NETWORK CONTROLLABILITY PERSPECTIVES ON GRAPH REPRESENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph representations in fixed dimensional feature space are vital in applying learning tools and data mining algorithms to perform graph analytics. Such representations must encode the graph's topological and structural information at the local and global scales without posing significant computation overhead. This paper employs a unique approach grounded in networked control system theory to obtain expressive graph representations with desired properties. We consider graphs as networked dynamical systems and study their *controllability* properties to explore the underlying graph structure. The controllability of a networked dynamical system profoundly depends on the underlying network topology, and we exploit this relationship to design novel graph representations using controllability Gramian and related metrics. We discuss the merits of this new approach in terms of the desired properties (for instance, permutation and scale invariance) of the proposed representations. Our evaluation of various benchmark datasets in the graph classification framework demonstrates that the proposed representations either outperform (sometimes by more than 6%), or give similar results to the state-of-the-art embeddings.

1 INTRODUCTION

The graph-theoretic framework provides means to analyze network characteristics and examine the influence of local interactions on global network behavior. In recent years, various data-driven approaches have been developed to solve real-world graph problems like graph classification, link prediction, community detection, and network evolution. Applying prevalent data mining techniques and learning algorithms to solve graph problems is not a straightforward task. The classical methods are designed for vector-valued data requiring graphs to be embedded in vector spaces. In other words, we need to define vector representations of graphs with some desired properties, such as permutation-invariance, expressiveness, and accuracy.

In this paper, we design a novel graph representation grounded in the *network controllability* paradigm (Mesbahi & Egerstedt, 2010). We perceive graphs as *networked dynamical systems* in which each vertex is an *agent* (dynamical unit) that maintains a *state*. Every agent updates its state through some dynamical process and interacts with other agents in its neighborhood defined by the underlying network graph. The states of all agents define the overall network's state. The network controllability paradigm concerns steering a network from one state to another by injecting some *control signals* into the system through a subset of agents. The network's ability to be manipulated and controlled through such external inputs directly depends on the underlying network graph (Ahmadizadeh et al., 2017; Egerstedt et al., 2012; Liu et al., 2011; Pasqualetti et al., 2014; Rahmani et al., 2009). As a result, by studying the controllability properties of dynamical processes over networks, one can gather valuable insights into the underlying graph's structure that are distinct from other approaches. *We propose to understand and harness the relationship between graph topology and networked dynamical system behavior to design expressive graph representations in this work.*

The controllability of networked dynamical systems has been a fundamental topic in control theory. In recent years, many studies have established profound connections between network controllability and the underlying graph-theoretic constructs, such as matching (Liu et al., 2011), graph

distances (Yazıcıoğlu et al., 2016), dominating sets (Nacher & Akutsu, 2014), equitable partitions (Egerstedt et al., 2012), and zero forcing sets (Monshizadeh et al., 2014). At the same time, graph-theoretic characterization of controllability for various families of network graphs, such as paths, cycles, trees, complete graphs, random graphs, symmetric graphs, circulant graphs, bipartite graphs, and product graphs, have been reported (Mesbahi & Egerstedt, 2010).

This paper demonstrates that by exploring controllability properties of networks, including how 'much' of the overall network can be controlled from a given set of input nodes, how 'easy' it is to steer the network towards desired states, how the 'location' of input nodes affect the controllability, and how the network topology influences these behaviors, we can construct effective graph representations (CTRL and CTRL+). We evaluate the proposed representations for the classification problem on several standard datasets and report improved or competitive classification accuracy compared to the existing approaches. We also discuss the expressiveness and invariance to node orderings of the proposed graph embeddings. This network control systems perspective to design graph representations is studied for the first time to the best of our knowledge.

2 NETWORKS AS DYNAMICAL SYSTEMS

A network of inter-connected entities is represented by a graph $G = (V, E)$, where the vertex set $V = V(G) = \{v_1, v_2, \ldots, v_N\}$ represents the entities, and the edge set $E = E(G) \subseteq V \times V$ represents the pairs of related entities. We use the terms vertex, node and agent alternatively. The neighborhood of a vertex v_i is the set $\mathcal{N}_i = \{v_j \in V : (v_j, v_i) \in E\}$. The degree of v_i, denoted by δ_i, is the size of the neighborhood $\mathcal{N}_i$. A graph with N nodes is represented by the adjacency matrix $J \in \{0, 1\}^{N \times N}$, where $J_{i,j} = 1$, if and only if $(v_j, v_i) \in E$, and $J_{i,j} = 0$, otherwise. The degree matrix of G, denoted by D, is a diagonal matrix with $D_{i,i} = \delta_i$. The Laplacian matrix of a graph is defined as $L = D - J$. The transpose of a matrix X is denoted by X^T. An N-dimensional vector with all zero entries is denoted by $\mathbf{0}_N$, and a vector with all 1's is represented by $\mathbf{1}_N$. We consider undirected graphs here for the ease of exposition; however, all the methods and results are also applicable to directed graphs. We provide the details for directed graphs in Appendix B.

2.1 PROBLEM DESCRIPTION

A graph embedding is defined as a function $\phi(G) : \mathcal{G} \to \mathbb{R}^d$, from the family of graphs, $\mathcal{G}$, to a d-dimensional Euclidean space. The objective of the graph embedding problem is to find *suitable* embeddings for the graphs, where the suitability of embeddings is driven by a few design goals discussed here. Most importantly, ϕ should be able to retain information about the *structural similarities* between pairs of graphs at both local and global scales, i.e., if two graphs are structurally similar, then their embeddings should generate vectors that are nearby with respect to the Euclidean distance in the target vector space. Note that the concept of similarities between two graphs is not a universal notion but rather depends on a particular application (e.g., graph classification, nearest neighbor search, clustering) and the family of graphs considered (e.g., chemical compounds, social networks). Furthermore, ϕ should be *permutation-invariant* in the sense that ϕ should return identical vectors for two graphs, G, H, with the same set of edges on a permuted vertex set, i.e., $\exists \pi : V(G) \to V(H), (u, v) \in E(G) \Leftrightarrow (\pi(u), \pi(v)) \in E(H)$, then we should have $\phi(G) = \phi(H)$. Another important design goal for graph embeddings is scale-adaptiveness. Not only should a graph embedding be able to map graphs of varying sizes to a fixed dimensional space, but mapping should also transcend the graph size to capture its structural properties. For example, an ideal graph embedding would map a cycle on ten nodes closer to the mapping of a cycle on twenty nodes as compared to the mapping of a wheel on fifteen nodes. In this paper, we address the problem of finding graph embedding while keeping the above-mentioned design goals in perspective.

2.2 CONTROL DYNAMICS PROPERTIES OVER NETWORKS

We design distinctive graph representations by studying controlled dynamical processes over networks and mapping the control behavior to the network topology. Consider a network graph in which each agent v_i is a dynamical unit with a *state* $x_i(t) \in \mathbb{R}$ at time t that the agent also shares with its neighbors $\mathcal{N}_i$. Each agent updates its state by following some dynamics (e.g., consensus dynamics) while incorporating its neighbors' states during the state update process. The state of the overall

system at time t is a vector of the states of all the agents, i.e., $x(t) = [x_1(t) \;\; x_2(t) \;\; \cdots \;\; x_N(t)]^T$. Each agent updates its state by the *consensus* dynamics given by

$$\dot{x}_i(t) = \sum_{v_j \in \mathcal{N}_i} (x_j(t) - x_i(t)) \,. \tag{1}$$

The system level dynamics (evolution of the state $x(t)$) is then defined by the following linear system:

$$\dot{x}(t) = -Lx(t), \tag{2}$$

where L is the Laplacian matrix of the underlying network graph. It is well known that if G is connected, then state of each agent will eventually converge to the average of the initial states of all agents (Mesbahi & Egerstedt, 2010). Thus, if $x_i(0)$ is the initial state (at $t = 0$) of agent v_i, then

$$x_i(t) \;\rightarrow\; \bar{x} \triangleq \frac{1}{N} \sum_{v_j \in V} x_j(0), \;\; \text{as } t \to \infty, \tag{3}$$

$\forall v_i \in V$. It means the overall network state $x(t)$ will be $[\bar{x} \;\; \bar{x} \;\; \cdots \;\; \bar{x}]^T = \bar{x}\mathbf{1}_N \in \mathbb{R}^N, \;\text{as } t \to \infty$. Thus, under the consensus dynamics in equation 2, all agents converge to the same state. The linear system defined over G in equation 2 is autonomous as the system's state is updated without any external input. We have no control over the state's evolution in the sense that we cannot steer the system to some desired state, say $x^*(t_f) \in \mathbb{R}^N$ at time t_f. For this purpose, external control signals are injected into the system through a small subset of agents called *leaders*. Through these exogenous signals, leaders' states can be directly manipulated, i.e., $\dot{x}_l = u_l(t)$, where $u_l(t)$ is the input signal to the leader agent v_l. The non-leader agents, often called *followers*, continue to update their states using equation 1.

By feeding appropriate control signals to leaders, which are typically very few, the network's overall state $x(t) \in \mathbb{R}^N$ can be manipulated. As a result, we get certain control over the system's (state) evolution. The set of states that can be achieved, that is, to which the system can be driven, depends on the underlying network graph, the number of leaders, and their locations within the network. This ability of a network to be controlled through external inputs is called *network controllability*. By studying network controllability, we can gather valuable information about the network's structure as the two are deeply connected. By studying the control-related properties of the network, for instance, the number of leaders needed to completely control it, the dimension of the subspace consisting of controllable states, and the amount of control energy needed to steer the system from one state to another, one can thoroughly examine the graph structure and design effective graph representations.

3 Network Controllability and Graph Representation

In this section, we define network controllability and discuss various measures to quantify it. In the later sections, we use these measures to design graph representations.

3.1 Network Dynamics

For a network graph $G = (V, E)$, we partition V into follower and leader nodes, denoted by V_f and V_ℓ, respectively, i.e., $V = V_f \cup V_\ell$. Here, $|V_f| = N_f$ and $|V_\ell| = N_\ell$. Without loss of generality, we assume that $V_f = \{v_1, v_2, \cdots, v_{N_f}\}$ and $V_\ell = \{v_{N_f+1}, \cdots, v_N\}$. The subgraph induced by V_f is called the *follower graph* and is denoted by G_f. The Laplacian matrix of G is partitioned as

$$L = \left[\begin{array}{c|c} A & B \\ \hline B^T & C \end{array}\right], \tag{4}$$

where $A \in \mathbb{R}^{N_f \times N_f}$, $B \in \mathbb{R}^{N_f \times N_\ell}$ and $C \in \mathbb{R}^{N_\ell \times N_\ell}$. An external input signal u_l is given to leader agent $v_l \in V_\ell$. The follower nodes update their states according to equation 1. The state vector corresponding to follower nodes is denoted by $\boldsymbol{x_f}(t) \in \mathbb{R}^{N_f}$, and is updated by the following system.

$$\dot{\boldsymbol{x}}_f(t) = -A\boldsymbol{x_f}(t) - Bu(t), \tag{5}$$

where A and B are in equation 4 and $u(t) = [u_{N_f+1}(t) \;\; \cdots \;\; u_N(t)]^T \in \mathbb{R}^{N_\ell}$ is a control signal at time t. We note that the system matrices $-A$ and $-B$ in equation 5 directly depend on the underlying

network structure and the selection of leader agents. As a result, the evolution of $\boldsymbol{x_f}$ is a function of the network graph and the control mechanism, which includes external inputs and the selection of leader agents in the network. From the control perspective, we are interested in knowing if it is possible to steer the system equation 5 from an arbitrary initial state to an arbitrary final state in a finite amount of time t_1. If it is possible, then how much control energy $\mathcal{E}(u)$, defined below, would be required.

$$\mathcal{E}(u) = \int_{\tau=0}^{t_1} \|u(\tau)\|^2 d\tau. \tag{6}$$

Similarly, if all states are not reachable, then what is the dimension of the subspace consisting of reachable states? How these control properties vary as a result of a change in leader agents? Answers to these questions encode information that could be conducive to learn the graph structure. For this, we need metrics to quantify various aspects of network controllability. These measures can then be used to obtain graph embeddings. Controllability of linear systems is a fundamental topic in Control theory and we use results from there to quantify the controllability properties of networks.

3.2 Network Controllability Metrics

Controlling a network corresponds to driving a network from a given initial state to a desired final state by applying control inputs to leaders in the network. If $\boldsymbol{x_f}(t_i)$ is the initial state at time t_i, then under the dynamics equation 5, the state at time t_s is

$$\boldsymbol{x_f}(t_s) = e^{-A(t_s - t_i)}\boldsymbol{x_f}(t_i) \; + \; \int_{t_i}^{t_s} e^{-A(t_s-\tau)}(-B)u(\tau)d\tau. \tag{7}$$

A state $\boldsymbol{x_f^*} \in \mathbb{R}^{N_f}$ is called *reachable* if there exists an input that can drive the network from origin $\mathbf{0}_{N_f}$ to $\boldsymbol{x_f^*}$ in a finite amount of time. The set of all reachable states constitutes the *controllable subspace*.[1] The *dimension* of the controllable subspace is an important control-theoretic property and can be computed by the *rank* of the *Controllability matrix* below, (Rahmani et al., 2009).

$$\mathcal{C} = \begin{bmatrix} -B & (-A)(-B) & \cdots & (-A)^{N_f-1}(-B) \end{bmatrix}. \tag{8}$$

The rank of the above matrix depends on A and B, which in turn depend on the network graph and the selection of leaders. The network is *completely controllable* if and only if $rank(\mathcal{C}) = N_f$. *Controllability Gramian* is an important mathematical object that provides crucial information about the control behavior of the network (Pasqualetti et al., 2014; Summers et al., 2015; Wu-Yan et al., 2018). Using controllability Gramian, we can quantify how 'easy' it is to go from one state to another in terms of the required control energy equation 6. For the system in equation 5, the *infinte horizon controllability Gramian* is defined as (Summers et al., 2015; Pasqualetti et al., 2014),

$$\mathcal{W} = \int_0^{\infty} e^{-A\tau}(-B)(-B)^T e^{-A^T\tau} d\tau \; \in \; \mathbb{R}^{N_f \times N_f}. \tag{9}$$

If the system is stable, that is, all eigenvalues of $-A$ have negative real parts, then $\mathcal{W}$ converges asymptotically and can be computed by the Lyapunov equation,

$$(-A)\mathcal{W} + \mathcal{W}(-A)^T + (-B)(-B)^T = 0, \tag{10}$$

which is a system of linear equations and is therefore easily solvable. For the solution of equation 10 to exist, $-A$ must be a stable matrix, which is true for connected graphs.

Lemma 1. *If we partition the Laplacian matrix L of an undirected connected graph as in equation 4, then the matrix A is positive definite (Mesbahi & Egerstedt, 2010).*

As a result, $-A$ is negative definite in the case of connected graphs and the system in equation 5 is stable, and the corresponding $\mathcal{W}$ can be computed. Controllability Gramian provides an energy-related quantification of controllability, and we can obtain several controllability statistics from $\mathcal{W}$ (Pasqualetti et al., 2014; Summers et al., 2015; Wu-Yan et al., 2018). We discuss some of them below.

[1] In continuous linear time-invariant systems, as in equation 5, if a state $\boldsymbol{x_f^*}$ is reachable from the origin, then $\boldsymbol{x_f^*}$ is also reachable from an arbitrary initial state in any duration of time.

- **Trace of** $\mathcal{W}$: The trace of the controllability Gramian is inversely related to the average control energy over random target states. It can also be considered as a measure of average controllability in all directions in the state space.
- **Minimum eigenvalue of** $\mathcal{W}$: It is the worst-case metric that is inversely proportional to the control energy required to steer the network in the least controllable direction in the controllable subspace.
- **Rank of** $\mathcal{W}$: The rank of $\mathcal{W}$ is the dimension of the controllable subspace.
- **Determinant of** $\mathcal{W}$: The quantity $\text{ld}(\mathcal{W}) = \log\left(\prod_j \mu_j(\mathcal{W})\right)$, where $\mu_j(\mathcal{W})$ is a non-zero eigenvalue of $\mathcal{W}$, is a volumetric measure of the controllable subspace reachable with one unit or less of control energy. If the system is completely controllable, then $\text{ld}(\mathcal{W})$ is the log determinant of $\mathcal{W}$.

3.3 Potential of Network Controls for Graph Representation

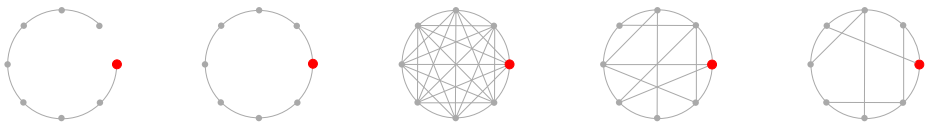
The controllability metrics, as defined in the above section, have the capability to capture the underlying local and global network topology and merit to distinguish certain graph families. For instance, Wang in (Wang, 2013) proves that every graph G satisfying a constraint on the determinant of the *controllability matrix* $\mathcal{C}$ can be distinguished by the collective spectrum (eigenvalues) of the adjacency matrix $A \in \mathbb{R}^{N\times N}$ of graph G and its complement. Theorem 1 in (Wei, 2017) states:

Theorem 3.1. *Every graph in family $\mathcal{F}_N$ can be determined completely by its generalized spectrum, where*

$$\mathcal{F}_N = \{G \mid \frac{det(\mathcal{C})}{2^{\lfloor \frac{N}{2} \rfloor}} \textit{is an odd square-free integer}\} \tag{11}$$

Wang et al. extends this theorem and proves in (Wang & Qiu, 2022) that the *tournament networks* belong to $\mathcal{F}_N$ and can be identified from the adjacency spectrum exhibiting the *usefulness of controllibility metrics in distinguishing graphs*. Similarly, there are numerous results in the literature providing insights into the relationship between the network topology and the controllability properties, thus, establishing the potential of controllability ideas to distinguish graphs in various families. Some simple examples to further illustrate the differentiating capabilities of the control metrics are presented below.

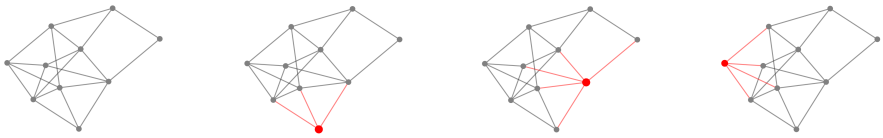
Examples: We illustrate through examples that network controllability depends on the topological organization of the network and the location of leaders in it. Figure 1 shows various networks, each of which has eight agents, including a single leader agent. The controllability properties of the resulting follower networks, for instance, the rank and the trace of the controllability Gramian, denoted by **rank** and **tr**, respectively, vary in networks. The path network in Figure 1(a) is completely controllable with a single leader agent, which is one of the end nodes. At the same time, the complete network in Figure 1(c) is least controllable as the rank of the controllability Gramian is 1. Similarly, in other networks, the controllability attributes are functions of the network graph. Along with the network topology, leader selection also affects the network controllability, as illustrated in Figure 2. We consider a network of 10 agents, of which one is a leader agent, which means $N_f = 9$. In Figure 2(b), the dimension of the controllable subspace is 9, which means that the follower network is completely controllable. The edges between a leader and follower nodes, which decide the structure of B matrix in equation 5 are shown in red. In Figure 2(c), we choose a different leader and observe that the network remains completely controllable; however, the trace of $\mathcal{W}$ changes.



(a) rank = 7, tr = 0.5 (b) rank = 4, tr = 1 (c) rank = 1, tr = 3.5 (d) rank = 5, tr = 2 (e) rank = 7, tr = 1.5

Figure 1: Controllability metrics are functions of the underlying network graph.

To collect valuable information about the graph structure, we need to probe the network effectively. This can be achieved by varying the number and locations of leader nodes and observing the resulting controllability behavior using measures $\mathbf{tr}(\mathcal{W})$, $\mu_j(\mathcal{W})$, $\mathbf{rank}(\mathcal{W})$ and $\mathbf{ld}(\mathcal{W})$. In the next section,



(a) input G (b) rank = 9, tr = 1.5 (c) rank = 9, tr = 2.5 (d) rank = 8, tr = 2

Figure 2: Controllability metrics vary with leader selection

we use these controllability metrics collected by various choices of leader selection to construct useful graph representations.

Remark 1. The consensus dynamics over the network can be defined in several other ways, for instance, by considering the overall state of the network (instead of only the followers' states $\boldsymbol{x_f}$) and therefore, selecting $-L$ as the system matrix.

4 Graph Representation Design

This section describes how we design our graph representation based on controllability characteristics defined in the previous section. If the graph is not connected, then the systems matrix $-A$ in equation 5 might not be stable for some choice of leader agents. Consequently, the solution of equation 10 might not exist. Therefore, we assume that the input network graph is connected. However, some real-world phenomena may generate networks that are disconnected. To handle such cases, we perform a preprocessing step in which we introduce a new vertex in such a graph and add an edge from this new vertex to all other vertices in the graph. This ensures that the graph becomes connected. Similarly, for extremely tiny graphs (with less than ten vertices), we perform a cloning step in which multiple copies of the original graph are generated to ensure that the input graph contains at least ten vertices. This is not an ideal solution as it tampers with the graph's structure, but we believe this preprocessing is rarely needed, and it retains enough structural information from the original graph, as we illustrate in the evaluation section.

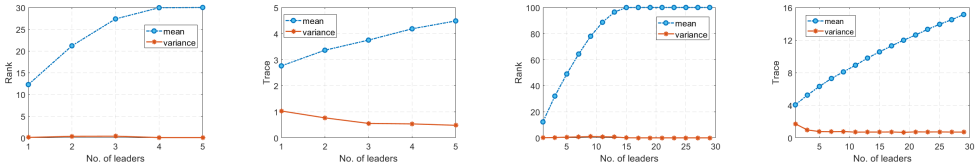
We have two main probes to explore the controllability aspects of a network graph: the *number of leader agents* and their *locations* in the network. For molecular datasets, we utilize node information and the leader selection process is deterministic. For the rest of the datasets, the leader selection process is uniformly random and the features of the Gramian are calculated for several different number of leaders. The details of this selection process for all datasets are provided in Section 5.

For a given set of leader agents, we first compute the corresponding system matrices, i.e., $-A$ and $-B$ by partitioning the Laplacian matrix as in equation 4. Then, we compute the corresponding controllability Gramian $\mathcal{W}$ as in equation 10. We note that to solve the Lyapunov equation equation 10, efficient algorithms and solvers exist that scale well to large networks (Li & White, 2002; Vandereycken & Vandewalle, 2010). We record the trace, rank, minimum (non-zero), and maximum eigenvalues of the Gramian. Along with these controllability measures, we also include a few easy-to-compute statistics about the input graph in our embedding. A step-wise description of this embedding, which we call **CTRL**, is given in Algorithm 1 in the Appendix C. We cater any structural changes to the input graph due to the preprocessing step by also designing an enhanced version, called **CTRL+** . In CTRL**+** , we simply concatenate the information in CTRL with another graph embedding that may have better expressiveness for disconnected and small graphs. For this work, we use a recent graph embedding called *Higher-Order Structure Descriptor (HOSD)*, which is easy to implement and contains the count of various small subgraphs present at multiple scales in the network (Ahmed et al., 2020). Since CTRL is reasonably sized, concatenating with HOSD does not create any severe computational overheads.

Next, we analyze some of the properties of CTRL that are vital for the network classification task as discussed in Section 2.1.

Permutation Invariance: The CTRL uses the spectrum (set of eigenvalues) of the controllability Gramian to calculate the feature descriptors of the given graph G. For a given graph G with N nodes and given leaders V_l, the Gramian is defined as:

$$\mathcal{W}_G^{(V_l)} = \int_0^\infty e^{-A\tau}(-B_{V_l})(-B_{V_l})^T e^{-A^T\tau} d\tau \tag{12}$$



(a) (b) (c) (d)

Figure 3: (a) & (b) show the mean and variance of the Rank and Trace obtained for the ER graphs with $N = 30$ for all combinations of leaders. (c) & (d) show the same features for the ER graphs with $N = 100$ for uniformly randomly selected 50000 leader sets.

where $B_{V_l} \in \mathbb{R}^{N_f \times N_l}$ for V_l leaders. We show that when we select the leader nodes uniformly at random, the expectation of spectrum of the Gramian remains independent of the permutation of vertices in the adjacency and Laplacian matrices.

Theorem 4.1. *The expected spectrum of the Gramian of follower dynamics (as in equation 12) of a leader-follower network G, in which the leaders selection is uniform random, is permutation invariant.*

We prove Theorem 4.1 in the Appendix, which theoretically proves the convergence of the expected spectrum of the Gramian. Additionally, we perform a simple numerical analysis on Erdős-Rényi (ER) graphs that shows that the CTRL descriptor converges towards the mean value with fairly low variance for uniformly random leader selection. We randomly generate 25 graphs with $N = 30$ nodes and the probability p of an edge between any two nodes to be 0.18. We plot the mean and variance of rank and trace of the Gramian for all the leaders combinations from 1 to 5 as shown in Figures 3a and 3b. We also generate 25 ER graphs with $N = 100$ and $p = 0.18$. For these larger graphs, we uniformly randomly select 50000 leaders sets for number of leaders from 1 to 30 and plot the mean and variance values of rank and trace of the Gramian shown in Figures 3c and 3d, respectively. Interestingly, the rank has very low variance for both graphs sizes. The variance of the trace value is relatively high for lower number of nodes (because of lower number of leaders combinations), but drops for higher number of leaders.

Scale Invariance: Another desirable property of graph descriptors is the scale invariance. Previous studies show that for certain families of graphs, some of the control descriptor features have the capability to be consistent with the variation of the size of the graphs. For instance, the Gramian is full rank when either terminal node of a path graph is selected as a leader (Rahmani et al., 2009). Liu et al. (Liu & Ji, 2018) provided ample theoretical evidence on scale invariance of the control properties. We provide some of the relevant theoretical results in appendix.

5 Experimental Evaluation

We evaluate the performance of the proposed embeddings on graph classification tasks and compare the results with the state-of-the-art graph representation methods. We consider classification accuracy as an evaluation metric and use 10-fold cross-validation in our experimental setting. We repeat the experiments ten times and report the mean of the best results of each iteration. CTRL and CTRL**+** embeddings are implemented in Python and all the experiments are performed on the Amazon Web Services instance ($c5.24xlarge$) with 96-cores and 192 GB of RAM.

Datasets: We perform experiments on 10 standard graph classification benchmark datasets. MUTAG, PTC_MR, PROTEINS, ENZYMES, NCI1, and DD, are six bioinformatics datasets. IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, and REDDIT-MULTI-5K are four social network datasets (Morris et al., 2020). The bioinformatics datasets describe small molecules and chemical compounds that belong to two classes except for the ENZYMES dataset which consists of six classes. Among the social network datasets, IMDB-BINARY and IMDB-MULTI describe actors' ego-networks while REDDIT-BINARY and REDDIT-MULTI are chosen subreddits from an online social network. The graphs in IMDB-BINARY and REDDIT-BINARY are labeled with two classes and there are three

and five classes in IMDB-MULTI and REDDIT-MULTI datasets, respectively. Source code and data are also made available for foster reproducibility of the results[2].

Table 1: Graph classification accuracy comparison of CTRL and CTRL+ against spectral and statistical graph representation methods. Top three results are highlighted by First, Second and **Third**. OME is out of memory and $> D$ indicates computations exceeds 24 hours.

Dataset	SP	SVM theta	GK	NetSIMILE	NetLSD	FGSD	CTRL	CTRL+
MUTAG	87.28	75.06	79.8	84.63	85.28	**88.07**	90.99	89.94
PTC_MR	62.50	58.40	64.87	59.09	61.19	58.91	65.70	**62.78**
PROTEINS	72.68	68.64	72.51	71.15	**74.63**	70.18	75.2	74.76
ENZYMES	36.33	17.50	34.33	42.10	**44.25**	33.95	69.67	65.17
NCI1	68.69	50.68	61.22	73.96	76.31	**79.60**	81.31	81.8
DD	77.51	58.66	71.54	74.72	**77.27**	76.60	76.23	78.19
IMDB-B	72.40	50.0	76.60	74.41	73.79	73.88	**74.8**	75.0
IMDB-M	48.73	41.13	**49.66**	49.32	50.57	50.55	48.0	49.0
REDDIT-B	85.30	50.4	$> D$	**89.53**	89.12	81.60	95.9	96.05
REDDIT-M-5K	44.95	20.0	$> D$	52.78	53.58	OME	**52.91**	54.69

Baselines: We consider six graph embeddings methods: Shortest Path (SP)(Borgwardt & Kriegel, 2005), SVM theta (Johansson et al., 2014) , GK (Shervashidze et al., 2009), NetSIMILE (Berlingerio et al., 2013), NetLSD (Tsitsulin et al., 2018), and FGSD (Verma & Zhang, 2017) for comparing the performance of the proposed method. Among them, the first three are state-of-the-art graph kernel methods while the later are recently proposed graph descriptors. NetLSD and FGSD use graphs' spectral features to extract graph information. NetSIMILE is a graph descriptor that is based on seven simple graph statistics including average vertex degree, average clustering coefficient, and standard deviation of the two-hops neighborhood. To evaluate the performance against state-of-the-art and recent GNN methods, we also consider nine Graph Neural Networks (GNNs) models: DGCNN (Zhang et al., 2018), DiffPool (Ying et al., 2018), ECC (Simonovsky & Komodakis, 2017), GIN (Xu et al., 2018), Nested Graph Neural Network (NGNN) with GIN (Zhang & Li, 2021), Cell Isomorphism Networks (CIN) (Bodnar et al., 2021a), Message Passing Simplicial Networks (SIN) (Bodnar et al., 2021b), Structural Semantic Readout (SSRead) (Lee et al., 2021) with SUM, and RepPool (Li et al., 2020) for comparison. The GNNs models chosen for comparison include earlier well-known models and recent models showing promising results on the graph classification task.

Experimental setup:

To obtain the control features, we consider different leader selection techniques. The leader set selection process is as follows:

- For molecular datasets i.e. MUTAG, PTC, and NCI1, we use node types for leaders selection process. We take all the nodes as leaders of the same type and repeat for all the distinct types of nodes in the dataset. For the ENZYMES dataset, the first node feature is considered as node type.
- For the rest of the datasets, the leader selection process is random. There are exponentially many choices to select N_l leaders among a set of N vertices. Therefore, we make this choice randomly. Formally, we uniformly select N_l leader nodes from the vertex set and repeat this random selection process c times for a given number of leaders. We first consider a fixed number of leaders, i.e., 1, 2, 5, 9, and then consider a fraction of overall nodes to be leaders, that is, 2%, 5%, 10%, 20%, 30% of the total nodes. For each random leader sets choice, we repeat the experiment 30 times. Every time a set of leader nodes is selected, we record graph measurements presented in the Algorithm 1 in Appendix C. We use minimum, maximum, and average values of these measures over c iterations in our graph embedding.

We ensure through a preprocessing step that each graph has at least 10 nodes and is connected. We use the Random Forest (RF) algorithm with grid search for the classification and reported $10-$fold cross validation accuracy and their standard deviation. In the hyper-parameter setting of RF, we choose $\sqrt{d}$ features for building a tree, where d is the feature vector's size, and the classical Gini impurity is used

[2] https://anonymous.4open.science/r/Control-Descriptor_ICLR-D1D3/README.md

Table 2: Graph classification accuracy with standard deviation comparison against GNNs. Top three results are highlighted by First, Second and **Third**. OME is out of memory and N/A is the unavailability of the reproduced results (see Section 5). $>$ 3D indicates training time exceeds 3 days.

datasets	DGCNN	DiffPool	ECC	GIN	NGNN	CIN	SIN	SSRead(SUM)	RepPool	CTRL	CTRL+
ENZYMES	38.9 ± 5.7	59.5 ± 5.6	29.5 ± 8.2	**59.6** ± 4.5	33.0 ± 6.1	N/A	N/A	43.83 ± 8.8	50.76 ± 7.9	69.67 ± 4.8	65.17 ± 5.9
NCI1	76.4 ± 1.7	76.9 ± 1.9	76.2 ± 1.4	80.0 ± 1.4	78.6 ± 1.2	84.13 ± 1.6	80.26 ± 1.8	82.70 ± 1.84	79.53 ± 3.4	81.31 ± 2.1	**81.8** ± 1.6
PROTEINS	72.9 ± 3.5	73.7 ± 3.5	72.3 ± 3.4	73.3 ± 4.0	74.1 ± 3.7	73.24 ± 5.5	76.30 ± 4.0	73.76 ± 4.8	77.78 ± 3.7	**75.2** ± 3.1	74.76 ± 2.9
DD	**76.6** ± 4.3	75.0 ± 3.5	72.6 ± 4.1	75.3 ± 2.9	76.0 ± 4.3	N/A	N/A	70.23 ± 1.0	79.77 ± 4.8	76.23 ± 3.0	78.19 ± 2.2
IMDB-B	69.2 ± 3.0	68.4 ± 3.3	67.7 ± 2.8	71.2 ± 3.9	72.0 ± 4.2	73.90 ± 6.0	75.5 ± 3.8	71.4 ± 2.0	73.93 ± 4.7	**74.8** ± 2.3	75.0 ± 2.6
IMDB-M	45.6 ± 3.4	45.6 ± 3.4	43.5 ± 3.1	48.5 ± 3.3	**50.9** ± 3.4	50.93 ± 2.6	52.06 ± 3.4	48.66 ± 2.8	47.03 ± 3.8	48.0 ± 3.9	49.0 ± 3.7
REDDIT-B	87.8 ± 2.5	89.1 ± 1.6	OME	89.9 ± 1.9	$>$ 3D	92.4 ± 2.1	93.0 ± 5.9	**93.8** ± 3.5	90.58 ± 4.5	95.9 ± 1.3	96.05 ± 1.5
REDDIT-5K	49.2 ± 1.2	53.8 ± 1.4	OME	56.1 ± 1.7	$>$ 3D	31.8 ± 3.1	57.09 ± 1.7	53.25 ± 2.0	N/A	52.91 ± 1.7	**54.69** ± 2.2

as a metric to build the tree. The number of estimators is chosen from $\{50, 100, 500\}$ and total number of samples for a split in a tree is chosen from the set $\{2, 3, 4, 5\}$. In FGSD experiments, we set 0.0001 bin-width as recommended in (Tsitsulin et al., 2018). For NetLSD, we use all variants mentioned in their paper and report the best results by utilizing the entire eigenspectrum. For NetSIMILE, we use their publicly available source codes and reproduce the results using our experimental setup. For graph kernel results, we use GraKel (Siglidis et al., 2020) library for computing kernel matrices and then use RF with the same setting for the classification. For GNNs, we reproduced the result of NGNN, CIN (Bodnar et al., 2021a), SIN (Bodnar et al., 2021b), GCN-SSRead (Lee et al., 2021) and RepPool (Li et al., 2020) with 10-fold cross validation using the source codes made publicly available by the authors. The hyper-parameters are same as in the original papers. Here, we would like to note that due to the unavailability of a few datasets in the desired formats, we were not able to reproduce some of the results, hence N/A is reported. For DGCNN (Zhang et al., 2018), DiffPool (Ying et al., 2018), ECC (Simonovsky & Komodakis, 2017) and GIN (Xu et al., 2018), we consider the results in (Errica et al., 2019) that are obtained through identical frameworks.

Classification results:

We present the classification results of our evaluation in Table 1 and Table 2. We conclude from the results that the proposed embeddings either outperform or achieve comparable performance in terms of prediction accuracy on all benchmarks. Specifically, in comparison to the embedding methods, CTRL and CTRL+ rank top on eight out of ten datasets. On the remaining datasets, the results are within 2% of the top results. In comparison to GNN models in Table 2, we observe the superior performance of both CTRL and CTRL+ on ENZYMES and REDDIT-B datasets. On the remaining datasets except IMDB-M, CTRL and CTRL+ achieve second or third place. Although the proposed descriptor did not place in the top three on the IMDB-M dataset, the results are within 3% of the top results. These results clearly demonstrate the effectiveness of the proposed descriptor for graph classification.

We expect the results to further improve when combined with existing graph embeddings (spectral, statistical or GNNs). Our results empirically confirm that the spectral information of the network Gramian is effective for constructing graph representations.

6 Conclusion and Future Work

This work asserts that the networked dynamical system perspective, in particular, the network controllability paradigm offers a unique approach to encode the network structure and obtain effective graph representations. There are several directions to advance the controllability framework for graph representations. For instance, instead of Laplacian dynamics, one can consider different dynamical processes over networks. Similarly, we can use other controllability notions, such as, structural controllability, output, or target controllability, which concerns controlling a focused set of (target) nodes instead of the entire network (Van Waarde et al., 2017; You et al., 2019). Also, there are alternative metrics that can be used to express the network's dynamical behavior, for instance, control centrality (Liu et al., 2012), Gramian-based edge centrality, control range index (Wang et al., 2012), and others (e.g., (Commault & Dion, 2013; Wu-Yan et al., 2018)). Network control and graph learning communities share several scientific grounds, and viewing graph learning problems from the lens of control theory offers fresh perspectives and approaches to advance the field.

REFERENCES

Saeed Ahmadizadeh, Iman Shames, Samuel Martin, and Dragan Nešić. On eigenvalues of Laplacian matrix for a class of directed signed graphs. *Linear Algebra and its Applications*, 523:281–306, 2017.

Ammar Ahmed, Zohair Raza Hassan, and Mudassir Shabbir. Interpretable multi-scale graph descriptors via structural compression. *Information Sciences*, 533:169–180, 2020.

Michele Berlingerio, Danai Koutra, Tina Eliassi-Rad, and Christos Faloutsos. Network similarity via multiple social theories. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 1439–1440. ACM, 2013.

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Lio, Guido F Montufar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks. *Advances in Neural Information Processing Systems*, 2021a.

Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F Montufar, Pietro Lio, and Michael Bronstein. Weisfeiler and lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning*, pp. 1026–1037. PMLR, 2021b.

Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *IEEE International Conference on Data Mining (ICDM)*, pp. 74–81. IEEE Computer Society, 2005.

Xiaodong Cheng and Jacquelien MA Scherpen. Novel Gramians for linear semistable systems. *Automatica*, 115:108911, 2020.

Christian Commault and Jean-Michel Dion. Input addition and leader selection for the controllability of graph-based systems. *Automatica*, 49(11):3322–3328, 2013.

Magnus Egerstedt, Simone Martini, Ming Cao, Kanat Camlibel, and Antonio Bicchi. Interacting with networks: How does structure relate to controllability in single-leader, consensus networks? *IEEE Control Systems Magazine*, 32(4):66–73, 2012.

Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. *arXiv preprint arXiv:1912.09893*, 2019.

Fredrik Johansson, Vinay Jethava, Devdatt Dubhashi, and Chiranjib Bhattacharyya. Global graph kernels using geometric embeddings. In *International Conference on Machine Learning*, pp. 694–702. PMLR, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2017.

Risi Kondor, Nino Shervashidze, and Karsten M Borgwardt. The graphlet spectrum. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 529–536. ACM, 2009.

Dongha Lee, Su Kim, Seonghyeon Lee, Chanyoung Park, and Hwanjo Yu. Learnable structural semantic readout for graph classification. In *2021 IEEE International Conference on Data Mining (ICDM)*, pp. 1180–1185. IEEE, 2021.

Jing-Rebecca Li and Jacob White. Low rank solution of Lyapunov equations. *SIAM Journal on Matrix Analysis and Applications*, 24:260–280, 2002.

Juanhui Li, Yao Ma, Yiqi Wang, Charu Aggarwal, Chang-Dong Wang, and Jiliang Tang. Graph pooling with representativeness. In *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 302–311. IEEE, 2020.

Xianzhu Liu and Zhijian Ji. Controllability of multiagent systems based on path and cycle graphs. *International Journal of Robust and Nonlinear Control*, 28(1):296–309, 2018.

Yang-Yu Liu, Jean-Jacques Slotine, and Albert-László Barabási. Controllability of complex networks. *Nature*, 473(7346):167–173, 2011.

Yang-Yu Liu, Jean-Jacques Slotine, and Albert-László Barabási. Control centrality and hierarchical structure in complex networks. *Plos One*, 7, 2012.

Mehran Mesbahi and Magnus Egerstedt. *Graph theoretic methods in multiagent networks*, volume 33. Princeton University Press, 2010.

Nima Monshizadeh, Shuo Zhang, and M Kanat Camlibel. Zero forcing sets and controllability of dynamical systems defined on graphs. *IEEE Transactions on Automatic Control*, 59(9):2562–2567, 2014.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020.

Jose C Nacher and Tatsuya Akutsu. Analysis of critical and redundant nodes in controlling directed and undirected complex networks using dominating sets. *Journal of Complex Networks*, 2(4): 394–412, 2014.

Alessandro Negro. *Graph-powered machine learning*. Simon and Schuster, 2021.

Fabio Pasqualetti, Sandro Zampieri, and Francesco Bullo. Controllability metrics, limitations and algorithms for complex networks. *IEEE Transactions on Control of Network Systems*, 1(1):40–52, 2014.

Thilo Penzl. A cyclic low-rank smith method for large sparse lyapunov equations. *SIAM Journal on Scientific Computing*, 21(4):1401–1418, 1999.

Amirreza Rahmani, Meng Ji, Mehran Mesbahi, and Magnus Egerstedt. Controllability of multi-agent systems from a graph-theoretic perspective. *SIAM Journal on Control and Optimization*, 48(1): 162–186, 2009.

Anwar Said, Saeed-Ul Hassan, Suppawong Tuarob, Raheel Nawaz, and Mudassir Shabbir. Dgsd: Distributed graph representation via graph statistical properties. *Future Generation Computer Systems*, 119:166–175, 2021.

Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pp. 488–495. PMLR, 2009.

Giannis Siglidis, Giannis Nikolentzos, Stratis Limnios, Christos Giatsidis, Konstantinos Skianis, and Michalis Vazirgiannis. Grakel: A graph kernel library in python. *Journal of Machine Learning Research*, 21(54):1–5, 2020.

Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Tyler H Summers, Fabrizio L Cortesi, and John Lygeros. On submodularity and controllability in complex dynamical networks. *IEEE Transactions on Control of Network Systems*, 3(1):91–101, 2015.

Anton Tsitsulin, Davide Mottin, Panagiotis Karras, Alexander Bronstein, and Emmanuel Müller. Netlsd: hearing the shape of a graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2347–2356. ACM, 2018.

Henk J Van Waarde, M Kanat Camlibel, and Harry L Trentelman. A distance-based approach to strong target control of dynamical networks. *IEEE Transactions on Automatic Control*, 62(12): 6266–6277, 2017.

Bart Vandereycken and Stefan Vandewalle. A Riemannian optimization approach for computing low-rank solutions of Lyapunov equations. *SIAM Journal on Matrix Analysis and Applications*, 31 (5):2553–2579, 2010.

Saurabh Verma and Zhi-Li Zhang. Hunt for the unique, stable, sparse and fast feature learning on graphs. In *Advances in Neural Information Processing Systems*, pp. 88–98. Curran Associates, Inc., 2017.

Bingbo Wang, Lin Gao, and Yong Gao. Control range: a controllability-based index for node significance in directed networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2012 (04):P04011, 2012.

Wei Wang. Generalized spectral characterization of graphs: Revisited. *Electronic Journal of Combinatorics*, 20(4), 2013.

Wei Wang and Lihong Qiu. Spectral characterizations of tournaments. *Discrete Mathematics*, 345(8): 112918, 2022.

Wang Wei. A simple arithmetic criterion for graphs being determined by their generalized spectra. *Journal of Combinatorial Theory, Series B*, 122:438–451, 2017.

Elena Wu-Yan, Richard F Betzel, Evelyn Tang, Shi Gu, Fabio Pasqualetti, and Danielle S Bassett. Benchmarking measures of network controllability on canonical graph models. *Journal of Nonlinear Science*, pp. 1–39, 2018.

Weiguo Xia and Ming Cao. Analysis and applications of spectral properties of grounded laplacian matrices for directed networks. *Automatica*, 80:10–16, 2017.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1365–1374. ACM, 2015.

AY Yazıcıoğlu, Waseem Abbas, and Magnus Egerstedt. Graph distances and controllability of networks. *IEEE Transactions on Automatic Control*, 61(12):4125–4130, 2016.

Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *arXiv preprint arXiv:1806.08804*, 2018.

Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems*, pp. 6410–6421. Curran Associates, Inc., 2018.

Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *International Conference on Machine Learning*, pp. 7134–7143. PMLR, 2019.

Muhan Zhang and Pan Li. Nested graph neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

APPENDIX A RELATED WORK

Graph representation methods can be divided into two main approaches: *Graph kernels* and *Graph Neural Networks (GNNs)*. Graph kernels are well-known and are historically dominant techniques for graph comparison or graph classification, whereas GNNs are recent end-to-end deep learning models for learning graph structures. We briefly survey both of these techniques in the following.

Graph Kernels: Graph kernels are widely known methods for learning graph-structured data. Some of the most popular approaches include, shortest-path kernel (Borgwardt & Kriegel, 2005), Weisfeiler Lehman kernel, deep graph kernel (Yanardag & Vishwanathan, 2015), and graphlet kernel (Kondor et al., 2009). These methods use various graph-theoretic measures such as pairwise distances, subgraphs mining, and neighborhood aggregation for extracting graph representations (Verma & Zhang, 2017; Tsitsulin et al., 2018; Said et al., 2021). Graph kernel methods, though useful on the graph classification tasks with attractive time complexities, have been outperformed by Graph Neural Networks (GNNs) approach that we review below.

Graph Neural Networks (GNN): GNNs are useful tools for learning graph representations. The last few years have seen a surge in GNNs approaches, introducing different techniques for improving models' capabilities. Recently, various graph representation approaches have been introduced that focus on different aspects of the GNN methods i.e. scalability, robustness, generalizability, and explanability (Negro, 2021). Such approaches include Graph Convolutional Networks (Kipf & Welling, 2017), and Graph Reinforcement Learning (You et al., 2018). These graph neural network approaches report a state-of-the-art classification accuracy on several standard graph datasets. However, their sizable variance is of concern for certain applications (Xu et al., 2018). Additionally, they utilize node features in the learning process, whereas the kernel methods usually work on unlabelled graphs.

Unlike the existing works, we pursue a unique approach to seek expressive graph representation. We consider graphs as networked dynamical systems and observe their controllability properties, revealing the extent to which a network can be manipulated. Using control theory, we employ tools to capture the relationship between networks' control behavior and their underlying topologies. We then propose a graph representation based on control properties that exhibit good classification accuracy for a broad range of datasets.

APPENDIX B LAPLACIAN DYNAMICS IN DIRECTED GRAPHS

For a directed graph $G = (V, E)$, the *in-neighbors* of v_i are $\mathcal{N}_i^{\mathbf{i}} = \{v_j \in V : (v_j, v_i) \in E\}$. Similarly, the *out-neighbors* of v_i are $\mathcal{N}_i^{\mathbf{o}} = \{v_j \in V : (v_i, v_j) \in E\}$. The *in-degree* and *out-degree* of v_i are simply $\delta_i^{\mathbf{i}} = |\mathcal{N}_i^{\mathbf{i}}|$ and $\delta_i^{\mathbf{o}} = |\mathcal{N}_i^{\mathbf{o}}|$, respectively. For $|V| = N$, we define Laplcaian matrix $\tilde{L}$ (of dimensions $N \times N$) of the directed graph as below (Xia & Cao, 2017):

$$[\tilde{L}]_{i,j} = \begin{cases} -1 & \text{if } (v_j, v_i) \in E \text{ and } i \neq j, \\ 0 & \text{if } (v_j, v_i) \notin E \text{ and } i \neq j, \\ \delta_i^{\mathbf{i}} & \text{if } i = j. \end{cases} \tag{13}$$

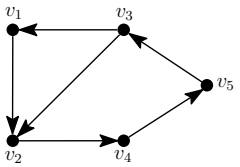
Note that $\tilde{L}$ is not a symmetric matrix here (unlike the undirected graphs). Suppose, we have a set of N_f followers given by $V_f = \{v_1, v_2, \cdots, v_{N_f}\}$ and N_ℓ leaders denoted by $V_\ell = \{v_{N_f+1}, \cdots, v_N\}$, then we can partition $\tilde{L}$ as

$$\tilde{L} = \left[\begin{array}{c|c} \tilde{A} & \tilde{B} \\ \hline \tilde{C} & \tilde{D} \end{array}\right], \tag{14}$$

where $\tilde{A} \in \mathbb{R}^{N_f \times N_f}$, $\tilde{B} \in \mathbb{R}^{N_f \times N_\ell}$, $\tilde{C} \in \mathbb{R}^{N_\ell \times N_f}$ and $\tilde{D} \in \mathbb{R}^{\mathbb{N}_\ell \times \mathbb{N}_\ell}$. Figure 4 illustrates an example of the directed Laplacian matrix.

The follower nodes update their states $\boldsymbol{x}_f(t)$ according to the following dynamics:

$$\dot{\boldsymbol{x}}_f(t) = -\tilde{A}\boldsymbol{x}_f(t) - \tilde{B}u(t). \tag{15}$$



$$\tilde{L} = \left[\begin{array}{cccc|c} 1 & 0 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & -1 & 1 \end{array}\right]$$

Figure 4: A directed graph with four followers $V_f = \{v_1, v_2, v_3, v_4\}$ and one leader $V_\ell = \{v_5\}$ along with the corresponding Laplacian matrix $\tilde{L}$ and its partition as in equation 14.

To obtain the infinite horizon controllability Gramian (defined in (9) in the main text) corresponding to the above system, $-\tilde{A}$ must be a stable matrix, i.e., the real parts of all eigen values of $-\tilde{A}$ must be negative. Under some connectivity condition on G (described in the Lemma below), we get a stable $-\tilde{A}$ (Xia & Cao, 2017). As a result, we can find the infinite horizon controllability Gramian and obtain the controllability metrics defined previously.

Lemma Let $G = (V_f \cup V_\ell, E)$ be a directed graph such that for every follower node $v_i \in V_f$, there exists some leader node $v_j \in V_\ell$ such that there is a directed path from v_j to v_i. If $\tilde{L}$ is the Laplacian matrix of G with $\tilde{A}$ as defined in equation 14, then $-\tilde{A}$ is stable (Xia & Cao, 2017).

Remark 1. If the systems matrix, for instance, $-\tilde{A}$ in equation 15, is semistable instead of being stable (may be due to graph connectivity conditions), then we can use the generalized controllability Gramian $\mathcal{W}_g$, instead of $\mathcal{W}$ (Cheng & Scherpen, 2020). In other words, if $-\tilde{A}$ has zero eigen value such that the geometric multiplicity of zero eigen value coincides with the algebraic multiplicity, and all the other eigen values have negative real parts, then we can compute the generalized controllability Gramian $\mathcal{W}_g$, which is given below for the system in equation 15 (Cheng & Scherpen, 2020).

$$\mathcal{W}_g = \int_0^\infty \left(e^{-\tilde{A}\tau} - \mathcal{J}\right)(-\tilde{B})(-\tilde{B})^T \left(e^{-\tilde{A}^T\tau} - \mathcal{J}\right) d\tau, \tag{16}$$

where $\mathcal{W}_g \in \mathbb{R}^{N_f \times N_f}$ and $\mathcal{J} = \lim_{\tau\to\infty} e^{-\tilde{A}\tau}$ is a constant matrix. For a semistable system, the generalized controllability Gramian is well defined and is a unique solution of the following linear matrix equations (Cheng & Scherpen, 2020).

$$\begin{aligned} &(-\tilde{A})\mathcal{W}_g + \mathcal{W}_g(-\tilde{A})^T + (I - \mathcal{J})(-\tilde{B})(-\tilde{B}^T)(I - \mathcal{J}^T) = 0, \\ &\mathcal{J}\mathcal{W}_g\mathcal{J}^T = 0. \end{aligned} \tag{17}$$

Hence, we can use $\mathcal{W}_g$ to compute the controllability metrics.

APPENDIX C ALGORITHM

To expand on graph representation design presented in section 4 of the main paper, here we present a complete step-by-step description of the **CTRL** embedding in Algorithm 1. Starting from a given graph G, the Algorithm presents all of the details to construct the desired **CTRL** embedding.

APPENDIX D PROSPECTIVE OF NETWORK CONTROL FOR GRAPH REPRESENTATION

Permutation and scale invariance are two challenging but necessary properties of any graph embedding method to posses. In this section, we provide a proof for permutation invariance as presented in Theorem 4.1 and as well as report some theoretical results from the literature that demonstrate the scale invariance properties for the control properties.

Permutation Invariance: *Proof of Theorem 4.1 :* For uniform random selection of leaders, the expectation of the Gramian matrix $\mathcal{W}_G^{(j)}$ is:

$$E[\mathcal{W}_G] = \frac{1}{\mathcal{K}_l} \sum_{j=1}^{\mathcal{K}_l} \int_0^\infty e^{-A\tau}(-B_{V_j})(-B_{V_j})^T e^{(-A\tau)^T} d\tau \tag{18}$$

Algorithm 1 CTRL: (Control embedding)

Input: Graph $G = (V, E)$, $|V| = N$, num_iterations
Output: Graph embedding $\mathbb{R}$

1: $L \leftarrow$ Laplacian of G.
2: Initialize $\mathbb{R}$ as an empty list.
3: Create a list of leader sets using the *leader selection strategy* `(see experimental setup).`
4: **for** each V_l in the list of leader sets **do**
5: Compute the corresponding system matrices $-A, -B$ (as in equation 5).
6: Compute the Gramian $\mathcal{W}$ (as in equation 9 and equation 10).
7: Compute the rank, trace, minimum non-zero eigenvalue, $\mu_{\min}$, and maximum eigenvalue, $\mu_{\max}$, of $\mathcal{W}$.
8: Compute the Resolvability (number of unique distance-to-leader vectors from (Yazıcıoğlu et al., 2016)) of G corresponding to selected leader nodes
9: Compute rank, trace, $\mu_{\min}$, $\mu_{\max}$, Resolvability value and concatenate to the list $\mathbb{R}$.
10: **end for**
`(Adding some simple stats about the graph structure to` $\mathbb{R}$`.)`
11: Concatenate N, $|E|$, number of bi-connected components, the Laplacian spectrum. `(We add three smallest and largest eigenvalues)`, Laplacian energy, eccentricity spectrum, eccentricity energy, Wiener index, trace degree sequence of G, and cycles information.
12: Return $\mathbb{R}$

where $\mathcal{K}_l = \binom{N}{N_l}$ is the number of choices for a leader set of size N_l, and B_{V_j} is the input matrix corresponding to V_j leader set. A permutation matrix Π is a square matrix, I , formed by a reordering of the rows of the identity matrix of the corresponding dimensions. If $G = (V, E)$ and $G' = (V', E')$ are isomorphic graphs, then there exists a permutation matrix Π such that $\Pi\ L\ \Pi^{-1} = L'$, where L and L' are the corresponding Laplacian matrices of G and G' respectively. Recall that Π satisfies the property $\Pi\ \Pi^T = \Pi^T \Pi = I$. Let $\Pi_f \in \mathbb{R}^{N_f \times N_f}$ be an arbitrary permutation matrix corresponding to the follower agents. The Gramian $\mathcal{W}_{G'}$ of the follower dynamics obtained from a permuted copy of the Laplacian matrix $L' = \Pi L \Pi^{-1}$ is

$$\begin{aligned}
\mathcal{W}_{G'}^{(j)} &= \int_0^\infty e^{-A'\tau}(-B'_{V_j})(-B'_{V_j})^T e^{(-A'\tau)^T} d\tau \\
&= \int_0^\infty e^{-\Pi_f A \tau \Pi_f^{-1}}(-B'_{V_j})(-B'_{V_j})^T e^{(-\Pi_f A \tau \Pi_f^{-1})^T} d\tau \\
&= \Pi_f \int_0^\infty e^{-A\tau} \Pi_f^T (-B'_{V_j})(-B_{V_j})^T \Pi_f e^{(A\tau)^T} d\tau \Pi_f^T
\end{aligned}$$

where $\Pi_f^{-T} = (\Pi_f^{-1})^T = (\Pi_f^T)^T = \Pi_f$. Further, we can write,

$$= \Pi_f \left(\int_0^\infty e^{-A\tau} (\Pi_f^T B'_{V_j})(\Pi_f^T B'_{V_j})^T \Pi_f e^{(A\tau)^T} d\tau \right) \Pi_f^T$$

Note that the summation of expectation in equation 18 is over all choices of leader sets of size N_l. For each $(\Pi_f^T B'_{V_j})$, there is a unique $1 \leq j' \leq \mathcal{K}_l$ such that $(\Pi_f^T B'_{V_j}) = B_{V_{j'}}$. Therefore,

$$\begin{aligned}
E[\mathcal{W}_{G'}] &= \frac{1}{\mathcal{K}_l} \sum_{j'=1}^{\mathcal{K}_l} \Pi_f \mathcal{W}_G^{(j')} \Pi_f^T \\
E[\mathcal{W}_{G'}] &= \Pi_f \left(E[\mathcal{W}_G] \right) \Pi_f^T.
\end{aligned}$$

Thus, the expected Gramian of a permuted graph is same as the the permutation of the expected Gramian of the original graph. Since the spectrum of a matrix is invariant to linear transformations, we conclude that the expected spectrum of the Gramian is preserved under vertex permutations.

Scale Invariance: As mentioned in Section 4, for certain families of graphs, some of the control descriptor features have the capability to be consistent with the variation of the size of the graphs.

Liu et al. in (Liu & Ji, 2018) provides a relation for $\mathbf{rank}(\mathcal{W})$ for path graphs when $N_l = 1$ and the leader is not a terminal node. Theorem 1 of (Liu & Ji, 2018) states that

Theorem D.1. *Suppose $P_N = (V, E)$ be the path graph where $V = \{v_1, \ldots, v_N\}$, and $(v_i, v_{i+1}) \in E \; \forall i = 1, \ldots, N$, and $\mathbb{F}_N = \{f_1, \ldots, f_m\}$, $f_1 > f_2 > \ldots > f_m > 1$, represent the set of the odd factors of N except for 1. If there exists a set M_i whose element m belongs to $\{2, 3, \ldots, N-1\}$ and f_i is the greatest common factor of $2m-1$ and N, and if $j \in M_k$, where $k \in \{1, \ldots, m\}$, then the **rank**$(\mathcal{W}) = N - (f_k - 1)/2$.*

Hence, the $\mathbf{rank}(\mathcal{W})$ normalized by the size of the path graph N is fairly independent of the size of the path graphs. Likewise, for cycle graphs C_N with N nodes, the Gramian is strictly full rank if any two adjacent nodes are selected as leader nodes whereas for a single leader, the $\mathbf{rank}(\mathcal{W})$ is always $\lfloor N/2 \rfloor$ (Liu et al., 2011). For the controllable subspace with two non-adjacent leaders in a cycle graph, the parity is important. Liu et al. in (Liu & Ji, 2018) discuss this controllability of a cycle with two leaders as a function of distances between the leaders. Theorem 4 of (Liu & Ji, 2018) states that

Theorem D.2. *Suppose $C_N = (V, E)$ be the cycle graph where $V = \{v_1, \ldots, v_N\}$, $(v_i, v_{i+1}) \in E \; \forall i = 1, \ldots, N$, $(v_1, v_N) \in E$. Let $I_k = \{mk | m \in \mathbb{N}_+\}$, and $I_k^n = \{x | x \in I_k \; \& \; x \leq [N/2]\}$. For even N, let $\mathbb{F}_N^E = \{f_1, \ldots, f_m\}$ denote all even factors of N except for 2, where $N = f_1 > f_2 > \ldots > f_m > 3$ and let $F_1^E = I_{f_1/2}^N$, $F_k^E = I_{f_k/2}^N \backslash \cup_{i=1}^m F_i^E$ and $k = 2, \ldots, m$. Let $d(v_i, v_j)$ be the number of edges in the shortest path between v_i and v_j. If $d(v_1, v_2) \in F_k^E$, where $V_l = \{v_1, v_2\}$, then **rank**$(\mathcal{W}) = N - f_k/2 + 1$.*

The same procedure can be followed for odd N, thus, exhibiting the scale-invariance of the normalized $\mathbf{rank}(\mathcal{W})$ for cycle graphs to large degree.

APPENDIX E LIMITATIONS

The results of our proposed method are encouraging for network classification. The approach to consider networks as dynamical systems and study its classification properties is novel. However, it has a few limitations discussed below.

Scalability – The major step involved in the computation of the proposed representations is the computation of the infinite horizon controllability Gramian equation 9. Since it requires matrix multiplication equation 10, the overall time complexity is super-quadratic in N_f. Though it was not an issue for the graph classification tasks considered in the paper, the method could pose computational challenges for very large graphs.

We note that the scalability issue was not the main focus of this work, instead, a new approach relying on the control behavior of networks through external perturbations to encode graph structure was the main consideration. Nonetheless, there are several ways to deal with it and would be included in future extensions. Lyapunov equation, whose solution is the controllability Gramian, is a particular case of a more general Sylvester equation. While solving the Sylvester equation through standard methods (e.g., Bartels-Stewart, Hammarling) could be computationally expensive in large networks, several techniques have been developed over the years to significantly improve the computation time for approximately solving Lyapunov equations (LE) with reasonable accuracy. These techniques utilize the additional structure of LE, which includes the low-rank condition of the matrix (as the number of leaders is typically quite small), stability and sparsity of the system matrix, and so on. Some of these methods include iterative methods (e.g., cyclic low-rank Smith method), alternating directions implicit (ADI) methods, Krylov subspace methods, projection methods (e.g., extended Arnoldi or Glarekin method), see (Penzl, 1999) and the references therein.

Leader Selection Mechanism –In the proposed work, we mainly considered random leader selection, albeit there can be other systematic ways; one being leader selection based on node types used for molecular datasets. Optimizing leader selection to maximize the performance for the task at hand could pose a significant computation overhead as leader selection problems are typically computationally challenging. For instance, it is NP-hard to determine if a graph with a fixed number of leader nodes is completely controllable or not. Thus, we desire a simple and computationally efficient scheme that gives good performance for a wide range of applications.

We note that in the control framework, leader nodes provide a mechanism to probe the network externally so that we can record the network control behavior and use it for graph embedding. In the

absence of optimal leader selection, it is a reasonable proposition to probe the network fairly from all directions to achieve this objective. Random leader selection achieves these objectives, that is, efficient computation and fair probing of the network. However, improved results can be expected with a more standardized leader selection. We note an increase of about 10% accuracy in *NCI1* dataset with a systematic leader selection process defined in Section 5. It would be an interesting question to devise an optimal leader selection problem for specific tasks and study rigorously the *trade-off between the accuracy and computational costs.*