2019 IEEE 58th Conference on Decision and Control (CDC)
Palais des Congrès et des Expositions Nice Acropolis
Nice, France, December 11-13, 2019

On the Computation of the Distance-based Lower Bound on Strong Structural Controllability in Networks

Mudassir Shabbir, Waseem Abbas, and A. Yasin Yazıcıoğlu

***Abstract*— A network of agents with linear dynamics is strong structurally controllable if agents can be maneuvered from any initial state to any final state independently of the coupling strengths between agents. If a network is not strong structurally controllable with given input nodes (leaders), then the dimension of strong structurally controllable subspace quantifies the extent to which a network can be controlled by the same inputs. Computing this dimension exactly is computationally challenging. In this paper, we study the problem of computing a sharp lower bound on the dimension of strong structurally controllable subspace in networks with Laplacian dynamics. The bound is based on a sequence of vectors containing distances between leaders and the remaining nodes in the underlying network graph. Such vectors are referred to as the distance-to-leader vectors. We provide a polynomial time algorithm to compute a desired sequence of distance-to-leader vectors with a fixed set of leaders, which directly provides a lower bound on the dimension of strong structurally controllable subspace. We also present a linearithmic approximation algorithm to compute such a sequence, which provides near optimal solutions in practice. Finally, we numerically evaluate and compare our bound with other bounds in the literature on various networks.**

I. Introduction

Network controllability has been an important research topic in the broad areas of network and cooperative control. A primary goal is to drive a network of dynamical agents, each of which shares information with a subset of others, to a desired state by an external control signal given to a subset of agents referred to as leaders. Building on the classical controllability results from linear systems theory and utilizing tools from graph theory, new insights on controlling networks have been developed in recent years. An important aspect of network controllability is the effect of coupling strengths between agents, which are often represented by edge weights in the underlying network graph. For a fixed set of leaders, a network's controllability might change with different choices of edge weights. However, it might be infeasible or extremely difficult in practice to assign precise edge weights due to uncertainties, numerical inaccuracies, and inexact system parameters.

Consequently, we desire to characterize controllability while discounting the effect of edge weights. In other words, we want to relate controllability purely with the structure of the underlying network graph, and independent of edge weights. Such a notion of controllability is referred to as the *strong structural controllability (SSC)*. It is easy to verify if a network is strong structurally controllable. However, if it is not, then computing how much of the network is strong structurally controllable, or more precisely the dimension of strong structurally controllable subspace (formally defined in Section II-B) is an extremely challenging problem.

In this paper, we study the problem of computing a tight lower bound on strong structural controllability of networks with Laplacian dynamics. We consider a bound proposed in [1] that depends on distances between nodes in a graph. The distance based bound is useful in characterizing SSC in terms of the underlying network structure, selecting leaders, and exploiting trade-off between controllability and robustness [2]. The main idea is to obtain distances between leaders and other nodes, arrange them in vectors called distance-to-leader vectors, and then construct a particular sequence of such vectors satisfying some montonicity conditions. Computing distances between nodes is straightforward, however, constructing sequences that ultimately provide bound on SSC is computationally challenging. We provide efficient algorithms with performance guarantees to compute such sequences. Our main contributions are as follows:

1) We provide a dynamic programming based exact algorithm to compute optimal sequence of vectors consisting of distances between leaders and other nodes in $O(m(n \log n + n^m))$ time, which directly gives a tight lower bound on SSC of networks. Here m is the number of leaders and n is the total number of nodes in the network.
2) We propose an approximation algorithm that computes near-optimal sequence of distance-to-leader vectors in practice and takes $O(mn \log n)$ time. If there exists a sequence of length n of distance-to-leader vectors, then the network is SSC, and our greedy algorithm always returns such a sequence if it exists.
3) We numerically evaluate and compare our bound with another bound that is based on the notion of zero forcing sets (ZFS).

A. Related Work

The notion of strong structural controllability was introduced in [3] and the first graph-theoretic condition for single input systems was presented. For multi-input systems, [4] provided a condition to check SSC in $\mathcal{O}(n^3)$ time, where n is the number of nodes. Authors in [5] refined previous

M. Shabbir is with the Computer Science Department at the Information Technology University of the Punjab, Lahore, Pakistan (Email: mudassir@rutgers.edu).

W. Abbas is with the Electrical Engineering Department at the Information Technology University of the Punjab, Lahore, Pakistan (Email: w.abbas@itu.edu.pk).

A. Y. Yazıcıoğlu is with the Department of Electrical and Computer Engineering at the University of Minnesota, Minneapolis, MN, USA (Email: ayasin@umn.edu).

978-1-7281-1398-2/19/$31.00 ©2019 IEEE

Authorized licensed use limited to: Qingdao University. Downloaded on October 29,2020 at 06:26:02 UTC from IEEE Xplore. Restrictions apply.

results and further provided a characterization of SSC. To check if the system is strong structurally controllable with given inputs, an algorithm based on constrained matchings in bipartite graphs with time complexity $\mathcal{O}(n^2)$ was given in [6]. In [7], an algorithm with a runtime linear in the number of nodes and edges was presented to verify whether a system is strong structurally controllable. The relationship of SSC and zero forcing sets (ZFS) was explored in [8], [9], and it was established that checking if a system is strong structurally controllable with given input nodes is equivalent to checking if the set of input nodes is a ZFS in the underlying network graph.

The notion of strong structurally controllable subspace [10], which is an extension of the ordinary controllable subspace, is particularly useful to quantify controllability in cases where networks are not strong structurally controllable. In fact, the dimension of such a subspace (formally described in Section II-B) quantifies how much of the network is controllable in the strong structural sense (that is, independently of edge weights) with a given set of inputs. Some lower bounds on the dimension of SSC have been proposed in the literature. In [10], a lower bound based on the derived set of input nodes (leaders) was presented, which was further studied in [11], [12]. With a single input node, the dimension of SSC can be at least the diameter of the underlying network graph [13]. In [1], a tight lower bound on the dimension of SSC was proposed that was based on distances between leaders and other nodes in the graph. The main idea was to compute a certain sequence of vectors consisting of distances between nodes in graph. The bound was used to explore trade-off between SSC and network robustness in [2].

Further studies in this direction include leader selection to achieve desired structural controllability (e.g., [14], [15]), and network topology design for a desired control performance (e.g., [16], [17]).

II. Preliminaries and a Lower Bound on Strong Structural Controllability

A. Notations

We consider a network of n dynamical agents represented by a simple (loop-free) undirected graph $G = (V, E)$ where the node set V represent agents, and the edge set E represents interconnections between agents. An edge between v_i and v_j is denoted by e_{ij}. The *neighborhood* of node v_i is $\mathcal{N}_i \triangleq \{v_j \in V : e_{ij} \in V\}$. The *distance* between v_i and v_j, denoted by $d(v_i, v_j)$, is simply the number of edges in the shortest path between v_i and v_j. Edges can be *weighted*, and the *weighting function* $w : E \to \mathbb{R}^+$ assigns positive weight w_{ij} to the edge e_{ij}. These weights define the coupling strength between nodes. If there is no edge between v_i and v_j, then we define $w_{ij} = 0$.

Each agent $v_i \in V$ has a state $x_i \in \mathbb{R}$ and the overall state of the system, without loss of generality, is $x = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T \in \mathbb{R}^n$. Agents update states following the Laplacian dynamics given below.

$$\dot{x}_i = -L_w x + Bu, \tag{1}$$

where $L_w \in \mathbb{R}^{n\times n}$ is the weighted *Laplacian matrix* of G and is defined as $L_w = \Delta - A_w$. Here, $A_w \in \mathbb{R}^{n\times n}$ is the weighted *adjacency matrix*, in which the ij^{th} entry is,

$$[A_w]_{ij} = \begin{cases} w_{ij} & \text{if } e_{ij} \in E, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

and $\Delta \in \mathbb{R}^{n\times n}$ is a degree matrix as below.

$$[\Delta]_{ij} = \begin{cases} \sum_{k=1}^{n} A_{ik} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The matrix $B \in \mathbb{R}^{n\times m}$ in (1) is an input matrix. m is the number of leaders (inputs), which are the nodes to which an external control signal is applied. Let $V_\ell = \{\ell_1, \ell_2, \cdots, \ell_m\} \subset V$ be the set of *leaders*, then $B_{ij} = 1$ if node $i \in v$ is also a leader ℓ_j, otherwise $B_{ij} = 0$.

B. Strong Structural Controllability (SSC)

A state $x_f \in \mathbb{R}^n$ is a *reachable state* if there exists an input u that can drive the network in (1) from the origin to x_f in a finite amount of time. A network $G = (V, E)$ with edge weights w and leaders V_ℓ is called *completely controllable* if every point in $\mathbb{R}^n$ is reachable. Complete controllability can be checked by computing the rank of the following matrix, called *controllability matrix*.

$$\Gamma(L_w, B) = \begin{bmatrix} B & (-L_w)B & (-L_w)^2B & \cdots & (-L_w)^{n-1}B \end{bmatrix}.$$

Network is completely controllable if and only if the rank of $\Gamma(L_w, B)$ is n, in which case (L_w, B) is called a *controllable pair*. Note that edges in G define the *structure*—location of zero and non-zero entries in the Laplacian matrix—of the underlying graph. The exact values of non-zero entries, and hence, the rank of resulting controllability matrix depends on the weights assigned to edges. For a given graph $G = (V, E)$ and V_ℓ leaders, $\text{rank}(\Gamma(L_w, B))$ might be different from $\text{rank}(\Gamma(L_{w'}, B))$, where w and w' are two different choices of edge weights.

A network $G = (V, E)$ with V_ℓ leaders is *strong structurally controllable* if and only if (L_w, B) is a controllable pair for any choice of non-zero edge weights w, or in other words, $\text{rank}(\Gamma(L_w, B)) = n$ for all w. At the same time, the *dimension of strong structurally controllable subspace*, or simply the *dimesnion of SSC*, denoted by $\gamma(G, V_\ell)$, is

$$\gamma(G, V_\ell) = \min_w \left(\text{rank } \Gamma(L_w, B)\right). \tag{4}$$

Roughly, $\gamma(G, V_\ell)$ quantifies how much of the network can be controlled with V_ℓ leaders and with any choice of edge weights.

C. A Distance-based Lower Bound on the Dimension of SSC

We use a tight lower bound on the dimension of SSC as proposed in [1]. The bound is based on the distances between nodes in a graph. Assuming m leaders $V_\ell = \{\ell_1, \cdots, \ell_m\}$, we define *distance-to-leader* vector for each $v_i \in V$ as

$$D_i = \begin{bmatrix} d(\ell_1, v_i) & d(\ell_2, v_i) & \cdots & d(\ell_m, v_i) \end{bmatrix}^T \in \mathbb{Z}^m.$$

Authorized licensed use limited to: Qingdao University. Downloaded on October 29,2020 at 06:26:02 UTC from IEEE Xplore. Restrictions apply.

The j^{th} component of D_i, denoted by $D_{i,j}$, is $d(\ell_j, v_i)$. Next, we define a sequence of distance-to-leader vectors, called as *pseudo-monotonically increasing* sequence below.

Definition (*Pseudo-monotonically Increasing Sequence (PMI)*) A sequence of distance-to-leader vectors $\mathcal{D}$ is PMI if for every i^{th} vector in the sequence, denoted by $\mathcal{D}_i$, there exists some $j \in \{1, 2, \cdots, m\}$ such that

$$\mathcal{D}_{i,j} < \mathcal{D}_{i',j}, \quad \forall i' > i. \tag{5}$$

We say that $\mathcal{D}_i$ satisfies the *PMI property* at coordinate j whenever $\mathcal{D}_{i,j} < \mathcal{D}_{i',j}$, $\forall i' > i$.

An example of distance-to-leader vectors is illustrated in Figure 1. A PMI sequence of length five is

$$\mathcal{D} = \left[\begin{bmatrix} 3 \\ \textcircled{0} \end{bmatrix}, \begin{bmatrix} 2 \\ \textcircled{1} \end{bmatrix}, \begin{bmatrix} \textcircled{0} \\ 3 \end{bmatrix}, \begin{bmatrix} 2 \\ \textcircled{2} \end{bmatrix}, \begin{bmatrix} \textcircled{1} \\ 3 \end{bmatrix} \right]. \tag{6}$$

Indices of circled values in (6) are the coordinates at which

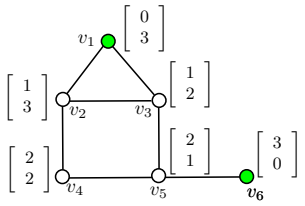


Fig. 1: A network with two leaders $V_\ell = \{\ell_1, \ell_2\} = \{v_1, v_6\}$, along with the distance-to-leader vectors of nodes. A PMI sequence of length five is $\mathcal{D} = [\mathcal{D}_1\ \mathcal{D}_2\ \cdots\ \mathcal{D}_5] = [D_6\ D_5\ D_1\ D_4\ D_2]$.

the corresponding distance-to-leader vectors are satisfying the PMI property. The length of PMI sequence of distance-to-leader vectors is related to the dimension of SSC as stated in the following result.

Theorem 2.1: [1] If δ is the length of longest PMI sequence of distance-to-leader vectors in a network $G = (V, E)$ with V_ℓ leaders, then

$$\delta \leq \gamma(G, V_\ell). \tag{7}$$

Problem: Our goal is to efficiently compute a PMI sequence of maximum length, and hence, a lower bound on the dimension of SSC. Moreover, we would like to compare the distance-based bound with the other known bounds.

In the next section, we provide an algorithm to compute a PMI sequence of maximum length, and in Section IV, we provide a greedy approximation algorithm.

III. Computing SSC Bound – An Exact Algorithm

Main result of this section is as follows:[1]

Theorem 3.1: Given a graph G on n vertices, and m leaders, a longest PMI sequence of distance-to-leader vectors can be computed in $O(m(n \log n + n^m))$.

We will provide a dynamic programming based algorithm of prescribed time complexity.

[1]Due to space limitations, some proofs are omitted and can be found in [18], which is an extended version of this paper.

Note that the distance-to-leader vector D_i can be viewed as a point in $\mathbb{Z}^m$ and problem of computing longest PMI sequence is equivalent to finding a corresponding subsequence of such points. We find that treating D_i's as points improves the readability of our algorithms, so we will stick to this view for the rest of the paper. With a slight abuse of notation, $D_{i,j}$ will now represent coordinate j of a point $D_i \in \mathbb{Z}^m$ for $1 \leq j \leq m$. We start our discussion with the following:

Fact 3.2: Without loss of generality, we may assume that points D_i are distinct.

Otherwise we can throw away multiple copies of the same point as duplicate points can not satisfy the PMI property (defined above) on any coordinate.

The following observation is crucial to our algorithms.

Observation 3.3: Given a set of points $D_1, D_2 \ldots, D_n$, if there exists a point D_i and an index j such that $D_{i,j} < D_{i',j}$ for all $D_i \neq D_{i'}$, then D_i is a unique minimum point in the direction j and there is a longest PMI sequence that starts with D_i.

However, it is possible that there is no unique minimum in any direction. This leads us to the definition of a *conflict* and *conflict-partition*.

Definition *(Conflict-partition)* A *conflict* is a set of points $\mathcal{X}$ that can be partitioned into $\mathcal{X}_1, \mathcal{X}_2 \ldots, \mathcal{X}_m$ such that all points $D_p \in \mathcal{X}_i$ have $D_{p,i} = D_{q,i}$ if $D_q \in \mathcal{X}_i$ and $D_{p,i} \leq D_{q,i}$ if $D_q \notin \mathcal{X}_i$. Further, $|\mathcal{X}_i| > 1$ for all i. Such a partition is called *conflict-partition* or *c-partition* for short.[2]

An example of conflict is illustrated in Figure 2.

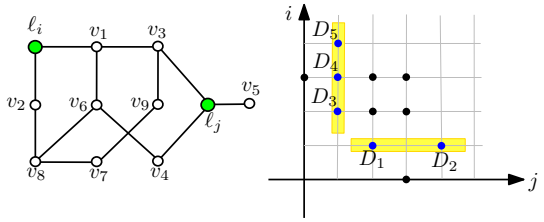


Fig. 2: A graph with two leaders and the plot of distance-to-leader vectors as points in a plane. Point set $\mathcal{X} = \{D_1, D_2, D_3, D_4, D_5\}$ constitutes a conflict, where $\mathcal{X}_i = \{D_3, D_4, D_5\}$ and $\mathcal{X}_j = \{D_1, D_2\}$.

It is easy to see that a PMI sequence can not contain all points in a conflict. In fact, we can strictly bound the number of points from a conflict that can be included in a PMI sequence.

Lemma 3.4: Let $\mathcal{X}_1, \mathcal{X}_2 \ldots, \mathcal{X}_m$ be a c-partition of a conflict $\mathcal{X}$ for a given set of points, then any PMI sequence contains at most $|\mathcal{X}| - \min(|\mathcal{X}_1|, |\mathcal{X}_2|, \ldots, |\mathcal{X}_m|) + 1$ points from $\mathcal{X}$.

A proof of the above lemma is available in [18]. As an example, consider points $\mathcal{X} = \{D_1, D_2, D_3, D_4, D_5\}$ in Figure 2. In this set, there are two points with the minimum i^{th} coordinate and three points with the minimum

[2]In general, parts of a partition do not intersect. For the lack of a better term, we are slightly abusing this term in the sense that parts $\mathcal{X}_i$ intersect at at most one element.

Authorized licensed use limited to: Qingdao University. Downloaded on October 29,2020 at 06:26:02 UTC from IEEE Xplore. Restrictions apply.

j^{th} coordinate. If D_1 is picked as the first point (among this set) we must either drop D_2 or all D_3, D_4, D_5 for future consideration in the PMI sequence. Similarly if D_3 is picked before everyone else, we can not pick either of D_4, D_5, or any of D_1, D_2 for future consideration regardless of the other points. Note that the bound in Lemma 3.4 is tight: if we remove $|\mathcal{X}_i| - 1$ points from the smallest part of a c-partition, all remaining points can easily satisfy the PMI property on coordinate i unless some of these remaining points are included in any other conflict.

A. Recursive Algorithm

In this section, we design a recursive algorithm that we will covert into a dynamic programming approach in Section III-B. First we define a few notations. In the following, $Ł^i$ denotes a list of points ordered by non-decreasing i^{th} coordinate. $Ł^i_j$ denotes the j^{th} point in the list $Ł^i$, and $Ł^i_{j,k}$ is the (integer) value of k^{th} coordinate of $Ł^i_j$ [3]. Let D be a set of n points in $\mathbb{Z}^m$. In the beginning we will sort all points with respect to all coordinates. So we will get m lists $\{Ł^1, Ł^2, \ldots, Ł^m\}$ of n points each. This enables us to perform a sweep-line (sweep-hyperplane to be precise) algorithm. We will return a sequence $\mathcal{D}$ of points which is initially empty. Without loss of generality all points in D are distinct. If there is a point D_q in D with a unique minimum along some coordinate j, we will add D_q at the end of current PMI sequence $\mathcal{D}$, remove D_q from all lists $Ł^i$, and recursively continue. Otherwise, if there is no unique point D_q with a minimum value along any axis, then for each i there are multiple points in $Ł^i$ whose i^{th} coordinate is equal to $Ł^i_{1,i}$. As suggested by Lemma 3.4, we can not include all of these points to PMI, so we need to make a decision to include some of these points and exclude the other points. We will recursively consider all possibilities and pick the one that results in longest PMI sequence. We outline the details in Algorithm 1, and state the following proposition (proof is available in [18]).

Proposition 3.5: Algorithm 1 returns a longest PMI sequence of distance-to-leader vectors in time $O(m^{(n-m)/2})$.

An example run of the algorithm on the graph in Figure 1 with $V_\ell = \{v_1, v_6\}$ is illustrated in [18]. Algorithm 1 takes prohibitively exponential time even for the case of two leaders. Next, we design an algorithm that is based on dynamic programming and returns an optimal solution in polynomial time when the number of leaders is fixed.

B. Dynamic Programming Algorithm

In this section we present a dynamic programming algorithm that returns length of a longest PMI sequence given distance-to-leader vectors as a set of points. To obtain a longest PMI sequence, standard augmentation methods can be easily employed.

Let $c_1, c_2, \ldots, c_m$ be a set of non-negative integers and $\mathcal{D}^{[c_1,c_2,\ldots,c_m]}$ be a longest PMI sequence in which the value at i^{th} coordinate of any point is at least c_i. Let $\alpha^{[c_1,c_2,\ldots,c_m]}$ be the length of such a sequence. Our algorithm will memoize on $\alpha^{[c_1,c_2,\ldots,c_m]}$.

[3]We recommend to use linked priority queues or similar data structure for these lists so that one could easily delete a point from lists while maintaining respective orders in logarithmic time.

Algorithm 1 Recursive Algorithm for PMI

```
1: procedure PMI-R(Ł^1, Ł^2 ..., Ł^m)  ▷ Recursive routine
2:     if |Ł^1| <= 1 then
3:         return Ł^1                  ▷ one point is always a PMI
4:     end if
5:     X_i ← {Ł^i_j : Ł^i_{j,i} = L^i_{1,i}} for all i.
6:     if ∃i such that |X_i| = 1 then    ▷ Check for unique min.
7:         return [Ł^i_1  PMI-R(Ł^1 \ X_i, Ł^2 \ X_i, ..., Ł^m \ X_i)].
8:     else
9:         for i ∈ 1 : m do▷ Check point in each direction
10:            D^i ← [Ł^i_1  PMI-R(Ł^1 \ X_i, ..., Ł^m \ X_i)]
11:        end for
12:        return largest D^i
13:    end if
14: end procedure
```

From our discussion in Section III-A, we conclude that $\alpha^{[c_1,c_2,\ldots,c_m]}$ can be obtained by the following recurrence:

$$\alpha^{[c_1,c_2,\ldots,c_m]} = \max_{1\le i\le m} (\alpha^{[c_1,c_2,\ldots,c_i+1,\ldots,c_m]} + \mathbf{1}_{c_i}), \quad (8)$$

where

$$\mathbf{1}_{c_i} = \begin{cases} 1 & \text{if } \exists\, D_p \text{ s.t. } D_{p,i} = c_i \text{ and } D_{p,j} \ge c_j, \forall j \ne i. \\ 0 & \text{otherwise.} \end{cases} \quad (9)$$

We plan to pre-compute and memoize all *required* values of $\alpha^{[c_1,\ldots c_d]}$ in a table. Clearly there are infinitely many possible values for c_i but we observe the following:

Observation 3.6: Let $Ł^i_{j,i}$, and $Ł^i_{j+1,i}$ be i^{th} coordinate values of two consecutive points in $Ł^i$ (as defined previously), then

$$\alpha^{[c_1,c_2,\ldots,x,\ldots,c_m]} = \alpha^{[c_1,c_2,\ldots,Ł^i_{j+1,i},\ldots,c_m]}$$

for all $Ł^i_{j,i} < x \le Ł^i_{j+1,i}$.

Observation 3.6 implies that there are at most n different values for each variable c_i, which gives at most n unique values for $\alpha^{[c_1,c_2,\ldots,c_m]}$. Thus, we only keep a table of size n^m for computation and storage of solutions to all subproblems. The details of dynamic program are in Algorithm 2. For some intuition on the working of dynamic program, we refer the reader to Figure 3.

We provide the proof of Proposition 3.7 and illustrate the algorithm in detail through an example in [18].

Proposition 3.7: Algorithm 2 computes a PMI sequence of maximum length in time $O(m \times (n \log n + n^m))$ where m is the number of leaders and n is total number of nodes.

Theorem 3.1 follows directly from Proposition 3.7.

Remark 3.8: For an implementation on platforms that do not offer ready to use multi-dimensional data-structures, we find it useful to hash the values of computed solution to a

Authorized licensed use limited to: Qingdao University. Downloaded on October 29,2020 at 06:26:02 UTC from IEEE Xplore. Restrictions apply.

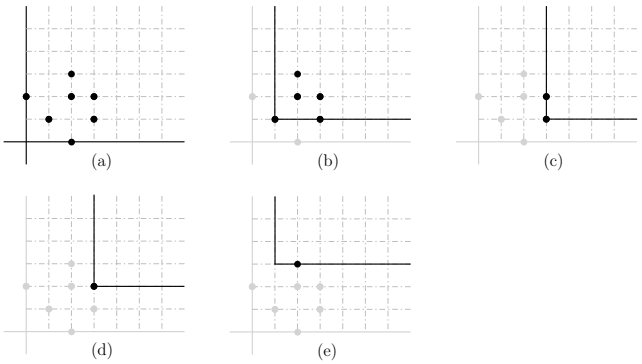


Fig. 3: The figure illustrates possible scenarios for PMI recurrence as used in the dynamic program with two leaders. Assume origin of these figures to be $(0, 0)$. In the case (a) there are separate points along both coordinates, so we have $A_{0,0} = \max(A_{0,1} + 1, A_{1,0} + 1)$. In the case (b) there are two points along x and one point along y, so we have $A_{1,1} = \max(A_{1,2} + 1, A_{2,1} + 1)$. Note that point $(1, 1)$ is minimum along both x and y coordinates. In the case (c) there are two points along y and one point along x, so $A_{3,1} = \max(A_{3,2} + 1, A_{4,1} + 1)$. In Figure (d), there is a point along x and a point along y (same point in this case), so we have $A_{3,2} = \max(A_{3,3} + 1, A_{4,2} + 1)$. In case (e) there is a point in x coordinate but no point along y, so $A_{1,3} = \max(A_{1,4}+1, A_{3,3}+0)$.

Algorithm 2 PMI - Dynamic Program

procedure PMI−DP ($\text{Ł}^1, \text{Ł}^2 \ldots, \text{Ł}^m$)
  z_i be number of unique values of i^{th} coordinate among all points.
  $z = \max(z_1, z_2 \ldots, z_m)$
  Define a m-dimensional array A with dimensions $(z + 1) \times (z + 1) \times \ldots (z + 1)$
  Let $A_{c_1,c_2,\ldots,c_m}$, i.e. value of A at index set $c_1, c_2, \ldots, c_m$ represents $\alpha^{[c_1,c_2,\ldots,c_m]}$ as in (8).
  for k from 1 to m **do**
    $A_{c_1,c_2,\ldots,c_m} \leftarrow 0$ for $c_k = z, c_{k'} \leq z, k' \neq k$.
  end for
  for j from $z - 1$ to 0 **do**
    for k from 1 to m **do**
      Compute $A_{c_1,c_2,\ldots,c_m}$ for $c_k = j, c_{k'} \leq j, k' \neq k$ using (8).
    end for
  end for
  return $A_{0,0,\ldots,0}$
end procedure

list, and use the recursive algorithm (PMI−R) with a check in the first line to not go through the subroutine if the current subproblem has already been solved and hashed once.

Remark 3.9: We note that an exact algorithm to compute longest PMI sequence in $O(m^n)$ was proposed in [1]. However dynamic programming solution in Algorithm 2 computes longest PMI sequence in a much lesser time, that is, $O(m \times (n \log n + n^m))$.

IV. Computing SSC Bound – A Greedy Approximation Algorithm

We observe that the dynamic programming algorithm described above runs well for graphs with several hundreds of nodes on a decent machine. However, when the number of nodes is much higher, or when the number of leaders is large, it becomes impractical. In the following, we propose a more efficient greedy approximation algorithm, which gives very close to optimal solutions in practice as illustrated numerically in Section V. Runtime complexity is linearithmic in the size of the input, thus it works well for computing quite accurate approximate PMI sequences for almost all practical networks.

The main idea behind the greedy algorithm is to make *locally* an optimal choice when faced with the situation in Lemma 3.4, that is, when including a point in PMI results in discarding a subset of points from possible future consideration. Locally best thing to do in this case is to pick a point that results in the loss of minimum number of other points. We outline the details in Algorithm 3.

Algorithm 3 PMI-Greedy Algorithm

procedure PMI−Greedy($\text{Ł}^1, \text{Ł}^2 \ldots, \text{Ł}^m$)
  $\mathcal{D} \leftarrow \emptyset$ ▷ Initially empty sequence
  while $\text{Ł}^1 \neq \emptyset$ **do**
    $X_i \leftarrow \{\text{Ł}^i_j : \text{Ł}^i_{j,i} = L^i_{1,i}\}$ for all i.
    if $\exists i$ such that $|X_i| = 1$ **then** ▷ Unique min.
      $\mathcal{D} \leftarrow [\mathcal{D}\ X_i]$
      Remove X_i from all lists.
    else
      Let $j \leftarrow \arg\min_i |X_i|$ ▷ Get smallest X_i
      $\mathcal{D} \leftarrow [\mathcal{D}\ \text{Ł}^j_1]$
      Remove all points in X_j from all lists.
    end if
  end while
  return $\mathcal{D}$
end procedure

Proposition 4.1: Algorithm 3 computes an approximate PMI sequence in time $O(mn \log n)$, and is an m-approximation, where m is the number of leaders and n is the total number of nodes in a graph. Further, Algorithm 3 also has an approximation ratio of $\log n$ if $m \leq \log n$, or $m \geq \frac{n}{\log n}$.

Proof: Available in [18].

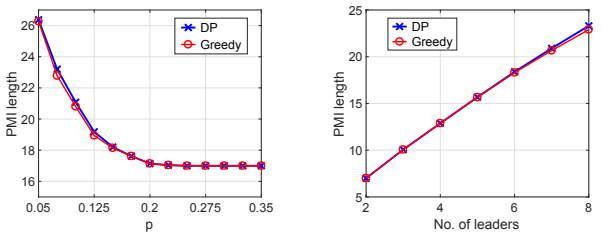
We also illustrate the greedy algorithm in [18]. If there exists a PMI sequence of length n, then the network is strong structurally controllable with a given set of leaders. The greedy algorithm presented above always returns a PMI sequence of length n if there exists one. We state this result in the following lemma, and provide a proof in [18].

Lemma 4.2: If there exists a PMI sequence of length n, then Algorithm 3 always returns an optimal PMI.

Authorized licensed use limited to: Qingdao University. Downloaded on October 29,2020 at 06:26:02 UTC from IEEE Xplore. Restrictions apply.

V. Numerical Evaluation

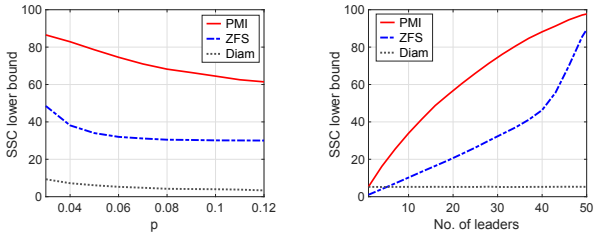
In this section, we numerically evaluate our results on Erdös-Rényi (ER) graphs in which any two nodes are adjacent with a probability p. First, we compare the performance of exact dynamic programming algorithm (Algorithm 2) and the approximate greedy algorithm (Algorithm 3) for computing the maximum length PMI sequences. In Figure 4(a), we plot lengths of PMI sequences computed using Algorithms 2 and 3 as a function of p while fixing the number of leaders, which are selected randomly. Second, we fix $p = 0.075$, and plot PMI length as a function of number of leaders in Figure 4(b). We mention that each point in plots in Figure 4 is an average of 50 randomly generated instances. From the plots, it is clear that the greedy algorithm, which is much faster as compared to the DP algorithm, performs almost as good as the dynamic programming algorithm.



(a) $n = 200$, $|V_\ell| = 8$ (b) $n = 200$, $p = 0.075$

Fig. 4: Comparison of dynamic programming and greedy algorithms for computing PMI sequences.

Next, we numerically compare PMI sequence based bound with another bound on the dimension of SSC that is based on the notion of Zero Forcing Sets (ZFS) [8], [10]. In Figure 5(a), we plot these bounds as function of p while fixing the number of leaders. In Figure 5(b), we fix p and plot SSC bound as a function of number of leaders. We observe that PMI-based bound significantly outperforms the ZFS-based bound in all the cases. Similar results are obtained in the case of Barabási-Albert (BA) graphs (illustrated in [18]).



(a) $n = 100$, $|V_\ell| = 30$ (b) $n = 100$, $p = 0.06$

Fig. 5: Comparison of PMI and ZFS-based bounds. Diameters of graphs are also plotted.

VI. Conclusion

We studied computational aspects of a lower bound on the dimension of SSC in networks with Laplacian dynamics. The bound is based on a sequence of distance-to-leader vectors, and has been used to explore trade-off between robustness and strong structural controllability [2]. However, no efficient algorithms to compute the bound were known. In this paper, we studied the problem in detail and provided first polynomial time algorithm to compute longest sequence of distance-to-leader vectors with a fixed set of leader nodes, which directly provided a bound on the dimension of SSC. We also presented a linearithmic approximation algorithm to compute the sequence, which provided near optimal solutions in practice. In the future, we hope to apply these results to further explore trade-offs between controllability and other desirable network properties including structural robustness.

References

[1] A. Yazıcıoğlu, W. Abbas, and M. Egerstedt, "Graph distances and controllability of networks," *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 4125–4130, 2016.

[2] W. Abbas, M. Shabbir, A. Y. Yazıcıoğlu, and A. Akber, "On the trade-off between controllability and robustness in networks of diffusively coupled agents." in *American Control Conference (ACC)*, 2019.

[3] H. Mayeda and T. Yamada, "Strong structural controllability," *SIAM Journal on Control and Optimization*, vol. 17, pp. 123–138, 1979.

[4] K. Reinschke, F. Svaricek, and H.-D. Wend, "On strong structural controllability of linear systems," in *31st IEEE Conference on Decision and Control (CDC)*, 1992, pp. 203–208.

[5] J. C. Jarczyk, F. Svaricek, and B. Alt, "Strong structural controllability of linear systems revisited," in *50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, 2011.

[6] A. Chapman and M. Mesbahi, "On strong structural controllability of networked systems: A constrained matching approach." in *American Control Conference (ACC)*, 2013, pp. 6126–6131.

[7] A. Weber, G. Reissig, and F. Svaricek, "A linear time algorithm to verify strong structural controllability," in *53rd IEEE Conference on Decision and Control (CDC)*, 2014, pp. 5574–5580.

[8] N. Monshizadeh, S. Zhang, and M. K. Camlibel, "Zero forcing sets and controllability of dynamical systems defined on graphs," *IEEE Transactions on Automatic Control*, vol. 59, pp. 2562–2567, 2014.

[9] M. Trefois and J.-C. Delvenne, "Zero forcing number, constrained matchings and strong structural controllability," *Linear Algebra and its Applications*, vol. 484, pp. 199–218, 2015.

[10] N. Monshizadeh, K. Camlibel, and H. Trentelman, "Strong targeted controllability of dynamical networks," in *54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 4782–4787.

[11] S. S. Mousavi and M. Haeri, "Controllability analysis of networks through their topologies," in *55th IEEE Conference on Decision and Control (CDC)*, 2016, pp. 4346–4351.

[12] S. S. Mousavi, M. Haeri, and M. Mesbahi, "On the structural and strong structural controllability of undirected networks," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 2234–2241, 2018.

[13] S. Zhang, M. Cao, and M. K. Camlibel, "Upper and lower bounds for controllable subspaces of networks of diffusively coupled agents," *IEEE Transactions on Automatic control*, vol. 59, pp. 745–750, 2014.

[14] V. Tzoumas, M. A. Rahimian, G. J. Pappas, and A. Jadbabaie, "Minimal actuator placement with bounds on control effort," *IEEE Transactions on Control of Network Systems*, vol. 3, pp. 67–78, 2016.

[15] A. Clark, B. Alomair, L. Bushnell, and R. Poovendran, "Submodularity in input node selection for networked linear systems: Efficient algorithms for performance and controllability," *IEEE Control Systems Magazine*, vol. 37, no. 6, pp. 52–74, 2017.

[16] C. O. Aguilar and B. Gharesifard, "Graph controllability classes for the laplacian leader-follower dynamics," *IEEE transactions on automatic control*, vol. 60, no. 6, pp. 1611–1623, 2015.

[17] S. S. Mousavi, M. Haeri, and M. Mesbahi, "Robust strong structural controllability of networks with respect to edge additions and deletions," in *American Control Conference (ACC)*, 2017.

[18] M. Shabbir, W. Abbas, and A. Y. Yazıcıoğlu, "On the computation of a lower bound on strong structural controllability in networks," 2019. [Online]. Available: https://arxiv.org/abs/1909.03565

Authorized licensed use limited to: Qingdao University. Downloaded on October 29,2020 at 06:26:02 UTC from IEEE Xplore. Restrictions apply.