Graph theory in Systems and Control

Mehran Mesbahi

Department of Aeronautics & Astronautics

University of Washington

CDC 2018
Miami Beach, Florida, December 2018

Networked systems

Why a tutorial on graph theory in systems and control?

- networks are all around us
- this trend will continue, e.g., internet of things, next generation mobility
- networked robotics and aerospace systems will play an ever increasing role in the society at all levels
- system and control theory can play a significant role in this new era of networked systems ...
- however, we need to start blending in combinatorial/discrete mathematics in mainstream control theory even more ...

This tutorial is framed around this objective ...

a Few Immediate Observations

- networked systems are coupled through information exchange
- inter-agent information exchange is through sensing and communication
- the collective dynamics is a function of "agent" dynamics and the information-induced coupling
- we can synthesize collective behavior by making the control action on each agent a function of the information available to the agent (sense, communicated, etc.)

a powerful abstraction for encoding "interactions" in a network is that of a graph

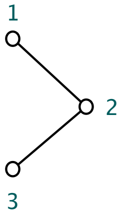
Graph Abstraction

- a finite, undirected, simple graph, or a graph for short, is built upon a finite set of "nodes" or vertex set $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$
- the edge set is a subset of the two-element subsets of $\mathcal{V}$, i.e., $\mathcal{E} \subseteq [\mathcal{V}]^2$
- the graph is then specified by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

for example, we can have $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where

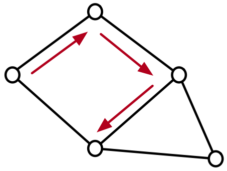
$$\mathcal{V} = \{1, 2, 3\} \quad \text{and} \quad \mathcal{E} = \{\{1,2\},\{2,3\}\}$$

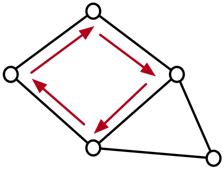
a simpler representation however would be

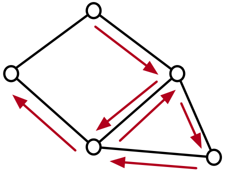


Some natural constructs based on the correspondence between set theoretic and graph-theoretic representation can now be defined– examples: paths, walks, cycles, etc.

Simple Constructs on Graphs


a path

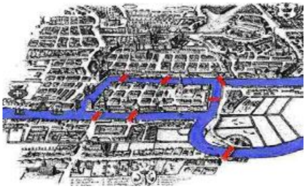

a cycle

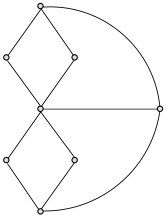

a walk

graphs can be used in general to encode relations between objects, e.g., existence of communication or sensing links, routes, etc.

Birth of Graph Theory

bridges of Konigsberg and Euler's abstraction:





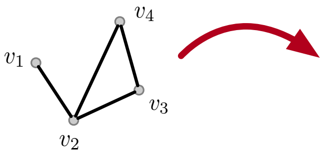
this is an important step, as it stripes away all particular details related to the Konigsberg bridges that are not relevant to the problem at hand! so now we have a graph! what are we looking for now? We want to find out if there is a closed walk traversing all edges of the graph exactly once. If such a walk exists we call the graph Eulerian.

Theorem

A connected graph $\mathcal{G}$ is Eulerian if and if only every vertex has an even degree.

Graphs and Matrices

As we aim to embed graph/networks in dynamic systems, it is natural to work with linear algebraic representation. For example, a graph can be represented as,



$$A(\mathcal{G}) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

the adjacency matrix for the n-node graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is the $n \times n$ matrix:

$$[A(\mathcal{G})]_{ij} = \begin{cases} 1 & \text{if } \; v_i v_j \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Degree Matrix and the Laplacian

note that the adjacency for the graph is symmetric by construction
there are other matrices associated with the graph, for example, let $d(v)$ be the number of neighbors of vertex v (its degree) and define the degree matrix as,

$$\Delta(\mathcal{G}) = \begin{pmatrix} d(v_1) & 0 & \cdots & 0 \\ 0 & d(v_2) & \cdots & 0 \\ \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d(v_n) \end{pmatrix}$$

note that the adjacency and the degree matrices are both square, say, $n \times n$, where n is the number of nodes

Another useful matrix representation is the Laplacian:

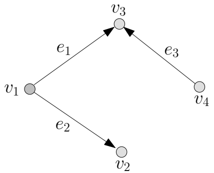
$$L(\mathcal{G}) = \Delta(\mathcal{G}) - A(\mathcal{G})$$

graph Laplacian has been very popular in multiagent networks!

Incidence Matrix

Yet another matrix representation can in fact capture the orientation of the edge as well: suppose the graph has n nodes and m edges: the $n \times m$ *incidence matrix* $E(\mathcal{G})$ is defined as

$$E(\mathcal{G}) = [E_{ij}], \quad \text{where} \quad E_{ij} = \begin{cases} -1 & \text{if } v_i \text{ is the tail of } e_j, \\ 1 & \text{if } v_i \text{ is the head of } e_j, \\ 0 & \text{otherwise.} \end{cases}$$



$$E(\mathcal{G}) = \begin{bmatrix} -1 & -1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

note that for different orientations on the edges we get a different incidence matrix! (same dimension though!)

Let us see what happens when we consider $E(\mathcal{G})E(\mathcal{G})^T$ for some arbitrary orientation. First notice that the resulting matrix will be $n \times n$.

Incidence and Laplacian

A compact formula for matrix multiplication is of course:

$$[AB]_{ij} = \sum_k A_{ik} B_{kj}$$

$$[E(\mathcal{G})E(\mathcal{G})^T]_{ij} = \sum_k E(\mathcal{G})_{ik} E(\mathcal{G})_{jk}$$

which is -1 when i and j are incident on the same edge k, that is if they are neighbors! Moreover,

$$[E(\mathcal{G})E(\mathcal{G})^T]_{ii} = \sum_k E(\mathcal{G})_{ik} E(\mathcal{G})_{ik}$$

counts the number of edges incident on node i, i.e., its degree! so guess what:

$$L(\mathcal{G}) = E(\mathcal{G})E(\mathcal{G})^T$$

independent of the orientation that you have given to the incidence matrix! This also shows that $L(\mathcal{G})$ is positive semi-definite, since for all $x \in \mathbf{R}^n$:

$$x^T L(\mathcal{G}) x = x^T E(\mathcal{G})E(\mathcal{G})^T x = \|E(\mathcal{G})^T x\|^2 \geq 0$$

which means that not only are the eigenvalues of the Laplacian real numbers (as the Laplacian is symmetric) but also non-negative

Spectra of the Graph Laplacian

For Laplacian, we can order the eigenvalues as follows,

$$0 \leq \lambda_1(\mathcal{G}) \leq \lambda_2(\mathcal{G}) \leq \ldots \lambda_n(\mathcal{G});$$

in this case, λ_k refers to the kth smallest eigenvalue of the (graph) Laplacian ...

- By construction, $L(\mathcal{G})\mathbf{1} = 0$ for any graph (why?). So $\lambda_1(\mathcal{G}) = 0$.
- A natural question (with many consequences) is whether $\lambda_2(\mathcal{G}) > 0$?
- In other words, we need to characterize the null space of $L(\mathcal{G})$:

$$\mathcal{N}(L(\mathcal{G})) = \{z \in \mathbf{R}^n \,|\, L(\mathcal{G})z = 0\}$$

What are the vectors in $\mathcal{N}(L(\mathcal{G}))$ except the subspace generated by $\mathbf{1}$, namely,

$$\mathcal{A} = \{x \,|\, x = \alpha\mathbf{1},\ \alpha \in \mathbf{R}\}$$

Null Space of the Laplacian

in order to answer this question, notice that if $z \in \mathcal{N}(L(\mathcal{G}))$, then

$$L(\mathcal{G})z = E(\mathcal{G})E(\mathcal{G})^T z = 0$$

that is,

$$z^T E(\mathcal{G})E(\mathcal{G})^T z = 0$$

or $\|E(\mathcal{G})^T z\|^2 = 0$ or $E(\mathcal{G})^T z = 0$ or $z^T E(\mathcal{G}) = 0$. This means that if $ij \in E$, then $z_i = z_j$; so if the graph is connected,

$$z_1 = z_2 = \ldots = z_n$$

that is $z = \alpha \mathbf{1}$ for some α! And in fact, if we think of z as

$$z : \mathcal{V}(\mathcal{G}) \to \mathbf{R}^n$$

then z is constant on each (connected) component of $\mathcal{G}$. What that means is that for each component we get one extra dimension for the null space of $L(\mathcal{G})$.

Lemma

Let $\mathcal{G}$ have c connected components (when $c = 1$ the graph is connected). Then **rank** $L(\mathcal{G})$ is $n - c$.

Rank, λ_2, and Connectivity

and in fact, **rank** $L(\mathcal{G}) = n-1$ if and only if $\mathcal{G}$ is connected! this is our first encounter with how the "linear algebra" of the Laplacian tells us something about the structure of the graph.
another way to say the same thing is that

$$\mathcal{G} \text{ is connected if and only if } \lambda_2(\mathcal{G}) > 0$$

a natural question now is whether more positive λ_2 captures some qualitative notion of "more" connectivity? For example, we can define the node connectivity of $\mathcal{G}$, denoted by $\kappa_0(\mathcal{G})$ as the minimum number of nodes that needs to be removed from the graph before the graph becomes disconnected.

Courant-Fisher to the rescue:

$$\lambda_2(\mathcal{G}) = \min_{x \perp \mathbf{1}, \|x\|=1} x^\top L(\mathcal{G}) x$$

So this means that

$$\lambda_2(\mathcal{G}) \le x^\top L(\mathcal{G}) x \quad \text{for all } x \perp \mathbf{1}, \|x\| = 1$$

Structure vs. Spectra

Let us consider removing $S \subset \mathcal{V}$ (subset of nodes) from the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; we denote the Laplacian of this new graph as $L(\mathcal{G} \backslash S)$.

Let y be the normalized eigenvector corresponding to $\lambda_2(\mathcal{G} \backslash S)$:

$$L(\mathcal{G} \backslash S) y = \lambda_2(\mathcal{G} \backslash S) y; \quad \|y\| = 1, y \perp \mathbf{1}$$

Now define the vector

$$z = \begin{bmatrix} y \\ 0 \end{bmatrix};$$

note that $\|z\| = 1$ and $z \perp 1$; as such $\lambda_2(\mathcal{G}) \leq z^\top L(\mathcal{G}) z$. That is,

$$\lambda_2(\mathcal{G}) \leq \sum_{uv \in E(\mathcal{G} \backslash S)} (y_u - y_v)^2 + \underbrace{\sum_{uv \in E(S)} (z_u - z_v)^2}_{0} + \sum_{u \in S} \sum_{v \in \mathcal{G} \backslash S} (\underbrace{z_u}_{0} - z_v)^2$$

so,

$$\lambda_2(\mathcal{G}) \leq \lambda_2(\mathcal{G} \backslash S) + \sum_{u \in S} 1 = \lambda_2(\mathcal{G} \backslash S) + |S|$$

Spectra vs. Structure

Okay! Now suppose that S is chosen as the cutset corresponding to $\kappa_0(\mathcal{G})$. Then $\lambda_2(\mathcal{G}\backslash S) = 0$ and

$$\lambda_2(\mathcal{G}) \leq \kappa_0(\mathcal{G})$$

Upshot: $\lambda_2(\mathcal{G})$ is a lower bound for node connectivity!

The bound is actually tight, for example $\lambda_2(C_4) = \kappa_0(C_4) = 2$

summary so far:

- $L(\mathcal{G}) = E(\mathcal{G})E(\mathcal{G})^\top = \Delta(\mathcal{G}) - A(\mathcal{G})$
- $L(\mathcal{G})$ is positive semidefinite
- $\lambda_2(\mathcal{G}) > 0$ iff $\mathcal{G}$ is connected
- $\lambda_2(\mathcal{G})$ is a measure of connectivity

Oh ... one last thing: trace of any matrix is the sum of its eigenvalues, so

$$\textbf{trace}\, L(\mathcal{G}) = \sum_i d(v_i) = 2\,|\mathcal{E}(\mathcal{G})|$$

Spectra of Some Classes of Graphs

Complete Graph

It would be good to develop some intuition for spectra of graphs, and in particular their dependencies on n, if any. Of course we have to start with the complete graph on n nodes, denoted by K_n:

$$L(K_n) = \begin{bmatrix} n-1 & -1 & \cdots & -1 & -1 \\ -1 & n-1 & \cdots & -1 & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -1 & -1 & -1 & -1 & n-1 \end{bmatrix} = nI - \mathbf{1}\mathbf{1}^T$$

as always, $\lambda_1(K_n) = 0$ and $u_1 = \mathbf{1}/\sqrt{n}$. The other eigenvectors, generically denoted by x for now, can chosen to be orthogonal to $\mathbf{1}$. So

$$L(K_n)x = (nI - \mathbf{1}\mathbf{1}^T)x = \lambda x$$

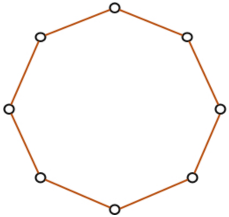
Hence for all these eigenvectors

$$nx = \lambda x!$$

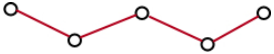
The spectrum of $L(K_n)$ is thus

$$0, n, n, \ldots n; \quad \text{check that} \quad \textbf{trace}\,\{L(K_n)\} = n(n-1)$$

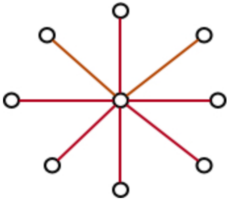
Spectra of some other classes of graphs



$2(1-\cos 2k\pi/n), \quad k=0,1,\ldots n-1$



$2(1-\cos k\pi/n), \quad k=0,1,\ldots n-1$



$n-2$ eigenvalues of 1, one eigenvalue of zero (as always) and last one is $2(n-1)-(n-2)=n$

dynamics on graphs

so far, graphs and some linear algebra, spectra vs. structure, and examples on how to find the spectra in closed form for certain classes of graphs. We now what to see how this machinery actually helps us understand dynamics on networks

Our Action Plan is as follows:

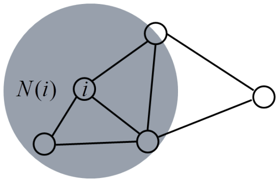
1. we start with a baseline dynamics/distributed algorithm called consensus
2. we relate consensus behavior to structure of the graph
3. this setup can then be extended to directed graphs

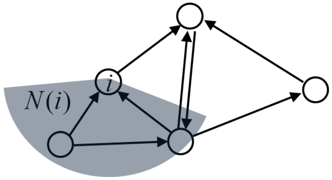
We then move on to show that this distributed algorithm can be used in many different context to do very useful distributed tasks for us

However, it is important to note that the same line of research could have been pursued with a different baseline/distributed protocol or view completely from the perspective of patterned matrices independent of particular protocol!

Network in the Dynamics- general setup

- Graph $\mathcal{G}$ is composed of physical nodes $\mathcal{V}$ and coupling edges $\mathcal{E}$
- Node i acquires information from the set of its neighbors $\mathcal{N}(i)$





- Node i has a state $x_i(t)$ and neighbor information $I_i(t) = \left\{x_j(t) | j \in \mathcal{N}(i)\right\}$
- Provides a naturally distributed dynamics over $\mathcal{G}$

$$\dot{x}_i(t) = f_i(x_i(t), I_i(t))$$

- some of the earlier works in distributed decision-making include: DeGroot ('74), Borkar and Varaiya ('82), Tsitsiklis ('84) ...

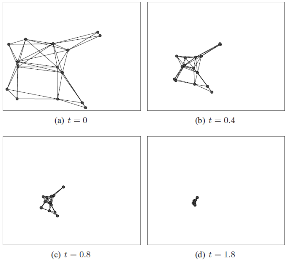
Agreement/Consensus Protocol

Consensus Model

$$\dot{x}_i(t) = -\sum_{j \in N(i)} w_{ij}\left(x_i(t) - x_j(t)\right)$$

$$\rightsquigarrow \dot{x}(t) = -L(\mathcal{G})x(t)$$

where $L(\mathcal{G})$ is the (weighted) Laplacian matrix.


(a) $t = 0$ (b) $t = 0.4$ (c) $t = 0.8$ (d) $t = 1.8$

- appears in: flocking, formation control, opinion dynamics, energy systems, synchronization, distributed estimation, distributed optimization, among many others!

Let us examine the convergence of the algorithm a bit more ... in terms of the graph structure. We will assume that $w_{ij} = 1$ for this purpose, although our observations generalize seamlessly to weighted graphs

Consensus and λ_2

Let us consider consensus on undirected networks ... spectral factorization of the Laplacian is of the form

$$L(\mathcal{G}) = U \Lambda U^\top$$

where

$$U = \begin{bmatrix} u_1 & u_2 & \cdots & u_n \end{bmatrix} \quad \text{and} \quad \Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

as such,

$$\begin{aligned} x(t) &= e^{-L(\mathcal{G})t} x(0) = U e^{-t\Lambda} U^T x(0) \\ &= u_1^\top x(0)\, u_1 + e^{-\lambda_2 t} u_2^\top x(0)\, u_2 + \ldots + e^{-\lambda_n t} u_n^\top x(0)\, u_n \end{aligned}$$

so if the graph is connected (noting that $u_1 = \mathbf{1}/\sqrt{n}$)

$$x(t) \to \frac{\mathbf{1}^T x(0)}{n}\,\mathbf{1} \quad \text{at a rate proportional to } \lambda_2(\mathcal{G})!$$

more on consensus and λ_2

in fact,

$$\begin{aligned}\|x(t) - \frac{\mathbf{1}^T x(0)}{n}\| &= \|\sum_{i=2}^{n} e^{-\lambda_i t} \underbrace{u_i^\top x(0)}_{\alpha_i} u_i\| \\ &= \sum_{i=2}^{n} e^{-\lambda_i t} |\alpha_i| \leq (n-1) \underbrace{\beta}_{\max_i |\alpha_i|} e^{-\lambda_2 t}\end{aligned}$$

so if we want $\|x(t) - \frac{\mathbf{1}^T x(0)}{n}\| \leq \varepsilon$ for some $\varepsilon > 0$, then we need

$$t \geq \{\ln \frac{\beta(n-1)}{\varepsilon}\}/\lambda_2(\mathcal{G}) \propto \frac{1}{\lambda_2(\mathcal{G})}$$

higher algebraic connectivity directly translates to faster convergence (in a linear way)!

what insights graph theory provides for consensus

some observations:

- Recall that $\lambda_2(P_n) = 2(1 - \cos k\pi/n)$, $\lambda_2(C_n) = 2(1 - \cos 2k\pi/n)$, $\lambda_2(S_n) = 1$, and $\lambda_2(K_n) = n$
- what this means is that as $n \to \infty$, the rate of convergence for P_n and C_n goes to zero!
- in the meantime, the rate of convergence for K_n grows linearly with n
- however, the number of edges for P_n, C_n grow linearly with n but for K_n the number of edges is $O(n^2)$!

this thread of thought leads to the area of graph synthesis

how baseline consensus can be used for more elaborate distributed algorithms

- as a distributed subroutine for mixing
- including the right inputs to consensus (not just driven by initial conditions)
- consensus with nonlinear and/or state-dependent weights (used in preserving connectivity in distributed robotics)
- consensus with negative, complex-valued, and matrix weights
- consensus across scales
- consensus with security and privacy considerations