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ have edges between the same nodes. There are 8 outward center vertices which are:

$$v_{10},\ v_{28},\ v_{33},\ v_{49},\ v_{56},\ v_{60},\ v_{68},\ v_{78}.$$

Let v_{10} be the initial vertex, the broadcasting set contains 83 vertices with 1-out-centrality equals 10973. The associated chained structure is $\{4, 2\}$-chained. The

inward center vertex is v_{25} and the receiving set contains 76 vertices. The 1-in-centrality is 12505 and the associated chained structure is $\{4, 2\}$-chained as well.

FIG. 5.1. *Airports in USA map.*

In order to display the broadcasting and receiving sets, we put red and blue dots on the map of to denote the nodes in the broadcasting set and the receiving set, respectively. We notice that there are 30 vertices appear in both broadcasting set and receiving set which are displayed by green dots on the map. This map is shown in Figure 5.1 and is available at `https://usamap360.com/usa-airports-map`. Not all of the 129 cities are marked on the map. It can be seen that the vertices marked by red and blue dots correspond to small and medium-sized cities while most of the vertices displayed by green dots represent large cities. Among the vertices in the receiving set, most of them are tourist destinations, such as San Diego and San Francisco in California, and Orlando, Tampa and West Palm Beach in Florida.

To investigate the effect of the weight changes of edges on the out- and in-centralities, we randomly pick a vertex v_i and assign it the number p_i, the number of passengers departing from v_i. We then update the weights and recalculate the out- and in-centralities as discussed in the previous sections. Table 5.1 reports the minimal and maximal relative changes of out- and in-centralities with the initial vertex and the associated chained structure. When the vertex Chicago is selected, the out-centrality of the obtained broadcasting set varies the least among all the 129 cities and the associated chained structure is $\{3, 1\}$. The largest out-centrality change of the broadcasting set is obtained when the initial vertex is located at Bangor and 7 other cities. These eight cities are also the outward center vertices. The associated chained structure is $\{4, 2\}$. There are 5 cities that lead to the same receiving set which has the largest in-centrality change. When Cincinnati is the initial vertex, the smallest in-centrality change of the receiving set is calculated.

6. Conclusion. This paper elucidates the relation between directed graphs and spanning forests. We define minimal broadcasting and receiving sets, as well as out-

TABLE 5.1

Flight network (129 vertices): the minimal and maximal relative changes of out- and in-centralities with the initial vertex and the associated chained structure.

cities	relative change of out-centrality	$\{\ell, k\}$-chained structure
Chicago	0.043	{3,1}
Bangor	0.12	{4,2}
Columbia	0.12	{4,2}
Dayton	0.12	{4,2}
Greenville/Spartanbur	0.12	{4,2}
Jackson	0.12	{4,2}
Key West	0.12	{4,2}
Martha's Vineyard	0.12	{4,2}
Nantucket	0.12	{4,2}
cities	**relative change of in-centrality**	**$\{\ell, k\}$-chained structure**
Cincinnati	0.023	{4,2}
Flint	0.709	{4,2}
Palm Springs	0.709	{4,2}
Pasco/Kennewick/Richland	0.709	{4,2}
Richmond	0.709	{4,2}
Santa Barbara	0.709	{4,2}

central nodes and in-central nodes. These notions are useful for studying communication networks and for city planning.

REFERENCES

[1] A. CONCAS, C. FENU, L. REICHEL, G. RODRIGUEZ, AND Y. ZHANG, *Chained structure of directed graphs with applications to social and transportation networks*, Appl. Netw. Sci., 7 (2022). Art. 64.

[2] A. CONCAS, L. REICHEL, G. RODRIGUEZ, AND Y. ZHANG, *Chained graphs and some applications*, Appl. Netw. Sci., 6 (2021). Art. 39.

[3] N. DEO, *Graph Theory with Applications to Engineering and Computer Science*, Dover, Mineola, 2017.

[4] E. ESTRADA, *The Structure of Complex Networks: Theory and Applications*, Oxford University Press, Oxford, 2012.

[5] M. E. J. NEWMAN, *Networks: An Introduction*, Oxford University Press, Oxford, 2010.