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 and Computer Engineering, The Ohio State University, Columbus, OH, USA,

[3]Research Information Solutions and Innovation, The Research 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

From Page 1-9, 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 three *link prediction* task datasets: NDFRT DDA, DrugBank DDI and STRING PPI (the results of CTD DDA is included in the main manuscript).

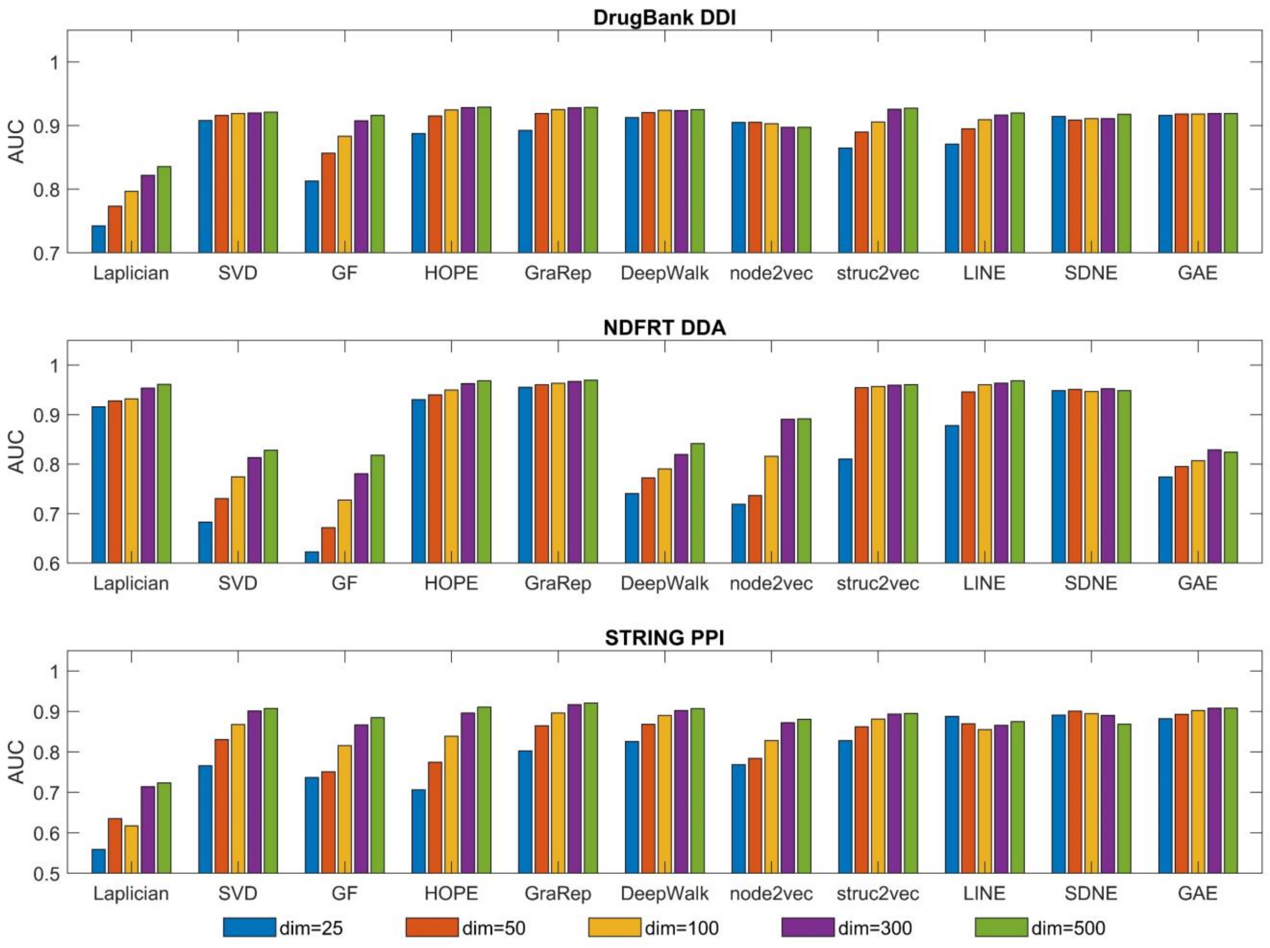


Fig. S2: The influence of *dimensionality* on different embedding methods on three *node classification* task datasets: Clini COOC, Node2vec PPI and MashUp PPI datasets.

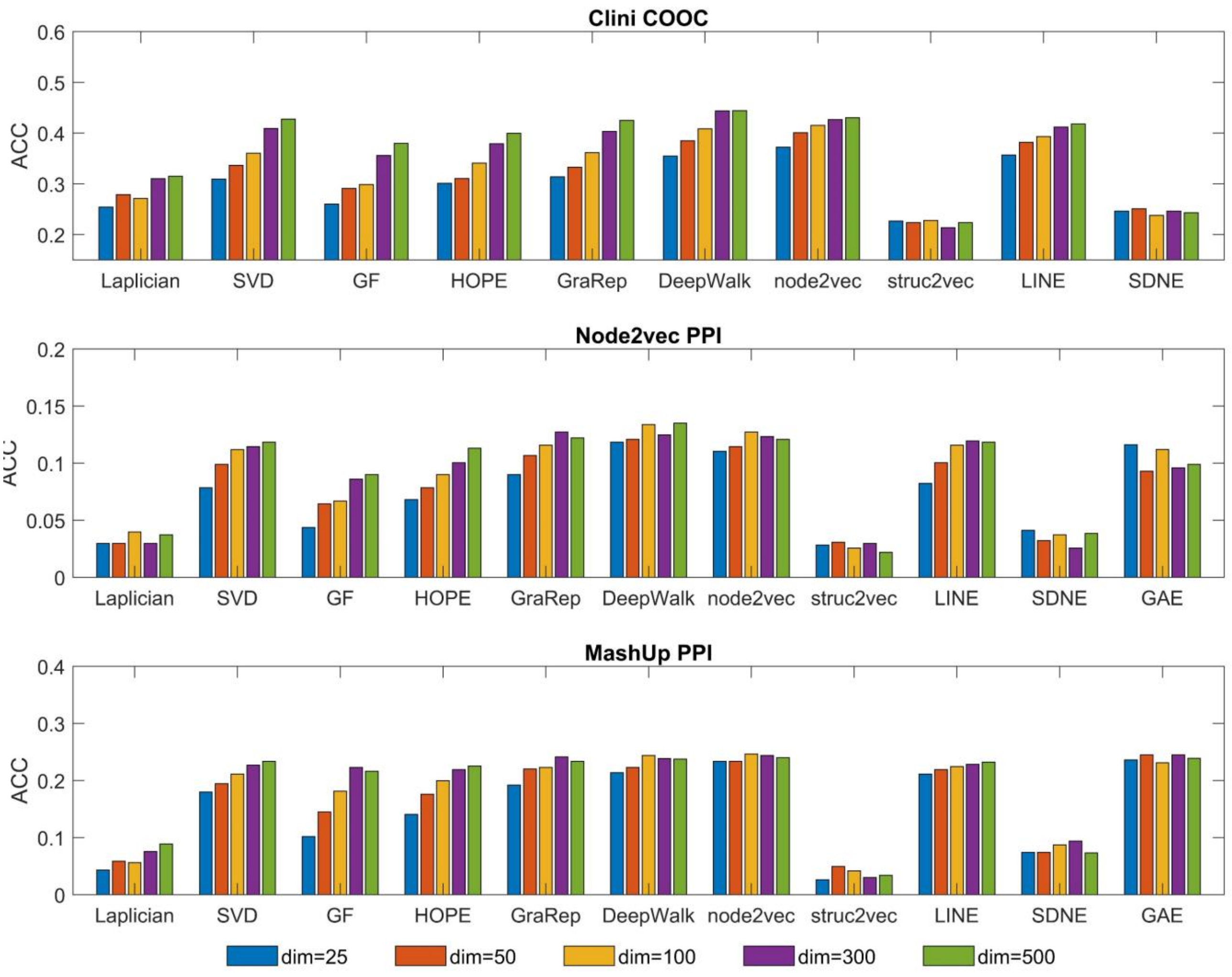


Fig. S3: The influence of the main hyper-parameter: *Ksteps* on *GraRep*.

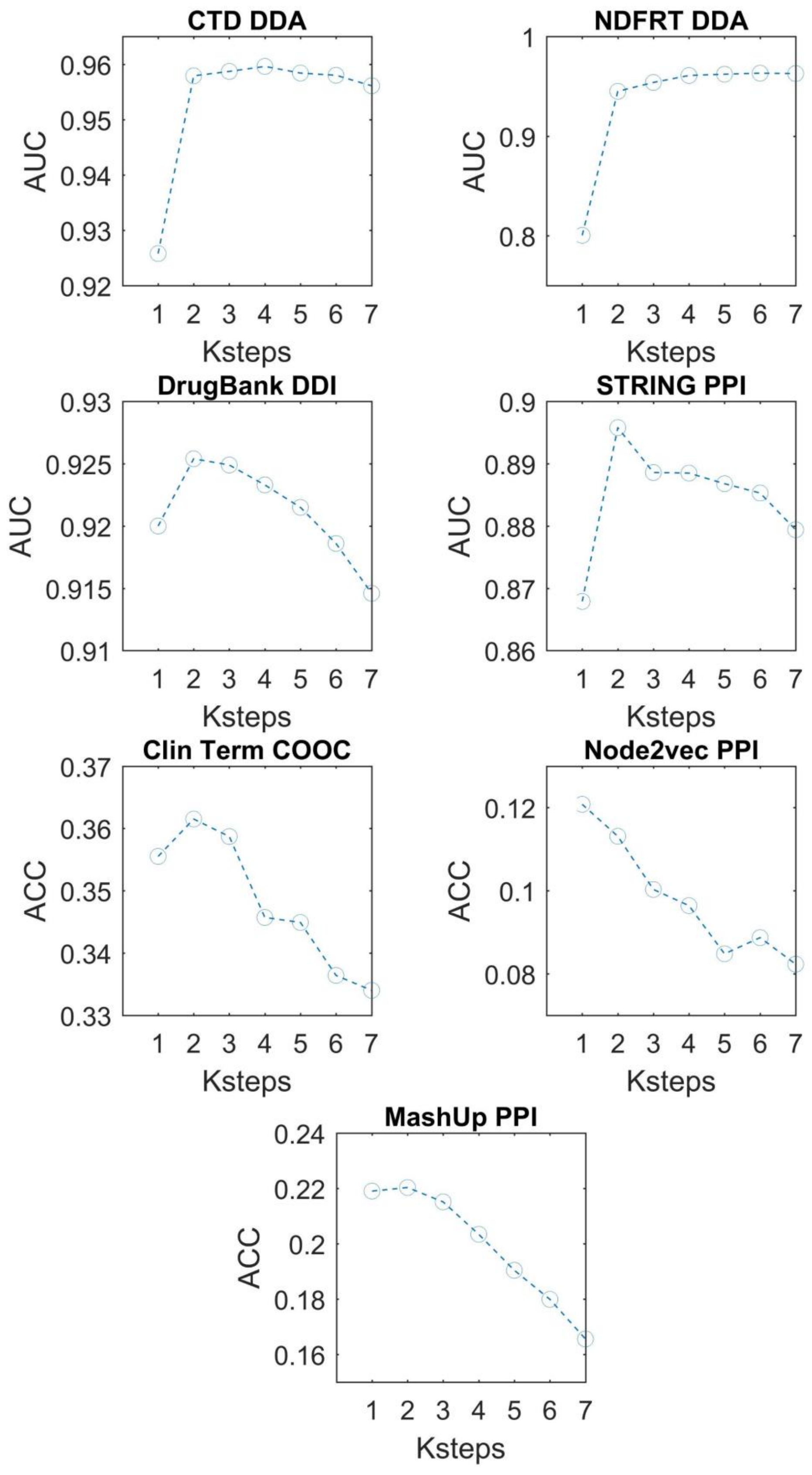


Fig. S4: The influence of the main hyper-parameters: *number of walks* and *walk length* on *DeepWalk*.

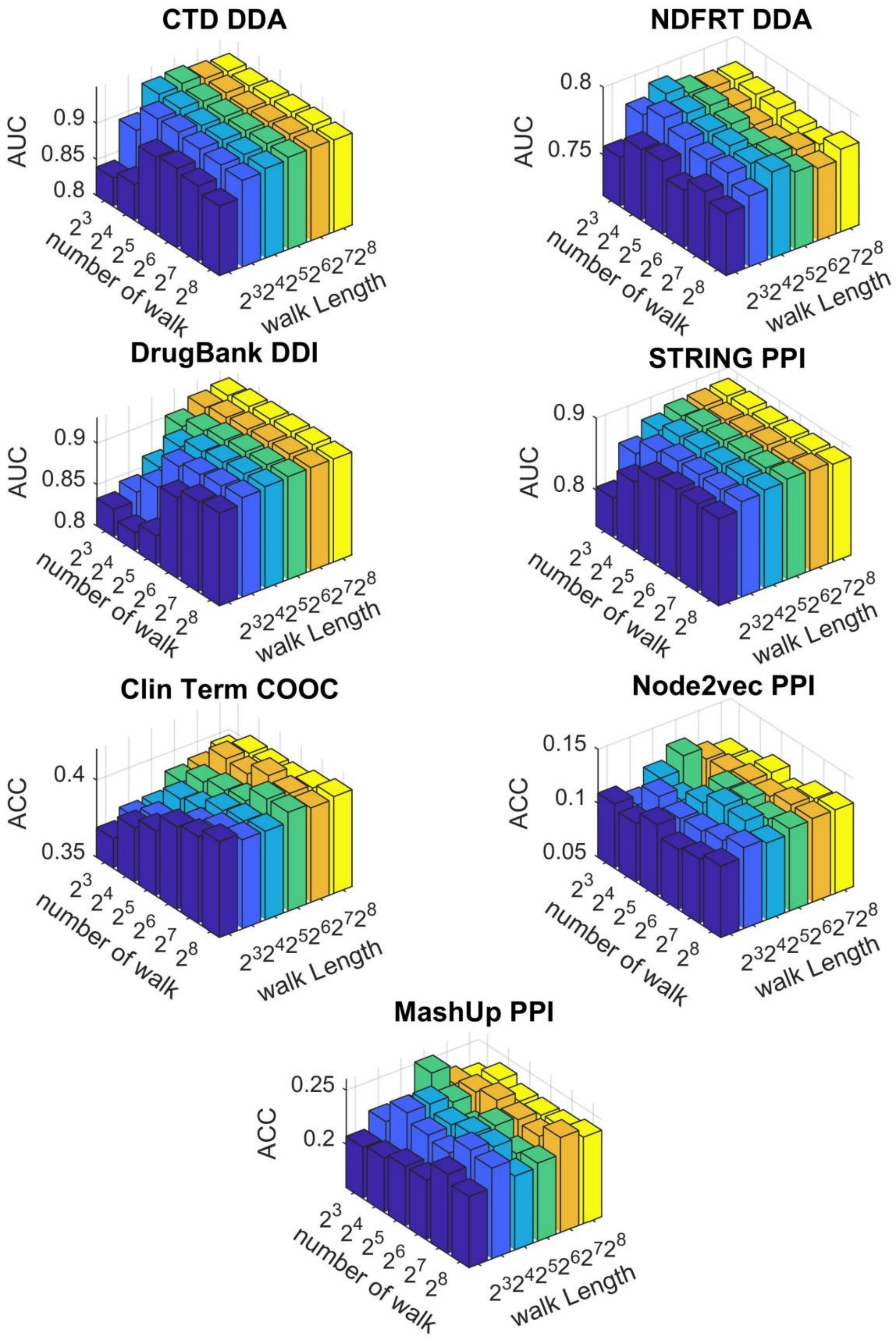


Fig. S5: The influence of the main hyper-parameters: *p* and *q* on *node2vec*.

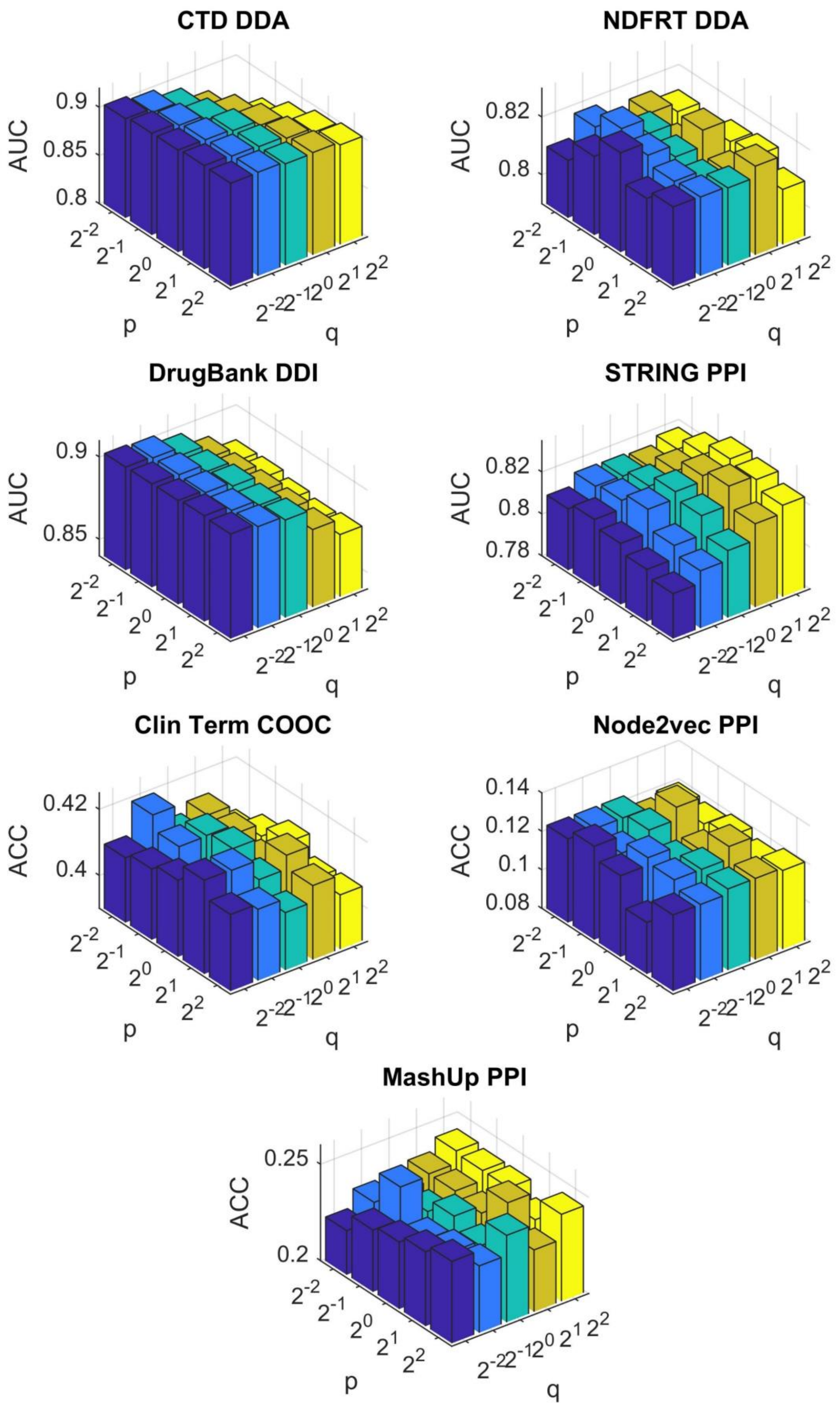


Fig. S6: The influence of the main hyper-parameters: ***number of walks*** and ***walk length*** **on *struc2vec*.**

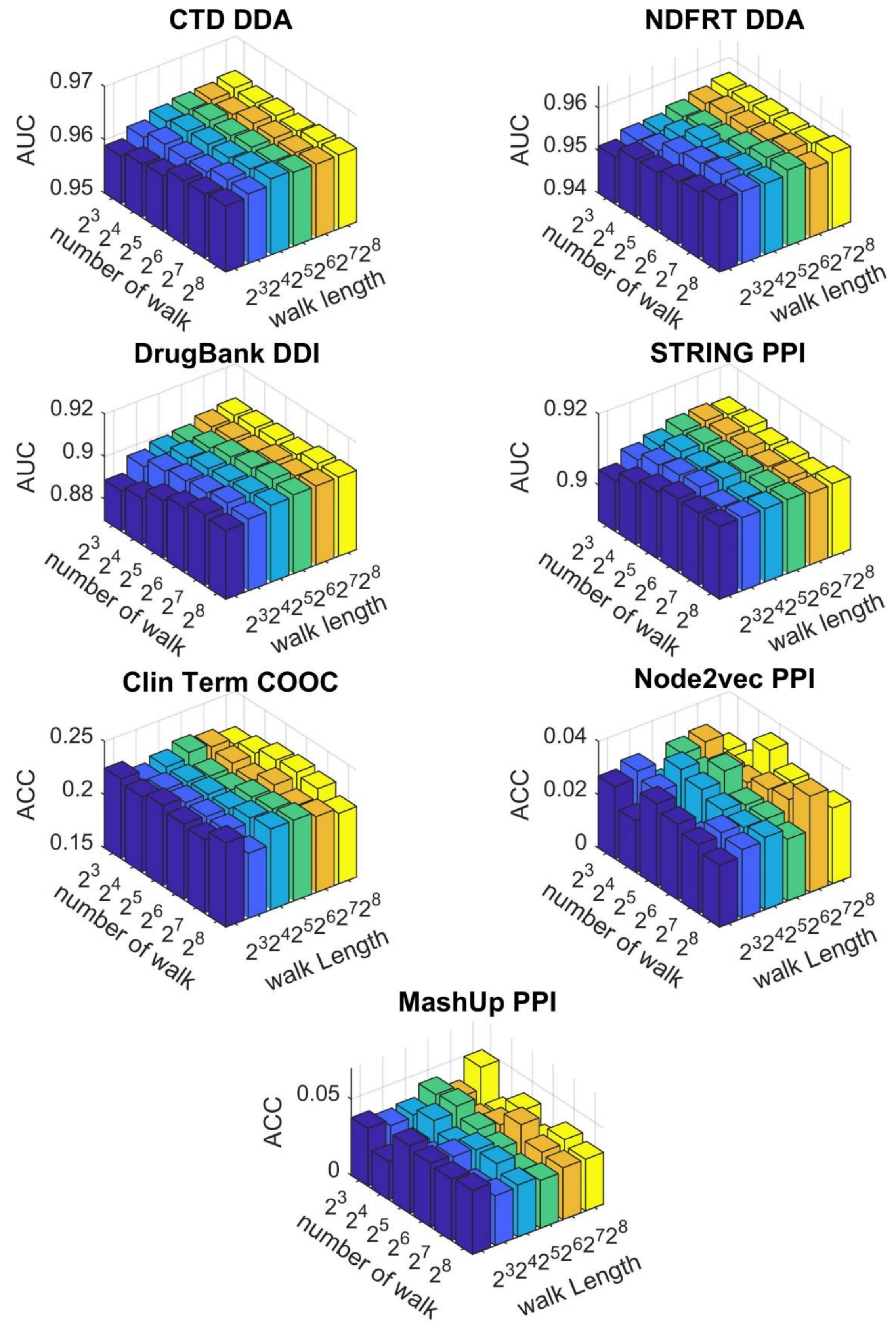
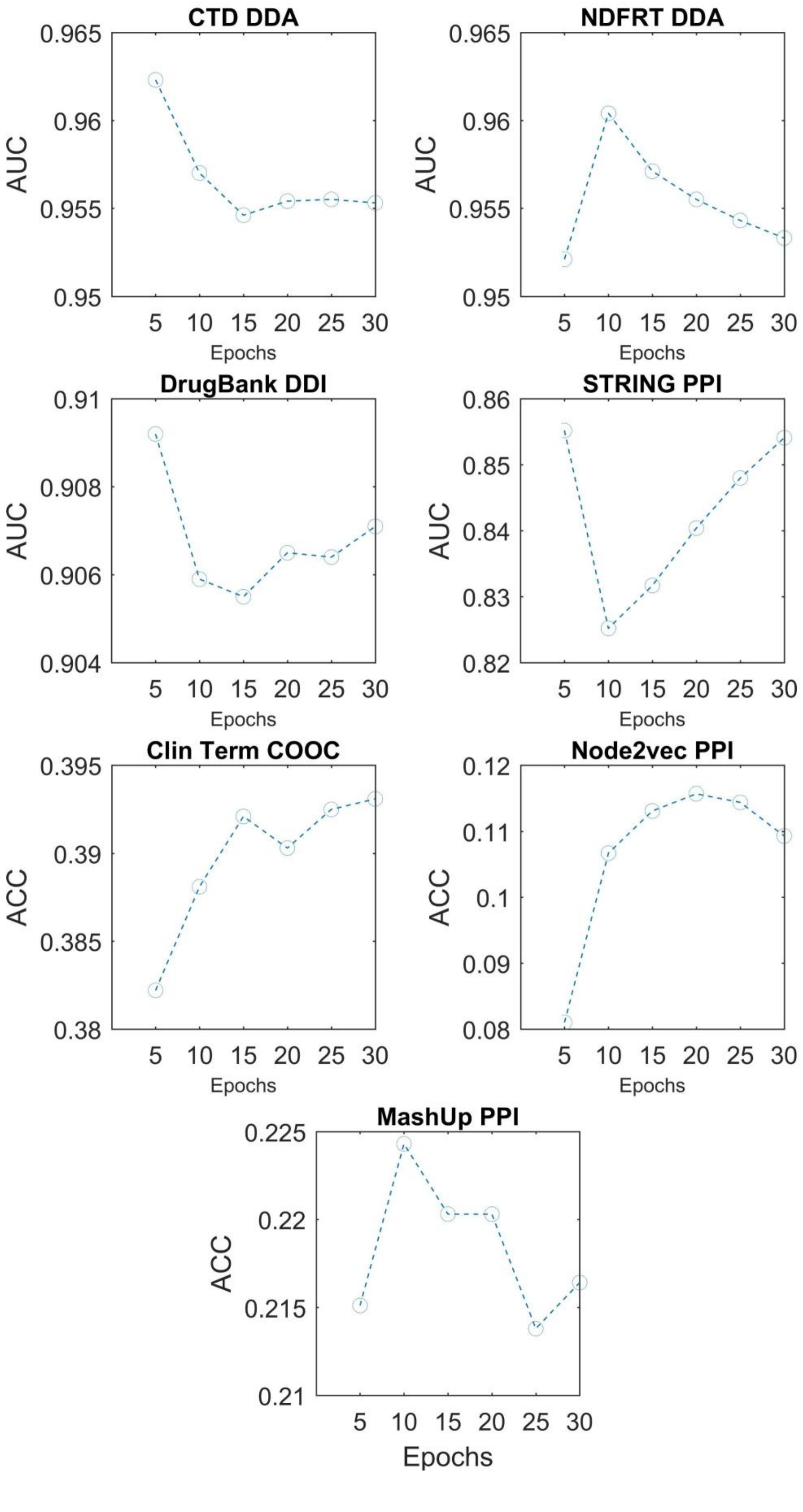


Fig. S7: The influence of the main hyper-parameter: *epochs* on *LINE*.



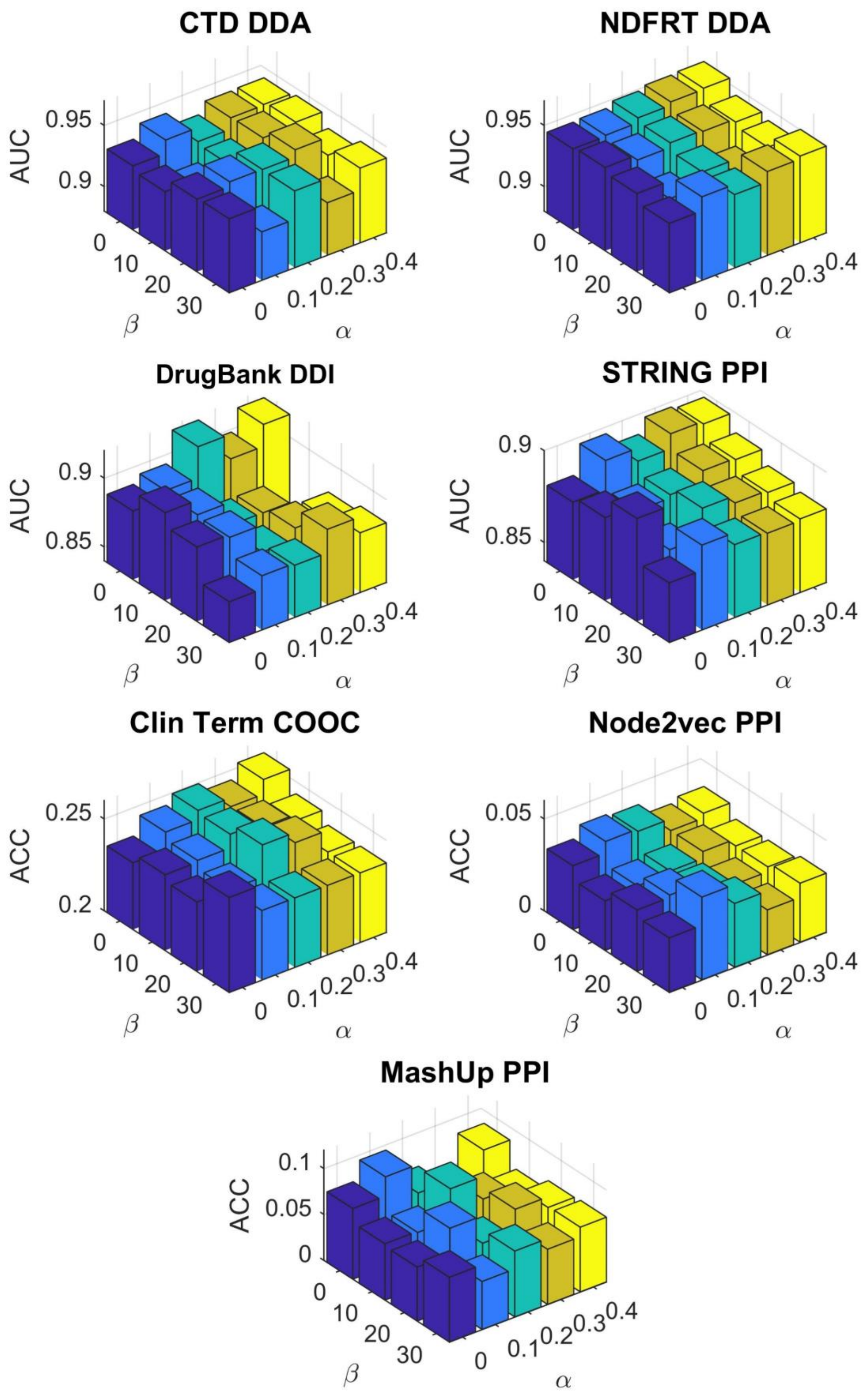


Fig. S8: The influence of the main hyper-parameters: *α* and *β* on *SDNE*.

Fig. S9: The influence of the main hyper-parameter: *hidden units* on *GAE*.

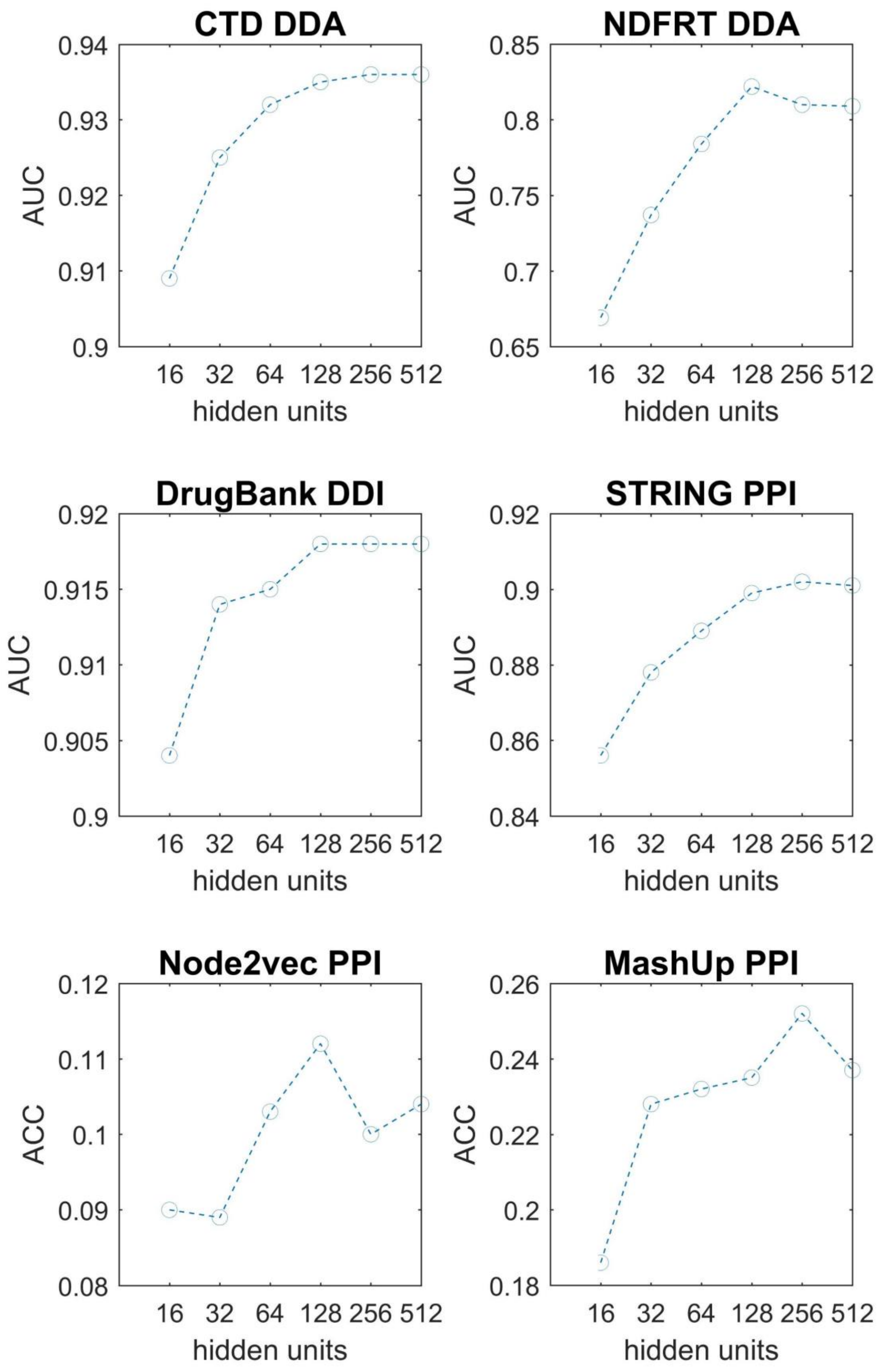


Table S1: Hyper-parameters set for different embedding methods in Table 3 and Table 4 of the main manuscript.

	Link Prediction				**Node Classification**		
Method name	**CTD DDA**	**NDFRT DDA**	**DrugBank DDI**	**STRING PPI**	**Clin Term COOC**	**Node2vec PPI**	**MashUp PPI**
GraRep	k-step = 4	k-step = 6	k-step = 2	k-step = 2	k-step = 2	k-step = 1	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=64, walk-length=128	number-walks=64, walk-length=32	number-walks=8, walk-length=64
node2vec	p=2, q=0.25	p=1, q=0.25	p=2, q=0.25	p=2, q=2	p=2, q=0.25	p=0.5, q=2	p=0.5, q=0.5
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=64, walk-length=256	number-walks=256, walk-length=128	number-walks=8, walk-length=256
LINE	epochs=5	epochs=10	epochs=5	epochs=5	epochs=30	epochs=20	epochs=10
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	α=0.1, β=30	α=0.1, β=0
GAE	hidden units=512	hidden units=128	hidden units=128	hidden units=256	-	hidden units=128	hidden units=256

Table S2: Performance of different embedding methods on CTD DDA, NDFRT DDA, DrugBank DDI and STRING PPI datasets of Link Prediction

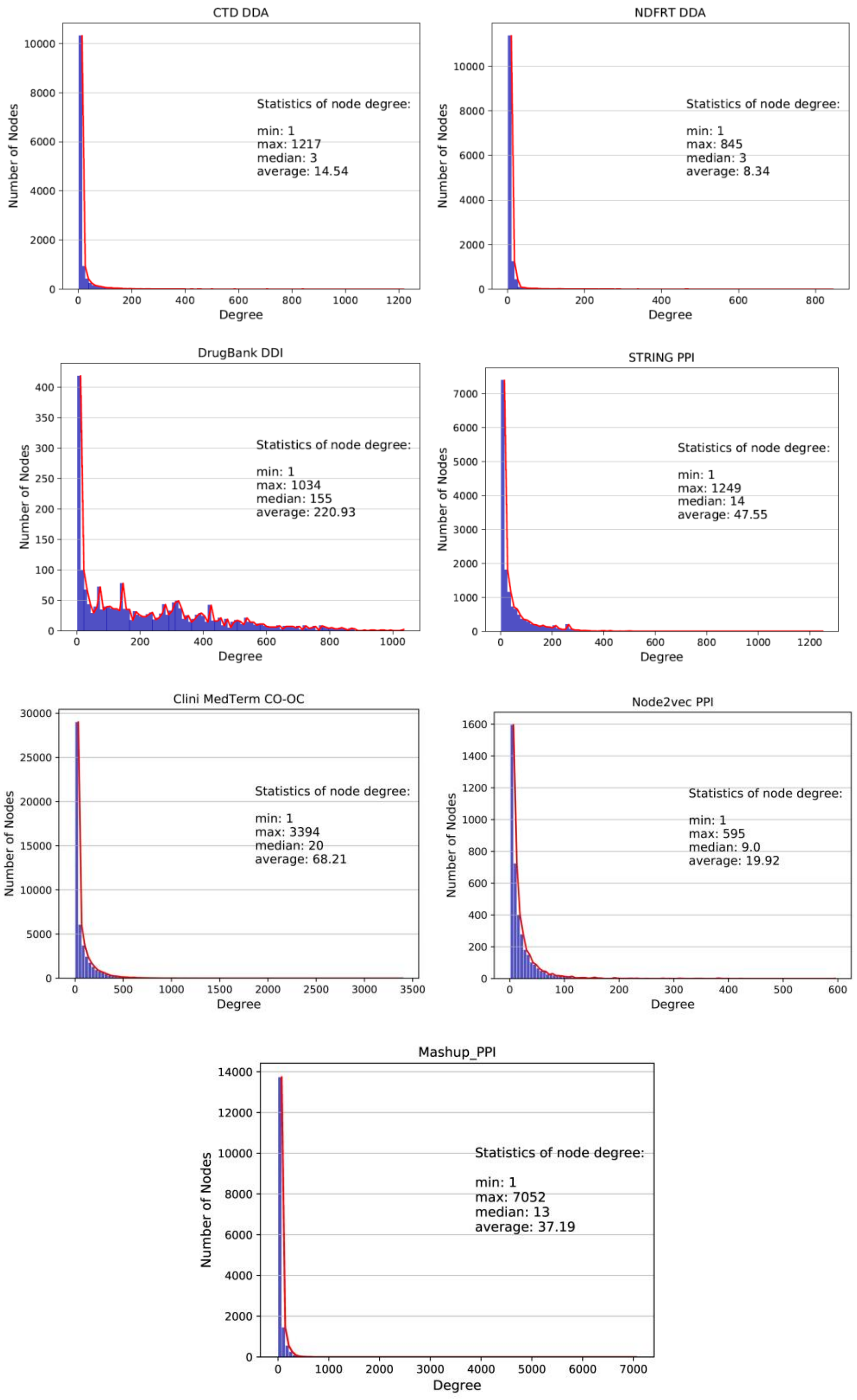
Method Category		Method Name	CTD DDA			NDFRT DDA			DrugBank DDI			STRING PPI		
			AUC	ACC	F1	AUC	ACC	F1	AUC	ACC	F1	AUC	ACC	F1
Traditional	Matrix Factorization Based	Laplician	0.856±0.004	0.793±0.003	0.802±0.003	0.930±0.003	0.917±0.004	0.921±0.004	0.796±0.002	0.720±0.002	0.729±0.002	0.639±0.021	0.596±0.016	0.586±0.017
		SVD	0.936±0.002	0.855±0.002	0.854±0.002	0.779±0.003	0.707±0.004	0.700±0.005	0.919±0.001	0.837±0.001	0.837±0.001	0.867±0.001	0.794±0.001	0.790±0.001
		GF	0.884±0.004	0.808±0.004	0.805±0.004	0.720±0.006	0.660±0.006	0.655±0.007	0.882±0.003	0.802±0.003	0.81±0.003	0.817±0.005	0.746±0.005	0.747±0.005
Recently Proposed		HOPE	0.951±0.001	0.886±0.002	0.887±0.002	0.949±0.001	0.928±0.002	0.931±0.002	0.923±0.001	0.844±0.002	0.846±0.002	0.839±0.001	0.764±0.001	0.764±0.001
		GraRep	0.960±0.001	0.899±0.002	0.900±0.002	0.963±0.001	0.931±0.002	0.934±0.002	0.925±0.001	0.845±0.001	0.846±0.001	0.894±0.001	0.823±0.001	0.822±0.001
	Random Walk Based	DeepWalk	0.929±0.002	0.866±0.002	0.864±0.002	0.783±0.004	0.710±0.004	0.709±0.005	0.921±0.001	0.840±0.002	0.842±0.002	0.884±0.001	0.813±0.002	0.814±0.002
		node2vec	0.911±0.002	0.838±0.002	0.835±0.002	0.819±0.005	0.742±0.005	0.741±0.006	0.902±0.001	0.819±0.001	0.819±0.001	0.828±0.003	0.758±0.003	0.756±0.003
		struc2vec	0.965±0.001	0.903±0.002	0.903±0.002	0.958±0.001	0.913±0.002	0.912±0.002	0.904±0.001	0.826±0.002	0.83±0.002	0.909±0.001	0.838±0.001	0.841±0.001
	Deep Learning Based	LINE	0.965±0.001	0.904±0.001	0.904±0.001	0.962±0.002	0.934±0.002	0.935±0.001	0.905±0.002	0.825±0.002	0.829±0.002	0.859±0.003	0.788±0.003	0.795±0.003
		SDNE	0.935±0.010	0.863±0.012	0.861±0.013	0.944±0.004	0.896±0.007	0.897±0.007	0.911±0.006	0.833±0.007	0.838±0.006	0.884±0.008	0.813±0.009	0.814±0.009
		GAE	0.937±0.001	0.857±0.002	0.856±0.002	0.813±0.007	0.735±0.006	0.730±0.007	0.917±0.001	0.836±0.001	0.840±0.001	0.900±0.001	0.827±0.001	0.829±0.002

Table S3: Performance of different embedding methods on Clini COOC, Node2vec PPI, MashUp PPI datasets of Node Classification

Method Catergory		Method Name	Clini COOC			Node2vec PPI			Mashup PPI		
			Accuracy	Micro-F1	Macro-F1	Accuracy	Micro-F1	Macro-F1	Accuracy	Micro-F1	Macro-F1
Traditional	**Matrix Factorization Based**	**Laplician**	0.277±0.007	0.313±0.005	0.073±0.002	0.039±0.007	0.101±0.008	0.070±0.007	0.058±0.007	0.132±0.009	0.107±0.008
		SVD	0.362±0.006	0.420±0.005	0.186±0.007	0.119±0.012	0.228±0.011	0.179±0.011	0.223±0.012	0.347±0.014	0.297±0.014
		GF	0.304±0.006	0.352±0.007	0.143±0.009	0.078±0.010	0.168±0.011	0.121±0.011	0.178±0.016	0.290±0.015	0.237±0.016
Recently Proposed		**HOPE**	0.340±0.006	0.395±0.005	0.163±0.006	0.102±0.009	0.208±0.011	0.152±0.011	0.205±0.013	0.322±0.013	0.266±0.013
		GraRep	0.370±0.007	0.424±0.006	0.177±0.005	0.127±0.012	0.238±0.010	0.193±0.013	0.216±0.014	0.334±0.011	0.283±0.011
	Random Walk Based	**DeepWalk**	0.414±0.006	0.472±0.005	0.227±0.007	0.129±0.012	0.243±0.011	0.194±0.011	0.231±0.012	0.357±0.011	0.311±0.012
		node2vec	0.420±0.006	0.479±0.005	0.231±0.010	0.129±0.010	0.243±0.009	0.190±0.011	0.241±0.014	0.367±0.012	0.313±0.013
		struc2vec	0.220±0.005	0.253±0.006	0.038±0.001	0.038±0.006	0.094±0.006	0.061±0.004	0.042±0.007	0.120±0.010	0.087±0.008
	Deep Learning Based	**LINE**	0.398±0.005	0.453±0.006	0.205±0.008	0.122±0.011	0.236±0.011	0.176±0.012	0.227±0.017	0.352±0.017	0.296±0.017
		SDNE	0.242±0.008	0.271±0.016	0.042±0.007	0.037±0.006	0.098±0.010	0.047±0.007	0.087±0.012	0.178±0.013	0.109±0.012
		GAE	-	-	-	0.120±0.014	0.237±0.014	0.186±0.014	0.234±0.013	0.358±0.013	0.307±0.014

2 Datasets

Fig. S10: Node degree histograms of seven compiled datasets.



3 Performance of “fine-tuning”

Table S4: 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, 4.4 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

For the comparison with Mashup, we run the original code provided by the authors to obtain embeddings for each protein. Then, we build the same classifier as other graph embedding methods to classify the protein functions.