Growing Complex Networks for Better Learning of Chaotic Dynamical Systems

David Joseph Passey Jr.

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mathematics

Brigham Young University

ABSTRACT

Growing Complex Networks for Better Learning of Chaotic Dynamical Systems

David Joseph Passey Jr.
Department of Mathematics, BYU
Master of Science

This thesis advances the theory of network specialization by characterizing the effect of network specialization on the eigenvectors of a network. We prove and provide explicit formulas for the eigenvectors of specialized graphs based on the eigenvectors of their parent graphs.

The second portion of this thesis applies network specialization to learning problems. Our work focuses on training reservoir computers to mimic the Lorentz equations. We experiment with random graph, preferential attachment and small world topologies and demonstrate that the random removal of directed edges increases predictive capability of a reservoir topology. We then create a new network model by growing networks via targeted application of the specialization model. This is accomplished iteratively by selecting top preforming nodes within the reservoir computer and specializing them. Our generated topology out-preforms all other topologies on average.

Keywords: Complex networks, dynamical systems, reservoir computing, network growth, isospectral transformations, spectral graph theory, chaos

Acknowledgements

Thanks to Ben Webb for many hours discussing exciting problems and planning how to put this together. This thesis exists in spite of the birth of two children and largely because of my wife Bryn. Thank you for the the math tips, the debugging, the late nights working together, and most of all, thank you for your confidence in me.

Contents

List of Tables

List of Figures

1.1 Complex Networks

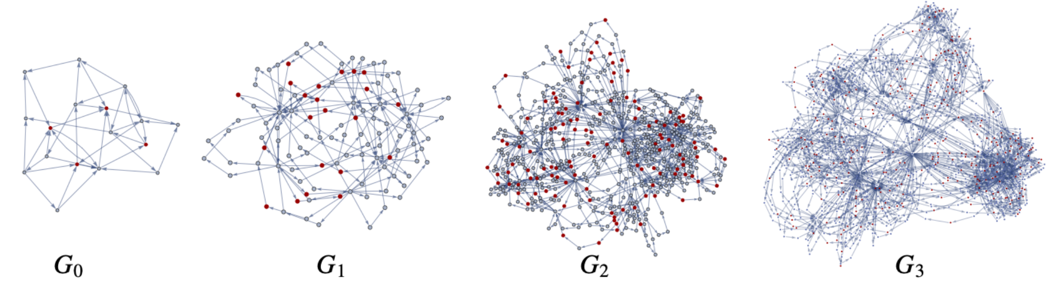


Figure 1.1: A network grown via the specialization model.

Many processes in the world involve a network of interactions. People clicking though websites and electricity moving through a power grid both travel through a complex network of links and hubs. The underlying structure of this network has an enormous impact on the process evolving over its links. The extent to which network structure helps or harms different processes is not well understood. This is in part because networks can be sensitive to small changes, i.e. adding or removing a single link can have large unpredictable impacts. This is illustrated by the complicated way road closures affect traffic or how a single line failure can wipe out an entire power grid. Other difficulties in understanding complex networks arise from the sheer scale of networks in question. The human brain, for example, is computationally intractable by virtue of its size.

What is a Complex Network? While there are many kinds of networks in the world, complex networks are characterized by a lack of predictable and uniform structure, yet still contain structural features not found in completely random networks. Complex networks may contain repeated motifs such as feed forward loops or repeated triangles and frequently have a long tailed degree distribution [1]. Most networks in the real world are complex.

The Importance of Studying Complex Networks: The study of complex networks is highly relevant and has potential to impact many fields. Consider the following examples:

The structure of social networks can be decisive in the spread of ideas, culture, conflict and disease. Understanding the impact of network structure could help us overcome cultural divides, bridge conflict and avoid epidemics. The human brain demonstrates a variety of structural motifs that appear to specialize in preforming different tasks [2]. Understanding the impact and function of these motifs could lead to exciting advances in artificial intelligence. Similarly, understanding the global state of signals moving though the brain could give insight into consciousness and thought. Network science may also provide solutions to species interaction problems in ecology, protein interaction problems in medicine and resource relocation problems in business, energy and transportation.

1.2 The Specialization Model

The specialization model of network growth provides a method for generating complex networks. It produces growth by splitting up component branches of a network in a structured way. The specialization model is useful because of its flexibility and theoretical guarantees. The model is flexible in that it only grows a chosen region of the network and leaves the rest of the network exactly the same. This allows for targeted expansion of certain groups of nodes and as such affords model users some control over which topological features are gained and which disappear as a network is specialized. In addition, the eigenvalues of a specialized network are the eigenvalues of the original network, together with the eigenvalues of each copied sub-network. This property guarantees that as a network grows according to this model, it will have the same spectral radius as the original network, if its edge weights are non-negative. Additionally, one can produce a targeted change in a network's eigenvalues by choosing appropriate subnetworks to specialize. Because of these features, the model is a useful tool for studying complex networks.

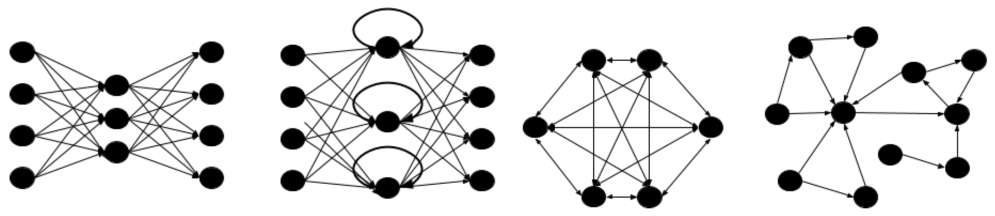


Figure 1.2: From left to right, an artificial neural network, a recurrent neural network, a Hopfield network and a complex network.

1.3 Networks in Learning Problems

Despite their prevalence in natural systems, complex networks have not found much traction in machine learning and artificial intelligence problems. Instead researchers have opted to use simpler network structures in their models. None of the mainstream models in machine learning use complex networks. Deep learning and neural network models are organized into layers. They use a feed-forward structure where each node sends it's output to every node in the next layer [3]. This type of structure contains no loops or feedback and exhibits a high degree of regularity. Because of this simple structure, neural networks are effectively an input to output mapping and do not leverage network structure for performance. These structures are mainly used in classification or pattern recognition tasks, where data points are mapped to labels.

Because other classes of problems require feedback and memory, machine learning researchers incorporated these features into their network structure. Recurrent neural networks and Long Short Term Memory networks allow interior node values to feed back on themselves, but only in a predetermined, structured way that does not fully leverage potential complex network structure [4].

There are examples of machine learning models that use network topology to solve learning problems such as Hopfield networks and Boltzman Machines, but theses models typically use an all-to-all edge topology and the use of these models is limited [5, 6].

A simplification of machine learning topologies can be seen in Figure 1.2. While the machine learning network topologies appear to be busier and denser that the pictured complex network (far right), their uniformity makes it much easier to understand how node states will evolve. This is not true for the pictured complex network where the mixture of cycles, hubs and feedback adds complexity to processes evolving over its nodes and edges. This in turn allows for node states to exhibit a variety of behavior as they evolve.

While artificial intelligence algorithms excel at pattern recognition, they lack an underlying reasoning structure that can extrapolate cause and effect. The complex structure of neuronal connections in the human brain suggest that complex networks could be a powerful tool for solving this type of learning problems. As things stand, no one has discovered how to leverage the power of complex networks for solving learning problems. The relationship between the structure of a complex network and learning processes remains an open question.

1.4 Reservoir Computing

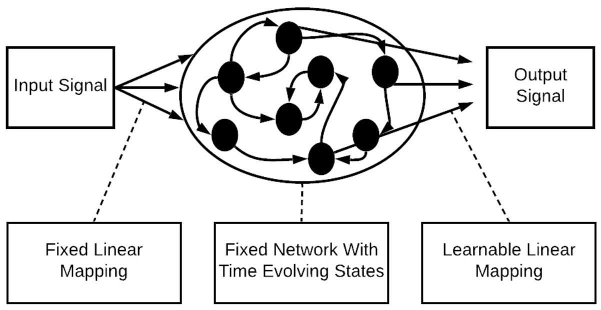


Figure 1.3: The structure of a reservoir computer

To investigate the connection between structure and function we turn to a machine learning model called a reservoir computer. In this thesis we use the model laid out in [7]. The model is trained on an input data stream and after it is fitted to the data, and is, to some extent, able to replicate the dynamics governing the training signal and predict a future trajectory. A reservoir computer consists of a random internal network or reservoir, and two linear mappings. Inside the reservoir, node values change according to a differential equation outlined in [7]. One of the mappings is randomly initialized and feeds the input signal to each reservoir node. The second mapping is learned. It attempts to map the node states back to the original input data stream. After training, the reservoir computer uses the second mapping to approximate the input data based on the internal node states and drive itself with the approximated data stream instead of with the true signal.

While in some ways, reservoir computers are not as powerful as other machine learning models, they are different from standard machine learning models in two ways that make them useful to us. First, because a reservoir computer does not rely on stochastic training methods, we know that the fit achieved by a reservoir computer is an optimal fit. This comes in contrast to stochastic gradient descent training method, where the final result is not guaranteed to be optimal. With stochastic gradient descent, it can be hard to say if the resulting trained model came about because of it's structure or because of hyper parameter optimization. Second, unlike other machine learning models, a reservoir computer's network can have any type of network topology. These two facts allow us to study the impact of complex network structure on a learning task.

Learning Chaotic Dynamical Systems It has been shown recently that reservoir computers with random graph topologies are capable of capturing key properties of dynamical systems such as the Lyapunov exponents [8, 9, 7]. In [7], reservoir computers are able replicate the Lorentz Equation's strange attractor for some time and capture it's Lyapunov exponents. The Lorentz system is chaotic, implying that it is impossible to predict the trajectory of any orbit indefinitely. We will apply reservoir computers to this same task and investigate the impact of network topology on success.

Chapter 2. Specialization

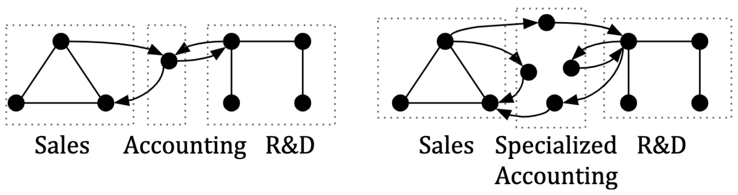


Figure 2.1: Specialization in Business

Real world networks exhibit a variety of growth mechanisms. The specialization model studies one specific mechanism—growth via copying subnetworks. As an example of growth by specialization, consider the firm in Figure 2.1. The firm consists of a sales team, an R&D team, and an overworked accountant. In this simplified example, the accountant interacts with the sales team and the R&D team, and coordinates certain tasks between them. The accountant can "specialize" in four different tasks:

(i) Working only with the sales team

(ii) Working only with the R&D team

(iii) Sending information from R&D to sales

(iv) Sending information from sales to R&D.

If the accountant specializes in one of these tasks and the company hires three new accountants to preform the remaining tasks, the network will be organized as pictured in Figure 2.1. Because this is a valid specialization according to the model, the network will maintain it's spectral radius and in this case, will add three new zeros to it's spectrum. Though this example has limitations, it does illustrate that the specialization mechanism happens in real world networks and demonstrates that it is a natural component of network growth.

In this chapter we will formulate the specialization model and state important results about the spectrum of specialized graphs. We will then extend the theory of specialization by characterizing it's effect on graph eigenvectors.

2.1 Notation and Definitions

In order to help analyze complex networks and define the specialization model, we state several important definitions. These definitions can be found in [11]. Each network that we consider can be represented as a graph. A graph, $G = (V, E, \omega)$ is composed of a vertex set V, and edge set E, and a mapping $\omega : E \to \mathbb{R}$ that associates each edge with a weight. If we were considering a real world network, the set V contains the objects in consideration, be they people in a social network or websites in the internet and E contains the interactions between these objects, i.e. friendship or hyperlinks. The function ω provides information about the strength of each edge.

For the graph $G = (V, E, \omega)$ we let $V = \{v_1, \ldots, v_n\}$, where v_i represents the ith vertex. We let e_{ij} denote the directed edge that begins at v_j and ends at v_i. In terms of the network, an edge e_{ij} is in the edge set E if the jth network element has some direct influence on or is linked to the ith network element in some way. In practice, edges can be directed or undirected and can be either weighted or unweighted. Since weighted directed graphs can be used to represent both unweighted and undirected graphs, without loss of generality we focus on weighted directed graphs. With this framework in place, we define several concepts that are needed to construct the specialization method and prove accompanying results.

Definition 2.1. (Paths, Cycles and Loops)

A *path* P in the graph $G = (V, E, \omega)$ is an ordered sequence of distinct vertices $P = v_1, \ldots, v_m$ in V such that $e_{i+1,i} \in E$ for $i = 1, \ldots, m-1$. If the first and last vertices v_1 and v_m are the same then P is a *cycle*. If it is the case that a cycle contains a single vertex then we call this cycle a *loop*.

Definition 2.2. (Strongly Connected Components)

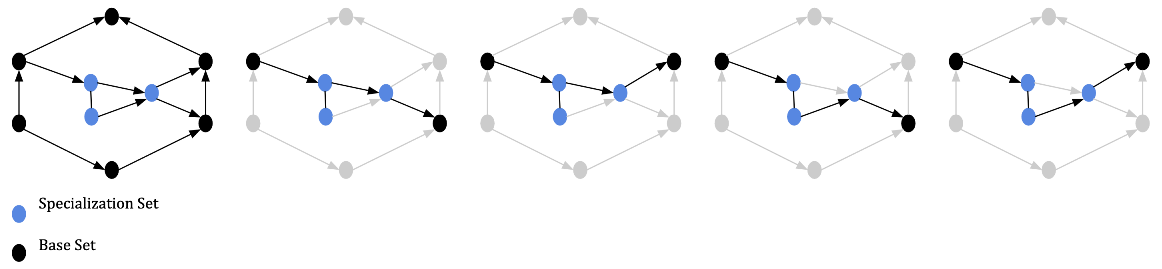


Figure 2.2: The component branch structure of a particular graph. The nodes in B are colored black. The two strongly connected components of $G|_{\overline{B}}$ are colored blue. Every component branch is visualized. (Undirected edges represent an edge in both directions.)

A graph $G = (V, E, \omega)$ is *strongly connected* if for any pair of vertices $v_i, v_j \in V$ there is a path from v_i to v_j or, in the trivial case, G consists of a single vertex. A *strongly connected component* of a graph G is a subgraph that is strongly connected and is maximal with respect to this property.

Definition 2.3. (Graph Restriction)

For a graph $G = (V, E, \omega)$ and a subset $B \subseteq V$ we let $G|_B$ denote the *restriction* of the graph G to the vertex set B, which is the subgraph of G on the vertex set B along with any edges of the graph G between the vertices in B. We let $\bar{B}$ denote the *complement* of B, so that the restriction $G|_{\bar{B}}$ is the graph restricted to the complement of vertices in B.

Definition 2.4. (Component Branches)

For a graph $G = (V, E, \omega)$ and vertex set $B \subseteq V$ let $C_1, \ldots, C_m$ be the strongly connected components of $G|_{\bar{B}}$. If there are edges $e_0, e_1, \ldots, e_m \in E$ and vertices $v_i, v_j \in B$ such that

(i) e_k is an edge from a vertex in C_k to a vertex in C_{k+1} for $k = 1, \ldots, m-1$;

(ii) e_0 is an edge from v_i to a vertex in C_1; and

(iii) e_m is an edge from a vertex in C_m to v_j, then we call the ordered set

$$\beta = \{v_i, e_0, C_1, e_1, C_2, \ldots, C_m, e_m, v_j\}$$

a *path of components* in G with respect to B. If $v_i = v_j$ then β is a *cycle of components.* We call the collection $\mathcal{B}_B(G)$ of these paths and cycles the *component branches* of G with respect to the base set of vertices B.

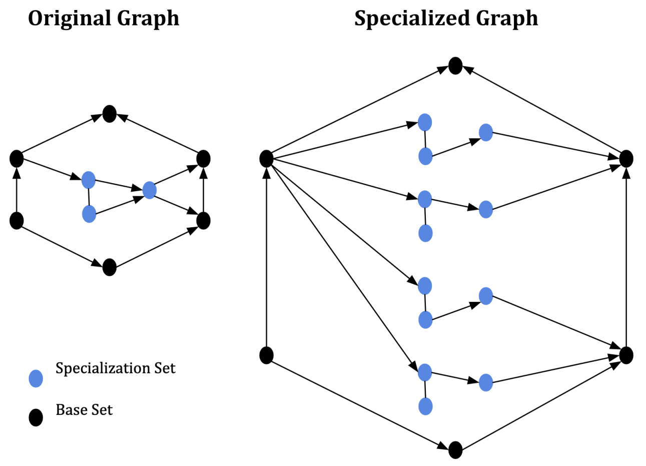


Figure 2.3: An example of graph specialization. Each of the component branches in Figure 2.2 are split apart and added to the specialized graph. (Undirected edges represent an edge in both directions)

The sequence of components $C_1, \dots, C_m$ in this definition can be empty in which case $m = 0$ and β is the trivial path $\beta = \{v_i, v_j\}$ or loop if $v_i = v_j$. It is worth emphasizing that each branch $\beta \in \mathcal{B}_B(G)$ is a subgraph of G. Consequently, the edges of β inherit the weights they had in G if G is weighted. If G is unweighted then its component branches are likewise unweighted. Consult Figure 2.2 for an example of component branch structure.

The process of specializing a graph $G = (V, E, \omega)$ involves choosing a set $B \subset V$, and splitting up the component branches of $G|_{\overline{B}}$. This can be formalized as follows:

Definition 2.5. (Graph Specialization)

Suppose $G = (V, E, \omega)$ and $B \subseteq V$. Let $\mathcal{S}_B(G)$ be the graph which consists of the component branches $\mathcal{B}_B(G) = \{\beta_1, \dots, \beta_\ell\}$ in which we *merge*, i.e. identify, each vertex $v_i \in B$ in any branch β_j with the same vertex v_i in any other branch β_k. We refer to the graph $\mathcal{S}_B(G)$ as the *specialization* of G over the *base* vertex set B.

Thus, a specialized graph is the original base vertices and all links between base nodes together with one copy of each component branch. Each copy is attached to the original base nodes where it was attached in the original graph. Consult Figures 2.3 and 2.2 for an example of specialization.

2.2 Spectrum of Specialized Graphs

For the purposes of this thesis we consider the spectrum of a graph to be the eigenvalues of it's weighted adjacency matrix . We define this as follows.

Definition 2.6. (Weighted Adjacency Matrix and Graph Spectrum) The *weighted adjacency matrix* of a graph $G = (V, E, \omega)$ is the matrix $A = \mathcal{A}(G)$ where

$$A_{ij} = \begin{cases} \omega(e_{ij}) & \text{if } e_{ij} \in E \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

If G is unweighted then each entry $A_{ij} = \omega(e_{ij}) = 1$ if $e_{ij} \in E$ and $A_{ij} = 0$ otherwise. The *eigenvalues* of the matrix $A \in \mathbb{R}^{n \times n}$ make up the graph's *spectrum*, which we denote by

$$\sigma(G) = \{\lambda \in \mathbb{C} : \det(A - \lambda I) = 0\},$$

where we consider $\sigma(G)$ to be a set that includes multiplicities, i.e. a *multiset*. The *spectral radius* of G is the spectral radius of $A = \mathcal{A}(G)$ denoted by

$$\rho(G) = \max\{|\lambda| : \lambda \in \sigma(A)\}.$$

With these definitions in place we may state the following theorem and corollary. The proof of these results can be found in [10].

Theorem 2.7. (Spectra of Specialized Graphs. Bunimovich, Smith and Webb) Let $G = (V, E, \omega)$, $B \subseteq V$, and let $C_1, \ldots, C_m$ be the strongly connected components of $G|_{\bar{B}}$. Then

$$sigma\big(\mathcal{S}_B(G)\big) = \sigma(G) \cup \sigma(C_1)^{n_1 - 1} \cup \sigma(C_2)^{n_2 - 1} \cup \cdots \cup \sigma(C_m)^{n_m - 1}$$

where n_i is the number of copies of the component C_i in $\mathcal{S}_B(G)$.

Simply stated, the spectrum of a specialized graph is equal to the spectrum of the original graph, together with the spectrum of each strongly connected component that is copied more than once. Because the eigenvalues of a graph play a large role in determining the dynamics of processes on the graph, understanding how the spectrum changes after specialization gives insight into how the dynamics will be change as the graph grows. The spectral radius of a network is instrumental to certain dynamic properties. Because many networks studied have positive edge weights, we state an important corollary:

Corollary 2.8. (Spectral Radius of Specialized Graphs. Bunimovich, Smith and Webb) Suppose $G = (V, E, \omega)$ has positive edge weights. Then for any $B \subseteq V$ the spectral radius $\rho(\mathcal{S}_B(G)) = \rho(G)$.

Certain types of dynamical systems with an underlying network structure can be specialized in a way analogous the the method described in this section. It is proven in [11] by way of this corollary, that such dynamical systems will maintain stability under mild conditions as they are specialized. Thus specialization gives insight into a key question in network theory: How does the topology of a graph influence dynamic processes on the network? By leveraging a structured change in spectrum, the specialization model demonstrates network growth that preserves a dynamic property: stability. This invariance gives us insight into the connection between topology changes and dynamics.

Thus, component branches appear to play an important role in the way a network processes information. It is possible that by separating a network into its component branches, we preserve an aspect of it's path and cycle structure that are important to dynamic processes.

2.3 Eigenvectors of Specialized Graphs

We turn now to a main focus of this thesis: the analysis of eigenvectors corresponding to specialized graphs. Just as the eigenvalues of a graph are preserved in a structured way, there is a clear structure in the preservation and creation of new eigenvector entries.

Here an *eigenvector* of a graph G corresponding to the eigenvalue $\lambda \in \sigma(G)$ is a vector $\mathbf{x}$ such that $\mathcal{A}(G)\mathbf{x} = \lambda\mathbf{x}$, in which case $(\lambda, \mathbf{x})$ an *eigenpair* of G. That is, the eigenvalues of

G are the eigenvalues of its adjacency matrix $A = \mathcal{A}(G)$.

To describe how specialization affects the eigenvectors of a network we require the following definition.

Definition 2.9. (Incoming and Outgoing Component Branches) For the graph $G = (V, E, \omega)$ and base $B \subseteq V$ let $\beta = \{v_i, e_0, C_1, e_1, C_2, \dots, C_m, e_m, v_j\} \in \mathcal{B}_B(G)$. We call the ordered set

$$In(\beta, C_k) = \{v_i, e_0, C_1, e_1, C_2, \dots, C_k\} \subset \beta$$

the *incoming branch* of β up to C_k. Similarly, we call the ordered set

$$Out(\beta, C_k) = \{C_k, e_k, C_{k+1}, \dots, C_m, e_m, v_j\} \subset \beta$$

the *outgoing branch* in β from C_k.

If Z is a strongly connected component of $G|_{\bar{B}}$ then $\ell \geq 0$ copies of Z will appear in the graph $\mathcal{S}_B(G)$. We denote the set of all such copies by $\mathcal{C}(Z) = \{Z_1, Z_2, \dots, Z_\ell\}$. Here, each Z_i is associated with the component Z in a specific branch $\beta_i \in \mathcal{B}_B(G)$. We say $Z_i, Z_j \in \mathcal{C}(Z)$ have the same incoming branch if $In(\beta_i, Z) = In(\beta_j, Z)$ and the same outgoing branch if $Out(\beta_i, Z) = Out(\beta_j, Z)$.

Definition 2.10. (Eigenvector Transfer Matrix) For a graph $G = (V, E, \omega)$ and a set $B \subset V$, let $\beta = \{v_i, e_0, C_1, e_1, C_2, \dots, C_m, e_m, v_j\} \in \mathcal{B}_B(G)$. Choose $k \in \{1, \cdots m\}$ and consider the graph $G|_B$ together with all nodes and edges in $In(\beta, C_k)$, the incoming branch of β up to C_k. The adjacency matrix of this graph is,

$$A = \begin{bmatrix} \underline{B} & & & & \\ Y_0 & \underline{C}_1 & & & \\ & Y_1 & \underline{C}_2 & & \\ & & \ddots & \ddots & \\ & & & Y_{k-1} & \underline{C}_k \end{bmatrix}$$

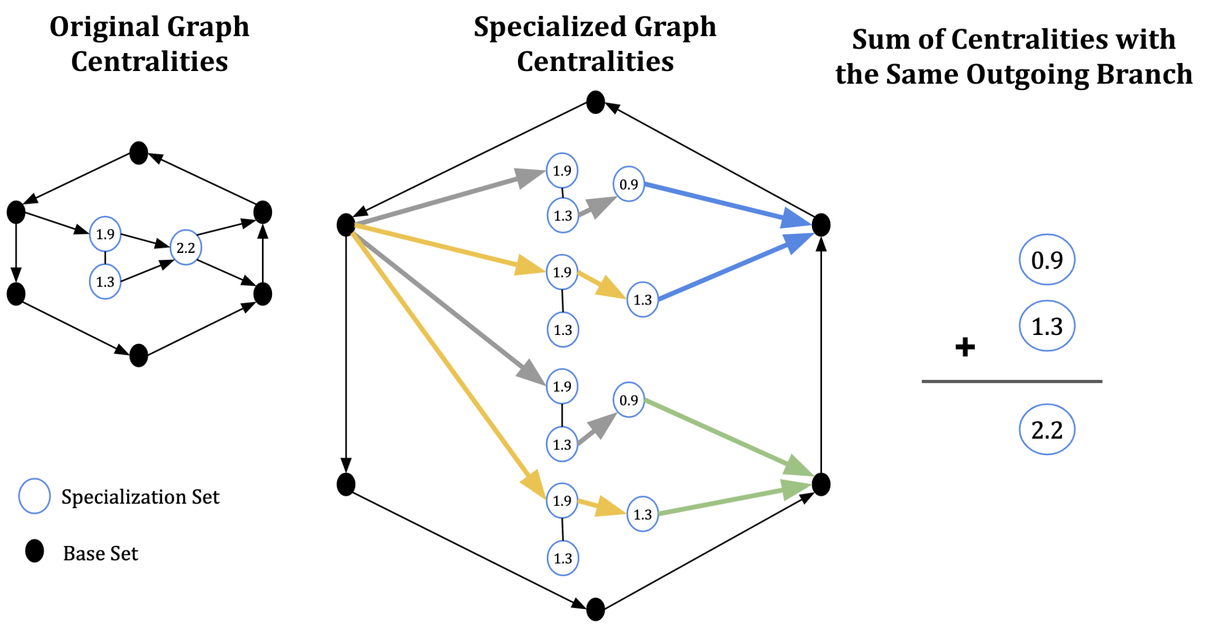

Original Graph Centralities
Specialized Graph Centralities
Sum of Centralities with the Same Outgoing Branch
1.9
1.3
2.2
0.9
+
Specialization Set
Base Set

Figure 2.4: Comparison of leading eigenvector entries (centrality) before and after specialization. (Undirected edges represent an edge in both directions.)

where $\underline{B} = \mathcal{A}(G|_B)$ and $\underline{C}_i = \mathcal{A}(C_i)$. Each Y_i contains a single non-zero entry corresponding to a single edge in the branch β. Using these submatrixes, we define

$$T(\beta, C_k, \lambda) = (\lambda I - \underline{C}_k)^{-1} Y_{k-1} (\lambda I - \underline{C}_{k-1})^{-1} Y_{k-2} \cdots (\lambda I - \underline{C}_1)^{-1} Y_0$$

and call $T(\beta, C_k, \lambda)$ the *eigenvector transfer matrix* of $In(\beta, C_k)$, where λ is a spectral parameter.

As we will see, this product of matrixes describes how eigenvectors of a graph are transformed to eigenvectors of a specialized version of that graph.

Before stating the main result, we outline some additional notation. If $\mathbf{x}$ is an eigenvector of a graph $G = (V, E, \omega)$ and $S \subseteq V$ is any subset of its vertex set, we let $\mathbf{x}_S$ denote the vector $\mathbf{x}$ restricted to the entries indexed by S. Similarly, by slight abuse of this notation, if Z is a subgraph of G with vertex set S we let $\mathbf{x}_Z = \mathbf{x}_S$. With this in place, we state the following theorem describing how the eigenvectors of a graph are affected by specialization.

Theorem 2.11. ***(Eigenvectors of Specialized Graphs. Bunimovich, Smith, P. and Webb)*** *Let $G = (V, E, \omega)$ be a graph, $B \subseteq V$ a base, and let Z be a strongly connected* *component of $\beta \in \mathcal{B}_B(G)$. If $(\lambda, \mathbf{u})$ is an eigenpair of G with $\lambda \notin \sigma(G|_{\bar{B}})$ then there is an eigenpair $(\lambda, \mathbf{v})$ of $\mathcal{S}_B(G)$ such that the following hold:*

(i) $\mathbf{u}_B = \mathbf{v}_B$.

(ii) For all $Z_i \in \mathcal{C}(Z)$ the eigenvector restriction

$$\mathbf{v}_{Z_i} = T(\beta, Z, \lambda)\mathbf{v}_B.$$

Hence, if $Z_i, Z_j \in \mathcal{C}(Z)$ have the same incoming branch then $\mathbf{v}_{Z_i} = \mathbf{v}_{Z_j}$.

(iii) For $Z_i \in \mathcal{C}(Z)$ let $\cup_{k=1}^{\ell}\{Z_k\}$ be the copies of Z that have the same outgoing branch as Z_i. Then

$$\mathbf{u}_Z = \sum_{k=1}^{\ell} \mathbf{v}_{Z_k} = \sum_{k=1}^{\ell} T(\beta, Z, \lambda)\mathbf{v}_B.$$

Figure 2.4 demonstrates the results of this theorem.

The proof of (i) is contained in [10]. We begin with a proof of part (ii)

Proof. Let Z be a strongly connected component of $G|_{\overline{B}}$ and let Z_i be a copy of Z in $\mathcal{S}_B(G)$. Then there exists a component branch β corresponding to Z_i, of the form $\beta = \{v_j, e_0, C_1, ..., C_k, e_k, Z, e_{k+1}C_{k+1}, ..., C_n\}$. Then, because specialization isolates each component branch, we can write the adjacency matrix A of $\mathcal{S}_B(G)$ in the form,

$$A = \begin{bmatrix} \mathcal{A}(In(\beta, Z)) & W \\ Y & X \end{bmatrix} = \begin{bmatrix} \underline{B} & & & & & & W \\ Y_0 & \underline{C}_1 & & & & & \\ & Y_1 & \underline{C}_2 & & & & \\ & & \ddots & \ddots & & & \\ & & & Y_{k-1} & \underline{C}_k & & \\ & & & & Y_k & \underline{Z}_i & \\ & & & & & Y_{k+1} & X \end{bmatrix}.$$

Here, $\underline{B} = \mathcal{A}(G|_B)$, $\underline{Z}_i = \mathcal{A}(Z_i)$ and for $1 \leq j \leq k$, $\underline{C}_j = \mathcal{A}(C_j)$. For $0 \leq j \leq k+1$, the matrix Y_j is a single entry matrix corresponding to the edge $e_j \in \beta$. The matrix X is the adjacency matrix for G restricted to nodes that are not in $In(\beta, Z)$, W contains information about edges from nodes represented in X to nodes in B.

Suppose $(\lambda, \mathbf{v})$ is an eigenpair of G, i.e. $A\mathbf{v} = \lambda\mathbf{v}$, such that $\lambda \notin \sigma(G|_{\bar{B}})$. As $\sigma(G|_{\bar{B}}) = \cup_{j=1}^{m}\sigma(C_i)$ where $C_1, C_2, \ldots, C_m$ are the strongly connected components of $G|_{\bar{B}}$ then λ is not an eigenvalue of any strongly connected component of β.

We may conformally partition $\mathbf{v}$ into $\mathbf{v} = [\mathbf{v}_B, \mathbf{v}_{C_1}, \cdots \mathbf{v}_{C_k}, \mathbf{v}_{Z_i}, \mathbf{v}_X]^T$ so that entries in each piece correspond with the appropriate sub-matrix of A. Then, applying the eigenvector equation produces,

$$A\mathbf{v} = \begin{bmatrix} \underline{B} & & & & & & W \\ Y_0 & \underline{C}_1 & & & & & \\ & Y_1 & \underline{C}_2 & & & & \\ & & \ddots & \ddots & & & \\ & & & Y_{k-1} & \underline{C}_k & & \\ & & & & Y_k & \underline{Z}_i & \\ & & & & & Y_{k+1} & X \end{bmatrix} \begin{bmatrix} \mathbf{v}_B \\ \mathbf{v}_{C1} \\ \mathbf{v}_{C2} \\ \vdots \\ \mathbf{v}_{C_k} \\ \mathbf{v}_{Z_i} \\ \mathbf{v}_X \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{v}_B \\ \mathbf{v}_{C_1} \\ \mathbf{v}_{C_2} \\ \vdots \\ \mathbf{v}_{C_k} \\ \mathbf{v}_{Z_i} \\ \mathbf{v}_X \end{bmatrix} = \lambda\mathbf{v}.$$

We solve for $\mathbf{v}_Z$ in terms of $\mathbf{v}_B$. Multiplying $\mathbf{v}$ by the appropriate block row of A gives $Y_k\mathbf{v}_{C_k} + \underline{Z}_i\mathbf{v}_{Z_i} = \mathbf{v}_{Z_i}$, implying that

$$\mathbf{v}_{Z_i} = \lambda\mathbf{v}_{Z_i}$$
$$\mathbf{v}_{Z_i} = (\lambda I - \underline{Z}_i)^{-1}Y_k\mathbf{v}_{C_k}.$$

In a similar manner for $2 \leq i \leq k$, we may solve for $\mathbf{v}_{C_i}$ in terms of $\mathbf{v}_{C_{i-1}}$ producing,

$$\mathbf{v}_{C_i} = (\lambda I - \underline{C}_i)^{-1}Y_{i-1}\mathbf{v}_{C_{i-1}}.$$

Combining this with the previous equation yields,

$$\mathbf{v}_{Z_i} = (\lambda I - \underline{Z}_i)^{-1}Y_k(\lambda I - \underline{C}_k)^{-1}Y_{k-1}\cdots Y_2(\lambda I - \underline{C}_2)^{-1}Y_1\mathbf{v}_{C_1}.$$

We solve for $\mathbf{v}_{C_1}$ in a similar manner and find that

$$\mathbf{v}_{C_1} = (\lambda I - \underline{C}_1)^{-1}Y_0\mathbf{v}_B.$$

Then,

$$\mathbf{v}_{Z_i} = (\lambda I - \underline{Z}_i)^{-1}Y_k(\lambda I - \underline{C}_k)^{-1}Y_{k-1}\cdots Y_1(\lambda I - \underline{C}_1)^{-1}Y_0\mathbf{v}_B.$$

Since Y_0,...,Y_k, C_1 ... C_k and Z_i are all the appropriate submatrixes of $\mathcal{A}(\,In(\beta, Z_i)\,)$, by definition of the eigenvector transfer matrix

$$\mathbf{v}_{Z_i} = (\lambda I - \underline{Z}_i)^{-1}Y_k(\lambda I - \underline{C}_k)^{-1}Y_{k-1}\cdots Y_1(\lambda I - \underline{C}_1)^{-1}Y_0\mathbf{v}_B = T(\beta, Z_i, \lambda)\mathbf{v}_B.$$

Note that each inverse in this equation exists since $\lambda \notin \sigma(Z_i)$ and $\lambda \notin \sigma(C_j)$ for $j = 1, 2, \ldots, k$. This completes the proof. $\square$

To prove part (iii) of Theorem 2.11 we use the following. Given a graph $H = (V, E, \omega)$ that is not strongly connected, let $S = \{S_1, S_2, ..., S_k\}$ be the strongly connected components of H. If there exist edges $e_1 \cdots e_{k-1}$ such that, e_j is an edge from S_j to S_{j+1} for $1 \leq j \leq k-1$, we call the ordered set $\alpha = \{S_1,\, e_1,\, S_2, ..., S_k\}$ a partial component branch from S_1 to S_k in H. We let $\mathcal{P}(S_1, S_k)$ denote the set of all partial component branches from S_1 to S_k in H. We let $\mathcal{P}(S_i, S_i)$ be the set containing only $\alpha = \{S_i\}$.

Definition 2.12. (Partial Eigenvector Transfer Matrix) Let $H = (V, E, \omega)$ be a graph that is not strongly connected and let S and T be strongly connected component subgraphs of H. For $\alpha = \{S, e_0, C_1, e_1, ...C_m, e_mT\} \in \mathcal{P}(S, T)$, let the adjacency matrix of α be

$$\begin{bmatrix} \underline{S} & & & & \\ Y_0 & \underline{C}_1 & & & \\ & Y_1 & \underline{C}_2 & & \\ & & \ddots & \ddots & \\ & & & Y_m & \underline{T} \end{bmatrix}$$

Where $\underline{S} = \mathcal{A}(S)$, $\underline{T} = \mathcal{A}(T)$ and $\underline{C}_i = \mathcal{A}(C_i)$ for $1 \leq i \leq m$. We call the matrix

$$P(\alpha, \lambda) = (\lambda I - T)^{-1}Y_m(\lambda I - C_m)^{-1}Y_{m-1}\cdots Y_1(\lambda I - T)^{-1}$$

the *partial eigenvector transfer matrix* of α, where λ is a spectral parameter.

Lemma 2.13. *Suppose $G = (V, E, \omega)$ is not strongly connected with strongly connected* *components S_1, S_2,...,S_k and*

$$A = \mathcal{A}(G) = \begin{bmatrix} \underline{S}_1 & & & \\ Y_{21} & \underline{S}_2 & & \\ \vdots & & \ddots & \\ Y_{k1} & \dots & Y_{kk-1} & \underline{S}_k \end{bmatrix}$$

where $\underline{S}_i = \mathcal{A}(S_i)$ for $1 \leq i \leq k$. If $\lambda \notin \sigma(G)$ then,

$$(\lambda I - A)^{-1} = \begin{bmatrix} X_{11} & & & \\ X_{21} & X_{22} & & \\ \vdots & \vdots & \ddots & \\ X_{kl} & X_{k2} & \cdots & X_{kk} \end{bmatrix}$$

where

$$X_{ij} = \sum_{\alpha \in \mathcal{P}(S_j, S_i)} P(\alpha, \lambda).$$

Proof. Since A is block lower triangular, $(\rho I - A)$ and $(\rho I - A)^{-1}$ are also block lower triangular. Also, since we can write,

$$(\rho I - A) = \begin{bmatrix} (\rho I - S_1) & & & \\ -Y_{21} & (\rho I - S_2) & & \\ \vdots & & \ddots & \\ -Y_{k1} & \dots & -Y_{kk-1} & (\rho I - S_k) \end{bmatrix}$$

we can write $(\rho I - A)^{-1}$ as,

$$(\rho I - A)^{-1} = \begin{bmatrix} X_{11} & & & \\ X_{21} & X_{22} & & \\ \vdots & \vdots & \ddots & \\ X_{kl} & X_{k2} & \cdots & X_{kk} \end{bmatrix}$$

where X_{ii} has the same dimensions as S_i for $1 \leq i \leq k$ and X_{ij} has the same dimensions as Y_{ij} when $1 \leq j < i \leq k$. It must then be the case that,

$$(\rho I - A)(\rho I - A)^{-1} = I_1 \oplus I_2 \oplus \ldots, \oplus I_k$$

where each I_i is the identity matrix with the same dimensions as S_i. Let $j \in \{1, 2, ..., k\}$. Then,

$$\begin{bmatrix} (\rho I - S_1) & & & \\ -Y_{21} & (\rho I - S_2) & & \\ \vdots & & \ddots & \\ -Y_{k1} & \ldots & -Y_{kk-1} & (\rho I - S_k) \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ X_{jj} \\ X_{(j+1)j} \\ \vdots \\ X_{kj} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ I_j \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Multiplying the jth row of $(\rho I - A)$ by the jth column of $(\rho I - A)^{-1}$ produces

$$(\rho I - S_j)X_{jj} = I_j$$

$$X_{jj} = (\rho I - S_j)^{-1} = P(\{S_j\}, \lambda).$$

Since $\mathcal{P}(S_j, S_j)$ only contains $\{S_j\}$ by definition, it follows that

$$X_{jj} = \sum_{\alpha \in \mathcal{P}(S_j, S_j)} P(\alpha, \lambda).$$

We will show by induction that

$$X_{ij} = \sum_{\alpha \in \mathcal{P}(S_j, S_i)} P(\alpha, \lambda)$$

when $i > j$. As a base case, consider $X_{(j+1)j}$. By multiplying row $(j+1)$ of $(\rho I - A)$ by the jth column of $(\rho I - A)^{-1}$, we obtain the equations

$$-Y_{(j+1)j}(\rho I - S_j)^{-1} + (\rho I - S_{(j+1)})X_{(j+1)j} = 0$$

$$X_{(j+1)j} = (\rho I - S_{(j+1)})^{-1} Y_{(j+1)j} (\rho I - S_j)^{-1}. \tag{2.2}$$

Let m be the number of nonzero entries in $Y_{(j+1)j}$. Then we can write

$$Y_{(j+1)j} = \sum_{k=1}^{m} Y_{(j+1)j}^{(k)}$$

where each $Y_{(j+1)j}^{(k)}$ has one non-zero entry and that entry is equal to a non-zero entry of $Y_{(j+1)j}$. Thus,

$$\begin{aligned} X_{(j+1)j} &= (\rho I - S_{j+1})^{-1} \sum_{k=1}^{m} Y_{(j+1)j}^{(k)} (\rho I - S_j)^{-1} \\ &= \sum_{k=1}^{m} (\rho I - S_{j+1})^{-1} Y_{(j+1)j}^{(k)} (\rho I - S_j)^{-1}. \end{aligned}$$

For each k, the block-diagonal matrix

$$\begin{bmatrix} S_j & \\ Y_{(j+1)j}^{(k)} & S_{j+1} \end{bmatrix}$$

is an adjacency matrix for $\alpha^{(k)} = \{S_j, e^{(k)}, S_{j+1}\}$ where $e^{(k)}$ is an edge from S_j to S_{j+1}. Then $\alpha^{(k)} \in \mathcal{P}(S_j, S_{j+1})$ and

$$X_{(j+1)j} = \sum_{k=1}^{m} P(\alpha^{(k)}).$$

Clearly, $\cup_{k=1}^{n} \{\alpha^{(k)}\} \subset \mathcal{P}(S_j, S_{j+1})$. We assert that $\cup_{k=1}^{n} \{\alpha^{(k)}\} = \mathcal{P}(S_j, S_{j+1})$. To verify this, let $\alpha \in \mathcal{P}(S_j, S_{j+1})$. Then $\alpha = \{S_j, e, S_{j+1}\}$ because if α contained any other strongly connected component S_l, it would imply that a path exists from S_j to S_l to S_{j+1} and because of the structure of A, it must be the case that $l < j$ or $j + 1 < l$. If an edge existed from S_j to S_l to S_{j+1} the matrix A would have an entry above the diagonal. This is a contradiction. Thus,

$$X_{(j+1)j} = \sum_{\alpha \in In(S_j, S_{j+1})} P(\alpha).$$

By the induction hypothesis assume that when $i < n$ we have

$$X_{(j+i)j} = \sum_{\alpha \in In(S_j, S_{j+i})} P(\alpha).$$

Consider $X_{(j+n)j}$. By multiplying the $j+n$th row of $(\rho I - A)$ by the jth column of $(\rho I - A)^{-1}$ we obtain the equations

$$-Y_{(j+n)j}(\rho I - S_j)^{-1} - Y_{(j+n)(j+1)}X_{(j+1)j} \cdots - Y_{(j+n)(j+n-1)}X_{(j+n-1)j} + (\rho I - S_{j+n})X_{(j+n)j} = 0$$

$$X_{(j+n)j} = (\rho I - S_{j+n})^{-1}Y_{(j+n)j}(\rho I - S_j)^{-1} + \sum_{i=1}^{n-1}(\rho I - S_{j+n})^{-1}Y_{(j+n)(j+i)}X_{(j+i)j} \tag{2.3}$$

As shown in the base case, the first term can be broken up into a sum of partial eigenvector transfer matrices. Let m_0 be the number of non-zero entries in $Y_{(j+n)j}$. If we define $Y^{(k)}_{(j+n)j}$ so that each $Y^{(k)}_{(j+1)j}$ has a single nonzero entry that is equal to a distinct non-zero entry of $Y_{(j+1)j}$ and

$$Y_{(j+n)j} = \sum_{k=1}^{m_0} Y^{(k)}_{(j+n)j}$$

then,

$$(\rho I - S_{j+n})^{-1}Y_{(j+n)j}(\rho I - S_j)^{-1} = \sum_{k=1}^{m_0}(\rho I - S_{j+n})^{-1}Y^{(k)}_{(j+n)j}(\rho I - S_j)^{-1}.$$

For each $1 \leq k \leq m_0$, $(\rho I - S_{j+n})^{-1}Y^{(k)}_{(j+n)j}(\rho I - S_j)^{-1}$ is the partial eigenvector transfer matrix for a distinct partial component branch α in $\mathcal{P}(S_{j+n}, S_j)$ that does not contain any strongly connected components except S_{j+n} and S_j. Let D_0 denote the set of all such branches. By definition of an adjacency matrix, D_0 contains one branch for each non zero entry in $Y_{(j+n)j}$. Thus,

$$(\rho I - S_{j+n})^{-1}Y_{(j+n)j}(\rho I - S_j)^{-1} = \sum_{\beta \in D_0} P(\beta, \lambda). \tag{2.4}$$

We consider the other terms in the sum (3). Let $1 \leq i \leq n-1$. By the induction hypothesis,

$$(\rho I - S_{j+n})^{-1}Y_{(j+n)(j+i)}X_{(j+i)j} = \textstyle\sum_{\alpha \in \mathcal{P}(S_j, S_{j+i})}(\rho I - S_{j+n})^{-1}Y_{(j+n)(j+i)}P(\alpha, \lambda).$$

Let m_i represent the number of nonzero entries in $Y_{(j+n)j+i}$. As before we write $Y_{(j+n)j+i}$ as a sum of m_i single entry matrices. $Y_{(j+n)j+i} = \sum_{k=1}^{m_i} Y^{(k)}_{(j+n)(j+i)}$. Then,

$$(\rho I - S_{j+n})^{-1} Y_{(j+n)(j+i)} X_{(j+i)j} = \sum_{\alpha \in \mathcal{P}(S_j, S_{j+i})} \sum_{k=1}^{m_i} (\rho I - S_{j+n})^{-1} Y^{(k)}_{(j+n)(j+i)} P(\alpha, \lambda).$$

It is clear that for each $\alpha \in \mathcal{P}(S_j, S_{j+i})$ and k, the term

$$(\rho I - S_{j+n})^{-1} Y^{(k)}_{(j+n)(j+i)} P(\alpha, \lambda)$$

is a partial eigenvector transfer matrix for some incoming branch $\gamma \in \mathcal{P}(S_j, S_j + n)$, because the matrix $Y^{(k)}_{(j+n)(j+i)}$ is non zero if and only if an edge exists from S_{j+i} to S_{j+n}. If $\mathcal{P}(S_j, S_{j+i})$ is non empty, there is a branch from S_j to S_{j+i}, implying that there must be a branch from S_j to S_{j+n} with partial eigenvector transfer matrix $(\rho I - S_{j+n})^{-1} Y^{(k)}_{(j+n)(j+i)} P(\alpha)$.

What we see here is that for a given i, the term

$$(\rho I - S_{j+n})^{-1} Y_{(j+n)(j+i)} X_{(j+i)j}$$

is equal to the sum of all centrality transfer matrices for the branches in $\mathcal{P}(S_j, S_{j+n})$ that pass through S_{j+i} immediately before reaching S_{j+n}. Let D_i denote the set of all such branches. Then,

$$(\rho I - S_{j+n})^{-1} Y_{(j+n)(j+i)} X_{(j+i)j} = \textstyle\sum_{\gamma \in D_i} P(\gamma, \lambda).$$

Putting (2.2), (2.3), and (2.4) together gives,

$$X_{(j+n)j} = \sum_{\beta \in D_0} P(\beta, \lambda) + \sum_{i=1}^{n-1} \sum_{\gamma \in D_i} P(\gamma, \lambda)$$

Let $D = \cup_{i=0}^{n-1} D_i$. Then

$$X_{(j+n)j} = \sum_{\alpha \in D} P(\alpha, \lambda)$$

Clearly, $D \subset \mathcal{P}(S_j, S_{j+n})$. We assert that $D = \mathcal{P}(S_j, S_{j+n})$. To show this, let $\alpha \in \mathcal{P}(S_j, S_{j+n})$. If α has only two components, then $\alpha = \{S_j, e, S_{j+n}\}$ for some edge e and

$\alpha \in D_0 \subset D$ by definition of D_0. If α has more then two components, then it has a second to last component, S_{j+i} where $1 \le i \le n-1$. By definition of D_i, $\alpha \in D_i$. Thus,

$$X_{j+n,j} = \sum_{\alpha \in \mathcal{P}(S_j, S_{j+n})} P(\alpha, \lambda)$$

This concludes the proof. □

We now give a proof of part (iii) of Theorem 2.11.

Proof. Since each Z_k is a copy of Z in $\mathcal{S}_B(G)$, each Z_k must correspond to a unique component branch. Let $D = \{\beta_1, ..., \beta_\ell\}$ be the set of such branches indexed so that β_k is the unique branch corresponding to Z_k for $1 \le k \le \ell$. Because each Z_k has the same outgoing branch, it follows that $In(\beta_j, Z) \neq In(\beta_k, Z)$ when $j \neq k$. Otherwise, there would exist $k \neq j$ such that $\beta_k = \beta_j$ which contradicts uniqueness of each β_k.

If $In(\beta_k, Z) = \{v_i, e_0, C_1, e_1, ..., e_{n-1}, C_m, e_m, Z\}$ define $F_k = \{C_1, C_2, ...C_m\}$ to be the set of all strongly connected components of $G|_{\overline{B}}$ in $In(\beta_k, Z)$. We let $F = \cup_{k=1}^{\ell} F_k$. Thus F is the set of all strongly connected components of $G|_B(Z)$ that appear before Z in some incoming branch of D.

We may order $F = \{C_1, C_2, ..., C_n, \}$ such that if $1 \le i < j \le n$ there are no paths from C_j to C_i. If no such ordering existed, it would imply for some $1 \le i < j \le n$, paths exist both from C_i to C_j and from C_j back to C_i. This implies that C_i and C_j must be part of the same strongly connected component which is a contradiction.

Thus, we can write the adjacency matrix $A = \mathcal{A}(G)$ of G in the form

$$A = \begin{bmatrix} \underline{B} & W_{BT} & W_{BZ} & W_{BX} \\ Y_{LB} & L & & \\ Y_{ZB} & Y_{ZL} & \underline{Z} & \\ Y_{XB} & Y_{XL} & Y_{XZ} & X \end{bmatrix}$$

where L is of the form,

$$L = \begin{bmatrix} \underline{C}_1 & & & \\ Y_{21} & \underline{C}_2 & & \\ \vdots & & \ddots & \\ Y_{k1} & \dots & Y_{kk-1} & \underline{C}_n \end{bmatrix}$$

and $\underline{B} = \mathcal{A}(G|_B)$, $\underline{Z} = \mathcal{A}(Z)$, $\underline{C}_i = \mathcal{A}(C_i)$ for $1 \leq i \leq m$. Since, λ is an eigenvalue of G, there is a vector $\mathbf{u}$ such that $A\mathbf{u} = \lambda\mathbf{u}$. We may partition $\mathbf{u}$ into $\mathbf{u} = [\mathbf{u}_B, \mathbf{u}_L, \mathbf{u}_Z, \mathbf{v}_X]^T$ so that the number of entries in each sub-vector corresponds with the size of the appropriate sub-matrix of A

We apply the eigenvector equation to solve for $\mathbf{u}_Z$. Given that

$$\begin{bmatrix} \underline{B} & W_{BT} & W_{BZ} & W_{BX} \\ Y_{LB} & L & & \\ Y_{ZB} & Y_{ZL} & \underline{Z} & \\ Y_{XB} & Y_{XL} & Y_{XZ} & X \end{bmatrix} \begin{bmatrix} \mathbf{u}_B \\ \mathbf{u}_L \\ \mathbf{u}_Z \\ \mathbf{u}_X \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{u}_B \\ \mathbf{u}_L \\ \mathbf{u}_Z \\ \mathbf{u}_X \end{bmatrix}$$

we have

$$Y_{ZB}\mathbf{u}_B + Y_{ZL}\mathbf{u}_L + \underline{Z}\mathbf{u}_Z = \lambda\mathbf{u}_Z$$
$$\mathbf{u}_Z = (\lambda I - \underline{Z})^{-1}Y_{ZB}\mathbf{u}_B + (\lambda I - \underline{Z})^{-1}Y_{ZL}\mathbf{u}_L.$$

Solving for $\mathbf{u}_L$ produces,

$$\mathbf{u}_L = (\lambda I - L)^{-1}Y_{LB}\mathbf{u}_B.$$

Thus,

$$\mathbf{u}_Z = \big(\,(\lambda I - \underline{Z})^{-1}Y_{ZB} + (\lambda I - \underline{Z})^{-1}Y_{ZL}(\lambda I - L)^{-1}Y_{LB}\,\big)\mathbf{u}_B.$$

We now show that,

$$\mathbf{u}_Z = \big(\,(\lambda I - \underline{Z})^{-1}Y_{ZB} + (\lambda I - \underline{Z})^{-1}Y_{ZL}(\lambda I - L)^{-1}Y_{LB}\,\big)\mathbf{u}_B = \textstyle\sum_{k=1}^{\ell} T(\beta_k, Z, \lambda)\mathbf{u}_B$$

by showing

$$(\lambda I - \underline{Z})^{-1}Y_{ZB} + (\lambda I - \underline{Z})^{-1}Y_{ZL}(\lambda I - L)^{-1}Y_{LB} = \sum_{k=1}^{\ell} T(\beta_k, Z, \lambda). \tag{2.5}$$

First, we consider $(\lambda I - \underline{Z})^{-1}Y_{ZB}$. Since every non-zero entry in Y_{ZB} corresponds to an edge from $G|_B$ to Z we let n_0 equal the number of non-zero entries in Y_{ZB} and write,

$$Y_{ZB} = \sum_{r=1}^{n_o} Y_{ZB}^{(r)}$$

where each $Y_{ZB}^{(r)}$ has exactly one non-zero entry and that entry is equal to a non-zero entry of Y_{ZB}. Thus,

$$(\lambda I - \underline{Z})^{-1}Y_{ZB} = \sum_{r=1}^{n_o} (\lambda I - \underline{Z})^{-1}Y_{ZB}^{(r)}.$$

We fix $r \in \{1, ..., n_0\}$ and consider $(\lambda I - \underline{Z})^{-1}Y_{ZB}^{(r)}$. The matrix $Y_{ZB}^{(r)}$ corresponds to exactly one edge from $G|_B$ to Z. Therefore, there must exist $\beta_r \in D$ such that $In(\beta_r, Z) = \{v, e, Z\}$ where $v \in B$ and e corresponds with the non-zero entry in $Y_{ZB}^{(r)}$. Hence,

$$(\lambda I - \underline{Z})^{-1}Y_{ZB}^{(r)} = T(\beta_r, Z, \lambda).$$

Furthermore, β_r must be the only branch in D that contains e. If not, there must be another $\beta_s \in D$ such that $In(\beta_s, Z) = \{v, e, Z\} = In(\beta_r, Z)$. Since β_r and β_s are in D, they have the same outgoing branch and it must be the case that $\beta_r = \beta_s$. This contradicts uniqueness of each β in D.

Thus, each $Y_{ZB}^{(r)}$ corresponds with exactly one $\beta_r \in D$. Furthermore, we see that $In(\beta_r, Z)$ does not contain any strongly connected components except Z, because the branch contains an edge directly from a node in B to Z. Thus, we may write it's adjacency matrix as follows:

$$\mathcal{A}(In(\beta_r, Z)) = \begin{bmatrix} \underline{B} & \\ Y_{ZB}^{(r)} & \underline{Z} \end{bmatrix}.$$

Then $(\lambda I - \underline{Z})^{-1}Y_{ZB}^{(r)}$ must the eigenvector transfer matrix of β_r with respect to Z.

Consider the set $D_0 = \{\beta_1, ...\beta_{n_0}\}$ of component branches corresponding to $\{Y_{ZB}^{(1)}, ..., Y_{ZB}^{(n_0)}\}$. In this case

$$(\lambda I - \underline{Z})^{-1}Y_{ZB} = \sum_{r=1}^{n_o} (\lambda I - \underline{Z})^{-1}Y_{ZB}^{(r)} = \sum_{\beta \in D_0} T(\beta, Z, \lambda). \qquad (2.6)$$

Note that for all $\beta \in D_0$, $In(\beta, Z)$ has no strongly connected components except Z. We assert that D_0 is the set of all branches in D that satisfy this property. To show this, note that if $\beta \in D$ and $In(\beta, Z) = \{v, e, Z\}$, then by definition of $\mathcal{A}(G)$, e must correspond to a non-zero entry in Y_{ZB} and therefore β corresponds to $Y_{ZB}^{(r)}$ for some $1 \leq r \leq n_0$. Then $\beta \in D_0$ and D_0 is the set of all $\beta \in D$ where $In(\beta, Z)$ contains no strongly connected components except Z.

Next we consider $(\lambda I - \underline{Z})^{-1} Y_{ZL} (\lambda I - L)^{-1} Y_{ZB}$ from equation (2.5). Once again, we write $Y_{ZL} = \sum_{s=1}^{n_1} Y_{ZL}^{(s)}$ and $Y_{LB} = \sum_{t=1}^{n_2} Y_{LB}^{(t)}$ as the sum of their non-zero entries.

By definition of an adjacency matrix, it must be the case that the set $\{Y_{ZL}^{(s)}\}_{s=1}^{n_1}$ is in a bijective correspondence with the edges from the components in $\{C_1, ...C_n\}$ to Z and the set $\{Y_{LB}^{(t)}\}_{t=1}^{n_2}$ is in a bijective correspondence with the edges from $G|_B$ to components in $\{C_1, ...C_n\}$. Thus we may let $\{f_s\}_{s=1}^{n_1}$ be the set of edges corresponding with $\{Y_{ZL}^{(s)}\}_{s=1}^{n_1}$ and $\{g_s\}_{t=1}^{n_1}$ be the set of edges corresponding with $\{Y_{LB}^{(s)}\}_{t=1}^{n_1}$.

We, therefore, have

$$(\lambda I - \underline{Z})^{-1} Y_{ZL} (\lambda I - L)^{-1} Y_{LB} = \sum_{s=1}^{n_1} \sum_{t=1}^{n_2} (\lambda I - \underline{Z})^{-1} Y_{ZL}^{(s)} (\lambda I - L)^{-1} Y_{LB}^{(t)}.$$

Fix, $s \in \{1, ..., n_1\}$ and $t \in \{1, ...n_2\}$ and consider, $Y_{ZL}^{(s)} (\lambda I - L)^{-1} Y_{LB}^{(t)}$. By definition $Y_{LB}^{(s)}$ represents an edge from $G|_B$ to some $C_i \in F$ and $Y_{ZL}^{(s)}$ represents an edge from some $C_j \in F$ to Z. Since there are no paths from C_j to C_i when $i > j$, it must be the case that $i \leq j$. This gives us information about the location of the non-zero entries in $Y_{LB}^{(s)}$ and $Y_{ZL}^{(s)}$. In particular, it must be the case that

$$\begin{bmatrix} Y_{LB}^{(t)} & L \\ & Y_{ZL}^{(s)} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ Y^{(t)} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} \underline{C}_1 & & & & & & \\ \vdots & \ddots & & & & & \\ Y_{i1} & & \underline{C}_i & & & & \\ \vdots & & & \ddots & & & \\ Y_{j1} & & & & \underline{C}_j & & \\ \vdots & & & & & \ddots & \\ Y_{k1} & \dots & Y_{ki} & \dots & Y_{kj} & \dots & \underline{C}_n \end{bmatrix} \\ & \begin{bmatrix} 0 & \dots & \dots & 0 & Y^{(s)} & 0 & 0 \end{bmatrix} \end{bmatrix}.$$

For some $Y^{(t)}$ and $Y^{(s)}$ that contain a single non-zero entry. This is because $Y_{LB}^{(t)}$ and $Y_{ZL}^{(s)}$ represent an edge from $G|_B$ to a component C_i and an edge from a component C_j to Z respectively. Thus, we conclude that all entries in $Y_{LB}^{(t)}$ and $Y_{ZL}^{(s)}$ that correspond to edges to or from components besides C_i and C_j respectively must be zero.

By lemma 2.13

$$(\lambda I - L)^{-1} = \begin{bmatrix} X_{11} & & & \\ X_{21} & X_{22} & & \\ \vdots & \vdots & \ddots & \\ X_{kl} & X_{k2} & \cdots & X_{kk} \end{bmatrix}$$

where

$$X_{ij} = \sum_{\alpha \in \mathcal{P}(C_j, C_i)} P(\alpha, \lambda).$$

Using this fact we simplify the expression $Y_{ZL}^{(s)}(\lambda I - L)^{-1}Y_{LB}^{(t)}$ to

$$\begin{bmatrix} 0 & \dots & 0 & Y^{(s)} & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} X_{11} & & & & & & \\ \vdots & \ddots & & & & & \\ X_{i1} & & X_{ii} & & & & \\ \vdots & & \vdots & \ddots & & & \\ X_{j1} & & X_{ji} & & X_{jj} & & \\ \vdots & & \vdots & & & \ddots & \\ X_{k1} & \dots & X_{ki} & \dots & X_{kj} & \dots & X_{nn} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ Y^{(t)} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

demonstrating that,

$$Y_{ZL}^{(s)}(\lambda I - L)^{-1}Y_{LB}^{(t)} = Y^{(s)}X_{ji}Y^{(t)} = \textstyle\sum_{\alpha \in \mathcal{P}(C_i, C_j)} Y^{(s)}P(\alpha, \lambda)Y^{(t)}.$$

Thus, for each s and t the term,

$$(\lambda I - \underline{Z})^{-1}Y_{ZL}^{(s)}(\lambda I - L)^{-1}Y_{LB}^{(t)} = \textstyle\sum_{\alpha \in \mathcal{P}(C_{i_t}, C_{j_s})}(\lambda I - Z)^{-1}Y^{(s)}P(\alpha, \lambda)Y^{(t)}$$

where $j_s, i_t \in \{1, ..., n\}$.

We now show for each $\alpha \in \mathcal{P}(C_{i_t}, C_{j_s})$ that

$$(\lambda I - \underline{Z})^{-1}Y^{(s)}P(\alpha, \lambda)Y^{(t)} = T(\beta, Z, \lambda)$$

where $\beta \in \mathcal{B}_B(G)$. Let $\alpha \in \mathcal{P}(C_{i_t}, C_{j_s})$. Then $\alpha = \{C_{\alpha_0}, e_1, C_{\alpha_1}, e_2, ...C_{\alpha_{N-1}}, e_N C_{\alpha_N}\}$ where $C_{\alpha_0} = C_{i_t}$, $C_{\alpha_N} = C_{j_s}$ and for each $1 \leq k \leq N$ we have $C_{\alpha_k} \in F$. Because $Y^{(s)}$ and $Y^{(t)}$ correspond to unique edges from C_{j_s} to Z and from $G|_B$ to C_{i_t}, respectively, there is a unique component branch $\beta \in \mathcal{B}_B(G)$ such that

$$\beta = \{v, g_t, C_{\alpha_0}, e_1, C_{\alpha_1}, e_2, ...C_{\alpha_{N-1}}, e_N, C_{\alpha_N}, f_s, Z, ...u\}$$

for some $v, u \in B$ and where $g_t, f_s \in E$ are edges corresponding to $Y^{(t)}$ and $Y^{(s)}$ respectively as defined previously. Thus, both

$$T(\beta, Z, \lambda) = (\lambda I - \underline{Z})^{-1}Y^{(s)}(\lambda I - \underline{C}_{\alpha_0})^{-1}Y_1(\lambda I - \underline{C}_{\alpha_1})^{-1}Y_2...Y_N(\lambda I - \underline{C}_{\alpha_N})^{-1}Y^{(t)}$$

$$T(\beta, Z, \lambda) = (\lambda I - \underline{Z})^{-1}Y^{(s)}P(\alpha, \lambda)Y^{(t)}$$

because,

$$P(\alpha, \lambda) = (\lambda I - \underline{C}_{\alpha_0})^{-1} Y_1 (\lambda I - \underline{C}_{\alpha_1})^{-1} Y_2 ... Y_N (\lambda I - \underline{C}_{\alpha_N})^{-1}.$$

We see given s, t and α that $(\lambda I - \underline{Z})^{-1} Y^{(s)} P(\alpha, \lambda) Y^{(t)}$ is equal to the eigenvector transfer matrix for some branch β that contains Z where the first edge in β is g_t and the edge before Z is f_s. Let D_{st} represent the set of all $\beta \in \mathcal{B}_B(G)$ where the first edge is g_t and the edge leading to Z is f_s. We have already shown that for each $\alpha \in \mathcal{P}(C_{i_t}, C_{j_s})$, there exists a $\beta \in D_{st}$ such that

$$(\lambda I - \underline{Z})^{-1} Y^{(s)} P(\alpha, \lambda) Y^{(t)} = T(\beta, Z, \lambda).$$

We now show that for each $\beta \in D_{st}$ there exists an $\alpha \in \mathcal{P}(C_{i_t}, C_{j_s})$ such that the above equation is true. Let $\beta \in D_{st}$. Then $\beta = \{v, g_t, C_{\beta_0}, e_1 ... e_M, C_{\beta_M}, f_s, Z ... u\}$ where for $1 \leq k \leq M$, $e_k \in E$ and $C_{\beta_k} \in F$ when $0 \leq k \leq M$. By definition of a component branch, each e_k is and edge from $C_{\beta_{k-1}}$ to C_{β_k} when $1 \leq k \leq M$. This implies that there is a partial component branch from C_{β_1} to C_{β_M} of the form, $\alpha = \{C_{\beta_0}, e_1 ... e_M, C_{\beta_M}\}$. Clearly $\alpha \in \mathcal{P}(C_{\beta_1}, C_{\beta_M})$. Hence, the adjacency matrix for α is of the form,

$$\mathcal{A}(\alpha) = \begin{bmatrix} \underline{C}_{\beta_0} & & & \\ Y_1 & \underline{C}_{\beta_1} & & \\ & \ddots & \ddots & \\ & & Y_m & \underline{C}_{\beta_M} \end{bmatrix}$$

and $P(\alpha, \lambda) = (\lambda I - \underline{C}_{\beta_M})^{-1} Y_M ... Y_1 (\lambda I - \underline{C}_{\beta_0})^{-1}$. Since the components and edges in α are in β, the adjacency matrix of $In(\beta, Z)$ can be written as

$$\mathcal{A}(In(\beta, Z)) = \begin{bmatrix} B & & & & & \\ Y^{(t)} & \underline{C}_{\beta_0} & & & & \\ & Y_1 & \underline{C}_{\beta_1} & & & \\ & & \ddots & \ddots & & \\ & & & Y_m & \underline{C}_{\beta_M} & \\ & & & & Y^{(s)} & Z \end{bmatrix}.$$

Thus, there exists an α such that $T(\beta, Z, \lambda) = (\lambda I - Z)^{-1} Y^{(s)} P(\alpha, \lambda) Y^{(t)}$. Thus, the sets $\mathcal{P}(C_{i_t}, C_{j_s}$ are in a bijective correspondence.

We know that for each $\alpha \in \mathcal{P}(C_{i_t}, C_{j_s})$ there is a $\beta \in D_{st}$ such that

$$(\lambda I - \underline{Z})^{-1} Y^{(s)} P(\alpha, \lambda) Y^{(t)} = T(\beta, Z, \lambda).$$

Using this fact, we may substitute each $(\lambda I - \underline{Z})^{-1} Y^{(s)} P(\alpha, \lambda) Y^{(t)}$ in the sum

$$\sum_{\alpha \in \mathcal{P}(C_{i_t}, C_{j_s})} (\lambda I - \underline{Z})^{-1} Y^{(s)} P(\alpha, \lambda) Y^{(t)}$$

for the corresponding $T(\beta, Z, \lambda)$, ($\beta \in D_{st}$ producing

$$\sum_{\alpha \in \mathcal{P}(C_{i_t}, C_{j_s})} (\lambda I - \underline{Z})^{-1} Y^{(s)} P(\alpha, \lambda) Y^{(t)} = \sum_{\beta \in D_{st}} T(\beta, Z, \lambda).$$

We know that every $\beta \in D_{st}$ is accounted for in the sum because of the correspondence demonstrated previously.

We have now shown that

$$\begin{aligned}
(\lambda I - \underline{Z})^{-1} Y_{ZL} (\lambda I - L)^{-1} Y_{LB} &= \sum_{s=1}^{n_1} \sum_{t=1}^{n_2} (\lambda I - \underline{Z})^{-1} Y_{ZL}^{(s)} (\lambda I - L)^{-1} Y_{LB}^{(t)} \\
&= \sum_{s=1}^{n_1} \sum_{t=1}^{n_2} \sum_{\alpha \in \mathcal{P}(C_{i_t}, C_{j_s})} (\lambda I - \underline{Z})^{-1} Y^{(s)} P(\alpha, \lambda) Y^{(t)} \\
&= \sum_{s=1}^{n_1} \sum_{t=1}^{n_2} \sum_{\beta \in D_{st}} T(\beta, Z, \lambda).
\end{aligned}$$

Since

$$\mathbf{u}_Z = \big((\lambda I - \underline{Z})^{-1} Y_{ZB} + (\lambda I - \underline{Z})^{-1} Y_{ZL} (\lambda I - L)^{-1} Y_{LB} \big) \mathbf{u}_B,$$

by equation (2.6) we have

$$\mathbf{u}_Z = \left(\sum_{\beta \in D_0} T(\beta, Z, \lambda) + \sum_{s=1}^{n_1} \sum_{t=1}^{n_2} \sum_{\beta \in D_{st}} T(\beta, Z, \lambda) \right) \mathbf{u}_B.$$

Finally we show that

$$D = D_0 \cup \left(\bigcup_{s=1}^{n_1} \bigcup_{t=1}^{n_2} D_{st} \right).$$

By definition, $D_0 \subset D$ and $D_{st} \subset D$ for all $1 \leq s \leq n1, 1 \leq t \leq n_2$. Then $D_0 \cup (\bigcup_{s=1}^{n_1} \bigcup_{t=1}^{n_2} D_{st}) \subset D$. Let $\beta \in D$. Then $In(\beta, Z)$ either contains a strongly connected component besides Z, or it does not. If it does not, $\beta \in D_0$ as shown previously. If it does, then the first edge in β must connect a vertex in $G|_B$ to a vertex in some component $C_i \in F$ and therefore correspond to f_t for some $1 \leq t \leq n_2$. Additionally the edge that appears prior to Z must correspond to g_s for some $1 \leq s \leq n_1$. Thus $\beta \in D_0 \cup (\bigcup_{s=1}^{n_1} \bigcup_{t=1}^{n_2} D_{st})$ and $D \subset D_0 \cup (\bigcup_{s=1}^{n_1} \bigcup_{t=1}^{n_2} D_{st})$. This produces

$$\mathbf{u}_Z = \sum_{\beta \in D} T(\beta, Z, \lambda)\mathbf{u}_B = \sum_{k=1}^{\ell} T(\beta_k, Z, \lambda)\mathbf{u}_B.$$

By part (ii) of Theorem 2.11 we have that $T(\beta_k, Z, \lambda)\mathbf{u}_B = \mathbf{v}_{Z_k}$ and

$$\mathbf{u}_Z = \sum_{k=1}^{\ell} \mathbf{v}_{Z_k}.$$

This completes the proof of part (iii). □

Chapter 3. Specialization and Machine Learning

3.1 Introduction

An important question in complex network network theory is: How does the structure of a network impact dynamics on the network? To investigate this question, we study the problem of chaotic attractor reconstruction with reservoir computers. We measure the effect of network topology on model performance and analyze the eigenvalues of the best preforming network topologies. We then design a model for producing a topology by specializing the most important nodes. Our designed networks out preform the other network topologies considered demonstrating the potential of the specialization model to connect structure with dynamics.

3.1.1 Network Structure and Dynamics. Many real world processes exhibit underlying network structure. It is clear that the structure of these networks plays an important role, but it is not always clear to what extent structure influences dynamics [1]. Because of the complexity in this problem, there is a shortage of theoretical results about dynamical systems on a large complex network.

Current theoretical work concerning dynamical systems on complex networks includes the study of synchronization, global stability, time scales and others [12, 13]. Much of this work maintains a strong reliance on spectral properties of the graph, or the eigenvalues of a network's adjacency matrix [14]. For example, spectral radius is a major factor in determining stability of a dynamical system on a network.

Many networks are modeled with undirected graphs thus gaining the advantage of a symmetric adjacency matrix and consequently real eigenvalues with a full set of orthogonal eigenvectors. However, directed graphs are ubiquitous in network science and their non-normal adjacency matrices make for dynamics that are much more interesting albiet more difficult to study [13]. We examine both directed and undirected graphs in our study of network structure.

3.1.2 Reservoir Computing.

In our investigation of structure and dynamics, we study reservoir computers because of their arbitrary internal network topology—a feature few machine learning models possess. Reservoir computers are the subject of much interest recently as they have been shown to capture dynamic properties of chaotic systems and recreate attractors [9]. What makes reservoir computers useful to us is their reliance on an internal network with time varying node states. This allows us to observe and measure the interplay of structure and dynamics in the system. We apply reservoir computers to the task of learning the Lorenz equations (as descibed on p. 36) and study the effect of network topology, and network spectrum on the ability of the model to learn.

We adopt the reservoir computer model used in [7]. The internal dynamics of the reservoir computer are governed by two equations. During training, the following equation is used:

$$\tfrac{d}{dt}\mathbf{r}(t) = \gamma\big[-\mathbf{r}(t) + \tanh\big(\mathbf{A}\mathbf{r}(t) + \sigma\mathbf{W}_{\text{in}}\mathbf{u}(t)\big)\big]$$

Here, $\mathbf{r}(t)$ is vector representing the state of all n nodes in the reservoir at time t. The matrix $\mathbf{A}$ is the $n \times n$ weighted adjacency matrix of the reservoir with $\mathbf{A}_{ij}$ representing the weight of the connection from node j to node i. The matrix $\mathbf{W}_{\text{in}}$ is a fixed $n \times m$ linear mapping that sends the m dimensional training signal $\mathbf{u}(t)$ to the n dimensional reservoir space. We can think of this mapping as sending a linear combination of the states of $\mathbf{u}$ to each reservoir node. The function tanh is applied elementwise. The variables γ and σ are scalar parameters.

To learn a particular signal, $\mathbf{u}(t)$, $t \in [0, T]$, a reservoir computer is trained, first by solving the above ode for $\mathbf{r}(t)$ when $t \in [0, T]$. Since $\mathbf{u}(t)$ is known, we can consider $\mathbf{r}(t)$ to be the reservoir state driven by the input $\mathbf{u}(t)$. We will refer to $\mathbf{r}(t)$ as the driven state.

Our goal is for an undriven reservoir computer to produce an output signal $\hat{\mathbf{u}}(t)$ such that $\hat{\mathbf{u}}(t) \approx \mathbf{u}(t)$. We accomplish this by solving for the mapping $\mathbf{W}_{\text{out}}$ such that

$$\mathbf{W}_{\text{out}} = \underset{\mathbf{W}\in\mathbb{R}^{m\times n}}{\operatorname{argmin}} \|\mathbf{W}\mathbf{r}(t) - \mathbf{u}(t)\|$$

For our purposes, $\hat{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}\mathbf{r}(t)$. Next, instead of driving the reservoir states with $\mathbf{u}(t)$ as done previously, we drive them with the approximated signal $\hat{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}\mathbf{r}(t)$ thus removing the system's dependancy on $\mathbf{u}(t)$. This produces the a new *trained* (dynamical) system,

$$\frac{d}{dt}\hat{\mathbf{r}}(t) = \gamma[-\hat{\mathbf{r}}(t) + \tanh\big(\mathbf{A}\hat{\mathbf{r}}(t) + \sigma\mathbf{W}_{\text{in}}\hat{\mathbf{u}}(t)\big)]$$
$$\frac{d}{dt}\hat{\mathbf{r}}(t) = \gamma[-\hat{\mathbf{r}}(t) + \tanh\big(\mathbf{A}\hat{\mathbf{r}}(t) + \sigma\mathbf{W}_{\text{in}}\mathbf{W}_{\text{out}}\hat{\mathbf{r}}(t)\big)]. \tag{3.1}$$

The trained reservoir can now be used to predict the trajectory of $\mathbf{u}(t)$ by solving 3.1 for $\hat{\mathbf{r}}(t)$ for $t \in (T, \infty)$ and then producing an approximated $\mathbf{u}(t), \hat{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}\hat{\mathbf{r}}(t)$ for $t \in (T, \infty)$.

It is not immediately clear that this process will produce a good approximation to $\mathbf{u}$. In fact, it is surprising that it is able to replicate chaos at all. See [7] for a discussion of the theoretical conditions for reservoir computer success.

3.1.3 Network Topologies.

The standard topology for a reservoir computer is an Erdős–Rényi random graph. In this thesis, we study the Small World model, the Preferential Attachment model and the Specialization model in addition to random graphs. All of these models, are undirected except for the Specialization model. For each of the undirected models, we explore directing their edges and removing a percentage of them at random. As we will see, this leads to networks with richer spectrum and better learning abilities.

Each of the undirected models before mentioned, accept two or three parameters and psuedo-randomly produce a network. The specialization model is different. Rather than producing a new network from parameters, the model acts on existing networks by copying components in a structured way. Previous work has shown that when about 10% of nodes in a random graph are repeatedly specialized, the random graph grows into a network with many real-world properties such as a right-skewed degree distribution, the small-world property and that density and clustering coefficient decrease as the network grows [1].

Besides this, we use the specialization model because of it's structured preservation of eigenvalues. When a network is specialized, it retains all of it's eigenvalues and gains the eigenvalues of any component that is copied. While this theoretical guarantee is useful, it comes at the cost of losing the ability to directly control the number of nodes in the new network. However, appropriate initial networks and some adjusting of parameters allows us to generate networks that are close to the correct size.

For the Barabasi Albert model we connected every new node to the graph with two edges (This parameter is called m in the original paper [15]. We use $m = 2$. For the Watts-Strogatz model we connected each node to its 5 nearest neighbors and use a rewiring probability of $p = 0.05$ [16]. We use $p = 2/n$ for the Erdős–Rény random graphs so that the number of edges scales linearly with the number of nodes n [17].

3.1.4 Node Importance Metric. To try to generate new types of complex network topology for reservoir computing, we borrow a tool from deep learning [18]. Our goal is to measure the importance of each node in the reservoir. The derivation proceeds as follows.

Let $\mathbf{r}(t)$ be a time varying vector of node states (each node state is a real number) from a trained reservoir and let $\mathbf{u}(t)$ be the desired output. We can measure the error in a reservoir computer's ability to reproduce the desired output at a particular point in time with

$$E\big(\mathbf{r}(t)\big) = \|\mathbf{W}_{\text{out}}\mathbf{r}(t) - \mathbf{u}(t)\|^2$$

Let ΔT be our time step size. Let

$$E_{ni}(x) = E\big(\mathbf{r}(n\Delta T),\, r_i(n\Delta T) = x\big) = \|\mathbf{W}_{\text{out}}\mathbf{r}^* - \mathbf{u}(t)\|^2$$

where $r_j^* = r_j(n\Delta T)$ when $j \neq i$ and $r_i^* = x$.

According to [18], the contribution of node i at time $n\Delta T$ can be approximated by

$$c_{ni} = |r_i(n\Delta T)\frac{dE_{ni}}{dx}\big(r_i(n\Delta T)\big)|$$

To measure the contribution of node i for all time, C_i we average over timesteps:

$$C_i = \frac{1}{N}\sum_{n=0}^{N} c_{ni}$$

Let $\mathbf{r}(t)$ be a time varying vector of node states (each node state is a real number) from a trained reservoir and let $\mathbf{u}(t)$ be the desired output. We can measure the error between the trained reservoir computer prediction and the true solution with,

Because we hold $r_j(t)$ $j \in \{1, ..., n\}$ constant for all $j \neq i$, the derivative,

$$\begin{aligned}
\frac{dE_{ni}}{dx}\big(r_i(n\Delta T)\big) &= \frac{\partial}{\partial r_i}\|\mathbf{W}_{out}\mathbf{r}(n\Delta T) - \mathbf{u}(n\Delta T)\|^2 \\
&= \frac{\partial}{\partial r_i}\sum_{k=0}^{m}\Big(\sum_{j=1}^{n}[W_{\text{out}}]_{kj}\, r_j - u_k\Big)^2 \\
&= \sum_{k=0}^{m}\frac{\partial}{\partial r_i}\Big(\sum_{j=1}^{n}[W_{\text{out}}]_{kj}\, r_j - u_k\Big)^2 \\
&= 2\sum_{k=0}^{m}\Big(\sum_{j=1}^{n}[W_{\text{out}}]_{kj}\, r_j - u_k\Big)[W_{\text{out}}]_{ki}
\end{aligned}$$

Putting this into matrix vector form gives,

$$\frac{dE_{ni}}{dx}\big(r_i(n\Delta T)\big) = 2\big[\mathbf{W}_{\text{out}}^T\big(\mathbf{W}_{\text{out}}\mathbf{r}(n\Delta T) - \mathbf{u}(n\Delta T)\big)\big]_i$$

Thus, we measure the effect of each node on the reservoir computer's accuracy by approximating the change in error when the node's signal is set to zero. Based on the derivation above, the score is computed for each node via

$$c_{ni} = |\, r_i(n\Delta T)\frac{dE_t}{dr}\big(r_i(n\Delta T)\,| = |\, 2\big[\mathbf{W}_{\text{out}}^T\big(\mathbf{W}_{\text{out}}\mathbf{r}(n\Delta T) - \mathbf{u}(n\Delta T)\big)\big]_i r_i(n\Delta T)\,|$$

then averaged over all time steps.

3.1.5 Reservoir Computer Implementation. For our implementation of a reservoir computer, we set $\gamma = 1$, $\sigma = 0.12$. Every network is initialized with uniform edge weights and then scaled so that the spectral radius is equal to 0.9. We initialize W_{in} with random values from -0.5 to 0.5 and solve the internal ode with an order four Runge-Kutta method built into the Python Scipy library. See [19] for details. The size of our reservoir computers varies from around 2000 nodes to 3500 nodes.

To solve for $\mathbf{W}_{\text{out}}$ we find the optimal solution to

$$\min_{\mathbf{W}}\sum_{k=0}^{N}\|\mathbf{W}\hat{\mathbf{r}}(k\Delta t) - \mathbf{u}(k\Delta t)\|^2 + \beta\|\mathbf{W}\|^2$$

by using a ridge regression with $\beta = 0.0001$ and $\Delta t = 0.01$. The complete reservoir computer code can be found in [20].

3.1.6 Measures of Model Performance.

To anaylyze the effect of network topology on model performance, we use two metrics. The first quantity we study is the average fit error. This quantity is computed via,

$$E = \frac{1}{k}\sum_{k=0}^{N} \|\mathbf{W}_{\text{out}}\hat{\mathbf{r}}(k\Delta t) - \mathbf{u}(k\Delta t)\|^2.$$

The second quantity studied, is the number of timesteps the reservoir computer could accurately predict. This quantity is computed by finding the smallest k such that,

$$\|\mathbf{W}_{\text{out}}\hat{\mathbf{r}}(k\Delta t) - \mathbf{u}(k\Delta t)\|^2 > \delta.$$

We call such a k, k_{pred}. In all of our experiments we set $\delta = 5$ because this was the tolerance where we could see a clear divergence between the reservoir computer orbit and the true orbit.

Some issues in computing these scores were that reservoir computers rely on random initialization of the $\mathbf{W}_{\text{in}}$ matrix and that scores vary depending on the particular orbit. To accurately assess the fit error and prediction length for a particular network, we test the network on 200 orbits. Each orbit is generated by drawing a random initial condition, from the uniform distribution on $[-20, 20] \times [-20, 20] \times [-20, 20]$. (This distribution was chosen because it produced a variety of different orbits.) For each orbit we reinitialize $\mathbf{W}_{\text{in}}$ to help eliminate the effect of the random initialization when performance scores are considered in aggregate.

We produce orbits from the Lorenz equations with standard parameter values in order to produce chaotic orbits on the strange attractor. i.e.

$$\begin{aligned}\tfrac{dx}{dt} &= 10(y - x)\\ \tfrac{dy}{dt} &= x(28 - z) - y\\ \tfrac{dz}{dt} &= xy - \tfrac{8}{3}z\end{aligned}$$

In the following section we will describe the results of our experiments.

3.2 Results

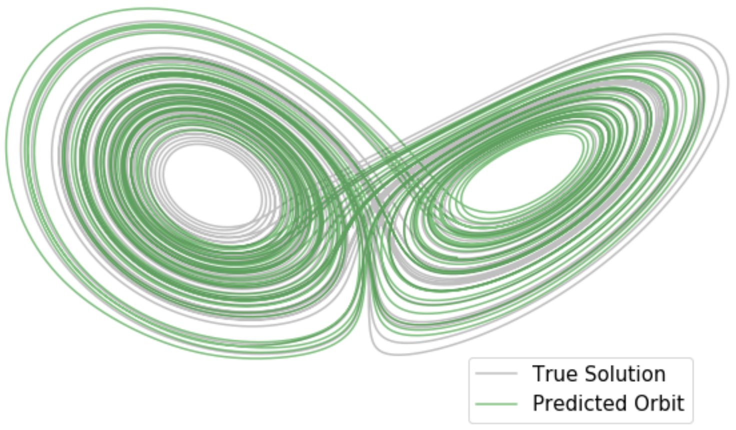


Figure 3.1: An orbit generated from the Lorentz equations (gray) overlayed with a reservoir computer's predicted orbit (green). A directed Barabasi-Albert topology with 75% of the edges removed was used for the internal reservoir.

3.2.1 Topology Comparison. We begin with the results of using each network topology as an internal reservoir. For each topology, 100 networks were generated and each network was trained on 200 random orbits. The reported prediction length and error values are the averages over all orbits and all networks. There was a high degree of variability in prediction length, with some reservoir computer initializations accurately predicting the given orbit for as many as 10,000 timesteps or as few as 100 timesteps. This variability is the reason each network was trained on a large number of orbits. After averaging, we saw a clear separation between each class of networks.

Of the topologies considered here, the winner for the best prediction length was the Barabasi-Albert model with 75% of the edges removed. The effect of removing edges from the Barabasi-Albert model was small and did not lead the the same magnitude of increased prediction length or decrease in error as it did for the other network topologies.

This attribute is interesting, especially in context of [21], where it is shown that scale free networks are extremely resistant to random attacks and maintain connectivity despite node removal. As a result of this, the best Barabasi-Albert network, pictured in Figure 3.2,

Erdős–Rényi			Barabsi-Albert			Watts-Strogatz			Specialized		
Per.	k_{pred}	E	Per.	k_{pred}	E	Per.	k_{pred}	E	Per.	k_{pred}	E
0%	2446	0.0565	0%	2589	0.0606	0%	1143	0.1695	0%	2376	0.0723
25%	2471	0.0511	25%	2582	0.0567	25%	1865	0.0975	10%	2411	0.0688
50%	2484	0.0465	50%	2608	0.0530	50%	2213	0.0630	20%	2418	0.0648
75%	2547	0.0434	75%	2609	0.0468	75%	2433	0.0471	30%	2459	0.0613

Table 3.1: Predictive ability and fit error of various toplogies as edges are removed uniformly at random. The first column, Per., for each topology is the percent of edges randomly removed from the graph. The variable k_{pred} is the number of timesteps predicted correctly and E represents fit error. Both of these quantities are described in section 3.1.6. The topologies ranked from best to worst preformers are: Barabasi-Albert, Erdős–Rényi, Specialized, Watts-Strogatz.

is highly connected. Despite losing 75% of its directed edges it retains a single tight knit component.

This stands in contrast the the Watts-Strogatz model. When 75% of it's edges are removed, the network deteriorates into many small components. This is pictured in Figure 3.3. While we may be tempted to assume that connectivity is a key to predictive power, the Watts-Strogatz model contradicts this by preforming *best* when it is disconnected. Because of this it seems unlikely that network connectivity plays a major role in the ability of a network to predict the Lorentz equations better than another. Therefore, because of the trends in this analysis, we expect that network features which are important for predicition will *appear* in the Watts-Strogatz model as edges are removed, and will be *preserved* in the Barabasi-Albert model as edges are removed.

The specialized networks are slightly less deterinistic than the other three topologies. Their structure is dependent on the initial graph, as well as on the number of nodes specialized. We preformed a single specialization of 6 random nodes on a random directed graph of 15 nodes where the probability of each directed edge existing was $p = 0.4$. These specialized networks display some fracturing as edges are removed, but retain a clear largest component. In response to edge removal, they exhibit mild increases in predictive ability. It should be noted that in our experiments, less edges were removed from the specialized networks since

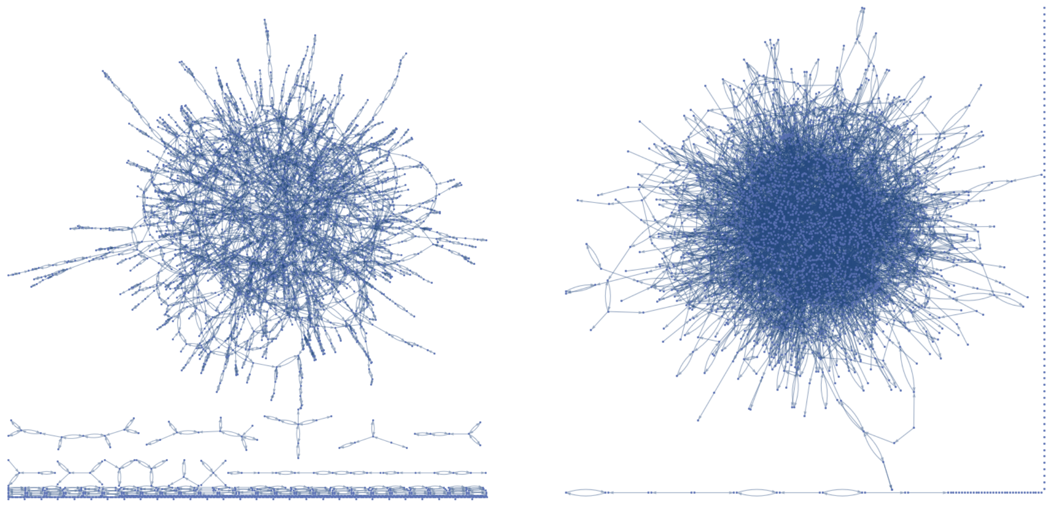


Figure 3.2: The top preforming Erdős–Rény network (left) and top preforming Barabasi Albert network (right).

they are not an undirected model. The specialized networks preform third best overall, but exhibited a wider variety of features and more variation in structure than the other network models. It is possible that a certain subset of the specialized networks, that contain key features may preform better than the rest of the group.

The Erdős–Rény topology exhibits the same mild increase in predictive ability as the specialization model when edges are removed. It also achieves the lowest fit error of any class of network. It is interesting that the model with the best fit does not achieve the best prediction. It may be the case that while the random graph can project it's node states onto the desired signal with the most accuracy, it's structure cannot replicate the chaotic dynamics for as long. It could also be overfitting to the input signal and not capturing the general trend as well.

3.2.2 Statistical Significance.

Prediction Length Differences in prediction length were compared for a particular model (as edges were removed) and the increase in prediction length after edge removal was found to be statistically significant with p-values $p < \alpha = 0.05$ for every model except the Barabasi-Albert model. The only statisically significant difference found within the

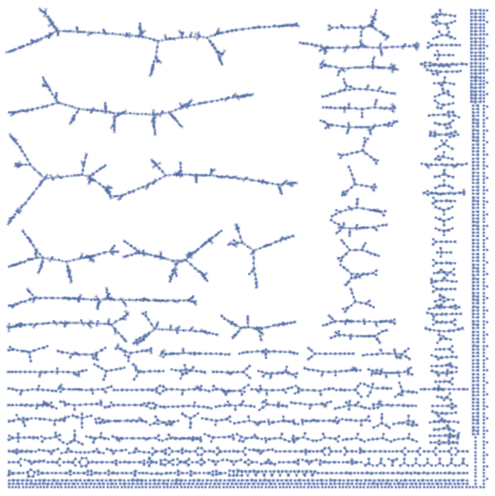
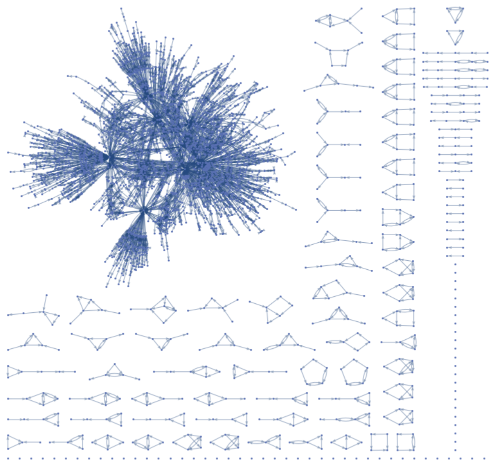



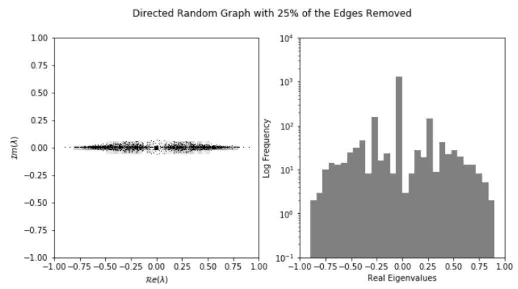
Figure 3.3: The top preforming Watts-Strogatz network (left) and top preforming specialized network (right).

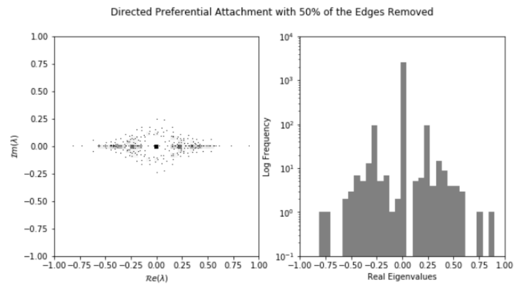
Barabasi-Albert topology classes was between networks with 25% of edges removed and 75% of edges removed.

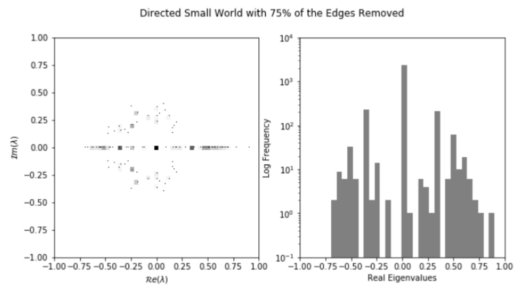
For each network model and percent of edges removed, the mean prediction length was also compared to the top preforming class (Barabasi-Albert with 75% of the edges removed) and the difference in means between each of them was statistically significant. Overall, this implies that all results were found to be statistically significant except the comparison Barabasi-Albert networks with themselves. This suggests that removing edges had little effect on predictive power of the Barabasi-Albert model.

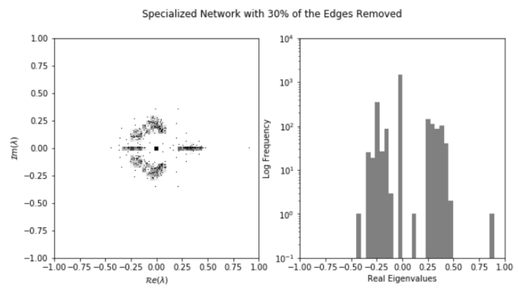
Error in Fit The decrease in fit error that occurred within each class of networks was found to be statistically significant as well with $\alpha = .05$. It is interesting that this decrease in error was clearly monotonic. Edge removal appears to have a strong impact on fit error. This is surprising because the current paradigm in machine learning is to incorporate as many edges as possible into a layered network and learn every edge parameter. In contrast to this, reservoir computers seem to benefit from sparsity. This highlights an important feature of complex networks, that they can exhibit more complexity with fewer edges.

3.2.3 Spectrum of Top Networks. Pictured on the right are the spectra of the top performers from each model. For a given model and percent of edges removed, the spectral plots were remarkably consistent. Key features such as peaks, skewness and shape of the imaginary eigenvalue spread were maintained within a group. The only exception was the specialization model. The eigenvalue plots for specialized networks had a variety of different features and shapes. A key feature of the Barabasi-Albert spectrum was symmetric peaks at the location of high multiplicity real eigenvalues. (It should be noted that the real spectrum is plotted on a log scale for frequency.) These peaks are visible when 25% of edges are removed and become more prominent as more edges are removed. Additionally, they seem to spread outward and move away from the origin. We see that the best preforming Watts-Strogatz model exhibits similar symmetric peaks, and though it is less symmetric, the best specialized network has symmetric peaks as well. This suggests that a symmetric spectrum with peaks in key places may be important for learning the Lorentz equations.



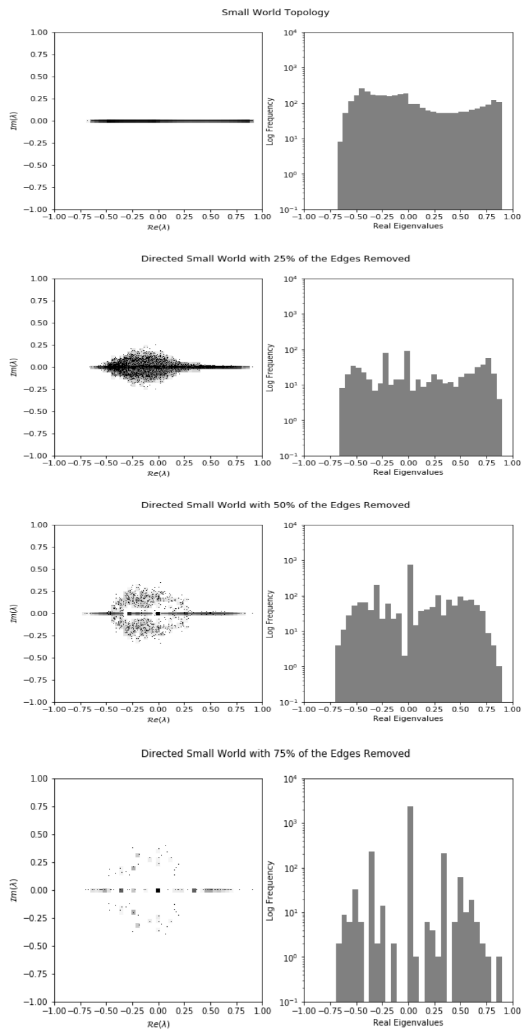






3.2.4 Edge Removal and Spectrum.

As edges were removed from a network, the distribution of eigenvalues would spread out into the complex plane. However, in the case of the Watts-Strogatz model and the Barabasi-Albert model, this did not occur uniformly. The appearance of complex eigenvalues seemed related to critical real eigenvalues. The complex eigenvalues appeared near these critical points initially, and then move outward. The critical real eigenvalues would persist as edges were removed and become peaks in the distribution. This robustness to edge removal suggests that the eigenvalues with high multiplicity are related to an inherent feature of the network. It is also interesting to note that when 75% of edges are removed from the Watts-Strogatz model, it's spectrum begins to resemble that of the Barabasi-Albert model, and preforms better than any other amount of edge removal. There appears to be a connection between symmetry, high multiplicities and predictive power.



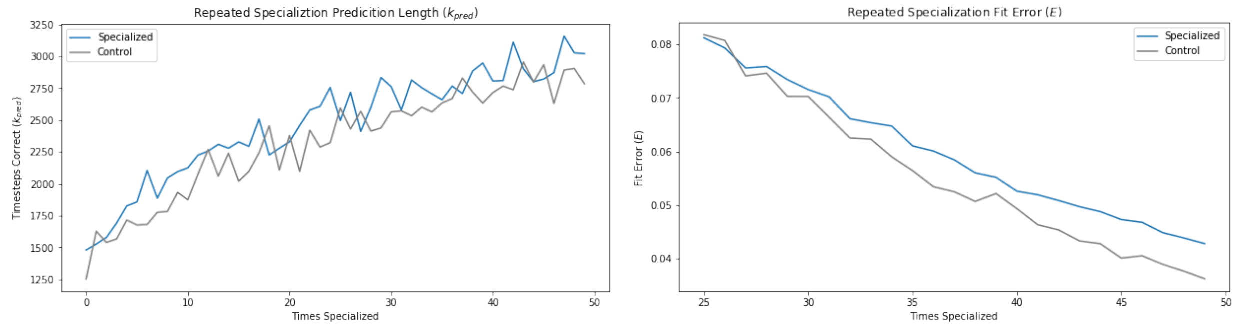


Figure 3.4: Performance of repeated specialization. As the top preforming nodes were repeatedly specialized, the generated networks outperformed a random control with the same sparsity and also outperformed all other topologies studied. The random control, however, exhibited lower fit error.

3.2.5 Iterative Network Growth via Specialization. In an attempt to out-perform the previously discussed topologies, we devise a scheme for iteratively growing complex networks with the specialization model. The scheme begins with a random directed graph on $n = 30$ nodes where each directed edges exists with probability $p = .12$. A random orbit from the Lorenz Equations is generated. Then the following steps are repeated 50 times: Initialize a reservoir computer with the current network topology and train it on the orbit. Identify the 3 most important nodes via the process outlined in section 3.1.4 in the network and specialize them. Set the current topology to be the specialized graph. Repeat.

At each iteration, we record the current prediction length (k_{pred}) and the fit error E. As a control, we also initialize a reservoir computer with a random digraph that has the same sparsity as the specialized network and record it's fit error and prediction length. (Random digraphs are the directed version of Erdős–Rény random graphs. Each directed edge exists with probability p.) Figure 3.4, that illustrates the average prediction length and error for 100 networks generated via this process.

After 50 iterations, the generated networks predict 3021 timesteps correctly on average. This was compared to prediction lengths of Barabasi-Albert topologies and found to be statistically significant with a p-value of $p = 0.01$. The average fit error after 50 iterations was 0.042 and this was also statistically significant compared to the Barabasi-Albert fit error with $p = 1 \times 10^{-8}$. It was not however statistically significant compared to the error in

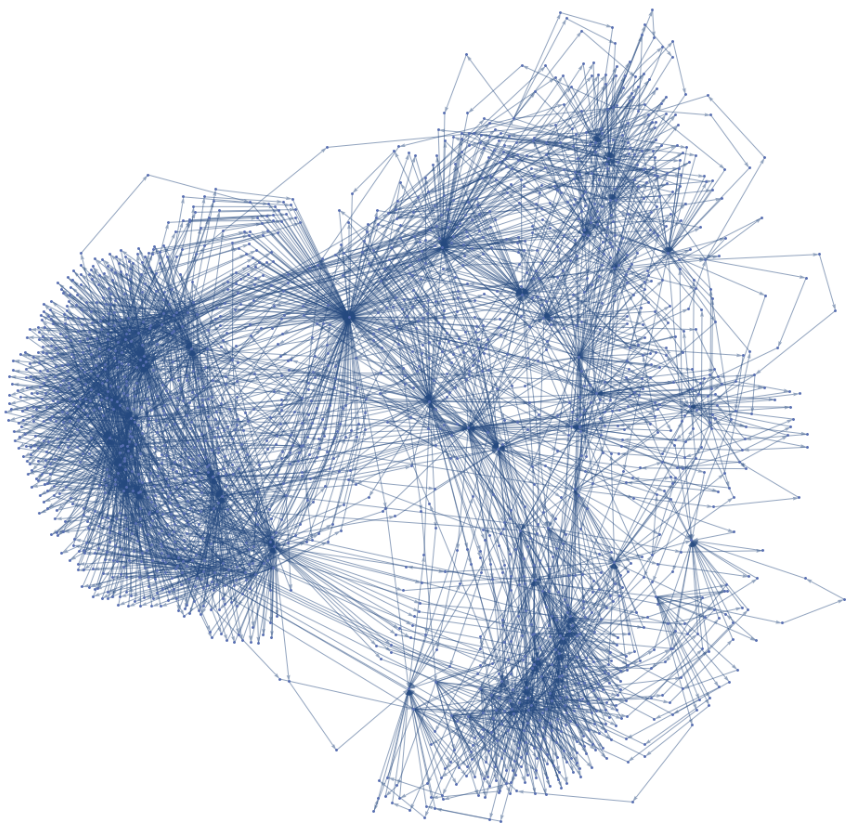


Figure 3.5: Network generated by repeated specialization

Erdős–Rény graphs with 75% of the edges removed.

Thus, the iteratively generated networks outperform the other classes of networks investigated in all but one category. Their structure was extremely sparse, containing long branches of nodes with only one edge in and one edge out. Additionally, they contained large hubs and clusters of single node loops. A network generated in this manner is pictured in Figure 3.5. The repeated motif of alternating single node, single edge chains contributes a huge amount of zeros to the spectrum and thus, the spectral plot is quite bland.

While these networks performed well, it is important to remember that they were generated based on a single orbit, and in a way, were tailored to that orbit. The other topologies in question were tested on arbitrary orbits. Though we know the superiority of this method on a single orbit, we do not know how well the networks generalize to arbitrary orbits.

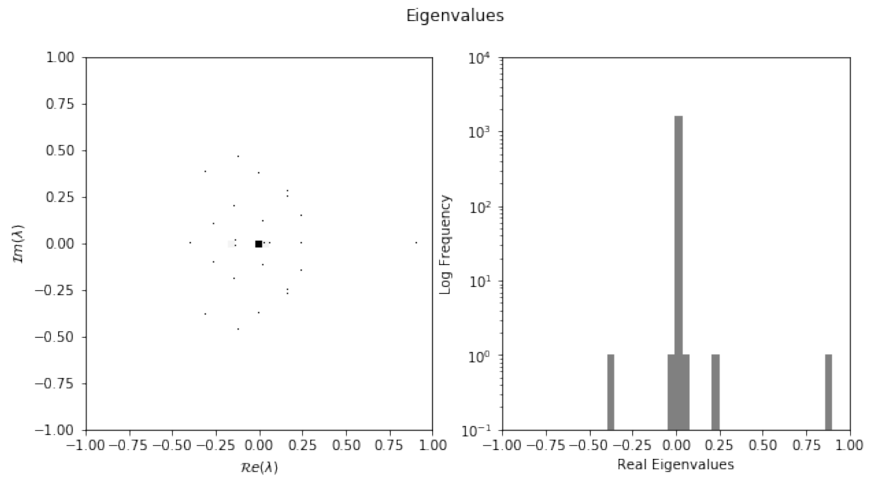


Figure 3.6: Eigenvalues of the network in Figure 3.5.

3.3 Conclusion

In this thesis, we have proven a result about the eigenvectors of specialized graphs and studied the effect of network topology on reservoir computers.

We have examined several different network topologies and compared their abilities to predict the Lorenz equations accurately, concluding that for an arbitrary orbit, a Barabasi-Albert topology with 75% of the (directed) edges removed preformed best with about 2600 accurate timesteps. We have also shown that by using a targeted approach to a specific orbit, networks can be generated that predict about 3000 time steps accurately. We analyzed the spectrum of each topology and found the existence of peaks in the eigenvalue distribution that seemed correlated with performance.

Most importantly, we have shown that contrary to popular belief, maximizing connectivity does not always increase performance, but rather, that performance may be improved by decreasing connectivity.

Bibliography

[1] M.E.J. Newman. The structure and function of complex networks. *Computer Physics Communications*, 147:40–45, 03 2003.

[2] Olaf Sporns and Rolf Kötter. Motifs in brain networks. *PLoS biology*, 2, 12 2004.

[3] David Rumelhart, Geoffrey Hinton, and Ronald Williams. *Learning Internal Representation by Error Propagation*, volume Vol. 1. 01 1986.

[4] Mikael Bodén. A guide to recurrent neural networks and backpropagation. 12 2001.

[5] John Hopfield and D Tank. Neural computation of decisions in optimization problems. *Biological cybernetics*, 52:141–52, 02 1985.

[6] Dave Ackley, GE Hinton, and Terrence Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science: A Multidisciplinary Journal*, 9:147–169, 01 1985.

[7] Zhixin Lu, Brian Hunt, and Edward Ott. Attractor reconstruction by machine learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28, 06 2018.

[8] Jaideep Pathak, Alexander Wikner, Rebeckah Fussell, Sarthak Chandra, Brian Hunt, Michelle Girvan, and Edward Ott. Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28, 03 2018.

[9] Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical Review Letters*, 120, 01 2018.

[10] Dallas Smith. *Network Specializations, Symmetries, and Spectral Properties*. PhD thesis, Brigham Young University, 2018.

[11] L. Bunimovich, D. Smith, B. Webb, and D. Passey. Spectral and dynamic consequences of network specialization. 06 2020.

[12] Young Cho, Takashi Nishikawa, and Adilson Motter. Stable chimeras and independently synchronizable clusters. *Physical Review Letters*, 119, 07 2017.

[13] Malbor Asllani, Renaud Lambiotte, and Timoteo Carletti. Structure and dynamical behavior of non-normal networks. *ScienceAdvances*, 4(12), dec 2018.

[14] P Mieghem. *Graph Spectra for Complex Networks*. 01 2011.

[15] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[16] Duncan Watts and Steven Strogatz. *Collective dynamics of 'small-world' networks*. 12 2011.

[17] M. Newman. Random graphs as models of networks. *Santa Fe Institute, Working Papers*, 02 2002.

[18] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. 11 2016.

[19] J. R. Dormand and P. J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6:19–26, mar 1980.

[20] D. Passey. Reservoir specialization. `https://github.com/djpasseyjr/ReservoirSpecialization`, 2020.

[21] R. Albert, H. Jeong, and A.L. Barabasi. Error and attack tolerance of complex networks. *Nature*, 406:378–382, 2000.