An Interactive Tool to Investigate the Inference Performance of Network Dynamics from Data

Veenadhar Katragadda,* Marigona Bokshi,† Yan Wan‡ and Xinrong Li§

University of North Texas, Denton, TX, 76201

Network structure plays a significant role in determining the performance of network inference tasks. In this paper, we develop an interactive tool to study the dependence of network topology on estimation performance. The tool allows end-users to easily create and modify network structures and observe the performance of pole estimation measured by Cramer-Rao bounds. The tool also automatically suggests the best measurement locations to maximize estimation performance, and thus finds its broad applications on the optimal design of data collection experiments. We end the paper with a series of theoretical results that explicitly connect subsets of network structures with inference performance.

I. Introduction

Systems have become structurally more complex and larger in size in a variety of applications, such as autonomous network control, transportation management, sensor networking, and power network surveillance, among others. Many of these systems are characterized by interconnected network topologies. While extensive advances have been achieved in the analysis, evaluation, and design of these networked systems, the inference of network dynamics/structure from data has just started to gain attention [5,11,16]. This growing interest is motivated by the fact that network inference tasks are critical for a range of design and control problems in networks. For instance, epidemic control relies on the inference of propagation dynamics from noisy outbreak data [19]. Genetic engineering is contingent upon the identification of gene regulatory networks from experimental microarray data [4,11,12]. Securing power networks is concerned with designing power network structure that is difficult for malicious attackers to estimate the configuration and dynamics of power systems from local observations (so as to disturb the network or to steal power) [9]. These network inference applications have motivated two critical research needs: 1) designing smart data-collection experiments to better understand a network’s behavior, and 2) securing a networked system effectively against malicious probing and attacks. We note that from dynamical systems point of view, security is concerned with preventing attackers from using observation data to detect a system’s internal dynamics [20].

We find that the key step to address the above two needs is to understand the pivotal factors impacting the performance of inference tasks in networks. This knowledge can then be utilized to design strategies to enhance security or surveillance capability. Previous efforts suggested that network topology has significant impact on the performance of inference tasks from noisy observation data [18]. In particular, estimation performance of system pole locations—as critical indicators of network dynamics—can be closely tied to network structure. However, mathematically capturing this structure dependence is nontrivial for general networks, as this problem is fundamentally concerned with directly relating eigenvectors, or equivalently zero-locations with network structure characteristics.

*Master. Student, Department of Electrical Engineering
†U.Grad. Student, Department of Electrical Engineering.
‡Assistant Professor, Department of Electrical Engineering, and AIAA Member.
§Associate Professor, Department of Electrical Engineering.
Computer-aided tools can significantly complement analytical limitations by providing automated analysis and design. The automated solutions can in turn provide rich insights to analytical discoveries. As such, in this paper, we construct an interactive graphical user interface (GUI) tool that allows end users to easily modify a network’s structure and observe the impact of such structural change on inference performance. The tool is available to download at [1] with the interface shown in Figure 1. This stand-alone GUI tool has the following features:

1) End users can choose between reading the network structure from a file or directly drawing the network’s graph structure in a plot area.

2) End users can select measurement noise levels and actuation and measurement locations to set up the network inference tasks. The tool then automatically displays the inference performance expressed in terms of Cramer-Rao bounds (CRB).

3) For any designated network structure, the tool displays the best measurement locations to observe network dynamics (e.g., with the tightest CRB). This information helps with the smart design of experiments to maximize inference performance.

4) End users can instantaneously observe the impact of structure change on inference performance, by directly modifying the graph structure through e.g., adding/removing nodes/edges, and dragging nodes.

In this paper, we illustrate the functionalities of the tool, the theoretical background and algorithms for constructing the tool, and the analytical development of results obtained using the tool. In particular, we focus on the inference of system pole locations from noisy measurement data, and utilize the tool to help with: 1) evaluating the impact of network structural change on inference performance measured by CRB, 2) identifying classes of network structures whose structural characteristics can be explicitly linked to inference performance, and 3) designing measurement locations to maximize inference performance. Building upon our previous development [18], we develop these results using meshed control theory, graph theory, and estimation theory. We envision that this interactive tool and resulting theoretical findings will provide us rich insights into structure-exploiting experimental design for smart data collection and strategies to enhance security.
The rest of this paper is organized as follows. In Section II, we present the functional description of the GUI. Section III presents the theoretical background behind the implementation of the tool. Section IV contains examples of theoretical results obtained using the tool. In particular, the tool allows us to identify subclasses of graphs, for which theoretical proofs on the explicit relationship between network structure and estimation performance are provided. Finally, Section V includes a brief conclusion and future work.

II. Functional Description of the Interactive Tool

In this section, we describe in detail the functionality and interface of the GUI tool. As this tool is designed to assist with studying a network structure’s impact on estimation performance, we integrate the following features into the tool: 1) The graph capturing network structure is displayed to allow an intuitive and visualized observation; 2) End users can directly modify the network structure through dragging nodes or adding/deleting edges/nodes and observe the change of estimation performance automatically; and 3) The best measurement location is automatically generated for any designated actuation locations.

The interface of the tool is composed of three parts: 1) the Options panel where end users can set up the network for estimation performance analysis, 2) the Plot panel where the graph capturing network structure is displayed, and 3) the Output panel where the system dynamics and a variety of information to help with interpreting estimation performance is displayed. Here let us describe each of these three panels in detail.

A. Options Panel

The Options panel allows users to select various input modes and set up the network for estimation performance analysis. This panel includes four major sub-panels: the File sub-panel, the Plot Options sub-panel, the Settings sub-panel and the Scaling Factor sub-panel.

File Sub-Panel: The File sub-panel allows end-users to select the source of network and save the current network structure. The drop-down menu has three options:

- **Plot a Network**: This is the default option. When this option is chosen, the user can draw a new network structure in the plot area.

- **Open Saved Network**: This option allows the user to load an existing network structure into the GUI. After the network structure is loaded, the associated graph will appear in the plot area, and the user can then freely modify it. The network structure is stored in a \(*.mat\) format. In particular, the \(*.mat\) file contains two matrices: \(\text{NodeLocation} \in \mathbb{R}^{n \times 2}\) and \(\text{Adjacency} \in \mathbb{R}^{n \times n}\). Matrix \(\text{NodeLocation}\) stores the \(x\) location (first column) and \(y\) location (second column) of each node respectively. Matrix \(\text{Adjacency}\) is the adjacency matrix of the network, storing the connectivity information among the nodes. For instance, if there is a directed edge from vertex \(i\) to \(j\), then the \(\text{Adjacency}\) matrix has a value ‘1’ at the \(i\)th row and \(j\)th column entry, denoted as \(\text{Adjacency}_{ij}\). Similarly, if there is an undirected edge between two vertices \(i\) and \(j\), both the entries \(\text{Adjacency}_{ij}\) and \(\text{Adjacency}_{ji}\) are ‘1’.

- **Save Current Network**: This option allows the user to store the current network structure in the plot area into a \(*.mat\) file. The format of the file is discussed above.

Plot Options Sub-Panel: This sub-panel allows end users to create and modify a network structure in the plot area. The sub-panel has three drop-down menus, namely Draw/Delete/ Drag, Vertex/Edge, and Directed/Undirected. Here let us discuss the functionality for each combination of the selections defined by the three drop-down menus.

- **Draw + Vertex**: This combination allows the user to draw nodes in the plot area. Once the mouse is clicked, a node appears in the plot area and is labeled with a natural number starting from 1. The current \(x\) and \(y\) locations of the mouse appear at the top of the plot area to facilitate the creation of a desired structure.
• **Delete + Vertex**: With this combination, once the mouse is clicked, the vertex that is closest to the click and all edges connected to the vertex will disappear.

• **Draw + Edge + Directed/Undirected**: At this mode, when the user presses the mouse close to one vertex $i$, moves it to close to another vertex $j$, and then releases it, an edge between $i$ and $j$ is drawn in the undirected case, and from $i$ to $j$ is drawn in the directed case. The analysis developed in this paper is only on the undirected network structures.

• **Delete + Edge + Directed/Undirected**: Similar to the above case, the edge that is the closest to the line defined by the press-release actions will be deleted.

• **Drag Nodes**: At this mode, when the user presses a node, moves it to another location and then releases it, the node is dragged to the new location and all edges connected to the node are moved simultaneously. The dragging capability is particularly useful to study the impact of network structural change on estimation performance.

**SETTINGS SUB-PANEL**: This panel allows the user to define actuation locations, measurement locations, and also the variance of Gaussian noise that the network is subject to.

• **Actuation Location**: The actuation location specifies the index of the node at which the network is actuated. In particular, we assume that a unit impulse input is applied at this location. Impulse-type inputs are widely used to capture short-time disturbances to systems (such as power failure in power grids, heat short to rats in biological experiments, etc.) [14,15].

• **Measurement Location**: This is where the system response is measured. We assume that the network is measured at only one location. Moreover, an infinite length of sampled measurement sequence is available, so as to obtain the maximum estimation performance available at this measurement location.

• **Noise Variance**: We assume that the measurement is corrupted by a Gaussian noise with the variance $\sigma^2$ specified here. As we will show in Section III, the variance of noise scales proportionally with CRB.

**SCALING FACTOR SUB-PANEL**: In this paper, we consider typical spread dynamics [19], which requires the state matrix $A$ to be a stochastic matrix that satisfies two properties: 1) row sum being 1, and 2) all the elements being in the range of 0 and 1 (i.e., $0 < A_{ij} < 1$). To ensure this requirement, a scaling process is required to convert distance-based edge weights into probability-like state matrix entries. This sub-panel displays the automatically selected scaling factor, and also allows the user to change it or fix its value so as to allow a fair comparison of estimation performance among networks subject to topological change. This sub-panel include three parts.

• **Display/Edit Box**: This component serves as a display box, showing the current scaling factor when **fix scaling factor** is unchecked. In this case, the scaling factor is automatically determined. It also serves as an edit box and allows the user to type the scaling factor when **Fix Scaling Factor** is checked.

• **Fix Scaling Factor Check-Box**: When this check box is checked, the scaling factor maintains the same despite all structural changes; otherwise it is determined automatically for each particular structure instance. This function is particularly useful when the user wishes to study the impact of structural change on estimation performance. Keeping the same scaling factor allows a fair comparison among different network structures. If the current scaling factor violates the properties of state matrices (e.g., with entries beyond the range of 0 to 1), the tool pops up an error message saying "Further change is not allowed because of the scaling constraint". When this occurs, the user can type a valid scaling factor directly in the **Display/Edit** box.

• **Ok button**: This button enables the typed scaling factor. All results are recalculated based upon the new scaling factor. The **Ok** button is only valid when **fix scaling factor** is checked.
B. Plot Panel

This panel includes the plot area and three buttons: Enter, Undo, and Redraw.

- **Enter**: After completing the network construction, pressing the Enter button displays the estimation performance results in the Output panel. At the mean time, the actuation, measurement, and best measurement locations are marked in blue, red, and green respectively. Moreover, edge weights are also displayed in the plot area.

- **Undo**: This button cancels the last change (e.g., draw/change/delete) to the network.

- **Redraw**: Pressing this button will pop up a window asking the user to delete the entire network and start with drawing a new network. Pressing Yes will clear everything and this operation cannot be undone.

C. Output Panel

This panel displays the network dynamics, and all results that assist users in assessing estimation performance. Specifically, the panel includes the following components:

**System Dynamics Sub-Panel:** The system dynamics associated with the network structure displayed in the plot area is represented in a state-space form (refer to [18] for more details):

$$\begin{align*}
\mathbf{x}[k+1] &= A\mathbf{x}[k] + B\mathbf{u}[k] \\
y[k] &= C\mathbf{x}[k] + w[k],
\end{align*}$$

where $\mathbf{x}[k] \in \mathbb{R}^n$ are the state variables representing the responses at each of the $n$ nodes in the network, $\mathbf{u}[k]$ is a unit impulse signal, and $y[k]$ represents the measurement signal. As there is only one actuation location in the network, we indicate the actuation location in the vector $\mathbf{B} \in \mathbb{R}^{n \times 1}$. Specifically, if the actuation is located at node $i$, $\mathbf{B}$ is a vector with its $i$th entry equal to 1, and all the other entries equal to 0. Similarly, $\mathbf{C} \in \mathbb{R}^{1 \times n}$ is a row vector, with its $j$th entry being 1 representing the measurement location, while all the other entries being 0. Moreover, $w[k]$ is the additive Gaussian noise with mean 0 and variance $\sigma^2$.

State matrix $A \in \mathbb{R}^{n \times n}$ is a stochastic matrix that captures the network structure shown in the plot area. In particular, $A$ is calculated from the network graph in the following. The tool first calculates the weight matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$ where $W_{ij}$ is the inverse of the distance between node $i$ and node $j$. The Laplacian matrix $\mathbf{L}$ is then generated from $\mathbf{W}$ in such a way that each row sum of $\mathbf{L}$ equals 0. In another words, the diagonal entries of $\mathbf{L}$ equal the negative sum of all off-diagonal entries in the same row. Matrix $\mathbf{L}$ is further scaled with a scaling factor $\delta$ such that all entries of $\mathbf{L}$ are between 0 and 1. The default scaling factor $\delta$ is given in Equation (2).

$$\delta = \frac{1}{1 + \max(\text{diag}(\mathbf{L}))},$$

where $\text{diag()}$ means placing the diagonal entries of $\mathbf{L}$ into a vector. Finally, $A$ matrix is formed as $A = I - L$, where $I$ is the identity matrix.

**Pole-Residue Representation Sub-Panel:** The pole-residue representation of the system transfer function $H[z]$ is the key toward the Cramer-Rao analysis, as it separates system poles as estimation parameters (see the detailed discussion in Section III and [13,18]). In particular, the representation is in the following format:

$$H[z] = \sum_{i=0}^{n-1} \frac{A_i}{z - p_i},$$

where $A_i$ are the residues and $p_i$ are the poles of the network. We note that all eigenvalues of $A$ lies between $-1$ and 1, with the largest eigenvalue at 1.
RIGHT EIGENVECTORS SUB-PANEL: The sub-panel displays the right eigenvector associated with each eigenvalue. As seen from Section IV, estimation performance studies rely on algebraic graph theory results that relate eigenvectors with structural change [3,6,7].

FISHER INFORMATION SUB-PANEL: This sub-panel displays the Fisher Information matrix $I(\theta)$. Fisher Information matrix specifies the level of uncertainty in the observation sequence $y$ with respect to the unknown parameter vector $\theta$ (see [10,13] for the details).

$$I(\theta) = -E(\frac{\partial \ln p(y; \theta)^T \partial \ln p(y; \theta)}{\partial \theta})$$

where $y$ is the observation signal, $p(y; \theta)$ is the probability function of $y$ expressed in terms of $\theta$, $^T$ represents matrix transpose, and $E()$ is the expectation operator.

Cramer-Rao Bounds for Poles Sub-Panel: Cramer-Rao bound specifies a lower bound on the variance of estimated parameter $\theta$. System poles and their associated CRB are displayed in this sub-panel. CRB is obtained by inverting the Fisher Information matrix as shown in Equation (5) [10].

$$\text{var}(\theta) \geq I(\theta)^{-1}$$

EFFICIENT OBSERVATION PLACEMENT SUB-PANEL: Given a fixed actuation location, this sub-panel provides the best measurement location to observe the second largest eigenvalue, or named the Fiedler eigenvalue [6]. The Fiedler eigenvalue is a critical indicator of system dynamics and has received extensive study (see e.g., [2,6,7,17,19]). For instance, it captures the mixing speed of a markov chain [2], the convergence speed of consensus building [17], and the spread speed of an epidemic [19]. The CRB associated with this optimal observation is also displayed.

III. Theoretical Background

In this section, we describe the theoretical foundations of the tool. In particular, we first show how the CRBs for pole estimations are obtained from the observation sequence. This analysis is based upon the pole-residue representation which relates system response (captured by the observation sequence) with pole locations. Moreover, in many applications, we are interested in finding observation locations associated with the best estimation performance. These locations represent the most effective places to ascertain system dynamics, or equivalently, the most vulnerable sites to release system dynamics if seized by adversaries. We thus also discuss the calculation of optimal measurement locations.

A. Fisher information matrix

In this section, we consider the system shown in Equation (1), and discuss how the Fisher information matrix for pole estimation can be obtained. Most of the analysis discussed here can be found in [18]. We summarize the results here for the completeness of our presentation.

We start from the pole-residue representation of system dynamics as shown in Equation (3). We denote the parameter of interest $\theta$ as a cascade of the parameters $A = [A_0 \ A_1 \ A_2 \ ...]$ and $p = [p_0 \ p_1 \ p_2 \ ...]$ in the pole-residue form. In particular, $\theta = [A \ P]$. Moreover, as the output sequence $y[k]$ can be written as [18]

$$y[k] = \sum_{i=0}^{n-1} A_i p_i^k + w[k],$$

we can easily obtain the Fisher information matrix as [13,18]
$$F(\theta) = \frac{1}{\sigma^2} \frac{\partial y_i}{\partial \theta} \frac{\partial y_j}{\partial \theta} = \frac{1}{\sigma^2} \left[ \frac{\partial y_i}{\partial \theta} \frac{\partial y_j}{\partial \theta} \right]$$

\[
\begin{bmatrix}
1 & p_0 & p_0^2 & p_0^3 & \cdots \\
p_1 & p_1 & p_1^2 & p_1^3 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots \\
0 & A_0 & 2A_0p_0 & 3A_0p_0^2 & \cdots \\
0 & A_1 & 2A_1p_1 & 3A_1p_1^2 & \cdots \\
0 & A_{n-1} & 2A_{n-1}p_{n-1} & 3A_{n-1}p_{n-1}^2 & \cdots 
\end{bmatrix}
\]

\[
= \frac{1}{\sigma^2} \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}
\]

where

\[
a_1 = \begin{bmatrix}
(1 + p_0^4 + \cdots) & (1 + p_0p_1 + p_0^2p_1^2 + \cdots) & \cdots & (1 + p_0p_{n-1} + p_0^2p_{n-1}^2 + \cdots) \\
(1 + p_0p_1 + p_0^2p_1^2 + \cdots) & (1 + p_1^4 + \cdots) & \cdots & (1 + p_1p_{n-1} + p_1^2p_{n-1}^2 + \cdots) \\
\vdots & \vdots & \ddots & \vdots \\
(1 + p_0p_{n-1} + p_0^2p_{n-1}^2 + \cdots) & (1 + p_1p_{n-1} + p_1^2p_{n-1}^2 + \cdots) & \cdots & (1 + p_{n-1}^4 + \cdots)
\end{bmatrix}
\]

\[
= \begin{bmatrix}
\frac{1}{1-p_0^2} & \frac{1}{1-p_0p_1} & \frac{1}{1-p_0p_{n-1}} \\
\frac{1}{1-p_0p_1} & \frac{1}{1-p_1^2} & \frac{1}{1-p_1p_{n-1}} \\
\vdots & \vdots & \vdots \\
\frac{1}{1-p_0p_{n-1}} & \frac{1}{1-p_1p_{n-1}} & \frac{1}{1-p_{n-1}^2}
\end{bmatrix}
\]

\[
a_2 = \begin{bmatrix}
(p_0A_0 + p_0^3(2A_0p_0) + \cdots) & (p_1A_0 + p_1^3(2A_0p_0) + \cdots) & \cdots & (p_{n-1}A_0 + p_{n-1}^3(2A_0p_0) + \cdots) \\
p_0A_1 + p_0^3(2A_1p_1) + \cdots) & (p_1A_1 + p_1^3(2A_1p_1) + \cdots) & \cdots & (p_{n-1}A_1 + p_{n-1}^3(2A_1p_1) + \cdots) \\
\vdots & \vdots & \ddots & \vdots \\
(p_0A_{n-1} + p_0^3(2A_{n-1}p_{n-1}) + \cdots) & (p_1A_{n-1} + p_1^3(2A_{n-1}p_{n-1}) + \cdots) & \cdots & (p_{n-1}A_{n-1} + p_{n-1}^3(2A_{n-1}p_{n-1}) + \cdots)
\end{bmatrix}
\]

\[
= \begin{bmatrix}
\frac{A_0A_0}{(1-p_0p_0)^2} & \frac{A_0A_1}{(1-p_0p_1)^2} & \frac{A_0A_{n-1}}{(1-p_0p_{n-1})^2} \\
\frac{A_1A_0}{(1-p_1p_0)^2} & \frac{A_1A_1}{(1-p_1p_1)^2} & \frac{A_1A_{n-1}}{(1-p_1p_{n-1})^2} \\
\vdots & \vdots & \vdots \\
\frac{A_{n-1}A_0}{(1-p_{n-1}p_0)^2} & \frac{A_{n-1}A_1}{(1-p_{n-1}p_1)^2} & \frac{A_{n-1}A_{n-1}}{(1-p_{n-1}p_{n-1})^2}
\end{bmatrix}
\]
The CRB for the estimate of parameter \(a\) are easier to estimate than the ones with smaller residues [18].

An important role in the performance of pole estimation. Generally, the poles associated with large residues

Of our particular interest, the CRB for the calculation of CRB is involved with inverting a large matrix that is prone to be ill-conditioned, we instead

\[
\begin{bmatrix}
(p_0A_0 + p_0^2(2A_0p_0) + p_0^3(3A_0p_0^2) + \ldots) & \ldots & (p_0A_{n-1} + p_0^2(2A_{n-1}p_{n-1}) + p_0^3(3A_{n-1}p_{n-1}^2) + \ldots) \\
(p_1A_0 + p_1^2(2A_0p_0) + p_1^3(3A_0p_0^2) + \ldots) & \ldots & (p_1A_{n-1} + p_1^2(2A_{n-1}p_{n-1}) + p_1^3(3A_{n-1}p_{n-1}^2) + \ldots) \\
\vdots & \ddots & \vdots \\
(p_{n-1}A_0 + p_{n-1}^2(2A_0p_0) + p_{n-1}^3(3A_0p_0^2) + \ldots) & \ldots & (p_{n-1}A_{n-1} + p_{n-1}^2(2A_{n-1}p_{n-1}) + p_{n-1}^3(3A_{n-1}p_{n-1}^2) + \ldots)
\end{bmatrix}
\]

\[
a_3 = \begin{bmatrix}
\frac{A_0p_0}{(1-p_0^2)} & \frac{A_0p_1}{(1-p_0p_1)} & \frac{A_0p_{n-1}}{(1-p_0p_{n-1})} \\
\frac{A_1p_0}{(1-p_0^2)} & \frac{A_1p_1}{(1-p_0p_1)} & \frac{A_1p_{n-1}}{(1-p_0p_{n-1})} \\
\vdots & \ddots & \vdots \\
\frac{A_{n-1}p_0}{(1-p_0^2)} & \frac{A_{n-1}p_1}{(1-p_0p_1)} & \frac{A_{n-1}p_{n-1}}{(1-p_0p_{n-1})}
\end{bmatrix}
\]  \quad (10)

\[
\begin{bmatrix}
(A_0^2 + (2A_0p_0)^2 + (3A_0p_0^2)^2 + \ldots) & \ldots & (A_0A_{n-1} + (2A_0p_{n-1})(2A_{n-1}p_{n-1}) + \ldots) \\
(A_0A_1 + (2A_1p_0)(2A_0p_0) + (3A_1p_0^2)(3A_0p_0^2) + \ldots) & \ldots & (A_1A_{n-1} + (2A_1p_{n-1})(2A_{n-1}p_{n-1}) + \ldots) \\
\vdots & \ddots & \vdots \\
(A_0A_{n-1} + (2A_{n-1}p_{n-1})(2A_0p_0) + (3A_{n-1}p_{n-1}^2)(3A_0p_0^2) + \ldots) & \ldots & (A_{n-1}^2 + (2A_{n-1}p_{n-1})^2 + (3A_{n-1}p_{n-1}^2)^2 + \ldots)
\end{bmatrix}
\]

\[
a_4 = \begin{bmatrix}
\frac{A_0^2(1+p_0^2)}{(1-p_0^2)^2} & \frac{A_0A_1(1+p_0p_1)}{(1-p_0p_1)^2} & \frac{A_0A_{n-1}(1+p_0p_{n-1})}{(1-p_0p_{n-1})^2} \\
\frac{A_1^2(1+p_0^2)}{(1-p_0^2)^2} & \frac{A_1A_1(1+p_0p_1)}{(1-p_0p_1)^2} & \frac{A_1A_{n-1}(1+p_0p_{n-1})}{(1-p_0p_{n-1})^2} \\
\vdots & \ddots & \vdots \\
\frac{A_{n-1}^2(1+p_0^2)}{(1-p_0^2)^2} & \frac{A_{n-1}A_1(1+p_0p_1)}{(1-p_0p_1)^2} & \frac{A_{n-1}A_{n-1}(1+p_0p_{n-1})}{(1-p_0p_{n-1})^2}
\end{bmatrix}
\]  \quad (11)

\[B. \quad \text{Cramer-Rao Bounds for Pole Estimation}\]

The CRB for the estimate of parameter \(\theta_i\) is the \(i\)th diagonal entry of the inverse of the Fisher information matrix:

\[C_i = (F(\theta)^{-1})_{ii}.\]  \quad (12)

Of our particular interest, the CRB for the \(i\)th pole \(p_i\) (denoted as \(C(p_i)\)) equals \(C_{n+i+1}\). Because the calculation of CRB is involved with inverting a large matrix that is prone to be ill-conditioned, we instead find a lower-bound of the CRB. In particular, the CRB associated with the \(i\)th pole is bounded by [18]

\[C(p_i) \geq \frac{1}{\sigma^2} \frac{A_i^2(1+p_i^2)}{(1-p_i^2)^2} = \sigma^2 \frac{(1-p_i^2)^3}{A_i^2(1+p_i^2)}\]  \quad (13)

The above result shows that CRB scales with the variance of Gaussian noise. Moreover, the residues play an important role in the performance of pole estimation. Generally, the poles associated with large residues are easier to estimate than the ones with smaller residues [18].

\[C. \quad \text{Effective Sensor Placement}\]

In many sensor network applications such as bridge health monitoring and environmental monitoring, sensors are placed ad hoc. However, as noted recently, smartly choosing sensor locations can significantly enhance
estimation performance \cite{8,21}. Our philosophy is that some basic understanding of network structure can lead to a structure-exploiting design of effective sensor locations.

In this tool, the best observation location is displayed for a fixed network structure and actuation location. The best observation location is achieved by looping through all possible locations and comparing the CRBs of the Fiedler eigenvalue associated with each location.

\section{Some Graphical Results on Estimation Performance for Subsets of Network Structures}

The tool allows us to investigate network structure’s impact on the performance of pole estimation. The graphical representation of network structure and the modification features permitted on the graph facilitate the discovery of theoretical results. In this section, we present some preliminary graphical results on estimation performance and their proofs.

The first theorem suggests that exchanging actuation and measurement locations does not change the estimation performance of any pole.

\textbf{Theorem 1} Consider the estimation of an eigenvalue $\lambda_r$ from a network with the dynamics described in Equation (1). If the network is actuated at node $i$ and measured at node $j$, the CRB for $\lambda_r$ does not change if the actuation and measurement locations are switched.

\textbf{Proof:} According to Theorem 6 in \cite{18}, the CRB for $\lambda_r$ scales inversely with $(v_{ri}v_{rj})^2$, where $v_{ri}$ is the $i$th entry in the eigenvector associated with the eigenvalue $\lambda_r$. After the actuation and measurement locations are switched, let us denote the new eigenvector entries associated with actuation location $j$ and measurement location $i$ as $v'_{rj}$ and $v'_{ri}$ respectively. The CRB now scales inversely with $(v'_{rj}v'_{ri})^2$. As exchanging actuation and measurement locations does not change the right eigenvector of a system, we have $v_r = v'_r$. As such, the CRB in the new setting now scales inversely with $(v'_{rj}v'_{ri})^2 = (v_{rj}v_{ri})^2 = (v_{ri}v_{rj})^2$. The proof is complete. $\square$

The second theorem extends the result in \cite{18} and informs the selection of observation locations in a tree graph when the network is actuated at a leaf of the tree.

\textbf{Theorem 2} Let us consider estimating the Fiedler eigenvalue of a network with a tree structure. If the network is actuated at a leaf, the minimum CRB is achieved when the measurement location is also at the leaves of the tree. Moreover, the worst observation location is among the nodes adjacent to the algebraic center of the tree.

\textbf{Proof:} The estimation performance of the Fiedler eigenvalue scales inversely with $(v_{li}v_{lj})^2$, where $i$ is the actuation location, $j$ is the measurement location, and $v_l$ is the eigenvector associated with the Fiedler eigenvalue, commonly called the Fiedler eigenvector. Moreover, according to \cite{6,7}, entries in the Fiedler eigenvector increase or decrease monotonically from the algebraic center of the tree. As such, the maximum values of eigenvector entries are achieved at the leaves and the minimum are among the nodes adjacent to the algebraic center of the tree. Combining the above two results leads to the conclusion. $\square$

The next theorem suggests the node in a tree graph where the estimation of Fiedler eigenvalue is impossible.

\textbf{Theorem 3} Consider the estimation of the Fiedler eigenvalue of a network with a tree structure. If the algebraic center is a node in the tree, inference is not possible if this particular node is either the actuation or measurement locations.

\textbf{Proof:} If the algebraic center is a node (denoted as $k$) in a tree, the entry $v_{1k}$ in the Fielder eigenvector is 0 according to Theorem 4.3 in \cite{6}. As the estimation performance of the Fiedler eigenvalue scales inversely with $(v_{1i}v_{1k})^2$ if the network is actuated at node $i$, or $(v_{1k}v_{1j})^2$ if the network is measured at node $j$, ...
we obtain that the CRB is $\infty$ in each case, indicating that estimating the Fiedler eigenvalue is impossible. □

The above theorem naturally leads to the conclusion that in any symmetric tree graph, the inference of the Fiedler eigenvalue is impossible if the network is actuated or measured at the center of the symmetry. Examples of symmetric trees include star graphs with all edge weights the same.

The next theorem is concerned with the trend of CRB when the edge weight is changed in a simple graph with only two nodes.

**Theorem 4** Consider a graph with two nodes. Increasing the edge weight between the two nodes from 0 to $\infty$ causes the CRB to increase monotonically and then decrease monotonically. The largest CRB occurs when the edge weight is $0.5$.

**Proof:** In the two-node case, as suggested by Equation (11), the CRB scales inversely with $\frac{(1+p_1^2)}{(1-p_1^2)^3}$ and $A_1$, where $A_1 = (v_{11}v_{12})^2$. Because the first norm of the Fiedler eigenvector equals 0, we have $v_{11} = -v_{12}$. Moreover, as $v_{11}^2 + v_{12}^2 = 1$, we can easily obtain that $v_{11} = v_{12} = \sqrt{0.5}$, independent of the length between the two nodes. Because $A_1$ is a constant, CRB is only affected by the value of the pole $p_1$. As the length between the two nodes increases from 0, the Fiedler eigenvalue $p_1$ increases from $-1$ to 1, and as such causing $\frac{(1+p_1^2)}{(1-p_1^2)^3}$ to decrease first and then increase, with the minimum occurring while $p_1 = 0$. This occurs when the edge weight between the two nodes is 0.5. □

V. Concluding Remark and Future Work

In this paper, we introduce an interactive tool to investigate the performance of network dynamics from noisy data. The contributions of the tool is that 1) it facilitates the theoretical analysis of network structure's impact on network estimation performance, and 2) it suggests the best measurement location to maximize network estimation performance, and thus provides guidance for the design of data collection experiments. In the future work, we will utilize the tool to develop further analytical results for non-tree structures. Moreover, we will extend the network structure from single layer to hierarchical structures, and consider network dynamics defined upon Laplacian and adjacency matrices.

Acknowledgment

We thank National Science Foundation for the support under grant number 1058110. We also thank Dr. Sandip Roy at Washington State University for various illuminating discussions.

References


