MASTER THESIS
Optimizing subgraph isomorphism prediction models with application orientation to the drug design process

Supervisor: *Assoc. Prof.* Quan Thanh Tho
Student: Nguyen Quang Duc - 2270664

Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology
Vietnam National University Ho Chi Minh City

January 15^{th} 2025

The fact that Science walks forward on two feet, namely theory and experiment...

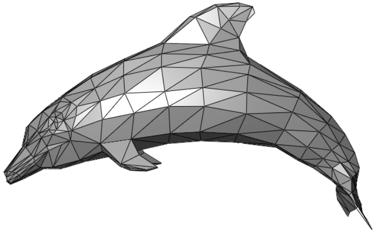
Prof. Robert Millikan - Nobel Laureate 1923

Table of Contents

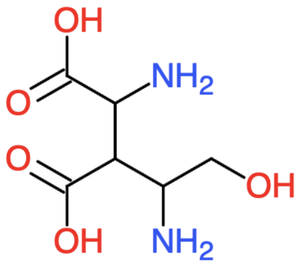
Table of Contents

Graphs exist in many aspects

Social networks [Link]



Mesh objects [Link]

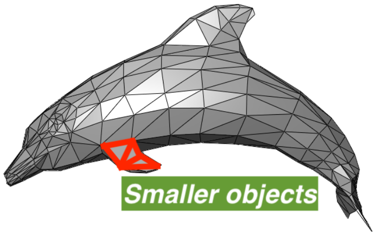


Molecules

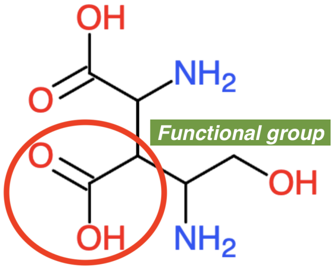
Figure 1: Examples of existence of graphs in real life

Subgraphs play an essential role in both theory and practical

Social networks



Mesh objects



Molecules

Figure 2: Examples of subgraph importance

Especially in drug design process

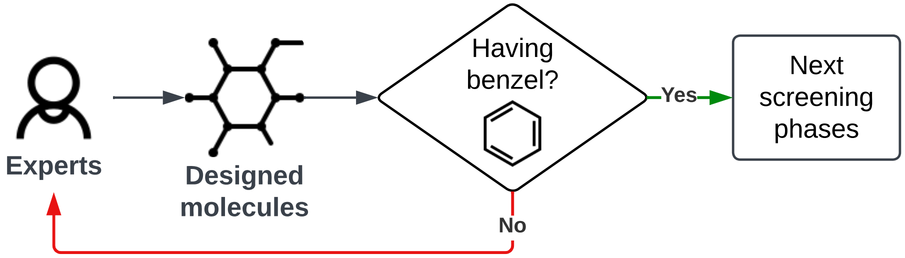
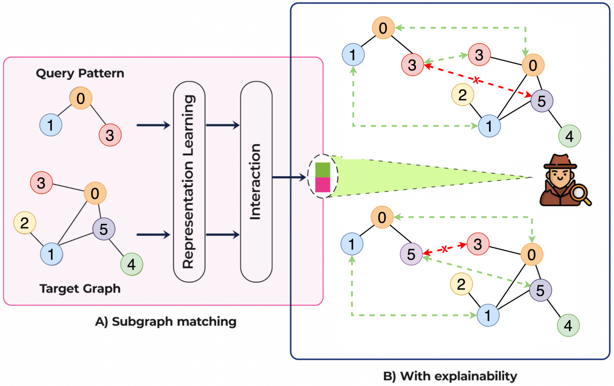


Figure 3: Example of early screening process in drug design

The problem of finding patterns

Question

Given a target graph and a pattern, how can we know whether the pattern exists in the target graph and its mapping in an efficient time manner?



Goal & Scope

Goal: Solving the problem of subgraph matching with explainability

1. Graph Learnable Multi-hop Attention Networks (GLeMa)
2. Theoretical Analysis and Justification
3. Multi-task Learning Framework

Goal & Scope

Goal: Solving the problem of subgraph matching with explainability

1. Graph Learnable Multi-hop Attention Networks (GLeMa)
2. Theoretical Analysis and Justification
3. Multi-task Learning Framework

Scope:

- Proposing a method for solving the problem of subgraph matching with explainability
- Evaluation proposed method mainly on chemistry and bioinformatics domains
- Final result: a theoretically substantiated GNN-based model

Table of Contents

Preliminaries

Labelled Undirected Connected Graph

A labeled undirected connected graph is a graph represented with a 3-tuple $\mathcal{G} = (V, E, l)$ where

1. V is a set of nodes,
2. $E \subseteq [V]^2$ is a set of edges (u, v), where $u, v \in V$
3. $\forall v \in V, \deg(v) \geq 1$
4. $l : V \rightarrow \Sigma$ is a labelling function and Σ is a set of node labels.

Preliminaries

Labelled Subgraph Isomorphism

Given two labeled graphs $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G}, l_\mathcal{G})$ and $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S}, l_\mathcal{S})$, $\mathcal{S}$ is subgraph isomorphic to $\mathcal{G}$ (denoted as $\mathcal{S} \subseteq \mathcal{G}$) if there exists a function $f : V_\mathcal{S} \to V_\mathcal{G}$ such that:

1. $\forall v \in V_\mathcal{S}, l_\mathcal{S}(v) = l_\mathcal{G}(f(v))$ and
2. $\forall u, v \in V_\mathcal{S}, (u, v) \in E_\mathcal{S} \implies (f(u), f(v)) \in E_\mathcal{G}$.

Preliminaries

Labelled Subgraph Isomorphism

Given two labeled graphs $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G}, l_\mathcal{G})$ and $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S}, l_\mathcal{S})$, $\mathcal{S}$ is subgraph isomorphic to $\mathcal{G}$ (denoted as $\mathcal{S} \subseteq \mathcal{G}$) if there exists a function $f : V_\mathcal{S} \to V_\mathcal{G}$ such that:

1. $\forall v \in V_\mathcal{S}, l_\mathcal{S}(v) = l_\mathcal{G}(f(v))$ and
2. $\forall u, v \in V_\mathcal{S}, (u, v) \in E_\mathcal{S} \implies (f(u), f(v)) \in E_\mathcal{G}$.

Induced Labelled Subgraph Isomorphism

Given two labeled graphs $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G}, l_\mathcal{G})$ and $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S}, l_\mathcal{S})$, $\mathcal{S}$ is induced subgraph isomorphic to $\mathcal{G}$ (denoted as $\mathcal{S} \subseteq_{id} \mathcal{G}$) if and only if:

1. $\mathcal{S} \subseteq \mathcal{G}$ and
2. $\forall u, v \in V_\mathcal{S}, (u, v) \notin E_\mathcal{S} \implies (f(u), f(v)) \notin E_\mathcal{G}$.

Preliminaries

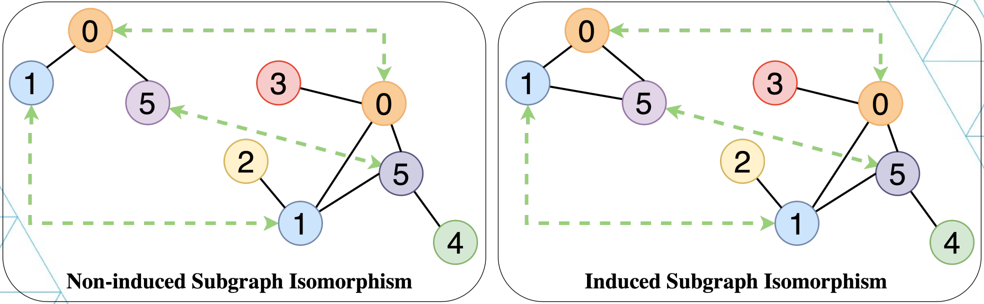

0
1
5
3
0
2
5
1
4
Non-induced Subgraph Isomorphism
0
1
5
3
0
2
5
1
4
Induced Subgraph Isomorphism

Figure 4: Examples of non-induced and induced subgraph isomorphism

Problem Statement

In this thesis, we intend to solve two problems that are derived from the problem of induced subgraph isomorphism.

Problem 1: Subgraph matching

Given a target graph $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T}, l_\mathcal{T})$ and a pattern $\mathcal{P} = (V_\mathcal{P}, E_\mathcal{P}, l_\mathcal{P})$, both are labeled connected graphs, the subgraph matching problem aims to determine whether $\mathcal{P}$ is induced subgraph isomorphic to $\mathcal{T}$ or not.

Problem Statement

In this thesis, we intend to solve two problems that are derived from the problem of induced subgraph isomorphism.

Problem 1: Subgraph matching

Given a target graph $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T}, l_\mathcal{T})$ and a pattern $\mathcal{P} = (V_\mathcal{P}, E_\mathcal{P}, l_\mathcal{P})$, both are labeled connected graphs, the subgraph matching problem aims to determine whether $\mathcal{P}$ is induced subgraph isomorphic to $\mathcal{T}$ or not.

Problem 2: Matching explanation

Let $\mathcal{P} = (V_\mathcal{P}, E_\mathcal{P}, l_\mathcal{P})$ and $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T}, l_\mathcal{T})$ represent two graphs, where $\mathcal{P}$ is a known induced subgraph of $\mathcal{T}$. The matching explanation problem seeks to determine a bijective mapping $\phi : V_\mathcal{P} \rightarrow V_\mathcal{T}$ that accurately identifies the correspondence between the nodes of $\mathcal{P}$ and their counterparts in $\mathcal{T}$.

Table of Contents

Subgraph Isomorphism Algorithms

- **Non-induced settings:** This setting holds significant utility in various aspects of data management, including tasks like graph indexing, graph similarity search, and graph retrieval[1].

[1] Roy et al., "Interpretable Neural Subgraph Matching for Graph Retrieval".
[2] Garey and Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*.
[3] Shang et al., "Taming verification hardness: an efficient algorithm for testing subgraph isomorphism".
[4] He and Singh, "Graphs-at-a-time: query language and access methods for graph databases".
[5] W.-S. Han, J. Lee, and J.-H. Lee, "TurboISO: Towards ultrafast and robust subgraph isomorphism search in large graph databases".
[6] Bi et al., "Efficient Subgraph Matching by Postponing Cartesian Products".
[7] Bhattarai, H. Liu, and Huang, "CECI: Compact Embedding Cluster Index for Scalable Subgraph Matching".
[8] M. Han et al., "Efficient Subgraph Matching: Harmonizing Dynamic Programming, Adaptive Matching Order, and Failing Set Together".

Subgraph Isomorphism Algorithms

- **Non-induced settings:** This setting holds significant utility in various aspects of data management, including tasks like graph indexing, graph similarity search, and graph retrieval[1].
- **Induced settings:** This setting has been proven to be NP-complete[2]. Various algorithms[3][4][5][6][7][8] have been proposed focusing on generating effective matching orders and designing robust filtering strategies to reduce the number of candidates in the data graph.

[1] Roy et al., "Interpretable Neural Subgraph Matching for Graph Retrieval".

[2] Garey and Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*.

[3] Shang et al., "Taming verification hardness: an efficient algorithm for testing subgraph isomorphism".

[4] He and Singh, "Graphs-at-a-time: query language and access methods for graph databases".

[5] W.-S. Han, J. Lee, and J.-H. Lee, "TurboISO: Towards ultrafast and robust subgraph isomorphism search in large graph databases".

[6] Bi et al., "Efficient Subgraph Matching by Postponing Cartesian Products".

[7] Bhattarai, H. Liu, and Huang, "CECI: Compact Embedding Cluster Index for Scalable Subgraph Matching".

[8] M. Han et al., "Efficient Subgraph Matching: Harmonizing Dynamic Programming, Adaptive Matching Order, and Failing Set Together".

Neural subgraph isomorphism and explanation

Modern approaches for subgraph isomorphism and explanation utilize Graph Neural Networks:

- The initial work[9] attempted to assess the feasibility of Graph Neural Networks in small subgraph matching.
- With recent advancements in GNNs[10][11][12], contemporary subgraph matching techniques[13][14][15] have achieved state-of-the-art results in terms of efficiency.
- Recent studies have utilized the interpretability of GNNs[16][17][18] through a model-intrinsic perspective.

[9] Scarselli et al., "The Graph Neural Network Model".
[10] Kipf and Welling, "Semi-Supervised Classification with Graph Convolutional Networks".
[11] Hamilton, Ying, and Leskovec, "Inductive representation learning on large graphs".
[12] K. Xu et al., "How Powerful are Graph Neural Networks?"
[13] Bai et al., "Convolutional set matching for graph similarity".
[14] Zhang and W. S. Lee, "Deep Graphical Feature Learning for the Feature Matching Problem".
[15] Ying et al., "Neural Subgraph Matching".
[16] Yuan et al., "XGNN: Towards Model-Level Explanations of Graph Neural Networks".
[17] Vu and Thai, "PGM-Explainer: Probabilistic Graphical Model Explanations for Graph Neural Networks".
[18] Wu et al., "Rethinking Explaining Graph Neural Networks via Non-parametric Subgraph Matching".

Neural subgraph isomorphism and explanation

NeuralMatch[19], a cutting-edge subgraph matching algorithm employing a specialized graph neural network architecture.

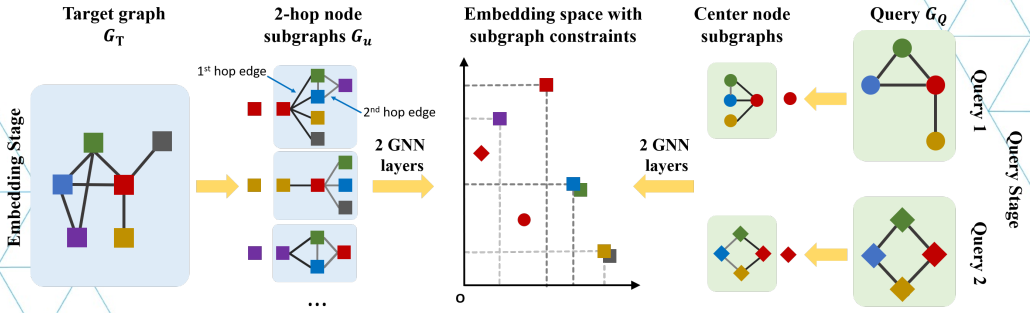


Figure 5: NeuralMatch architecture[19]

[19] Ying et al., "Neural Subgraph Matching".

Neural subgraph isomorphism and explanation

DualMP[20], one of state-of-the-art approaches for performing subgraph counting and matching. This method leverages Dual Message Passing Neural Networks (DMPNNs) to learn both node and edge representations simultaneously in an aligned space through an efficient asynchronous update mechanism.

[20] X. Liu and Song, "Graph Convolutional Networks with Dual Message Passing for Subgraph Isomorphism Counting and Matching".

Conventional Graph Neural Networks

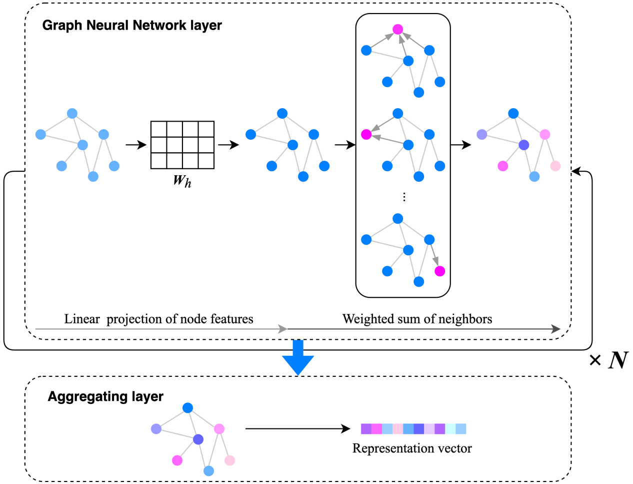
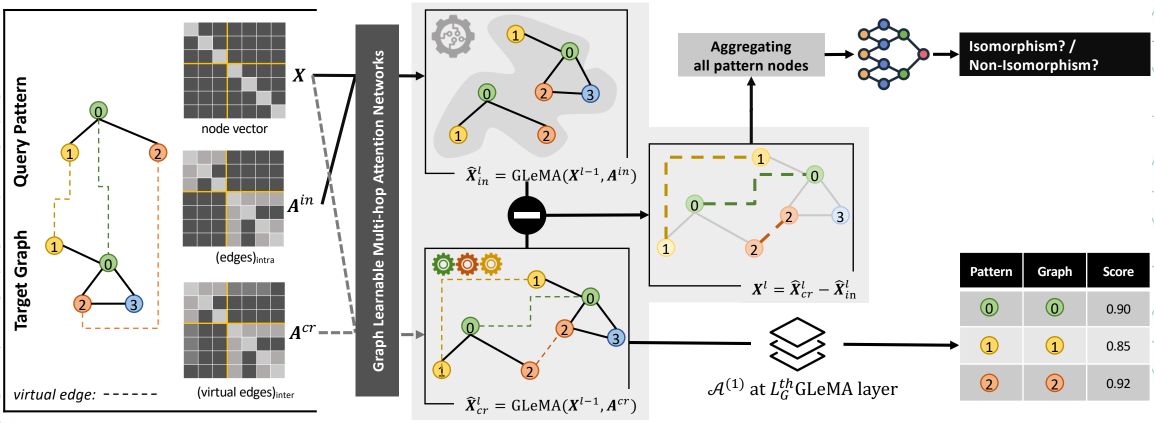

Graph Neural Network layer
W_h
Linear projection of node features
Weighted sum of neighbors
$\times N$
Aggregating layer
Representation vector

Figure 6: Common Graph Neural Networks architecture

Table of Contents

Approach Overview



Query Pattern
Target Graph
virtual edge:
node vector
(edges)intra
(virtual edges)inter
X
A^{in}
A^{cr}
Graph Learnable Multi-hop Attention Networks
$\hat{X}^l_{in} = \text{GLeMA}(X^{l-1}, A^{in})$
$\hat{X}^l_{cr} = \text{GLeMA}(X^{l-1}, A^{cr})$
$X^l = \hat{X}^l_{cr} - \hat{X}^l_{in}$
Aggregating all pattern nodes
Isomorphism? / Non-Isomorphism?
$\mathcal{A}^{(1)}$ at L^{th}_G GLeMA layer

Pattern	Graph	Score
0	0	0.90
1	1	0.85
2	2	0.92

Figure 7: xNeuSM architecture is inspired by the study of Lim *et al.*[21]

[21] Lim et al., "Predicting Drug–Target Interaction Using a Novel Graph Neural Network with 3D Structure-Embedded Graph Representation".

Input Representation

Suppose that $\mathcal{P} = \{V_{\mathcal{P}}, E_{\mathcal{P}}, l_{\mathcal{P}}\}$ and the target as $\mathcal{T} = \{V_{\mathcal{T}}, E_{\mathcal{T}}, l_{\mathcal{T}}\}$ where V, E are the sets of nodes and edges respectively; $l : V \to T_V$ is the labelling function.

Input Representation

Suppose that $\mathcal{P} = \{V_\mathcal{P}, E_\mathcal{P}, l_\mathcal{P}\}$ and the target as $\mathcal{T} = \{V_\mathcal{T}, E_\mathcal{T}, l_\mathcal{T}\}$ where V, E are the sets of nodes and edges respectively; $l : V \to T_V$ is the labelling function.

$$\boldsymbol{X} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{|V_\mathcal{P}|}, \vec{x}_{|V_\mathcal{P}|+1}, \ldots, \vec{x}_{|V_\mathcal{P}|+|V_\mathcal{T}|}\} \text{ with } \vec{x}_i \in \mathbb{R}^{2|T_V|} \tag{1}$$

$$\boldsymbol{A}^{in}_{ij} = \begin{cases} 1 & \text{if there is an undirected edge or} \\ & \text{a directed edge that connects } j \text{ to } i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$\boldsymbol{A}^{cr}_{ij} = \begin{cases} \boldsymbol{A}^{in}_{ij} & \text{if } i, j \in \mathcal{P} \text{ or } i, j \in \mathcal{T} \\ 1 & \text{if } l(i) = l(j) \text{ and } i \in \mathcal{P} \text{ and } j \in \mathcal{T}, \\ & \text{or if } l(i) = l(j) \text{ and } i \in \mathcal{T} \text{ and } j \in \mathcal{P} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Graph Multi-hop Attention layer

Assuming that we are applying this layer on an abstract graph $\mathcal{G} = \{V, E, l\}$, we present this graph with $(\boldsymbol{X}, \boldsymbol{A})$ where $\boldsymbol{X} \in \mathbb{R}^{|V| \times F}$ is the set of node features and E is the set of edges.

$$\boldsymbol{X}_h = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_{|V|}\}, \ \vec{x_i} \in \mathbb{R}^F \tag{4}$$

$$\boldsymbol{A}_{ij} = \begin{cases} 1 & \text{if there is an edge that connects } j \text{ to } i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Graph Multi-hop Attention layer

Assuming that we are applying this layer on an abstract graph $\mathcal{G} = \{V, E, l\}$, we present this graph with $(\boldsymbol{X}, \boldsymbol{A})$ where $\boldsymbol{X} \in \mathbb{R}^{|V| \times F}$ is the set of node features and E is the set of edges.

$$\boldsymbol{X}_h = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{|V|}\}, \ \vec{x}_i \in \mathbb{R}^F \tag{4}$$

$$\boldsymbol{A}_{ij} = \begin{cases} 1 & \text{if there is an edge that connects } j \text{ to } i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Then, we project the input features into embedding space and calculate 1-hop attention among all nodes using Luong's Attention.

$$\boldsymbol{X'} = \{\vec{x}'_i = \boldsymbol{W}_h \vec{x}_i\}_{i=1}^{|V|}, \vec{x}'_i \in \mathbb{R}^{F'} \tag{6}$$

$$e_{ij} = \begin{cases} \delta(\vec{x}'^T_i \boldsymbol{W}_e \vec{x}'_j) & \text{a directed edge } j \text{ to } i, \\ \delta(\vec{x}'^T_i \boldsymbol{W}_e \vec{x}'_j + \vec{x}'^T_j \boldsymbol{W}_e \vec{x}'_i) & \text{an undirected edge } j \text{ to } i, \end{cases} \tag{7}$$

$$\mathcal{A}^{(1)} = \{a_{ij}^{(1)} = \frac{\exp(e_{ij})}{\sum_{n \in \mathcal{N}_i} \exp(e_{in})} \boldsymbol{A}_{ij} | i, j = \overline{1, |V|}\} \tag{8}$$

Graph Multi-hop Attention layer

The attention diffusion matrix can be calculated by the weighted sum of all k-hop attention matrix with attention decay factors θ_k.

$$\begin{cases} (\mathcal{A}^{(1)})^0 & = \boldsymbol{I} \\ \mathcal{A} & = \sum_{k=0}^{\infty} \theta_k (\mathcal{A}^{(1)})^k \text{ where } \sum_{k=0}^{\infty} \theta_k = 1 \text{ and } \theta_k > 0 \end{cases} \tag{9}$$

Graph Multi-hop Attention layer

The attention diffusion matrix can be calculated by the weighted sum of all k-hop attention matrix with attention decay factors θ_k.

$$\begin{cases} (\mathcal{A}^{(1)})^0 & = \boldsymbol{I} \\ \mathcal{A} & = \sum_{k=0}^{\infty} \theta_k (\mathcal{A}^{(1)})^k \text{ where } \sum_{k=0}^{\infty} \theta_k = 1 \text{ and } \theta_k > 0 \end{cases} \tag{9}$$

The output of GMA layer is a projection of the combination H-head attention node features.

$$\widehat{\boldsymbol{X}} = \left(\Big\Vert_{h=1}^{H} \delta \left(\mathcal{A}_h \boldsymbol{X'}_h \right) \right) \boldsymbol{W}_o \text{ with } \boldsymbol{W}_o \in \mathbb{R}^{HF' \times F'}. \tag{10}$$

Learnable multi-hop attention mechanism

There are two main problems[22] associated with computing exact attention diffusion matrix $\mathcal{A}$:

1. The elevated computational intricacies involved in computing $\mathcal{A}$ due to matrix multiplication.
2. The judicious selection of the suitable values for θ_k, which significantly influences the augmentation or attenuation of the model performance.

[22] Gasteiger, Bojchevski, and Günnemann, "Predict then Propagate: Graph Neural Networks meet Personalized PageRank".

Learnable multi-hop attention mechanism

There are two main problems[22] associated with computing exact attention diffusion matrix $\mathcal{A}$:

1. The elevated computational intricacies involved in computing $\mathcal{A}$ due to matrix multiplication.
2. The judicious selection of the suitable values for θ_k, which significantly influences the augmentation or attenuation of the model performance.

$\Longrightarrow$ **Approximate, learnable multi-hop attention mechanism**

[22] Gasteiger, Bojchevski, and Günnemann, "Predict then Propagate: Graph Neural Networks meet Personalized PageRank".

Learnable Multi-hop Attention Mechanism

Reducing multi-hop attention matrix computation complexity Following the methodology outlined in the previous work[23], we adopt the geometric distribution to determine θ_k, wherein $\theta_k = \alpha(1-\alpha)^k, \alpha \in (0,1)$ represents the teleport probability. The approximation for $\mathcal{A}\boldsymbol{X'}$ is achieved as follows.

$$\begin{cases} \boldsymbol{Z}^{(0)} & = \boldsymbol{X'} \\ \boldsymbol{Z}^{(k)} & = (1-\alpha)\mathcal{A}^{(1)}\boldsymbol{Z}^{(k-1)} + \alpha\boldsymbol{Z}^{(0)},\ k = \overline{1,K} \end{cases} \tag{11}$$

[23] Wang et al., "Multi-hop Attention Graph Neural Networks".

Learnable Multi-hop Attention Mechanism

Reducing multi-hop attention matrix computation complexity Following the methodology outlined in the previous work[23], we adopt the geometric distribution to determine θ_k, wherein $\theta_k = \alpha(1-\alpha)^k, \alpha \in (0,1)$ represents the teleport probability. The approximation for $\mathcal{A}\boldsymbol{X'}$ is achieved as follows.

$$\begin{cases} \boldsymbol{Z}^{(0)} & = \boldsymbol{X'} \\ \boldsymbol{Z}^{(k)} & = (1-\alpha)\mathcal{A}^{(1)}\boldsymbol{Z}^{(k-1)} + \alpha\boldsymbol{Z}^{(0)}, \ k = \overline{1,K} \end{cases} \tag{11}$$

Proposition 1

$$lim_{K\to\infty}\boldsymbol{Z}^{(K)} = \mathcal{A}\boldsymbol{X'}$$

This proposition was proven in (Wang *et al.*, 2020).

[23] Wang et al., "Multi-hop Attention Graph Neural Networks".

Learnable Multi-hop Attention Mechanism

Reducing multi-hop attention matrix computation complexity

Proposition (Proposition 1)

$lim_{K\to\infty}\boldsymbol{Z}^{(K)} = \mathcal{A}\boldsymbol{X'}$

However, there is a tradeoff between the approximation error and computing resource.

- **Question:** How much large is K?
- **Question:** How much is the approximate error of $\boldsymbol{Z}^{(K)}$?

Learnable Multi-hop Attention Mechanism

Reducing multi-hop attention matrix computation complexity

Proposition (Proposition 1)

$lim_{K\to\infty}\boldsymbol{Z}^{(K)} = \mathcal{A}\boldsymbol{X'}$

However, there is a tradeoff between the approximation error and computing resource.

- **Question:** How much large is K?
- **Question:** How much is the approximate error of $\boldsymbol{Z}^{(K)}$?

In this thesis, we propose the theoretical evidence for $\mathcal{A}$ approximation error.

Proposition 2

The average approximate error of each element in $\mathcal{A}^{(K)} = \boldsymbol{Z}^{(K)}\boldsymbol{X'}^{-1}$ is bound by $(1-\alpha)^{K+1}$

Learnable Multi-hop Attention Mechanism

Learning distinct teleport probabilities for nodes

- Personalized PageRank (PPR)[24] which reveals the importance of each node
- The original work of GAT has proved that the attention matrix in GAT can be viewed as the transition matrix in PPR[25] but it used the same teleport probability for all nodes, which limits the power of PPR theoretically.

[24] Lofgren, "Efficient Algorithms for Personalized PageRank".

[25] Wang et al., "Multi-hop Attention Graph Neural Networks".

Learnable Multi-hop Attention Mechanism

Learning distinct teleport probabilities for nodes

- Personalized PageRank (PPR)[24] which reveals the importance of each node
- The original work of GAT has proved that the attention matrix in GAT can be viewed as the transition matrix in PPR[25] but it used the same teleport probability for all nodes, which limits the power of PPR theoretically.

$\Longrightarrow$ Leveraging the capabilities of PPR, we propose a novel approach that involves the customization of teleport probabilities for individual nodes, denoted as $\beta = \{\beta_v\}_{v=1}^{|V|}$.

[24] Lofgren, "Efficient Algorithms for Personalized PageRank".
[25] Wang et al., "Multi-hop Attention Graph Neural Networks".

Learnable Multi-hop Attention Mechanism

Learning distinct teleport probabilities for nodes Inspired by Gated Recurrent Units[26], we devise a method wherein the network autonomously learns the teleport probabilities via a straightforward linear transformation.

$$\beta = \sigma((\boldsymbol{X'} || \mathcal{A}^{(1)} \boldsymbol{X'}) \boldsymbol{W}_{\beta} + b), \tag{12}$$

[26] Cho et al., "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation".

Learnable Multi-hop Attention Mechanism

Learning distinct teleport probabilities for nodes Inspired by Gated Recurrent Units[26], we devise a method wherein the network autonomously learns the teleport probabilities via a straightforward linear transformation.

$$\beta = \sigma((\boldsymbol{X'} \| \mathcal{A}^{(1)} \boldsymbol{X'}) \boldsymbol{W}_\beta + b), \tag{12}$$

Employing distinct teleport probabilities for each node results in modifications to Equation 9. These alterations encompass $\theta_{kj} = \beta_j (1 - \beta_j)^k$, $\sum_{k=0}^{\infty} \theta_{kj} = 1$ for $j \in \overline{1, N}$, and $\theta_{kj} > 0$.

[26] Cho et al., "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation".

Learnable Multi-hop Attention Mechanism

Learning distinct teleport probabilities for nodes Inspired by Gated Recurrent Units[26], we devise a method wherein the network autonomously learns the teleport probabilities via a straightforward linear transformation.

$$\beta = \sigma((\boldsymbol{X'} || \mathcal{A}^{(1)} \boldsymbol{X'}) \boldsymbol{W}_{\beta} + b), \tag{12}$$

Employing distinct teleport probabilities for each node results in modifications to Equation 9. These alterations encompass $\theta_{kj} = \beta_j (1 - \beta_j)^k$, $\sum_{k=0}^{\infty} \theta_{kj} = 1$ for $j \in \overline{1, N}$, and $\theta_{kj} > 0$.

Proposition 3

$lim_{K \to \infty} \boldsymbol{Z}_{\beta}^{(K)} = \mathcal{A}_{\eta} \boldsymbol{X'}$

[26] Cho et al., "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation".

Learnable multi-hop attention mechanism

Learning distinct teleport probabilities for nodes

Algorithm 1: Learnable Multi-hop Attention

```
Input  : 1-hop attention matrix $\mathcal{A}^{(1)}$
         Node feature matrix $\boldsymbol{X'}$
         Number of approximate hops $K$
Output : Diffused node feature matrix $\widehat{\boldsymbol{X}}$
1 $\boldsymbol{Z}^{(0)} \leftarrow \boldsymbol{X'}$
2 $\beta \leftarrow \sigma((\boldsymbol{X'} || \mathcal{A}^{(1)}\boldsymbol{X'})\boldsymbol{W}_{\beta} + b)$
3 for k in Range($1 \dots K$) do
4     $\boldsymbol{Z}^{(k)} = (1-\beta)\mathcal{A}^{(1)}\boldsymbol{Z}^{(k-1)} + \beta\boldsymbol{Z}^{(0)}$
5 end
6 $\widehat{\boldsymbol{X}} \leftarrow \boldsymbol{Z}^{(K)}$
```

Graph Learnable Multi-hop Attention Networks

With the GLeMA layer, using the input of triple $(\boldsymbol{X}, \boldsymbol{A}^{in}, \boldsymbol{A}^{cr})$, we can compute higher representation for the query and target graphs as follows.

$$\begin{cases} \boldsymbol{X}^0 = \boldsymbol{X} \\ \widehat{\boldsymbol{X}}^l_{in} = \mathsf{GLeMA}_l(\boldsymbol{X}^{l-1}, \boldsymbol{A}^{in}), l = \overline{1, L_G} \\ \widehat{\boldsymbol{X}}^l_{cr} = \mathsf{GLeMA}_l(\boldsymbol{X}^{l-1}, \boldsymbol{A}^{cr}), l = \overline{1, L_G} \\ \boldsymbol{X}^l = \widehat{\boldsymbol{X}}^l_{cr} - \widehat{\boldsymbol{X}}^l_{in}, l = \overline{1, L_G} \end{cases} \tag{13}$$

Graph Learnable Multi-hop Attention Networks

With the GLeMA layer, using the input of triple $(\boldsymbol{X}, \boldsymbol{A}^{in}, \boldsymbol{A}^{cr})$, we can compute higher representation for the query and target graphs as follows.

$$\begin{cases} \boldsymbol{X}^0 = \boldsymbol{X} \\ \widehat{\boldsymbol{X}}^l_{in} = \text{GLeMA}_l(\boldsymbol{X}^{l-1}, \boldsymbol{A}^{in}), l = \overline{1, L_G} \\ \widehat{\boldsymbol{X}}^l_{cr} = \text{GLeMA}_l(\boldsymbol{X}^{l-1}, \boldsymbol{A}^{cr}), l = \overline{1, L_G} \\ \boldsymbol{X}^l = \widehat{\boldsymbol{X}}^l_{cr} - \widehat{\boldsymbol{X}}^l_{in}, l = \overline{1, L_G} \end{cases} \tag{13}$$

The node features at the l^{th} layer are computed by taking the difference between the inter-graph features and the intra-graph features from the previous $(l-1)^{th}$ layer. This learning of disparities between inter-graph and intra-graph features enhances the signal for verifying subgraph isomorphism.

Subgraph Matching Task

The methodology for computing the representation vector is elucidated in Equation 14. In this equation, all pattern node feature vectors are aggregated to form the representation for our input.

$$x^0_{repr} = \frac{1}{|V_{\mathcal{P}}|} \sum_{i \in V_{\mathcal{P}}} x_i^{L_G} \tag{14}$$

Subgraph Matching Task

The methodology for computing the representation vector is elucidated in Equation 14. In this equation, all pattern node feature vectors are aggregated to form the representation for our input.

$$x^0_{repr} = \frac{1}{|V_{\mathcal{P}}|} \sum_{i \in V_{\mathcal{P}}} x_i^{L_G} \tag{14}$$

Equation 15 provides the mathematical formulations underpinning the classifier of subgraph matching.

$$\begin{cases} x^i_{repr} & = \delta(\boldsymbol{W}_i x^{i-1}_{repr} + b_i), i = \overline{1, L_{FC} - 1} \\ \widehat{y} & = \sigma(\boldsymbol{W}_y x^{L_{FC}-1}_{repr} + b_y) \end{cases} \tag{15}$$

Matching Explanation Task

The mapping of nodes between the query and target graph is established if the mapping probability is larger than a predefined threshold ϵ.

$$\begin{aligned} \mathcal{M} =& \{(i, j, p_{ij}) | p_{ij} \geq \epsilon\}, \text{where } i \in V_{\mathcal{P}}, j \in V_{\mathcal{T}} \text{ and} \\ p_{ij} =& \frac{1}{2}\left((a_{ij}^{(1)})^{L_G} + (a_{ji}^{(1)})^{L_G}\right). \end{aligned} \tag{16}$$

In (16), $\mathcal{M}$ is the set of mapping nodes between the pattern and target graph; p_{ij} which is computed by average of 1-hop attention coefficients $((a_{ij}^{(1)})^{L_G}, (a_{ji}^{(1)})^{L_G})$ is the mapping probability between pattern i^{th} node and target j^{th} node.

Optimization Objective

- $\mathcal{L}_{sm}$ is a binary cross-entropy loss designed to assess the model capacity to predict subgraph isomorphism accurately.
- $\mathcal{L}_{me}$ is an attention-based loss, aimed at reinforcing the attention coefficients between nodes i and j $(i \in V_{\mathcal{P}}, j \in V_{\mathcal{T}})$ that correspond to actual mappings, while simultaneously diminishing the coefficients for node pairs sharing the same label (represented as $m \in V_{\mathcal{P}}, n \in V_{\mathcal{T}}, l(m) = l(n))$ but lacking a mapping relationship.
- $\mathcal{L}$ is the final objective function where λ is an hyperparameter.

Optimization Objective

- $\mathcal{L}_{sm}$ is a binary cross-entropy loss designed to assess the model capacity to predict subgraph isomorphism accurately.
- $\mathcal{L}_{me}$ is an attention-based loss, aimed at reinforcing the attention coefficients between nodes i and j ($i \in V_{\mathcal{P}}, j \in V_{\mathcal{T}}$) that correspond to actual mappings, while simultaneously diminishing the coefficients for node pairs sharing the same label (represented as $m \in V_{\mathcal{P}}, n \in V_{\mathcal{T}}, l(m) = l(n)$) but lacking a mapping relationship.
- $\mathcal{L}$ is the final objective function where λ is an hyperparameter.

$$\begin{cases} \mathcal{L}_{sm} & = -\frac{1}{|\mathcal{D}|}\sum_{k=1}^{|\mathcal{D}|} y_k \cdot \log(\widehat{y}_k) + (1 - y_k) \cdot \log(1 - \widehat{y}_k) \\ \mathcal{L}_{me} & = \frac{1}{|\mathcal{D}|}\sum_{k=1}^{|\mathcal{D}|} \frac{\sum \exp\left(-\left(a_{ij}^{(1)(L_G)}\right)_k\right)}{\sum \exp\left(-\left(a_{mn}^{(1)(L_G)}\right)_k\right) - \sum \exp\left(-\left(a_{ij}^{(1)(L_G)}\right)_k\right) + 1} \\ \mathcal{L} & = \mathcal{L}_{sm} + \lambda \mathcal{L}_{me} \end{cases} \quad (17)$$

Table of Contents

Datasets

We assess the performance of our framework across a diverse range of real datasets encompassing various domains, including bioinformatics, chemistry, computer vision, and social networks.

Table 1: Statistics of real datasets

Domain	$\mathcal{D}$	$\overline{\lvert V_{\mathcal{T}} \rvert}$	$\overline{\lvert E_{\mathcal{T}} \rvert}$	$\overline{deg}$	$\lvert\mathcal{D}\rvert$	$\lvert\Sigma\rvert$
Bioinformatics	KKI	26.96	48.42	3.19	83	190
Chemistry	COX2	41.22	43.45	2.10	467	8
Chemistry	COX2_MD	26.28	335.12	25.27	303	7
Chemistry	DHFR	42.43	44.54	2.10	756	9
Social networks	DBLP-v1	10.48	19.65	3.43	19456	39
Computer vision	MSRC-21	77.52	198.32	5.10	563	22

Baselines

Exact approach We utilize seven distinct approaches to compare with `xNeuSM`, including: *VF3*[27], *TurboISO*[28], *CFL*[29], *CECI*[30], *QuickSI*[31], *DAF*[32], *GraphQL*[33].

Approximate approach We compare our approach with *NeuralMatch*[34] and DualMP[35].

[27] Carletti et al., "Introducing VF3: A New Algorithm for Subgraph Isomorphism".

[28] W.-S. Han, J. Lee, and J.-H. Lee, "TurboISO: Towards ultrafast and robust subgraph isomorphism search in large graph databases".

[29] Bi et al., "Efficient Subgraph Matching by Postponing Cartesian Products".

[30] Bhattarai, H. Liu, and Huang, "CECI: Compact Embedding Cluster Index for Scalable Subgraph Matching".

[31] Shang et al., "Taming verification hardness: an efficient algorithm for testing subgraph isomorphism".

[32] M. Han et al., "Efficient Subgraph Matching: Harmonizing Dynamic Programming, Adaptive Matching Order, and Failing Set Together".

[33] He and Singh, "Graphs-at-a-time: query language and access methods for graph databases".

[34] Ying et al., "Neural Subgraph Matching".

[35] X. Liu and Song, "Graph Convolutional Networks with Dual Message Passing for Subgraph Isomorphism Counting and Matching".

Metrics

Subgraph matching task

- *Execution time*
- *Accuracy, ROC AUC, PR AUC, F1 score*

Metrics

Subgraph matching task

- *Execution time*
- *Accuracy, ROC AUC, PR AUC, F1 score*

Matching explanation task

- *Average Top-K Accuracy*

$$TopK = \frac{1}{|\mathcal{D}_{test}|} \sum_{(\mathcal{T},\mathcal{P})\in\mathcal{D}_{test}} \left(\frac{1}{|V_{\mathcal{P}}|} \sum_{i\in V_{\mathcal{P}}} Acc_i^K \right)$$

- *Mean Reciprocal Rank*

$$MRR = \frac{1}{|\mathcal{D}_{test}|} \sum_{(\mathcal{T},\mathcal{P})\in\mathcal{D}_{test}} \left(\frac{1}{|V_{\mathcal{P}}|} \sum_{i\in V_{\mathcal{P}}} \frac{1}{rank_i} \right)$$

End2end subgraph isomorphism - Runtime

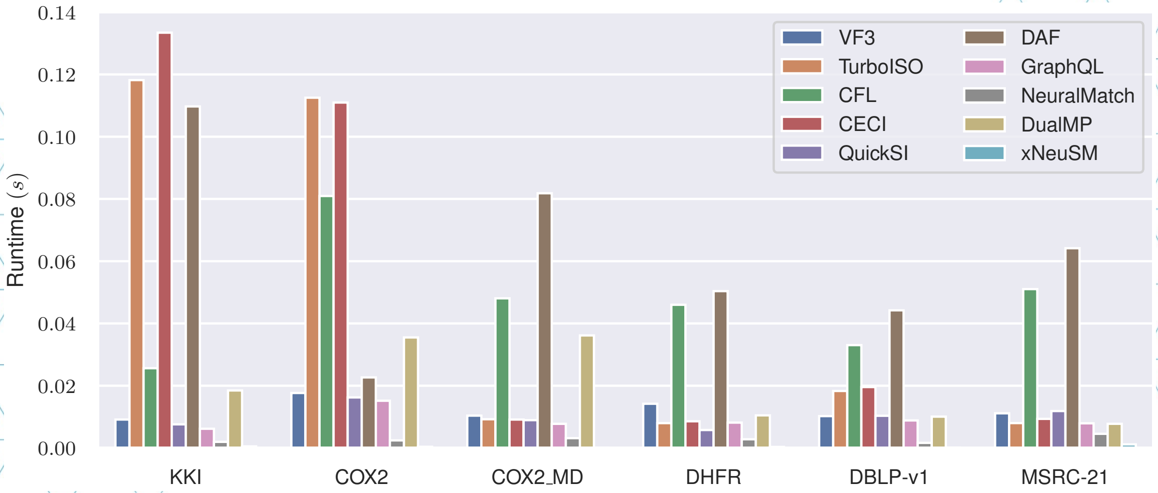


Figure 8: Execution time on subgraph matching task.

End2end subgraph isomorphism - Accuracy

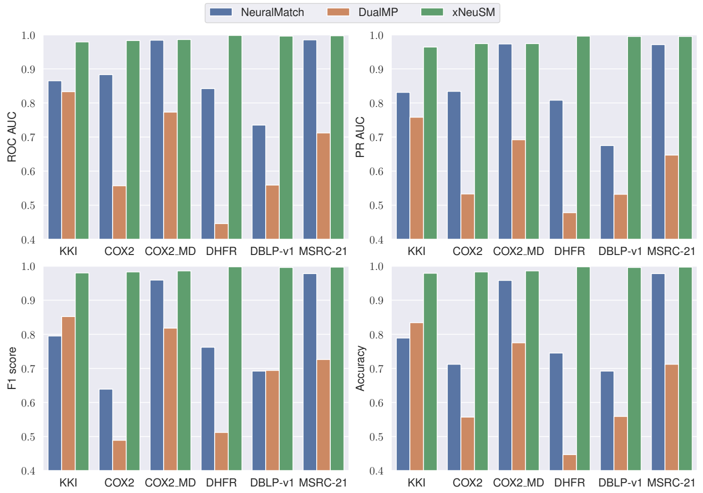


Figure 9: Performance comparison between Neural Match and xNeuSM

Confidence assessment

By elevating the output probability threshold for the subgraph matching task, we demonstrate that our model maintains high performance levels.

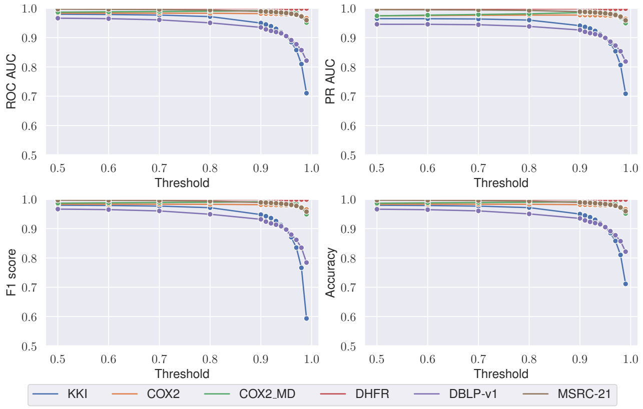


Figure 10: Relation between confidence threshold and model performance

Scalability

We evaluated the performance of all techniques using the above datasets with different levels of graph density, ranging from sparse to dense graphs.

- Vary $D(\mathcal{P})$: We divided queries into two subsets based on their average degree. The first subset, labeled “dense”, included queries with a degree of three or higher. The second subset, labeled “sparse”, encompassed queries with a degree less than 3.
- Vary $|V_{\mathcal{P}}|$: We partitioned the query set into four groups based on query size thresholds: $|V_{\mathcal{P}}| \leq 20$, $20 < |V_{\mathcal{P}}| \leq 40$, $40 < |V_{\mathcal{P}}| \leq 60$, and $60 < |V_{\mathcal{P}}|$.

Scalability

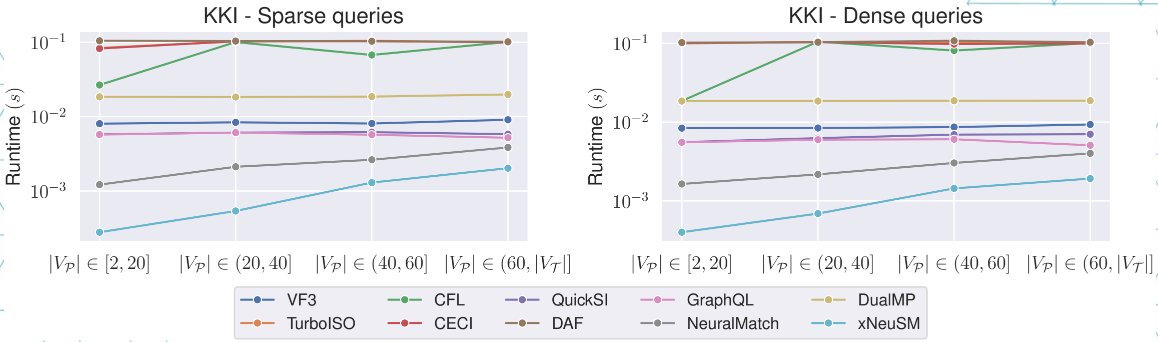


Figure 11: Execution time on KKI dataset

Scalability

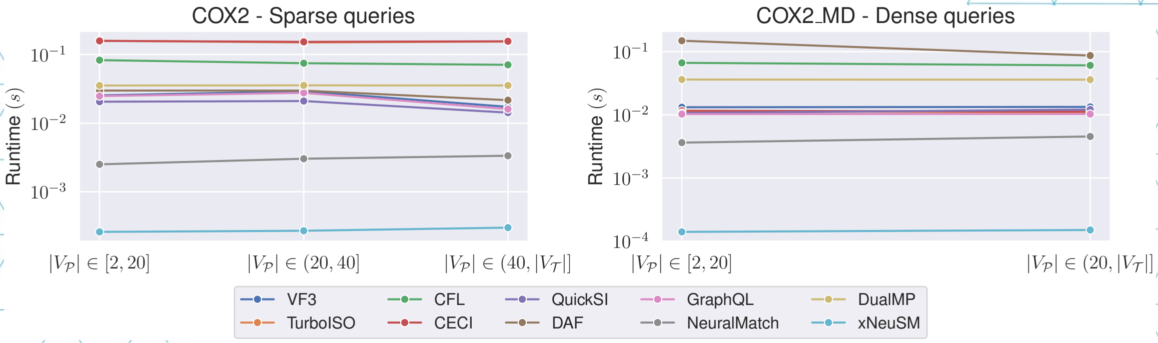


Figure 12: Execution time on COX2 and COX2_MD dataset

Scalability

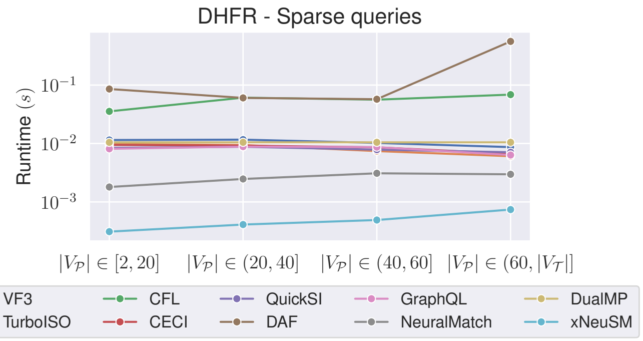


Figure 13: Execution time on DHFR dataset

Scalability

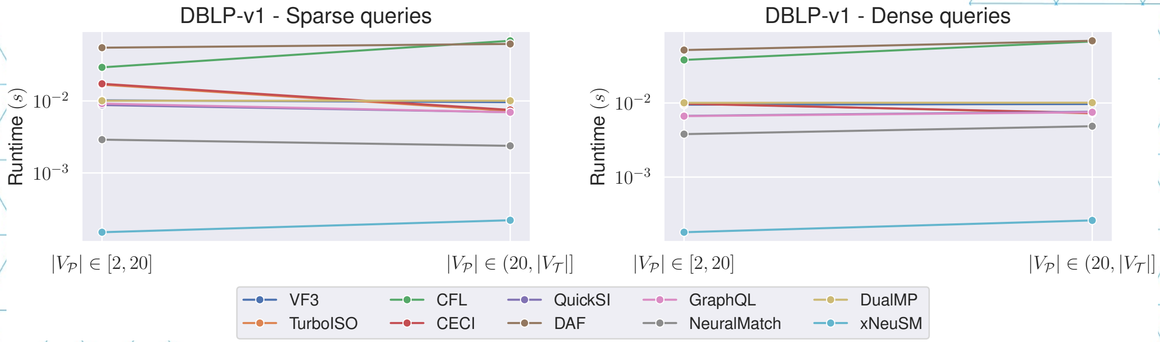


Figure 14: Execution time on DBLP-v1 dataset

Scalability

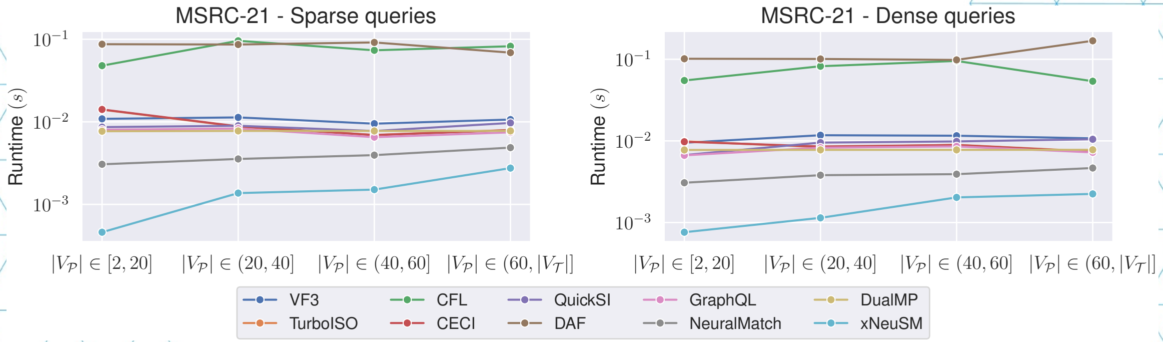


Figure 15: Execution time on MSRC-21 dataset

Matching explanation

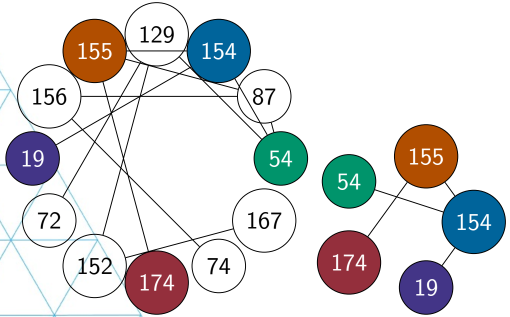
Quantitative analysis: It is important to note that this task is exclusively applicable to known isomorphism pairs of (pattern, target). Non-isomorphic cases were deliberately excluded from our testing, as they may not be representative of real-world use cases.

Table 2: Performance of in subgraph aligning task

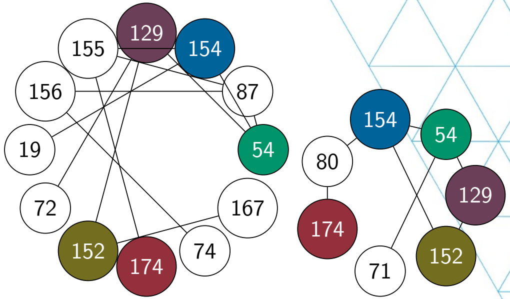
Dataset	Top-1↑	Top-5↑	Top-10↑	MRR↑
KKI	0.9978	0.9999	0.9999	0.9987
COX2	0.2513	0.6259	0.8395	0.4273
COX2_MD	0.9481	0.9828	0.9881	0.9630
DHFR	0.9999	0.9999	0.9999	0.9999
DBLP-v1	0.9994	0.9999	0.9999	0.9996
MSRC-21	0.9994	0.9999	0.9999	0.9999

Matching explanation

Qualitative analysis



(a) Subgraph isomorphism case. From left to right: The target graph and the isomorphic pattern graph



(b) Subgraph non-isomorphism case. From left to right: The target graph and the non-isomorphic pattern graph

Figure 16: Examples of isomorphism and non-isomorphism cases resulted from our model in the KKI dataset

Ablation study

- **Model Architecture:**
  - `cross-only`: This configuration exclusively employs interconnections between the graph and subgraph.
  - `intra-only`: This configuration solely relies on intra-connections within the graph and subgraph.
  - `both`: This configuration combines both intra- and inter-connections of the graph and subgraph.
- **Multi-hop Mechanism:**
  - `1-hop`: Here, we replace the GLeMA layer with a standard 1-hop GAT layer.
  - `increasing-hop`: This configuration employs the GLeMA with a continuously increasing number of hops in the deeper layers ($K^{(L_G)} = L_G$).
  - `interleaved-hop`: This configuration uses the GLeMA with an interleaving-increasing number of hops. In this study, we use $K^{(L_G)} = 2L_G - 1$.
- **Attention head:** We modify the base `xNeuSM` with 2 and 4 attention heads. These settings are used to understand the relationship between increasing model complexity (by increasing attention heads) and model performance.

Ablation study

Table 3: Impact of `xNeuSM` components on KKI

Model	Time↓	ROC↑	PR↑	F1↑	Acc↑	MRR↑
Cross-only 1-hop	0.56	0.979	0.959	0.979	0.979	**0.999**
Cross-only increasing-hop	0.60	0.977	0.956	0.977	0.977	0.996
Cross-only interleaved-hop	0.42	0.978	0.958	0.978	0.978	0.997
Intra-only 1-hop	0.49	0.611	0.578	0.485	0.612	–
Intra-only increasing-hop	0.40	0.628	0.593	0.515	0.628	–
Intra-only interleaved-hop	0.42	0.669	0.626	0.602	0.670	–
Both 1-hop	0.62	0.968	0.939	0.968	0.968	**0.999**
Both increasing-hop	0.70	**0.980**	0.963	**0.980**	**0.980**	**0.999**
Both interleaved-hop	0.51	0.979	**0.964**	**0.980**	0.979	0.998
With 2 attention heads	0.86	0.956	0.935	0.954	0.957	0.998
With 4 attention heads	1.19	0.938	0.923	0.936	0.937	**0.999**

Ablation study

Table 4: Impact of `xNeuSM` components on COX2

Model	Time↓	ROC↑	PR↑	F1↑	Acc↑	MRR↑
Cross-only 1-hop	0.12	0.967	0.946	0.968	0.967	0.306
Cross-only increasing-hop	0.12	0.962	0.931	0.964	0.962	0.191
Cross-only interleaved-hop	0.11	0.974	0.953	0.974	0.974	0.203
Intra-only 1-hop	0.11	0.472	0.498	0.007	0.472	–
Intra-only increasing-hop	0.11	0.457	0.499	0.003	0.458	–
Intra-only interleaved-hop	0.12	0.491	0.495	0.180	0.491	–
Both 1-hop	0.12	0.962	0.950	0.961	0.962	0.177
Both increasing-hop	0.39	0.972	0.961	0.972	0.972	0.298
Both interleaved-hop	0.38	**0.983**	**0.974**	**0.984**	**0.983**	**0.427**
With 2 attention heads	0.49	0.954	0.925	0.955	0.954	0.298
With 4 attention heads	0.63	0.907	0.892	0.900	0.907	0.215

Ablation study

Table 5: Impact of xNeuSM components on DBLP-v1

Model	Time↓	ROC↑	PR↑	F1↑	Acc↑	MRR↑
Cross-only 1-hop	0.07	**0.996**	**0.995**	0.996	**0.996**	0.996
Cross-only increasing-hop	0.09	0.980	0.964	0.981	0.980	0.983
Cross-only interleaved-hop	0.08	0.980	0.963	0.981	0.980	0.985
Intra-only 1-hop	0.09	0.640	0.598	0.579	0.640	–
Intra-only increasing-hop	0.13	0.643	0.593	0.634	0.643	–
Intra-only interleaved-hop	0.09	0.618	0.576	0.573	0.618	–
Both 1-hop	0.09	**0.996**	0.992	0.996	**0.996**	0.995
Both increasing-hop	0.10	0.918	0.910	0.912	0.918	0.989
Both interleaved-hop	0.13	**0.996**	**0.995**	**0.997**	**0.996**	**0.999**
With 2 attention heads	0.17	0.976	0.964	0.975	0.976	**0.999**
With 4 attention heads	0.28	0.986	0.984	0.985	0.986	**0.999**

Generalization

We additionally conduct experiments to demonstrate the generalization capabilities of `xNeuSM` in out-of-distribution settings. In these settings, we utilize the model trained on one dataset to test on the others, with the same datasets as in the previous experiments.

Generalization

We additionally conduct experiments to demonstrate the generalization capabilities of `xNeuSM` in out-of-distribution settings. In these settings, we utilize the model trained on one dataset to test on the others, with the same datasets as in the previous experiments.

Table 6: ROC AUC of out-distribution settings. For each dataset, the model trained on a different dataset that achieved the highest ROC AUC is in [*italic*].

Train \ Test	KKI	COX2	COX2_MD	DHFR	DBLP-v1	MSRC-21
KKI	**0.979**	0.634	0.499	*0.970*	*0.923*	*0.928*
COX2	0.500	**0.983**	0.500	0.500	0.501	0.500
COX2_MD	0.534	0.412	**0.986**	0.499	0.565	0.497
DHFR	0.547	0.797	0.499	**0.998**	0.758	0.668
DBLP-v1	0.502	*0.883*	0.491	0.689	**0.996**	0.505
MSRC-21	*0.863*	0.539	*0.604*	0.961	0.712	**0.997**

Generalization

Observations:

- A model trained on a dataset with a large $|\Sigma|$ exhibits generalization to datasets with smaller $|\Sigma|$ (trained on KKI and tested on DHFR, DBLP-v1, MSRC-21).
- A model trained on a dataset with a lower incidence of duplicated node graphs exhibits poor generalization to datasets characterized by a higher frequency of duplicated node graphs (no model trained on other datasets well generalizes to COX2, COX2_MD).
- Furthermore, a model trained on dense graphs can generalize to datasets with sparser graphs (trained on MSRC-21 and tested on DHFR, DBLP-v1).

Directed subgraph matching

We conducted an assessment of our proposed methodology using directed graphs. We effectuated the transformation of all edges within these datasets into directed edges, wherein we designated the tail node as the one with a smaller label and the head node as the one with a larger label.

Directed subgraph matching

We conducted an assessment of our proposed methodology using directed graphs. We effectuated the transformation of all edges within these datasets into directed edges, wherein we designated the tail node as the one with a smaller label and the head node as the one with a larger label.

Table 7: Performance of `xNeuSM` directed subgraph matching and matching explanation

Dataset	ROC↑	PR↑	F1↑	Acc↑	Top-1↑	Top-2↑	Top-10↑	MRR↑
KKI	0.975	0.953	0.975	0.975	0.996	0.999	0.999	0.998
COX2	0.947	0.908	0.949	0.947	0.103	0.396	0.640	0.261
COX2_MD	0.989	0.979	0.989	0.989	0.999	0.999	0.999	0.999
DHFR	0.969	0.944	0.970	0.969	0.999	0.999	0.999	0.999
DBLP-v1	0.960	0.940	0.960	0.960	0.745	0.996	0.999	0.866
MSRC-21	0.988	0.977	0.988	0.988	0.999	0.999	0.999	0.999

Table of Contents

Summary

- This thesis proposes a framework called xNeuSM for explainable neural subgraph matching using novel Graph Learnable Multi-hop Attention Networks.
- `xNeuSM` represents the structural attributes of graphs as adjacency matrices to explicitly capture cross-graph features between the query and target graphs.
- The learnable muti-hop attention mechanism is introduced to solve limitations of previous multi-hop attention one and its effectiveness of theoretically verified.
- The framework optimizes subgraph search measures through end-to-end neural networks while concurrently learning node alignments akin to classical combinatorial methods.
- Empirically, xNeuSM has the fastest execution time over baseline techniques, with comparable accuracy and explicit explainability.
- Published two papers including one at IJCNN 2023 conference(B-ranked)[36] and IEEE Access (Q1 journal)[37].

[36] T. T. Nguyen et al., "10X Faster Subgraph Matching: Dual Matching Networks with Interleaved Diffusion Attention".

[37] D. Q. Nguyen et al., "Explainable Neural Subgraph Matching With Learnable Multi-Hop Attention".

Future Developments

`xNeuSM` can be adapted for many other problems:

- Inexact subgraph matching, partial subgraph matching, geometric subgraph isomorphism, etc.
- Predicting molecule function, finding patterns in molecules, and other applications in the drug discovery process
- About the model: incorporating edge labels, utilizing more robust GNN modules, etc.
- Develop an efficient node-matching strategy for cases with a high number of similar nodes.

- THE END -

Thank you for your attention

Acknowledgements

We acknowledge Vingroup Innovation Foundation for supporting Master, PhD Scholarship, under code VINIF.2022.ThS.023.

Contact

nqduc@hcmut.edu.vn

Preliminaries

Labelled Directed Connected Graph

A labeled directed connected graph is a graph represented with a 3-tuple $\mathcal{G} = (V, E, l)$ where

1. V is a set of nodes,
2. $E \subseteq [V]^2$ is a set of edges (u, v), where u is tail node, v is head node and $u, v \in V$
3. $\forall v \in V, (\text{deg}_{in}(v) \geq 1) \vee (\text{deg}_{out}(v) \geq 1)$
4. $l : V \rightarrow \Sigma$ is a labelling function and Σ is a set of node labels

Preliminaries

Non-Induced Labelled Subgraph

Let $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G}, l_\mathcal{G})$ and $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S}, l_\mathcal{S})$ be two labelled graphs. $\mathcal{S}$ is a non-induced subgraph of $\mathcal{G}$ (denoted as $\mathcal{S} \subseteq_{ni} \mathcal{G}$) if and only if:

1. $\mathcal{S} \subseteq \mathcal{G}$ and
2. $\exists u, v \in V_\mathcal{S}, (u, v) \notin E_\mathcal{S} \wedge (u, v) \in E_\mathcal{G}$.

Preliminaries

Non-Induced Labelled Subgraph

Let $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}}, l_{\mathcal{G}})$ and $\mathcal{S} = (V_{\mathcal{S}}, E_{\mathcal{S}}, l_{\mathcal{S}})$ be two labelled graphs. $\mathcal{S}$ is a non-induced subgraph of $\mathcal{G}$ (denoted as $\mathcal{S} \subseteq_{ni} \mathcal{G}$) if and only if:

1. $\mathcal{S} \subseteq \mathcal{G}$ and
2. $\exists u, v \in V_{\mathcal{S}}, (u, v) \notin E_{\mathcal{S}} \wedge (u, v) \in E_{\mathcal{G}}$.

Non-Induced Subgraph Isomorphism

Given two graphs $\mathcal{P} = (V_{\mathcal{P}}, E_{\mathcal{P}}, l_{\mathcal{P}})$ and $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}}, l_{\mathcal{T}})$, $\mathcal{P}$ is considered as non-induced subgraph isomorphic to $\mathcal{T}$ if there exists $\mathcal{S} = (V_{\mathcal{S}}, E_{\mathcal{S}}, l_{\mathcal{S}})$ such that:

1. $\mathcal{S} \subseteq_{ni} \mathcal{T}$ and
2. $\mathcal{P} \cong \mathcal{S}$.

Subgraph Isomorphism Algorithms

There are two distinct settings for subgraph isomorphism algorithms

1. **Non-induced settings:** The pattern can be a partial embedding of a subgraph in the target graph.
$$\exists v_1, v_2 \in V_{\mathcal{P}} : (v_1, v_2) \notin E_{\mathcal{P}} \wedge (f(v_1), f(v_2)) \in E_{\mathcal{T}}$$
2. **Induced settings:** The pattern is a subgraph in the target graph.
$$\forall v_1, v_2 \in V_{\mathcal{P}} : (v_1, v_2) \notin E_{\mathcal{P}} \implies (f(v_1), f(v_2)) \notin E_{\mathcal{T}}$$

Design Principles & Challenges

To ensure the effectiveness of the graph neural network architecture, we consider three crucial properties:

- **(R1) Explanability.** A proficient subgraph matching framework must identify pattern presence and provide approximate "alignment witnesses".

[38] J. Xu, Wickramarathne, and Chawla, "Representing higher-order dependencies in networks".
[39] Sussman et al., "Matched Filters for Noisy Induced Subgraph Detection".
[40] Jiménez-Luna, Grisoni, and Schneider, "Drug discovery with explainable artificial intelligence".

Design Principles & Challenges

To ensure the effectiveness of the graph neural network architecture, we consider three crucial properties:

- **(R1) Explanability.** A proficient subgraph matching framework must identify pattern presence and provide approximate "alignment witnesses".
- **(R2) High-order Dependency.** Recent studies indicate complex systems often exhibit dependencies as high as fifth-order[38], necessitating scalable solutions for subgraph matching. However, elevating dependency orders can burden the model computationally.

[38] J. Xu, Wickramarathne, and Chawla, "Representing higher-order dependencies in networks".
[39] Sussman et al., "Matched Filters for Noisy Induced Subgraph Detection".
[40] Jiménez-Luna, Grisoni, and Schneider, "Drug discovery with explainable artificial intelligence".

Design Principles & Challenges

To ensure the effectiveness of the graph neural network architecture, we consider three crucial properties:

- **(R1) Explanability.** A proficient subgraph matching framework must identify pattern presence and provide approximate "alignment witnesses".
- **(R2) High-order Dependency.** Recent studies indicate complex systems often exhibit dependencies as high as fifth-order[38], necessitating scalable solutions for subgraph matching. However, elevating dependency orders can burden the model computationally.
- **(R3) Multi-task with Configurability.** Certain scenarios prioritize closely matched patterns over exact matches, such as in vaccine development[39]. Configuring the model to prioritize such scenarios aligns better with human intuition[40]. Designing a neural model seamlessly accommodating multiple objectives for solving a multi-task problem remains a significant architectural challenge.

[38] J. Xu, Wickramarathne, and Chawla, "Representing higher-order dependencies in networks".
[39] Sussman et al., "Matched Filters for Noisy Induced Subgraph Detection".
[40] Jiménez-Luna, Grisoni, and Schneider, "Drug discovery with explainable artificial intelligence".

Approach Overview

Our approach contains three main stages:

1. **Input representation:** Representing the pattern and target graph in matrices
2. **Feature extraction:** Utilizing Graph Learnable Multi-hop Attention layer to form high-level node features
3. **Task aggregation and intepretation:** Performing subgraph matching and matching explaination tasks concurrently

Approach Overview

Algorithm 2: xNeuSM Framework

```
Input  : Node feature matrix $\boldsymbol{X}$; Intra-graph adjacency matrix $\boldsymbol{A}^{in}$; Cross-graph adjacency
         matrix $\boldsymbol{A}^{cr}$
Output : Prediction $\hat{y}$; Weighted mapping matrix $\boldsymbol{P}$
1  $\boldsymbol{X}^0 \leftarrow \boldsymbol{X}$
2  for l in Range($1 \ldots L_G$) do
3      $\widehat{\boldsymbol{X}}^l_{in} \leftarrow \text{GLeMa}_l(\boldsymbol{X}^{l-1}, \boldsymbol{A}^{in})$
4      $\widehat{\boldsymbol{X}}^l_{cr} \leftarrow \text{GLeMa}_l(\boldsymbol{X}^{l-1}, \boldsymbol{A}^{cr})$
5      $\boldsymbol{X}^l \leftarrow \widehat{\boldsymbol{X}}^l_{cr} - \widehat{\boldsymbol{X}}^l_{in}$
6  end
7  $(\mathcal{A}^{(1)})^{L_G} \leftarrow \text{ExtractAttnMat}(\text{GLeMa}_{L_G}, \boldsymbol{X}^{l-1}, \boldsymbol{A}^{cr})$
8  $x^0_{repr} \leftarrow \frac{1}{|V_\mathcal{P}|} \sum_{i \in V_\mathcal{P}} x^{L_G}_i$, where $x^{L_G}_i \subset \boldsymbol{X}^{L_G}$
9  for l in Range($1 \ldots L_{FC} - 1$) do
10     $x^l_{repr} \leftarrow \delta(\boldsymbol{W}_l x^{l-1}_{repr} + b_l)$
11 end
12 $\widehat{y} \leftarrow \sigma(\boldsymbol{W}_y x^{L_{FC}-1}_{repr} + b_y)$
13 $\boldsymbol{P} \leftarrow \left\{ p_{ij} = \frac{1}{2}\left((a^{(1)}_{ij})^{L_G} + (a^{(1)}_{ji})^{L_G}\right) \right\}$, where $i \in V_\mathcal{P}, j \in V_\mathcal{T}$, and $(a^{(1)}_{ij})^{L_G} \in (\mathcal{A}^{(1)})^{L_G}$
```

Learnable Multi-hop Attention Mechanism

Proposition 2

The average approximate error of each element in $\mathcal{A}^{(K)} = \boldsymbol{Z}^{(K)}\boldsymbol{X'}^{-1}$ is bound by $(1-a)^{K+1}$

Learnable Multi-hop Attention Mechanism

Proposition 2

The average approximate error of each element in $\mathcal{A}^{(K)} = \boldsymbol{Z}^{(K)}\boldsymbol{X'}^{-1}$ is bound by $(1-a)^{K+1}$

Proof:
By Proposition 1, we can derive:

$$\begin{aligned} \lim_{K\to\infty} \boldsymbol{Z}^{(K)} &= \mathcal{A}\boldsymbol{X'} \\ \lim_{K\to\infty} \boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1} &= \mathcal{A}. \end{aligned} \tag{18}$$

Let $\mathcal{A}^{(K)} = \boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1}$ be the approximated attention diffusion matrix at K-hop. We will show that the error $\text{Err}(\mathcal{A} - \mathcal{A}^{(K)}) \leq (1-\alpha)^{K+1}$, where α is the teleport probability and K is the number of hops.

Learnable Multi-hop Attention Mechanism

Firstly, we decompose $\boldsymbol{Z}^{(K)}$ as following.

$$\begin{aligned}\boldsymbol{Z}^{(K)} &= (1-\alpha)^K(\mathcal{A}^{(1)})^K\boldsymbol{X'} + \alpha(1-\alpha)^{K-1}(\mathcal{A}^{(1)})^{K-1}\boldsymbol{X'} \\ &+ \cdots + \alpha(1-\alpha)(\mathcal{A}^{(1)})\boldsymbol{X'} + \alpha\boldsymbol{X'}\end{aligned} \tag{19}$$

Learnable Multi-hop Attention Mechanism

Firstly, we decompose $\boldsymbol{Z}^{(K)}$ as following.

$$\begin{aligned}\boldsymbol{Z}^{(K)} &= (1-\alpha)^K(\mathcal{A}^{(1)})^K\boldsymbol{X'} + \alpha(1-\alpha)^{K-1}(\mathcal{A}^{(1)})^{K-1}\boldsymbol{X'} \\ &+ \cdots + \alpha(1-\alpha)(\mathcal{A}^{(1)})\boldsymbol{X'} + \alpha\boldsymbol{X'}\end{aligned} \tag{19}$$

Then, we obtain:

$$\begin{aligned}\boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1} &= (1-\alpha)^K(\mathcal{A}^{(1)})^K + \alpha(1-\alpha)^{K-1}(\mathcal{A}^{(1)})^{K-1} \\ &+ \cdots + \alpha(1-\alpha)(\mathcal{A}^{(1)}) + \alpha \\ &= (1-\alpha)^K(\mathcal{A}^{(1)})^K + \sum_{k=0}^{K-1}\alpha(1-\alpha)^k(\mathcal{A}^{(1)})^k\end{aligned} \tag{20}$$

Learnable Multi-hop Attention Mechanism

Now, let us consider the difference between attention diffusion matrix $\mathcal{A}$ and its approximate form $\boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1}$.

Learnable Multi-hop Attention Mechanism

Now, let us consider the difference between attention diffusion matrix $\mathcal{A}$ and its approximate form $\boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1}$.

$$\begin{aligned}\mathcal{A} - \mathcal{A}^{(K)} &= \mathcal{A} - \boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1} \\ &= \sum_{k=0}^{\infty} \alpha(1-\alpha)^k(\mathcal{A}^{(1)})^k - (1-\alpha)^K(\mathcal{A}^{(1)})^K - \sum_{k=0}^{K-1} \alpha(1-\alpha)^k(\mathcal{A}^{(1)})^k\end{aligned}$$

Learnable Multi-hop Attention Mechanism

Now, let us consider the difference between attention diffusion matrix $\mathcal{A}$ and its approximate form $\boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1}$.

$$\begin{aligned}
\mathcal{A} - \mathcal{A}^{(K)} &= \mathcal{A} - \boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1} \\
&= \sum_{k=0}^{\infty} \alpha(1-\alpha)^k (\mathcal{A}^{(1)})^k - (1-\alpha)^K (\mathcal{A}^{(1)})^K - \sum_{k=0}^{K-1} \alpha(1-\alpha)^k (\mathcal{A}^{(1)})^k \\
&= \sum_{k=K}^{\infty} \alpha(1-\alpha)^k (\mathcal{A}^{(1)})^k - (1-\alpha)^K (\mathcal{A}^{(1)})^K
\end{aligned}$$

Learnable Multi-hop Attention Mechanism

Now, let us consider the difference between attention diffusion matrix $\mathcal{A}$ and its approximate form $\boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1}$.

$$\begin{aligned}
\mathcal{A} - \mathcal{A}^{(K)} &= \mathcal{A} - \boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1} \\
&= \sum_{k=0}^{\infty} \alpha(1-\alpha)^k(\mathcal{A}^{(1)})^k - (1-\alpha)^K(\mathcal{A}^{(1)})^K - \sum_{k=0}^{K-1} \alpha(1-\alpha)^k(\mathcal{A}^{(1)})^k \\
&= \sum_{k=K}^{\infty} \alpha(1-\alpha)^k(\mathcal{A}^{(1)})^k - (1-\alpha)^K(\mathcal{A}^{(1)})^K \\
&\leq \sum_{k=K}^{\infty} \alpha(1-\alpha)^k(\mathcal{A}^{(1)})^k - \alpha(1-\alpha)^K(\mathcal{A}^{(1)})^K \text{ by } \alpha, a_{ij}^{(1)} \in (0,1)
\end{aligned}$$

Learnable Multi-hop Attention Mechanism

Now, let us consider the difference between attention diffusion matrix $\mathcal{A}$ and its approximate form $\boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1}$.

$$\begin{aligned}
\mathcal{A} - \mathcal{A}^{(K)} &= \mathcal{A} - \boldsymbol{Z}^{(K)}(\boldsymbol{X'})^{-1} \\
&= \sum_{k=0}^{\infty} \alpha(1-\alpha)^k (\mathcal{A}^{(1)})^k - (1-\alpha)^K (\mathcal{A}^{(1)})^K - \sum_{k=0}^{K-1} \alpha(1-\alpha)^k (\mathcal{A}^{(1)})^k \\
&= \sum_{k=K}^{\infty} \alpha(1-\alpha)^k (\mathcal{A}^{(1)})^k - (1-\alpha)^K (\mathcal{A}^{(1)})^K \\
&\leq \sum_{k=K}^{\infty} \alpha(1-\alpha)^k (\mathcal{A}^{(1)})^k - \alpha(1-\alpha)^K (\mathcal{A}^{(1)})^K \text{ by } \alpha, a_{ij}^{(1)} \in (0,1) \\
&\leq \sum_{k=K+1}^{\infty} \alpha(1-\alpha)^k (\mathcal{A}^{(1)})^k
\end{aligned} \tag{21}$$

Learnable Multi-hop Attention Mechanism

We also have $a_{ij}^{(1)} \in (0, 1)$ so that

$$(\mathcal{A}^{(1)})^k \leq (\mathcal{A}^{(1)})^{k-1}. \tag{22}$$

Learnable Multi-hop Attention Mechanism

We also have $a_{ij}^{(1)} \in (0,1)$ so that

$$(\mathcal{A}^{(1)})^k \leq (\mathcal{A}^{(1)})^{k-1}. \tag{22}$$

As a consequence, we have:

$$(\mathcal{A}^{(1)})^k \leq \mathcal{A}^{(1)}, \forall k \geq 1 \tag{23}$$

Learnable Multi-hop Attention Mechanism

We also have $a_{ij}^{(1)} \in (0,1)$ so that

$$(\mathcal{A}^{(1)})^k \leq (\mathcal{A}^{(1)})^{k-1}. \tag{22}$$

As a consequence, we have:

$$(\mathcal{A}^{(1)})^k \leq \mathcal{A}^{(1)}, \forall k \geq 1 \tag{23}$$

Using (23), equation (21) can be derived as follows:

$$\boldsymbol{E} = \mathcal{A} - \mathcal{A}^{(K)} \leq \left(\sum_{k=K+1}^{\infty} \alpha(1-\alpha)^k \right) \mathcal{A}^{(1)} \tag{24}$$

Learnable Multi-hop Attention Mechanism

It is easy to observe that $\boldsymbol{E} \in \mathbb{R}^{|V|\times|V|}$. Then we can define the average difference between the exact and approximate attention diffusion matrix as

$$Err(\mathcal{A} - \mathcal{A}^{(K)}) = \frac{1}{|V|^2} \sum_{i,j} \boldsymbol{E}_{ij}$$

Learnable Multi-hop Attention Mechanism

It is easy to observe that $\boldsymbol{E} \in \mathbb{R}^{|V| \times |V|}$. Then we can define the average difference between the exact and approximate attention diffusion matrix as

$$
\begin{aligned}
Err(\mathcal{A} - \mathcal{A}^{(K)}) &= \frac{1}{|V|^2} \sum_{i,j} \boldsymbol{E}_{ij} \\
&\leq \frac{1}{|V|^2} \sum_{i,j} \left(\left(\sum_{k=K+1}^{\infty} \alpha(1-\alpha)^k \right) a_{ij}^{(1)} \right)
\end{aligned}
$$

Learnable Multi-hop Attention Mechanism

It is easy to observe that $\boldsymbol{E} \in \mathbb{R}^{|V|\times|V|}$. Then we can define the average difference between the exact and approximate attention diffusion matrix as

$$\begin{aligned} Err(\mathcal{A} - \mathcal{A}^{(K)}) &= \frac{1}{|V|^2} \sum_{i,j} \boldsymbol{E}_{ij} \\ &\leq \frac{1}{|V|^2} \sum_{i,j} \left(\left(\sum_{k=K+1}^{\infty} \alpha(1-\alpha)^k \right) a_{ij}^{(1)} \right) \\ &\leq \sum_{k=K+1}^{\infty} \alpha(1-\alpha)^k \text{ by } a_{ij}^{(1)} \in (0,1) \end{aligned}$$

Learnable Multi-hop Attention Mechanism

It is easy to observe that $\boldsymbol{E} \in \mathbb{R}^{|V|\times|V|}$. Then we can define the average difference between the exact and approximate attention diffusion matrix as

$$\begin{aligned}
Err(\mathcal{A} - \mathcal{A}^{(K)}) &= \frac{1}{|V|^2}\sum_{i,j}\boldsymbol{E}_{ij} \\
&\leq \frac{1}{|V|^2}\sum_{i,j}\left(\left(\sum_{k=K+1}^{\infty}\alpha(1-\alpha)^k\right)a_{ij}^{(1)}\right) \\
&\leq \sum_{k=K+1}^{\infty}\alpha(1-\alpha)^k \text{ by } a_{ij}^{(1)} \in (0,1) \\
&\leq \alpha\sum_{k=K+1}^{\infty}(1-\alpha)^k \leq \alpha\frac{(1-\alpha)^{K+1}}{1-(1-\alpha)}
\end{aligned}$$

Learnable Multi-hop Attention Mechanism

It is easy to observe that $\boldsymbol{E} \in \mathbb{R}^{|V|\times|V|}$. Then we can define the average difference between the exact and approximate attention diffusion matrix as

$$\begin{aligned} Err(\mathcal{A} - \mathcal{A}^{(K)}) &= \frac{1}{|V|^2}\sum_{i,j}\boldsymbol{E}_{ij} \\ &\leq \frac{1}{|V|^2}\sum_{i,j}\left(\left(\sum_{k=K+1}^{\infty}\alpha(1-\alpha)^k\right)a_{ij}^{(1)}\right) \\ &\leq \sum_{k=K+1}^{\infty}\alpha(1-\alpha)^k \text{ by } a_{ij}^{(1)} \in (0,1) \\ &\leq \alpha\sum_{k=K+1}^{\infty}(1-\alpha)^k \leq \alpha\frac{(1-\alpha)^{K+1}}{1-(1-\alpha)} \\ &\leq (1-\alpha)^{K+1} \end{aligned} \tag{25}$$

Thus, the Proposition 2 is proven.

Learnable Multi-hop Attention Mechanism

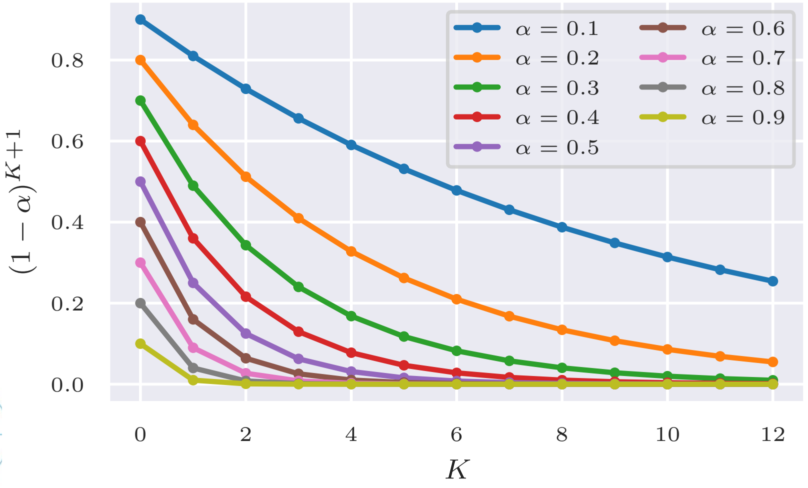


Figure 17: Maximal approximation error of each attention coefficient

Learnable multi-hop attention mechanism

Proposition 3

$$lim_{K\to\infty}\boldsymbol{Z}_{\beta}^{(K)} = \mathcal{A}_{\eta}\boldsymbol{X'}$$

Learnable multi-hop attention mechanism

Proposition 3

$lim_{K\to\infty}\boldsymbol{Z}_{\beta}^{(K)} = \mathcal{A}_{\eta}\boldsymbol{X'}$

Proof:

With $\beta_v \in (0,1), \forall v \in V$:

$$(\eta_v)_k = \beta_v(1-\beta_v)^k > 0 \tag{26}$$

Learnable multi-hop attention mechanism

Proposition 3

$$lim_{K\to\infty} \boldsymbol{Z}_{\beta}^{(K)} = \mathcal{A}_{\eta}\boldsymbol{X'}$$

Proof:

With $\beta_v \in (0,1), \forall v \in V$:

$$(\eta_v)_k = \beta_v(1-\beta_v)^k > 0 \quad (26)$$

This results in the important property:

$$\forall v \in V, \sum_{k=0}^{\infty}(\eta_v)_k = \sum_{k=0}^{\infty}\beta_v(1-\beta_v)^k = \frac{\beta_v}{1-(1-\beta_v)} = 1. \quad (27)$$

Learnable multi-hop attention mechanism

Let $\eta_k = \{(\eta_v)_k\}_{v=1}^{|V|}$ be the attention decay vector at the k-th hop. With the property in (27), we can generalize equation (9) as follows:

$$\begin{cases} (\mathcal{A}^{(1)})^0 & = \boldsymbol{I} \\ \mathcal{A}_\eta & = \sum_{k=0}^{\infty} \eta_k (\mathcal{A}^{(1)})^k. \end{cases} \tag{28}$$

Learnable multi-hop attention mechanism

Let $\eta_k = \{(\eta_v)_k\}_{v=1}^{|V|}$ be the attention decay vector at the k-th hop. With the property in (27), we can generalize equation (9) as follows:

$$\begin{cases} (\mathcal{A}^{(1)})^0 & = \boldsymbol{I} \\ \mathcal{A}_\eta & = \sum_{k=0}^{\infty} \eta_k (\mathcal{A}^{(1)})^k. \end{cases} \tag{28}$$

Then, we can approximate $\mathcal{A}_\eta \boldsymbol{X'}$ as:

$$\begin{cases} \boldsymbol{Z}^{(0)} & = \boldsymbol{X'} \\ \boldsymbol{Z}_\beta^{(k)} & = (\vec{1} - \beta)\mathcal{A}^{(1)} \boldsymbol{Z}^{(k-1)} + \beta \boldsymbol{Z}^{(0)}, \; k = \overline{1, K} \end{cases} \tag{29}$$

Learnable multi-hop attention mechanism

Firstly, we decompose all elements of $\boldsymbol{Z}_\beta^{(k)}$:

$$\begin{aligned}\boldsymbol{Z}_\beta^{(k)} &= \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}}\boldsymbol{X'} + \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k-1 \text{ times}}\beta\boldsymbol{X'} \\ &+ \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k-2 \text{ times}}\beta\boldsymbol{X'} + \dots + (\vec{1}-\beta)\mathcal{A}^{(1)}\beta\boldsymbol{X'} + \beta\boldsymbol{X'}\end{aligned} \tag{30}$$

Learnable multi-hop attention mechanism

Firstly, we decompose all elements of $\boldsymbol{Z}_{\beta}^{(k)}$:

$$\boldsymbol{Z}_{\beta}^{(k)} = \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}}\boldsymbol{X'} + \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k-1 \text{ times}}\beta\boldsymbol{X'} + \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k-2 \text{ times}}\beta\boldsymbol{X'} + \dots + (\vec{1}-\beta)\mathcal{A}^{(1)}\beta\boldsymbol{X'} + \beta\boldsymbol{X'} \tag{30}$$

We also have:

$$\begin{aligned} \mathcal{A}_{\eta}\boldsymbol{X'} &= \left(\sum_{k=0}^{\infty}\eta_k(\mathcal{A}^{(1)})^k\right)\boldsymbol{X'} \\ &= \eta_0\boldsymbol{X'} + \eta_1\mathcal{A}^{(1)}\boldsymbol{X'} + \eta_2(\mathcal{A}^{(1)})^2\boldsymbol{X'} + \dots \\ &= \beta\boldsymbol{X'} + \beta(\vec{1}-\beta)\mathcal{A}^{(1)}\boldsymbol{X'} + \beta(\vec{1}-\beta)^2(\mathcal{A}^{(1)})^2\boldsymbol{X'} + \dots \end{aligned} \tag{31}$$

Learnable multi-hop attention mechanism

To prove Proposition 3, we need to prove these lemmas.

Learnable multi-hop attention mechanism

To prove Proposition 3, we need to prove these lemmas.

Lemma 1

$$\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}}\beta\boldsymbol{X'} = \beta(\vec{1}-\beta)^k(\mathcal{A}^{(1)})^k\boldsymbol{X'}$$

Learnable multi-hop attention mechanism

To prove Proposition 3, we need to prove these lemmas.

Lemma 1

$$\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\ldots}_{k\text{ times}}\beta\boldsymbol{X'} = \beta(\vec{1}-\beta)^k(\mathcal{A}^{(1)})^k\boldsymbol{X'}$$

and

Lemma 2

$$\lim_{k\to\infty}\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\ldots}_{k\text{ times}}\boldsymbol{X'} = \vec{0}.$$

Learnable multi-hop attention mechanism

Lemma 1

$$\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k\text{ times}}\beta\boldsymbol{X'} = \beta(\vec{1}-\beta)^k(\mathcal{A}^{(1)})^k\boldsymbol{X'}$$

Learnable multi-hop attention mechanism

Lemma 1

$$\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\ldots}_{k \text{ times}}\beta\boldsymbol{X'} = \beta(\vec{1}-\beta)^k(\mathcal{A}^{(1)})^k\boldsymbol{X'}$$

Proof:

Due to the commutative property of row-wise multiplication between a vector and a matrix, we can write:

$$(\vec{1}-\beta)\mathcal{A}^{(1)} = \mathcal{A}^{(1)}(\vec{1}-\beta). \tag{32}$$

Learnable multi-hop attention mechanism

Lemma 1

$$\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}}\beta \boldsymbol{X'} = \beta(\vec{1}-\beta)^k(\mathcal{A}^{(1)})^k\boldsymbol{X'}$$

Proof:

Due to the commutative property of row-wise multiplication between a vector and a matrix, we can write:

$$(\vec{1}-\beta)\mathcal{A}^{(1)} = \mathcal{A}^{(1)}(\vec{1}-\beta). \tag{32}$$

This leads to:

$$\begin{aligned}\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}}\beta \boldsymbol{X'} &= \underbrace{(\vec{1}-\beta)(\vec{1}-\beta)\dots}_{k \text{ times}}\underbrace{\mathcal{A}^{(1)}\mathcal{A}^{(1)}\dots}_{k \text{ times}}\beta \boldsymbol{X'}\\ &= (\vec{1}-\beta)^k(\mathcal{A}^{(1)})^k\beta\boldsymbol{X'}\\ &= \beta(\vec{1}-\beta)^k(\mathcal{A}^{(1)})^k\boldsymbol{X'}.\end{aligned} \tag{33}$$

Learnable multi-hop attention mechanism

Lemma 2

$$\lim_{k\to\infty} \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}} \boldsymbol{X'} = \vec{0}.$$

Learnable multi-hop attention mechanism

Lemma 2

$$\lim_{k\to\infty} \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}}\boldsymbol{X'} = \vec{0}.$$

Proof:

Because $\beta_v \in (0,1)$, it follows that $(1-\beta_v) \in (0,1)$. This implies that:

$$\lim_{k\to\infty}(1-\beta_v)^k = 0.$$

Learnable multi-hop attention mechanism

Lemma 2

$$\lim_{k\to\infty} \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\ldots}_{k \text{ times}} \boldsymbol{X'} = \vec{0}.$$

Proof:

Because $\beta_v \in (0,1)$, it follows that $(1-\beta_v) \in (0,1)$. This implies that:

$$\lim_{k\to\infty} (1-\beta_v)^k = 0.$$

We also have:

$$(\vec{1}-\beta)^k = \{(1-\beta_v)^k\}_{v=1}^{|V|}, \tag{34}$$

Learnable multi-hop attention mechanism

Lemma 2

$$\lim_{k\to\infty} \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}} \boldsymbol{X'} = \vec{0}.$$

Proof:

Because $\beta_v \in (0,1)$, it follows that $(1-\beta_v) \in (0,1)$. This implies that:

$$\lim_{k\to\infty} (1-\beta_v)^k = 0.$$

We also have:

$$(\vec{1}-\beta)^k = \{(1-\beta_v)^k\}_{v=1}^{|V|}, \tag{34}$$

so that

$$\lim_{k\to\infty} (\vec{1}-\beta)^k = \{\lim_{k\to\infty} (1-\beta_v)^k\}_{v=1}^{|V|} = \vec{0}. \tag{35}$$

Learnable multi-hop attention mechanism

With Lemma 1 and (35), we can conclude:

$$\lim_{k\to\infty} \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}} \boldsymbol{X'} = \lim_{k\to\infty} (\vec{1}-\beta)^k (\mathcal{A}^{(1)})^k \boldsymbol{X'}$$

Learnable multi-hop attention mechanism

With Lemma 1 and (35), we can conclude:

$$\lim_{k\to\infty} \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}} \boldsymbol{X'} = \lim_{k\to\infty} (\vec{1}-\beta)^k(\mathcal{A}^{(1)})^k \boldsymbol{X'}$$

$$= (\lim_{k\to\infty} (\vec{1}-\beta)^k)(\lim_{k\to\infty} (\mathcal{A}^{(1)})^k \boldsymbol{X'})$$

Learnable multi-hop attention mechanism

With Lemma 1 and (35), we can conclude:

$$\begin{aligned}
\lim_{k\to\infty} \underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}} \boldsymbol{X'} &= \lim_{k\to\infty} (\vec{1}-\beta)^k (\mathcal{A}^{(1)})^k \boldsymbol{X'} \\
&= (\lim_{k\to\infty} (\vec{1}-\beta)^k)(\lim_{k\to\infty} (\mathcal{A}^{(1)})^k \boldsymbol{X'}) \\
&= \vec{0}(\lim_{k\to\infty} (\mathcal{A}^{(1)})^k \boldsymbol{X'}) \\
&= O
\end{aligned} \tag{36}$$

Learnable multi-hop attention mechanism

By proving Lemma 1 and Lemma 2, we can prove the Proposition 3 as follows.

$$\lim_{k\to\infty} \boldsymbol{Z}_{\beta}^{(k)} = \lim_{k\to\infty} \Big[\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k\text{ times}} \boldsymbol{X'} \Big] + \lim_{k\to\infty} \Big[\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k-1\text{ times}} \beta\boldsymbol{X'} + \cdots + (\vec{1}-\beta)\mathcal{A}^{(1)}\beta\boldsymbol{X'} + \beta\boldsymbol{X'} \Big]$$

Learnable multi-hop attention mechanism

By proving Lemma 1 and Lemma 2, we can prove the Proposition 3 as follows.

$$\begin{aligned}
\lim_{k\to\infty} \boldsymbol{Z}_\beta^{(k)} &= \lim_{k\to\infty} \Big[\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)} \dots}_{k \text{ times}} \boldsymbol{X'} \Big] \\
&+ \lim_{k\to\infty} \Big[\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)} \dots}_{k-1 \text{ times}} \beta \boldsymbol{X'} + \dots + (\vec{1}-\beta)\mathcal{A}^{(1)}\beta\boldsymbol{X'} + \beta\boldsymbol{X'} \Big] \\
&= O + \lim_{k\to\infty} \Big[\beta(\vec{1}-\beta)^{k-1}(\mathcal{A}^{(1)})^{k-1}\boldsymbol{X'} + \dots + \beta(\vec{1}-\beta)\mathcal{A}^{(1)}\boldsymbol{X'} + \beta\boldsymbol{X'} \Big]
\end{aligned}$$

Learnable multi-hop attention mechanism

By proving Lemma 1 and Lemma 2, we can prove the Proposition 3 as follows.

$$\begin{aligned}
\lim_{k\to\infty} \boldsymbol{Z}_\beta^{(k)} &= \lim_{k\to\infty} \Big[\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)} \dots}_{k \text{ times}} \boldsymbol{X'} \Big] \\
&+ \lim_{k\to\infty} \Big[\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)} \dots}_{k-1 \text{ times}} \beta \boldsymbol{X'} + \dots + (\vec{1}-\beta)\mathcal{A}^{(1)}\beta\boldsymbol{X'} + \beta\boldsymbol{X'} \Big] \\
&= O + \lim_{k\to\infty} \Big[\beta(\vec{1}-\beta)^{k-1}(\mathcal{A}^{(1)})^{k-1}\boldsymbol{X'} + \dots + \beta(\vec{1}-\beta)\mathcal{A}^{(1)}\boldsymbol{X'} + \beta\boldsymbol{X'} \Big] \\
&= \lim_{k\to\infty} \Big[\eta_{k-1}(\mathcal{A}^{(1)})^{k-1}\boldsymbol{X'} + \dots + \eta_1\mathcal{A}^{(1)}\boldsymbol{X'} + \eta_0\boldsymbol{X'} \Big]
\end{aligned}$$

Learnable multi-hop attention mechanism

By proving Lemma 1 and Lemma 2, we can prove the Proposition 3 as follows.

$$
\begin{aligned}
\lim_{k\to\infty} \boldsymbol{Z}_\beta^{(k)} &= \lim_{k\to\infty} \Big[\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}} \boldsymbol{X'}\Big] \\
&+ \lim_{k\to\infty} \Big[\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k-1 \text{ times}} \beta\boldsymbol{X'} + \dots + (\vec{1}-\beta)\mathcal{A}^{(1)}\beta\boldsymbol{X'} + \beta\boldsymbol{X'}\Big] \\
&= O + \lim_{k\to\infty} \Big[\beta(\vec{1}-\beta)^{k-1}(\mathcal{A}^{(1)})^{k-1}\boldsymbol{X'} + \dots + \beta(\vec{1}-\beta)\mathcal{A}^{(1)}\boldsymbol{X'} + \beta\boldsymbol{X'}\Big] \\
&= \lim_{k\to\infty} \Big[\eta_{k-1}(\mathcal{A}^{(1)})^{k-1}\boldsymbol{X'} + \dots + \eta_1\mathcal{A}^{(1)}\boldsymbol{X'} + \eta_0\boldsymbol{X'}\Big] \\
&= \lim_{k\to\infty} \left[\left(\sum_{k=0}^{k-1} \eta_k(\mathcal{A}^{(1)})^k\right)\boldsymbol{X'}\right]
\end{aligned}
$$

Learnable multi-hop attention mechanism

By proving Lemma 1 and Lemma 2, we can prove the Proposition 3 as follows.

$$\begin{aligned}
\lim_{k\to\infty} \boldsymbol{Z}_\beta^{(k)} &= \lim_{k\to\infty}\Big[\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k \text{ times}}\boldsymbol{X'}\Big] \\
&+ \lim_{k\to\infty}\Big[\underbrace{(\vec{1}-\beta)\mathcal{A}^{(1)}\dots}_{k-1 \text{ times}}\beta\boldsymbol{X'} + \cdots + (\vec{1}-\beta)\mathcal{A}^{(1)}\beta\boldsymbol{X'} + \beta\boldsymbol{X'}\Big] \\
&= O + \lim_{k\to\infty}\Big[\beta(\vec{1}-\beta)^{k-1}(\mathcal{A}^{(1)})^{k-1}\boldsymbol{X'} + \cdots + \beta(\vec{1}-\beta)\mathcal{A}^{(1)}\boldsymbol{X'} + \beta\boldsymbol{X'}\Big] \\
&= \lim_{k\to\infty}\Big[\eta_{k-1}(\mathcal{A}^{(1)})^{k-1}\boldsymbol{X'} + \cdots + \eta_1\mathcal{A}^{(1)}\boldsymbol{X'} + \eta_0\boldsymbol{X'}\Big] \\
&= \lim_{k\to\infty}\left[\left(\sum_{k=0}^{k-1}\eta_k(\mathcal{A}^{(1)})^k\right)\boldsymbol{X'}\right] \\
&= \mathcal{A}_\eta\boldsymbol{X'} \quad \text{as} \quad \mathcal{A}_\eta = \sum_{k=0}^{\infty}\eta_k(\mathcal{A}^{(1)})^k \quad \text{in (28).}
\end{aligned} \tag{37}$$