JDSSV

Journal of Data Science, Statistics, and Visualisation

MMMMMM YYYY, Volume VV, Issue II. *doi: XX.XXXXX/jdssv.v000.i00*

Network Comparison with Interpretable Contrastive Network Representation Learning

Takanori Fujiwara
University of California, Davis

Jian Zhao
University of Waterloo

Francine Chen
Toyota Research Institute

Yaoliang Yu
University of Waterloo

Kwan-Liu Ma
University of California, Davis

Abstract

Identifying unique characteristics in a network through comparison with another network is an essential network analysis task. For example, with networks of protein interactions obtained from normal and cancer tissues, we can discover unique types of interactions in cancer tissues. This analysis task could be greatly assisted by contrastive learning, which is an emerging analysis approach to discover salient patterns in one dataset relative to another. However, existing contrastive learning methods cannot be directly applied to networks as they are designed only for high-dimensional data analysis. To address this problem, we introduce a new analysis approach called *contrastive network representation learning* (cNRL). By integrating two machine learning schemes, network representation learning and contrastive learning, cNRL enables embedding of network nodes into a low-dimensional representation that reveals the uniqueness of one network compared to another. Within this approach, we also design a method, named i-cNRL, which offers interpretability in the learned results, allowing for understanding which specific patterns are only found in one network. We demonstrate the effectiveness of i-cNRL for network comparison with multiple network models and real-world datasets. Furthermore, we compare i-cNRL and other potential cNRL algorithm designs through quantitative and qualitative evaluations.

Keywords: Machine learning, contrastive learning, network analysis, explainable AI, visualization.

1. Introduction

Networks are commonly used to model various types of relationships in real-world applications, such as social networks (Crnovrsanin et al. 2014), cellular networks (Chen and Sharp 2004), and communication networks (Bhanot et al. 2005). Comparative analysis of networks is an essential task in practice, where we want to identify differentiating factors between two networks or the uniqueness of one network compared to another (Emmert-Streib et al. 2016; Tantardini et al. 2019). For instance, when a neuroscientist is studying the effect of Alzheimer's disease on a human brain (Gaiteri et al. 2016), they want to compare the brain network of a patient with Alzheimer's disease to that of a healthy subject. Also, for collaboration networks of researchers in different fields (Larivière et al. 2006), an analyst in a funding agency may want to discover any unique ways of collaborations in the fields for decision making.

Several approaches have been proposed for network comparison (Tantardini et al. 2019). When two networks have the same node-set and the pairwise correspondence between nodes is known, we can compute a similarity between two networks (e.g., the Euclidean distance between two adjacency matrices). When the node-correspondence is unknown or does not exist, a network-statistics based approach is commonly used (e.g., the clustering coefficient, network diameter, or node degree distribution). Another popular approach is using graphlets (Pržulj 2007)—small, connected, and non-isomorphic subgraph patterns in a graph (e.g., the complete graph of three nodes). The similarities of two networks can be characterized by comparing the frequency of appearance of each graphlet in each network (Kwon et al. 2018).

While the existing approaches can provide a (dis)similarity between networks, they compare networks only based on one selected measure (e.g., node degree), which is often insufficient. Also, these approaches only provide network-level similarities, and thus cannot compare networks in more detailed levels (e.g., a node-level). Without such a detailed-level comparison, it is difficult to find which part of a network relates to its uniqueness.

To address these challenges, we introduce a new approach that integrates the concept of contrastive learning (Zou et al. 2013; Abid et al. 2018) together with network representation learning (NRL), which we call *cNRL*. Within cNRL, NRL enables the characterization of networks with comprehensive measures without overwhelming a user with information by embedding nodes into a low-dimensional space; contrastive learning allows for discovering unique patterns in one dataset relative to another[1] (Abid et al. 2018). By leveraging the benefits of both, we can reveal unique patterns in one

[1]There are two different machine learning schemes both called contrastive learning. One aims to learn an embedding from similar and dissimilar pairs of input samples so that similar samples are placed close together in the embedding space, and vice versa (Le-Khac et al. 2020; Jaiswal et al. 2021). Contrastive learning in our work refers to the other scheme introduced by Zou et al. (2013), where the learning purpose is finding unique/salient factors in one group of samples compared to another group.

network by contrasting with another, in a thorough (i.e., using multiple essential measures to capture the network characteristics) and detailed (i.e., analyzing a node or subnetwork level) manner.

With our approach, we consider the generality and interpretability of cNRL, and introduce a method called *i-cNRL*. First, i-cNRL is designed not to require node-correspondences or network alignment (Emmert-Streib et al. 2016), and thus is applicable to various networks. Also, unlike many other NRL methods, such as node2vec (Grover and Leskovec 2016) and graph neural networks (Zhang et al. 2020), i-cNRL offers interpretability (Adadi and Berrada 2018), providing information about the meaning of an identified pattern and the reason why that pattern can be seen in only that network.

In summary, our main contributions include:

- A new approach, called contrastive network representation learning (cNRL), which aims to reveal unique patterns in one network relative to another network.
- A method exemplifying cNRL, called i-cNRL, which (1) offers general applicability, including networks without node-correspondence or network alignment, (2) provides interpretability for helping understand revealed patterns, and (3) equips automatic hyperparameter selection for contrastive learning.
- Experiments using multiple network models and real-world datasets, which demonstrate the capability of comparative network analysis.
- Quantitative and qualitative comparisons with other potential designs of cNRL methods.

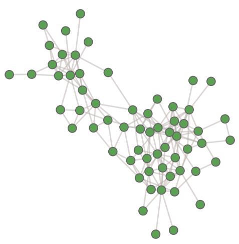
We provide Python implementations of cNRL and i-cNRL, datasets, and source code used for the evaluations at `https://takanori-fujiwara.github.io/s/cnrl/`.

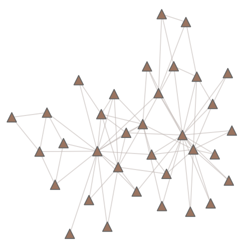
2. Problem Definition

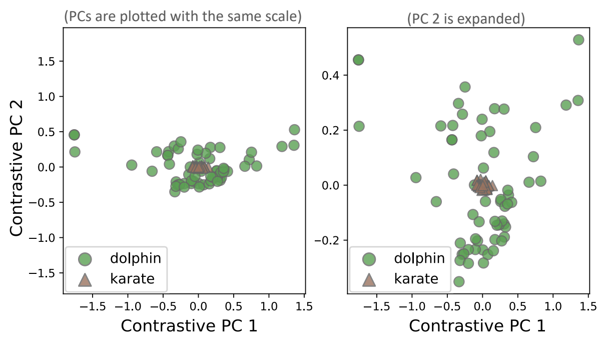
We here define the problem to be addressed by cNRL. Given two networks, a target network G_T and a background network G_B, we want to seek unique patterns in G_T relative to G_B. Similar to contrastive learning (Zou et al. 2013), the unique patterns can be represented as relationships (e.g., the structural differences among network nodes) that appear in G_T but do not appear in G_B.

For example, when finding unique patterns in a scale-free network G_T (i.e., its node-degree distribution follows a power law) relative to a random network G_B (i.e., each node pair is connected with a fixed probability) (Barabási 2016), we should be able to capture the unique patterns related to node degrees since G_T has more variety in node degrees. For practical usage, the unique patterns could relate to more complicated centralities, measures, combinations of them, and many more.

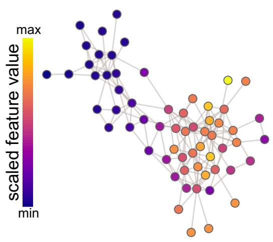
Note that, as with the existing work of contrastive learning (Zou et al. 2013; Abid et al. 2018), cNRL does not aim to discriminate graph elements (i.e., nodes or links) in G_T from G_B, but to identify unique patterns in G_T. For example, when comparing scale-free


(a) G_T: Dolphin

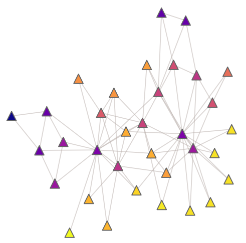

(b) G_B: Karate

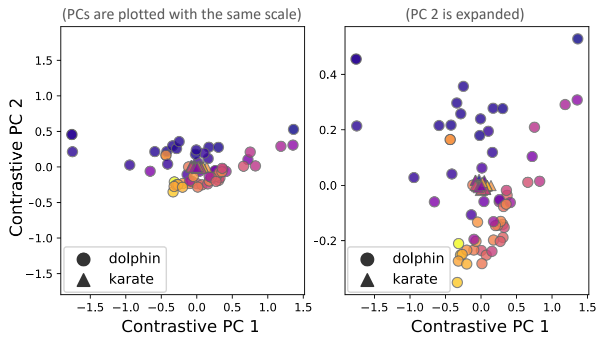


(c) The result of i-cNRL



(d) G_T: Dolphin


(e) G_B: Karate



(f) The result of i-cNRL

Figure 1: (a) and (b) show the dolphin social network and the Zachary's karate club network, used as G_T and G_B for i-cNRL, respectively. (c) shows the i-cNRL results with 2D embedding. (d), (e), and (f) colorcode each node in (a), (b), and (c) based on the top-contributed feature (`F1-10`) of the first contrastive principal component (`cPC 1`): $(\Phi_{\text{mean}})(\mathbf{x})$ with 'eigenvector' as the base feature $\mathbf{x}$ (see Table 2).

and random networks, the node degree should be able to highlight the aforementioned unique patterns; however, it cannot well distinguish nodes in G_T from ones in G_B as many nodes in G_T and G_B could have similar degrees.

3. Analysis Example

To provide an illustrative example of analysis with cNRL, we begin by comparing two social networks. We use the Dolphin social network (Lusseau et al. 2003) as G_T and the Zachary's karate club network (Zachary 1977) as G_B. Figure 1(a) and (b) depict the network structures of these networks. The statistics of these networks can be found in Table 1 (see `N1` and `N2`). By comparing these two networks, we want to reveal unique patterns in the Dolphin social network and identify which network characteristics relate to the patterns.

We apply our i-cNRL to the two networks and then plot a 2D embedding result with contrastive PCA (cPCA) (Abid et al. 2018), as shown in Figure 1(c). The x- and y-directions in Figure 1(c) represent the first and second contrastive principal components (cPCs), respectively. Details of i-cNRL and related techniques will be described in Section 5. Figure 1(c) shows that the nodes in G_T are more widely distributed, whereas the nodes in G_B are placed only around the center, which reveals some patterns specific

Table 1: Statistics of network datasets.

ID	Name	# of nodes	# of links	Directed
`N1`	Dolphin (Lusseau et al. 2003)	62	159	False
`N2`	Karate (Zachary 1977)	34	78	False
`N3`	Random	100	471	True
`N4`	Price	100	294	True
`N5`	p2p-Gnutella08 (Ripeanu et al. 2002)	6,301	20,777	True
`N6`	Price 2	6,301	18,897	True
`N7`	Enhanced Price	6,301	18,281	True
`N8`	Combined-AP/MS (Collins et al. 2007)	1,622	9,070	False
`N9`	LC-multiple (Reguly et al. 2006)	1,536	2,925	False
`N10`	School-Day1 (Stehlé et al. 2011)	236	5,899	False
`N11`	School-Day2 (Stehlé et al. 2011)	238	5,539	False

Table 2: Learned features and their cPC loadings for the Dolphin vs. Karate example.

ID	relational function f	base feature $\mathbf{x}$	`cPC 1`	`cPC 2`
`F1-1`	$(\mathbf{x})$	total-degree	0.01	-0.09
`F1-2`	$(\mathbf{x})$	betweenness	-0.02	-0.02
`F1-3`	$(\mathbf{x})$	closeness	0.01	0.01
`F1-4`	$(\mathbf{x})$	eigenvector	-0.11	0.00
`F1-5`	$(\mathbf{x})$	PageRank	0.11	0.18
`F1-6`	$(\mathbf{x})$	Katz	0.01	-0.08
`F1-7`	$(\Phi_{\text{mean}})(\mathbf{x})$	total-degree	-0.18	-0.42
`F1-8`	$(\Phi_{\text{mean}})(\mathbf{x})$	betweenness	0.15	-0.04
`F1-9`	$(\Phi_{\text{mean}})(\mathbf{x})$	closeness	-0.24	0.03
`F1-10`	$(\Phi_{\text{mean}})(\mathbf{x})$	eigenvector	**0.80**	0.10
`F1-11`	$(\Phi_{\text{mean}})(\mathbf{x})$	PageRank	-0.35	**0.76**
`F1-12`	$(\Phi_{\text{mean}})(\mathbf{x})$	Katz	-0.25	-0.43
`F1-13`	$(\Phi_{\text{max}})(\mathbf{x})$	PageRank	-0.02	0.00
`F1-14`	$(\Phi_{\text{mean}} \circ \Phi_{\text{mean}})(\mathbf{x})$	total-degree	-0.17	-0.01
`F1-15`	$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	PageRank	0.02	-0.00

to G_T when compared with G_B.

Moreover, since i-cNRL offers interpretability to the learned results, we can analyze why the above patterns appear. As shown in Table 2, the method provides loadings of each cPC (or cPC loadings), of which the absolute value indicates how large each learned feature contributes to each cPC direction. Each learned feature can be represented as a combination of the relational function f and the base feature $\mathbf{x}$ (see Section 5 for details). Table 2 indicates that feature `F1-10` has the highest contribution to `cPC 1`. From the relational function $(\Phi_{\text{mean}})(\mathbf{x})$ and the base feature 'eigenvector' (Newman 2018), this feature is interpreted as "the mean eigenvector centrality of the neighbors of a node."

To investigate the relationships between this feature and the i-cNRL result, we colorcode the network nodes in Figure 1(a), (b), and (c) based on the feature values, as shown in Figure 1(d), (e), and (f). Here, the feature values are scaled with the standardization when applying i-cNRL. We can see that, in Figure 1(f) (right), the nodes around the top-left corner tend to have smaller feature values while the nodes around the bottom-

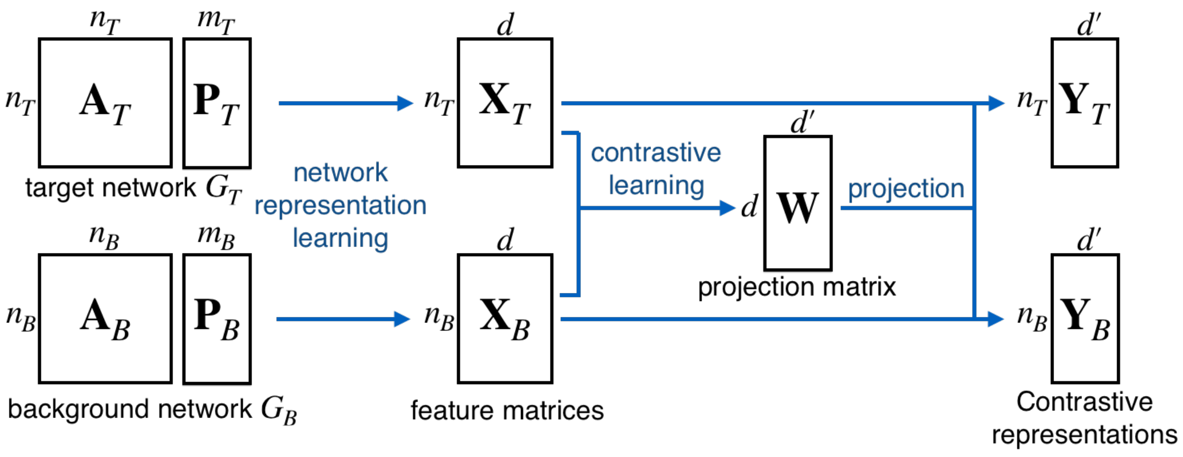


Figure 2: The general architecture for cNRL.

right tend to have higher values. By comparing with Figure 1(d), we notice that these two node groups correspond to the top-left and bottom-right communities in Figure 1(d). Since the feature value shows the mean eigenvector centrality of the neighbors of a node, the nodes in the top-left community tend to have a low eigenvector centrality including their neighbors. On the other hand, the nodes in the right-bottom community have neighbors with a high eigenvector centrality. Figure 1(e) indicates that G_B does not have such clearly separated communities by the feature values, unlike G_T. Therefore, i-cNRL learns the patterns highly related to the eigenvector centralities of each node's neighbors, which can clearly separate the two communities in the Dolphin social network.

4. cNRL Architecture

Figure 2 shows a general architecture for cNRL. Notations used for the following sections are listed in Table 3. The current contrastive learning methods (Zou et al. 2013; Abid et al. 2018; Dirie et al. 2019; Abid and Zou 2019; Severson et al. 2019) require target and background feature matrices ($\mathbf{X}_T$ and $\mathbf{X}_B$) sharing the same features as inputs. However, matrices that represent target and background networks (G_T and G_B) such as adjacency matrices ($\mathbf{A}_T$ and $\mathbf{A}_B$) might have a different number of nodes or no correspondence in nodes of $\mathbf{A}_T$ and $\mathbf{A}_B$. Thus, we cannot directly apply the contrastive learning methods to G_T and G_B. To address this issue, our core idea of cNRL consists of two main steps: (1) generating feature matrices $\mathbf{X}_T$ and $\mathbf{X}_B$ from networks G_T and G_B respectively by using NRL, and (2) applying contrastive learning on $\mathbf{X}_T$ and $\mathbf{X}_B$. Also, we want to emphasize that cNRL cannot be achieved by simply combining NRL and contrastive learning. For example, a method for NRL needs to satisfy a certain requirement to enable contrastive learning in the ensuing step. We identify such requirements for cNRL.

Below we describe the details of each part of the cNRL architecture with requirements on inputs, NRL, and contrastive learning methods. Here we focus only on node feature learning to provide a simple and clear explanation. However, the architecture is generic enough to be used for link (or edge) feature learning.

Table 3: Summary of notation.

Notations for cNRL	
G_T, G_B	target and background networks
$\mathbf{A}_T$, $\mathbf{A}_B$	adjacency matrices of G_T and G_B
$\mathbf{P}_T$, $\mathbf{P}_B$	matrices of node attributes of G_T and G_B
n_T, n_B	numbers of nodes in G_T and G_B
m_T, m_B	numbers of attributes in G_T and G_B
l_T, l_B	numbers of edges in G_T and G_B
d, d'	numbers of features learned by NRL and contrastive learning
$\mathbf{X}_T$, $\mathbf{X}_B$	target and background feature matrices
$\mathbf{W}$	projection matrix learned by contrastive learning
$\mathbf{Y}_T$, $\mathbf{Y}_B$	contrastive representations of $\mathbf{X}_T$ and $\mathbf{X}_B$
Notations for DeepGL	
$\mathbf{x}$	base feature (e.g., in-degree)
f	relational function
Φ^-, Φ^+, Φ	relational feature operators for in-, out-, total neighbors
S	summary measure (e.g., mean, sum, and maximum)
$\mathcal{F}_i$	set of learned features with i relational feature operators
$\mathcal{F}$	set of learned features: $\mathcal{F} = \{\mathcal{F}_0, \cdots, \mathcal{F}_h\}$
h	maximum numbers of relational feature operators to use
Notations for cPCA	
$\mathbf{C}_T$, $\mathbf{C}_B$	covariance matrices
α	contrast parameter

Inputs. cNRL takes G_T and G_B as inputs. These networks can be any combination of being undirected or directed, unweighted or weighted, and non-attributed or attributed. The numbers of G_T and G_B nodes (i.e., n_T and n_B) do not have to be the same. Similarly, the numbers of attributes m_T and m_B may be different.

Network representation learning. The first step in Figure 2 is applying an NRL method in order to transform the inputs G_T and G_B to feature matrices $\mathbf{X}_T$ and $\mathbf{X}_B$, respectively. Contrastive learning requires that $\mathbf{X}_T$ and $\mathbf{X}_B$ share the same features by nature of its learning purpose. Therefore, for this process, we need to use an NRL method that can produce the same features across networks.

Contrastive learning. Once we obtain $\mathbf{X}_T$ and $\mathbf{X}_B$, which have the same d learned features, we can apply any of the contrastive learning methods using $\mathbf{X}_T$ and $\mathbf{X}_B$ as target and background datasets, respectively. Contrastive learning generates a parametric mapping (or a projection matrix $\mathbf{W}$) from d features learned by NRL to d' contrastive features ($d' \leq d$). With this projection matrix, $\mathbf{X}_T$ and $\mathbf{X}_B$ can be transformed to contrastive representations $\mathbf{Y}_T$ and $\mathbf{Y}_B$, respectively. As the existing contrastive learning works (Zou et al. 2013; Abid et al. 2018; Dirie et al. 2019; Abid and Zou 2019; Severson et al. 2019) only produced $\mathbf{Y}_T$ for their analysis, the generation of $\mathbf{Y}_B$ is optional. However, as demonstrated in Figure 1(c), by visualizing both $\mathbf{Y}_T$ and $\mathbf{Y}_B$ in one plot, we can clearly see whether contrastive learning has found unique patterns in G_T relative to G_B. This visualization is a new approach to understanding contrastive learning results, where we can judge whether or not contrastive learning successfully finds target dataset's (in cNRL, target network's) unique patterns.

5. Interpretable cNRL Method

As a specific method using the architecture above, we describe i-cNRL, which employs DeepGL (Rossi et al. 2018) for NRL and cPCA (Abid et al. 2018) for contrastive learning. This algorithm selection is vital to provide interpretability; thus, we also provide the design rationale for the selection.

5.1. Network Representation Learning

As stated in Section 4, NRL needs to generate $\mathbf{X}_T$ and $\mathbf{X}_B$ that have the same features. To achieve this, we can employ any *inductive* NRL method (Rossi et al. 2018) (e.g., GraphSAGE by Hamilton et al. 2017 and FastGCN by Chen et al. 2018). However, we want to provide the interpretability in the contrastive representations obtained by cNRL; thus, an NRL method needs to generate interpretable features as the learned result. As a result, we specifically use DeepGL (Rossi et al. 2018) in the first step of i-cNRL.

DeepGL

DeepGL learns node and link features consisting of the *base feature* $\mathbf{x}$ and *relational function* f. For a concise explanation, we describe DeepGL for only node feature learning.

A base feature $\mathbf{x}$ is any simple feature or measure we can obtain for each node. For example, $\mathbf{x}$ can be (weighted) in-, out-, total-degree, degeneracy (or k-core numbers), PageRank (Newman 2018), or a node attribute (e.g., gender of a node in a social network).

A relational function f is a combination of *relational feature operators*, which is applied to a base feature. A relational feature operator summarizes base feature values of one-hop neighbors of a node. For example, the operator can be a computation of the mean, sum, maximum base feature values of one-hop neighbors' of a node. Also, the neighbors can be either in-, out-, total-neighbors. Together with the summary measure S (e.g., mean), the operators can be denoted Φ^-_S, Φ^+_S, and Φ_S, respectively. For example, $\Phi^-_{\text{mean}}(\mathbf{x})$ computes the mean $\mathbf{x}$ of the in-neighbors of a node. Moreover, the relational feature operator can be applied repeatedly. For example, $f = (\Phi^+_{\text{mean}} \circ \Phi^-_{\text{max}})(\mathbf{x})$ first computes the maximum $\mathbf{x}$ of in-neighbors for each out-neighbor of a node and then produces the mean of these maximum values. As described with the examples above, $\mathbf{x}$ and f are combinations of simple measures and operators; thus, both are interpretable.

In DeepGL, we can select as many different base features and relational feature operators as we want to consider. The learning process contains h number of iterations (indicated by the user), and in the end we obtain all the learned features $\mathcal{F} = \{\mathcal{F}_0, \mathcal{F}_1, \cdots, \mathcal{F}_h\}$, each of which is a relational function over a base feature $f(\mathbf{x})$. During each iteration, DeepGL prunes redundant features based on the similarities of the obtained feature values (refer to Rossi et al. 2018 for details). Table 2 shows an example of learned features from the Dolphin social network (Lusseau et al. 2003).

Use of Transfer Learning with DeepGL for cNRL

As described above, the learned features $\mathcal{F}$ by DeepGL are the combinations of the base features and relational functions. Once we obtain $\mathcal{F}$ from one network, we can naturally compute $\mathcal{F}$ for other networks. That is, DeepGL is inductive and can be used for transfer learning (Rossi et al. 2018).

In cNRL, we need to decide which network(s), G_T and/or G_B, should be used for learning $\mathcal{F}$. One possible choice is applying DeepGL for both to learn the features of target and background networks ($\mathcal{F}_T$ and $\mathcal{F}_B$, respectively). Then, we can use the union of these features (i.e., $\mathcal{F}_T \cup \mathcal{F}_B$) for producing feature matrices $\mathbf{X}_T$ and $\mathbf{X}_B$. However, since cNRL aims to identify unique patterns in G_T relative to G_B, such as patterns where only G_T has high variance (see Section 5.2), only a set of features capturing G_T's characteristics is required. Thus, we apply DeepGL to G_T and use the learned features $\mathcal{F}_T$ for both G_T and G_B to generate $\mathbf{X}_T$ and $\mathbf{X}_B$. It can also avoid unnecessary computation for learning $\mathcal{F}_B$ from G_B.

5.2. Contrastive Learning

The above NRL step generates feature matrices $\mathbf{X}_T$ and $\mathbf{X}_B$. The remaining step is learning contrastive representations $\mathbf{Y}_T$ and $\mathbf{Y}_B$ through contrastive learning. While we can use any contrastive learning method, one of our goals is to provide interpretability. Since DeepGL generates interpretable features for $\mathbf{X}_T$ and $\mathbf{X}_B$, we can provide interpretable $\mathbf{Y}_T$ and $\mathbf{Y}_B$ by using a method that reveals interpretable relationships between d features learned by NLR and d' features learned by contrastive learning. Among current contrastive learning methods (Zou et al. 2013; Abid et al. 2018; Dirie et al. 2019; Abid and Zou 2019; Severson et al. 2019), only cPCA or its variants (e.g., sparse cPCA (Boileau et al. 2020)) can provide such relationships by utilizing the linearity of its algorithm in a similar manner to ordinary PCA (Jolliffe 1986). Thus, we select cPCA for the second step of i-cNRL, though, it can be replaced with any other interpretable contrastive learning methods developed in the future.

Contrastive PCA (cPCA)

cPCA (Abid et al. 2018) is a variant of PCA for contrastive learning. Similar to the classical PCA, cPCA first applies centering to each feature of $\mathbf{X}_T$ and $\mathbf{X}_B$ and then obtains their corresponding covariance matrices $\mathbf{C}_T$ and $\mathbf{C}_B$. Let $\mathbf{v}$ be any unit vector of d length. Then, with a given direction $\mathbf{v}$, the variances for $\mathbf{X}_T$ and $\mathbf{X}_B$ can be written as: $\sigma_T^2(\mathbf{v}) \overset{\text{def}}{=} \mathbf{v}^\mathsf{T}\mathbf{C}_T\mathbf{v}$, $\sigma_B^2(\mathbf{v}) \overset{\text{def}}{=} \mathbf{v}^\mathsf{T}\mathbf{C}_B\mathbf{v}$. The optimization that finds a direction $\mathbf{v}^*$ where $\mathbf{X}_T$ has high variance but $\mathbf{X}_B$ has low variance can thus be written as:

$$\mathbf{v}^* = \underset{\mathbf{v}}{\operatorname{argmax}} \ \sigma_T^2(\mathbf{v}) - \alpha\sigma_B^2(\mathbf{v}) = \underset{\mathbf{v}}{\operatorname{argmax}} \ \mathbf{v}^\mathsf{T}(\mathbf{C}_T - \alpha\mathbf{C}_B)\mathbf{v} \tag{1}$$

where α is a contrast parameter ($0 \leq \alpha \leq \infty$). From Equation (1), we can see that $\mathbf{v}^*$ corresponds to the first eigenvector of the matrix $\mathbf{C} \overset{\text{def}}{=} (\mathbf{C}_T - \alpha\mathbf{C}_B)$. The eigenvectors of $\mathbf{C}$ can be calculated with eigenvalue decomposition. These computed eigenvectors are called contrastive principal components (cPCs) and are orthogonal to each other. Similar to the classical PCA, we can obtain top-d' cPCs as the learned features. With projection matrix $\mathbf{W}$ consisting of d' cPCs (i.e., $\mathbf{W}$ is a $d \times d'$ matrix), we can obtain the contrastive representation $\mathbf{Y}_T$ of $\mathbf{X}_T$.

The above contrast parameter α controls the trade-off between having high target variance and low background variance. When $\alpha = 0$, cPCs only maximize the variance

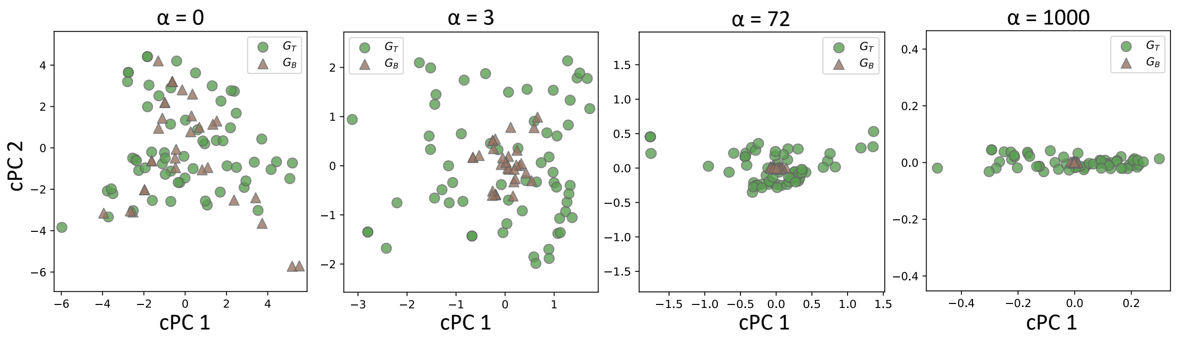


Figure 3: The cPCA results with different α values, applied on feature matrices $\mathbf{X}_T$ and $\mathbf{X}_B$ generated from G_T (the dolphin network) and G_B (the Karate network) in Figure 1. When $\alpha = 0$, the result is the same with applying PCA on $\mathbf{X}_T$. A decrease of $\mathbf{X}_B$'s variances is observed as α increases. The result with $\alpha = 72$ corresponds to the results in Figure 1.

of $\mathbf{X}_T$, the same as those in the classical PCA. As α increases, cPCs place greater emphasis on directions that reduce the variance of $\mathbf{X}_B$. Figure 3 shows the results of cPCA with different α values. Because α has a strong impact on the result, Abid et al. (2018) introduced the semi-automatic selection of α utilizing spectral clustering (Ng et al. 2002). We go one step further to provide a fully automatic selection of α (see Automatic Contrast Parameter Selection explained below).

Representation Learning with cPCA in cNRL

By applying cPCA to $\mathbf{X}_T$ and $\mathbf{X}_B$, we can generate the projection matrix $\mathbf{W}$ and contrastive representations $\mathbf{Y}_T$ and $\mathbf{Y}_B$. Because each learned feature by DeepGL could have a different scale, as a default, our method applies the standardization (i.e., scaling each feature to have zero mean and unit variance). Also, there could be a scale difference between $\mathbf{X}_T$ and $\mathbf{X}_B$ (e.g., G_T and G_B's mean total-degrees are 1,000 and 100, respectively). Thus, instead of a concatenated matrix of $\mathbf{X}_T$ and $\mathbf{X}_B$, we apply the standardization to each of $\mathbf{X}_T$ and $\mathbf{X}_B$ for both learning and projection. This approach can avoid generating a 2D embedding result where G_T's nodes are placed extremely far away from G_B's nodes. In addition, the selection of α becomes easier. For example, without the above standardization, $\mathbf{X}_T$ could have a much smaller variance in any direction $\mathbf{v}$ when compared to $\mathbf{X}_B$; consequently, to find unique patterns, we may need to set excessively small α (e.g., $\alpha = 0.0001$), and vice versa. On the other hand, when the comparison of the absolute value differences between $\mathbf{X}_T$ and $\mathbf{X}_B$ is important, we can apply the standardization to the concatenated matrix.

To provide interpretable relationships between NLR features d and contrastive learning features d', we extract contrastive PC loadings (cPC loadings), which are coefficients of $\mathbf{W}$[2] These cPC loadings indicate how strongly each of the d input features contributes to the corresponding cPC. Table 2 shows an example of cPC loadings for the first and

[2]Similar to ordinary PCA, instead of loadings, we could also refer to loadings scaled with a square root of the corresponding eigenvalue. In ordinary PCA, the scaled loadings show the correlations between each feature and the corresponding PC. However, cPCA's scaled loadings do not show such correlations as cPCA performs eigenvalue decomposition on $(\mathbf{C}_T - \alpha\mathbf{C}_B)$, instead of $\mathbf{C}_T$ or $\mathbf{C}_B$.

second cPCs. As demonstrated in Section 3, by referring to a list of the learned features via NRL and cPC loadings, we can interpret the obtained representations $\mathbf{Y}_T$ and $\mathbf{Y}_B$. Also, as discussed in Section 4, we visualize both $\mathbf{Y}_T$ and $\mathbf{Y}_B$ in one plot to judge whether or not i-cNRL has found unique patterns in G_T. cPCA contrasts variances of $\mathbf{Y}_T$ and $\mathbf{Y}_B$; thus, when G_T has unique patterns, $\mathbf{Y}_T$ has a much higher variance than $\mathbf{Y}_B$ (e.g., when $\alpha = 72$ and $\alpha = 1000$ in Figure 3). Note that, in such a case, G_B's nodes tend to be highly overlapped with each other in a visualized result, and it becomes difficult to see the difference of each of G_B's nodes. However, this is preferable for analyses using i-cNRL as the visualization of the embedding result should highlight unique patterns in G_T but not the differences within G_B's nodes.

Automatic Contrast Parameter Selection

We now show how to automatically select the parameter α in cPCA. Since we want to maximize the variation in the target feature matrix while simultaneously minimizing the variation in the background feature matrix, we can solve the following ratio problem:

$$\max_{\mathbf{W}^\top \mathbf{W} = I_{d'}} \frac{\mathrm{tr}(\mathbf{W}^\top \mathbf{C}_T \mathbf{W})}{\mathrm{tr}(\mathbf{W}^\top \mathbf{C}_B \mathbf{W})}. \tag{2}$$

While directly solving Equation (2) may be difficult, there is a convenient iterative algorithm due to Dinkelbach (1967). The algorithm consists of two steps. Given $\mathbf{W}_t$, we perform

- $\alpha_t \leftarrow \dfrac{\mathrm{tr}(\mathbf{W}_t^\top \mathbf{C}_T \mathbf{W}_t)}{\mathrm{tr}(\mathbf{W}_t^\top \mathbf{C}_B \mathbf{W}_t)}$
- $\mathbf{W}_{t+1} \leftarrow \operatorname*{argmax}_{\mathbf{W}^\top \mathbf{W} = I_{d'}} \mathrm{tr}(\mathbf{W}^\top (\mathbf{C}_T - \alpha_t \mathbf{C}_B) \mathbf{W})$.

Clearly, α_t is just the objective value of our ratio problem, Equation (2), evaluated at the current solution $\mathbf{W}_t$. α_t monotonically increases to the maximum value, and the convergence is usually very quick (e.g., less than 10 iterations, as evaluated in Section C.5). Conveniently, the second step for finding the next solution $\mathbf{W}_{t+1}$ is just the original cPCA problem, where we use α_t as our trade-off parameter. We can also regard cPCA as a one-shot algorithm for the ratio problem, Equation (2), where the user specifies α. One problem of the method above is that α_t reaches close to infinite when $\mathbf{C}_B$ is nearly singular. In this case, the algorithm finds an embedding where $\mathbf{C}_T$ has nonzero variance while $\mathbf{C}_B$ has zero variance. To avoid this problem, our method simply adds a small constant value ϵ, as a default $\epsilon = 10^{-3}$, to each diagonal element of $\mathbf{C}_B$. As discussed, by default, $\mathbf{X}_B$ is standardized and each diagonal element $\mathbf{C}_B$ is close to one; thus, $\epsilon = 10^{-3}$ is reasonably small to avoid introducing a strong bias. We note that the above algorithm of Dinkelbach (1967) has been used in discriminant analysis (Guo et al. 2003; Jia et al. 2009), whose motivation is different from ours.

5.3. Complexity Analysis

The time and space complexities of i-cNRL are comparable to those of DeepGL and cPCA. According to Rossi et al. (2018), DeepGL's time and space complexities for learning from G_T are $\mathcal{O}(d(l_T + dn_T))$ and $\mathcal{O}(dn_T)$, respectively, where l_T is the number of links in G_T. Note that the time and space complexities for computing base features are assumed lower than these. When including the transfer learning step to obtain $\mathbf{X}_B$, the space complexity becomes $\mathcal{O}(d(n_T + n_B))$. For a fixed α, cPCA has the similar time and space complexities with PCA, which are $\mathcal{O}(d^2(n_T + n_B) + d^3))$ and $\mathcal{O}(d^2)$. Even with the automatic selection of α in Section 5.2, we can assume that these complexities stay the same. This is because the automatic selection usually only needs a small number of iterations and does not require storing of additional information. Thus, in total, i-cNRL has the time complexity $\mathcal{O}(d(l_T + d(n_T + n_B) + d^2))$ and the space complexity $\mathcal{O}(d(n_T + n_B + d))$. However, in practice, d, the number of features learned by NRL, should be much smaller than the numbers of nodes and links of G_T and G_B. Under this assumption, the time and space complexities are $\mathcal{O}(d(l_T + d(n_T + n_B)))$ and $\mathcal{O}(d(n_T + n_B))$, respectively. This indicates that the computational cost is largely due to DeepGL.

The above fact has guided our finer level design choices of i-cNRL. To reduce the computational cost of DeepGL, we utilize DeepGL's feature pruning, instead of keeping all features for cPCA. However, if fast computation is not required, we could skip feature pruning and apply sparse cPCA (Boileau et al. 2020). Sparse cPCA uses a limited number of features when constructing cPCs (i.e., $\mathbf{W}$ becomes a sparse matrix), and thus helps produce interpretable results from many features. Also, we could choose another design alternative to more tightly connect cPCA with feature pruning in DeepGL. For example, instead of the similarities of features, we can prune features by checking whether or not an inclusion of the corresponding feature highly influences the optimization result with Equation (1). However, this design also requires more computations as the pruning needs to derive feature values for both G_T and G_B and apply cPCA repeatedly during the learning process of DeepGL.

To provide better computational scalability, DeepGL can utilize parallelization. When learning $\mathcal{F}_i$ (i.e., features with i relational feature operators), we can individually apply relational feature operators to each graph element (e.g., graph node) and/or each base feature. For details, refer to the evaluation performed by Rossi et al. (2018).

6. Related Work

To the best of our knowledge, our work is the first to introduce the use of contrastive learning for networks and provide a general and interpretable method under this approach. There exists little work in the exact area. Thus, we here review typical NRL and contrastive learning techniques.

6.1. Network Representation Learning (NRL)

Various NRL methods have been developed for learning latent representations of network nodes and/or links. For a comprehensive description of NRL methods, refer to the recent survey papers, such as Cai et al. (2018) and Zhang et al. (2020). Here we focus on describing the closely related work using inductive and cross-network embedding methods.

Inductive NRL

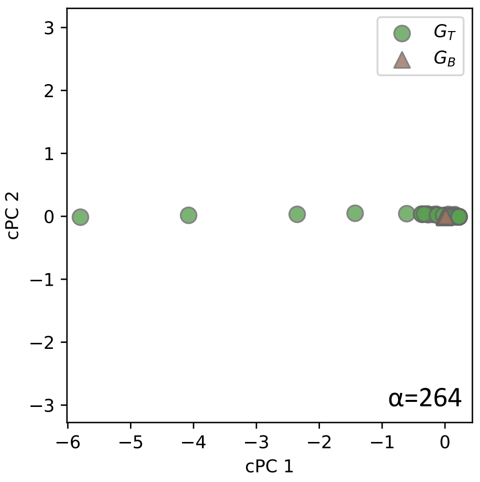
GraphSAGE (Hamilton et al. 2017) is an inductive NRL method that shares many similar ideas with DeepGL (Rossi et al. 2018). Analogous to the relational functions f in DeepGL, GraphSAGE learns *aggregator functions*. However, GraphSAGE proposes more complex aggregators using LSTM and max-pooling concepts, compared to DeepGL's simple aggregators (e.g., mean). Moreover, GraphSAGE tunes parameters required by the aggregators and matrices that decide the weight for each learned feature, instead of the feature pruning in DeepGL. These differences might enable GraphSAGE to better capture complex characteristics of networks without manual parameter tuning; however, the learned features might be difficult to interpret. FastGCN (Chen et al. 2018) takes a similar approach to GraphSAGE except that FastGCN employs node sampling to save memory space. Also, HetGNN (Zhang et al. 2019) enhances the aggregators to learn representations of heterogeneous networks. These methods, including other variants of graph neural networks (Zhang et al. 2020) (e.g., GAT by Veličković et al. 2018, h/cGAO by Gao and Ji 2019, and InfoGraph by Sun et al. 2020), still suffer from lack of interpretability in the learned features. Although GNNExplainer (Ying et al. 2019) aims to provide interpretable explanations for predictions made by these methods, it does not support explaining the learned features themselves.

Cross-Network Embedding

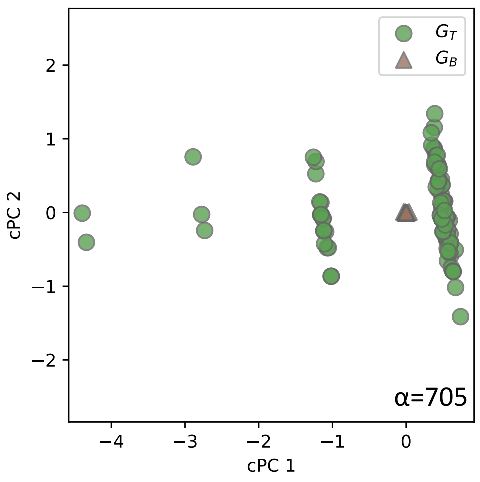
The inductive methods learn the features that can be generalized for unobserved nodes or other networks from one input network. On the contrary, the cross-network methods generate embeddings directly from multiple input networks. Most of the cross-network methods focus on finding similarities of nodes across networks, such as for node classification (Shen et al. 2021), network similarity calculation (Ma et al. 2019), and network alignment (Heimann et al. 2018). While CrossMVA by Chu et al. (2019) is developed mainly for network alignment, it can produce embeddings that contain both similarity and dissimilarity information. However, a major drawback of CrossMVA is that anchor nodes are necessary as inputs (i.e., at least we need to know a small portion of node-correspondence), which we cannot obtain in many cases (e.g., the example in Section 3). Also, CrossMVA's embeddings of the dissimilarity information only preserve discriminative structures across networks; as a result, it cannot find unique patterns in a specific network.

6.2. Contrastive Learning

Unlike discriminant analysis, such as linear discriminant analysis (Jia et al. 2009), which aims to discriminate samples based on their classes, contrastive learning (Zou et al. 2013) focuses on finding salient patterns in one dataset compared to another. Several machine learning methods have been extended for contrastive learning. For example, there are contrastive versions of latent Dirichlet allocation (Zou et al. 2013), hidden Markov models (Zou et al. 2013), and regressions (Ge and Zou 2016). More recently, including cPCA (Abid et al. 2018), contrastive learning methods for representation learning have been introduced (Abid et al. 2018; Dirie et al. 2019; Abid and Zou 2019; Severson et al. 2019; Fujiwara et al. 2020; Fujiwara and Liu 2020; Zhang et al. 2021; Fujiwara et al. 2022). For example, Dirie et al. (2019) introduced contrastive multivari-



(a) G_T: Price, G_B: Random



(b) G_T: Random, G_B: Price

Figure 4: Results for Section 7.1 with 2D embeddings by i-cNRL.

ate singular spectrum analysis (cMSSA) for decomposition of time-series data. Similar to cPCA, cMSSA could provide the interpretability by computing the PC loadings; however, cMSSA is not suitable for our case where we handle non-time series data. On the other hand, contrastive variational autoencoder (cVAE) (Abid and Zou 2019; Severson et al. 2019) can be used as a contrastive learning method in cNRL. The strength of cVAE over cPCA is that it can find unique patterns in a target dataset even when its samples and latent features have nonlinear relationships. However, cVAE relies on multiple layers of neural networks, and thus the results of cVAE are difficult to interpret as similar to other neural-network-based methods. Therefore, to use cVAE for interpretable cNRL, we need additional effort to help interpret the results.

7. Experimental Evaluation

In the previous sections, we have introduced the concepts of cNRL and i-cNRL, as well as the related work. We have also demonstrated the effectiveness of i-cNRL in comparing social networks in Section 3. To further evaluate the method, we first test i-cNRL with synthetic datasets that are generated with popular network models. Then, we demonstrate several analysis examples using i-cNRL with publicly available real-world datasets (see Table 1). Lastly, we provide quantitative and qualitative comparisons among i-cNRL and other potential cNRL implementations. In each subsection, we list only the information closely related to our findings. Throughout the evaluation, we focus on analyzing the first two cPCs in order to provide 2D embedding visualizations. Also, similar to PCA, the first two cPCs are usually more important than other cPCs. Details of learning parameters and results are provided in Appendix C.

7.1. Evaluation with Network Models

We apply i-cNRL to compare two types of synthetic networks: *random* and *scale-free* networks (`N3` and `N4` in Table 1). We generate the random and scale-free networks with

Table 4: The features with the top-3 absolute loadings for `cPC 1` for different pairs of networks highlighted in gray.

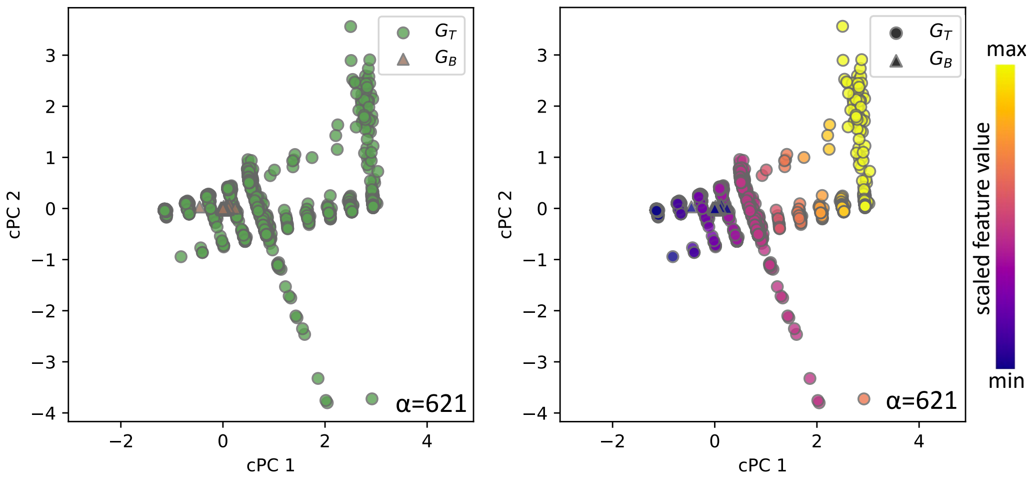
ID	relational function f	base feature $\mathbf{x}$	`cPC 1`	`cPC 2`
G_T: Price, G_B: Random (Section 7.1)				
`F2-1`	$(\mathbf{x})$	total-degree	-0.79	-0.01
`F2-1`	$(\mathbf{x})$	out-degree	0.52	0.00
`F2-3`	$(\mathbf{x})$	Katz	0.24	-0.71
G_T: Random, G_B: Price (Section 7.1)				
`F3-1`	$(\mathbf{x})$	k-core	0.99	0.13
`F3-2`	$(\mathbf{x})$	total-degree	0.11	-0.79
`F3-3`	$(\mathbf{x})$	in-degree	-0.06	0.43
G_T: p2p-Gnutella08 , G_B: Price 2 (Case Study 1)				
`F4-1`	$(\mathbf{x})$	k-core	0.97	0.25
`F4-2`	$(\mathbf{x})$	total-degree	0.20	-0.79
`F4-3`	$(\mathbf{x})$	in-degree	-0.12	0.43
G_T: p2p-Gnutella08, G_B: Enhanced Price (Case Study 1)				
`F5-1`	$(\mathbf{x})$	total-degree	-0.82	0.03
`F5-2`	$(\mathbf{x})$	in-degree	0.43	0.67
`F5-3`	$(\mathbf{x})$	Katz	0.35	-0.74
G_T: LC-multiple , G_B: Combined-AP/MS (Case Study 2)				
`F6-1`	$(\Phi_{\text{mean}})(\mathbf{x})$	Katz	0.83	0.01
`F6-2`	$(\Phi_{\text{mean}})(\mathbf{x})$	eigenvector	-0.44	-0.03
`F6-3`	$(\Phi_{\text{mean}})(\mathbf{x})$	total-degree	-0.33	0.06
G_T: School-Day2 , G_B: School-Day1 (Case Study 3)				
`F7-1`	$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	PageRank	-0.53	-0.21
`F7-2`	$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	closeness	-0.40	0.33
`F7-3`	$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	betweenness	0.33	0.09

the Gilbert's random graph (Barabási 2016) and the Price's preferential attachment models (Newman 2018), respectively. We produce two 2D embedding results, using one network as G_T and the other as G_B, as shown in Figure 4(a) and (b). Each of the results shows unique patterns in G_T. The cPC loadings in Table 4 show that the Price network's unique patterns are related to the degree centralities (e.g., total-degree). This seems to be due to the fact that most nodes have the same number of links in a random network while a scale-free network contains hubs with a large number of links. In contrast, we can see that the random network's uniqueness is mostly related to k-core numbers. This is because the Price's model generates a network by adding a new node and then connecting it to other fixed number of nodes (e.g., 3 nodes) which are selected with a certain computed probability. As a result, all nodes in the network have the same k-core numbers (e.g., 3-core).

7.2. Case Studies

Case Study 1: Network Model Refinement

Designing a network model that can simulate real-world networks is fundamental to



(a) G_T: p2p-Gnutella08, G_B: Price 2 (b) G_T: p2p-Gnutella08, G_B: Price 2

Figure 5: Results for Case Study 1. (a) presents the 2D embedding by i-cNRL. (b) shows the nodes in (a) colored by the k-core number (`F4-1` in Table 4).

understand network formation mechanisms, to perform hypothetical analyses (e.g., if there is a growth of the number of nodes, what will happen?), to generate more available datasets for machine learning, and more (Goldenberg et al. 2010). This case study demonstrates the usage of i-cNRL to guide a refinement of network models.

Here, we use a peer-to-peer (P2P) network, specifically the Gnutella peer-to-peer file sharing network (Ripeanu et al. 2002; Leskovec et al. 2007) available in SNAP Datasets (Leskovec and Krevl 2014) (`N5` in Table 1) as a modeling subject. Once we have a P2P network generation model, we can use it for analyzing network robustness, studying effective searching strategies on a P2P network, etc (Liu et al. 2009).

P2P networks are often scale-free (Liu et al. 2009), so we use the Price's model (Newman 2018) to mimic a P2P network. To identify the characteristics that the Price's model does not simulate well, we set the P2P network (`N5`) as G_T and the Price network (`N6`) as G_B.

The result is shown in Figure 5(a). From the cPC loadings in Table 4, we notice that the k-core number (`F4-1`) has a strong contribution to `cPC 1`. Thus, we colorcode the result based on the k-core number, as shown in Figure 5(b). We can clearly see that the P2P network has variations in the k-core number, but the Price network does not. Because the k-core number indicates that a node at least connects to other k nodes, the Price network makes a significant difference in the network robustness from the P2P network.

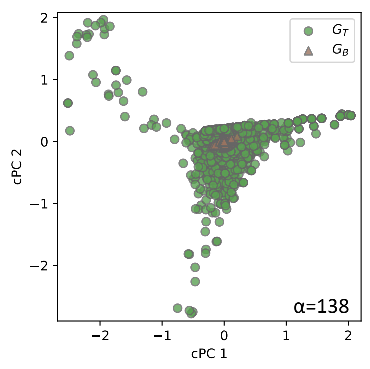
From the result above, we decide to refine the Price model to generate various k-core numbers. As discussed in Section 7.1, the problem comes from the fact that the Price's model always adds a new node with a fixed number of links. Similar to the dual-Barabási-Albert model by Moshiri (2018), we can avoid the problem by attaching a new node to a variable number of links according to a probability distribution. Specifically, we design an enhanced version of Price's model to select the number of links from 1 to 10 with specified probabilities (for details, refer to Section C.3). Then, we generate a network with this model, which is referred to as the Enhanced Price (`N7`) network in Table 1. Next, we apply i-cNRL to the P2P (as G_T) and Enhanced Price (as G_B) networks. The resultant cPC loadings are listed in Table 4. While G_T seems to still have the uniqueness in degree centralities, it does not in the k-core number. By iteratively performing refinement procedures such as the one above, we can build a better network model to simulate real-world networks.

Case Study 2: Comparison of Two Networks

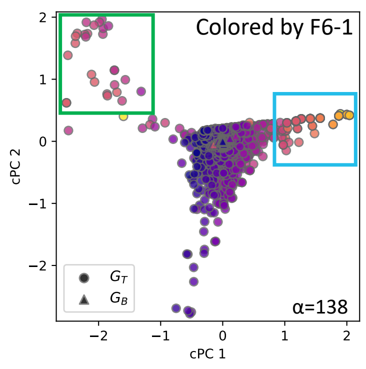
In this case study, we compare "interactome" networks—networks of physical DNA-, RNA-, and protein-protein interactions (Yu et al. 2008). Specifically, we compare two interactome networks, Combined-AP/MS (`N8` in Table 1) and LC-multiple (`N9`), available in CCSB Interactome Database (Stanford Center for Cancer Systems Biology 2003). Both networks represent the interactome of the yeast *S. cerevisiae*; however, they are obtained through different analysis approaches. Combined-AP/MS is generated from two studies using a "high-throughput" approach, specifically, affinity purification/mass spectrometry (AP/MS) (Collins et al. 2007). In contrast, LC-multiple is the literature-curated (LC) network from multiple "low-throughput" experiments (Yu et al. 2008; Reguly et al. 2006). Because each analysis approach has its own strength in identifying the yeast's interactions, the generated networks may vary (Yu et al. 2008). Comparing these networks is essential to understand the quality and characteristics of each approach (Yu et al. 2008).

Here we analyze the uniqueness in LC-multiple by using LC-multiple and Combined-AP/MS as G_T and G_B, respectively. The 2D embedding result by i-cNRL is shown in Figure 6(a). We first notice that, in G_T, there are two distinct regions: one spreading out towards the top-left and the other in the bottom-right quadrant. To understand why this pattern appears, we obtain the cPC loadings (Table 4) and colorcode the nodes based on values of the feature that has the top cPC loading for `cPC 1` (i.e., `F6-1`, f: $(\Phi_{\text{mean}})(\mathbf{x})$ and $\mathbf{x}$: the Katz centrality). The result is shown in Figure 6(b). We observe that either going to the left or right side along `cPC 1` tends to produce a high value of this feature, as annotated with the green and teal rectangles, respectively. While this feature has a strong positive loading for `cPC 1`, another feature in Table 4—`F6-2`, f: $(\Phi_{\text{mean}})(\mathbf{x})$ and $\mathbf{x}$: the eigenvector centrality—has a strong negative loading. Therefore, if a node has a higher value for `F6-2`, it tends to be placed on the more left side in Figure 6(b). This indicates that the green rectangle region in Figure 6(b) seems to have high values for both of these features while the teal region has low values for the latter feature (`F6-2`). This could happen because the eigenvector centrality tends to be low when a node is in a weakly connected region while the Katz centrality is high whenever a node is linked by many others (Newman 2018).

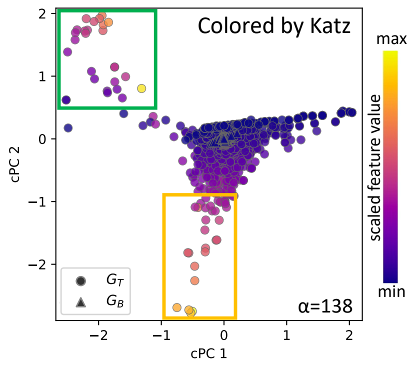
To visually observe the above patterns, we draw the network structures of G_T and G_B with scalable force directed placement (Hu 2005) and then color them based on the values of `F6-1`, as shown in Figure 6(d) and (e). We here only show the largest component of each network (i.e., the nodes connected with only several nodes are filtered out). Figure 6(e) shows that one strongly connected region around the center contains all nodes with high feature values. On the other hand, in Figure 6(d), multiple regions contain nodes with high feature values. To further investigate this pattern, we select the nodes corresponding to the green and teal regions in Figure 6(b) and then highlight these nodes in Figure 6(d). Afterward, we zoom into the related regions of



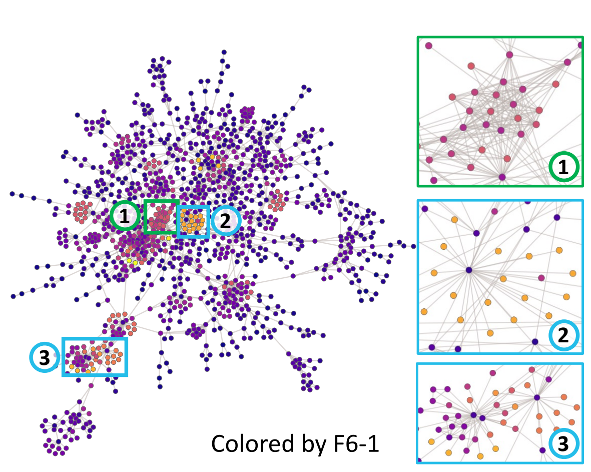
(a) G_T: LC-multiple, G_B: Combined-AP/MS



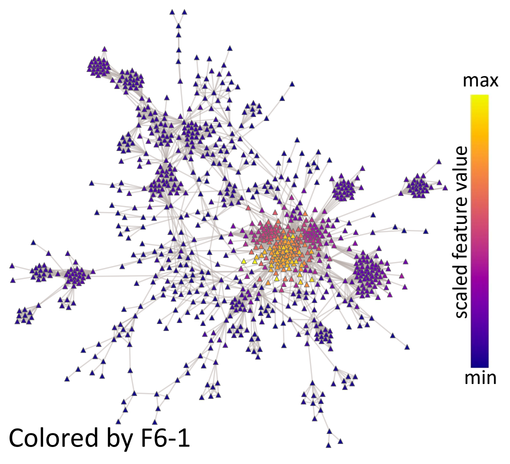
(b) G_T: LC-multiple, G_B: Combined-AP/MS



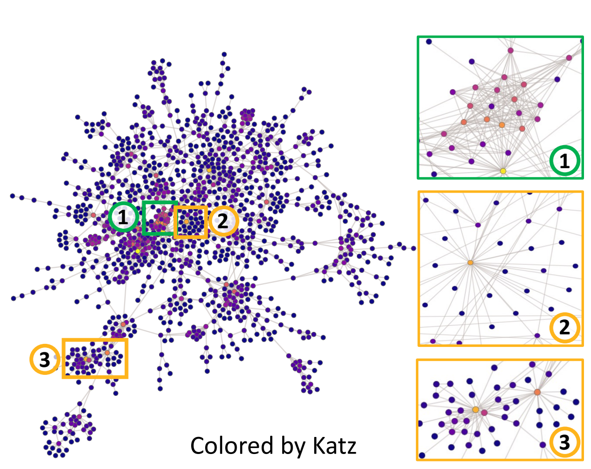
(c) G_T: LC-multiple, G_B: Combined-AP/MS



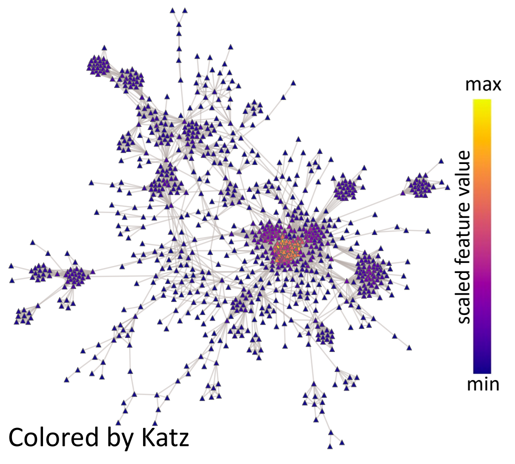
(d) G_T: LC-multiple



(e) G_B: Combined-AP/MS



(f) G_T: LC-multiple



(g) G_B: Combined-AP/MS

Figure 6: Results for Case Study 2. (a) presents the 2D embedding by i-cNRL. (b) shows the nodes in (a) colored by the feature—f: $(\Phi_{\text{mean}})(\mathbf{x})$, $\mathbf{x}$: the Katz centrality (`F6-1` in Table 4). (c) is colored by the Katz centrality. (d–g) show the network structures with the same colorcoding (d and e: `F6-1`, f and g: Katz).

Table 5: The features with the top-3 absolute loadings for `cPC 2` for Case Study 2.

ID	relational function f	base feature $\mathbf{x}$	`cPC 1`	`cPC 2`
`F6-4`	$(\mathbf{x})$	Katz	0.00	-0.81
`F6-5`	$(\mathbf{x})$	total-degree	-0.05	0.42
`F6-6`	$(\mathbf{x})$	eigenvector	0.03	0.41

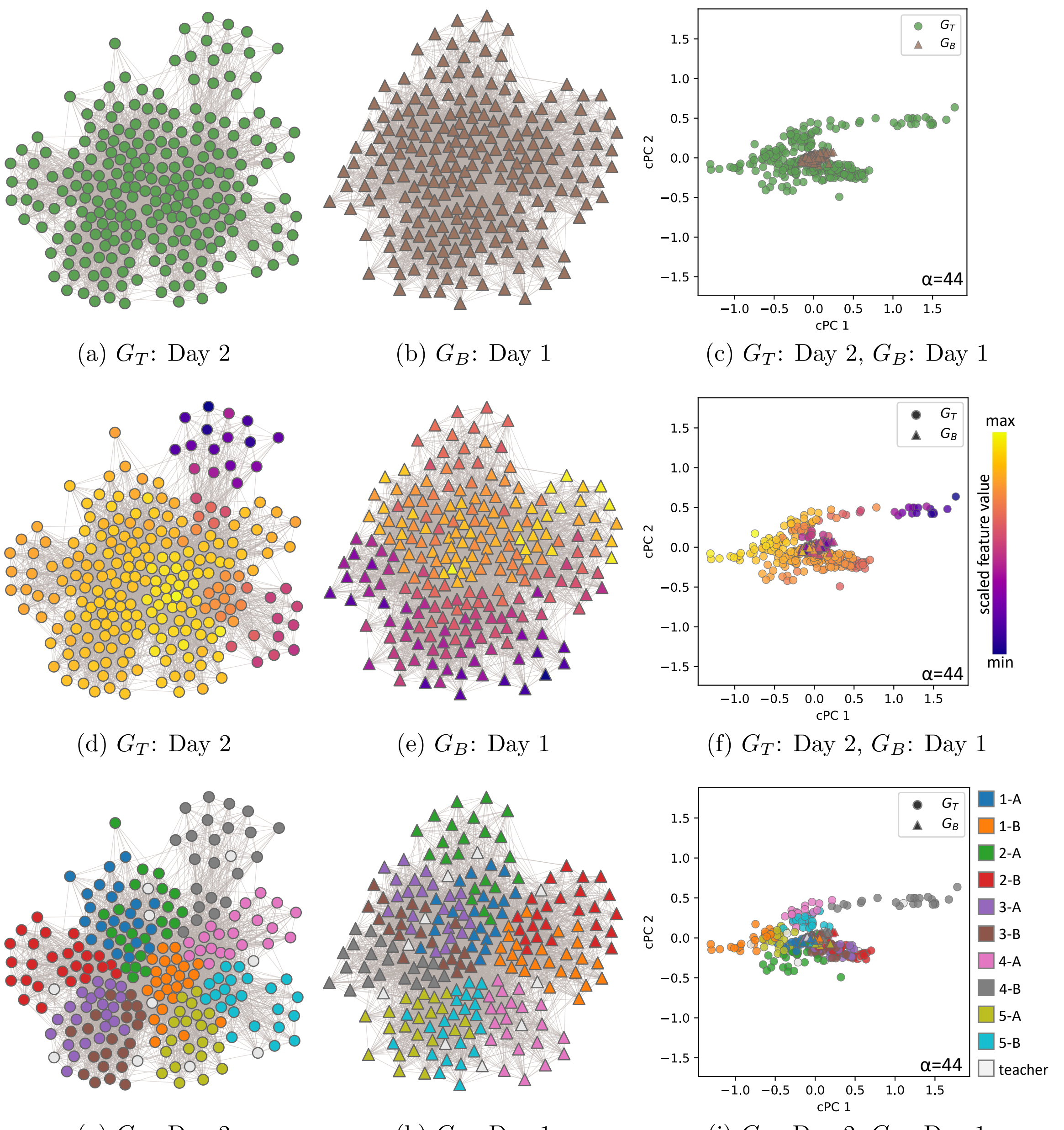
the highlighted nodes. Figure 6(d)-① shows a region related to the nodes in the green rectangle, while Figure 6(d)-② and ③ are two example regions related to the teal rectangle region. We can see that the nodes in Figure 6(d)-① are strongly connected, but not in Figure 6(d)-② and ③. From these observations, i-cNRL reveals that only G_T has two different types of nodes linked to the high Katz centrality node(s) in either strongly or weakly connected region.

Similarly, we further interpret `cPC 2`. As shown in Table 5, the base features of the Katz, total-degree, and eigenvector centralities strongly contribute to `cPC 2`. From the 2D embedding result in Figure 6(c), which is colorcoded based on the Katz centrality, we can see that the Katz centrality also shows high values in two regions, as annotated with the green and orange rectangles. As with the analysis of `cPC 1`, the Katz and eigenvector centralities have strong negative and positive loadings to `cPC 2`, respectively. To investigate the two regions, we generate visualizations corresponding to the analysis of `cPC 1` (Figure 6(f) and (g)). From these visualizations, we can see that, in the orange rectangle area of Figure 6(c), `cPC 2` seems to capture central nodes in the aforementioned weekly connected region. In summary, i-cNRL highlights the three different types of nodes unique in the target network—(1) nodes in the strongly connected regions, (2) central nodes, and (3) surrounding nodes in the weakly connected regions.

Case Study 3: Analysis of Network Changes

As an example of analyzing dynamic networks, we compare two different days of contact networks in a primary school (Stehlé et al. 2011; The SocioPatterns Collaboration 2008). The networks represent face-to-face contact patterns between students and teachers, which are collected with RFID devices. Information of the network at each day is listed in Table 1 (`N10` and `N11`). Figure 7(a) and (b) visualize the network structures drawn with scalable force directed placement. Also, these networks have multiple node attributes including genders, grades, and class names. In addition to multiple network centralities, we utilize the attribute information by including gender as the base feature, i.e., encoding 'male', 'female', and 'unknown' as -1, 1, and 0, respectively.

To analyze changes in contact patterns, we set the networks of the second day and the first day as G_T and G_B, respectively. Figure 7(c) shows the 2D embedding result. To interpret G_T's unique patterns, we review the cPC loadings listed in Table 4 and colorcode the nodes in Figure 7(a), (b), and (c) based on the learned feature `F7-1`—f: $(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$, $\mathbf{x}$: PageRank. The results are shown in Figure 7(d), (e), and (f). We can see that i-cNRL discovers that G_T has both strongly (colored with more yellow in Figure 7(d) and (f)) and weakly connected regions from others (colored with more purple), while all of G_B's nodes have relatively strong connections between each other,



(a) G_T: Day 2 (b) G_B: Day 1 (c) G_T: Day 2, G_B: Day 1

(d) G_T: Day 2 (e) G_B: Day 1 (f) G_T: Day 2, G_B: Day 1

(g) G_T: Day 2 (h) G_B: Day 1 (i) G_T: Day 2, G_B: Day 1

Figure 7: Results for Case Study 3. (a) and (b) show the network structures of G_T and G_B. (c) presents the 2D embedding by i-cNRL. (d-f) show the colorcoded nodes in (a-c) based on the feature—f: $(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$, $\mathbf{x}$: the PageRank (`F7-1` in Table 4). (g-i) show the nodes colored by the class name where the first number indicates the grade (e.g., '1-A' is the first grade class). The networks include 'teacher' nodes.

as seen in the laid-out result in Figure 7(b).

According to the study by Stehlé et al. (2011), the students tended to have more contact within the same class than between classes. To relate the class information and the found unique patterns, we colorcode the nodes (i.e., students) based on their class, as shown in Figure 7(g), (h), and (i). From these results, we notice that i-cNRL well separates groups of students who have less (e.g., gray, pink, or teal nodes) and more (e.g., orange nodes) contact between classes in G_T.

7.3. Comparison with Other Potential Designs

Our i-cNRL utilizes DeepGL and cPCA for cNRL's two essential components—NRL and contrastive learning—to provide interpretable results. However, if the interpretability is not required, we can replace each of the learning methods with other alternatives. Here we compare three different designs for cNRL: (1) DeepGL & cPCA, (2) GraphSAGE (Hamilton et al. 2017) & cPCA, and (3) DeepGL & cVAE (Abid and Zou 2019).

Quantitative Results

We compare the quality of contrastive representations obtained with each design. A good contrastive representation should more widely distribute nodes in the target network than the background, and it should also show different patterns in the target and background networks. For example, as shown in Figure 3, cPCA ($\alpha = 72$) provides a better contrastive representation than PCA ($\alpha = 0$). To compare the aspects above, we use three different dissimilarity measures: dispersion ratio, Bhattacharyya distance (Bi et al. 2017), and Kullback-Leibler (KL) divergence (Wang et al. 2009) from a set of nodes in $\mathbf{Y}_B$ to that in $\mathbf{Y}_T$. The dispersion ratio represents how widely nodes in $\mathbf{Y}_T$ are scattered relative to $\mathbf{Y}_B$. The Bhattacharyya distance indicates closeness or overlaps of nodes in $\mathbf{Y}_T$ and $\mathbf{Y}_B$. The KL divergence of $\mathbf{Y}_T$ from $\mathbf{Y}_B$ shows the difference between their probability distributions of nodes. For all the above measures, the higher the value, the better the design.

We calculate the dispersion ratio of $\mathbf{Y}_T$ to $\mathbf{Y}_B$ with: $\frac{\mathrm{tr}(\mathbf{Y}'^{\top}_T \mathbf{Y}'_T)/n_T}{\mathrm{tr}(\mathbf{Y}'^{\top}_B \mathbf{Y}'_B)/n_B}$, where $\mathbf{Y}'_T$ and $\mathbf{Y}'_B$ are the scaled matrices of $\mathbf{Y}_T$ and $\mathbf{Y}_B$, respectively, obtained by applying the standardization to a concatenated matrix of $\mathbf{Y}_T$ and $\mathbf{Y}_B$. We use $\mathbf{Y}'_T$ and $\mathbf{Y}'_B$, instead of $\mathbf{Y}_T$ and $\mathbf{Y}_B$, to avoid the scaling differences in the embedding's axes across the three designs. For the Bhattacharyya distance and KL divergence, since we do not have the exact probability distributions of $\mathbf{Y}_T$ and $\mathbf{Y}_B$, we employ the estimation methods described by Bi et al. (2017) and Wang et al. (2009).

For GraphSAGE, we specifically select the GraphSAGE-maxpool model because it produces the best result according to Hamilton et al. (2017). We use the default parameter values used by Hamilton et al. (2017) and Abid and Zou (2019) for GraphSAGE and cVAE, except that we set 24 as the number of features leaned by GraphSAGE. For the input features of GraphSAGE, we set the same base features used for DeepGL (see Table 7 for details). We obtain 2D embeddings with the cPCs (with cPCA) or *salient latent variables* (with cVAE) (Abid and Zou 2019). Since cVAE relies on the probabilistic encoders, the results could be different for each trial, and thus we compute the mean value of each measure for 10 trials.

Table 6 shows a comparison of the three methods on different pairs of networks using the measures above. We can see that in general DeepGL & cPCA and GraphSAGE & cPCA have better scores than DeepGL & cVAE. Between DeepGL & cPCA and GraphSAGE & cPCA, DeepGL & cPCA tends to provide better results except for the dolphin and Karate networks, which have small numbers of nodes.

Table 6: Comparison of contrastive representation quality.

		dispersion ratio			Bhattacharyya			KL of $\mathbf{Y}_T$ from $\mathbf{Y}_B$		
G_T	G_B	DG& cPCA	GS& cPCA	DG& cVAE	DG& cPCA	GS& cPCA	DG& cVAE	DG& cPCA	GS& cPCA	DG& cVAE
Dolphin	Karate	174	**9,754**	1.48	1.40	**1.73**	0.92	6.82	**12.76**	0.96
P2P	Price 2	**21,744**	1,801	3.36	**7.52**	4.72	1.13	**45.73**	14.09	36.4
LC-multi.	C.-AP/MS	**376**	54	2.71	1.52	**1.76**	0.31	**18.49**	16.61	15.09
Sch.-Day2	Sch.-Day1	**57**	6	2.20	**1.81**	0.60	0.56	**5.82**	1.80	0.80

*DG=DeepGL, GS=GraphSAGE, P2P=p2p-Gnutella08, C.-AP/MS=Combined-AP/MS

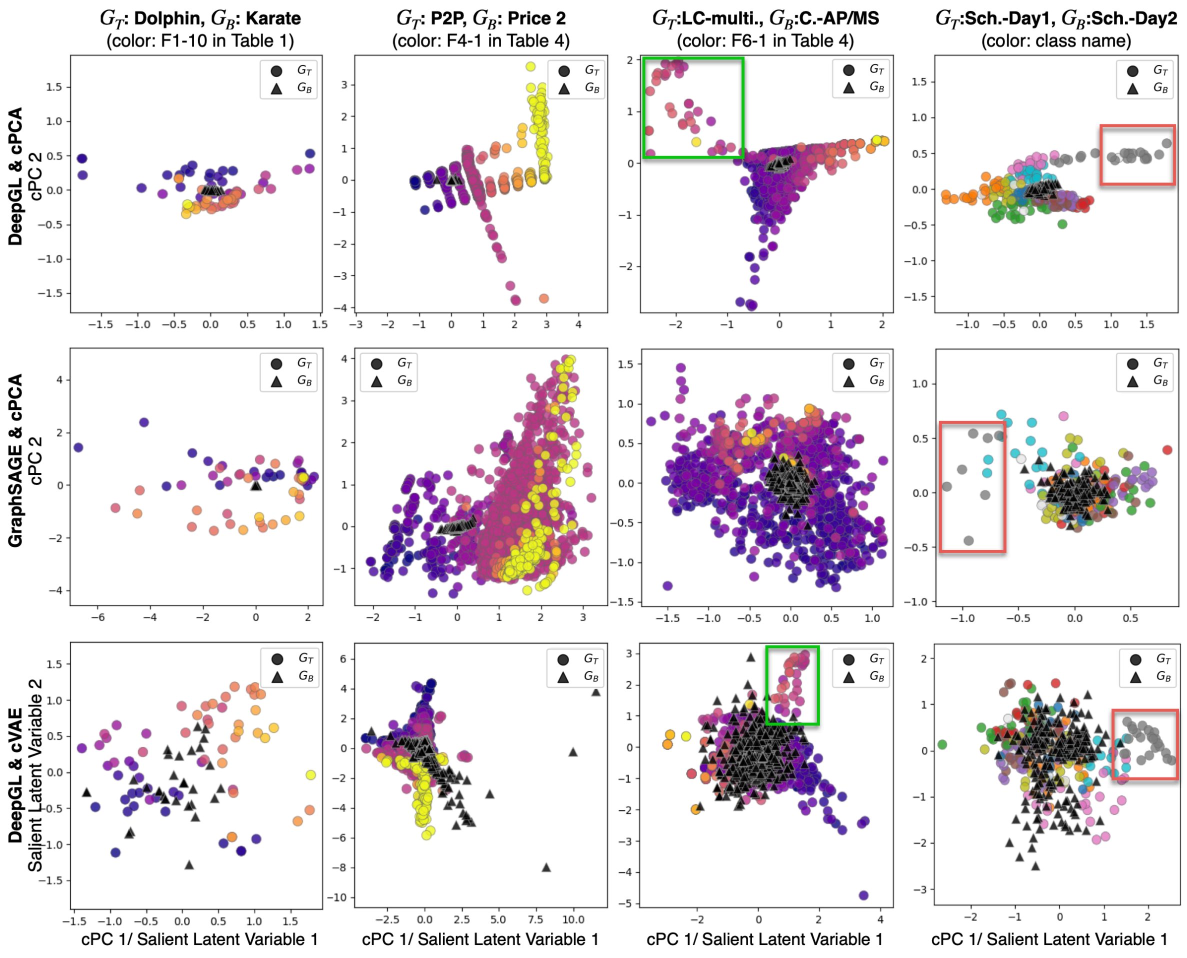


Figure 8: Visual comparison of the 2D embeddings. In the left three columns, the nodes in G_T are colored based on values of the feature that most contributes to `cPC 1`. The nodes in the far right column are colored by their class name. The green and red rectangles annotate distinct groups that can be seen in the 2D embeddings (refer to the description in Qualitative Results).

Qualitative Results

We visually compare the embedding results to review more detailed differences, as shown in Figure 8. For cVAE, we show the results that have the longest Bhattacharyya distance from 10 trials. Because GraphSAGE and cVAE do not provide interpretable features, for the comparison, we colorcode the nodes of the target network by the feature values from the DeepGL results. In specific, the left three columns in Figure 8 are colored based on values of the feature that has the top absolute loadings for `cPC 1` and the far right column is colored by their class name.

We can see that although the quality of the contrastive representation in Table 6 is different, these different designs seem to identify similar unique patterns. For instance, all the results of P2P and Price 2 show monotonic increase of the feature value (`F4-1`—k-core numbers). Also, for LC-mupltiple and Combined-AP/MS, both DeepGL & cPCA and DeepGL & cVAE depict clearly separated patterns, as indicated with the green rectangles while GraphSAGE & cPCA does not show the same pattern. Furthermore, in each result of the school networks, we can see a distinct group that consists of gray nodes, as annotated with the red rectangles.

From the above quantitative and qualitative comparisons, we can see that DeepGL & cPCA (i.e., i-cNRL) generates similar quality results when compared with the alternatives. However, the other two designs do not provide interpretable results.

8. Conclusion and Future Work

This work introduces contrastive network representation learning (cNRL), which aims to reveal unique patterns in one network relative to another. Furthermore, we demonstrate a method of cNRL, i-cNRL, that is generic and interpretable. With these contributions, our work provides a new approach to network comparison.

We have demonstrated the usability of i-cNRL with small- or medium-scale networks (less than 10,000 nodes) to provide intelligible examples. As a next step, we plan to apply i-cNRL on larger networks (e.g., networks with millions of nodes). When analyzing such large, complex networks, the linearity of cPCA used in i-cNRL might limit the capability of finding unique patterns. Therefore, we will investigate how to incorporate nonlinear contrastive learning methods (such as cVAE) for cNRL while retaining interpretability.

Acknowledgments

This research was partially carried out at FXPAL. This research is sponsored in part by the U.S. National Science Foundation through grants IIS-1741536 and IIS-1528203, and FXPAL through its internship program.

References

Abid, A., Zhang, M. J., Bagaria, V. K., and Zou, J. (2018). Exploring patterns enriched in a dataset with contrastive principal component analysis. *Nature Communications*, 9(1):2134.

Abid, A. and Zou, J. (2019). Contrastive variational autoencoder enhances salient features. *arXiv:1902.04601.*

Adadi, A. and Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence. *IEEE Access*, 6:52138–52160.

Barabási, A.-L. (2016). *Network Science*. Cambridge University Press.

Beauchamp, M. A. (1965). An improved index of centrality. *Behavioral Science*, 10(2):161–163.

Bhanot, G., Gara, A., Heidelberger, P., Lawless, E., Sexton, J. C., and Walkup, R. (2005). Optimizing task layout on the Blue Gene/L supercomputer. *IBM Journal of Research and Development*, 49(2.3):489–500.

Bi, S., Prabhu, S., Cogan, S., and Atamturktur, S. (2017). Uncertainty quantification metrics with varying statistical information in model calibration and validation. *AIAA Journal*, pages 3570–3583.

Boileau, P., Hejazi, N. S., and Dudoit, S. (2020). Exploring high-dimensional biological data with sparse contrastive principal component analysis. *Bioinformatics*, 36(11):3422–3430.

Cai, H., Zheng, V. W., and Chang, K. C.-C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637.

Chen, H. and Sharp, B. M. (2004). Content-rich biological network constructed by mining pubmed abstracts. *BMC Bioinformatics*, 5(1):147.

Chen, J., Ma, T., and Xiao, C. (2018). FastGCN: Fast learning with graph convolutional networks via importance sampling. In *Proceedings of the International Conference on Learning Representations*.

Chu, X., Fan, X., Yao, D., Zhu, Z., Huang, J., and Bi, J. (2019). Cross-network embedding for multi-network alignment. In *Proceedings of the World Wide Web Conference*, pages 273–284.

Collins, S. R., Kemmeren, P., Zhao, X.-C., Greenblatt, J. F., Spencer, F., Holstege, F. C., Weissman, J. S., and Krogan, N. J. (2007). Toward a comprehensive atlas of the physical interactome of *Saccharomyces cerevisiae*. *Molecular & Cellular Proteomics*, 6(3):439–450.

Crnovrsanin, T., Muelder, C., Faris, R., Felmlee, D., and Ma, K.-L. (2014). Visualization techniques for categorical analysis of social networks with multiple edge sets. *Social Networks*, 37:56–64.

Dinkelbach, W. (1967). On nonlinear fractional programming. *Management Science*, 13(7):492–498.

Dirie, A.-H., Abid, A., and Zou, J. (2019). Contrastive multivariate singular spectrum analysis. In *Proceeding of Allerton Conference on Communication, Control, and Computing*, pages 1122–1127. IEEE.

Emmert-Streib, F., Dehmer, M., and Shi, Y. (2016). Fifty years of graph matching, network alignment and network comparison. *Information Sciences*, 346:180–197.

Fujiwara, T., Kwon, O.-H., and Ma, K.-L. (2020). Supporting analysis of dimensionality reduction results with contrastive learning. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):45–55.

Fujiwara, T. and Liu, T.-P. (2020). Contrastive multiple correspondence analysis (cMCA): Using contrastive learning to identify latent subgroups in political parties. *arXiv:2007.04540*.

Fujiwara, T., Wei, X., Zhao, J., and Ma, K.-L. (2022). Interactive dimensionality reduction for comparative analysis. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):758–768.

Gaiteri, C., Mostafavi, S., Honey, C. J., De Jager, P. L., and Bennett, D. A. (2016). Genetic variants in Alzheimer disease-molecular and brain network approaches. *Nature Reviews Neurology*, 12(7):413.

Gao, H. and Ji, S. (2019). Graph representation learning via hard and channel-wise attention networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 741–749.

Ge, R. and Zou, J. (2016). Rich component analysis. In *Proceedings of the International Conference on Machine Learning*, pages 1502–1510.

Goldenberg, A., Zheng, A. X., Fienberg, S. E., and Airoldi, E. M. (2010). A survey of statistical network models. *Foundations and Trends® in Machine Learning*, 2(2):129–233.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864.

Guo, Y.-F., Li, S.-J., Yang, J.-Y., Shu, T.-T., and Wu, L.-D. (2003). A generalized Foley-Sammon transform based on generalized fisher discriminant criterion and its application to face recognition. *Pattern Recognition Letters*, 24(1):147–158.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034.

Heimann, M., Shen, H., Safavi, T., and Koutra, D. (2018). REGAL: Representation learning-based graph alignment. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 117–126.

Hu, Y. (2005). Efficient, high-quality force-directed graph drawing. *Mathematica Journal*, 10(1):37–71.

Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., and Makedon, F. (2021). A survey on contrastive self-supervised learning. *Technologies*, 9(1).

Jia, Y., Nie, F., and Zhang, C. (2009). Trace ratio problem revisited. *IEEE Transactions on Neural Networks*, 20(4):729–735.

Jolliffe, I. T. (1986). Principal component analysis and factor analysis. In *Principal Component Analysis*, pages 115–128. Springer.

Kwon, O.-H., Crnovrsanin, T., and Ma, K.-L. (2018). What would a graph look like in this layout? a machine learning approach to large graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):478–488.

Larivière, V., Gingras, Y., and Archambault, É. (2006). Canadian collaboration networks: A comparative analysis of the natural sciences, social sciences and the humanities. *Scientometrics*, 68(3):519–533.

Le-Khac, P. H., Healy, G., and Smeaton, A. F. (2020). Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934.

Leskovec, J., K2leinberg, J., and Faloutsos, C. (2007). Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1):2–es.

Leskovec, J. and Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data. Accessed: 2021-8-11.

Liu, L., Xu, J., Russell, D., Townend, P., and Webster, D. (2009). Efficient and scalable search on scale-free P2P networks. *Peer-to-Peer Networking and Applications*, 2(2):98–108.

Lusseau, D., Schneider, K., Boisseau, O. J., Haase, P., Slooten, E., and Dawson, S. M. (2003). The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405.

Ma, G., Ahmed, N. K., Willke, T. L., Sengupta, D., Cole, M. W., Turk-Browne, N. B., and Yu, P. S. (2019). Deep graph similarity learning for brain data analysis. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 2743–2751.

Moshiri, N. (2018). The dual-Barabási-Albert model. *arXiv:1810.10538*.

Newman, M. (2018). *Networks*. Oxford university press.

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856.

Pržulj, N. (2007). Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183.

Reguly, T., Breitkreutz, A., Boucher, L., Breitkreutz, B.-J., Hon, G. C., Myers, C. L., Parsons, A., Friesen, H., Oughtred, R., Tong, A., Stark, C., Ho, Y., Botstein, D., Andrews, B., Boone, C., Troyanskya, O. G., Ideker, T., Dolinski, K., Batada, N. N., and Tyers, M. (2006). Comprehensive curation and analysis of global interaction networks in saccharomyces cerevisiae. *Journal of Biology*, 5(4):11.

Ripeanu, M., Iamnitchi, A., and Foster, I. (2002). Mapping the gnutella network. *IEEE Internet Computing*, 6(1):50.

Rossi, R. A., Zhou, R., and Ahmed, N. (2018). Deep inductive graph representation learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(3):438–452.

Severson, K. A., Ghosh, S., and Ng, K. (2019). Unsupervised learning with contrastive latent variable models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4862–4869.

Shen, X., Dai, Q., Mao, S., Chung, F.-l., and Choi, K.-S. (2021). Network together: Node classification via cross network deep network embedding. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5):1935–1948.

Stanford Center for Cancer Systems Biology (2003). CCSB interactome database. http://interactome.dfci.harvard.edu/. Accessed: 2021-8-11.

Stehlé, J., Voirin, N., Barrat, A., Cattuto, C., Isella, L., Pinton, J.-F., Quaggiotto, M., Van den Broeck, W., Régis, C., Lina, B., and Vanhems, P. (2011). High-resolution measurements of face-to-face contact patterns in a primary school. *PLOS One*, 6(8).

Sun, F.-Y., Hoffman, J., Verma, V., and Tang, J. (2020). InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *Proceedings of the International Conference on Learning Representations*.

Tantardini, M., Ieva, F., Tajoli, L., and Piccardi, C. (2019). Comparing methods for comparing networks. *Scientific Reports*, 9(1):17557.

The SocioPatterns Collaboration (2008). SocioPatterns. http://www.sociopatterns.org/. Accessed: 2021-8-11.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2018). Graph attention networks. In *Proceedings of the International Conference on Learning Representations*.

Wang, Q., Kulkarni, S. R., and Verdú, S. (2009). Divergence estimation for multidimensional densities via k-nearest-neighbor distances. *IEEE Transactions on Information Theory*, 55(5):2392–2405.

Ying, Z., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. (2019). GNNExplainer: Generating explanations for graph neural networks. In *Advances in Neural Information Processing Systems*, pages 9240–9251.

Yu, H., Braun, P., Yıldırım, M. A., Lemmens, I., Venkatesan, K., Sahalie, J., Hirozane-Kishikawa, T., Gebreab, F., Li, N., Simonis, N., Hao, T., Rual, J.-F., Dricot, A., Vazquez, A., Murray, R. R., Simon, C., Tardivo, L., Tam, S., Svrzikapa, N., Fan, C., de Smet, A.-S., Motyl, A., Hudson, M. E., Park, J., Xin, X., Cusick, M. E., Moore, T., Boone, C., Snyder, M., Roth, F. P., Barabási, A.-L., Tavernier, J., Hill, D. E., and Vidal, M. (2008). High-quality binary protein interaction map of the yeast interactome network. *Science*, 322(5898):104–110.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473.

Zhang, C., Song, D., Huang, C., Swami, A., and Chawla, N. V. (2019). Heterogeneous graph neural network. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 793–803.

Zhang, X., Fujiwara, T., Chandrasegaran, S., Brundage, M. P., Sexton, T., Dima, A., and Ma, K.-L. (2021). A visual analytics approach for the diagnosis of heterogeneous and multidimensional machine maintenance data. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 196–205.

Zhang, Z., Cui, P., and Zhu, W. (2020). Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*. (early access).

Zou, J. Y., Hsu, D. J., Parkes, D. C., and Adams, R. P. (2013). Contrastive learning using spectral methods. In *Advances in Neural Information Processing Systems*, pages 2238–2246.

A. Implementation Details

We have implemented the cNRL architecture with Python 3 (refer to https://github.com/takanori-fujiwara/cnrl). The implemented cNRL architecture allows the user to apply any NRL and contrastive learning methods that provide "`fit`" and "`transform`" methods (as similar to machine learning methods supported in **scikit-learn**[3]). For the implementation of i-cNRL, we have integrated DeepGL and cPCA into the cNRL architecture. Because there is no implementation of DeepGL available from Python[4], we have implemented DeepGL with **graph-tool**[5]. For cPCA, we have modified the implementation available online[6] to add the automatic contrastive automatic selection described in Section 5.2.

B. Datasets

For the evaluation, we use the datasets in various data repositories, including SNAP, CCSB Interactome Database, and SocioPatterns as well as the synthetic datasets that we generated. To allow the reproducibility of this work, we provide links to the original network datasets, processed datasets, and feature matrices learned by DeepGL and GraphSAGE in https://takanori-fujiwara.github.io/s/cnrl/.

C. Experiment Details

The source code for generating the experimental results is available in https://takanori-fujiwara.github.io/s/cnrl/.

C.1. Learning Parameters of i-cNRL

DeepGL Settings

DeepGL has multiple adjustable settings as it is introduced as a comprehensive inductive NRL framework. We follow the terminologies in the work by Rossi et al. (2018) to describe the detailed settings for each evaluation. Refer to the work by Rossi et al. (2018) for those not explained in this paper (indicated with *italic* fonts below). For all the cNRL we performed, we have used DeepGL with $h = 3$ and the logarithmic binning to feature values with 0.5 as the *transformation parameter*, but without the *feature diffusion*. For the other settings, generally, we have used as many different relational feature operators and base features as possible for each network dataset. As for the relational feature operators, for directed networks, we have used all the combinations of $\{\Phi_S^-, \Phi_S^+, \Phi_S\}$ with $S = \{\text{mean}, \text{sum}, \text{max}, \text{L}^2\text{norm}\}$ (i.e., 12 operators in total). For undirected networks, we have used Φ_S where $S = \{\text{mean}, \text{sum}, \text{max}, \text{L}^2\text{norm}\}$. As for the base feature $\mathbf{x}$, we have used all centralities and measures available in **graph-tool**.

[3]**scikit-learn**, https://scikit-learn.org/, accessed 2021-8-11.

[4]Implementation using Java with **Neo4j** database is available from https://github.com/neo4j-graph-analytics/ml-models, accessed 2021-8-11.

[5]**graph-tool**, https://graph-tool.skewed.de/, accessed 2021-8-11.

[6]**ccpca**, https://github.com/takanori-fujiwara/ccpca, accessed 2021-8-11.

Table 7: The detailed DeepGL settings for each analysis.

G_T	G_B	$\mathbf{x}$	λ
Dolphin	Karate	{total-degree, betweenness, closeness, eigenvector, PageRank, Katz}	0.7
Price	Random	{in-degree, out-degree, total-degree, PageRank, betweenness, Katz, k-core}	0.3
Random	Price	{in-degree, out-degree, total-degree, PageRank, betweenness, Katz, k-core}	0.3
p2p-Gnutella08	Price 2	{in-degree, out-degree, total-degree, PageRank, betweenness, Katz, k-core}	0.5
p2p-Gnutella08	Enhanced Price	{in-degree, out-degree, total-degree, PageRank, betweenness, Katz, k-core}	0.5
LC-multiple	Combined-AP/MS	{total-degree, betweenness, eigenvector, PageRank, Katz}	0.7
School-Day2	School-Day1	{gender, total-degree, closeness, betweenness, eigenvector, PageRank, Katz}	0.7

However, for each network, some of these features have produced '`NaN`' values (e.g., for a disconnected network, the closeness centrality of each node is '`NaN`' because each node does not have a path to some other node). In that case, we have excluded such features from the base features. Note that, consequently, i-cNRL would not capture the unique patterns if the patterns are highly related to the excluded features. This is a limitation of our implementation where the computation of the base feature replies on **graph-tool**. However, if needed, without using our implementation for the base feature computation, analysts can precompute a variant of the centrality that does not produce '`NaN`' (e.g., instead of ordinary closeness, using the adjusted closeness by Beauchamp (1965)). Table 7 shows the base features we used for each analysis. For feature pruning of the learned $\mathcal{F}_i$, we have applied the same method used in the work by Rossi et al. (2018) with the *feature similarity threshold*, λ. As λ becomes larger, the number of features learned by NRL (i.e., d) increases. We have set a different λ value for each analysis, as listed in Table 7. In general, for the undirected networks, we have used relatively higher values ($\lambda = 0.7$) because the number of base features used is smaller when compared with the directed networks.

cPCA Settings

For all results, we have used cPCA with the automatic contrastive parameter selection and default settings. That is, we have applied the standardization to each of $\mathbf{X}_T$ and $\mathbf{X}_B$ for both learning and projection and the automatic contrastive parameter selection with $\epsilon = 10^{-3}$.

C.2. Full Sets of cPC Loadings

The full sets of cPC loadings obtained with i-cNRL for each analysis in Section 7.1 and Section 7.2 are listed in Table 8-11.

Table 8: All cPC loadings for Section 7.1.

relational function f	base feature $\mathbf{x}$	`cPC 1`	`cPC 2`
G_T: Price, G_B: Random			
$(\mathbf{x})$	in-degree	0.23	0.71
$(\mathbf{x})$	out-degree	0.52	0.00
$(\mathbf{x})$	total-degree	-0.79	-0.01
$(\mathbf{x})$	PageRank	0.00	0.00
$(\mathbf{x})$	betweenness	0.00	0.00
$(\mathbf{x})$	Katz	0.24	-0.71
$(\mathbf{x})$	k-core	0.00	0.00
$(\Phi^-_{\text{mean}})(\mathbf{x})$	in-degree	-0.01	0.01
$(\Phi^-_{\text{mean}} \circ \Phi^-_{\text{mean}})(\mathbf{x})$	in-degree	-0.00	0.00
G_T: Random, G_B: Price			
$(\mathbf{x})$	in-degree	-0.06	0.43
$(\mathbf{x})$	out-degree	0.02	0.04
$(\mathbf{x})$	total-degree	0.11	-0.79
$(\mathbf{x})$	PageRank	0.02	-0.01
$(\mathbf{x})$	betweenness	-0.01	0.00
$(\mathbf{x})$	Katz	-0.05	0.40
$(\mathbf{x})$	k-core	0.99	0.13
$(\Phi^-_{\text{mean}})(\mathbf{x})$	in-degree	-0.00	-0.00
$(\Phi^-_{\text{mean}} \circ \Phi^-_{\text{mean}})(\mathbf{x})$	in-degree	-0.00	-0.00

Table 9: All cPC loadings for Case Study 1.

relational function f	base feature $\mathbf{x}$	`cPC 1`	`cPC 2`
G_T: p2p-Gnutella08 , G_B: Price 2			
$(\mathbf{x})$	in-degree	-0.12	0.43
$(\mathbf{x})$	out-degree	0.04	0.01
$(\mathbf{x})$	total-degree	0.20	-0.79
$(\mathbf{x})$	PageRank	0.05	-0.00
$(\mathbf{x})$	betweenness	-0.00	0.00
$(\mathbf{x})$	Katz	-0.09	0.37
$(\mathbf{x})$	k-core	0.97	0.25
$(\Phi^-_{\text{mean}})(\mathbf{x})$	in-degree	-0.00	-0.00
$(\Phi^-_{\text{mean}})(\mathbf{x})$	out-degree	0.00	-0.00
$(\Phi^-_{\text{mean}})(\mathbf{x})$	betweenness	0.00	-0.00
$(\Phi_{\text{mean}})(\mathbf{x})$	out-degree	-0.00	0.00
$(\Phi^-_{\text{mean}} \circ \Phi^-_{\text{mean}})(\mathbf{x})$	in-degree	-0.00	0.00
$(\Phi^-_{\text{mean}} \circ \Phi^-_{\text{mean}})(\mathbf{x})$	out-degree	0.00	0.00
$(\Phi_{\text{mean}} \circ \Phi^-_{\text{mean}})(\mathbf{x})$	out-degree	0.00	-0.00
G_T: p2p-Gnutella08 , G_B: Enhanced Price			
$(\mathbf{x})$	in-degree	0.43	0.67
$(\mathbf{x})$	out-degree	0.16	-0.01
$(\mathbf{x})$	total-degree	-0.82	0.03
$(\mathbf{x})$	PageRank	-0.02	0.04
$(\mathbf{x})$	betweenness	0.00	-0.00
$(\mathbf{x})$	Katz	0.35	-0.74
$(\mathbf{x})$	k-core	0.01	-0.01
$(\Phi^-_{\text{mean}})(\mathbf{x})$	in-degree	-0.01	0.01
$(\Phi^-_{\text{mean}})(\mathbf{x})$	out-degree	0.00	0.00
$(\Phi^-_{\text{mean}})(\mathbf{x})$	betweenness	-0.00	-0.00
$(\Phi_{\text{mean}})(\mathbf{x})$	out-degree	0.00	0.00
$(\Phi^-_{\text{mean}} \circ \Phi^-_{\text{mean}})(\mathbf{x})$	in-degree	0.00	-0.00
$(\Phi^-_{\text{mean}} \circ \Phi^-_{\text{mean}})(\mathbf{x})$	out-degree	0.00	-0.00
$(\Phi_{\text{mean}} \circ \Phi^-_{\text{mean}})(\mathbf{x})$	out-degree	0.00	0.00

Table 10: All cPC loadings for Case Study 2.

relational function f	base feature $\mathbf{x}$	`cPC 1`	`cPC 2`
$(\mathbf{x})$	total-degree	-0.05	0.42
$(\mathbf{x})$	betweenness	-0.00	-0.00
$(\mathbf{x})$	eigenvector	0.03	0.41
$(\mathbf{x})$	PageRank	0.02	0.00
$(\mathbf{x})$	Katz	0.00	-0.81
$(\Phi_{\text{mean}})(\mathbf{x})$	total-degree	-0.33	0.06
$(\Phi_{\text{mean}})(\mathbf{x})$	betweenness	0.00	0.00
$(\Phi_{\text{mean}})(\mathbf{x})$	eigenvector	-0.44	-0.03
$(\Phi_{\text{mean}})(\mathbf{x})$	PageRank	-0.01	-0.01
$(\Phi_{\text{mean}})(\mathbf{x})$	Katz	0.83	0.01
$(\Phi_{\text{max}})(\mathbf{x})$	betweenness	0.01	0.00
$(\Phi_{\text{max}})(\mathbf{x})$	PageRank	-0.02	0.00
$(\Phi_{\text{mean}} \circ \Phi_{\text{mean}})(\mathbf{x})$	total-degree	-0.08	-0.03
$(\Phi_{\text{mean}} \circ \Phi_{\text{mean}})(\mathbf{x})$	betweenness	0.00	-0.00
$(\Phi_{\text{mean}} \circ \Phi_{\text{mean}})(\mathbf{x})$	PageRank	-0.02	-0.02
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	betweenness	0.00	0.00
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	PageRank	0.03	0.03
$(\Phi_{\text{max}} \circ \Phi_{\text{max}})(\mathbf{x})$	betweenness	-0.00	-0.01
$(\Phi_{\text{max}} \circ \Phi_{\text{max}})(\mathbf{x})$	PageRank	0.00	-0.01

Table 11: All cPC loadings for Case Study 3.

relational function f	base feature $\mathbf{x}$	`cPC 1`	`cPC 2`
$(\mathbf{x})$	total-degree	-0.18	-0.29
$(\mathbf{x})$	closeness	0.03	-0.00
$(\mathbf{x})$	betweenness	0.03	-0.01
$(\mathbf{x})$	eigenvector	0.22	-0.18
$(\mathbf{x})$	PageRank	-0.01	0.32
$(\mathbf{x})$	Katz	-0.08	0.14
$(\mathbf{x})$	gender	0.02	0.01
$(\Phi_{\text{mean}})(\mathbf{x})$	total-degree	-0.24	-0.10
$(\Phi_{\text{mean}})(\mathbf{x})$	betweenness	0.08	0.06
$(\Phi_{\text{mean}})(\mathbf{x})$	gender	0.04	0.02
$(\Phi_{\text{sum}})(\mathbf{x})$	gender	-0.02	-0.01
$(\Phi_{\text{max}})(\mathbf{x})$	total-degree	0.04	-0.23
$(\Phi_{\text{max}})(\mathbf{x})$	closeness	-0.12	0.02
$(\Phi_{\text{max}})(\mathbf{x})$	betweenness	0.06	0.00
$(\Phi_{\text{max}})(\mathbf{x})$	eigenvector	0.12	-0.07
$(\Phi_{\text{max}})(\mathbf{x})$	PageRank	-0.10	0.10
$(\Phi_{\text{max}})(\mathbf{x})$	Katz	0.03	0.17
$(\Phi_{\text{max}})(\mathbf{x})$	gender	0.00	0.00
$(\Phi_{\text{L}^2\text{norm}})(\mathbf{x})$	gender	-0.02	-0.01
$(\Phi_{\text{mean}} \circ \Phi_{\text{mean}})(\mathbf{x})$	total-degree	0.23	0.09
$(\Phi_{\text{mean}} \circ \Phi_{\text{mean}})(\mathbf{x})$	betweenness	-0.14	-0.02
$(\Phi_{\text{mean}} \circ \Phi_{\text{mean}})(\mathbf{x})$	gender	-0.09	-0.05
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	total-degree	0.19	-0.38
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	closeness	-0.40	0.33
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	betweenness	0.33	0.09
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	eigenvector	0.29	-0.29
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	PageRank	-0.53	-0.21
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	Katz	0.22	0.50
$(\Phi_{\text{mean}} \circ \Phi_{\text{max}})(\mathbf{x})$	gender	-0.00	0.00
$(\Phi_{\text{max}} \circ \Phi_{\text{mean}})(\mathbf{x})$	betweenness	0.00	-0.01
$(\Phi_{\text{max}} \circ \Phi_{\text{mean}})(\mathbf{x})$	gender	-0.01	-0.01
$(\Phi_{\text{max}} \circ \Phi_{\text{sum}})(\mathbf{x})$	gender	-0.00	-0.00
$(\Phi_{\text{max}} \circ \Phi_{\text{L}^2\text{norm}})(\mathbf{x})$	gender	-0.00	-0.00

C.3. Network Generation Models and Parameters

We have used the Gilbert's and Price's network models to generate Random (`N3`), Price (`N4`), and Price 2 (`N6`) in Table 1. Also, in Case Study 1, we have introduced the enhanced Price's network model as the solution to generate a network of which nodes have different k-core numbers—Enhanced Price (`N7`). In the following, we explain the details of the parameters we used for the network generation and the enhanced Price's model.

Parameters for the Gilbert's and Price's Models

The Gilbert's model generating a random network requires the fixed probability of a connection of each pair of nodes. We have set the probability to 0.05 for generating Random (`N3`). The Price's model requires the fixed number of out-degree of newly added nodes as its parameter. We have set this parameter to 3 for both Price (`N4`) and Price 2 (`N6`).

Enhanced Price's Model

For the enhanced Price's model, we modify the Price's model to be able to generate nodes with various k-core numbers. To achieve this, in the enhanced Price's model, we allow the user to set multiple positive integer numbers of out-degree of newly added nodes. We denote this input as $\kappa = \{\kappa_1, \cdots, \kappa_u\}$ where u is the length of the input. To select one number from κ when a new node is added, we need to set the probability of selecting each number. We denote the probabilities as $p = \{p_1, \cdots, p_u\}$ where $\sum p = 1$. To generate Enhanced Price (`N7`), we have set these parameters to $\kappa = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and $p = \{0.3, 0.25, 0.15, 0.1, 0.075, 0.05, 0.025, 0.025, 0.0125, 0.0125\}$.

C.4. Settings of GraphSAGE and cVAE

We describe the detailed settings and parameters of GraphSAGE and cVAE used in Section 7.3. We have used the source code provided by the authors of GraphSAGE[7] and cVAE[8]. For GraphSAGE, we have used the unsupervised model `graphsage_maxpool` with 24 as the number of features learned (i.e., `dim_1 = 12` and `dim_2 = 12`) while we have followed the default values for other parameters (e.g., `learning_rate = 0.00001` and `model_size = 'small'`). We have used cVAE with the default parameters (i.e., `intermediate_dim = 12`, `latent_dim = 2`, `batch_size = 64`, and `epochs = 500`).

C.5. Automatic Contrastive Parameter Selection

Figure 9 shows transitions of α value during the automatic selection in i-cNRL. For all the experiments, we can see that α reaches the convergence before 10 iterations.

[7]GraphSAGE: `https://github.com/williamleif/GraphSAGE`, accessed 2021-8-11.
[8]Contrastive VAE: `https://github.com/abidlabs/contrastive_vae`, accessed 2021-8-11.

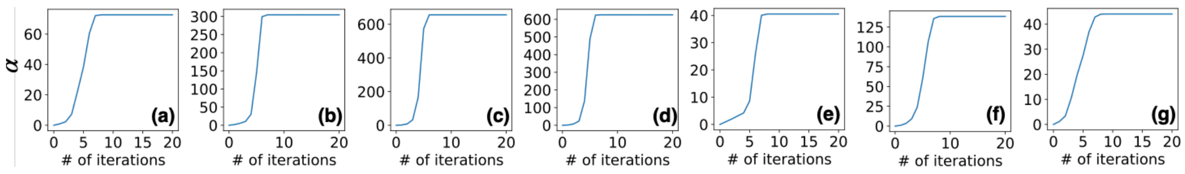


Figure 9: Transitions of α with the automatic selection: (a) G_T: Dolphin, G_B: Karate, (b) G_T: Price, G_B: Random, (c) G_T: Random, G_B: Price, (d) G_T: p2p-Gnutella08, G_B: Price 2, (e) G_T: p2p-Gnutella08, G_B: Enhanced Price, (f) G_T: LC-multiple, G_B: Combined-AP/MS, and (g) G_T: School-Day2, G_B: School-Day2.

Affiliation:

Takanori Fujiwara
Department of Computer Science
University of California, Davis
2136 Kemper Hall, One Shields Avenue, Davis, CA 95616, USA
E-mail: tfujiwara@ucdavis.edu
URL: https://takanori-fujiwara.github.io/