51st IEEE Conference on Decision and Control
December 10-13, 2012. Maui, Hawaii, USA

Eulerian Consensus Networks

Daniel Zelazo and Frank Allgöwer

Abstract— **This work considers a special class of consensus seeking networks that we term *Eulerian consensus networks*. These consensus networks are defined over the class of graphs known as *Eulerian*, and admit many combinatorial features that are beneficial for both analysis and synthesis purposes. We first consider the $\mathcal{H}_2$ performance of Eulerian consensus systems and reveal that the performance is related to the length of cycles in the graph. The structure of Eulerian graphs motivates a procedure for designing large-scale Eulerian consensus networks. We propose a suite of algorithms for constructing these networks and demonstrate their numerical effectiveness on a graph with 5000 nodes.**

I. Introduction

A fundamental challenge related to the *design* of multi-agent systems is overcoming the computational burden associated with them. Indeed, tasks involving determining which information structures are best suited for a particular multi-agent application inevitably must address their combinatorial nature, often leading to $\mathcal{NP}$-hard problem formulations. A common approach to this problem is to consider optimization over weighted graphs or other convex relaxations [1]–[3]. However, in many scenarios, there may not be a proper justification for using weighted graphs. Even beyond that point, there is an elegant and intuitive benefit from understanding, in a purely binary way, why certain information links should, or should not, be present.

The design of information structures for systems comprised of multiple *dynamic* units introduces another layer of complexity beyond the purely combinatorial aspect described above. Recall that for controller design of dynamic systems, a clear characterization of system performance must be determined to design the controller in an optimal sense. For multi-agent systems, if the objective is to design the underlying information structure to achieve a specified performance for the aggregate system, then a clear characterization of how combinatorial properties affect system-theoretic properties must be attained. An important research direction for multi-agent systems, therefore, is a *graph-theoretic* characterization of system-performance metrics.

To address this problem, significant attention has been given to understanding what has emerged as a canonical model for multi-agent systems, referred to as the *consensus protocol* [4]. The simplicity of the model, most often presented as a collection of single integrators interacting over a communication graph, reveals a deep connection between its dynamic behavior and the underlying properties of the graph [5]–[9]. Following this research thread, there has also been progress related to how certain notions of systems performance, such as the $\mathcal{H}_2$ or $\mathcal{H}_\infty$ performance, relates to the underlying graph [10]–[12]. These results have led to synthesis scenarios for variations of the consensus problem, including leader selection with $\mathcal{H}_2$ performance [13], [14], or maximization of the algebraic connectivity of the graph [15]–[17].

Lost in many of the synthesis strategies of these approaches is the connection to *combinatorial properties* of the graph. We emphasize here a distinction between spectral properties of the graph, i.e. the eigenvalues of the Laplacian matrix, and combinatorial properties of a graph such as path lengths and cycles. Indeed, spectral properties introduce a layer of abstraction to the underlying graph that makes more tangible design issues, such as edge costs or communication distances, less intuitive. An important extension of these works, therefore, is the introduction of more general notions of systems performance in coordination with the design of these networks.

A thorough treatment in this direction was recently given in [12] via the introduction of the *Edge Laplacian* and its corresponding *edge agreement problem*. The edge Laplacian is a variant of the graph Laplacian that provides a more transparent understanding of how spanning trees and cycles affect certain algebraic properties of a graph. When the consensus protocol is analyzed using this construction, clear graph theoretic interpretations of the $\mathcal{H}_2$ and $\mathcal{H}_\infty$ performance of the system can be derived [12]. An open conjecture from the work in [12] was that the cycle structure of the graph has an impact on the corresponding system performance. A first contribution of this work, therefore, is a resolution of this conjecture for the class of graphs known as *Eulerian graphs* [18].

Eulerian graphs, named after Leonard Euler in connection with his seminal work on the *bridges of Königsberg problem* [19], have many special graph-theoretic features. Of particular interest with this work is the property that all Eulerian graphs can be decomposed into edge-disjoint cycles. Surprisingly, Eulerian graphs have been identified as useful in many applications including DNA sequencing, circuit design, and combinatorial problems like the *Chinese postman delivery problem* [20]–[23].

The contributions of this work are two-fold. First, we extend the results in [12] providing a combinatorial char-

The authors thank the German Research Foundation (DFG) for financial support within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart and within the Priority Program 1305 "Control Theory of Digitally Networked Dynamical Systems". The authors are with the Institute for Systems Theory and Automatic Control, University of Stuttgart, Pfaffenwaldring 9, 70550 Stuttgart, Germany, {daniel.zelazo, frank.allgower}@ist.uni-stuttgart.de.

978-1-4673-2066-5/12/$31.00 ©2012 IEEE

acterization of how cycles in consensus networks influence the $\mathcal{H}_2$ performance. In particular, we provide an exact characterization of the performance when the graph contains only edge-disjoint cycles, and we call such networks *Eulerian consensus networks*. We show that the addition of edge disjoint cycles to a graph improves the performance by an amount related to the inverse of the length of the cycle. This analytic characterization is then used to motivate a suite of algorithms for designing large-scale Eulerian consensus networks.

This paper is organized as follows. Section §II reviews some fundamental properties of graphs, their algebraic representations, and Eulerian graphs. The edge agreement problem and a combinatorial interpretation of its $\mathcal{H}_2$ performance is given in §III. In §IV we propose two companion algorithms for designing Eulerian consensus networks. Numerical simulations are presented in §V. Finally, we offer some concluding remarks in §VI.

II. Trees, Cycles, and Eulerian Graphs

The main graph-theoretic tool used in this work is the basic notion of spanning trees and cycles of a graph [18]. Recall that an undirected graph, denoted $\mathcal{G}$, is composed of a node set $\mathcal{V} = \{v_1, \ldots, v_n\}$ and an edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ describing the incidence relation between nodes. The *neighborhood* of a node v is the set of nodes adjacent to it; $\mathcal{N}(v) = \{u \in \mathcal{V} | \{v, u\} \in \mathcal{E}\}$. The *degree* of a node is the cardinality of its neighborhood, $d_v = |\mathcal{N}(v)|$.Any connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be expressed as the union of a spanning tree $\mathcal{T} = (\mathcal{V}, \mathcal{E}_\tau)$ and the remaining edges $\mathcal{E}_c = \mathcal{E} \setminus \mathcal{E}_\tau$. That is,

$$\mathcal{G} \quad = \quad \mathcal{T} \cup \mathcal{C}, \tag{1}$$

where $\mathcal{C} = (\mathcal{V}, \mathcal{E}_c)$.

The edges in the graph $\mathcal{C}$ can be viewed as those edges that, when combined with certain edges in the tree $\mathcal{T}$, form *cycles* in $\mathcal{G}$. Recall that a *path* between two nodes $v_i, v_j \in \mathcal{V}$ is a sequence of distinct nodes such that each node in the sequence is adjacent to the previous node. As an example, the path from node v_1 to v_6 in the graph in Figure 1 can be expressed as the sequence $v_1 v_2 v_6$, or $v_1 v_2 v_3 v_5 v_6$. For any connected graph $\mathcal{G}$ with a spanning tree $\mathcal{T}$, there always exists a path using only the edges in $\mathcal{T}$ [18]. If the initial node and terminal node of a path are the same, then it is called a *cycle*. Therefore, any cycle can be formed by first finding a path from, say nodes v_i to v_j using only the edges in $\mathcal{E}_\tau$, and then using a single edge $\{v_j, v_i\} \in \mathcal{E}_c$.

In this work, we will denote (and enumerate) the ith cycle in a graph as a set of edges, $c_i = \{e_1, \ldots, e_r\}$, where a subset of $r-1$ edges belong to the spanning tree $\mathcal{T}$, and the remaining edge to $\mathcal{E}_c$. The *length of a cycle* is then its cardinality, i.e., $|c_i|$. Referring again to Figure 1, observe that four cycles are labeled; for example, the cycle $c_3 = \{\{v_1, v_2\}, \{v_2, v_4\}, \{v_1, v_4\}\}$ has length $|c_3| = 3$.

Definition 1: Two cycles c_i and c_j are *edge disjoint* if they do not have any edges in common; i.e. $c_i \cap c_j = \emptyset$. Two cycles are *correlated* if they are not edge disjoint.

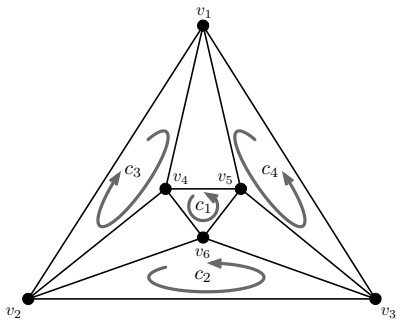


Fig. 1. An Eulerian graph with 4 edge-disjoint cycles.

One of the oldest problems in graph theory attributed to Euler is determining if there exists a walk[1] on a graph beginning and ending at the same node that traverses each edge only once; such a walk is called *Eulerian cycle* [18]. Any graph that contains an Eulerian cycle is called an *Eulerian graph*. There are a number of immediate properties that can be inferred from the definition of a Eulerian graph, summarized in the following proposition:

Proposition 1 ([18]): Given a connected undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the following statements are equivalent:

i) $\mathcal{G}$ is an Eulerian graph.
ii) The degree of each node is even.
iii) $\mathcal{G}$ is the union of edge-disjoint cycles; i.e., $\mathcal{E} = \cup_{i=1}^{k} c_i$ and $c_i \cap c_j = \emptyset$, $\forall i, j$.

The graph in Figure 1 is Eulerian, with disjoint cycles c_1, c_2, c_3, and c_4; note that each node has even degree. It is also worth mentioning that the decomposition of a Eulerian graph into disjoint cycles is not unique. It is the third property of Proposition 1 that will be exploited when considering the performance of consensus networks.

A. Algebraic Representations

The characterization of trees and cycles in a graph have up to now been purely set-theoretic. The use of algebraic representations for graphs are also useful for the analysis of consensus networks. The main algebraic construct we employ in this work is the node-edge *incidence matrix* $E(\mathcal{G}) \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$ of a graph with an arbitrary orientation assigned [18].

Using an appropriate labeling of the edges in the graph, we can always express the incidence matrix in terms of the subgraphs $\mathcal{T}$ and $\mathcal{C}$ for a particular choice of spanning tree, $E(\mathcal{G}) = \left[\begin{array}{cc} E(\mathcal{T}) & E(\mathcal{C}) \end{array} \right]$. This representation aids in the interpretation of several results relating the sub-graphs $\mathcal{T}$ and $\mathcal{C}$. In particular, the observation that any cycle can be represented by a sequence of edges from $\mathcal{T}$ and one edge from $\mathcal{C}$ can be formalized algebraically using the incidence matrix by defining a matrix $T_{(\mathcal{T},\mathcal{C})} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}_c|}$ such that $E(\mathcal{T}) T_{(\mathcal{T},\mathcal{C})} = E(\mathcal{C})$. It is then straight-forward to verify that

$$T_{(\mathcal{T},\mathcal{C})} \quad = \quad \left(E(\mathcal{T}) E(\mathcal{T})^T \right)^{-1} E(\mathcal{T})^T E(\mathcal{C}). \tag{2}$$

[1] A *walk* is a sequence of vertices, not necessarily distinct, such that each node in the sequence is adjacent to the previous node.

The matrix $T_{(\mathcal{T},\mathcal{C})}$, therefore, encodes information related to the cycles that can be formed from the spanning tree $\mathcal{T}$. To further aid in the following exposition, we express $T_{(\mathcal{T},\mathcal{C})}$ in terms of its columns, $T_{(\mathcal{T},\mathcal{C})} = \begin{bmatrix} t_1 & \cdots & t_{|\mathcal{E}_c|} \end{bmatrix}$, and using a slight abuse in convention, we will also refer to the ith column of $T_{(\mathcal{T},\mathcal{C})}$ as the ith cycle of the graph $\mathcal{G}$. Similarly, we will refer to the ith column of $E(\mathcal{T})$ with τ_i as the ith edge in the spanning tree. At times, we will refer to the matrix $R_{(\mathcal{T},\mathcal{C})} = \begin{bmatrix} I & T_{(\mathcal{T},\mathcal{C})} \end{bmatrix}$. Using this notation, note that $E(\mathcal{G}) = E(\mathcal{T})R_{(\mathcal{T},\mathcal{C})}$.

Proposition 2: The matrix $T^T_{(\mathcal{T},\mathcal{C})}T_{(\mathcal{T},\mathcal{C})}$ encodes the following information about the cycles in $\mathcal{G}$.

i) $[T^T_{(\mathcal{T},\mathcal{C})}T_{(\mathcal{T},\mathcal{C})}]_{ii} = t_i^T t_i = |c_i| - 1$.
ii) $[T^T_{(\mathcal{T},\mathcal{C})}T_{(\mathcal{T},\mathcal{C})}]_{ij} = t_i^T t_j = 0$ if and only if cycles c_i and c_j are edge-disjoint.
iii) $[T^T_{(\mathcal{T},\mathcal{C})}T_{(\mathcal{T},\mathcal{C})}]_{ij} \neq 0$ if and only if cycles c_i and c_j are correlated.

Corollary 1: For a Eulerian graph $\mathcal{G}$ with spanning tree $\mathcal{T}$, the matrix $T^T_{(\mathcal{T},\mathcal{C})}T_{(\mathcal{T},\mathcal{C})}$ is diagonal with each element containing the length of the ith cycle minus 1.

The *edge Laplacian* is a $|\mathcal{E}|\times|\mathcal{E}|$ symmetric matrix defined as [12]

$$L_e(\mathcal{G}) \quad := \quad E(\mathcal{G})^T E(\mathcal{G}). \tag{3}$$

One of the main results in [12] showed that the edge Laplacian is related to the graph Laplacian via a similarity transformation. We summarize the results here and refer the reader to [12] for the proof.

Theorem 1 ([12]): The graph Laplacian for a connected graph, $L(\mathcal{G}) = E(\mathcal{G})E(\mathcal{G})^T$, has the same non-zero eigenvalues as the matrix $L_e(\mathcal{T})R_{(\mathcal{T},\mathcal{C})}R^T_{(\mathcal{T},\mathcal{C})}$.

We refer to the matrix $L_e(\mathcal{T})R_{(\mathcal{T},\mathcal{C})}R^T_{(\mathcal{T},\mathcal{C})}$ as the *essential edge Laplacian*. A detailed discussion of this matrix along with its combinatorial interpretations is given in [12]. The main feature is that the essential edge Laplacian encodes explicitly information about the cycles in a graph. The essential edge Laplacian is the key algebraic construct used to derive the performance of consensus networks.

III. Performance of Consensus Networks

The main analytic tool used to derive the $\mathcal{H}_2$ performance of consensus networks is based on an edge variant of the classic consensus protocol that we term the *edge agreement problem* [12]. In this section, we briefly summarize the main results of [12], and refer the reader to that work for a more in-depth study.

The standard noise-free consensus model is usually presented as the autonomous system [4],

$$\dot{x}(t) \quad = \quad -L(\mathcal{G})x(t). \tag{4}$$

Here, the vector $x(t) \in \mathbb{R}^n$ is the concatenated state of each agent, and $L(\mathcal{G})$ is the Laplacian matrix of the graph.

The consensus protocol is based on integrator models for each agent, and consequently noises can be injected into the process, $\dot{x}_i(t) = u_i(t) + w_i(t)$, and into the measurement for use in generating a control as $u_i(t) = \sum_{j\in\mathcal{N}(v_i)}(x_j(t) - x_i(t)+v_{ij}(t))$.[2] This leads to a two-port representation of the consensus protocol and provides a framework for considering the presence of exogenous inputs, such as reference signals and noises entering the measurement and process. Note that this generalized model is not a minimal realization of the system. Indeed, the system has an unobservable mode in the direction of the $\mathbb{1}$ vector [12], [24]. As discussed in [12], a minimal realization of the system can be expressed using the essential edge Laplacian matrix via a coordinate transformation,

$$\Sigma_e(\mathcal{G}) : \begin{cases} \dot{x}_\tau(t) \;=\; -L_e(\mathcal{T})R_{(\mathcal{T},\mathcal{C})}R^T_{(\mathcal{T},\mathcal{C})}x_\tau(t) + \\ \qquad \begin{bmatrix} E(\mathcal{T})^T & -L_e(\mathcal{T})R_{(\mathcal{T},\mathcal{C})} \end{bmatrix} \begin{bmatrix} w(t) \\ v(t) \end{bmatrix} \\ z(t) \;=\; x_\tau(t). \end{cases} \tag{5}$$

Here, the transformed state vector $x_\tau(t) \in \mathbb{R}^{|\mathcal{E}_\tau|}$ can be interpreted as a state associated with the edges of the spanning tree $\mathcal{T}$. The system (5) is referred to as the *edge agreement problem*. In this work we consider $\mathcal{T}$ as a *skeletal system* for the complete consensus network. In this regard, we only observe the states along the tree, $x_\tau(t)$, as the controlled variable in the edge agreement problem.[3]

The $\mathcal{H}_2$ performance of the minimal edge agreement problem, therefore, captures how noises entering the system affect the asymptotic deviation of the states from the consensus value. The $\mathcal{H}_2$ performance was originally considered in [12] and we only present the main result here.

Theorem 2 ([12]): The $\mathcal{H}_2$ performance of edge agreement problem (5) is

$$\|\Sigma_e(\mathcal{G})\|_2^2 \quad = \quad \frac{1}{2}\mathbf{tr}\left[(R_{(\mathcal{T},\mathcal{C})}R^T_{(\mathcal{T},\mathcal{C})})^{-1}\right] + (n-1). \tag{6}$$

A direct consequence of Theorem 2 leads to an upper and lower bound for the performance of the edge agreement problem. In particular, we observe that the performance is upper bounded by any choice of spanning tree, and is lower bounded by the complete graph, K_n. This observation also indicates that cycles generally improve performance. An open problem, therefore, is to precisely characterize how cycles impact the performance. The first result considers how the addition of a single edge to a spanning tree improves the performance.

Theorem 3: Consider the edge agreement problem where the underlying graph is a spanning tree, $\mathcal{T} = (\mathcal{V}, \mathcal{E}_\tau)$, and consider a single edge $e \in (\mathcal{V}\times\mathcal{V})\setminus\mathcal{E}_\tau$. Then the graph $\mathcal{T}\cup e$ has one cycle, c, and the performance of the edge agreement problem is given as

$$\|\Sigma_e(\mathcal{T}\cup e)\|_2^2 \quad = \quad \|\Sigma_e(\mathcal{T})\|_2^2 - \frac{1}{2}(1-|c|^{-1}). \tag{7}$$

Proof: From (2), the matrix $T_{(\mathcal{T},\mathcal{C})}$ can be computed and will have only one column, therefore $T_{(\mathcal{T},\mathcal{C})}T^T_{(\mathcal{T},\mathcal{C})}$ is a rank-one matrix. To compute the performance of $\Sigma_e(\mathcal{T}\cup e)$, the result of Theorem 2 must be applied. In this setting, $(R_{(\mathcal{T},\mathcal{C})}R^T_{(\mathcal{T},\mathcal{C})})^{-1}$ can be seen as the inverse of the identity

[2] For ease of presentation we assume the noises are uncorrelated white Gaussian noises with unit covariance.

[3] As opposed to considering $R^T_{(\mathcal{T},\mathcal{C})}x(t)$ as the controlled variable.

with a rank-one update. Applying the Sherman-Morrison formula [25] leads to

$$(R_{(\mathcal{T},\mathcal{C})}R_{(\mathcal{T},\mathcal{C})}^T)^{-1}=\big(I+T_{(\mathcal{T},\mathcal{C})}T_{(\mathcal{T},\mathcal{C})}^T\big)^{-1}=I-\frac{1}{|c|}T_{(\mathcal{T},\mathcal{C})}T_{(\mathcal{T},\mathcal{C})}^T.$$

Computing the trace yields $\mathbf{tr}[(R_{(\mathcal{T},\mathcal{C})}R_{(\mathcal{T},\mathcal{C})}^T)^{-1}] = n-1-\frac{|c|-1}{|c|}$. Therefore,

$$\|\Sigma_e(\mathcal{T}\cup e)\|_2^2=\frac{1}{2}\left(n-1-\frac{|c|-1}{|c|}\right)+n-1.$$

■

Theorem 3 immediately leads to the following corollary, bounding the performance increase achievable by adding only one edge.

Corollary 2: Consider the edge agreement problem where the underlying graph is a spanning tree, $\mathcal{T}=(\mathcal{V},\mathcal{E}_\tau)$, and consider a an edge $e \in (\mathcal{V}\times\mathcal{V})\setminus\mathcal{E}_\tau$. The maximum performance gain is achieved by adding a cycle of length equal to $\mathbf{diam}[\mathcal{G}]+1$, while the minimum performance gain is obtained by adding a cycle of length three (i.e., a triangle).[4]

This result states that in terms of the $\mathcal{H}_2$ performance, longer cycles are better than shorter ones when adding a single edge. Furthermore, it establishes a very strong connection between a purely combinatorial property of the graph, i.e., the length of the cycle, to a system theoretic property of the edge agreement problem, i.e., its $\mathcal{H}_2$ performance.

An immediate extension of Theorem 3 allows for an exact characterization of the $\mathcal{H}_2$ performance for all Eulerian graphs.

Corollary 3: Consider the edge agreement problem where the underlying graph is an Eulerian graph, $\mathcal{G}=(\mathcal{V},\mathcal{E}_\tau)$. Then the graph can be expressed as the union of k edge-disjoint cycles c_i, $\mathcal{G}=\cup_{i=1}^k\mathcal{C}_i$ with $\mathcal{C}_i=(\mathcal{V},c_i)$, and the performance of the corresponding edge agreement problem is given as

$$\|\Sigma_e(\mathcal{G})\|_2^2 \;=\; \|\Sigma_e(\mathcal{T})\|_2^2-\frac{1}{2}\left(k-\sum_{i=1}^k\frac{1}{|c_i|}\right). \quad (8)$$

Proof: The proof is a direct consequence of Theorem 3, the structure of $T_{(\mathcal{T},\mathcal{C})}$ for edge disjoint graphs, and repeated application of the Sherman-Morrison formula [25]. ■

The appeal of Corollary 3 is that it provides an exact characterization of the performance for a very large class of graphs. This has direct implications when considering design problems for consensus networks, that we consider in the sequel.

IV. Design of Eulerian Consensus Networks

The results of the previous section provides a clear combinatorial interpretation of the role cycles play in the performance of Eulerian consensus networks. A natural progression is then to consider designing these networks with a guarantee on the system's performance. Stated naively, this reduces to an optimization problem of the form

$$\min_{\mathcal{G}\in\mathbf{G}} \quad \|\Sigma_e(\mathcal{G})\|_2, \quad (9)$$

where $\mathbf{G}$ denotes an abstract set describing desired features of the underlying graph (i.e. ensuring connectivity or inclusion of certain sub-graphs). Indeed, the problem (9) can be viewed as a *mixed-integer* optimization problem and, as a result, is considered numerically intractable [26].

A common approach for solving problems of the form (9) is to consider convex relaxations, as discussed in §I. Other relaxation methods try to recover sparse solutions, such as with ℓ_1 relaxations [27]–[29]. Despite the appeal of these methods, they also suffer from computational issues as the problem size increases. In fact, relaxation methods for solving (9) for sizes on the order of 1,000s of nodes will generally not be tractable.

In this section we provide a computationally efficient algorithm for designing large-scale consensus networks with a known $\mathcal{H}_2$ performance. The proposed algorithm leverages the result in Corollary 3 by considering the design of Eulerian graphs. The first algorithm describes how to construct an Eulerian graph by adding edges to an existing tree structure. The algorithm also returns the decomposition of the resulting graph into its edge-disjoint cycles, and as a result, the exact value of the $\mathcal{H}_2$ performance.

The general idea of the algorithm is as follows. We assume that a spanning tree is given in the network (if not, one can be created a variety of ways; for example, using a minimum weight spanning tree algorithm). Every node $v_i \in \mathcal{V}$ is then classified as one of three types: i) a *leaf*, a node with degree one, ii) a *stem*, a node with degree two, and iii) a *branch*, a node with degree greater than two. The algorithm adds edges to the graph by connecting leaves with branches and then updating the degree of each node. Note that the algorithm also includes additional logic to handle conflicts when it is not clear where to add the new edge (lines 14-21). Bookkeeping of the paths between leaves and branches ensure that each edge in the tree is used only once, and the final degree of each node will be even. The algorithm is described in more detail in Algorithm 1.

Observe that with each iteration of the algorithm, the number of leaves must decrease by at least one (and at most two), ensuring that the algorithm will terminate. Construction of the routing Θ in the algorithm can be used to maintain the cycles and their length (i.e. $|\Theta|+1$) for monitoring the performance improvement with each new cycle. The last steps in the algorithm remove the edges in the tree that have been used in a routing. It is also worth observing that this algorithm in fact has a *distributed* architecture. Indeed, finding routings involves only local connectivity information and it is easy to envision a fully distributed implementation of this algorithm.

The appeal of Algorithm 1 is in its ability to construct truly large-scale graphs with low computational complexity and an exact characterization of the resulting performance. This algorithm, however, does not necessarily find the $\mathcal{H}_2$-optimal

[4]The *diameter* of a graph, $\mathbf{diam}[\mathcal{G}]$ is the maximum distance between any two nodes, where the distance is the length of the shortest path.

Algorithm 1: Eulerian Graph Construction Algorithm

Data: A connected undirected spanning tree $\mathcal{T} = (\mathcal{V}, \mathcal{E}_\tau)$ with $|\mathcal{V}| > 2$

Result: An Eulerian graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{T} \subset \mathcal{G}$.

begin
- Initialize the graph $\mathcal{G} = (\mathcal{V}, \emptyset)$
- Identify the set of leaves: $\mathcal{L} = \{l_1, \ldots, l_l\}$
- Identify the set of stems: $\mathcal{S} = \{s_1, \ldots, s_s\}$
- Identify the set of branches: $\mathcal{B} = \{b_1, \ldots, b_b\}$
- **while** $|\mathcal{L}| > 0$ **do**
  - Select a leaf $l \in \mathcal{L}$
  - Initialize the routing $\Theta = \{l\}$
  - $v = \mathcal{N}(l) \setminus \Theta$
  - **while** $d_v = 2$ **do**
    - $\Theta = \Theta \cup \{v\}$
    - $v = \mathcal{N}(v) \setminus \Theta$
  - $\Theta = \Theta \cup \{v\}$
  - **if** $|\Theta| = 2$ **then**
    - $U = \mathcal{N}(v) \setminus \Theta$
    - **if** $U \cap \mathcal{L} \neq \emptyset$ **then**
      - Set $v \in U \cap \mathcal{L}$
    - **else if** $U \cap \mathcal{B} \neq \emptyset$ **then**
      - Set $v \in U \cap \mathcal{B}$
    - **else**
      - Set $v \in U \cap \mathcal{S}$
  - $\mathcal{E} = \mathcal{E} \cup \{l, v\} \cup \mathcal{E}_\tau(\Theta)$ ($\mathcal{E}_\tau(\Theta)$ denotes edges used in routing)
  - $\hat{\mathcal{E}}_\tau = \mathcal{E}_\tau \setminus \mathcal{E}_\tau(\Theta)$, $\mathcal{T} = (\mathcal{V}, \hat{\mathcal{E}}_\tau)$
  - Update $\mathcal{L}, \mathcal{S}, \mathcal{B}$ and node degrees with new $\mathcal{T}$
  - $\Theta = \emptyset$

Eulerian graph given the spanning tree $\mathcal{T}$. In an effort to address this, we propose a *performance correction algorithm* that maintains the Eulerian property of the graph while leading to an improvement in the performance. First, we make the following observation demonstrating that multiple shorter edge-disjoint cycles can have a larger impact than fewer long cycles.

Proposition 3: Consider two edge-disjoint cycles c_1 and c_2 that share a common node v. Assume that the edge $\{v, u\} \in c_1$ and $\{v, u'\} \in c_2$. Construct a new cycle c' by adding an edge $\{u, u'\}$ and removing the edges $\{v, u\}$ and $\{v, u'\}$; that is, $c' = (c_1 \cup c_2 \cup \{u, u'\}) \setminus \{\{v, u\}, \{v, u'\}\}$. Then one has the following inequality,

$$1 - \frac{1}{|c'|} = 1 - \frac{1}{|c_1| + |c_2| - 1} < 2 - \frac{1}{|c_1|} - \frac{1}{|c_2|}.$$

Proof: To show that c' is a cycle simply requires appending the routings from v to u, u to u' using the new edge, and u' to v. The length of the new cycle is easily verified. It is straightforward to verify that the inequality is true, recalling that any cycle must have length at least equal to 3. ■

The implications of Proposition 3 when viewed in the context of Corollary 3 suggest that multiple short cycles are more beneficial than fewer longer cycles. This then motivates the new algorithm that effectively breaks longer cycles into shorter ones. A simple combinatorial observation is that for a cycle to be 'split', it must have a length of at least 5. The proposed algorithm searches for cycles of length greater than 5, and then splits that cycle into a smaller one of length 3 and one with the remaining edges. This operation will add a new cycle at each iteration.

Algorithm 2: Performance Enhancing Algorithm

Data: An Eulerian graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with its cycle decomposition $\mathbf{C} = \{c_1, \ldots, c_c\}$.

Result: An Eulerian graph $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$ with improved $\mathcal{H}_2$ performance.

begin
- Initialize the graph $\mathcal{G}' = \mathcal{G}$
- **while** $\max_i |c_i| \geq 5$ **do**
  - Select cycle $c \in \mathbf{C}$ with maximum length
  - Select any edge $e = \{u, v\} \in c$
  - Set $\mathcal{E}' = (\mathcal{E}' \cup \{\{u, s\}, \{v, s\}\} \setminus e$ where s is contained in the cycle c, and is 2 hops away from u and $|c| - 3$ hops away from v.
  - $\mathbf{C} = \mathbf{C} \setminus c$

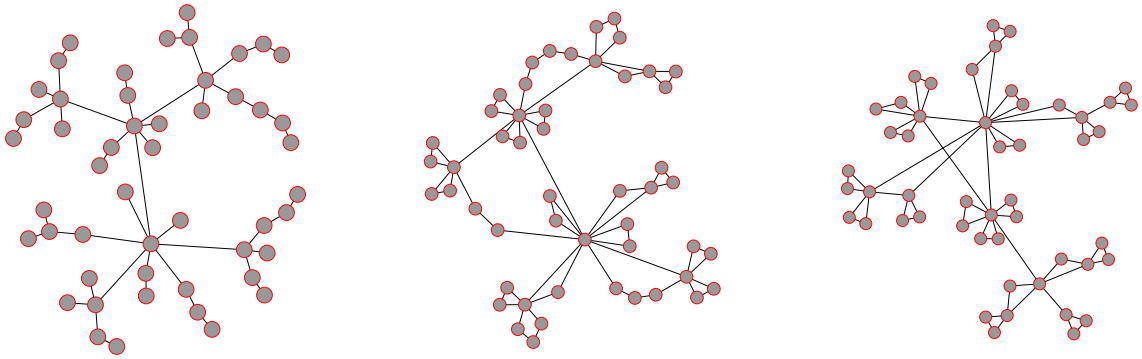
V. Simulation Example

In this section we present some simulations demonstrating the applicability of Algorithm 1 for designing large-scale Eulerian consensus networks. First, we present a simulation on only 50 nodes to better visualize the construction of Eulerian graphs. The $\mathcal{H}_2$ performance for the tree (Figure 2(a)) can be calculated as $\|\Sigma(\mathcal{T})\|_2^2 = 73.5$. The Eulerian graph generated by Algorithm 1 (Figure 2(b)) has performance $\|\Sigma(\mathcal{G})\|_2^2 = 65.813$ and added 20 new cycles to the graph. The Eulerian graph generated by Algorithm 2 has performance $\|\Sigma(\hat{\mathcal{G}})\|_2^2 = 64.125$ and created 3 new cycles using the split procedure. The next simulation is for a Eulerian graph on 5000 nodes.[5] This algorithm took under 1 minute to generate the graph running MATLAB 7.12 on a 2.66 GHz Intel Core 2 Duo processor. The performance of the spanning tree can be calculated as $\|\Sigma(\mathcal{T})\|_2^2 = 7498.5$. The Eulerian graph generated by Algorithm 1 has performance $\|\Sigma(\mathcal{G})\|_2^2 = 6,700.224$ and added 2,061 new cycles to the graph. The Eulerian graph generated by Algorithm 2 has performance $\|\Sigma(\hat{\mathcal{G}})\|_2^2 = 6535.75$ and created 268 new cycles using the split procedure. The affect the the performance enhancing algorithm is that the resulting graph only has cycles of length 3 and 4.

VI. Concluding Remarks

This work provided an analysis for the $\mathcal{H}_2$ performance of a special class of consensus networks. We considered con-

[5]The size of the graph prohibits any meaningful visualization so a figure is not included.



(a) A spanning tree on 50 nodes. (b) Result from Algorithm 1. (c) Result from Algorithm 2.

Fig. 2. Illustration of Algorithms 1 and 2 on 50 nodes.

sensus networks defined over Eulerian graphs, and used the special structure of these graphs to make precise statements on the system performance. In particular, we showed that the performance is proportional to the inverse of the length of the cycles in the Eulerian decomposition, and that multiple short cycles are better than fewer long cycles in terms of the overall performance. The particular structure of the Eulerian graph motivated a constructive algorithm for the design of Eulerian consensus networks. The algorithms permits very large-scale design with an exact characterization of the resulting system performance.

REFERENCES

[1] S. P. Boyd, "Convex Optimization of Graph Laplacian Eigenvalues," in *International Congress of Mathematicians*, vol. 3, no. 1-3, Madrid, dec 1998, pp. 1311–1310.

[2] S. Y. Shafi, M. Arcak, and L. El Ghaoui, "Designing node and edge weights of a graph to meet Laplacian eigenvalue constraints," in *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, Sep. 2010, pp. 1016–1023.

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

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

[5] M. Nabi-abdolyousefi and M. Mesbahi, "Circulant Networks : Controllability , Observability , and Linear Quadratic Control," *Control*, no. 1, 2011.

[6] A. Rahmani, M. Ji, M. Mesbahi, and M. Egerstedt, "Controllability of multiagent systems: from a graph-theoretic perspective," *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 162–186, 2008.

[7] B. Briegel, D. Zelazo, M. Bürger, and F. Allgöwer, "On the Zeros of Consensus Networks," in *Proc. 50th IEEE Conference on Decision and Control (to appear)*, Orlando, FL, 2011.

[8] S. Tonetti and R. M. Murray, "Limits on the network sensitivity function for homogeneous multi-agent systems on a graph," in *American Control Conference (ACC)*, June 2010, pp. 1–33.

[9] M.-G. Yoon and K. Tsumura, "Transfer function representation of cyclic consensus systems," *Automatica*, vol. 47, no. 9, pp. 1974 – 1982, 2011.

[10] P. Lin, Y. Jia, and L. Li, "Distributed robust H-consensus control in directed networks of agents with time-delay," *Systems & Control Letters*, vol. 57, no. 8, pp. 643–653, Aug. 2008.

[11] L. Scardovi, M. Arcak, and E. D. Sontag, "Synchronization of Interconnected Systems With Applications to Biochemical Networks: An Input-Output Approach," *IEEE Transactions on Automatic Control*, vol. 55, no. 6, pp. 1367–1379, Jun. 2010.

[12] 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, Mar. 2011.

[13] A. Clark and R. Poovendran, "A Submodular Optimization Framework for Leader Selection in Linear Multi-agent Systems," in *Proc. 50th IEEE Conference on Decision and Control*, Orlando, FL, 2011, pp. 3614–3621.

[14] F. Lin, M. Fardad, and M. R. Jovanovic, "Algorithms for Leader Selection in Large Dynamical Networks: Noise-corrupted Leaders," in *Proc. 50th IEEE Conference on Decision and Control*, Orlando, FL, 2011.

[15] R. Dai and M. Mesbahi, "Optimal Topology Design for Dynamic Networks," in *50th IEEE Conference on Decision and Control and European Control Conference (CDC 2011)*. Orlando, FL: IEEE, 2011, pp. 1280–1285.

[16] S. P. Boyd and A. Ghosh, "Growing Well-connected Graphs," in *Proceedings of the 45th IEEE Conference on Decision & Control*, San Diego, 2006, pp. 6605–6611.

[17] Y. Kim and M. Mesbahi, "On maximizing the second smallest eigenvalue of a state-dependent graph Laplacian," in *Proceedings of the 2005 American Control Conference*. Portland, OR: IEEE, 2005, pp. 99–103.

[18] C. Godsil and G. Royle, *Algebraic Graph Theory*. Springer, 2009.

[19] L. Euler, "Solutio problematis ad geometriam situs pertinentis," *Commentarii academiae scientiarum Petropolitanae*, vol. 8, pp. 128–140, 1741.

[20] M. Guan, "The Maximum Weighted Cycle-packing Problem and Its Relation to the Chinese Postman Problem," in *Progress in Graph Theory*, J. Bondy and U. Murty, Eds. New Yotk: Academic Press, 1984, pp. 323–326.

[21] J. Blazewicz, "On some properties of DNA graphs," *Discrete Applied Mathematics*, vol. 98, no. 1-2, pp. 1–19, Oct. 1999.

[22] B. Carlson, C. Chan, and D. Meliksetian, "An efficient algorithm for the identification of dual Eulerian graphs and its application to cell layout," in *1992 IEEE International Symposium on Circuits and Systems*, vol. 5. San Diego, CA: IEEE, 1992, pp. 2248–2251.

[23] P. A. Pevzner, H. Tang, and M. S. Waterman, "An Eulerian path approach to DNA fragment assembly." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 98, no. 17, pp. 9748–53, Aug. 2001.

[24] D. Zelazo and M. Mesbahi, "Graph-Theoretic Analysis and Synthesis of Relative Sensing Networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 5, pp. 971–982, May 2011.

[25] K. Petersen and M. Pedersen, *The Matrix Cookbook*. Technical University of Denmark, 2008.

[26] B. H. Korte and J. Vygen, *Combinatorial optimization: theory and algorithms*. Berlin: Springer-Verlag, 2000.

[27] E. J. Candes, M. B. Wakin, and S. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 877–905, 2008.

[28] F. Lin, M. Fardad, and M. R. Jovanović, "Design of Optimal Sparse Feedback Gains via the Alternating Direction Method of Multipliers," *IEEE Transaction on Automatic Control (submitted)*, p. 32, Nov. 2011.

[29] S. Schuler, P. Li, J. Lam, and F. Allgöwer, "Design of Structured Dynamic Output-feedback Controllers for Interconnected Systems," *International Journal of Control*, vol. 84, no. 12, pp. 2081–2091, 2011.