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

Cycles and Sparse Design of Consensus Networks

Daniel Zelazo, Simone Schuler, and Frank Allgöwer

Abstract— **This work considers the role that cycles play in consensus networks. We show how the presence of cycles improve the $\mathcal{H}_2$ performance of the consensus network. In particular, we provide an explicit combinatorial characterization relating the length of cycles to the improvement in the performance of the network. This analysis points to a general trade-off between the length of the cycle and how many edges the cycle shares with other cycles. These analytic results are then used to motivate a design procedure for consensus networks based on an ℓ_1 relaxation. This relaxation method leads to sparse and $\{0,1\}$-solutions for the design of consensus graphs. A feature of the ℓ_1 relaxation is the ability to include weighting terms in the objective. The choice of weighting functions are related to the combinatorial properties of the graph. The applicability of this scheme is then shown via a set of numerical examples.**

I. Introduction

As the scale of multi-agent systems increases, it becomes essential to understand how certain features of the interconnection structure affect the performance, stability, and behavior of the entire network. The dynamics of each agent comprising these systems and the interaction structure that couples them leads to a highly complex network that requires new perspectives and tools for their analysis [1].

In an attempt to better understand the role of the interconnection structure of these systems, many simpler models have been proposed. Of these models, the *consensus protocol* has emerged as a canonical system for multi-agent networks. The appeal of this model is the transparency between its dynamic behavior and the properties of the graph describing the interaction structure. Despite its simplicity, the consensus protocol has proven to be an essential component of many multi-agent systems ranging from formation control to distributed computation and sensor fusion [2], [3].

The importance of consensus networks requires a complete theory closing the loop between analysis and design. Notions of systems performance, such as the $\mathcal{H}_2$ norm, should be interpreted from the perspective of the interconnection structure. There have been recent contributions attempting to connect certain properties of the underlying graph with the dynamic behavior of the system [4]–[6]. A detailed treatment of the $\mathcal{H}_2$ performance for consensus networks was recently given in [7]. Despite this progress, it still remains unclear how basic combinatorial properties of the network, such as spanning trees and cycles, directly impact its performance.

A first contribution of this work, therefore, is a characterization of how cycles affect the $\mathcal{H}_2$ performance of a consensus seeking network. The main analytical tool used to derive this result is based on a companion system, known as the *edge agreement problem* [7]. We consider this system with noises and disturbances entering both the process and the measurement and show that cycles can improve the performance. In particular, we show that the length of a cycle and the number of edges it shares with other cycles affects how much it can improve the $\mathcal{H}_2$ performance.

The combinatorial understanding of performance motivates new perspectives for the *design* of consensus networks. Many approaches related to the synthesis of consensus networks have focused on designing graphs to improve the rate of convergence using, for example, convex relaxations, mixed-integer programs, or simple heuristics [8]–[11]. Related design problems, such as leader selection, have also been considered with an $\mathcal{H}_2$ performance objective [12], [13].

The second contribution of this work is a methodology for designing consensus networks. We formulate an optimization problem that aims to add a fixed number of edges to an existing network with the objective of maximizing the improvement of the $\mathcal{H}_2$ performance. While this problem can be considered as a combinatorial one, we develop an ℓ_1 relaxation method for solving it. An important observation regarding the problem formulation is that it can be stated as an ℓ_0-optimization problem, which is a measure of sparsity in a vector. Using recent results from compressive sensing, we are able to relax and re-formulate the problem as an ℓ_1-optimization problem [14], [15]. The ℓ_1-optimization formulation is known to achieve sparse solutions [16], [17], and we demonstrate this in our set-up. An important aspect of the relaxation is the introduction of a weighting mechanism for the optimizer. It turns out this mechanism serves as an important engineering tuning parameter, and we discuss how the combinatorial insights gained from the analysis leads to useful weighting functions, such as cycle lengths and correlations.

This paper is organized as follows. Section §II reviews some fundamental properties of graphs and their algebraic representations. The edge agreement problem and a combinatorial interpretation of its $\mathcal{H}_2$ performance is given in §III. In §IV we formulate the synthesis problem and derive the ℓ_1-relaxation method. Numerical simulations are presented in §V. Finally, we offer some concluding remarks in §VI.

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, simone.schuler, frank.allgower}@ist.uni-stuttgart.de.

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

A. Notations

The notation used is standard. The set of real numbers is denoted by $\mathbb{R}$. For a vector $x \in \mathbb{R}^n$, its transpose is given by x^T and the ith component by x_i; The ijth element of a matrix A is denoted $[A]_{ij}$. The cardinality of a set M is denoted as $|M|$. The ℓ_1-norm of a vector $x \in \mathbb{R}^n$ is defined as $\|x\|_1 = \sum_i |x_i|$. The ℓ_0-norm of a vector is defined as $\|x\|_0 = |\{i \,|\, x_i \neq 0\}|$, the number of non-zero elements in the vector x.[1]

II. Graph Cycles and the Edge Laplacian

We first provide a brief review of concepts from graph theory [18]. A *graph*, denoted $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, consists of a set of *nodes* $\mathcal{V} = \{v_1, \ldots, v_n\}$, and a set of *edges* $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, describing the incidence relation between pairs of nodes. In this work we deal with *undirected* graphs but at times will assign an arbitrary orientation to each edge. In this way, we denote an edge $e \in \mathcal{E}$ with the ordered pair $(v_i, v_j) \in \mathcal{V} \times \mathcal{V}$ as the *directed edge* connecting v_i to v_j; we also use the notation $v_i \sim v_j$ to denote that these two nodes are connected (or adjacent). A *path* is a sequence of distinct nodes such that consecutive nodes are adjacent to each other. If the initial and terminal node of a path are the same, it is called a *cycle*. The length of a path (cycle) is the number of edges traversed in the path sequence. For example, a *triangle* is a cycle of length 3. The *diameter* of a graph, denoted **diam**$[\mathcal{G}]$, is the maximum distance between any two nodes, where the distance is the length of the shortest path. A graph is *connected* if there exists a path between any pair of nodes; otherwise it is called *disconnected*. In this work, we always assume connected graphs.

Any connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be decomposed (non-uniquely) into a *spanning tree*, denoted $\mathcal{T} = (\mathcal{V}, \mathcal{E}_\tau)$, and a graph we term the *cycle graph* $\mathcal{C} = (\mathcal{V}, \mathcal{E} \setminus \mathcal{E}_\tau)$. In this way, we can always express a graph as the union of a tree and cycle graph, i.e. $\mathcal{G} = \mathcal{T} \cup \mathcal{C}$. Note that the edges in $\mathcal{C}$ are used to create cycles in the graph $\mathcal{G}$.

Of particular use in this work are the algebraic representations of graphs [18]. The *incidence matrix* of the graph $\mathcal{G}$ with arbitrary orientation, $E(\mathcal{G}) \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$, is a $\{0, \pm 1\}$-matrix with rows and columns indexed by the vertices and edges of $\mathcal{G}$ such that $[E(\mathcal{G})]_{ik}$ has the value '+1' if node i is the initial node of edge k, '-1' if it is the terminal node, and '0' otherwise. 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 $\mathcal{C}$. For example, a *signed path vector* $\xi \in \mathbb{R}^{\mathcal{E}}$ is a $\{0, \pm 1\}$-vector corresponding to a path in $\mathcal{G}$, such that ξ_i takes the value '+1' ('-1') if edge $e_i \in \mathcal{E}$ is traversed positively (negatively), and '0' otherwise. Any path, therefore, can be expressed using only edges from the sub-graph $\mathcal{T}$. Observe that the length of a path can be computed from its signed path vector as $\xi^T \xi$. Furthermore, a cycle can be expressed using exactly one edge from $\mathcal{C}$, and the remaining edges from $\mathcal{T}$.

The notion that cycles are dependent on edges in a spanning tree can be formalized algebraically by the notion of linear dependence of columns of the incidence matrix. We define the matrix $T_{(\mathcal{T},\mathcal{C})} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}_\mathcal{C}|}$ as [7]

$$T_{(\mathcal{T},\mathcal{C})} \;=\; \left(E(\mathcal{T})E(\mathcal{T})^T\right)^{-1} E(\mathcal{T})^T E(\mathcal{C}), \qquad (1)$$

satisfying $E(\mathcal{T})T_{(\mathcal{T},\mathcal{C})} = E(\mathcal{C})$.

In the following, we express $T_{(\mathcal{T},\mathcal{C})}$ in terms of its columns, $T_{(\mathcal{T},\mathcal{C})} = \left[\begin{array}{ccc} c_1 & \cdots & c_{|\mathcal{E}_\mathcal{C}|} \end{array} \right]$, 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 $R_{(\mathcal{T},\mathcal{C})} = \left[\begin{array}{cc} I & T_{(\mathcal{T},\mathcal{C})} \end{array} \right]$. Using this notation, note that $E(\mathcal{G}) = E(\mathcal{T})R_{(\mathcal{T},\mathcal{C})}$.

The matrix $T_{(\mathcal{T},\mathcal{C})}$ also encodes information related to the length of each cycle, denoted $l(c_i)$, and correlations between different cycles in $\mathcal{G}$. In this direction, we first define the notion of *edge-disjoint cycles* and *correlated cycles*.

Definition 1: Two cycles are said to be *edge-disjoint* if they do not have any edges in common. Two cycles are *correlated* if they are not edge-disjoint.

For correlated cycles, we denote the quantity s_{ij} as the number of edges cycles c_i and c_j share; note that $|c_i^T c_j| = s_{ij}$. Also observe that $c_i^T c_i = l(c_i) - 1$, the length of cycle c_i minus 1. Figure 1 shows a graph with cycles illustrating the definitions.

Proposition 1: The matrix $T_{(\mathcal{T},\mathcal{C})}T_{(\mathcal{T},\mathcal{C})}^T$ encodes the following information about the cycles in $\mathcal{G}$.

1) $[T_{(\mathcal{T},\mathcal{C})}T_{(\mathcal{T},\mathcal{C})}^T]_{ii}$ is the number of times edge τ_i is used to construct the cycles in $\mathcal{C}$.
2) $[T_{(\mathcal{T},\mathcal{C})}T_{(\mathcal{T},\mathcal{C})}^T]_{ij}$ is the number of times edges τ_i and τ_j are used in the same cycle. The sign of the entry is positive if both edges are traversed in the same direction, and negative otherwise.
3) $(I + T_{(\mathcal{T},\mathcal{C})}T_{(\mathcal{T},\mathcal{C})}^T)$ is invertible.

The properties described in Proposition 1 will prove useful in the sequel.

The matrix $T_{(\mathcal{T},\mathcal{C})}$ becomes useful when considering an edge-variant of the Laplacian, that we term the *edge Laplacian* [7],

$$L_e(\mathcal{G}) \;:=\; E(\mathcal{G})^T E(\mathcal{G}). \qquad (2)$$

One of the main results in [7] 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 [7] for the proof.

Theorem 1 ([7]): The graph Laplacian for a connected graph $L(\mathcal{G}) = E(\mathcal{G})E(\mathcal{G})^T$ with cycles is similar to the matrix

$$\left[\begin{array}{cc} L_e(\mathcal{T})R_{(\mathcal{T},\mathcal{C})}R_{(\mathcal{T},\mathcal{C})}^T & 0 \\ 0 & 0 \end{array} \right].$$

We refer to the matrix $L_e(\mathcal{T})R_{(\mathcal{T},\mathcal{C})}R_{(\mathcal{T},\mathcal{C})}^T \in \mathbb{R}^{|\mathcal{E}_\mathcal{T}| \times |\mathcal{E}_\mathcal{T}|}$ as the *essential edge Laplacian*. A direct consequence of Theorem 1 is that for any connected graph the essential

[1] Note that the ℓ_0-norm is not a true norm and we use the term to facilitate the understanding that it represents a sparsity measure for the vector x.

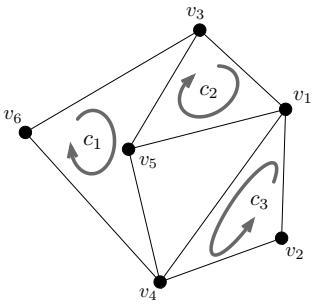


Fig. 1. The cycles c_1 and c_2 are correlated, but are edge-disjoint with c_3.

edge Laplacian has only positive eigenvalues, and they are precisely the non-zero eigenvalues of $L(\mathcal{G})$. Furthermore, when the underlying graph is a tree (i.e. $\mathcal{G} = \mathcal{T}$), then $L_e(\mathcal{T})R_{(\mathcal{T},\mathcal{C})}R^T_{(\mathcal{T},\mathcal{C})} = E(\mathcal{T})^T E(\mathcal{T})$ is a symmetric positive-definite matrix.

The essential edge Laplacian is the main tool used to derive an edge variant of the consensus protocol. It is important to emphasize that the similarity transformation discussed in Theorem 1 preserves both the algebraic properties of the Laplacian, along with structural properties relating the graph to its matrix representation. The benefit of this transformation is explored in the sequel.

III. Performance of Consensus Networks

The standard noise-free consensus model is built on a collection of n single integrator agents that exchange relative state information over an information exchange graph to generate a control [1],

$$\dot{x}(t) \;=\; -L(\mathcal{G})x(t). \tag{3}$$

Noises and external disturbances can be injected into the process as $\dot{x}_i(t) = u_i(t) + w_i(t)$, 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. Using the result of Theorem 1 one can obtain a minimal realization; such a realization leads to what we term the *edge agreement problem*. Its derivation is given in [7] and the reader is referred to that work for details. We present here only the minimal system.

$$\Sigma_e(\mathcal{G}) : \begin{cases} \dot{x}_\tau(t) \;=\; -L_e(\mathcal{T})R_{(\mathcal{T},\mathcal{C})}R^T_{(\mathcal{T},\mathcal{C})}x_\tau(t) + \\ \qquad \left[\begin{array}{cc} E(\mathcal{T})^T & -L_e(\mathcal{T})R_{(\mathcal{T},\mathcal{C})} \end{array}\right] \left[\begin{array}{c} w(t) \\ v(t) \end{array}\right] \\ z(t) \;=\; x_\tau(t). \end{cases} \tag{4}$$

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 $\mathcal{T}$. In this work we treat $\mathcal{T}$ as a given *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]

[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.

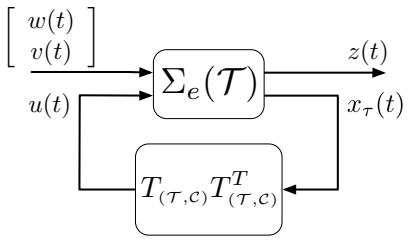


Fig. 2. Cycles as a feedback mechanism.

An interesting interpretation of the minimal system $\Sigma_e(\mathcal{G})$ is that the cycles in the graph can be viewed as an internal feedback mechanism for the system. To elucidate on this idea, consider the following dynamic system over a spanning tree $\mathcal{T}$,

$\dot{x}_\tau(t) \;=\; -L_e(\mathcal{T})x_\tau + L_e(\mathcal{T})u(t) + \Gamma \left[\begin{array}{c} w(t) \\ v(t) \end{array}\right]$, with a state-feedback control $u(t) = -T_{(\mathcal{T},\mathcal{C})}T^T_{(\mathcal{T},\mathcal{C})}x_\tau(t)$, and $\Gamma = \left[\begin{array}{cc} E(\mathcal{T})^T & -L_e(\mathcal{T})R_{(\mathcal{T},\mathcal{C})} \end{array}\right]$. The state-feedback gain, therefore, can be viewed as a cycle-creating gain, and thus leads to the feedback interpretation. This is visualized in Figure 2.

Cycles as a feedback mechanism can lead to a deeper understanding of their role in consensus. Indeed, when cast as a feedback problem one can conclude that cycles can be used to reduce the sensitivity of the system to external disturbances. The two-port feedback paradigm in control systems provides a powerful framework for formulating and solving problems related to *optimal controller design* [19]. Therefore, based on this new interpretation of cycles as feedback, one can now attempt to cast problems related to the *optimal design of graphs* to that of the *optimal synthesis of feedback controllers*. We will return to this after first characterizing the performance of the edge agreement problem.

In the context of consensus systems, the $\mathcal{H}_2$ performance captures how noises entering the system affects the asymptotic deviation of the states from the consensus value. The $\mathcal{H}_2$ performance was originally considered in [7] and we summarize the main result here.

Theorem 2 ([7]): The $\mathcal{H}_2$ performance of edge agreement problem (4) is

$$\|\Sigma_e(\mathcal{G})\|_2^2 \;=\; \frac{1}{2}\mathbf{tr}\left[(R_{(\mathcal{T},\mathcal{C})}R^T_{(\mathcal{T},\mathcal{C})})^{-1}\right] + (n-1). \tag{5}$$

Theorem 2 states that the performance is intimately related to the cycle structure of the graph. However, at first glance it is difficult to decipher the impact of cycles by examining the matrix $(R_{(\mathcal{T},\mathcal{C})}R^T_{(\mathcal{T},\mathcal{C})})^{-1}$. To resolve this, we first consider how the performance changes by adding a single edge to an already existing graph.

Theorem 3: Consider the edge agreement problem over the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with spanning tree $\mathcal{T}$ and its associated performance $\|\Sigma_e(\mathcal{G})\|_2^2$. Adding a single edge $e \in \mathcal{V}\times\mathcal{V}\setminus\mathcal{E}$ to the graph will always improve the performance, and the resulting system has an $\mathcal{H}_2$ performance of

$$\|\Sigma_e(\mathcal{G}\cup e)\|_2^2 \;=\; \|\Sigma_e(\mathcal{G})\|_2^2 - \frac{\left(R_{(\mathcal{T},\mathcal{C})}R^T_{(\mathcal{T},\mathcal{C})}\right)^{-1} cc^T \left(R_{(\mathcal{T},\mathcal{C})}R^T_{(\mathcal{T},\mathcal{C})}\right)^{-1}}{2(1 + c^T\left(R_{(\mathcal{T},\mathcal{C})}R^T_{(\mathcal{T},\mathcal{C})}\right)^{-1}c)}, \tag{6}$$

where c is the cycle formed by adding the new edge e with the underlying spanning tree $\mathcal{T}$.

Proof: The proof is a direct application of the Sherman-Morrison formula for the inverse of a rank-one update [20]. Note that the second term on the right-hand side of (6) must always be positive, showing that the performance always improves with the addition of an edge. ■

An immediate corollary of Theorem 3 reveals the essential combinatorial feature of this result.

Corollary 1: 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

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

Proof: Observe that for a spanning tree $R_{(\mathcal{T},\mathcal{C})} = I$. The proof follows from the interpretation of $T_{(\mathcal{T},\mathcal{C})}$ given in §II and the result of Theorem 3. ■

An interesting and important consequence of Corollary 1 is the relation of the *cycle length* to the improvement in performance. Indeed, when adding only a single edge to a tree, long cycles will yield the largest improvement in the performance. However, it is also easy to verify from Corollary 1 that adding many short and edge-disjoint cycles can improve the performance more than fewer long cycles.

It is also interesting to consider the affect of adding cycles that are correlated.

Corollary 2: Consider the edge agreement problem where the underlying graph is a spanning tree, $\mathcal{T} = (\mathcal{V}, \mathcal{E}_\tau)$, and consider adding 2 edges $e_1, e_2 \in \overline{\mathcal{E}}_\tau$, such that the new edges add 2 new cycles. Then the performance of the edge agreement problem is given as

$$\|\Sigma_e(\mathcal{T} \cup \{e_1, e_2\})\|_2^2 \quad = \quad \|\Sigma_e(\mathcal{T})\|_2^2 - \left(1 - \frac{l(c_1) + l(c_2)}{2(l(c_1)l(c_2) - s_{12}^2)}\right). \quad (8)$$

Proof: The matrix $T_{(\mathcal{T},\mathcal{C})}$ must have two columns and consequently $T_{(\mathcal{T},\mathcal{C})}T_{(\mathcal{T},\mathcal{C})}^T$ has rank 2. The non-zero eigenvalues of $T_{(\mathcal{T},\mathcal{C})}T_{(\mathcal{T},\mathcal{C})}^T$, denoted μ_i, are the same as the non-zero eigenvalues of

$$T_{(\mathcal{T},\mathcal{C})}^T T_{(\mathcal{T},\mathcal{C})} = \begin{bmatrix} l(c_1) - 1 & \pm s_{12} \\ \pm s_{12} & l(c_2) - 1 \end{bmatrix}.,$$

which can be determined analytically. The result can then be obtained using the Sherman-Morrison formula for the inverse of a rank-one update [20]. ■

Corollary 2 leads to the interpretation that correlated cycles are not as beneficial as edge-disjoint cycles. In this direction, we observe that there is a fundamental performance tradeoff that must be considered when adding edges to a consensus network. On the one hand, long and edge disjoint cycles provide the greatest performance improvement. However, longer cycles are also more likely to be correlated with other cycles. These insights will prove important when considering the design of consensus networks.

IV. Design of Consensus Networks

The analysis results provide a combinatorial interpretation of system-theoretic properties of the consensus network. More importantly, these results can be used as guidelines for considering the design of consensus networks. When combined with the feedback interpretation of Figure 2, we can consider formal approaches for the synthesis of consensus systems. The synthesis problem we consider then takes the following form.

Problem 1 (Consensus Design): Consider a consensus network over a spanning tree $\mathcal{T} = (\mathcal{V}, \mathcal{E}_\tau)$ and a set of candidate edges in $\mathcal{E}_c = (\mathcal{V} \times \mathcal{V} \setminus \mathcal{E}_\tau)$. Add k edges to $\mathcal{T}$ from the set $\mathcal{E}_c$ that leads to the largest improvement in the $\mathcal{H}_2$ performance of the edge agreement problem. That is, solve the following optimization problem:

$$\min_{T_{(\mathcal{T},\mathcal{C})} \in \mathbb{R}^{|\mathcal{V}| \times k}} \quad \|\Sigma_e(\mathcal{G})\|_2, \quad (9)$$

subject to the constraint that $T_{(\mathcal{T},\mathcal{C})}$ satisfies the relationship induced by (1).

The main challenge associated with Problem 1 is that the decision to add a new cycle to the tree is a *binary* one; either a candidate edge is added or not added. This means that Problem 1 falls under the realm of *mixed-integer programming* (MIP), implying it is a computationally difficult problem to solve [21]. We address this problem by considering an ℓ_1 relaxation of the problem. Before discussing the relaxation, we first formulate a more detailed description of the problem (9).

To begin, note that given a spanning tree $\mathcal{T} = (\mathcal{V}, \mathcal{E}_\tau)$, all the candidate edges that can be added belong to the set $\mathcal{E}_c = (\mathcal{V} \times \mathcal{V} \setminus \mathcal{E}_\tau)$ with cardinality $(1/2)(n-1)(n-2)$. The selection of edges can be equated to the determining of weights $w_i \in \{0, 1\}$ assigned to each edge in $\mathcal{E}_c$. In this direction, the essential edge Laplacian for an arbitrary graph with spanning tree $\mathcal{T}$ can be represented as $L_e(\mathcal{T}) \left(I + T_{(\mathcal{T},\overline{\mathcal{T}})} W T_{(\mathcal{T},\overline{\mathcal{T}})}^T\right)$, where $W = \mathbf{diag}\{w_1, \ldots, w_{|\overline{\mathcal{E}}_\tau|}\}$, and $\overline{\mathcal{T}}$ is the complement of the graph $\mathcal{T}$, with edge-set $\mathcal{E}_c$.

The two-port feedback interpretation of the edge agreement problem leads to a standard transformation of the objective in (9) to a (mixed-integer) semi-definite program; see for example [22].

$$\min_{M, w_i} \quad \mathbf{tr}\,[M] \quad (10a)$$

$$\text{s.t.} \quad \begin{bmatrix} M & I \\ I & I + T_{(\mathcal{T},\overline{\mathcal{T}})} W T_{(\mathcal{T},\overline{\mathcal{T}})} \end{bmatrix} \geq 0 \quad (10b)$$

$$\sum_i w_i = k, \; w_i \in \{0, 1\} \quad (10c)$$

A common strategy for relaxing this integer program is by relaxing the integer constraints to a box constraint; the weights can take values continuously on the interval $[0,\, 1]$. The main drawback of this approach is there will be no guarantee that the solution will be integer, or even sparse. An attempt to force sparse solutions is by reformulating the objective (10a) by including an ℓ_0 objective, $\mathbf{tr}\,[M] + \|w\|_0$.

Note, however, that optimization of the ℓ_0 norm is still combinatorial [14]. Similar to the convex relaxation for rank minimization in [23], we will use the convex envelope of $\|w\|_0$ defined next.

Let the map f be defined as $f : \mathbb{X} \mapsto \mathbb{R}$, where $\mathbb{X} \subseteq \mathbb{R}^n$. The convex envelope of f (on $\mathbb{X}$) , denoted f_{env}, is defined as the point-wise largest convex function g such that $g(x) \leq f(x)$ for all $x \in \mathbb{X}$.

Lemma 1 ([24]): The convex envelope of the function $f = \|x\|_0 = \sum_{i=1}^n |\text{sign}(x_i)|$ on $\mathbb{X} = \{x \in \mathbb{R}^n | \|x\|_\infty \leq 1\}$ is $f_{\text{env}}(x) = \|x\|_1 = \sum_{i=1}^n |x_i|$.

With this, we can relax the non-convex ℓ_0-minimization with the convex ℓ_1-minimization objective. note that this can be solved using linear programming. Additionally, this is the best possible convex relaxation since the ℓ_1-norm is the convex envelope of the ℓ_0-norm.

As described in [16], *reweighted* ℓ_1-minimization can be used to improve the results of the minimization. In this direction, weights $m_i > 0$ can be assigned to each variable w_i as

$$\min_{M,\, w_i} \ \mathbf{tr}\,[M] + \sum_{i=1}^{n} m_i w_i. \tag{11}$$

These weights should be considered as free design parameters. In fact, in the context of the results of §III, the weights can be chosen to promote or discourage certain edges from appearing; for example, edges forming long cycles or edge-disjoint cycles.

This brings us to the complete ℓ_1-relaxation of Problem 1,

$$\min_{M,w_i} \quad \alpha \mathbf{tr}\,[M] + (1-\alpha) \sum_i m_i w_i \tag{12a}$$

$$\text{s.t.} \quad \begin{bmatrix} M & I \\ I & I + T_{(\mathcal{T},\overline{\mathcal{T}})} W T_{(\mathcal{T},\overline{\mathcal{T}})} \end{bmatrix} \geq 0 \tag{12b}$$

$$\sum_i w_i = k, \quad 0 \leq w_i \leq 1. \tag{12c}$$

Here we have also introduced a weighting factor $\alpha \in [0,\, 1]$ as a tuning parameter for the relative emphasis on each term in the objective function.

We now discuss possible weighting options for the relaxed version. In the following, we explore a variety of edge weights based on the results of §III. Possible weights are summarized in Table I. Weights can also be combined. For

TABLE I

WEIGHTING OPTIONS FOR CONSENSUS DESIGN.

long cycle weight	$m_i^{lc} = \mathbf{diam}(\mathcal{G}) - \|c_i\|_1 + 1$
short cycle weight	$m_i^{sc} = \|c_i\|_1$
cycle correlation weight	$m_i^s = \frac{1}{\|\mathcal{E}_c\|} \sum_{j \neq i} \left\|[T_{(\mathcal{T},\mathcal{C})}]_{ij}\right\|$
uniform weight	$m_i = 1$

example, one might choose a weight of the form $\beta m_i^{lc} m_i^s$ to try and promote long cycles that are not highly correlated.

While in practice these weighting functions will lead to sparse and $\{0, 1\}$ solutions for the cycle weights w_i, there may arise situations where the optimization problem does not return an integer solution. This can be remedied via an iterative approach, where problem (12) is solved repeatedly and the weights are updated at each iteration count h as $m_i^{(h+1)} = \frac{1}{w_i^{(h)} + \epsilon}$, for some $\epsilon > 0$. See [16] for more details on this approach.

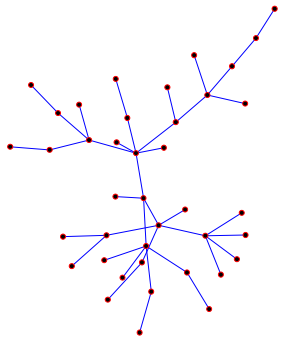


Fig. 3. A spanning tree on 30 nodes.

As discussed earlier, the first step of the algorithm, i.e., the initial choice of edge weights, greatly influences the solution. From an engineering design stand-point, this can be seen as a favorable feature. Indeed, there may be additional features a designer might want to promote when solving Problem 1 without including additional constraints. This can be accomplished via an appropriate choice of edge weights. This point will be illustrated in the simulations provided in the sequel.

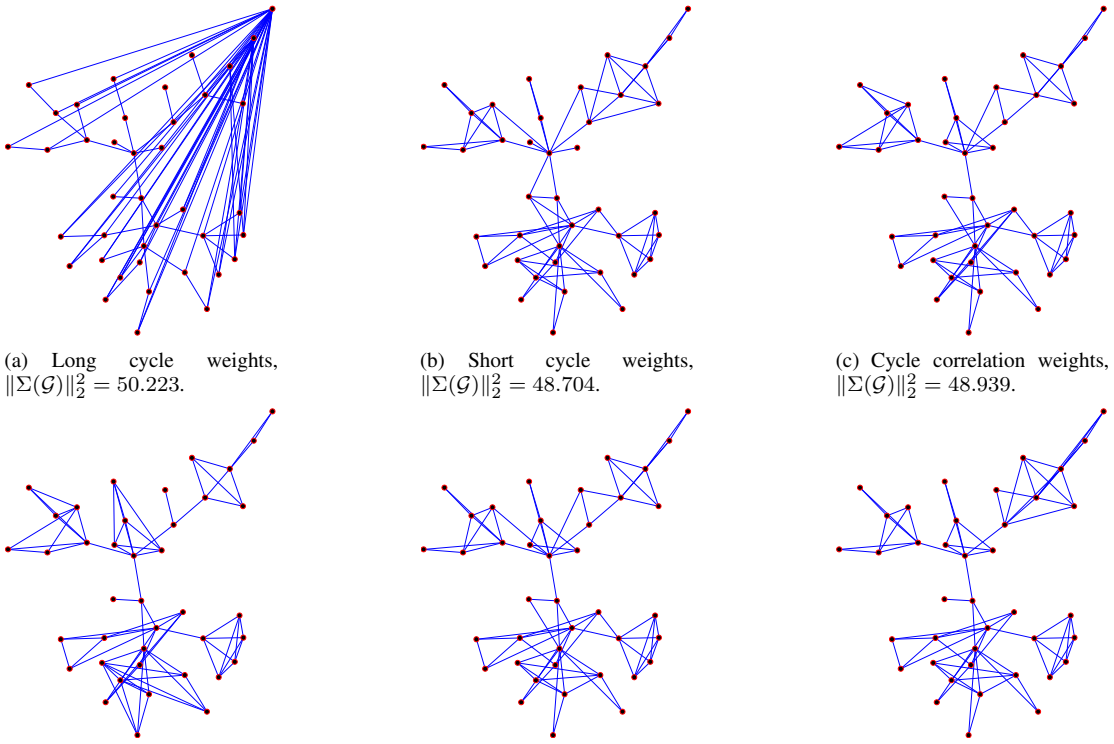
V. Simulation Example

In this section we demonstrate the design procedure described in §IV with a few numerical examples. For each example we will work with the same spanning tree graph on $|\mathcal{V}| = 40$ nodes, generated randomly in MATLAB; see Figure 3. The $\mathcal{H}_2$ performance for this graph can be determined as $\|\Sigma(\mathcal{T})\|_2^2 = 58.5$ and for the complete graph $\|\Sigma(K_n)\|_2^2 = 39.975$. The longest cycle in this graph is determined by its diameter, $\mathbf{diam}(\mathcal{G}) = 9$.

For this example, there are $|\mathcal{E}_c| = 741$ possible edges that can be added. We will consider Problem 1 while attempting to add 40 new edges. First, we compare the $\mathcal{H}_2$ performance when using the different weighting functions in problem (12). The resulting graphs are shown in Figure 4. Observe that the different weighting options generate graphs with qualitatively different features, as expected by the analysis. The resulting performance for each case is given besides the captions in Figure 4. In this simulation, choosing a weight that attempts to promote short cycles (m_i^s) produces the best result, while the weights promoting long cycles produced the worst performance, likely due to the large number of correlated cycles.

VI. Concluding Remarks

This work provided a combinatorial characterization of how cycles impact the $\mathcal{H}_2$ performance of consensus networks. In particular, the purely combinatorial property of cycle lengths and cycle correlations were shown to directly impact the $\mathcal{H}_2$ performance of the system. The analytic results were then used to formulate an optimization problem for the design of consensus networks. Using an ℓ_1-relaxation,



(a) Long cycle weights, $\|\Sigma(\mathcal{G})\|_2^2 = 50.223$.

(b) Short cycle weights, $\|\Sigma(\mathcal{G})\|_2^2 = 48.704$.

(c) Cycle correlation weights, $\|\Sigma(\mathcal{G})\|_2^2 = 48.939$.

(d) Long cycle and correlation weights, $\|\Sigma(\mathcal{G})\|_2^2 = 50.017$.

(e) Short cycle and correlation weights, $\|\Sigma(\mathcal{G})\|_2^2 = 48.795$.

(f) Uniform weights, $\|\Sigma(\mathcal{G})\|_2^2 = 49.489$.

Fig. 4. The weighting function for cycle design has a large affect on the resulting graph.

the problem was transformed into a semi-definite program. This relaxation turns out to be very sensitive to the weighting function used on the edges. This provides an important tuning parameter for design of these systems. The results were demonstrated via some numerical simulations.

REFERENCES

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

[2] I. Akyildiz, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[3] R. Olfati-saber, J. A. Fax, R. M. Murray, and R. Olfati-Saber, "Consensus and Cooperation in Networked Multi-Agent Systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

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

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] E. J. Candès, J. K. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[15] D. L. Donoho, "Compressed sensing," *IEEE Trans. Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[16] 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.

[17] 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.

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

[19] G. Dullerud and F. Paganini, *A course in robust control theory: a convex approach*. New York: Springer-Verlag, 2000.

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

[21] A. Schrijver, *Theory of linear and integer programming*. West Sussex, England: John Wiley & Sons Ltd., 1986.

[22] C. W. Scherer and P. Gahinet, "Multiobjective output-feedback control via LMI optimization," *IEEE Transactions on Automatic Control*, vol. 42, no. 7, pp. 896–911, Apr. 1997.

[23] M. Fazel, H. Hindi, and S. Boyd, "A rank minimization heuristic with application to minimum order system approximation," in *Proc. American Control Conference (ACC)*, 2001, pp. 4734–4739.

[24] M. Fazel, "Matrix rank minimization with applications," Ph.D. dissertation, Department of Electrical Engineering, Stanford University, 2002.