Graph Embedding on Biomedical Networks: Methods, Applications, and Evaluations

Supplementary Materials

Xiang Yue[1,*], Zhen Wang[1], Jingong Huang[2], Srinivasan Parthasarathy[1], Soheil Moosavinasab[3], Yungui Huang[3], Simon M. Lin[3], Wen Zhang[4], PingZhang[1,5], and Huan Sun[1,*]

[1]Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA,

[2]Department of Electrical andComputer Engineering, The Ohio State University, Columbus, OH, USA,

[3]Research Information Solutions and Innovation, TheResearch Institute at Nationwide Children's Hospital, Columbus, OH, USA,

[4]College of Informatics, Huazhong Agricultural University,Wuhan, Hubei, China,

[5]Department of Biomedical Informatics, The Ohio State University, Columbus, OH, USA

∗To whom correspondence should be addressed.

(yue.149@osu.edu, sun.397@osu.edu)

Index

1 Hyper-parameters

Table S1: The meanings of main hyper-parameters in different embedding methods.

Methods	Hyper-parameters
GraRep	*Ksteps*: *k*-step relational information (*k*-step transition probability matrix)
DeepWalk	*number of walks*: the number of walks at each node *walk length*: the length of each walk
node2vec	*p*, *q*: two parameters that control how fast the walk explores and leaves the neighborhood of starting node
struc2vec	*number of walks*: the number of walks at each node *walk length*: the length of each walk
LINE	*epochs*: number of training epochs
SDNE	α: balances the weight of 1st-order and 2nd-order proximities β: controls the reconstruction weight of the nonzero elements in the training graph
GAE	hidden units: number of units in hidden layer

General guidelines for setting hyper-parameters of various embedding methods.

- **GraRep**. As the *Ksteps* increases, the performance will often raise, since modeling high-order proximity is often important in networks. But when the *Ksteps* continuously increases, the performance starts to drop slowly as it may include some noise. So we suggest practitioners set the *Ksteps* to a smaller value (e.g., *Ksteps* = 2 or 3).
- **DeepWalk, node2vec and struc2vec**. In most cases, larger *number of walks* will lead to better performances since a store of "node sequences" could better model the node proximity. But the *walk length* is suggested not being too large considering that "walking" further may produce some noise. For *p, q* in node2vec, in general, the prediction performance is better when *p* and *q* are set to smaller values. So, we suggest practitioners begin to tune *p, q* at a smaller value (meaning "walking" further).
- **LINE**. For a smaller graph, smaller training epochs could lead to the model converged and vice versa for larger graph.
- **SDNE**. There are no general rules for choosing the values of α, β. But through our observation, practitioners could begin tuning at a smaller β and a smaller α since they often lead to a promising result.
- **GAE**. More *hidden units* often lead to better performance. But it could be expected that the training time will also increase. When reaching some threshold, increasing the *hidden units* would probably not result in big improvements. Considering the above, we suggest practitioners set the hidden units to a relatively bigger value (e.g., 256).

From page 4-10, we plot the performance of different embedding methods when these hyper-parameters are tuned.

Fig. S1: The influence of *dimensionality* on different embedding methods on NDFRT DDA, DrugBank DDI and STRING PPI datasets (the results of CTD DDA and Clin Term COOC are included in the main manuscript).

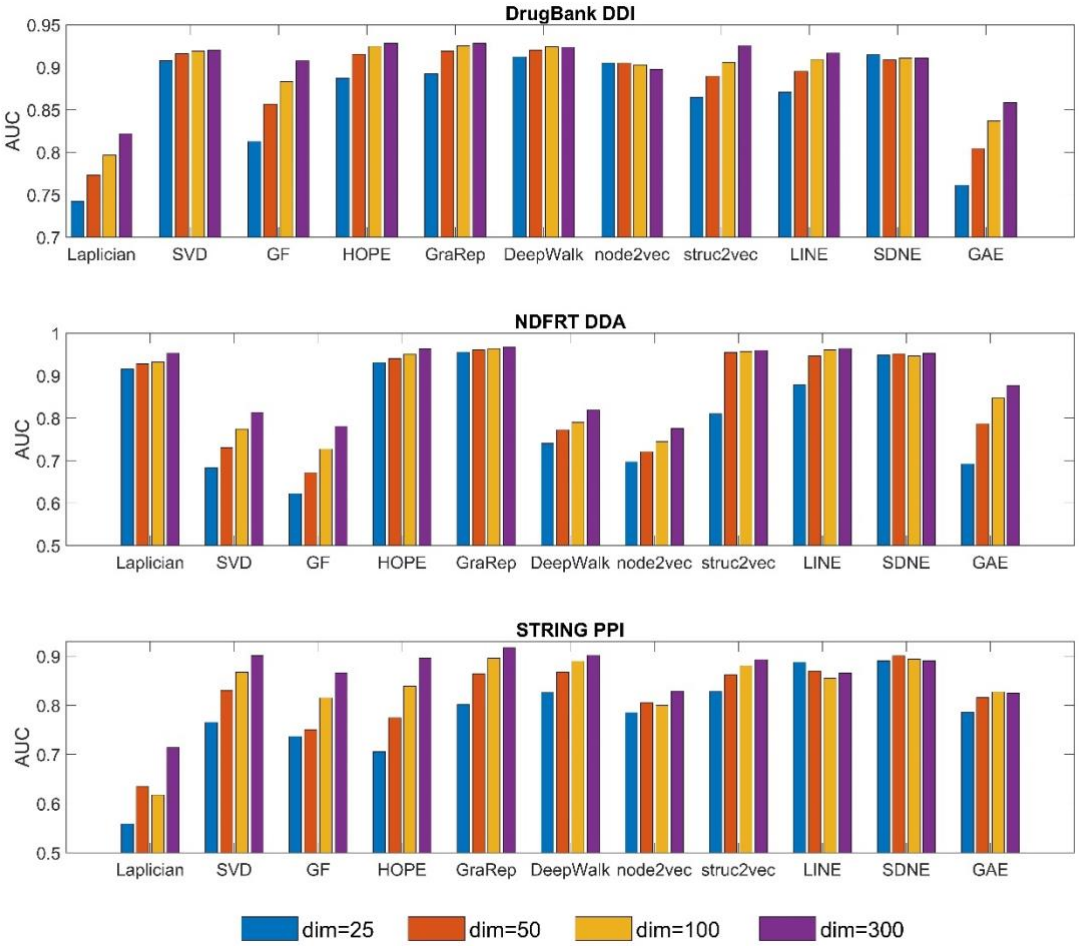


Fig. S2: The influence of the main hyper-parameter: *Ksteps* on *GraRep*.

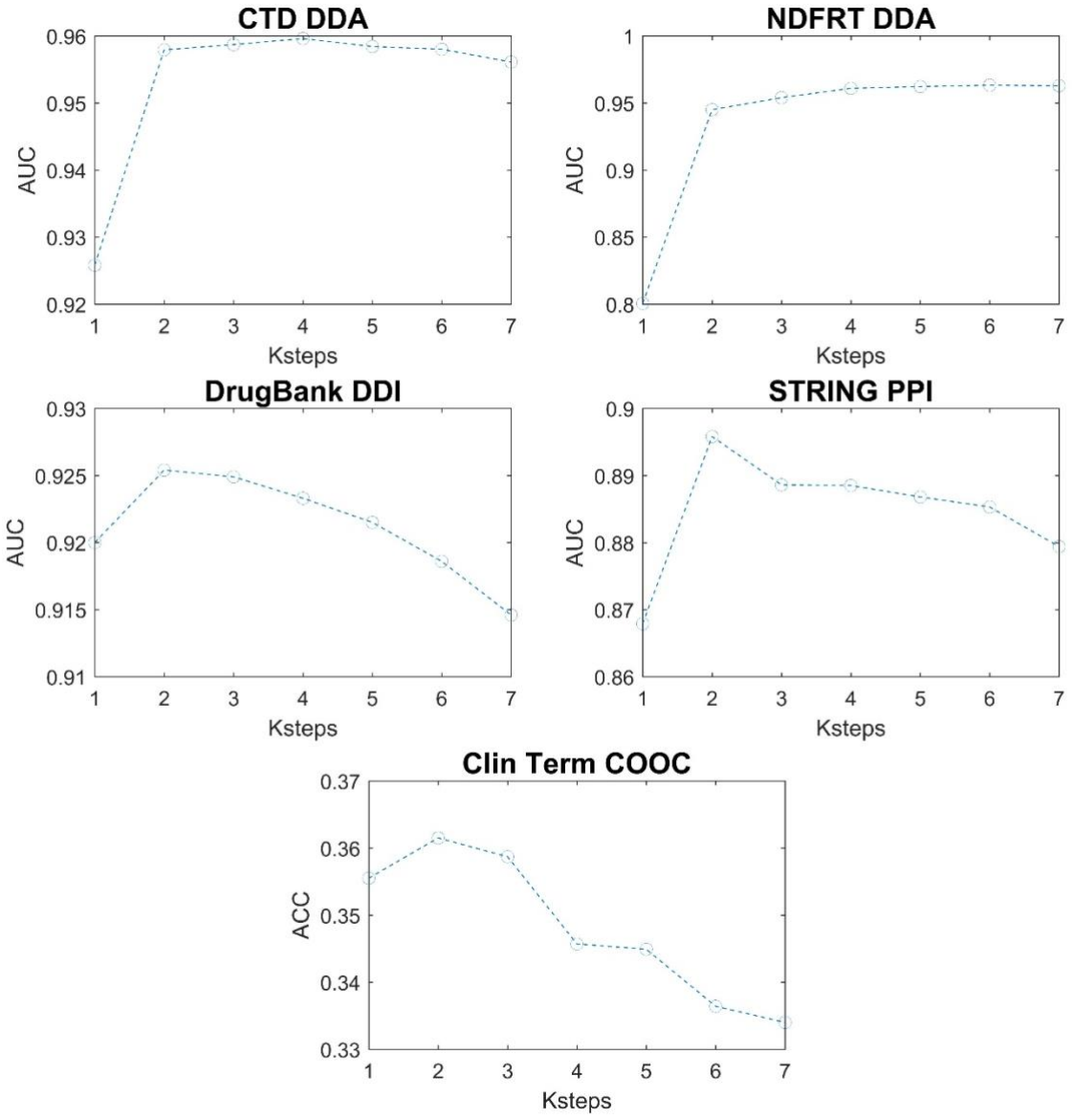


Fig. S3: The influence of the main hyper-parameters: *number of walks* and *walk length* on *DeepWalk*.

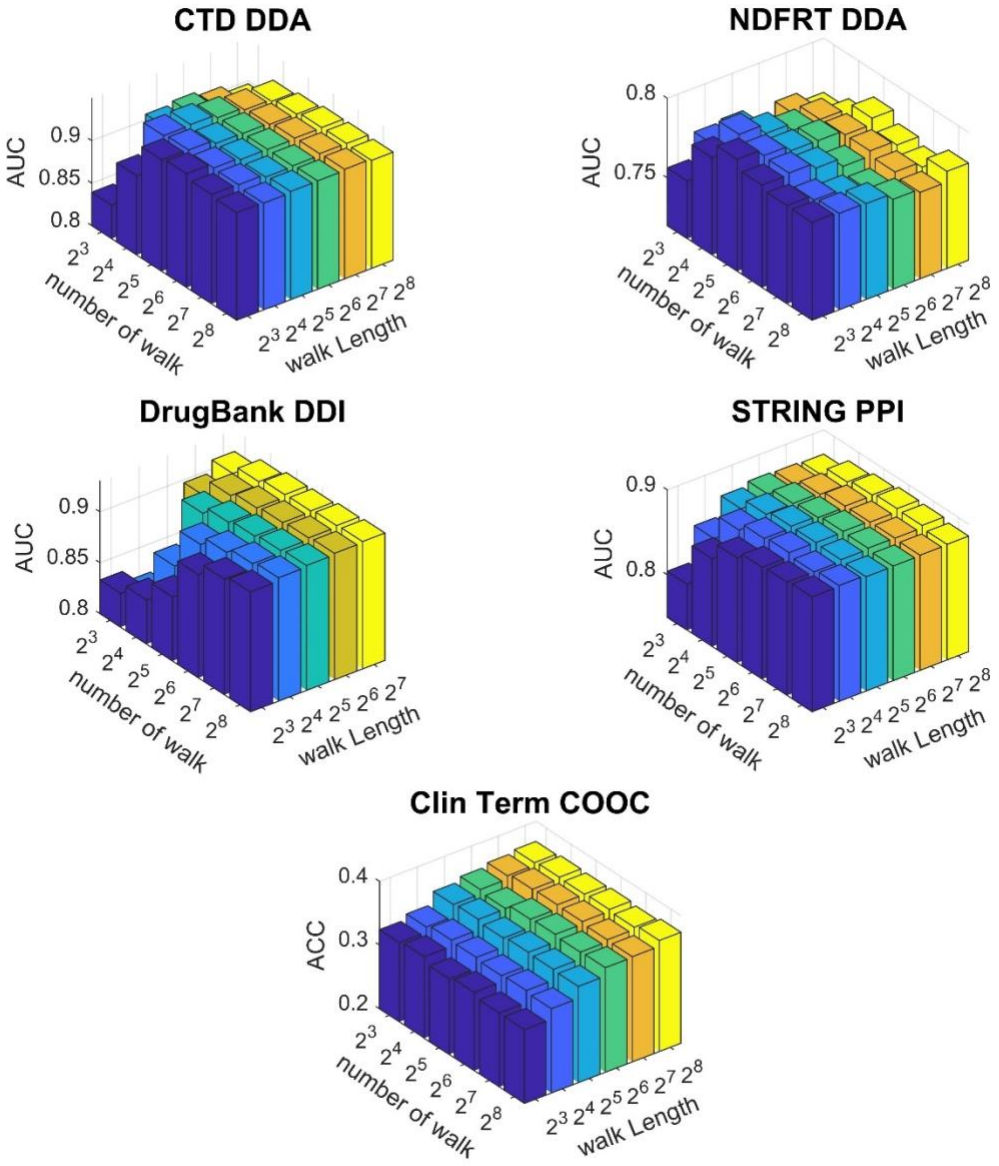


Fig. S4: The influence of the main hyper-parameters: *p* and *q* on *node2vec*.

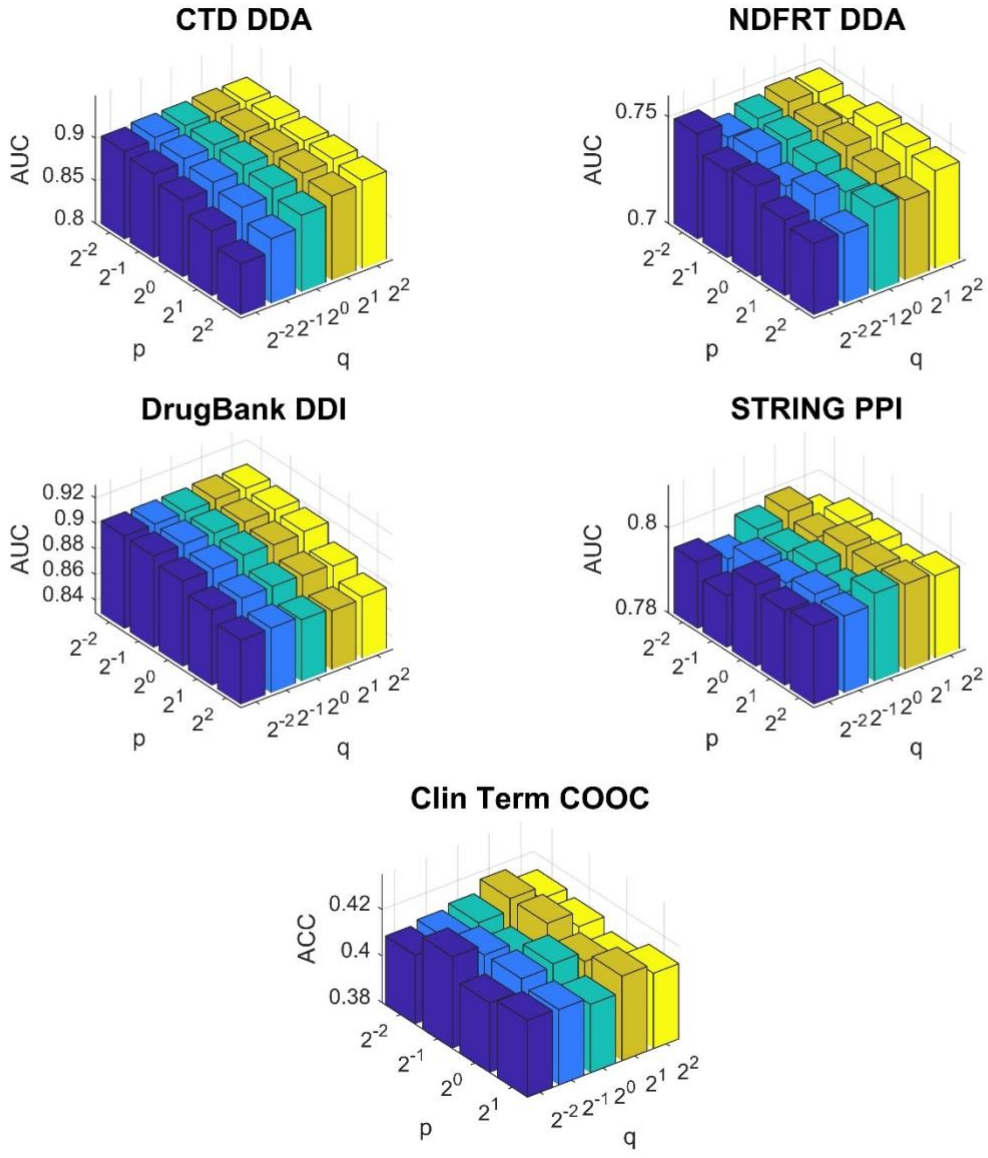


Fig. S5: The influence of the main hyper-parameters: ***number of walks*** and ***walk length*** **on** ***struc2vec*****.**

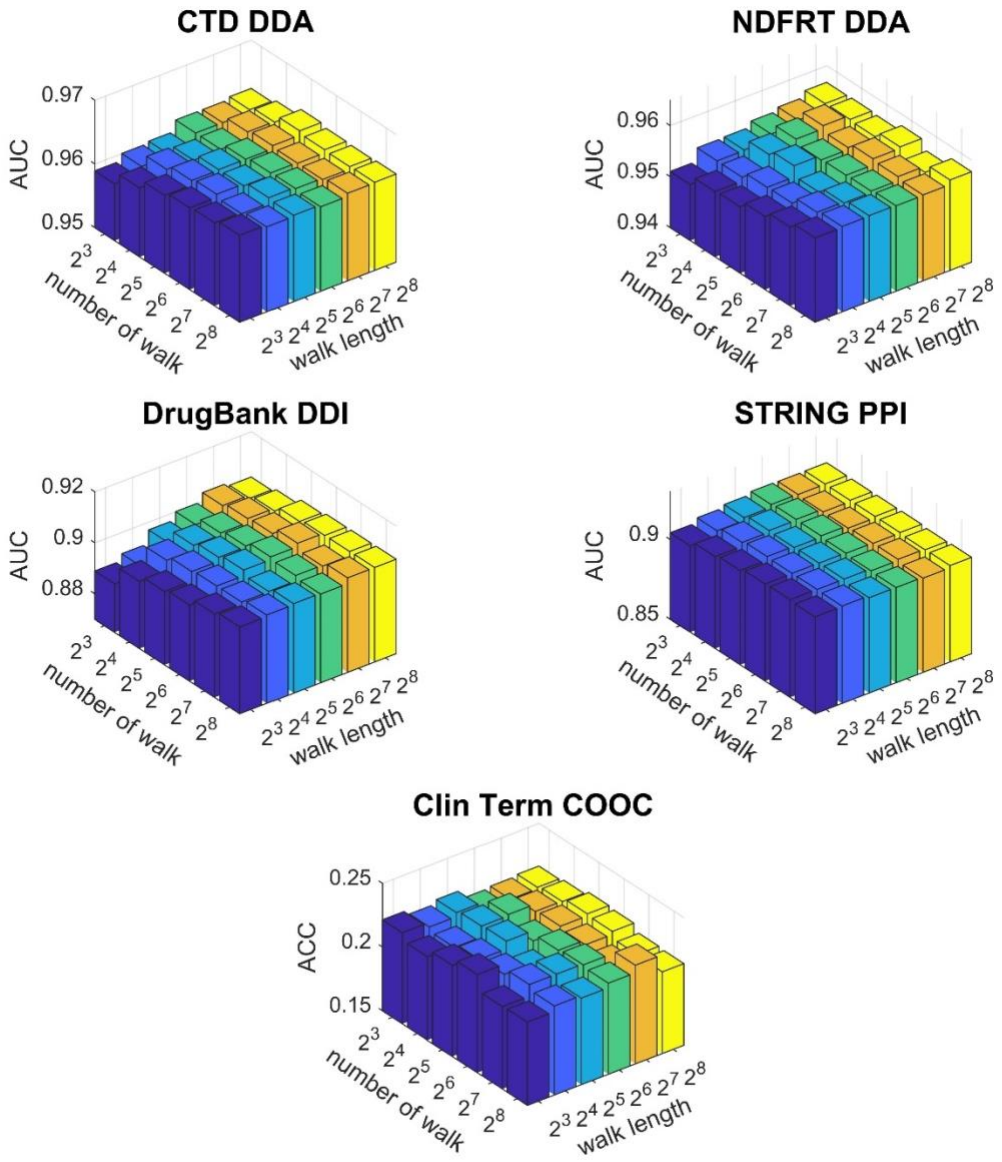


Fig. S6: The influence of the main hyper-parameter: *epochs* on *LINE*.

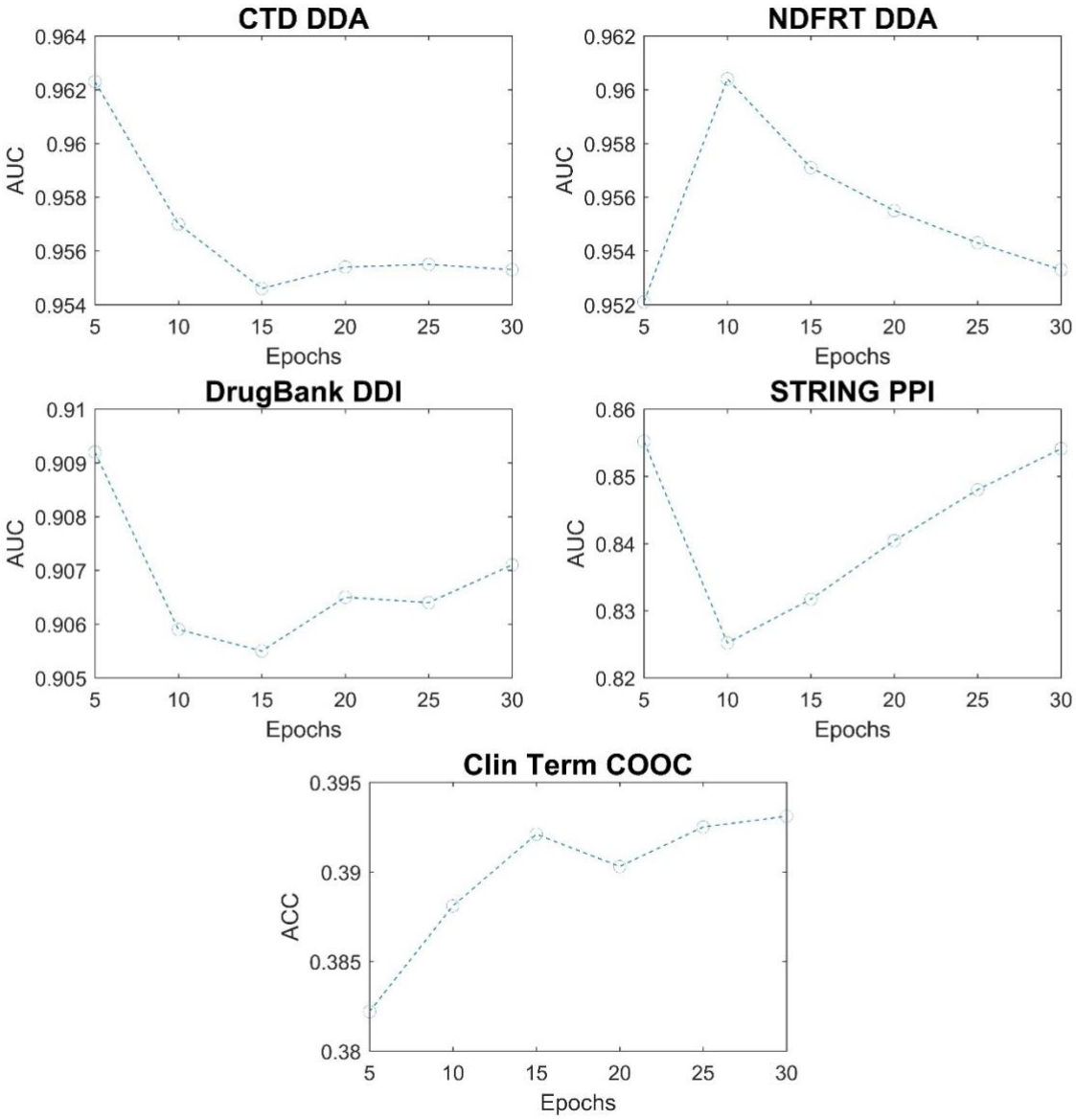


Fig. S7: The influence of the main hyper-parameters: α and β on *SDNE*.

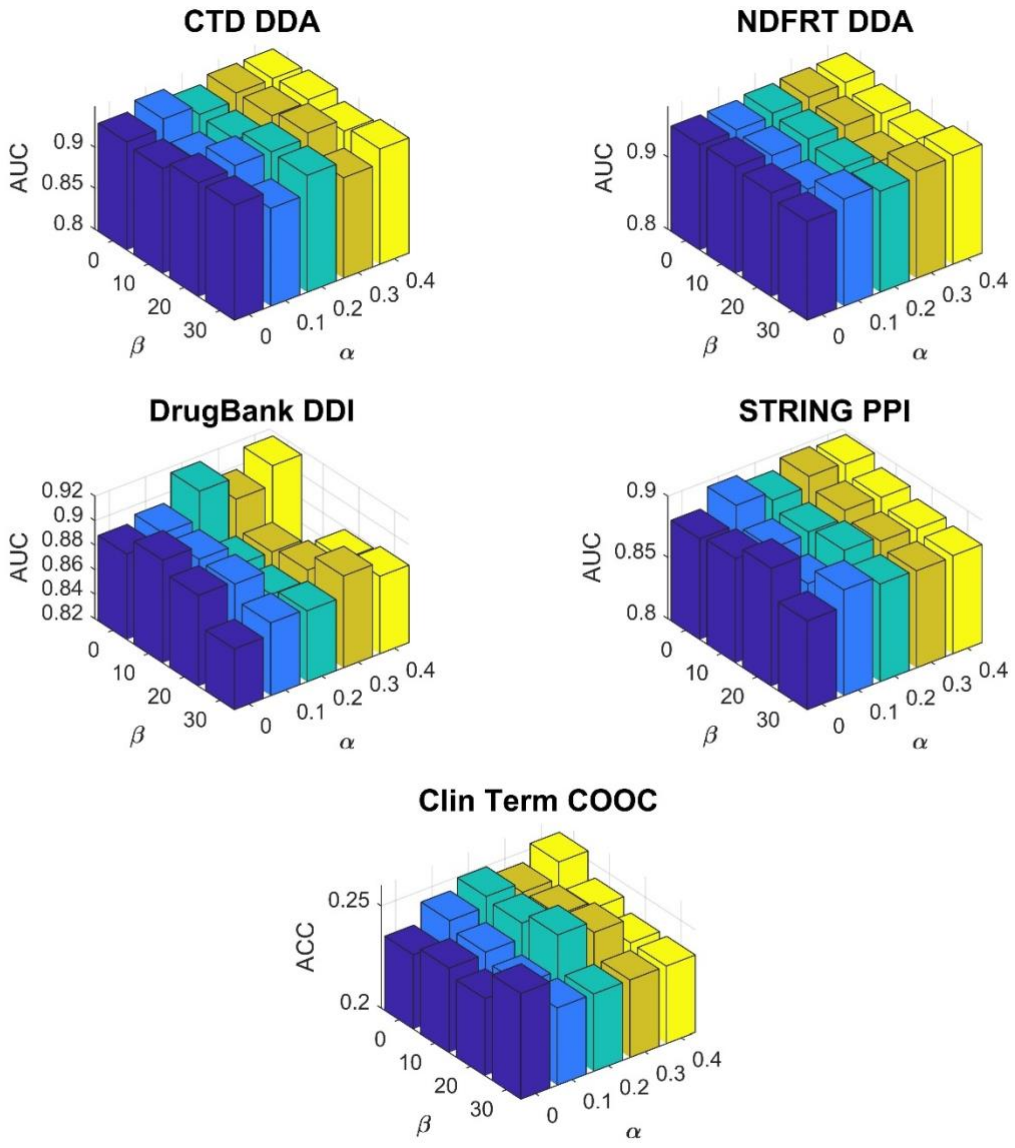


Fig. S8: The influence of the main hyper-parameter: *hidden units* on *GAE*.

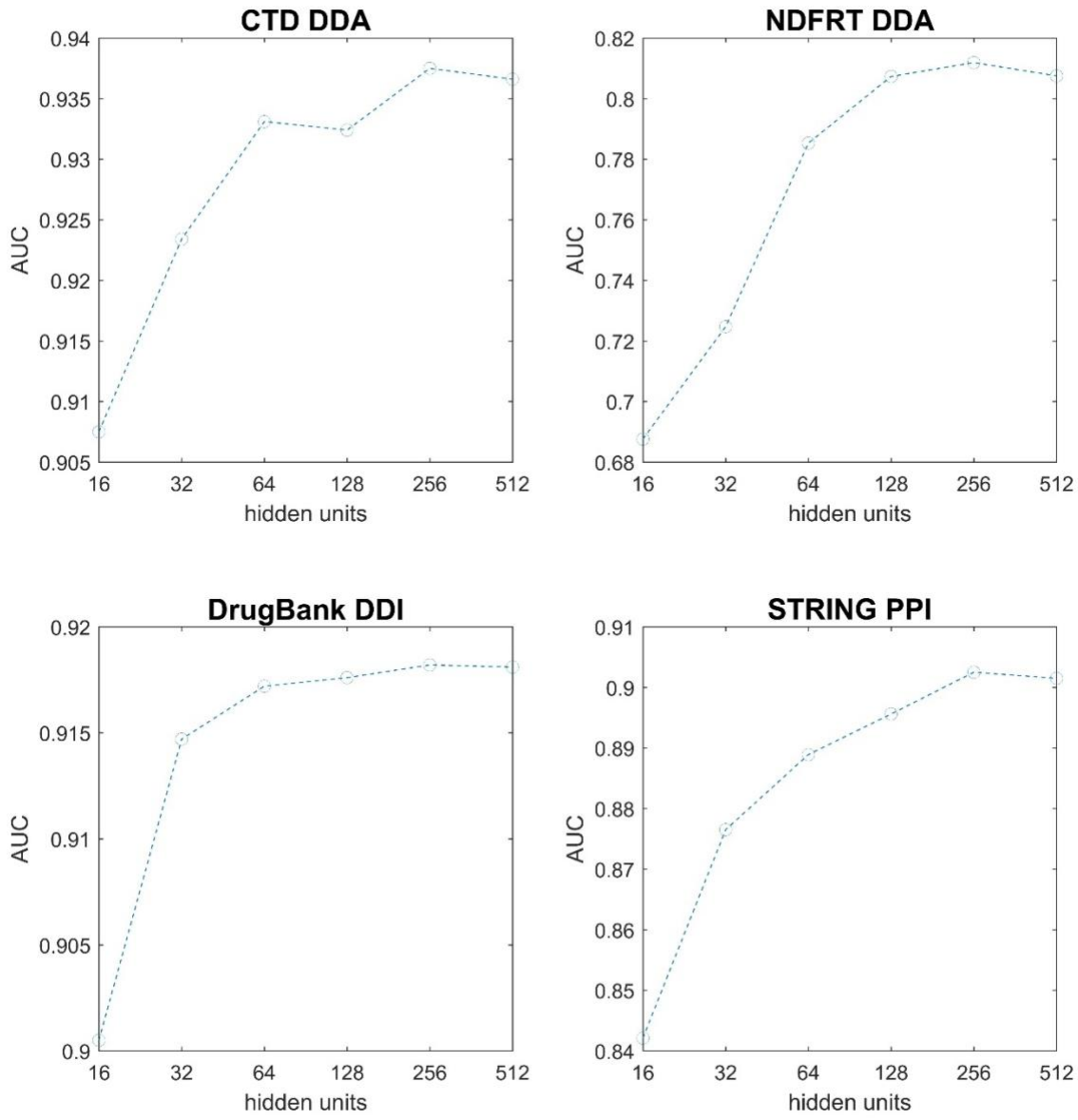
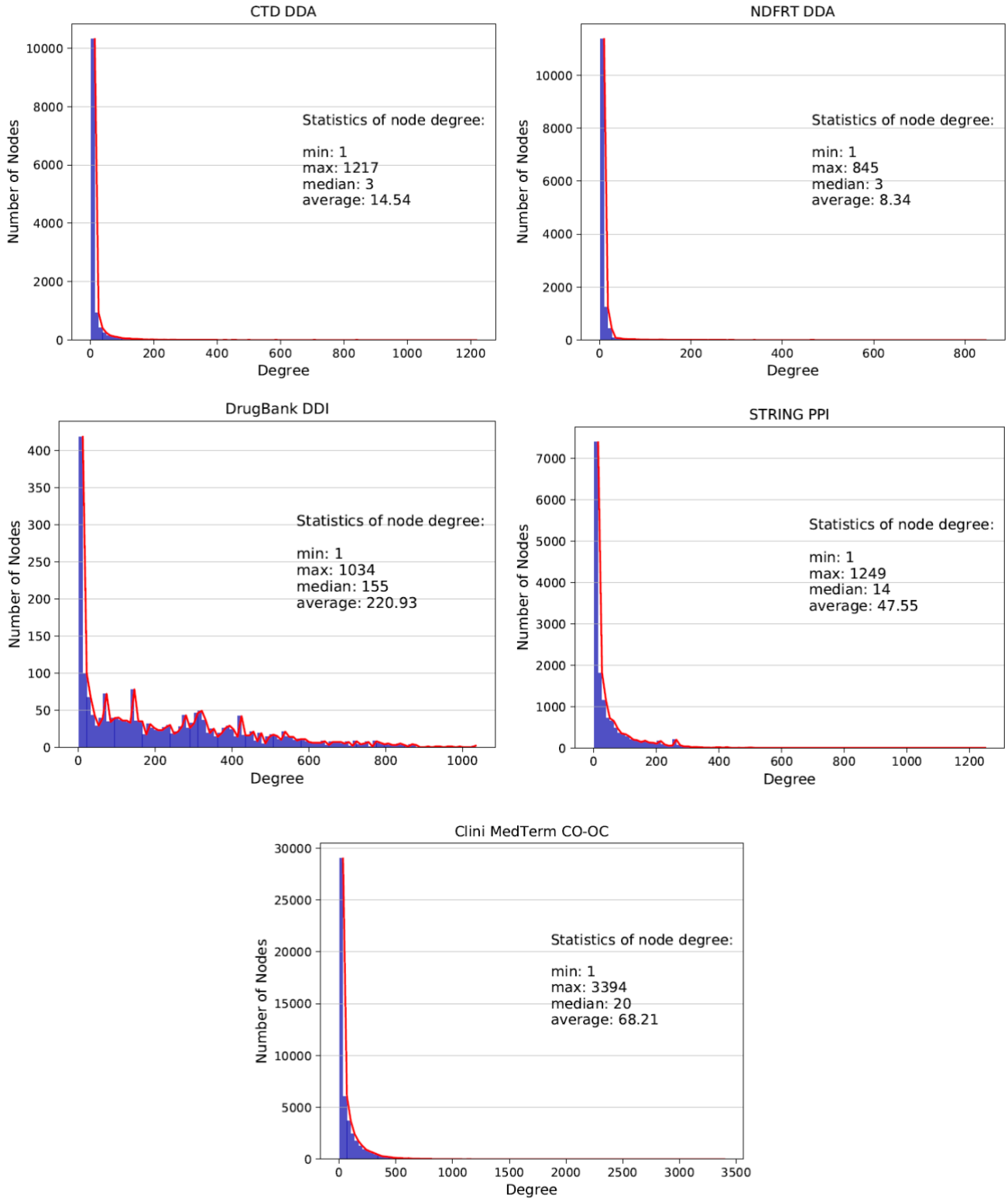


Table S2: Hyper-parameters set for different embedding methods in Table 3 and Table 4 of the main manuscript.

Method name	CTD DDA	NDFRT DDA	DrugBank DDI	STRING PPI	Clin Term COOC
GraRep	k-step = 4	k-step = 6	k-step = 2	k-step = 2	k-step = 2
DeepWalk	number-walks=8, walk-length=64	number-walks=8, walk-length=32	number-walks=128, walk-length=256	number-walks=256, walk-length=64	number-walks=256, walk-length=256
node2vec	p=2, q=0.25	p=0.25, q=0.25	p=2, q=0.25	p=1, q=4	p=2, q=0.25
struc2vec	number-walks=256, walk-length=32	number-walks=256, walk-length=64	number-walks=128, walk-length=64	number-walks=64, walk-length=16	number-walks=128, walk-length=256
LINE	epochs=5	epochs=10	epochs=5	epochs=5	epochs=30
SDNE	α=0.3, β=20	α =0.4, β =0	α =0.4, β =0	α =0.3, β =0	α =0.2, β =20
GAE	hidden units=256	hidden units=256	hidden units=256	hidden units=256	-

2 Datasets

Fig. S9: Node degree histograms of five compiled datasets.



3 Performance of “fine-tuning”

Table S3: Empirical results of “fine-tuning” on CTD DDA graph.

		AUC	ACC	F1
DeepWalk	w/o pre-train	0.9311	0.8599	0.8606
	w/ pretrain	**0.9368**	**0.8668**	**0.8670**
LINE	w/o pre-train	0.9467	0.8768	0.8752
	w/ pretrain	**0.9518**	**0.8856**	**0.8844**

4 Implementation Details

4.1 Hardware and Software

We train all embeddings using *Ohio Supercomputer Center (OSC)* Linux servers with 24 cores Dell Intel Xeon E5-2680 v4 machine, 128GB main memory. For algorithms which need GPUs (i.e., LINE, SDNE, GAE), we use NVIDIA Tesla P100 units with 16GB memory.

To make fairly comparison, all the algorithms are executed using Python 3.6. Towards that, we use a public graph embedding learning python package: OpenNE for Laplacian Eigenmaps, HOPE, GF, DeepWalk, LINE and SDNE. According to the instructions of OpenNE, for some embedding methods (e.g., LINE, node2vec), a little difference may exist between the original implementations and theirs.

And the required python packages are listed below (see details in our released package **BioNEV** on GitHub):

- numpy==1.14.0
- networkx==2.0
- scipy==0.19.1
- tensorflow==1.10.0
- gensim==3.0.1
- scikit-learn==0.19.0
- tqdm==4.28.1

4.2 Experimental settings for comparison with state-of-the-arts in Section 4.3 of the main manuscript.

For the comparison with LRSSL, we build a single hidden layer (128 hidden units) Multi-layer Perceptron (MLP) for each embedding method. All the hyper-parameters for the MLP is set to default in *scikit-learn*.

For the comparison with DeepDDI, we try 4 different classifiers: Navie Bayes, Linear SVM, Logistic Regression and Deep Neural Network (DNN). For the Navie Bayes, Linear SVM, Logistic Regression, all the hyper-parameters are set to default in *scikit-learn.* For DNN, we implement it using Keras in tensorflow. The DNN architecture and its hyper-parameters are the same as the ones in the original paper. The details are listed below:

Architecture	8 hidden layers, 2048 hidden units for each layer	
Hyperparameters	Activation function	Rectified linear unit
	Batch normalization	Yes
	Batch size	256
	Learning rate	0.0001
	Optimizer	Adam
	Weight initialization	Uniform