Topic 4: Graph Mining - *VQNE: Variational Quantum Network Embedding with Application to Network Alignment* - Implementation

Wai Leuk Lo
Student ID: 20153384
Group 3
wlloaa@connect.ust.hk

Abstract

We choose Topic 4: Graph Mining - *VQNE: Variational Quantum Network Embedding with Application to Network Alignment* as our implementation project. We introduce the problem and the algorithms, present the datasets and implementation details, and finally analyze the evaluation results and do some case studies.

1 Introduction

A (social) network can be modeled as a graph with the node representing the users and edges representing a relation between two users. Studying networks based purely on their structures not only can reveal useful information about the networks but also avoid privacy issues originated from individual node attributes. Learning the node embedding from the network structure has been a well-established research area, and among the downstream tasks, network alignment tries to find the correspondence between the nodes of two networks based on a few labeled correspondences. The alignment can then be used to predict cross-network links and make recommendations.

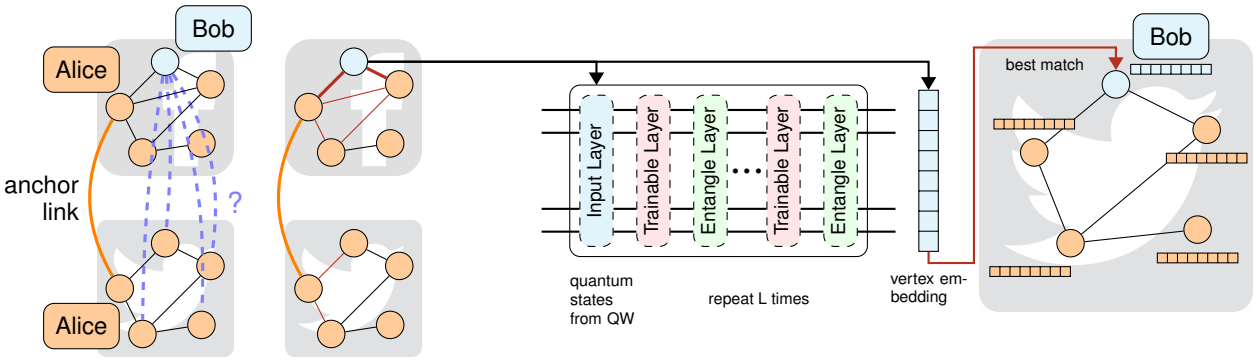
Both classical and quantum algorithms exist for network alignment, but each have their drawbacks. The classical algorithms often lacks scalability due to NP-hardness, while existing pure quantum or quantum-classical hybrid embedding methods either require an exponential number of qubits with respect to the feature dimensions, or suffers from performance issues due to data being passed between quantum and classical devices. The paper (Ye et al., 2023) introduces a paradigm that combines discrete-time quantum walk (QW) and quantum embedding ansatz to obtain a pure quantum network embedding algorithm named variational quantum network embedding (VQNE) that tackles the problems mentioned above.

2 Methodology

In this project, we implement the VQNE and train it on the Facebook-Twitter (Fb-Tt) and Weibo-Douban (Wb-Db) datasets. The results will be compared with the baseline method BTWalk and the VQNE implemented by the original paper (Ye et al., 2023). In the following, we will first introduce the concept of graphs and our problem setup. Then we describe the algorithm for merging two graphs and performing the SQW among them, in preparation for network embedding. Finally we introduce the quantum embedding ansatz and the objective functions for training the quantum network.

2.1 Problem Setup and Model Overview

A network is defined as a graph $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. Given a source network G^s, a target network G^t and anchor links $\mathcal{T}_{train} \subset \mathcal{V}^s \times \mathcal{V}^t$ containing partial node alignments between G^s and G^t, network alignment aims to predict the unknown anchor links in $\mathcal{T}_{test} \subset \mathcal{V}^s \times \mathcal{V}^t$. VQNE is decomposed into two algorithms, where the first one merges the two networks G^s and G^t using the anchor links $\mathcal{T}_{train}$ to obtain a network G, and the second one performs Szegedy quantum walks (SQW) (Szegedy, 2004) on each vertex of G to obtain an evolved quantum state before feeding it to a quantum embedding ansatz for transformations and entanglements. The output quantum state is the embedding vector for the vertex then used to calculate the loss function, which aims to minimize the difference between the embedding vectors of the same vertex while maximizing that between different vertices. A schematic diagram for the work flow is given in Fig. 1.



(a) Network Alignment (b) Quantum Walk (c) Quantum Embedding Ansatz (d) Find Best Match

Figure 1: Schematic Diagram of the VQNE Model

2.2 Merging Graphs and Discrete-time Quantum Walk

Merging two graphs via the anchor links entails expanding the source graph G^s by adding all edges from the target graph G^t while only adding those vertices from G^t that are not anchor links, and the precise procedures are given in Alg. 1.

For the merged graph $G = (\mathcal{V}, \mathcal{E})$ with the number of vertices $N = |\mathcal{V}|$, we can define an $N \times N$ adjacency matrix A by:

$$A_{i,j} = \begin{cases} 1 & \text{if } (v_i, v_j) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases}$$

The transition matrix P for graph G can then be defined as:

$$P_{i,j} = \frac{A_{i,j}}{d_j},$$

where $d_j = \sum_{i=1}^{N} A_{i,j}$ is the in-degree of the vertex v_j.

The SQW is defined as a unitary operation on the Hilbert space $\mathcal{H} = \mathcal{H}^N \otimes \mathcal{H}^N$ where the state vectors representing the vertices live. A quantum state $|\Psi\rangle \in \mathcal{H}$ can be written in a superposition state:

$$|\Psi\rangle = \sum_{i=0}^{N-1} \sum_{i=0}^{N-1} a_{ij} |i\rangle \otimes |j\rangle = \sum_{i=0}^{N-1} \sum_{i=0}^{N-1} a_{ij} |ij\rangle ,$$

where $|ij\rangle$ represents the state where SQW currently reaches vertex i and will move to vertex j in the next step, and $|a_{ij}|^2$ represents the probability of the state $|ij\rangle$. Then the projector states $|\psi_i\rangle$ of the Markov chain is defined as:

$$|\psi_i\rangle = |i\rangle \otimes \sum_{j=0}^{N-1} \sqrt{P_{j,i}} |j\rangle , \quad i \in \{0, \ldots, N-1\}.$$

Algorithm 1: The process of merging two networks

Input: Source network $G^s = (\mathcal{V}^s, \mathcal{E}^s)$ and target network $G^t = (\mathcal{V}^t, \mathcal{E}^t)$, ground-truth mapping $\pi : \mathcal{V}^t \to V^s$, anchor links $\mathcal{T}_{train} = \{\{p, q\} | p \in \mathcal{V}^s, q \in \mathcal{V}^t\}$;
Output: Merged network $G = (\mathcal{V}, \mathcal{E})$;
Initialize G as G^s, meaning $\mathcal{V} = \mathcal{V}^s$ and $\mathcal{E} = \mathcal{E}^s$;
foreach *edge* $(v_i^t, v_j^t) \in \mathcal{E}^t$ *in* G^t **do**
  if $(\pi(v_i^t), v_i^t) \notin \mathcal{T}_{train}$ **then**
    $\mathcal{V}.append[v_i^t]$, and $\mathcal{E}.append[(v_i^t, v_j^t)]$;
    if $(\pi(v_j^t), v_j^t) \notin \mathcal{T}_{train}$ **then**
      $\mathcal{V}.append[v_j^t]$;
    end
  else
    if $(\pi(v_j^t), v_j^t) \notin \mathcal{T}_{train}$ **then**
      $\mathcal{V}.append[v_j^t]$, and $\mathcal{E}.append[(v_i^t, v_j^t)]$;
    else
      if $(v_i^t, v_j^t) \notin \mathcal{E}$ **then**
        $\mathcal{E}.append[(v_i^t, v_j^t)]$;
      end
    end
  end
end
Result: Merged network G

The projection opeartor onto the space spanned by $\{|\psi_i\rangle\}_{i=0}^{N-1}$ is given by:

$$\Pi = \sum_{i=0}^{N-1} |\psi_i\rangle \langle\psi_i| ,$$

and the associated reflection operator is $C = 2\Pi - I$. A swap operator is defined as:

$$S = \sum_{i=0}^{N-1} \sum_{i=0}^{N-1} |i,j\rangle \langle i,j| .$$

At last, the one step SQW is defined as:

$$U_W = S(2\Pi - I) = SC,$$

and the t-step SQW is defined as U_W^t. The $|i,j\rangle$ state is implemented by an $N \times N$ matrix and the SQW is simulated by (Bai et al., 2017; Wang et al., 2022b):

$$|i,j\rangle \xrightarrow{U_W} (\frac{2}{d_j} - 1) |j,i\rangle + \frac{2}{d_j} \sum_{k\neq i, \{j,k\}\in\mathcal{E}} |j,k\rangle . \tag{1}$$

For a graph with N vertices, we can then use $n = \lceil \log_2 N \rceil$ qubits to represent the quantum walk.

There are two versions of discrete-time quantum walk (Wang et al., 2022b). One starts from a particular site $|\psi_i\rangle$ and performs a t-step SQW as given in Eq. 1 to obtain the quantum encoding state $|\psi_i^t\rangle$ for the i^{th} site. The quantum walk then consists of doing this for all sites $i, i = 0, ..., N-1$. This type of quantum walk focuses on the local features around a site (as it deals with one site at a time) and is called the biased quantum walk. The other version of quantum walk starts from a matrix $[|\psi_i\rangle]_{i=0}^{N-1}$, where each row is the initial state discussed above, and evolves the entire matrix according to Eq. 1. This captures the global behaviour of the whole graph (as a site receives contributions from potentially all other sites in each step of evolution) and we call it the un-biased quantum walk. We refer to the implementation of both versions[1] in (Wang et al., 2022b) and translate it into a PyTorch implementation, but select the un-biased one for training as it yields better performance.

2.3 Quantum Embedding Ansatz

After SQW, the quantum encoding state $|\Psi_i^t\rangle$ is input into the quantum embedding ansatz (Kandala et al., 2017) to learn the latent representation of vertex v_i. One block of the ansatz consists of a trainable parameter block U_l constructed by single-qubit gates (rotation gates in this paper) and an entanglement block U_{ent} constructed by two-qubit gates (CNOT gates in this paper):

$$U_l(\theta) = \bigotimes_{k=n+1}^{2n} (R_z(\theta_z^{(k,l)})) \times \bigotimes_{k=1}^{n} (R_y(\theta_y^{(k,l)})) \times \bigotimes_{k=1}^{n} (R_z(\theta_z^{(k,l)})),$$

where $\theta_z^{(k,l)}$ and $\theta_y^{(k,l)}$ are the trainable parameters for the rotation gates R_z and R_y at the l-th layer on the k-th qubit. The quantum embedding ansatz is obtained by applying the block L times:

$$U(\theta) = \prod_{l=1}^{L} (U_{ent} U_l(\theta)).$$

Note that the θ's are different for different layers and there is no parameter sharing. The embedding vector of the vertex v_i is obtained after L layers of ansatz:

$$|\Psi_i(L)\rangle = \prod_{l=1}^{L} (U_{ent} U_l(\theta)) |\Psi_i(0)\rangle .$$

The model with the unitary layers and the entanglement layers are illustrated in Fig. 2.

The training of VQNE is done by minimizing the Marginal Triplets (MT) objective (Chen et al., 2020; Xiong et al., 2021) and the Elastic Potential Energy (EPE) function (Xiong et al., 2023):

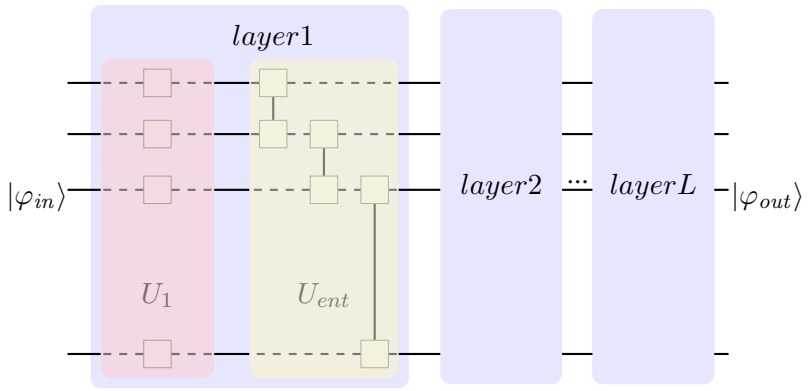
$$O_{MT} + \eta O_{EPE}, \tag{2}$$

and

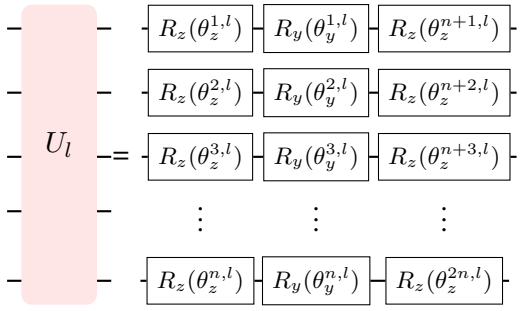
$$O_{MT} = \mathbb{E}_{(i,j)\sim\mathbb{P}_{pos}, j'\sim\mathbb{P}_{neg}} \max(x_i^T x_{j'} - x_i^T x_j + \gamma, 0),$$
$$O_{EPE} = \mathbb{E}_{(i,j)\sim\mathbb{P}_{pos}} ||x_i - x_j||_2^2,$$

where both x_i and x_j indicate the embedding vector of vertex v_i, while $x_{j'}$ indicates the embedding of another vertex randomly selected in the merged graph G. As there is only one match for each v_i in the entire graph (could be v_i itself for the anchor links), the number of positive samples is 1. For the negative samples, a set of 10 vertices that are not the match are selected uniformly at random from G. γ is a hyper-parameter indicating the margin between positive and negative samples and η represents the weight of O_{EPE} in the total objective.

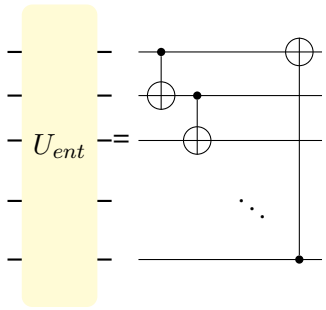
[1] https://github.com/Dywangxin/RED_experiments



(a) The genereal architecture of the quantum embedding ansatz.



(b) The circuit of the trainable parameter block U_l of the embedding ansatz.



(c) The circuit of the entanglement block U_{ent} of the embedding ansatz.

Figure 2: The architecture of the quantum embedding ansatz and its building blocks.

During test time, the model is evaluated on the vertices in $\mathcal{T}_{test}$ with the metric *precision@K*:

$$precision@K = \frac{\sum_{(i,\cdot)\in\mathcal{T}_{test}} \mathbb{I}_{rank(x_i^s, x_i^t)<K}}{|\mathcal{T}_{test}|}. \quad (3)$$

In particular, $precision@1$ represents the percentage of exact matches. The algorithm for SQW, embedding, training and testing is given by Alg. 2.

3 Experiments and Results

In this section, we present the datasets for training the quantum network and the evaluation results, with an analysis on some cases. The VQNE is implemented in TorchQuantum (Wang et al., 2022a), a library built on PyTorch that is used for quantum machine learning and quantum simulation, and our code is available at `https://github.com/waileuklo/VQNE`.

3.1 Datasets and Implementation Details

The datasets used for training are the Weibo-Doban (Wb-Db) and Facebook-Twitter (Fb-Tt) datasets[2] (Xiong and Yan, 2022). Some statistics of the two datasets are given in Table 1. Each dataset is composed of six text files containing the anchor links of the two graphs, ranging from 10% to 50%, and finally 100% of the total number of vertices, and two text files each containing the edges of the two graphs.

Apart from comparing our VQNE model with the one in the original paper (Ye et al., 2023), a multiplex network embedding baseline method BT-Walk is used for comparison as well.

We use the training sets with anchor links from 10%, 20%, 30%, 40% and 50% of the total number of vertices, and select K = 1, 5 and 10 for the evaluation metric given in Eq. 3. As mentioned above, our model is implemented using the framework of TorchQuantum. Both training with shifting technique[3] (Wang et al., 2022a) and training with back-propagation are implemented, and while the two are supposed to be equivalent on a classical computer, the latter is used as it is less time-consuming for training. The number of qubits is set to 13. We use a batch size of 32 and the Adam optimizer with a learning rate of 10^{-7} for training. The number of epochs is set to 1. The number of negative samples is set to 10 and the margin γ is set to 0.07 (Xiong

[2]https://github.com/ShawXh/BTWalk/tree/main/networks

[3]https://github.com/mit-han-lab/torchquantum/tree/main/examples/param_shift_onchip_training

Algorithm 2: Training and testing of VQNE

Input: Source network $G^s = (\mathcal{V}^s, \mathcal{E}^s)$ and target network $G^t = (\mathcal{V}^t, \mathcal{E}^t)$, anchor links $\mathcal{T}_{train} = \{\{p, q\} | p \in \mathcal{V}^s, q \in \mathcal{V}^t\}$, unknown links $\mathcal{T}_{test}$, the number of steps T, the number of layers L;

Output: $precision@K$;

Merge the corresponding vertices according to anchor links between G^s and G^t to form the network G.

foreach $v_i \in G$ **do**
  $|\Psi_i^0\rangle \leftarrow \sum_{k=0}^{N-1} \sqrt{P_{ik}} \, |i, k\rangle$;
  for $t = 1 to T$ **do**
    $|\Psi_i^t\rangle \leftarrow U_W \, |\Psi_i^{t-1}\rangle$;
    $RW.append(|\Psi_i^t\rangle)$;
  end
end

Training procedure:

for $iter = 1$ **to** $iter\_num$ **do**
  foreach $v_i \in G$ **do**
    $Emb(v_i) \leftarrow U(\theta) \, |\Psi_i^t\rangle$;
    Sampling num_pos positive samples to construct RW_{pos};
    Sampling num_neg negative samples to construct RW_{neg};
    $Emb_{pos} \leftarrow U(\theta) RW_{pos}$;
    $Emb_{neg} \leftarrow U(\theta) RW_{neg}$;
    $\mathcal{L} \leftarrow$ $loss(Emb_{v_i}, Emb_{pos}, Emb_{neg})$ as in Eq. 2;
  end
end

Testing procedure:

foreach $v_i in \mathcal{T}_{test} \cap G^s$ **do**
  $Emb(v_i) \leftarrow U(\theta) \, |\Psi_i^t\rangle$;
  foreach $v_j in G^t$ **do**
    $Emb(v_j) \leftarrow U(\theta) \, |\Psi_j^t\rangle$;
    $sim(v_i, v_j) \leftarrow$ $Emb(v_i) \odot Emb(v_j)$;
  end
end

Compute $precision@K$ using Eq. 3;

Result: $precision@K$

	Fb-Tt	Wb-Db
$\lvert\mathcal{V}\rvert$	2,458	3,154
$\lvert\mathcal{E}^s\rvert$	40,298 (Fb)	241,736 (Wb)
$\lvert\mathcal{E}^t\rvert$	95,034 (Tt)	301,074 (Db)
$Avg.Deg.of G^s$	16	77
$Avg.Deg.of G^t$	39	95
Overlap	0.28	0.36

Table 1: Statistics of the datasets for experiments

et al., 2021) in the Marginal Triplet objective, and the weight η of the Elastic Potential Energy objective is set as 0.02. The number of steps T of the quantum walk is set to 2 and the number of layers L of the quantum embedding ansatz is set to 8. The total number of parameters is $3 \cdot n \cdot L = 312$.

3.2 Results and Analysis

The evaluation results of our model and a comparison with the other two models are presented in Tab. 2. As can be seen from the table, our model almost learns nothing from the Fb-Tt dataset, while performing relatively well on the Wb-Db dataset. The reason might be that the Wb-Db dataset is much denser than the Fb-Tt one, and the quantum walk has more chance of capturing the patterns of two matching vertices in a limited number of steps (two steps in our case). Another observation from our training is that the model stops improving any further in terms of the evaluation metric after one epoch of training, and that is why we only train the model for one epoch.

We log the matched vertices as well as the unmatched ones, and we plotted some cases for further analysis. As can be seen from Fig. 3, matched vertices have anchor links between them while unmatched vertices may or may not. That having anchor links in between is a necessary condition for matching two vertices is further confirmed by checking the log file.

4 Conclusion

We implement two algorithms in VQNE in the framework of TorchQuantum and train it on the Fb-Tt and Wb-Db datasets. The results are compared with the BTWalk and the VQNE from the paper (Ye et al., 2023). We observe that while our model learns very little from the Fb-Tt dataset, it performs moderately well on the Wb-Db one, and a possible reason is that the quantum walk can capture more information about a denser network

Dataset		Facebook-Twitter					Weibo-Doban				
Anchor Nodes Percentage		10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
Precision@1	BTWalk	6.35	11.69	15.48	23.97	28.07	**11.01**	**21.08**	**29.12**	**35.90**	**41.72**
	VQNE (paper)	**8.85**	**12.56**	**17.72**	**24.20**	**30.19**	5.38	7.96	14.09	23.72	40.39
	VQNE (ours)	0.00	0.00	0.00	0.14	0.08	0.70	2.61	6.16	12.20	19.72
Precision@5	BTWalk	**20.06**	**31.57**	**39.43**	**49.93**	54.68	**39.06**	**43.30**	**53.46**	**60.64**	**65.25**
	VQNE (paper)	19.34	30.10	37.01	47.19	**55.17**	12.85	15.61	26.22	38.62	57.51
	VQNE (ours)	0.00	0.00	0.06	0.07	0.08	2.36	7.21	14.40	23.77	34.05
Precision@10	BTWalk	**30.70**	**44.28**	**53.39**	**62.00**	**66.97**	**36.98**	**54.20**	**63.34**	**70.23**	**75.27**
	VQNE (paper)	27.75	41.43	49.16	59.32	**64.04**	18.47	21.39	35.37	47.75	65.88
	VQNE (ours)	0.05	0.05	0.12	0.07	0.16	4.76	11.41	21.38	31.54	41.53

Table 2: Network alignment accuracy (%) with best in bold and second best underline

with only a limited number of steps. We also study some cases on the Wb-Db dataset and conclude that having anchor links in between is a necessary condition for matching two vertices from the two graphs.

Acknowledgments

We reference the code from (Wang et al., 2022b) for discrete-time quantum walk. We also thank the authors of (Ye et al., 2023) for the email correspondence and their input in debugging our model.

Declaration

This project is done solely within the course but not other scopes.

References

Lu Bai, Luca Rossi, Lixin Cui, Zhihong Zhang, Peng Ren, Xiao Bai, and Edwin Hancock. 2017. Quantum kernels for unattributed graphs using discrete-time quantum walks. *Pattern Recognition Letters*, 87:96–103. Advances in Graph-based Pattern Recognition.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. 2017. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246.

M. Szegedy. 2004. Quantum speed-up of markov chain based algorithms. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 32–41.

Hanrui Wang, Yongshan Ding, Jiaqi Gu, Yujun Lin, David Z. Pan, Frederic T. Chong, and Song Han. 2022a. Quantumnas: Noise-adaptive search for robust quantum circuits. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 692–708.

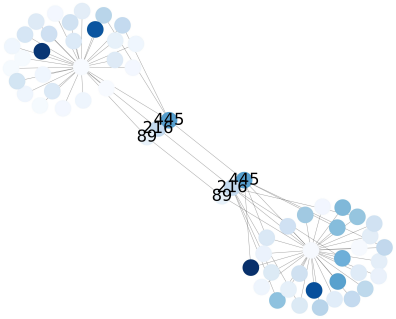
Xin Wang, Songlei Jian, Kai Lu, Yi Zhang, and Kai Liu. 2022b. Red: Learning the role embedding in networks via discrete-time quantum walk. *Applied Intelligence*, 52(2):1493–1507.

Hao Xiong and Junchi Yan. 2022. Btwalk: Branching tree random walk for multi-order structured network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3611–3628.

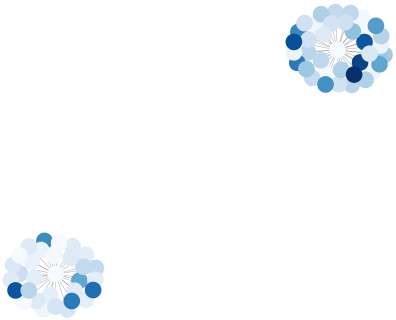
Hao Xiong, Junchi Yan, and Zengfeng Huang. 2023. Learning regularized noise contrastive estimation for robust network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):5017–5034.

Hao Xiong, Junchi Yan, and Li Pan. 2021. Contrastive multi-view multiplex network embedding with applications to robust network alignment. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 1913–1923, New York, NY, USA. Association for Computing Machinery.

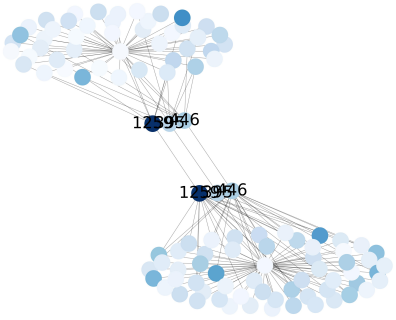
Xinyu Ye, Ge Yan, and Junchi Yan. 2023. Vqne: Variational quantum network embedding with application to network alignment. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 3105–3115, New York, NY, USA. Association for Computing Machinery.



(a) Subgraph containing matched nodes 3129 and 6157 from Wb-Db dataset (20% anchor links).



(b) Subgraph containing un-matched nodes 96 and 3164 from Wb-Db dataset (20% anchor links).



(c) Subgraph containing un-matched nodes 2179 and 3154 from Wb-Db dataset (20% anchor links).

Figure 3: Subgraphs containing matched or un-matched nodes from Wb-Db dataset (20% anchor links). Deeper color represents more degrees of a node. For anchor nodes, their degree is further shown as a number on the node.