On the Computation of a Lower Bound on Strong Structural Controllability in Networks

Mudassir Shabbir, Waseem Abbas, and A. Yasin Yazıcıoğlu

***Abstract*—Networks are strong structurally controllable if they can be maneuvered from any initial state to any final state independently of the coupling strengths between nodes. If a network is not strong structurally controllable with a given set of 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-leader vectors. We show analytically that the bound is sharper than the previously known bounds. We provide a polynomial time algorithm to compute a desired sequence of distance-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. Using these results, we also explore connections between graph-theoretic properties and length of longest such sequences in path and cycle graphs. Finally, we numerically evaluate our results 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. Several interesting problems emerge in the area of network controllability and have been addressed by researchers in recent years. For instance, how can we characterize and relate the underlying network structure, often modeled as graphs, to the network controllability [1], [2], [3]? What is the minimum number of inputs (leaders) needed to control networks, and to which nodes should we apply such inputs within the network [4], [5], [6], [7]? Based on the extent to which a network can be controlled, how can we quantify its control performance [8]? 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. A coupling strength between two agents indicate the weightage agents give to each others information while updating their own state. 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 [9] 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 [10]. The main idea is to obtain distances between leaders and other nodes, arrange them in vectors called distance-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-leader vectors in prac-

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).

tice and takes $O(mn \log n)$ time. If there exists a sequence of length n of distance-leader vectors, then the network is SSC, and our greedy algorithm always returns such a sequence if it exists.

3) We analytically show that the distance based bound on SSC is always at least as good as other known bounds, in particular the one based on zero forcing sets (ZFS). In fact, the distance-based bound is significantly better than the ZFS-based bound in practice.
4) We analyze sequences of distance-leader vectors in paths and cycles with arbitrary leaders, thus characterizing strong structural controllability in such graphs.
5) Finally, we numerically evaluate our results on various graphs including Erdös-Rényi (ER) and Barabási-Albert (BA) graphs.

A. Related Work

The notion of strong structural controllability was introduced in [11] and the first graph-theoretic condition for single input systems was presented. For multi-input systems, [12] provided a condition to check SSC in $\mathcal{O}(n^3)$ time, where n is the number of nodes. Authors in [13] refined previous 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 [3]. In [14], an algorithm with a run time 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 [15], [16], 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.

Whether a network is strong structurally controllable or not is a binary decision. The notion of strong structurally controllable subspace [17], 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 [17], a lower bound based on the derived set of input nodes (leaders) was presented, which was further studied in [18], [19]. With a single input node, the dimension of SSC can be at least the diameter of the underlying network graph [20]. In [9], a tight lower bound on the dimension of SSC was proposed that was based on the distances between leaders and remaining 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 [10]. In this paper, we show that the distance-based lower bound on the dimension of SSC is sharper than the other bounds. Further studies in this direction include enumerating and counting strong structurally controllable graphs for a given set of network parameters (leaders and nodes) [21], [22], leader selection to achieve desired structural controllability (e.g., [4], [23], [24], [5], [7], [6]), and network topology design for a desired control performance (e.g., [25], [26], [27], [28]).

The rest of the paper is organized as follows: Section II introduces notations and preliminary concepts used throughout the paper. Section III provides recursive and dynamic programming based exact algorithms to compute a distance-based lower bound on the dimension of SSC. Section IV presents and analyzes a greedy approximation algorithm to compute a lower bound on the dimension of SSC. Section V analytically compares the distance-based bound with the zero forcing set based bound. Section VI characterizes SSC of path and cycle graphs using distance-leader vectors. Section VII provides a numerical evaluation of our results, and Section VIII concludes the paper.

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.

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

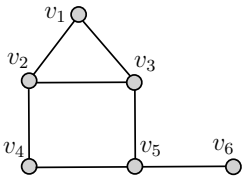
The matrix $B \in \mathbb{R}^{n \times m}$ in (1) is an input matrix. m is the number of leaders (inputs), which are the nodes to which an external control signal is applied. Let $V_\ell = \{\ell_1, \ell_2, \cdots, \ell_m\} \subset V$ be the set of *leaders*, then $B_{ij} = 1$ if node $i \in v$ is also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



$$\begin{bmatrix} \times & \times & \times & 0 & 0 & 0 \\ \times & \times & \times & \times & 0 & 0 \\ \times & \times & \times & 0 & \times & 0 \\ 0 & \times & 0 & \times & \times & 0 \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{bmatrix}$$

Fig. 1: A graph $G = (V, E)$ and its structured Laplacian matrix.

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 [9]. The bound is based on the distances between nodes in a graph. Assuming m leaders $V_\ell = \{\ell_1, \cdots, \ell_m\}$, we define a *distance-leader* vector for each $v_i \in V$ as below,

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

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-leader vectors, called as *pseudo-monotonically increasing* sequence as below.

Definition (*Pseudo-monotonically Increasing Sequence (PMI)* A sequence of distance-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

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

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-leader vectors is illustrated in Figure 2. 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

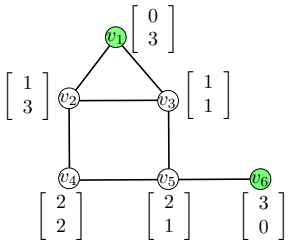


Fig. 2: A network with two leaders $V_\ell = \{\ell_1, \ell_2\} = \{v_1, v_6\}$, along with the distance-leader vectors of nodes. A PMI sequence of length five is $\mathcal{D} = [\mathcal{D}_1 \ \ \mathcal{D}_2 \ \ \cdots \ \ \mathcal{D}_5] = [D_6 \ D_5 \ D_1 \ D_4 \ D_2]$.

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

Theorem 2.1: [9] If δ is the length of longest PMI sequence of distance-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 longest 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 analytically and numerically.

In the next section, we provide an algorithm to compute a longest PMI sequence, 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:

Theorem 3.1: Given a graph G on n vertices, and m leaders, a longest PMI sequence of distance-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.

Note that the distance-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 a following simple fact:

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't satisfy the PMI property (stated 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 direction j and there is a longest PMI sequence which starts with D_i.

Definition *(Conflict-partition)* But it is possible that there is no unique minimum in any direction. This leads us to the definition of a *conflict* and *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.[1]

An example of conflict is illustrated in Figure 3.

It is easy to see that a PMI sequence can't 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 $\mathcal{X}$.

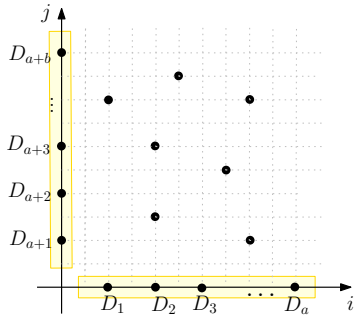


Fig. 3: Pointset $\mathcal{X} = \{D_1, D_2 \ldots, D_{a+b}\}$ constitute a conflict where $\mathcal{X}_i = \{D_1, D_2, \ldots, D_a\}$ and $\mathcal{X}_j = \{D_{a+1}, D_{a+2}, \ldots, D_{a+b}\}$.

Proof: Let $k_i = |\mathcal{X}_i|$ for all $1 \leq i \leq m$ for the partition defined in the statement, then for the sake of contradiction, let's assume that there is a sequence $\mathcal{D}'$ that contains more points from $\mathcal{X}$. Let $D_p \in \mathcal{X}_i$ be a point that appears first in $\mathcal{D}'$. If D_p satisfies PMI property on i^{th} coordinate then the remaining $k_i - 1$ points with the same minimum i^{th} coordinate in $\mathcal{X}_i$ can't be included in $\mathcal{D}'$. So D_p must satisfy PMI property on some j^{th} coordinate for following points in the sequence, where $j \neq i$. But then $\mathcal{D}'$ must miss at least k_j points which have smaller or equal j coordinate by definition of conflict, which is a contradiction. Thus claim follows. ∎

As an example, consider points in Figure 3. There are a points with the minimum i^{th} coordinate and b points with the minimum j^{th} coordinate. If D_1 is picked as first point (among this set) we must either drop all $D_2, D_3, \ldots, D_a$ or all $D_{a+1}, D_{a+2}, \ldots, D_{a+b}$. Similarly if D_{a+1} is picked before everyone else, we can't pick any of $D_{a+2}, \ldots, D_{a+b}$ or any of $D_2, D_3, \ldots, D_a$. In either choice we must lose at least $\min(a-1, b-1)$ points for future consideration regardless of positions of 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 $Ł^i_j$ [2]. 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't include all of these points to PMI so we need to make a decision to include some of these points and exclude other points. We will recursively consider all possibilities and pick the one that results in longest PMI sequence. Details are outlined in Algorithm 1. We state the following proposition:

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

Proof: Correctness of the algorithm follows from Observation 3.3 and from the fact that we try all possibilities. As far as time complexity is concerned, at each recursive call we may need to explore m possibilities and we can get rid of at least two points. This implies that the runtime follows recurrence:

$$T(n) \leq m \times T(n-2)$$

Also at least m points must be unique minima, one along each coordinate, so we have an upper bound of $O(m^{(n-m)/2})$ on the runtime of Algorithm 1. ∎

[1] 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.

[2] 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

procedure PMI−R($\text{Ł}^1, \text{Ł}^2 \ldots, \text{Ł}^m$) ▷ Recursive routine
  if $|\text{Ł}^1| <= 1$ **then**
    return Ł^1 ▷ one point is always a PMI
  end if
  $X_i \leftarrow \{\text{Ł}^i_j : \text{Ł}^i_{j,i} = L^i_{1,i}\}$ for all i.
  if $\exists i$ such that $|X_i| = 1$ **then** ▷ Check for unique min.
    return $[\text{Ł}^i_1$ PMI−R($\text{Ł}^1 \backslash X_i, \text{Ł}^2 \backslash X_i, \ldots, \text{Ł}^m \backslash X_i)]$.
  else
    for $i \in 1 : m$ **do** ▷ Check point in each direction
      $D^i \leftarrow [\text{Ł}^i_1$ PMI−R($\text{Ł}^1 \setminus X_i, \ldots, \text{Ł}^m \setminus X_i)]$
    end for
    return largest D^i
  end if
end procedure

An example run of the algorithm on the graph in Figure 1 with $V_\ell = \{v_1, v_6\}$ is illustrated in Appendix A. Algorithm 1 takes prohibitively exponential time even for the case of two leaders. In the following, we design an algorithm that is based on dynamic programming approach and will yield an optimal solution in polynomial runtime 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-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 $\alpha^{[c_1,c_2,\ldots,c_m]}$.

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

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

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

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.

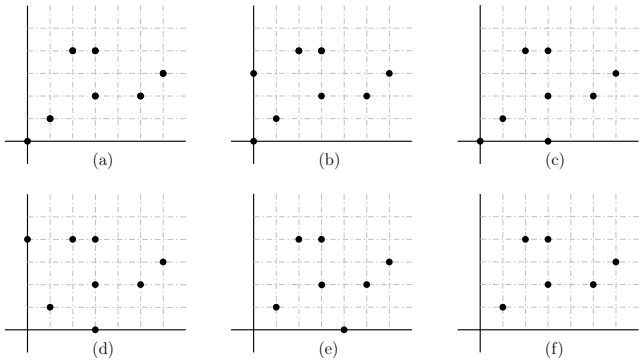


Fig. 4: The figure illustrates possible scenarios for PMI recurrence as used in the dynamic program with two leaders. Assume origin of these figures to be $(1, 1)$. In the cases (a), (b) and (c), there is a point on the origin which means $\mathbf{1}_{c_i} = 1$ and $A_{1,1} = \max(A_{1,2} + 1, A_{2,1} + 1)$. In the case (d) there are separate points along both coordinates, so we have $A_{1,1} = \max(A_{1,2} + 1, A_{2,1} + 1)$ again. In the case (e) there is a point along y coordinate but no point along x. So we have, $A_{1,1} = \max(A_{1,2} + 1, A_{2,1} + 0)$. And in the last case there are no points x or y, so $A_{1,1} = \max(A_{1,2} + 0, A_{2,1} + 0)$. Terms $A_{1,2}$ and $A_{2,1}$ are already pre-computed.

The details of dynamic program are in Algorithm 2. For some intuition on the working of dynamic program, we refer the reader to Figure 4.

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 $c_k = z, c_{k'} \le z, k' \ne k$.
  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 $c_k = j, c_{k'} \le j, k' \ne k$ using (8).
    end for
  end for
  return $A_{0,0,\ldots,0}$
end procedure

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

Proof: Correctness of Algorithm 2 follows from the correctness of Algorithm 1 and Observation 3.6. Computing sorted lists $\text{Ł}^1, \text{Ł}^2, \ldots, \text{Ł}^m$ takes $O(m \times n \log n)$ time. Each value of $A_{c_1,c_2,\ldots,c_m}$ can be computed by taking maximum of m known values in the table. The value of $\mathbf{1}_{c_i}$ can be computed

in constant time by looking up relevant $Ł^i$ as defined in (9). The table contains n^m such values. Therefore running time of this algorithm is bounded by $O(m \times (n \log n + n^m))$. ∎

Proof of Theorem 3.1: Theorem 3.1 follows directly from Proposition 3.7. ∎

Remark 3.8: For the sake of 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 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 [9]. However dynamic programming solution in Algorithm 2 computes longest PMI in much less time, that is, $O(m \times (n \log n + n^m))$.

We illustrate the algorithm in detail through an example in Appendix.

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 VII. 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 greedy algorithm is to make *locally* optimal choices 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. Details are provided in Algorithm 3.

Algorithm 3 PMI-Greedy Algorithm

```
procedure PMI−Greedy(Ł^1, Ł^2 ..., Ł^m)
    D ← ∅                                  ▷ Initially empty sequence
    while Ł^1 ≠ ∅ do
        X_i ← {Ł^i_j : Ł^i_{j,i} = L^i_{1,i}} for all i.
        if ∃i such that |X_i| = 1 then      ▷ Unique min.
            D ← [D  X_i]
            Remove X_i from all lists.
        else
            Let j ← arg min_i |X_i|         ▷ Get smallest X_i
            D ← [D  Ł^j_1]
            Remove all points in X_j from all lists.
        end if
    end while
    return D
end procedure
```

Proposition 4.1: Algorithm 3 computes an approximate PMI sequence in time $O(mn \log n)$. The algorithm is an m-approximation. Further, Algorithm also has an approximation ratio of $\log n$ if $m \leq \log n$ or $m \geq \frac{n}{\log n}$.

Proof: Regarding time complexity, computing sorted lists $Ł^1, Ł^2, \ldots, Ł^m$ takes $O(m \times n \log n)$ time. Once we have m sorted lists, we can keep the indices and count of points with minimum coordinate value in $(m+1)$ Min-Priority queues (m queues to maintain lists $Ł^1, Ł^2, \ldots, Ł^m$ and one queue for $X_1, X_2 \ldots, X_m$). Cost of one *update*, or *delete* operation is $O(\log n)$ in a Priority queue. Since we will *update* and/or *delete* at most n points from m queues. In total we will perform at most $m \times n$ deletions and $m \times n$ updates. Overall time complexity is $O(m \times n \log n)$.

Regarding m-approximation ratio, we observe that there are at least $\frac{n}{m}$ different values in at least one coordinate. Otherwise, we may assume that we have at most $\frac{n}{m} - 1$ unique values in each coordinate. This would imply that are at most $(\frac{n}{m} - 1) \times m$ distinct points by the pigeonhole principle; which contradicts that we have n unique points. As Algorithm 3 picks all distinct values in any coordinate, returned PMI sequence has size at least $\frac{n}{m}$.

It is obvious to see that when m is at most $\log n$ there are at least $\frac{n}{\log n}$ different values in at least one coordinate by the same argument. To see why $\log n$-approximation ratio holds when m is large, note that there is at least one unique minimum point (corresponding to the leader itself) in each of m directions so when $m \geq \frac{n}{\log n}$, the algorithm will clearly include all of those unique points in the PMI sequence returned. Thus, approximation ratios follows. ∎

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.

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

Proof: We observe that if there exists a PMI sequence of length n, then by Lemma 3.4 we can't have a conflict as defined in Section III. In the absence of any conflict, we can always find a unique minimum point along some coordinate. Consequently, in Algorithm 3, statements in **else** will never be executed and algorithm will return a PMI sequence of length n.

We illustrate the greedy algorithm in Appendix. ∎

Remark 4.3: While in great many cases Algorithm 3 achieves a solution close to optimum, we observe that examples can be engineered for which this may not be a globally optimal choice though. In the example outlined in Figure 5, we have two leaders and points are placed at $S = \{(2,2),(2,3),(3,3),(3,4),(4,4)\ldots,(k+1,k+1),(k+1,k+2),(k+2,k+2)\}$ and at $T = \{(1,2),(1,3),(1,4),\ldots,(1,k+2)\}$. An optimal PMI has all $2k$ points while Algorithm 3 above may only pick $k+3$ points.

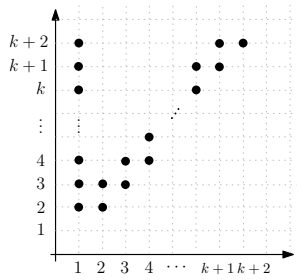


Fig. 5: Example discussed in Remark 4.3.

V. Comparison of PMI-based and Zero Forcing Set (ZFS)-based Bounds

Method of Zero Forcing Set (ZFS), described below, provides a well-known bound on the dimension of SSC. Here we compare ZFS and PMI-based bounds, and analytically show that the PMI-based bound is always at least as good as the ZFS-based bound. In fact, PMI-based bound significantly outperforms the other in practice. Thus, for the networks considered in this paper, the PMI-based bound is the best known bound on the dimension of SSC to the best of our knowledge.

First, we define the notion of Zero Forcing Set (ZFS). Given a graph $G = (V, E)$ where each node is colored either *white* or *black*, we repeatedly apply the following coloring rule: *If $v \in V$ is colored black and has exactly one white neighbor u, then color of u is changed to black.*

Definition *(Derived Set)* Given an initial set of black nodes (called *input set*) in G, a derived set $V' \subseteq V$ is set of all black nodes obtained after repeated application of the coloring rule until no color changes are possible.

It is easy to see that for a given input set, the resulting derived set is unique. Input set is called a ZFS if corresponding derived set contains all nodes in V. These notions are illustrated in Figure 6.

Theorem 5.1: [15], [17] Given a set of leader nodes as input set, the size of the corresponding derived set is a lower bound on the dimension of SSC.

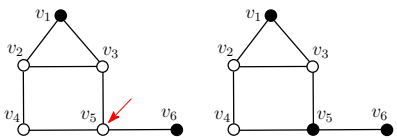


Fig. 6: (a) $\{v_1, v_6\}$ is the set of input (black) nodes. (b) Since v_5 is the only white neighbor of input node v_6, the color of v_5 is changed to black by the coloring rule. $\{v_1, v_5, v_6\}$ is the derived set of the input nodes. Note that the given set of input nodes is not a ZFS as the derived set does not contain all nodes in the graph.

Next, we show for a given set of leaders V_ℓ, length of the longest PMI is always greater than or equal to the size of the corresponding derived set. In the following, we are interested in the derived set $V' = \{v_1, v_2, \ldots, v_k\}$ as well as the order in which white nodes are included in V' as described in the Algorithm 4

Algorithm 4 ZFS-Algorithm

```
procedure ZFS(G, V_ℓ)
    m ← |V_ℓ|                          ▷ V_ℓ is input set (leaders)
    V' ← V_ℓ                           ▷ V' is the derived set
    π(v) ← v for all v ∈ V'
    for i ← m + 1 to n do
        if ∃v ∈ V' s.t. v has unique white neighbor u[3]. then
            Change the color of u to black.
            V' ← [V' u]                ▷ insertion order is important
            π(u) ← π(v)
        end if
    end for
    return V'
end procedure
```

Lemma 5.2: For each node $v_i \in V' = [v_1, v_2, \ldots, v_k]$ obtained from Algorithm 4, there exists an input node $\ell_j \in V_\ell$ such that v_i is the only node at distance $d(v_i, \ell_j)$ from ℓ_j. Moreover for all indices $\hat{i} > i$, $d(v_{\hat{i}}, \ell_j) > d(v_i, \ell_j)$ holds.

Proof: For each node v_i in V', we associate with v_i a parent $\pi(v_i) \in V_\ell$ in the input set as described in Algorithm 4. We prove Lemma 5.2 by induction on $d(v_i, \pi(v_i))$.

When $d(v_i, \pi(v_i)) = 0$ then trivially there is an input node ($\pi(v_i)$ itself) such that v_i is the only node with distance 0 from it and all other nodes are farther away. Now we show that claim is true for a node v_i at distance $s > 0$ with the associated input node $\pi(v_i)$ assuming it's true for all nodes at distance $< s$; v_i is added to the derived set so it must be the only *white* node in neighborhood of some $v_{i'}, i' < i$. By inductive hypothesis, $v_{i'}$ is the only node at distance $s - 1$ from $\pi(v_{i'}) = \pi(v_i)$. This implies that v_i is the only node at distance s from $\pi(v_i)$. As nodes following $v_{i'}$ are at distance more than $s-1$ from $\pi(v_{i'})$, and v_i is the only node at distance s, all nodes following v_i are at distance more than s from $\pi(v_i)$. This completes the proof. ∎

Theorem 5.3: For a given set of leaders (input set) V_ℓ, size of the derived set is a lower bound on the length of corresponding PMI sequence.

Proof: Let $V' = [v_1, v_2, \ldots, v_k]$ be the derived set as obtained from the Algorithm 4. By Lemma 5.2 for each v_i there is an input node ℓ_j such that $d(v_i, \ell_j) < d(v_{\hat{i}}, \ell_j)$, for all $\hat{i} > i$. If D_i is the distance leader vector corresponding to node v_i then the previous statement implies that the sequence $D_1, D_2, \ldots, D_k$ is a PMI sequence. Thus, for each derived set there exists a PMI sequence of equivalent length. The claim follows. ∎

In fact, we can show that length of PMI obtained from greedy Algorithm 3 is also greater than or equal to the size of the derived set.

Lemma 5.4: For a given set of leaders V_ℓ, size of the derived set is a lower bound on the length of PMI sequence returned by greedy Algorithm 3.

[3] If there are multiple such v, then we arbitrarily pick one of them.

Proof: Lemma 5.2 implies that for each v_i in the derived set, there is at least one input node ℓ_j such that v_i is the only node at distance $d(v_i, \ell_j)$. It is obvious by looking at greedy algorithm that all nodes with a unique minimum along any coordinate are included in the returned PMI sequence. Since v_i is the only node at distance $d(v_i, \ell_j)$ from ℓ_j it must be a unique minimum along the coordinate corresponding to ℓ_j at some step. Hence PMI returned by greedy contains all nodes of a derived set. ∎

VI. Characterization in Paths and Cycles

In this section, we explore connections between graph-theoretic properties and length of longest PMI in path and cycle graphs. As a result, we show interesting topological bounds on the dimension of SSC in such graphs with a given set of leader nodes. We note that our results differ from previous works in this direction in two aspects: first, we specifically study the strong structural controllability of such graphs; second, instead of focusing on complete controllability, we provide tight bounds on the dimesnion of SSC even when the graph is not strong structurally controllable with a given set of leaders [29], [30], [31].

We want to show that longest PMI sequence in a path and cycle graph can be characterized by the distances between leader nodes. A node with a single neighbor is called a *leaf*. We start with following obvious fact.

Fact 6.1: A path graph in which a leaf is a leader has a PMI sequence of length n.

For a graph $G = (V, E)$, let G^{-S} denote the subgraph of G induced on vertices $V \setminus S$ where $S \subseteq V$. Then we observe,

Theorem 6.2: Let G be a path graph on n nodes and let V_ℓ be a set of $m \leq n$ leader nodes. Then the following holds:

(i) If the number of connected components in G^{-V_ℓ} is less than $m+1$, then PMI sequence induced by V_ℓ has length n.

(ii) If the number of connected components in G^{-V_ℓ} is exactly $m+1$, then V_ℓ induces a PMI sequence of length $n-a$ where a is the size of smallest connected component in G^{-V_ℓ}.

Proof: (i). Removal of a node from a path results in at most two connected components hence G^{-V_ℓ} has at most $m+1$ such components. If the number of components is less than $m+1$ then either one of the leader nodes is a leaf or at least two leaders nodes are adjacent. If a leaf x is chosen as a leader then by Fact 6.1, we can get a PMI sequence of length n. Assuming none of the leaders is a leaf node, let v_i and v_{i+1} be adjacent leader nodes; further assume that $i < n/2$ without loss of generality. We will construct a PMI sequence of length n based on these two leaders as follows:

$$\left[\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right], \left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix} \right], \left[\begin{smallmatrix} 1 \\ 2 \end{smallmatrix} \right], \left[\begin{smallmatrix} 2 \\ 1 \end{smallmatrix} \right], \cdots, \left[\begin{smallmatrix} i-1 \\ i \end{smallmatrix} \right], \left[\begin{smallmatrix} i \\ i-1 \end{smallmatrix} \right], \left[\begin{smallmatrix} i+1 \\ i \end{smallmatrix} \right], \left[\begin{smallmatrix} i+2 \\ i+1 \end{smallmatrix} \right], \cdots, \left[\begin{smallmatrix} n-i \\ n-i-1 \end{smallmatrix} \right] \right]$$

(ii). If a smallest connected component X contains either of the leaf nodes then G^{-X} has a leaf leader node and thus has a PMI sequence of length $n - |X|$ by Fact 6.1. If X doesn't contain leaf nodes then there exist two leader nodes v_i, v_j are adjacent to some nodes in X. Also assume that v_i is not farther away from a leaf node than v_j is. Then the following sequence of distance-leader vectors defines a PMI sequence of claimed length:

$$\left[\left[\begin{smallmatrix} 0 \\ a+1 \end{smallmatrix} \right], \left[\begin{smallmatrix} a+1 \\ 0 \end{smallmatrix} \right], \left[\begin{smallmatrix} 1 \\ a+2 \end{smallmatrix} \right], \left[\begin{smallmatrix} a+2 \\ 1 \end{smallmatrix} \right], \cdots, \left[\begin{smallmatrix} i-1 \\ a+i \end{smallmatrix} \right], \left[\begin{smallmatrix} a+i \\ i-1 \end{smallmatrix} \right], \left[\begin{smallmatrix} a+i+1 \\ i \end{smallmatrix} \right], \left[\begin{smallmatrix} a+i+2 \\ i+1 \end{smallmatrix} \right], \cdots, \left[\begin{smallmatrix} n-i \\ n-i-a-1 \end{smallmatrix} \right] \right]$$

∎

Similarly for cycle graphs, we show the following:

Theorem 6.3: Let G be a cycle on n nodes and let V_ℓ be a set of $2 \leq m \leq n$ leader nodes.

(i) If the number of connected components in G^{-V_ℓ} is less than m, then PMI sequence induced by V_ℓ has length n.

(ii) If the number of connected components in G^{-V_ℓ} is exactly m, then V_ℓ induces a PMI sequence of length $n-a$ where a is the size of smallest connected component in G^{-V_ℓ}.

Proof: (i). Removing a single node from a cycle doesn't affect the number of connected components. However, removal of every subsequent node will result in at most one extra component. Thus total number of connected components is at most m after removal of m nodes. If the number of components is less than m in G^{-V_ℓ}, then at least two nodes in V_ℓ are neighbors in G. Let v_1 and v_2 be an arbitrary adjacent pair in V_ℓ, without loss of generality. We will construct a PMI sequence of length n based on these two leaders. Consider the nodes in G with following distance-leader vectors

$$\left[\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right], \left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix} \right], \left[\begin{smallmatrix} 1 \\ 2 \end{smallmatrix} \right], \left[\begin{smallmatrix} 2 \\ 1 \end{smallmatrix} \right], \cdots, \left[\begin{smallmatrix} \frac{n}{2}-1 \\ \frac{n}{2} \end{smallmatrix} \right], \left[\begin{smallmatrix} \frac{n}{2} \\ \frac{n}{2}-1 \end{smallmatrix} \right] \right]$$

when n is even and

$$\left[\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right], \left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix} \right], \left[\begin{smallmatrix} 1 \\ 2 \end{smallmatrix} \right], \left[\begin{smallmatrix} 2 \\ 1 \end{smallmatrix} \right], \cdots, \left[\begin{smallmatrix} \lfloor \frac{n}{2} \rfloor \\ \lfloor \frac{n}{2} \rfloor \end{smallmatrix} \right] \right]$$

when n is odd. This defines a PMI sequence of length n.

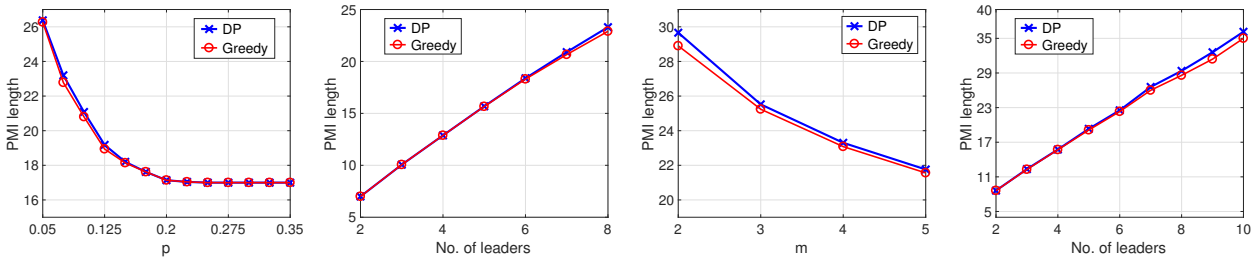
(ii). An argument identical to proof of Theorem 6.2(ii) can be used here to prove (ii) as well. ∎

Theorems 6.2 and 6.3 imply simple graph theoretic bounds on the dimension of SSC in the case of path and cycle graphs. A path (cycle) graph is strong structurally controllable with V_ℓ leaders if G^{-V_ℓ} has $m+1$ components (m components in the case of cycle). An other direct implication of the above results is the following corollary.

Corollary 6.4: Let G be a path or cycle graph and let V_ℓ be a set of leaders, then the dimension of SSC is at least $n-a$, where a is the smallest distance between any two leader nodes.

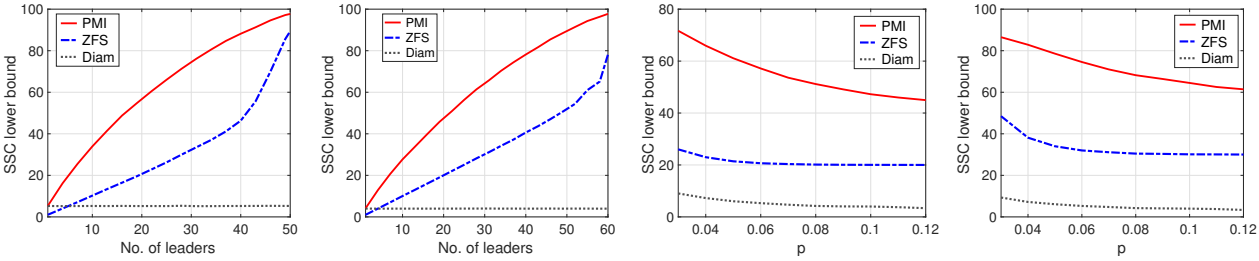
VII. Numerical Evaluation

In this section, we numerically evaluate our results on Erdös-Rényi (ER) and Barabási-Albert (BA) graphs. In ER graphs, any two nodes are adjacent with a probability p. BA graphs are obtained by adding nodes to an existing graph one at a time. Each new node is connected to m existing nodes with the probability proportional to the degree of the nodes.



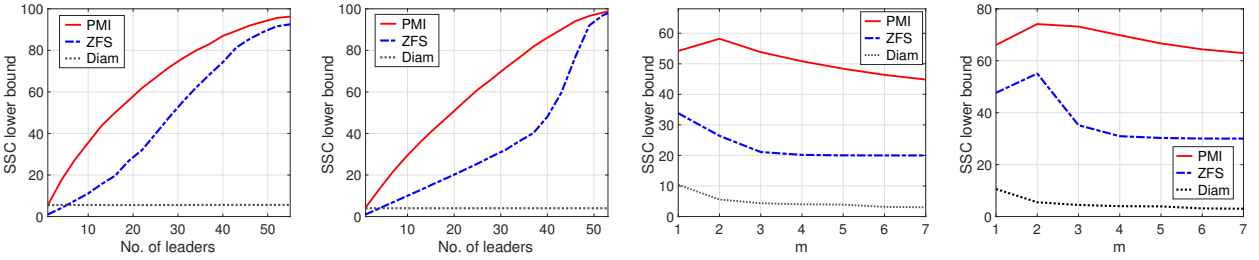
(a) ER (b) ER (c) BA (d) BA

Fig. 7: Comparison of dynamic programming and greedy algorithms for computing PMI sequences.



(a) $p = 0.06$ (b) $p = 0.1$ (c) $|V_\ell| = 20$ (d) $|V_\ell| = 30$

Fig. 8: Comparison of PMI and ZFS-based bounds in ER graphs. Diameters of graphs are also plotted.



(a) $m = 2$ (b) $m = 4$ (c) $|V_\ell| = 20$ (d) $|V_\ell| = 30$

Fig. 9: Comparison of PMI and ZFS-based bounds in BA graphs. Diameters of graphs are also plotted.

A. Dynamic Programming and Greedy Algorithms for Computing PMI Sequences

First, we compare the performance of exact dynamic programming algorithm (Algorithm 2) and the approximate greedy algorithm (Algorithm 3) for computing the longest length PMI sequences. For simulations, we consider graphs with $n = 200$ nodes. For ER graphs, we first plot the length of PMI sequences computed using Algorithms 2 and 3 as a function of p while fixing the number of leaders (selected randomly) to be eight (Figure 7(a)). Second, we fix $p = 0.075$, and plot PMI length as a function of the number of leaders selected randomly (Figure 7(b)). We repeat similar plots for BA graphs in Figures 7(c) and (d). We fix the number of leaders to be eight in Figure 7(c), whereas fix $m = 2$ in Figure 7(d). We mention that each point in plots in Figure 7 is an average of 50 randomly generated instances. From the plots, it is clear that greedy algorithm, which is much faster as compared to the DP algorithm, performs almost as good as the dynamic programming algorithm. The lentgh of PMI sequences returned by greedy algorithm is very close to the length of optimal PMI sequences, and hence, we get good lower bounds on the dimension of SSC with given sets of leader nodes.

B. Comparison of PMI and ZFS-based Bounds

Next, we numerically compare PMI and ZFS-based bounds on the dimension of SSC as discussed in Section V. For both ER and BA models, we consider graphs with $n = 100$ nodes. In Figures 8(a) and (b), we plot these bounds For ER graphs as a function of number of leaders, which are selected randomly, while fixing $p = 0.06$ and $p = 0.1$ respectively. Next, we fix the number of leaders, 20 in Figure 8(c) and 30 in Figure 8(d), and plot SSC bound as a function of p. As previously, each point in the plots is an average of 50 randomly generated instances. It is obvious that PMI-based bound significantly outperforms the ZFS-based bound in all the cases as also indicated by the analysis in Section V. Similar results are obtained in the case of BA graphs and are shown in Figure 9. We fix $m = 2$ and $m = 3$ in Figures 9(a) and (b) respectively. Further, the number of leaders selected in Figures 9(c) and (d)

are 20 and 30 respectively. In all the plots, for a given set of leaders, lengths of PMI sequences are always greater than the derived sets, thus, PMI-based bound on the dimension of SSC is better than the one based on the derived sets.

VIII. 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-leader vectors, and is sharper than the other known bounds as we have shown in the paper. The distance based bound has also been used to explore trade-off between robustness and strong structural controllability [10]. 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-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. Using our results, we explored connections between graph-theoretic properties and length of longest such sequences in path and cycle graphs. We hope to use these results to further explore trade-offs between controllability and other desirable network properties including its structural robustness. We also believe that the study of finding longest distance-leader vector sequences is an interesting direction in its own respect as it naturally generalizes Erdös-Szekeres type sequences to higher dimensions [32], [33], [34].

References

[1] A. Rahmani, M. Ji, M. Mesbahi, and M. Egerstedt, "Controllability of multi-agent systems from a graph-theoretic perspective," *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 162–186, 2009.

[2] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási, "Controllability of complex networks," *nature*, vol. 473, no. 7346, p. 167, 2011.

[3] 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.

[4] A. Olshevsky, "Minimum input selection for structural controllability," in *American Control Conference (ACC), 2015*, 2015, pp. 2218–2223.

[5] 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.

[6] 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.

[7] C. Commault and J.-M. Dion, "Input addition and leader selection for the controllability of graph-based systems," *Automatica*, vol. 49, no. 11, pp. 3322–3328, 2013.

[8] F. Pasqualetti, S. Zampieri, and F. Bullo, "Controllability metrics, limitations and algorithms for complex networks," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 1, pp. 40–52, 2014.

[9] 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.

[10] 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.

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

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

[13] 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.

[14] 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)*. IEEE, 2014, pp. 5574–5580.

[15] 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.

[16] 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.

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

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

[19] 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.

[20] 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.

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

[22] T. Menara, G. Bianchin, M. Innocenti, and F. Pasqualetti, "On the number of strongly structurally controllable networks," in *American Control Conference (ACC), 2017*. IEEE, 2017, pp. 340–345.

[23] S. Pequito, S. Kar, and A. P. Aguiar, "On the complexity of the constrained input selection problem for structural linear systems," *Automatica*, vol. 62, pp. 193–199, 2015.

[24] K. Fitch and N. E. Leonard, "Optimal leader selection for controllability and robustness in multi-agent networks," in *2016 European Control Conference (ECC)*. IEEE, 2016, pp. 1550–1555.

[25] 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.

[26] S. Pequito, G. Ramos, S. Kar, A. P. Aguiar, and J. Ramos, "The robust minimal controllability problem," *Automatica*, vol. 82, 2017.

[27] 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.

[28] M. Xue and S. Roy, "Input-output properties of linearly-coupled dynamical systems: Interplay between local dynamics and network interactions," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 487–492.

[29] G. Parlangeli and G. Notarstefano, "On the reachability and observability of path and cycle graphs," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 743–748, 2012.

[30] B. She, S. Mehta, C. Ton, and Z. Kan, "Topological characterizations of leader-follower controllability on signed path and cycle networks," in *IEEE Conference on Decision and Control (CDC)*, 2018, pp. 6157–6162.

[31] X. Liu and Z. Ji, "Controllability of multiagent systems based on path and cycle graphs," *International Journal of Robust and Nonlinear Control*, vol. 28, no. 1, pp. 296–309, 2018.

[32] P. Erdös and G. Szekeres, "A combinatorial problem in geometry," *Compositio Mathematica*, vol. 2, pp. 463–470, 1935.

[33] W. Samotij and B. Sudakov, "On the number of monotone sequences," *Journal of Combinatorial Theory, Series B*, vol. 115, pp. 132–163, 2015.

[34] N. Linial and M. Simkin, "Monotone subsequences in high-dimensional permutations," *Combinatorics, Probability and Computing*, vol. 27, no. 1, pp. 69–83, 2018.

Appendix

A. Illustration of Recursive Algorithm (Algorithm 1)

Consider a graph in Figure 1, in which nodes v_1 and v_6 are leaders. A run of the Algorithm 1 to compute an optimal PMI sequence of distance-leader vectors is illustrated in Figure 10. Note that $Ł^i$ is a list of points (distance-leader vectors) that

are sorted in a non-decreasing order with respect to the i^{th} coordinate. In our example, such lists are given below.

$$Ł^1 = \left[\begin{bmatrix} 0 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 3 \\ 0 \end{bmatrix}\right],$$

$$Ł^2 = \left[\begin{bmatrix} 3 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \end{bmatrix}\right].$$

Step-wise details of the algorithm are shown in Figure 10.

B. Illustration of Dynamic Programming Algorithm (Algorithm 2)

For illustration, we consider the same graph as in Figure 1, where $V_\ell = \{v_1, v_6\}$. The example run is described in Figure 11. The values of j and k denote the loop variables in lines 9 and 10 of the Algorithm 2 respectively. In each iteration of j, one column and one row of the memoization variable A is updated as shown in matrices in Figure 11. In fact, the value of each cell in the matrix is computed from values in the neighboring cells to the right and below using (8). For instance, $A_{1,0}$ is computed from $A_{1,1}$ (neighboring cell on the right) and $A_{2,0}$ (neighboring cell below). The first entry of the matrix, that is $A_{0,0}$ returns the length of longest PMI sequence.

C. Illustration of Greedy Algorithm (Algorithm 3)

We illustrate it on the same graph as in Figure 1. The initial lists $Ł^1$ and $Ł^2$ are same as in Section A. The sets X_1 and X_2 (line 4 of the algorithm) are $\left\{\begin{bmatrix} 0 \\ 3 \end{bmatrix}\right\}$ and $\left\{\begin{bmatrix} 3 \\ 0 \end{bmatrix}\right\}$ respectively, as also shown in Figure 12(b). Since both sets contain a unique minimum, the algorithm arbitrarily includes one of these two points in the sequence, that is $\begin{bmatrix} 3 \\ 0 \end{bmatrix}$ in this case. In the next two steps, the points $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 3 \end{bmatrix}$ are included in the sequence. In the next step illustrated in Figure 12(e), the sets X_1 and X_2 are $\left\{\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}\right\}$ and $\left\{\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \end{bmatrix}\right\}$ respectively. Since there is no unique minimum, cardinalities of X_1 and X_2 are compared (line 9 in the Algorithm 3) and a point from a smaller set will be chosen. In this example, $\begin{bmatrix} 2 \\ 2 \end{bmatrix}$ is chosen. The sequence returned by the algorithm is as follows:

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

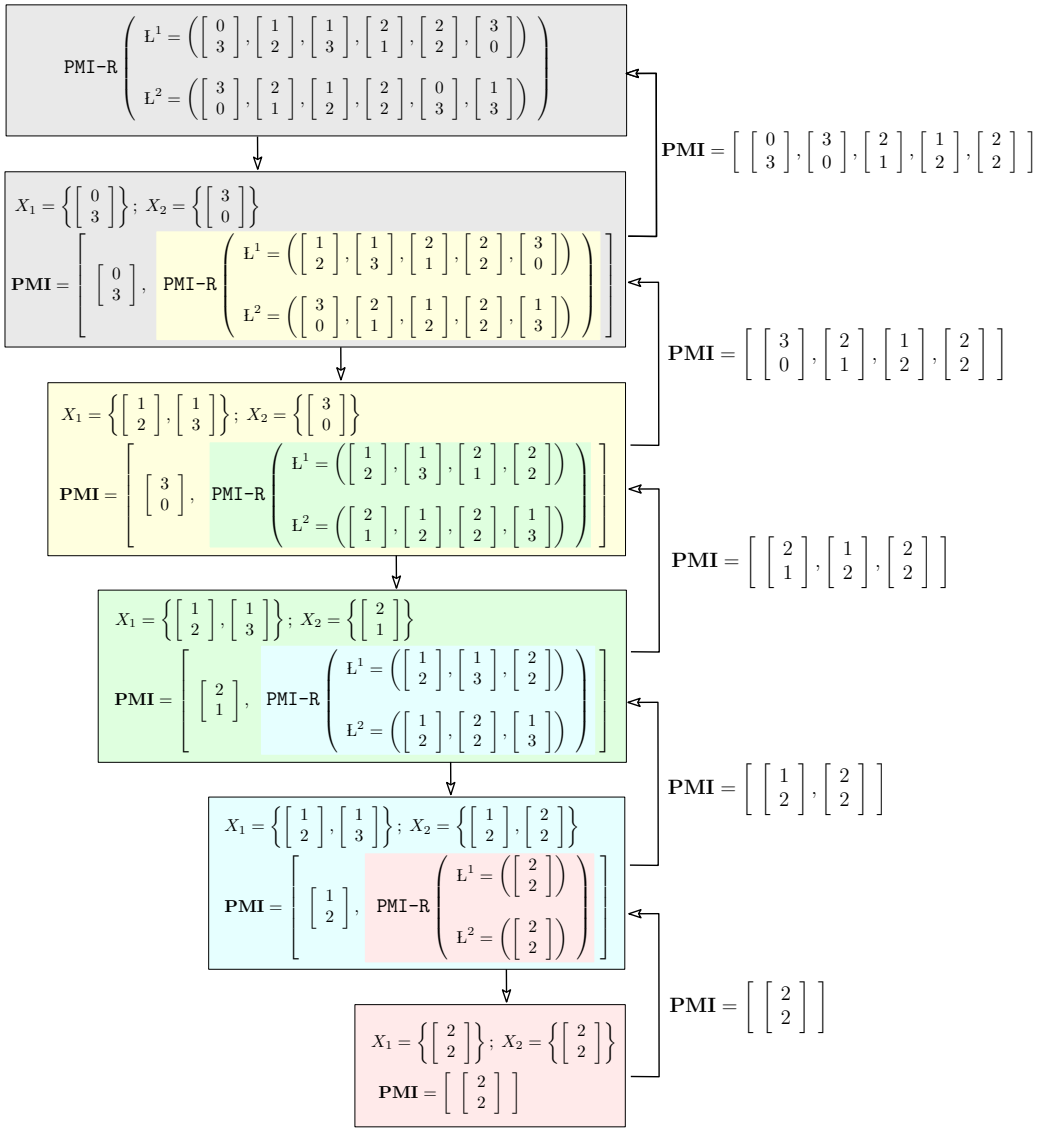


Fig. 10: Illustration of the run of Algorithm 1 for the graph in Figure 1 with v_1 and v_6 as leaders.

j	3		2		1		0	
k	1	2	1	2	1	2	1	2
	$c_1 = 3$ $c_2 \in \{0,1,2,3\}$ $A_{3,3} = 0$ $A_{3,2} = 0$ $A_{3,1} = 0$ $A_{3,0} = 1$	$c_2 = 3$ $c_1 \in \{0,1,2,3\}$ $A_{3,3} = 0$ $A_{2,3} = 0$ $A_{1,3} = 1$ $A_{0,3} = 2$	$c_1 = 2$ $c_2 \in \{0,1,2\}$ $A_{2,2} = 1$ $A_{2,1} = 2$ $A_{2,0} = 3$	$c_2 = 2$ $c_1 \in \{0,1,2\}$ $A_{2,2} = 1$ $A_{1,2} = 2$ $A_{0,2} = 3$	$c_1 = 1$ $c_2 \in \{0,1\}$ $A_{1,1} = 3$ $A_{1,0} = 4$	$c_2 = 1$ $c_1 \in \{0,1\}$ $A_{1,1} = 3$ $A_{0,1} = 4$	$c_1 = 0$ $c_2 = 0$ $A_{0,0} = 5$	$c_2 = 0$ $c_1 = 0$ $A_{0,0} = 5$

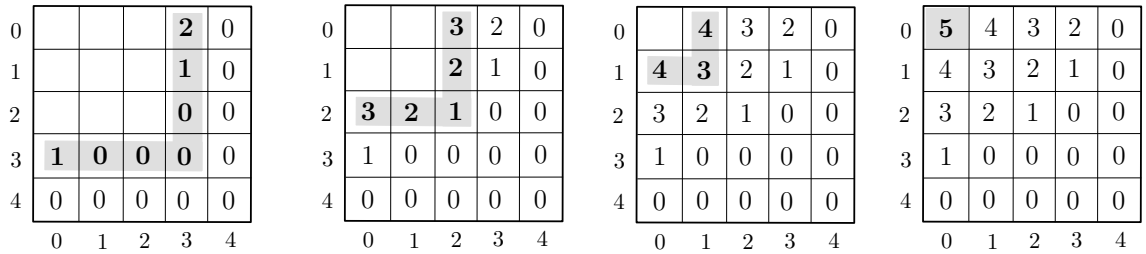
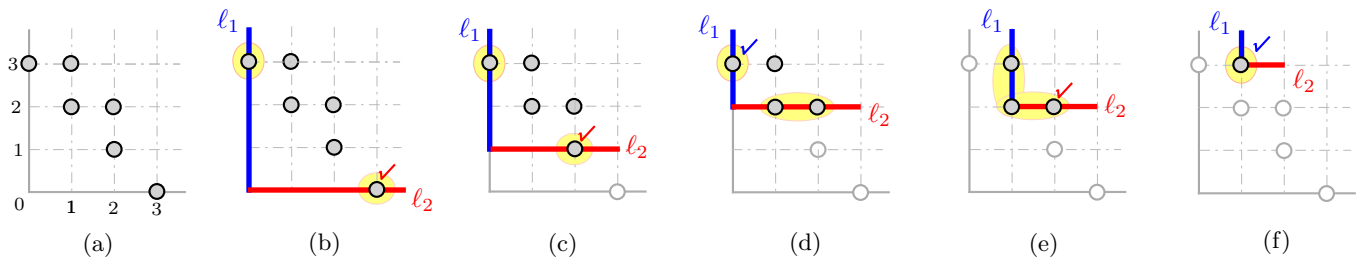


Fig. 11: Illustration of the run of Algorithm 2 for the graph in Figure 1 with leaders $V_\ell = \{v_1, v_6\}$.



(a) (b) (c) (d) (e) (f)

Fig. 12: Illustration of the greedy algorithm.