2018 IEEE Conference on Decision and Control (CDC)
Miami Beach, FL, USA, Dec. 17-19, 2018

Graph Theory in Systems and Controls

Daniel Zelazo, Mehran Mesbahi, and Mohamed-Ali Belabbas

Abstract— **This tutorial paper aims to explore the role of graph theory for studying networked and multi-agent systems. The session will cover basic concepts from graph theory along with surveying its role in problems related to cooperative control and distributed decision-making. Finally, we will also introduce some advanced topics from graph theory in the hope of encouraging further discussion and explore new research opportunities in system and control theory.**

I. Introduction and Motivation

The rapid pace of innovation in the areas of control theory, computation, and communication is leading the way for a new class of networked systems characterized by their complex interconnections, diversity of components, and interactions with the physical world. The potential benefit of these networked systems from environmental, economic, and social perspectives are simultaneously tempered by their complexity. New tools are required to efficiently analyze the performance and stability of these systems, simulate their operation, and control their behavior.

Coupled with the optimistic vision of these systems are the challenges associated with their design and operation. In particular, understanding the role of the system interconnections in the overall behavior and performance of a networked system is of paramount importance. Traditional approaches for analysis and synthesis of monolithic dynamic systems must be extended to this networked framework.

Motivated by these engineering goals and technical challenges, the mathematical theory of *graphs* has emerged as the powerful tool to approach these challenges. Graphs are useful abstractions for networked systems that facilitate examining the role of the underlying information-exchange and interaction geometry on the behavior (e.g., noise attenuation, tracking, robustness, synchronization patterns) that these systems exhibit. In the meantime, graph theory is an elegant branch of discrete mathematics with a rich history and a number of sub-disciplines, such as algebraic graph theory, random graphs, and extremal graph theory. When graphs are embedded in linear control systems, the (linear) algebraic framework for analyzing graphs becomes particularly useful.

The use of graphs to study the coordination and control of multi-agent systems has seen rapid developments from within the systems and controls community [1]–[3]. Fundamental problems for multi-agent systems (MAS), such as synchronization, formation control, or assignment problems, have matured owing to our collective improved understanding of the critical role that graphs play. In many system models, the graph itself becomes the focal point, even more-so than the dynamics of the agents itself.

The importance that graph theory plays in studying multi-agent systems provides the motivation and justification for this tutorial paper. The paper takes the perspective that the *combinatorial structure* of the underlying graph in a networked system can reveal deep insights into its system theoretic properties. This includes fundamental properties such as stability, controllability, and performance. Graphs are used to represent the structure of the dynamics as well as the corresponding control system. The type of dynamics allowed over this structure leads to two major strands of research. In the first, we consider the set of all (linear) dynamics that are supported on the graph, in the sense that an edge of the graph corresponds to a free parameter in the associated state matrix. We make this relationship precise through the introduction of zero-patterns. This point of view gives rise to the study of structural properties of systems, which aims to understand when a property is verified, regardless of the particular entries of the system matrix. In the second approach, one singles out a particular type of dynamics on the graph. A prime example for this setting is the diffusion dynamics on graphs, mirroring its infinite-dimensional twin, yet characteristically combinatorial. Although diffusion dynamics has been a cornerstone of applied mathematics, e.g., modeling heat conduction in a medium, examining its finite-dimensional realization on graphs is relatively recent, and aims to highlight the role of graph-theoretic constructs in what often amounts to distributed averaging.[1] The resulting distributed algorithms– simple in their form–lead to a rather effective means of synthesizing a wide array of coordination mechanisms on a network– they essentially provide means of "agreeing" on the states needed for coordination via local interactions.

The first part of this paper is meant to acquaint the reader with the tools and terminology used to understand this rich theory. We then follow with an overview of some fundamental problems in the study of multi-agent and networked systems and show how they can be better understood through the lens of graph theory. The emphasis of this work is on providing a general flavor of the results as opposed to technical rigor or comprehensiveness. The hope is to

D. Zelazo is with the Faculty of Aerospace Engineering, Israel Institute of Technology, Haifa, Israel dzelazo@technion.ac.il; M. Mesbahi is with the Department of Aeronautics and Astronautics, University of Washington, Seattle,WA 98195-2400 USA mesbahi@uw.edu; M.-A. Belabbas is with the Electrical and Computer Engineering department and the Coordinated Science Laboratory, University of Illinois, Urbana-Champaign, IL 61801, USA belabbas@illinois.edu

[1]However, allowing directed edges and inputs to these networks can significantly broaden their "computational" power beyond averaging.

stimulate interest and provide an engaging overview of this emerging field.

II. Graph Theory

The origin of graph theory goes back to the 18th century with the work of Leonard Euler and the celebrated *Königsberg bridge problem*. Euler laid the foundations of the now formal discipline of mathematics. In this section, we will provide an overview of the basic set-theoretic and algebraic description of graphs, and highlight some special topics that will play important roles in the study of networked dynamic systems.

A. Graphs and their structures

A *graph* is a set-theoretic structure that describes how elements in a particular set are related to each other. Formally, we denote a graph by $\mathcal{G}$, and it is specified by a vertex set $\mathcal{V} = \{1, 2, ..., n\}$,[2] and an edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, whose elements characterize the incidence relation between distinct pairs of $\mathcal{V}$. Thus, a graph is completely defined by the pair, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. As graphs are set-theoretic objects, it is also straight forward to define the notion of subgraphs, where $\mathcal{H} = (\mathcal{V}_{\mathcal{H}}, \mathcal{E}_{\mathcal{H}}) \subset \mathcal{G}$ means that $\mathcal{V}_{\mathcal{H}} \subset \mathcal{V}$ and $\mathcal{E}_{\mathcal{H}} \subset \mathcal{E}$. A subgraph containing edges incident to all nodes in $\mathcal{V}$ is termed a *spanning* subgraph.

Two vertices i and j are called adjacent (or neighbors) when $\{i, j\} \in \mathcal{E}$; we denote this by writing $i \sim j$. An orientation of an undirected graph $\mathcal{G}$ is the assignment of directions to its edges, i.e., an edge e_k is an ordered pair (i, j) such that i and j are, respectively, the initial and the terminal nodes of e_k. A *simple graph*, which will be used predominately in this note, can contain only a single edge between incident nodes.[3] We finally note that graphs are typically drawn using circles to represent nodes, and lines to represent edges.

There are many properties that one can derive from the basic structure of a graph. The *neighborhood* of a vertex $v \in \mathcal{V}$, denoted $\mathcal{N}(v)$, is the set of all nodes adjacent to it; that is, $\mathcal{N}(v) = \{u \in \mathcal{V} \,|\, \{v, u\} \in \mathcal{E}\}$. From the neighborhood, it is possible to define the *degree* of a node as $d(v) = |\mathcal{N}(v)|$.[4] Of particular interest to problems like the bridges of Königsberg, is to characterize how one can traverse through a graph by moving from one node to another using the edges in the graph. A *path* in a graph is a sequence of distinct adjacent vertices, while a *walk*, on the other hand, is a sequence of nondistinct adjacent vertices. We say that an undirected graph is *connected* if there is a path between every pair of nodes. A *closed walk* is a sequence of adjacent vertices that start and end with the same vertex, and a *circuit* is a closed walk that uses distinct edges. Euler proved that a circuit in a graph exists if and only if the degree of each node is even. Graphs that poses this structure are said to have an *Eulerian Circuit*, in Euler's honor [4].

[2]We will use i and v_i interchangeably to denote a vertex.

[3]Whereas a *multi-graph* may have multiple edges between a pair of nodes.

[4]For a finite set A, $|A|$ denotes its cardinality.

Paths can also be defined for directed graphs. A *directed path* of length $1 \leq k \leq n$ in $\mathcal{G}$ is a sequence of vertices $p = (v_1, v_2, \ldots, v_k)$, such that $(v_i, v_{i+1}) \in \mathcal{E}$ for $1 \leq i < k$. For the graph depicted in Figure 3, $(1, 2, 3)$ is a path, whereas $(1, 2, 3, 4)$ is not. We say that a node v belongs to p, and write $v \in p$ if v is in the sequence of nodes of p. Reciprocally, we say that the path p visits v if $v \in p$. A path p is called *simple* if it does not visit the same node more than once.

B. Trees, cycles, and decompositions

The notion of paths and closed walks also form the basis of two fundamental sub-structures of a graph. A *cycle* is a closed walk where no vertices or edges are repeated,[5] and a *tree* is a connected graph that contains no cycles. Figure 1 shows an example of a graph on four nodes, one possible spanning tree, and a subgraph containing a cycle.

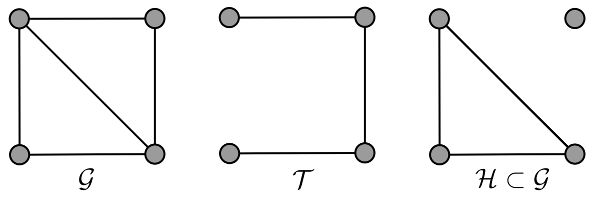


Fig. 1. A graph $\mathcal{G}$, its spanning tree subgraph $\mathcal{T}$, and a disconnected subgraph $\mathcal{H} \subset \mathcal{G}$ containing a cycle.

A *directed cycle* of length k in the directed graph $\mathcal{G}$ is a directed path of the form $(v_1, \ldots, v_k, v_1)$, and a cycle $p = (v_1, \ldots, v_k, v_1)$ is called simple if the path $(v_1, \ldots, v_k)$ is simple, and $(v_k, v_1) \in \mathcal{E}$. A loop or self-loop in $\mathcal{G}$ is an edge $(v_i, v_i) \in \mathcal{E}$, or a cycle of length 1.

An important property of any connected graph is that it contains at least one subgraph with edges incident to all the nodes. Let $\mathcal{T} = (\mathcal{V}, \mathcal{E}_{\mathcal{T}}) \subset \mathcal{G}$ be a spanning tree of $\mathcal{G}$ and consider an edge $e = (v, u) \in \mathcal{E} \setminus \mathcal{E}_{\mathcal{T}}$. Then $\mathcal{E}_{\mathcal{T}} \cup e$ forms a *fundamental cycle*, $\mathcal{C}_e$, in $\mathcal{G}$. Note that there is a path from node u to v using only edges in $\mathcal{T}$, and the edge e completes the cycle by joining v back to u. The *co-tree* of the spanning tree $\mathcal{T}$ is the graph $\mathcal{C} = (\mathcal{V}, \mathcal{E} \setminus \mathcal{E}_{\mathcal{T}})$, consisting of all the edges forming fundamental cycles with the spanning tree $\mathcal{T}$.

Beyond identifying the important subgraphs of a graph, there are many other important characterizations of graph structures. One such problem is that of graph *decompositions*, which aims to partition the nodes or edges of a graph into sets satisfying certain properties. One basic example of a decomposition problem is to determine if there exists a partition of the node set into two sets, $\mathcal{V}_1$ and $\mathcal{V}_2$, with $\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$ such that the edge set can be expressed as $\mathcal{E} = \{\{u, v\} \,|\, u \in \mathcal{V}_1,\, v \in \mathcal{V}_2\}$. These graphs are known as the *bipartite graphs*, see Figure 2.

Bipartite graphs have important relation to other decompositions. For example, a *coloring* of a graph is the assignment of unique colors to each vertex such that adjacent vertices do not share the same color. Of interest is to find the smallest

[5]Except, of course, the start and end vertex.

coloring (i.e., using the least number of colors) of a graph– this is called the *chromatic number* of a graph. Bipartite graphs thus admit a 2-coloring of the nodes, shown in Figure 2. One may also look for the smallest *edge coloring*, which is to assign a color to each edge such that adjacent edges do not share the same color, also shown in Figure 2. In bipartite graphs, the minimum edge coloring is equal to the largest vertex degree in the graph, $\max_i d(i)$, and is known as the *chromatic index* of a graph.

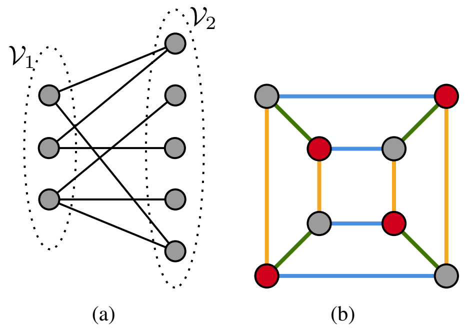


Fig. 2. Examples of bipartite graphs. The graph in (b) shows the 2-coloring of the nodes, and 3-coloring of the edges.

Edge coloring of a graph is also a special type of *matching*. A matching is a collection of non-adjacent edges, and a maximum matching is a matching that includes the largest number of edges. A matching is called perfect if every node in the graph is incident to an edge of the matching. Thus, the edge coloring is a decomposition of a graph into disjoint matchings. This problem can be cast as an optimization problem with complexity $\mathcal{O}(|\mathcal{V}||\mathcal{E}|)$ [5]. A dual problem to the maximum matching is the *minimum vertex cover* problem, which aims to find a set of vertices such that each edge of the graph is incident to at least one vertex of the set. This problem can be cast as an integer linear program, and for general graphs is classified as NP-hard. A fascinating result related to bipartite graphs is *König's Theorem*.

Theorem 1 ([6]). *For bipartite graphs, the size of the minimum vertex cover and maximum matching are equal.*

This result implies a polynomial time algorithms for solving either problem. Note that the graph in Figure 2 has a minimum vertex cover of 4, which is also equal to the maximum matching.

Another type of graph decompositions aims to find special paths in the graph. We say that a set of directed paths $p_1, \ldots, p_j$ *covers* the graph $\mathcal{G}$ if every node of $\mathcal{G}$ is visited by at least one path p_i, $1 \leq i \leq j$. Given two paths $p_1, p_2 \in \mathcal{G}$, we say that they are *disjoint* if the set of nodes they visit are distinct. For the graph of Figure 3, the cycles $(3, 6, 5, 3)$ and $(1, 2, 1)$ are disjoint, but $(1, 2, 1)$ and $(2, 3, 2)$ are not. We call $p_1, \ldots, p_j$ a *disjoint cover* if it is a cover by disjoint paths, and a *disjoint cycle cover* if it is a cover by disjoint cycles.

We now introduce the type of graph decompositions that appear in the study of stability of decentalized systems.

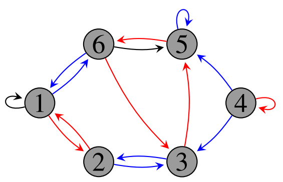


Fig. 3. A directed graph with 6 vertices. The red edges illustrate the disjoint cycle cover of the graph $(12)(356)(4)$. Other disjoint cycle covers are $(12356)(4)$ and $(23)(56)(1)(4)$. A 4-decomposition of the graph is given by $(23)(56)$, or $(1)(635)$.

We call a *k-cycle (disjoint) decomposition* of $\mathcal{G}$, or k-decomposition in short, a set of disjoint simple cycles in $\mathcal{G}$ whose union visits *exactly* k vertices. Hence a n-decomposition is a disjoint cycle cover of $\mathcal{G}$, a $1 < k < n$-decomposition is a disjoint cycle cover of a subgraph of $\mathcal{G}$ of cardinality k, and a 1-decomposition is a node with a loop.

We relate below simple cycle in a graph with n vertices with elements of the so-called permutation group S_n, and we adopt here a notation that makes this connection more transparent: we denote the cycle $(v_1, \ldots, v_k, v_1)$ by $(v_1 v_2 \cdots v_k)$, and we denote a cycle decomposition as a product of cycles. For example $(12)(3)$ refers to the 3-decomposition containing the cycle $(1, 2, 1)$ and the loop $(3, 3)$. We illustrate some of these notions in Figure 3.

We now relate cycles in digraphs to matchings in bipartite graphs as they were introduced in Sec. II-A. To this end, we first exhibit a *natural* bipartite graph associated to a digraph. Given a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define the bipartite graph $\mathcal{G}^2 = (\mathcal{V}^2, \mathcal{E}^2)$ with V^2 and E^2 defined as follows: if $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$, we set

$$\mathcal{V}^2 = \{v_1, v_2, \ldots, v_n, v_{1'}, v_{2'}, \ldots, v_{n'}\} \tag{1}$$

and

$$\mathcal{E}^2 = \{(i, j') \text{ for } (i, j) \in E\}. \tag{2}$$

It is clear from its definition that the graph G^2 is a bipartite graph, with edges going from $\mathcal{V} = \{v_1, \ldots, v_n\}$ to $\mathcal{V}' = \{v_{1'}, \ldots, v_{n'}\}$. We have the following correspondence:

Lemma 1. *Disjoint cycle covers of $\mathcal{G}$ are in one-to-one correspondence with perfect bipartite matchings of $\mathcal{G}^2$.*

We illustrate the definition of $\mathcal{G}^2$ and the correspondence of Lemma 1 in Figure 4.

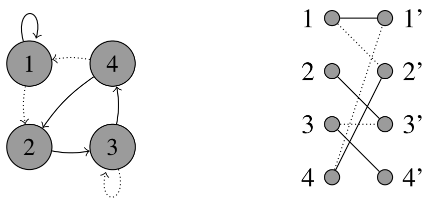


Fig. 4. The bipartite graph to the right has two nodes for each node i of the graph on the left, labeled i and i'. A directed edge (k, l) of G corresponds to an edge (k, l') in G^2. The "plain" edges show a 4-decomposition of G and the corresponding perfect matching of G^2.

This very short overview of some notions from graph theory is meant to illustrate the richness of the field, and is by no means comprehensive.

C. Graphs and their algebraic representations

The subject of algebraic graph theory aims to explore how properties of graphs can be related, or revealed, by the algebraic properties of certain matrices that can be associated to graphs. In this section, we will highlight some of the important matrix representations of graphs along with their interpretations.

Observe that any square matrix can be identified with a graph by associating each column/row with a vertex, and defining an edge between vertices whenever the corresponding matrix entry is non-zero.[6] We denote the graph of a matrix M as $\mathcal{G}(M)$. Similarly, a graph can therefore be directly associated to an appropriately defined matrix. This leads to a natural representation of a graph, and is most clearly demonstrated with what is termed the *adjacency matrix* of a graph, denoted $A(\mathcal{G})$. The adjacency matrix is defined such that

$$[A(\mathcal{G})]_{ij} = \begin{cases} 1, & (i,j) \in \mathcal{E} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Note that for undirected graphs, the adjacency matrix is symmetric.

One of the profound results relating properties of a graph to that of the corresponding matrix is the Perron-Frobenius Theorem.[7]

Theorem 2 ([7]). *A square matrix M is irreducible if and only if the graph of the matrix, $\mathcal{G}(M)$, is strongly connected.*

We recall that a matrix is irreducible if it can not be placed to an upper block-triangular form using simultaneous row/column permutations. Furthermore, this result pertains generally to directed graphs, of which undirected graphs are a special case. For directed graphs, strong connectedness means there exists a directed path between every pair of vertices.

The adjacency matrix can also be used to recover the degree information of the graph. Let $\mathbb{1}$ be the vector of all ones, then $[A(\mathcal{G})\mathbb{1}]_i = d(i)$ is the degree of node i. That is, the row sum of the adjacency matrix is precisely the degree of each node. Another useful matrix related to the node degree is the *degree matrix*, denoted as $\Delta(\mathcal{G})$ and defined as $[\Delta(\mathcal{G})]_{ii} = d(i)$. Thus, $\Delta(\mathcal{G})$ contains the degree of each node on the diagonal. It further follows that $\Delta(\mathcal{G})\mathbb{1} = A(\mathcal{G})\mathbb{1}$ and $\mathbb{1}^\top \Delta(\mathcal{G})\mathbb{1} = 2|\mathcal{E}|$.

The adjacency matrix and degree matrix can be used to define one of the most celebrated algebraic representations of a graph–the graph Laplacian matrix, $L(\mathcal{G})$. The Laplacian is defined as

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

Our discussion here focuses on undirected graphs, but the Laplacian can also be defined for directed graphs using the adjacency matrix and the in- or out-degree matrices.

[6]Note that in this case we can obtain a graph with a *self-loop*–an edge connecting a vertex to itself. This corresponds to a non-zero entry on the diagonal of the matrix.

[7]The theorem presented here represents only a partial statement of the complete result.

We can observe some immediate properties of the graph Laplacian. First, it is a symmetric matrix and therefore has real eigenvalues. From the properties of $\Delta(\mathcal{G})$ and $A(\mathcal{G})$, it also immediately follows that $L(\mathcal{G})\mathbb{1} = 0$. That is, 0 is an eigenvalue associated with the eigenvector $\mathbb{1}$. We also observe that the quadratic form of the Laplacian can be expressed as

$$\begin{aligned} x^\top L(\mathcal{G})x &= x^\top(\Delta(\mathcal{G}) - A(\mathcal{G}))x \\ &= \sum_{i=1}^{|\mathcal{V}|} \left(d(i)x_i^2 - \sum_{\{i,j\}\in\mathcal{E}} x_i x_j \right) = \sum_{\{i,j\}\in\mathcal{E}} (x_i - x_j)^2. \end{aligned}$$

Thus, the Laplacian is a positive semi-definite matrix. With this observation, we denote the eigenvalues of $L(\mathcal{G})$ as $\lambda_i(\mathcal{G})$, and have $0 = \lambda_1(\mathcal{G}) \leq \lambda_2(\mathcal{G}) \leq \cdots \leq \lambda_{|\mathcal{V}|}(\mathcal{G})$. This leads to an important result relating the eigenvalues of $L(\mathcal{G})$ to the connectedness of $\mathcal{G}$.

Theorem 3 ([8]). *An undirected graph $\mathcal{G}$ is connected if and only if $\lambda_2(\mathcal{G}) > 0$.*

The spectrum of the Laplacian is also related to other combinatorial properties of the underlying graph. For example, given a connected graph $\mathcal{G}$, one may be interested in enumerating the number of spanning trees, $t(\mathcal{G})$, it contains. This number can certainly be found by counting, and Cayley's formula provides the counting result for the complete graph on n nodes as $t(\mathcal{G}) = n^{n-2}$, which is exponential. On the other hand, for an arbitrary (connected) graph $\mathcal{G}$, the number of spanning trees can be expressed as

$$t(\mathcal{G}) = \frac{1}{n}\prod_{i=2}^{n} \lambda_i(\mathcal{G});$$

this is known as *Kirchoff's Matrix-Tree Theorem* [9]. This result is remarkable considering that $t(\mathcal{G})$ must be an integer value, while the eigenvalues of $L(\mathcal{G})$ may be irrational.

We now turn our attention to perhaps the most descriptive algebraic representation of a graph - the graph incidence matrix. The incidence matrix, denoted here by $E(\mathcal{G})$, is indexed by the vertices of $\mathcal{G}$ for the rows, and the edges of $\mathcal{G}$ for the columns - so $E(\mathcal{G}) \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$. To define the incidence matrix, we first assign an arbitrary orientation to the edges, leading to a directed graph. We label the edges as $\mathcal{E} = \{e_1, \ldots, e_{|\mathcal{E}|}\}$. Then, $[E(\mathcal{G})]_{ij} = +1$ if node i is the initial node of edge e_j, $[E(\mathcal{G})]_{ij} = -1$ if node i is the terminal node of e_j, and $[E(\mathcal{G})]_{ij} = 0$ otherwise.

From the structure of the incidence matrix, it is immediately apparent that $\mathbb{1}^\top E(\mathcal{G}) = 0$. Moreover, the transpose of the incidence matrix can therefore be interpreted as a *difference* operator over a graph. This interpretation leads to the following result relating the rank of the incidence matrix and the number of connected components in the graph.

Theorem 4 ([4]). *Let $\mathcal{G}$ be a graph on $|\mathcal{V}| = n$ nodes with c connected components. Then $rk[E(\mathcal{G})] = n - c$.*

In fact, there is an intimate connection between the fundamental subspaces of the incidence matrix $E(\mathcal{G})$ and the combinatorial structure of the underlying graph. In this direction, we examine the graph-theoretic interpretation of the kernel of $E(\mathcal{G})$, known as the *cycle space* (or *flow space*). The cycle space of $\mathcal{G}$, can be expressed as

$$\text{Ker}[E(\mathcal{G})] = \{x \in \mathbb{R}^{|\mathcal{E}|} \mid E(\mathcal{G})x = 0\}.$$

One may now consider a vector $x \in \text{Ker}[E(\mathcal{G})]$ describing a commodity flowing through each edge from its initial node to its terminal node. This commodity should be conserved, meaning the flow entering each node must equal the flow leaving the node. For example, a conserved flow on a spanning tree can only be the "zero flow," meaning $E(\mathcal{T})x = 0$ if and only if $x = 0$. More generally, a conserved flow in $\mathcal{G}$ corresponds to flows around fundamental cycles to be zero - this is also referred to as *Kirchoff's Current Law*. As a result, one can always construct a conserved flow by examining the flows around the fundamental cycles in the graph.

Theorem 5 ([4])**.** *The cycle space of $\mathcal{G}$ is spanned by fundamental cycles of $\mathcal{G}$.*

This result can be seen algebraically by partitioning the columns of $E(\mathcal{G})$ into those corresponding to the edges of a spanning tree, $\mathcal{T}$, and the remaining edges that complete the fundamental cycles (the co-tree $\mathcal{C}$). By an appropriate labelling of the edges, we can then write

$$E(\mathcal{G}) = \begin{bmatrix} E(\mathcal{T}) & E(\mathcal{C}) \end{bmatrix}.$$

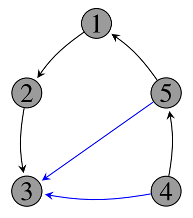
Since $E(\mathcal{T})$ is a matrix with full column rank, we can express the incidence matrix of the co-tree as a linear combination of $E(\mathcal{T})$. Thus,

$$E(\mathcal{C}) = E(\mathcal{T})R. \tag{4}$$

The matrix R, in some communities referred to as the *Tucker representation* of a graph [5], describes which edges in $\mathcal{T}$ are needed to form the fundamental cycle with edges in $\mathcal{C}$. This description leads to the following characterization of $E(\mathcal{G})$,

$$E(\mathcal{G}) = E(\mathcal{T}) \begin{bmatrix} I & R \end{bmatrix}.$$

This is illustrated in Figure 5.



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

$$R = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

Fig. 5. A graph with an arbitrary orientation. A spanning tree is indicated by the black edges, and the co-tree by the blue edges.

A basis for the flow space can therefore be expressed in terms of the rows of the matrix $\begin{bmatrix} -R^\top & I \end{bmatrix}$,

$$\text{Ker}[E(\mathcal{G})] = \text{Im} \begin{bmatrix} -R \\ I \end{bmatrix}.$$

Finally, we mention a relationship between the incidence matrix and the Laplacian matrix of $\mathcal{G}$. Indeed, it can be verified that

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

holds. While $L(\mathcal{G})$ is defined for undirected graphs, we note the orientation of edges used to define $E(\mathcal{G})$ can be arbitrary. This representation also motivates an alternative matrix representation of a graph, mapping "edges to edges." This matrix, termed the *edge Laplacian*, can be expressed as [10],

$$L_e(\mathcal{G}) = E(\mathcal{G})E(\mathcal{G})^\top.$$

A useful result relating the graph and edge Laplacian matrices is stated below.

Proposition 1 ([10])**.** *For a connected graph $\mathcal{G}$ with spanning tree $\mathcal{T}$ and co-tree $\mathcal{C}$, the graph Laplacian $L(\mathcal{G})$ is similar to the matrix*

$$\begin{bmatrix} L_e(\mathcal{T})(I + RR^\top) & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix},$$

where $L_e(\mathcal{T})$ is the edge Laplacian associated with $\mathcal{T}$, and R is defined in (4).

The matrix $L_e(\mathcal{T})(I+RR^\top)$ is referred to as the *essential edge Laplacian*, and we denote it as $L_{ess}(\mathcal{G})$ [11].

III. Graphs and Structural Stability of Networks

In the spirit of results such as Theorem 2 that highlight the combinatorial aspects of matrix theory, in this section we explore how dynamic systems can be represented by graphs, and how that graph structure can be used to assess the stability properties of the system.

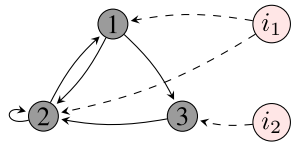
A. Graphs and system's structure: zero-patterns

We start by defining precisely how a graph describes the structure of a system, as mentioned in the Introduction. Let $n > 0$ be a positive integer, we call a *sparse matrix space* (SMS) or *zero pattern* a vector space of matrices in $\mathbb{R}^{n \times n}$ with entries either free or zero. Precisely, we have the following definition:

Definition 1 (Zero pattern)**.** *Let α be a set of pairs of integers between* 1 *and n, that is $\alpha \subset \{1, \ldots, n\} \times \{1, \ldots, n\}$ and denote by E_{ij} the $n \times n$ matrix with zero entries except for the ijth entry, which is one. We define the* zero pattern *Σ_α to be the vector space of matrices of the form $A = \sum_{(i,j)\in\alpha} a_{ij}E_{ij}$ for $a_{ij} \in \mathbb{R}$.*

A zero-pattern associated to the actuator matrix B of a linear system can similarly be defined via $\beta \subseteq \{1, \ldots, n\} \times \{1, \ldots, m\}$. For example, if $n = 3, m = 2$ and $\alpha = \{(1,2),(1,3),(2,1),(2,2),(3,2)\}, \beta = \{(1,1),(1,2),(2,3)\}$, then $A \in \Sigma_\alpha$ and $B \in \Sigma_\beta$ are given in Figure 6, where $*$ are arbitrary real values. We call the elements referred to in α as *free variables* or *free entries*, and the other elements are *zero variables* or *zero entries*.

Building on Sec. II-C, we associate a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to a zero-pattern Σ_α uniquely as follows: the vertex set $V = \{1, 2, \ldots, n\}$ and edge set $E = \alpha$. Note that this is similar to considering the matrix A in (III-A) to be the adjacency matrix (see Eq. (3), where $*$ are to be interpreted as 1). If there is an actuation vector $B \in \Sigma_\beta$, we modify the above graph as follows: to each column of B is associated a so-called "input-node," and there is a directed edge from an input node i to node j if the entry $(i, j) \in \beta$. Hence the graphical representation of the zero patterns above is given by the graph in Figure 6.



$$A = \begin{pmatrix} 0 & * & * \\ * & * & 0 \\ 0 & * & 0 \end{pmatrix}, B = \begin{pmatrix} * & 0 \\ * & 0 \\ 0 & * \end{pmatrix}$$

Fig. 6. A zero-pattern and its corresponding digraph.

We call a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *symmetric* if for all $(i, j) \in \mathcal{E} \Leftrightarrow (j, i) \in \mathcal{E}$: every edge in the graph has its reciprocal in the graph. We can represent these graphs as *undirected graphs* with self-loops. We associate two zero-patterns to undirected graphs as follows: first, from the general correspondence between graphs and zero-patterns introduced above, it is easy to see that to an undirected graph corresponds a zero-pattern such that if entry i, j is free, so is entry j, i. We call a zero-pattern with this property *weakly symmetric*. Note that matrices in a weakly symmetric zero-pattern are not necessarily symmetric. If a zero-pattern contains only symmetric matrices, we call it *strongly symmetric*. We summarize the above discussion in the following definition:

Definition 2 (Symmetric and weakly symmetric zero-patterns)**.** *A zero-pattern Σ defined by $\alpha \subset \{1, \ldots, n\} \times \{1, \ldots, n\}$ is* weakly symmetric *or simply* symmetric *if $(i, j) \in \alpha \Leftrightarrow (j, i) \in \alpha$. A* strongly symmetric *zero-pattern is a symmetric zero-pattern which only contains symmetric matrices, that is $\Sigma = \sum_{(i,j)\in\alpha} a_{ij} E_{ij}$ with $a_{ij} = a_{ji}$.*

Symmetric and weakly symmetric zero-patterns can be represented using undirected graphs, whereas general zero-patterns are represented by digraphs.

B. Structural stability of linear Systems

We illustrate how graph theoretic notions enter in the study of decentralizaed stabilization. Recall that a matrix is called *Hurwitz* or *stable* if all of its eigenvalues have negative real parts. We introduce the following definition:

Definition 3 (Stable graphs and stable zero-patterns)**.** *We say that a* zero pattern is stable *if it contains Hurwitz matrices. We refer to its associated graph as a* stable graph*.*

Our goal is to obtain conditions to characterize stable zero-patterns or graphs, and algorithm to create such stable graphs. To this end, we call a sequence of k-decompositions for $1 \le k \le n$ *nested* if the i-decomposition covers all vertices covered by the $i-1$-decomposition plus one additional vertex. Observe that the edges used in each decompositions need not be the same. For example, the decompositions $(2), (12), (123)$ for the graph in Figure 6 (left) are nested.

1) The general case:

Theorem 6 ([12])**.** *The following holds:*

1) Necessary condition for stability: *If a graph $\mathcal{G}$ is stable, then it contains at least one k-decomposition for every $1 \le k \le n$ and every node of the graph is strongly connected to at least one self-loop.*
2) Sufficient condition for stability: *If a graph $\mathcal{G}$ contains a sequence of nested k-decompositions for $1 \le k \le n$ then the associated zero-pattern is stable.*

We illustrate the result on the following zero-pattern,

$$\Sigma = \begin{pmatrix} * & * & 0 & 0 & 0 & * \\ * & 0 & * & 0 & 0 & 0 \\ 0 & * & 0 & 0 & * & 0 \\ 0 & 0 & * & * & * & 0 \\ 0 & 0 & 0 & 0 & * & * \\ * & 0 & * & 0 & * & 0 \end{pmatrix} \tag{5}$$

whose corresponding graph is in Figure 3. We first verify that every node is strongly connected to a node with a self-loop. Since $1, 4$ and 5 have self-loops, it suffices to verify the statement for nodes $2, 3$ and 6. We see that these tree nodes are strongly connected to 1. We now exhibit k-cycle disjoint decompositions for $1 \le k \le 6$: $k = 1$: (5); $k = 2$: (56); $k = 3$: $(16)(5)$; $k = 4$: $(356)(1)$; $k = 5$: $(23)(56)(1)$; $k = 6$: $(23)(56)(1)(4)$.

Hence, the above zero-pattern meets the necessary condition of Theorem 6. Now observe that the covers exhibited above are in fact nested, since the nodes of a k-cover and included in the nodes of a $k+1$ cover. Hence, the necessary condition is also met and there thus exists an assignment of the $*$ in (5) so that the resulting matrix is stable.

2) The symmetric case: We start with strongly symmetric zero-patterns, that is vector spaces of symmetric sparse matrices. This case is easily dealt with by the following result:

Lemma 2. *Let Σ be a strongly symmetric zero-pattern. Then Σ is stable if and only if all its diagonal elements are free.*

We now focus to the more general case of (Weakly) symmetric zero patterns. We show that the necessary and sufficient conditions for stability of Theorem 6 coincide in this case, thus completely characterizing the weakly symmetric sparse matrix spaces:

Theorem 7 ([13])**.** *Let $\mathcal{G}$ be a graph corresponding to a (weakly) summetry zero-pattern Σ. Then Σ is stable* if and only if*:*

1) *Every node in $\mathcal{G}$ is strongly connected to a self-loop.*
2) *The graph $\mathcal{G}$ contains a disjoint cycle cover.*

We make two remarks. First, the notion of strong connectedness is redundant in the symmetric case–indeed, if there exists a path from vertex u to vertex v in the graph, then there exists a path going in the other direction by symmetry. Second, notice that conditions in Theorem 7 are weaker than the necessary conditions of Theorem 6—namely, we do not require the existence of k-decompositions for every $1 \leq k \leq n$. However, conditions of Theorem 7 imply the sufficient conditions of Theorem 6, that is, we have the following result:

Proposition 2 ([13])**.** *If the symmetric graph $\mathcal{G}$ contains a disjoint cycle cover, and if every node in $\mathcal{G}$ is connected to a self-loop, then $\mathcal{G}$ admits nested k-decompositions for $1 \leq k \leq n$.*

It is clear that Theorem 7 is a direct consequence of Proposition 2. We illustrate the result to the symmetric graphs shown below.

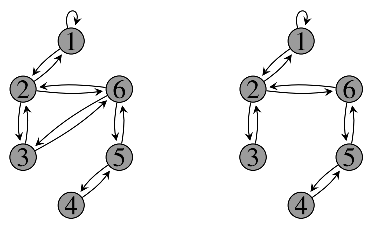


Fig. 7. Even though both graphs are connected and contain self-loops, the first one has a n-decomposition $(1), (236), (45)$, whereas the second one does not. Therefore only the graph on the left is stable.

The proof is entirely graph theoretic and relies on the fact that a *planar subgraph* can be exhibited in the graph of the zero-pattern. We provide a sketch here:

Proof of Proposition 2. First, it is a simple exercise to see that we can assume that $\mathcal{G}$ is connected. By hypothesis, $\mathcal{G}$ contains at least one n-decomposition. We pick one arbitrarily and denote by H_0, H_1, ... , H_m its constituent cycles. Let v_0 be a vertex of $\mathcal{G}$ incident to a self-loop and without loss of generality, assume that $v_0 \in H_0$. We now show that we can discard edges of $\mathcal{G}$ to reduce it to a *planar graph $\mathcal{P}$ that still satisfies the conditions of the theorem.* This reduction to a planar graph allows us to construct the desired nested k-decompositions and prove stability of $\mathcal{G}$. We construct $\mathcal{P}$ in three steps. Starting from a disconnected, planar subgraph of $\mathcal{G}$, we expand it by adding edges from $\mathcal{G}$ without creating intersections.

1) The cycles $H_0, \ldots, H_m$ do not have any nodes or edges in common by definition. We draw these cycles in the plane without any intersecting edges. We call the resulting graph $\mathcal{P}_0$. Note that $\mathcal{P}_0$ has $m+1$ disconnected components, see Figure 8.
2) Add the smallest number of edges (of $\mathcal{G}$) to $\mathcal{P}_0$ so that all the nodes of $\mathcal{P}_0$ are connected to the node $v_0 \in H_0$. We can do this by adding exactly m edges, as can be observed in Figure 8.
3) Finally, we add all missing reciprocal edges to the ones we have already chosen, to obtain a symmetric graph - we clearly can do this without intersecting any edges. We draw the newly added edges with dashed lines. The resulting planar graph is $\mathcal{P}$, and isomorphic to a subgraph of $\mathcal{G}$ by construction, see Figure 9.

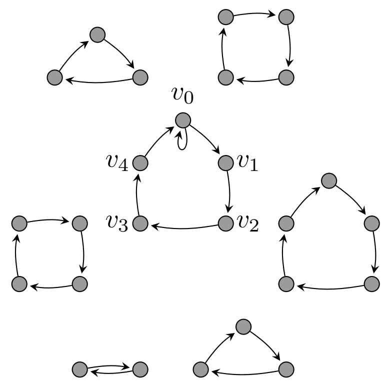


Fig. 8. The cycles of a Hamiltonian decomposition of G can be plotted in the plane without intersecting edges. The graph above is called P_0.

We now show that the graph $\mathcal{P}$ contains nested k-decompositions, and thus $\mathcal{G}$ does as well. We will exhibit such k-decompositions explicitly by giving an ordering of the nodes such that the subgraph of $\mathcal{G}$ induced by the first k nodes in the order admits a k-decomposition.

We describe the order using an integer-valued function F which assigns to each node its position in the ordering. We set $F(v_0) = 1$ and proceed with the ordering in a counter-clockwise fashion. This defines a unique order once the second node in the sequence is chosen. There is a canonical way of choosing it described in [13]. Because $\mathcal{P}$ is connected and because all of its nodes lie on the boundary, every one of them will be visited in this fashion. We illustrate this in Figure 9.

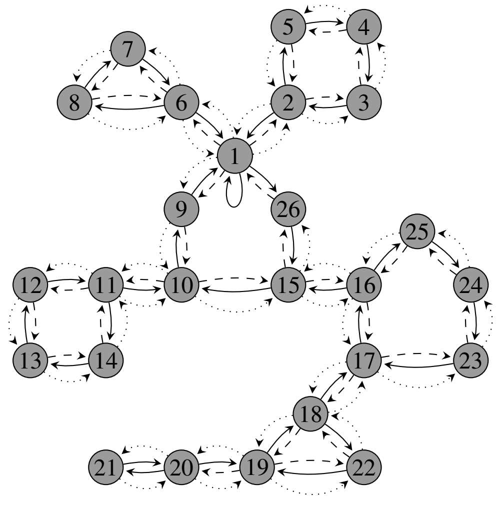


Fig. 9. Starting from node v_0 at position 1, we follow a counter-clockwise direction to assign an order to all nodes. The outside dotted arrows depict the path followed to set the ordering. By construction, no node lies inside of a cycle and thus every node will be given a position in the ordering.

Next, we verify that for each k, the set of vertices v for which $F(v) \leq k$—call it V_k—induces a k-decomposition in $\mathcal{P}$. To do this, let us first connect the vertex with label k to v_0 using a path only containing plain edges (i.e. no dashed edges). We can do this in a unique way, moving along the cycles clock-wise until we get to v_0. If we write

$w_0, \dots, w_{k-1}$ for the nodes contained in this path (with $w_0 = v_0$ and $F(w_{k-1}) = k$) then $V_k - \{w_0, \dots, w_{k-1}\}$ contains exactly all the nodes of some cycles H_l. Indeed, because we constrain the path to use plain edges, if some vertex w_i belongs to a cycle H_j, then all vertices from H_j with labels smaller than i also belong to the path $w_0, \dots w_{i-1}$.

Therefore if we show that the nodes belonging to the path $w_0, \dots, w_{k-1}$ can be grouped in simple cycles with edges from $\mathcal{P}$, we would have exhibited a k-decomposition. Consider two cases: 1: if the length of the path is $2l$, then we can just group the consecutive nodes in pairs, which yields l disjoint 2-cycles. 2: If the length of the path is odd, then we can do the same, but this time leaving the vertex v_0 out. However, v_0 is incident to a self-loop, so this yields a k-decomposition as well. □

The connection to graph theory allows us to easily derive bounds on the complexity of deciding stability of a weakly-symmetric zero-pattern: recall the construction of Sec. II-B assigning a bipartite graph $\mathcal{G}^2$ uniquely to a digraph $\mathcal{G}$. From Lemma 1, we know that n-decompositions of $\mathcal{G}$ correspond to perfect matchings of $\mathcal{G}^2$. Furthermore, we also mentioned in Sec.II-A the existence of polynomial-time algorithm to find maximal matchings. We thus obtain the following result.

Corollary 1. *There exists a polynomial-time algorithm to verify whether a weakly-symmetric zero-pattern is stable.*

IV. Graphs and Controllability of Networks

As mentioned earlier, graphs can be used to describe either the stucture of an underlying system, or to describe a discrete topology over which a dynamic process such as diffusion takes place. In both cases, the study of controllability can be done through graph theoretic techniques. We start with the structural case followed by the controllability analysis of diffusion dynamics.

A. Structural controllability of linear systems

The study of structural controllability of linear systems was one of the first instances of the use of graph theoretic notions in the study of control systems. The problem is the following: given zero patterns Σ_α and Σ_β, is there a pair $(A, B) \in \Sigma_\alpha \times \Sigma_\beta$ so that the system $\dot{x} = Ax + Bu$ is controllable? The problem was studied by CT Lin [14] in the single input case, and was generalized to the multi-input case in, e.g., [15]. The first result was to show that the controllable property is *generic* for a pair of zero-patterns, that is if there exist *one* controllable pair $(A_0, B_0) \in \Sigma_\alpha \times \Sigma_\beta$, then *almost all* pairs $(A, B) \in \Sigma_\alpha \times \Sigma_\beta$ are controllable.

Before stating the result of Lin in graph theoretic terms, we need to introduce a particular type of graph, called a *cactus graph*. To this end, consider a pair $\Sigma_\alpha, \Sigma_\beta$ where the number of columns in Σ_β is one. A *stem* is a graph associated to the pair

$$A = \begin{pmatrix} 0 & * & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & & & * \\ x & 0 & \cdots & 0 \end{pmatrix}, B = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ * \end{pmatrix} \tag{6}$$

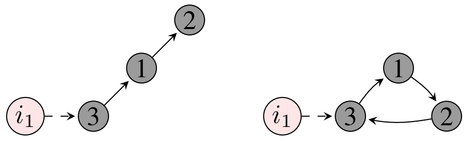


Fig. 10. A stem graph (left) and a bud graph (right).

with $*$ in A only in entries immediately above the main diagonal, with $x = 0$, and a *bud* is a graph associated to the pair of Eq,(6) with $x = *$. We depict them in Figure 10. The node i_1 is called the *origin* of the stem/bud.

The graph $\mathcal{G}$ of the pair $\Sigma_\alpha, \Sigma_\beta$ is a cactus if it can be written as $\mathcal{G} = S \cup B_1 \cup \cdots \cup B_p$ where S is a stem, the B_i's are buds and the origin e_i of the bud B_i is also the origin of an oriented edge in $S \cup B_1 \cdots \cup B_{i-1}$ and e_i is the only vertex which belongs to both B_i and $S \cup B_1 \cdots \cup B_{i-1}$. Intuitively, one should think of the buds as being attached to the stem (through their orgin node) to form a cactus. See [14] for more details. The main result of Lin provides the following characterization of controllable zero-patterns pairs $\Sigma_\alpha, \Sigma_\beta$.

Theorem 8 ([14])**.** *There is a controllable pair $(A, B) \in \Sigma_\alpha \times \Sigma_\beta$ if and only if the associated graph is spanned by a cactus.*

Structural controllability of networks has recently been a subject of great interest, as Theorem 8 leads to connections with matching and rank optimization problems; see [16].

B. Network diffusion and its controllability

Linear time-invariant dynamics on graphs assume the form,

$$\dot{x} = A(\mathcal{G})x + B_1(\mathcal{G})u + B_2(\mathcal{G})w, \tag{7}$$

where x denotes the state, and the matrices A, B_1, and B_2, represent the system matrix, input matrix, and disturbance matrix, respectively. The dependency of these matrices on the underlying graph is emphasized as this dependency is exactly what distinguishes this line of investigation. An output or observation equation with the corresponding system, input, and disturbance matrices can augment the state dynamics as well, once again with the dependency of these matrices on the graph $\mathcal{G}$; see Figure 11. The correspondence between sys-

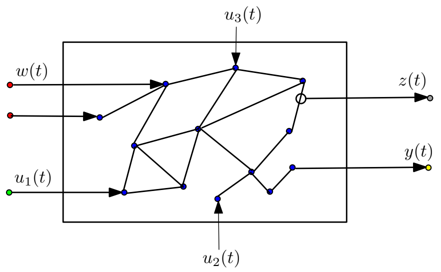


Fig. 11. Systems with a graph backbone; $z(t)$ and $y(t)$ represent the output and observation signals, respectively.

tems and graph theoretic constructs is more direct when the system matrices are in fact, one of the matrix representations of the network presented in Section II-C; this has proved to be a particularly fruitful line of investigation, when the corresponding system accomplishes a useful task and have implications in terms of its distributed nature and information overhead. An iconic example that represents this scenario, is the so-called consensus dynamics or distributed averaging that assumes the form

$$\dot{x} = -L(\mathcal{G})x, \tag{8}$$

encoding that the state of the nodes evolve according the sum of the differences between the state of the node and its neighbors. The study of consensus dynamics highlights the utility and importance of algebraic graph theory in the analysis of the system. Indeed, for the initial conditions $x_0 = x(0)$, the solution of (8) can be expressed as

$$x(t) = e^{-L(\mathcal{G})t}x_0 = \frac{1}{|\mathcal{V}|}\mathbb{1}\mathbb{1}^\top x_0 + \sum_{i=2}^{|\mathcal{V}|} e^{-\lambda_i(\mathcal{G})t} v_i v_i^\top x_0,$$

where v_i satisfies the eigenvalue/eigenvector relation $L(\mathcal{G})v_i = \lambda_i(\mathcal{G})v_i$. This leads to the following result.

Theorem 9 ([3]). *Let $\mathcal{G}$ be a connected graph. Then the consensus dynamics (8) converges to the agreement set, $\mathcal{A} = \{x \,|\, x = \alpha\mathbb{1},\ \alpha \in \mathbb{R}\}$ with a rate of convergence determined by $\lambda_2(\mathcal{G})$.*

Let us now turn our attention to controllability of diffusion dynamics on graphs. The setting involves considering a network system, such as the one shown in Figure 12, and examine whether the network is controllable from the input "a." A moment reflection on this interaction model provides our first glimpse into the potential role that network symmetry plays in the context of controllability analysis. In particular, it is intuitive to reason that if the network consists of homogeneous group of nodes with a interaction pattern with some notion of symmetry, and that furthermore, the input node is invariant under this symmetry, then the network might be uncontrollable from this input node. This intuition stems from the observation that as far as the input node is concerned, it would be impossible to steer a group of nodes that are "mirror" reflections of each other under the action of a symmetry.

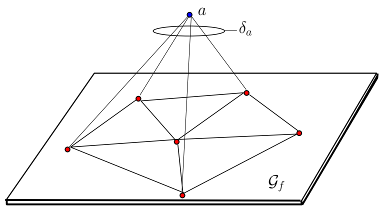
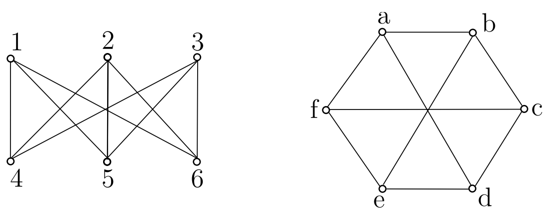


Fig. 12. A networked system with an input

In order to make this intuition precise, let us explore how the notion of symmetry can be made more precise. Graph isomorphisms provide the entry point to characterize symmetries in the graph, as they capture invariance properties under relabeling of the nodes–see for example, Figure 13. Such relabelings can be represented by a "similarity" transformation on the adjacency matrix of the graph as $P^T A(\mathcal{G})P$,



$$\theta = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ a & c & e & d & f & b \end{pmatrix}$$

Fig. 13. An example of graph isomorphism; θ represents the node relabeling between the two isomorphic graphs

where P is a permutation matrix. Graph symmetries are now made more precise through isomorphisms between the graph and itself. What that means, is that the node relabeling has to preserve the graph incidence relation. As such, the permutation matrix P characterizes the graph symmetry, referred to as an *automorphism*, when $P^T A(\mathcal{G})P = A(\mathcal{G})$, or equivalently, when

$$A(\mathcal{G})P = PA(\mathcal{G}); \tag{9}$$

in this case, we call the matrix P an automorphism.[8]

We are now ready to tie in the graph automorphism with network controllability. Consider a class of networked systems with system dynamics that commutes with the automorphism of the graph. Furthermore, we say that the input is invariant under the action of the automorphism when $PB = B$. Examples of such matrices include the adjacency, Laplacian, as well as the so-called grounded Laplacian. The key property for this setup is that these combinatorial matrices be *non-defective*.

Theorem 10 ([17]). *The pair $(A(\mathcal{G}), B(\mathcal{G}))$ is uncontrollable if $A(\mathcal{G})$ admits an automorphism for which $B(\mathcal{G})$ is invariant under its action.*

The Popov-Belevich-Hautus test for controllability turns out to be indispensible in the proof of the above statement. The steps involve starting with an eigenvalue-eigenvector pair (λ, v) for the system matrix and showing that Pv is also an eigenvector corresponding the same eigenvalue. Since P is non-trivial and $A(\mathcal{G})$ is non-defective, there has to be some eigenvector of $A(\mathcal{G})$ for which $Pv \neq v$ and as such, $v - Pv$ is also an eigenvector of $A(\mathcal{G})$. But that would then imply that $v - Pv$ is orthogonal to all columns of $B(\mathcal{G})$.

V. Graphs and Performance of Network Systems

State-controllability is a rather strong notion for large-scale networks as it is difficult to justify how (and why) the state of every node in the network should be controlled in order to influence the network. Another class of influence measures can be quantified through the notion of system norms. As we see shortly, system norms such as the $\mathcal{H}_2$ norm of the networked system in the context of influenced consensus, is closely related to various combinatorial structures of the underlying graph, including the length of fundamental

[8]More precisely, P is a representation of the automorphism.

cycles, and the so-called effective resistance of the graph. This, in turn, has been extensively studied in the context of other types of dynamic processes on graph, namely random walks.

In this direction, we consider the consensus protocol over a connected graph $\mathcal{G}$, introduced in (8), that is corrupted by zero-mean Gaussian noise at the process. In this discussion, we consider the performance variable of the network in terms of the relative states of a given set of agents - this can most generally be captured by the incidence matrix of some other network, $\mathcal{H}$, e.g., $\mathcal{H} \subseteq \mathcal{G}$ or the complete graph $\mathcal{K}_n$. The corresponding controlled consensus model now has the form

$$\left\{ \begin{array}{rl} \dot{x} = & -L(\mathcal{G})x + w \\ z = & E(\mathcal{H})^\top x. \end{array} \right. \tag{10}$$

We first note that when the consensus protocol is corrupted by noise, it will not converge to a constant steady state. Rather, the presence of noise leads to a random walk of the consensus value, where the covariance of the average of the system state becomes unbounded as $t \to \infty$ [18]. Thus, a useful performance metric for studying (10) is to consider the steady-state variance of the system trajectories orthogonal to the agreement set. This abstractly captures how tightly around the moving agreement value the states remain. Formally, this is exactly the interpretation given by the $\mathcal{H}_2$ system norm of (10) when restricted to the subspace $\mathbb{1}^\perp$.

In this direction, we employ the similarity transformation proposed in Proposition 1 to consider a transformation of (10) to the "edge dynamics," allowing to study the system trajectories orthogonal to the agreement set. This transformation can be used to eliminate the "agreement" portion of the dynamics, leading to a minimal realization of (10). Using the transformation

$$\tilde{x} = \begin{bmatrix} x_{\mathcal{T}} \\ x_{\texttt{avg}} \end{bmatrix} = \begin{bmatrix} E(\mathcal{T})^\top \\ \frac{1}{|\mathcal{V}|}\mathbb{1}^\top \end{bmatrix} x,$$

where $\mathcal{T} \subset \mathcal{G}$ is a spanning tree in $\mathcal{G}$, we obtain the *edge agreement protocol* [11],

$$\Sigma_e(\mathcal{G}) : \left\{ \begin{array}{rl} \dot{x}_{\mathcal{T}} = & -L_{ess}(\mathcal{G})x_{\mathcal{T}} + E(\mathcal{T})^\top w \\ z = & E(\mathcal{H})^\top E(\mathcal{T})^{\dagger^\top} x_{\mathcal{T}}, \end{array} \right. \tag{11}$$

where $E(\mathcal{T})^\dagger = L_e(\mathcal{T})^{-1}E(\mathcal{T})^\top$ is the left-inverse of $E(\mathcal{T})$, and $L_{ess}(\mathcal{G})$ is the essential edge Laplacian.

The edge agreement protocol has a clear interpretation showing the dynamics of the system trajectories along the edges of a given spanning tree $\mathcal{T}$. We now recall that the $\mathcal{H}_2$ performance of (11) can be computed as

$$\|\Sigma_e(\mathcal{G})\|_2^2 = \mathrm{Tr}[E(\mathcal{H})^\top E(\mathcal{T})^{\dagger^\top} X E(\mathcal{T})^\dagger E(\mathcal{H})], \tag{12}$$

where X is the positive definite solution to the Lyapunov equation

$$\mathcal{L}(X) = -L_{ess}(\mathcal{G})X - XL_{ess}(\mathcal{G})^\top + L_e(\mathcal{T}) = 0.$$

The solution of the Lyapunov equation can be verified to be

$$X = \frac{1}{2}\left(I + RR^\top\right)^{-1}.$$

Therefore, the $\mathcal{H}_2$ performance of the edge agreement system can be expressed directly using (12). What remains is to determine what combinatorial properties of the underlying graph, if any, influence this performance metric.

To begin, we first consider the scenario where $\mathcal{H} = \mathcal{K}_n$, the complete graph. For this we take a brief detour to introduce the notion of the *effective resistance* of a graph [19]. It is well known that the weighted Laplacian of a graph can be interpreted as a resistor network. Each edge in the network can be thought of as a resistor with resistance equal to the inverse of the edge weight. The resistance between any two pairs of nodes $u, v \in \mathcal{V}$, denoted $\mathcal{R}_{uv}(\mathcal{G})$, can be computed using the Moore-Penrose pseudo-inverse of the graph Laplacian, denoted $L(\mathcal{G})^\dagger$, as [19]

$$\begin{aligned} \mathcal{R}_{uv}(\mathcal{G}) &= (\mathbf{e}_u - \mathbf{e}_v)^T L(\mathcal{G})^\dagger (\mathbf{e}_u - \mathbf{e}_v) \\ &= [L(\mathcal{G})^\dagger]_{uu} - 2[L(\mathcal{G})^\dagger]_{uv} + [L(\mathcal{G})^\dagger]_{vv}, \end{aligned}$$

where $\mathbf{e}_u \in \mathbb{R}^n$ is the indicator vector for node u. Note, therefore, that $(\mathbf{e}_u - \mathbf{e}_v)$ can be thought of as the incidence matrix for a graph on $\mathcal{V}$ with a single edge (u, v). This is illustrated in Figure 14. The *total effective resistance* of a graph is the sum over all pairs of nodes of $\mathcal{R}_{uv}(\mathcal{G})$,

$$\mathcal{R}_{tot}(\mathcal{G}) = \sum_{\{u,v\} \in \mathcal{E}} \mathcal{R}_{uv}(\mathcal{G}).$$

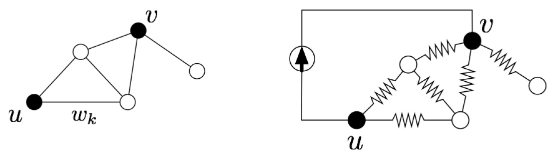


Fig. 14. Effective resistance interpretation of influence.

It turns out the effective resistance can also be calculated using the matrix R defined in (4) [20].

Proposition 3. *Consider a graph $\mathcal{G}$ with spanning tree $\mathcal{T}$ and Tucker matrix R. Let R_{uv} satisfy $(\mathbf{e}_u - \mathbf{e}_v) = E(\mathcal{T})R_{uv}$. Then the effective resistance between nodes u and v can be computed as*

$$\mathcal{R}_{uv}(\mathcal{G}) = R_{uv}^\top (I + RR^\top)^{-1} R_{uv}.$$

This result follows from Proposition 1. This can be extended to derive an expression for the total effective resistance. Let $R_{\mathcal{K}_n}$ satisfy $E(\mathcal{K}_n) = E(\mathcal{T})R_{\mathcal{K}_n}$, representing the Tucker matrix for all possible edges, then

$$\mathcal{R}_{tot}(\mathcal{G}) = \mathrm{Tr}[R_{\mathcal{K}_n}^\top (I + RR^\top)^{-1} R_{\mathcal{K}_n}].$$

It is now apparent that the $\mathcal{H}_2$ performance of the consensus protocol, when considering $\mathcal{H} = \mathcal{K}_n$, is precisely the total effective resistance of the graph,

$$\|\Sigma_e(\mathcal{G})\|_2^2 = \frac{1}{2}\mathcal{R}_{tot}(\mathcal{G}).$$

We now consider the scenario where the performance is measured along the edges of a spanning tree of $\mathcal{G}$ such that

$\mathcal{H} = \mathcal{T}$. In this case, we find that the performance reduces to the expression

$$\|\Sigma_e(\mathcal{G})\|_2^2 = \text{Tr}[(I + RR^\top)^{-1}].$$

Here we recall a few properties of the matrix R. Each column of R indicates which edges in $\mathcal{T}$ are used to form a fundamental cycle. Thus, the sum of the square of the non-zero entries in each column are in correspondence to the *length of each fundamental cycle*. We also observe that the matrix $(I + RR^\top)^{-1}$ can be iteratively computed using the Sherman-Morrison formula (the so-called "rank-1 update") [21]. Using these properties, we arrive at few results. First, we conclude that the performance is minimized for the complete graph and maximized for spanning trees. From this we immediately conclude that there is a tradeoff between the number of edges (i.e., the sparsity of the system) and its performance. Viewed in another way, the addition of cycles is the mechanism that improves the system performance.

With this characterization, one may consider a *network design* problem that is tasked with adding a fixed number of well-placed edges to obtain the largest improvement in the $\mathcal{H}_2$ performance. The following result shows that when adding a single edge to a spanning tree, the length of the created cycle influences the resulting performance.

Theorem 11 ([11]). *Consider the edge agreement problem* (11) *with $\mathcal{G} = \mathcal{H} = \mathcal{T}$ and an edge $e \notin \mathcal{G}$. Then*

$$\|\Sigma_e(\mathcal{T} \cup e)\|_2^2 = \|\Sigma_e(\mathcal{T})\|_2^2 - \frac{\ell(c) - 1}{2\ell(c)},$$

where $\ell(c)$ is the length of the fundamental cycle created by adding the edge e.

This result shows that long cycles are better for filtering noises than short ones. It is also discussed in [11] that when adding multiple new edges, there is a trade-off between the length of the cycles and number of edges it shares with other cycles in the performance improvement. This leads to a combinatorial problem of trying to find long cycles that are edge disjoint.

VI. Outlooks

This tutorial provides a glimpse, albeit a biased one, into the fruitful interactions between systems and control theory on one hand and graph theory on the other.[9] Some of these interactions go back to the origins of control theory, for example in the context of signal flow graphs that are natural graphical representation of signal transformations through cascade and feedback pathways. The structural approach to patterned matrices and systems also has a rich history in system theory with recent renewed interest in this approach leading to new combinatorial and algebraic insights for networked systems. In the meantime, we believe that the interactions between graph theory and control theory will continue to grow and deepen. Here we list a few representative areas in this direction:

[9]We fully realize that some of our readers' favorite graph-theoretic constructs in system theory have been overlooked in our presentation.

1) **Extremal graphs**: Extremal graph theory, in the most general sense, studies how global properties of a graph influences its local substructure, and vice versa. One of the classic problems in extremal graph theory is the so-called *problem of forbidden subgraphs*. The problem asks what is the maximum size of a graph with a given order[10] such that it does not contain a prescribed subgraph $\mathcal{F}$? One important version of this problem is the following theorem by Turán,

   Theorem 12 ([22]). *Let $\mathcal{G}$ be any graph with n vertices such that $\mathcal{G}$ is K_{r+1}-free. Then the number of edges in $\mathcal{G}$ is at most $r^{-1}(r-1)\frac{n^2}{2}$.*

   Turán's theorem, therefore, provides a characterization of the size of a graph (in this case, the number of edges) in terms of the presence of complete subgraphs of a certain order. An example of a Turán graph is shown in Figure 4.

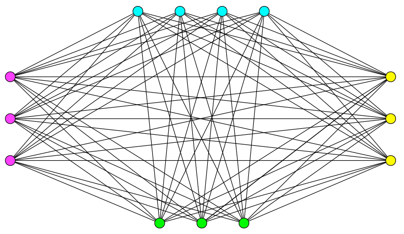


Fig. 15. A Turàn graph.

   Extremal graphs may shed insight on the performance and stability of large-scale networked systems. Indeed, one may consider an extremal perspective for the design of large-scale networks, or fault detection in networks where centralization is not feasible. For example, performance metrics such as those discussed in Section V may be studied for special classes of extremal graphs, leading to performance guarantees (bounds) even without a complete knowledge of the network structure. A simple example can be adapted from a result shown in [23], which provides a characterization of the $\mathcal{H}_2$-performance of a relative sensing network as $\|\Sigma(\mathcal{G})\|_2^2 = 2|\mathcal{E}|\|\Sigma\|_2^2$, where $\|\Sigma\|_2^2$ is the performance of the agents comprising the network system. For this simple example it immediately becomes clear that a corresponding 'Turán type' result can be stated as follows: Let $\Sigma(\mathcal{G})$ be a relative sensing network with n agents such that $\mathcal{G}$ is K_{r+1}-free. Then the $\mathcal{H}_2$ performance of $\Sigma(\mathcal{G})$ is at most $r^{-1}(r-1)n^2\|\Sigma\|_2^2$. We believe that adopting such an "extremal" point of view for system theory on networks will further deepen its unique combinatorial character.

2) **Learning and network evolution**: Networks not only support the dynamic evolution of the states associated with each node, but also can be the *subject of* the evolution, possibly through network-level feedback via the node states. For example, the edge weights in a network can be a function of the state of the nodes, that

[10]The order of a graph is the number of its vertices.

in turn evolve according to interaction characterized by these weights. The term "co-evolution" captures this interplay between node and edge level interactions. Although co-evolution often leads to nonlinear dynamics, its analysis can benefit from graph theoretic methods. For example, graph theoretic analysis can unravel time-scale separation in the co-evolution process, facilitating multiple time-scales analysis. On the other hand, one can embed learning algorithms in the node/edge evolution, such that the network assumes desired input-output properties. Such a point of view in turn can be used to embed network-level learning for robust and resilient operation of networked systems [24].

3) **Composite networks**: Over the past decade, a number of compositional rules for networks have been proposed. These compositional rules not only provide mechanisms for synthesizing large-scale networks from the smaller "atomic" ones, but often lead to effective decompositional approach for examining large networks. The corresponding compositional approach for the analysis and synthesis of networked dynamic systems, however, is in its infancy. A few representative works in this direction–for example–have examined how network controllability is preserved under Cartesian products [25], [26]. Much less is known on how performance of large scale networks can be examined from such a point of view, or how network decomposition can lead to efficient synthesis procedures. This point of view is particularly relevant for control theoretic treatment of networks, as computational requirements for typical control synthesis problems do not readily facilitate their application to large scale networks.

4) **Random graphs and minimal properties**: Random graph theory provides powerful tools to study structural and dynamical properties of large scale systems, scale at which the fine structure of the graph has less impact on its global properties, such as controllability or stability. One of the first and still most widely-used random graph model is the one of Erdös and Rényi [27]. In this model, given a fixed set of nodes, an edge between a pair of nodes is present with probability p, and each edge can be present independently of the other edges. Under this graph model, properties such as the controllability of the diffusion dynamics [28] or the structural stability of an associated zero-pattern [29] have been studied, but many open problems remain and it is fair to say that the area is vastly unexplored. Another area that is still unexplored is the one of *minimal structures*: when studying structural properties, some properties of the system are preserved when changing a 0 entry of a zero-pattern to a $*$ entry or equivalently, when adding an edge to the corresponding graph. The reciprocal is however not true; hence we can introduce *minimal graphs* for property P as the ones with the least amount of edges that meet property P. Characterizing these minimal graphs for different classes of properties remains wide-open.

REFERENCES

[1] W. Ren and R. W. Beard, *Distributed Consensus in Multi-vehicle Cooperative Control*. London: Springer London, 2008.

[2] F. Bullo, J. Cortés, and S. Martinez, *Distributed Control of Robotic Networks: a Mathematical Approach to Motion Coordination Algorithms*. Princeton, NJ: Princeton University Press, 2009.

[3] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multiagent Networks*. Princeton, NJ: Princeton University Press, 2010.

[4] C. Godsil and G. Royle, *Algebraic graph theory*. Springer, 2001.

[5] R. T. Rockafellar, *Network Flows and Monotropic Optimization*. Athena Scientific, 1998.

[6] A. Schrijver, *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., 1986.

[7] R. A. Brualdi and H. J. Ryser, *Combinatorial Matrix Theory*. Cambridge University Press, 1991.

[8] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 98, pp. 298 – 305, 1973.

[9] W. Tutte, *Graph Theory*. Cambridge: Cambridge University Press, 2001.

[10] D. Zelazo and M. Mesbahi, "Edge agreement: Graph-theoretic performance bounds and passivity analysis," *IEEE Transactions on Automatic Control*, vol. 56, no. 3, pp. 544–555, 2011.

[11] D. Zelazo, S. Schuler, and F. Allgöwer, "Performance and design of cycles in consensus networks," *Systems & Control Letters*, vol. 62, no. 1, pp. 85–96, 2013.

[12] M.-A. Belabbas, "Sparse stable systems," *Systems & Control Letters*, vol. 62, no. 10, pp. 981–987, 2013.

[13] A. Kirkoryan and M.-A. Belabbas, "Decentralized stabilization and planar graphs," in *IEEE Conference on Decision and Control*, 2014.

[14] C.-T. Lin, "Structural controllability," *IEEE Transactions on Automatic Control*, vol. 19, no. 3, pp. 201–208, 1974.

[15] R. Shields and J. Pearson, "Structural controllability of multiinput linear systems," *IEEE Transactions on Automatic Control*, vol. 21, no. 2, pp. 203–212, 1976.

[16] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabsi, "Controllability of complex networks," *Nature*, vol. 473, pp. 167–173, May 2011.

[17] A. Rahmani, M. Ji, M. Mesbahi, and M. Egerstedt, "Controllability of Multi-Agent Systems from a Graph-Theoretic Perspective," *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 162–186, 2009.

[18] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33 – 46, 2007.

[19] D. Klein and M. Randić, "Resistance distance," *Journal of Mathematical Chemistry*, vol. 12, pp. 81–95, 1993.

[20] D. Zelazo and M. Bürger, "On the Definiteness of the Weighted Laplacian and its Connection to Effective Resistance," in *IEEE Conference on Decision and Control*, 2014.

[21] M. S. Bartlett, "An inverse matrix adjustment arising in discriminant analysis," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 107–111, 1951.

[22] P. Turàn, "On an extremal problem in graph theory," *Matematikai ès Fizikai Lapok*, vol. 48, pp. 436–452, 1941.

[23] D. Zelazo and M. Mesbahi, "Graph-Theoretic Analysis and Synthesis of Relative Sensing Networks," *IEEE Transactions on Automatic Control*, vol. 56, pp. 971–982, May 2011.

[24] A. Chapman, E. Schoof, and M. Mesbahi, "Online adaptive network design for disturbance rejection," in *Principles of Cyber-Physical Systems* (S. Roy and S. Das, eds.), Cambridge University Press, to appear.

[25] W. Imrich and S. Klavzar, *Product Graphs: Structure and Recognition*. New York: Wiley, 2000.

[26] A. Chapman, M. Nabi-Abdolyousefi, and M. Mesbahi, "Controllability and observability of network-of-networks via Cartesian products," *IEEE Transactions on Automatic Control*, vol. 59, no. 10, pp. 2668–2679, 2014.

[27] P. Erdős and A. Rényi, "On random graphs I," *Publicationes Mathematicae Debrecen*, vol. 6, pp. 290–297, 1959.

[28] S. O'Rourke and B. Touri, "On a conjecture of Godsil concerning controllable random graphs," *SIAM Journal on Control and Optimization*, vol. 54, no. 6, pp. 3347–3378, 2016.

[29] A. Kirkoryan, *On the design of linear time invariant systems*. PhD thesis, University of Illinois, Urbana-Champaign, 2018.