10X Faster Subgraph Matching: Dual Matching Networks with Interleaved Diffusion Attention

Thanh Toan Nguyen[1], Quang Duc Nguyen[2], Zhao Ren[3], Jun Jo[1], Quoc Viet Hung Nguyen[1], Thanh Tam Nguyen[1]

[1]*Griffith University, Australia,* [2]*HCMC University of Technology, Vietnam,*
[3]*L3S Research Center, Leibniz Universität Hannover, Germany*

2023 International Joint Conference on Neural Networks (IJCNN) | 978-1-6654-8867-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/IJCNN54540.2023.10191159

***Abstract*—The goal of subgraph matching is to determine the presence of a particular query pattern within a large collection of data graphs. Despite being a hard problem, subgraph matching is essential in various disciplines, including bioinformatics, text matching, and graph retrieval. Although traditional approaches could provide exact solutions, their computations are known to be NP-complete, leading to an overwhelmingly querying latency. While recent neural-based approaches have been shown to improve the response time, the oversimplified assumption of the first-order network may neglect the generalisability of fully capturing patterns in varying sizes, causing the performance to drop significantly in datasets in various domains. To overcome these limitations, this paper proposes `xDualSM`, a dual matching neural network model with interleaved diffusion attention. Specifically, we first embed the structural information of graphs into different adjacency matrices, which explicitly capture the intra-graph and cross-graph structures between the query pattern and the target graph. Then, we introduce a dual matching network with interleaved diffusion attention to carefully capture intra-graph and cross-graph information while reducing computational complexity. Empirically, our proposed framework not only boosted the speed of subgraph matching more than 10× compared to the fastest baseline but also achieved significant improvements of 47.64% in Recall and 34.39% in F1-score compared to the state-of-the-art approximation approach on COX2 dataset. In addition, our results are comparable with exact methods.**

***Index Terms*—subgraph matching, graph neural networks, graph mining, diffusion attention, matching by embedding**

I. Introduction

Due to the variety of publicly available graph data [1], significant efforts have been devoted to developing practical solutions for challenging graph analysis tasks over the past few decades. *Subgraph matching* is one of the most widespread problems when dealing with large data graphs. Given a query pattern, the goal of subgraph matching is to discover if the query pattern is isomorphic to a subgraph of a target graph. Despite the NP-completeness nature of the problem [2], subgraph matching is an essential task as it lies at the core of numerous application domains, including object molecular detection [3], text matching [4]–[6], and graph retrieval [7]. Given the great significance, subgraph matching has received considerable attention over the last several decades, and numerous approaches have been proposed [3], [8]–[13].

The conventionally exact methods [8]–[10] employ combinatorial search algorithms. Although the exact computation of subgraph matching could yield precise results, its scalability is severely limited due to the NP-completeness nature of the problem. More recent approaches [3], [11], [12] accelerate the searching process by establishing efficient matching orders and using heuristic filtering mechanisms to reduce the number of candidates in the data graph. Although these approaches enable the matching to scale to large target graphs, the query size cannot extend beyond a few tens of nodes. Therefore, most exact approaches suffer an overwhelming latency, leading to significant overheads, which is far from being desirable required by practical applications [14].

Given the importance, but the huge challenge, of subgraph matching, approximately neural-based approaches [13], [15] have been proposed to trade-off between speed and accuracy. These works demonstrate that by learning a matching function to approximate the matching metric, it is possible to locate the candidates for a query pattern more rapidly than traditional combinatorial approaches. Their learning algorithms heavily rely on first-order dependency in each layer of the graph neural network (GNN) architecture [13]. This suggests that the receptive field of a single GNN layer is limited to one-hop network neighbors. However, recent research showed that data obtained from multiple complex systems might exhibit up to fifth-order dependencies [16] rather than the first-order only. Oversimplifying the assumption of the first-order network may neglect the generalisability of fully capturing patterns in varying sizes, causing the performance to drop significantly in various domains (see Section VII-D for more detail).

Designing a learning-based method, which can achieve excellent efficiency while approximating the subgraph matching performance somewhat identical to classical methods, is notoriously hard. Such an approach induces several fundamental challenges: *(C1) High-order integration:* graphs contain patterns of varying scales that may be unknown beforehand. The algorithm must capture and utilise structurally high-order information and features of multiple granularities. *(C2) High computational complexity:* although stacking many GNN layers theoretically has the potential to broaden the receptive field and learn non-neighboring interactions, such deep GNN architectures are plagued by the over smoothing issue [17] and high computational complexity. *Cross-graph relation integration (C3):* while cross-graph relations have widely been proved to enhance performance on many graph analysis [18], integrating them to enhance performance without compromising efficiency introduces another critical challenge.

In this paper, we propose xDualSM, 10X faster subgraph matching: dual matching networks with interleaved diffusion attention. xDualSM addresses the three fundamental challenges for subgraph matching directly. In particular, we first embed the structural information between the query pattern and the data graph in two adjacency matrices: the first matrix represents the purely structural information of the graphs, and the second matrix represents the structures and their cross-graph interactions. By constructing these two matrices, we let our model learn how graph-graph interactions affect the node embedding during the learning representation. Then we introduce a novel variant of GNN—the dual matching network with interleaved attention diffusion—that is suitable for learning both graph presentation and graph-graph interaction (C3). By adopting the interleaved hop-by-hop schema [19], we achieve high-receptive field capturing (C1) while reducing computational complexity (C2). Finally, subtracting each feature of a target graph and a query graph will highlight the attention values between the case with and without the cross-graph interaction, and therefore is adopted for the subgraph matching task. It is worth noting that the dual matching network is trained in an end-to-end manner to facilitate the overall efficiency of the approach.

We summarise our contributions, as well as the structure of the paper, following some background (Section II), as follows:

- *xDualSM Framework:* Section III proposes a framework focuses on efficiently approximating the subgraph matching task. The framework carefully encodes high-order dependencies to enhance the approximate performance while greatly reducing computational complexity.
- *Construction of proxy inputs:* Section IV proposes a mapping mechanism to construct new data representations of the pattern and the data graph. The newly constructed representations facilitate both intra-graph and cross-graph representation.
- *Dual Matching Network with Interleaved Diffusion Attention*: Section V proposes a novel variant of GNN—the dual matching network with interleaved attention diffusion—that is suitable for learning both graph presentation and graph-graph interaction.
- *Embedding Aggregation:* Section VI aggregations the node embeddings to find the pattern's embedding. It is important to note that our model can be adapted to any matching metric by training its parameters.

In Section VII, we evaluate our approach using public datasets in various domains. Our proposed framework successfully not only boosted the speed of subgraph matching more than $10\times$ compared to the fastest baseline, but also achieved significant improvements of 47.64% in Recall and 34.39% in F1-score compared to *NeuroMatch*—the state-of-the-art approximation approach—on the COX2 dataset. In addition, our results are comparable with exact methods. Finally, we discuss related works in Section VIII before concluding the paper in Section IX.

II. Background

In this section, we set up a notation and briefly describe our broad goal in this paper.

A. Preliminaries

This study focuses on labelled, undirected, connected graphs. However, our framework is readily extensible to directed graphs. We summarised the notations used in the present study in Table I.

TABLE I: Summary of notation used

Symbol	Definition
D	a set of data graphs
T	the target graph ($T \in D$)
P	the query pattern
$\boldsymbol{A}^{in}$	proxy intra-graph adjacency matrix
$\boldsymbol{A}^{cr}$	proxy cross-graph adjacency matrix
h_i^{ℓ}	embedding of node v_i at ℓ-layer
H^{ℓ}	all node embeddings at ℓ-layer
$\mathcal{A}^1$	normalised 1-hop attention matrix
$\mathcal{A}^K$	normalised K-hop attention matrix

DEFINITION 1 (**LABELLED GRAPH**). *A labelled graph, or simply graph, is a 3-tuple $T = (V_T, E_T, L_T)$ where*

(1) V_T is a set of nodes,

(2) $E_T \subseteq [V_T]^2$ is a set of undirected edges (two-sets of nodes),

(3) $L_T : V_T \to \Sigma$ assigns labels to nodes, where Σ is a set of node labels.

We assume the alphabet of labels Σ to be defined as $\mathbb{R}^l$. That is, labels are l-dimensional real vectors. Next, we define the notions of a subgraph and a graph isomorphism.

DEFINITION 2 (**SUBGRAPH ISOMORPHISM (MATCH)**). *Two graphs $P = (V_P, E_P, L_P)$ and $T = (V_T, E_T, L_T)$ are* isomorphic*, if there exists a label-preserving bijection $f : V_P \to V_T$:*

(1) $\forall\ v \in V_P : L_P(v) = L_T(f(v))$, and

(2) $\forall\ \{v_1, v_2\} \in E_P : \{f(v_1), f(v_2)\} \in E_T$.

We consider a graph database as a set of graphs D. The answer set of a query pattern is then defined as the number of graphs $T \in D$ that contains P as an isomorphic subgraph.

B. Problem statement

Here, we formulate the central problem addressed by our proposed framework using the notation above.

PROBLEM 1 (**SUBGRAPH MATCHING QUERY**). *Given a set D of data graphs and a query pattern P, the subgraph matching query problem is to predict if P is isomorphic to a graph T in D.*

In subsequent sections, we will describe how to create a neural network architecture for this purpose and explain why such a design is appropriate for subgraph matching tasks.

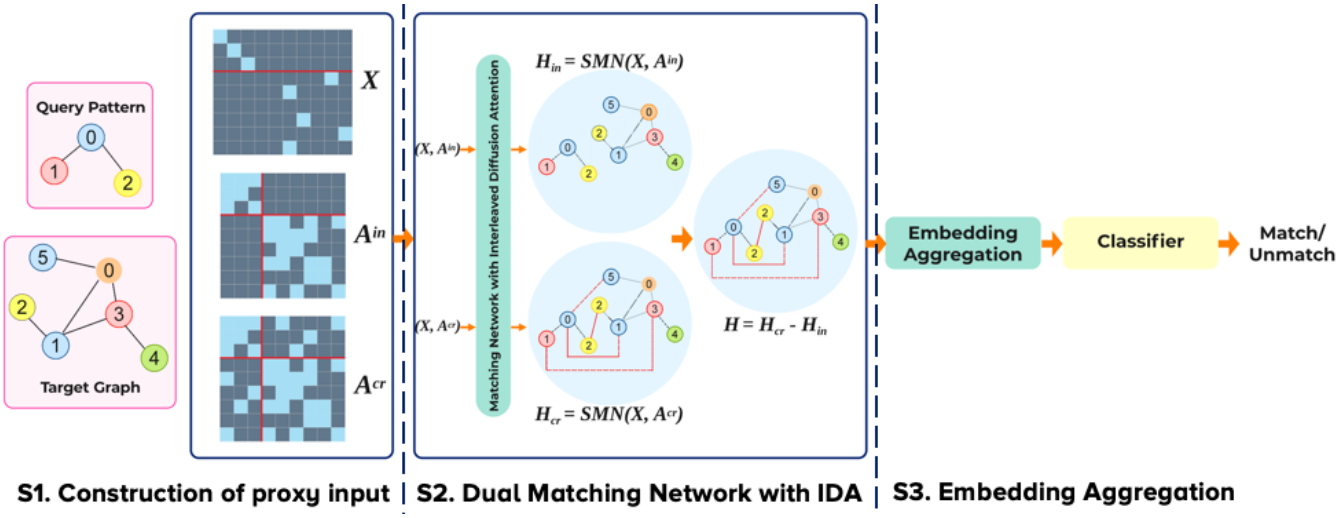

Fig. 1: **xDualSM framework**. In S1, the query pattern and data graph's structural information is encoded in two adjacency matrices. S2 learns both intra-graph and cross-graph representations, while S3 aggregates the features in an end-to-end process.

III. Overview Approach

A. Design Principles

Due to the nature of neural networks, xDualSM has a significant advantage in terms of efficiency. In terms of effectiveness, however, we must carefully design the neural network architecture to satisfy the three properties listed below:

- ***(R1) High-order dependency.*** Conventional network models that presume first-order dependency can be limited [16]. First-order network assumptions can neglect pattern scalability. Recent investigations reveal that complex system data can include up to fifth-order dependencies [16]. As we move towards a scalable solution for subgraph matching, high-order dependency representation becomes a must-have design principle.
- ***(R2) Cross-graph relation.*** While it has been well demonstrated that cross-graph relations improve the performance of various graph [18], combining them to improve performance without sacrificing efficiency is crucial for a subgraph matching task.
- ***(R3) Configurability.*** The model's parameters should be trained to match any metric. In some cases, nearly-matched patterns are more important than exactly-matched ones. For example, in vaccination development, the illness-to-be (closely-matched) candidate is more significant than the exactly diseased pattern (exactly-matched) since it helps respond to the disease early. In some scenarios, it is necessary to configure the model to meet human intuition [14].

B. The Approach

Based on design principles, we propose xDualSM, and its architecture is revealed in Fig. 1. The approach consists of three stages:

Stage 1: Construction of proxy inputs. Unlike earlier approximately neural-based approaches [13] that learn the pattern and the target graph directly from separated adjacency matrices, our newly created and unified proxy inputs enable capturing the cross-graph relation (**R2**), as well as enhancing the learning process simultaneously. Details of this stage are described in Section IV.

Stage 2: Dual matching network with interleaved diffusion attention. By adopting the interleaved hop-by-hop schema [19], we propose a novel variant of GNN—the dual matching network with interleaved diffusion attention. Our network achieves high-receptive field capturing (**R1**) while significantly reducing computational complexity. We present the details of this stage in Section V.

Stage 3: Embedding aggregation. To carry out the subgraph matching process, the final embedding is aggregated. It is important to note that our model can be adapted to any matching metric such as graph edit distance and graph similarity estimation [7] by training its parameters (**R3**). Details of this stage are presented in Section VI.

IV. Construction of Proxy Inputs

This section discusses the main steps by which the original pattern and the target graph are converted into novel input representations to facilitate the subsequent stages.

Given a target graph $T = \{V_T, E_T, L_T\}$, where we aim to identify the query pattern $P = \{V_P, E_P, L_P\}$. It is worth noting that L_T and L_P assign labels to nodes of the target graph and the query pattern, respectively. The node labels Σ are in l-dimensional real vectors, where l is the number of labels. The input for our proposed model consists of three components: the set of node feature vectors $\boldsymbol{X}$, the *proxy intra-graph* adjacency matrix $\boldsymbol{A}^{in}$, and the *proxy cross-graph* adjacency matrix $\boldsymbol{A}^{cr}$. The constructions of each component are outlined below:

Node feature vectors. Each pattern node or target graph node is encoded as a $2l$-dimension one-hot vector where l is the maximum number of distinct node labels. The first l dimensions are preserved for the input pattern, and the rest is

for the target graph. We separate pattern node features and graph node features because we desire the model to learn different embedding of pattern nodes and graph nodes, which results in better mapping performance. After that, all the node vectors are put together to form the input set $\boldsymbol{X}$ as follows.

$$\boldsymbol{X} = \{x_1, x_2, \ldots, x_{|V_P|}, x_{|V_P|+1}, \ldots, x_{|V_P|+|V_T|}\} \text{ where } x_i \in \mathbb{R}^{2l} \tag{1}$$

Proxy intra-graph adjacency matrix. The adjacency matrix $\boldsymbol{A}^{in}$ is created by flagging intra-graph edges. That means there are no proxy edges between the pattern and the transaction. The $\boldsymbol{A}^{in}$ is formally defined below.

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

Proxy cross-graph adjacency matrix. As opposed to $\boldsymbol{A}^{in}$, the $\boldsymbol{A}^{cr}$ matrix considers a "proxy" edge between a pattern node and a target graph node in case they have the same label. The $\boldsymbol{A}^{cr}$ is defined as follows.

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

Different from other approximately neural-based approaches [13] that learn the pattern and the target graph from separated inputs, our newly constructed proxy inputs accelerate the learning process of the dual matching network, which is revealed next.

V. Dual Matching Network with Interleaved Diffusion Attention

Here, we first present how a single layer of GNN works (Section V-A), and then we reveal how to incorporate the interleaved diffusion attention mechanism into each layer (Section V-B). Finally, we describe how to design a dual matching network for learning graph nodes representation to facilitate the subgraph matching task (Section V-C).

A. A Graph Neural Network Layer

A graph neural network (GNN) is similar to a convolutional neural network (CNN) in terms of weight sharing between ℓ layers [20]. Each GNN layer computes the hidden features for the nodes through recursive neighborhood aggregation. Each node representation derives its features from the previous layer and from its neighbors to preserve the local structure of the graph. Stacking of ℓ layers allows the model to learn node representation from the ℓ-hop neighborhood of the nodes. We, for simplicity, use the notions $\mathcal{N}_i$, $j \in \mathcal{N}_i$ to indicate $\mathcal{N}(v_i)$, $v_j \in \mathcal{N}(v_i)$, respectively.

Data input. Let (P, T) be a pair of graphs consisting of the pattern and the target graph. Without loss of generality, we consider a unified graph $G = P \cup T$. The neighboring relations between nodes in G are defined by an adjacency matrix $\boldsymbol{A}$. Depending on the semantic we desire to capture, adjacency matrix $\boldsymbol{A}$ can be either $\boldsymbol{A}^{in}$ or $\boldsymbol{A}^{cr}$, which is defined in the previous section. A node v_i of G has the node feature x_i, and it is encoded to a d-dimensional hidden feature ${h_i}^0$ using a simple linear mapping function before passing it into a graph neural network:

$${h^0}_i = \boldsymbol{W}^0 x_i + b^0 \tag{4}$$

where $\boldsymbol{W}^0 \in \mathbb{R}^{d \times 2l}, b^0 \in \mathbb{R}^d$.

GNN Layer. We denote by h_i^ℓ the feature vector of node i at the layer ℓ. The new feature vector $h_i^{\ell+1}$ for node i at layer $\ell+1$ is obtained by computing a non-linear transformation to the feature ${h_i}^\ell$ and the features ${h_j}^\ell$, where nodes j are the node's neighborhood i. Such transformation guarantees that the node representations are invariant to the graph size and node ordering. Formally, the feature of node i at the next layer $(\ell+1)$ is defined as:

$$h_i^{\ell+1} = \sigma\left(h^\ell_{\mathcal{N}_i}\right) \tag{5}$$

where $h^\ell_{\mathcal{N}_i}$ is the aggregation function from the set of neighbouring nodes j that points to node i, and σ is an activation function. The formation of $h^\ell_{\mathcal{N}_i}$ is defined as a weighted average of the following.

$$h^\ell_{\mathcal{N}_i} = \sum_{j \in \mathcal{N}_i} w_{ij} h_i^\ell \tag{6}$$

For example, in the context of Graph Convolutional Network (GCN [21]), the w_{ij} is determined by the degree of the nodes that the messages coming from (i.e., $w_{ij} = \sqrt{\frac{1}{d_i}} \cdot \sqrt{\frac{1}{d_j}}$). Instead of using a weighted average mechanism in which the weights only depend on the number of connections of the nodes, we adopt a function that calculates the weights. That function concerns the embeddings of both the source and the neighboring nodes; this allows the weights to depend on more than just the number of neighbors but instead should include anything the embedding could capture, such as data-driven features or dynamic local structures. Such function is called the attention function [22], [23], which is revealed next.

B. Interleaved Diffusion Attention Layer

The fixed weighted function for neighbor aggregation in GNN neglects the generalisation of the representation learning algorithm, especially for the challenging task like subgraph matching. Therefore, we propose an attention-based adaptive aggregation function in this section, to find the optimal weight from different neighborhood nodes.

Graph Attention. By adopting the attention mechanism, we replace the weighted aggregation w_{ij} in Eq. 6 with an attention function [24].

$$h^\ell_{\mathcal{N}_i} = \sum_{j \in \mathcal{N}_i} a(h_i^\ell, h_j^\ell) \times h_i^\ell \tag{7}$$

where a is a non-linear function learned by a deep neural network. Formulation of this function is defined as:

$$e_{ij} = a(h_i^\ell, h_j^\ell) = \sigma(\gamma^T.[\mathbf{W}h_i^\ell || \mathbf{W}h_j^\ell]) \tag{8}$$

where σ is an activation; γ, $\mathbf{W}$ are learnable vector and weight matrix, respectively. Next, the softmax function normalises all of the attention coefficients. We also apply the *masked attention*, which means that only the nodes $j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is the set of neighbor nodes of node i, are taken into consideration. The normalisation is defined as:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{j \in \mathcal{N}_i} \exp(e_{ij})} \tag{9}$$

Moreover, to strictly strike off the effect of non-neighbor nodes, we replace the normalised attention values of two nodes i and j with minus infinity in case there is no edge connecting i to j. Then, we obtain the 1-hop attention matrix $\mathcal{A}^1$ based on the normalise attention values as follows:

$$\mathcal{A}^1 = \begin{cases} \alpha_{ij}, & \text{if there is an edge that connects } j \text{ to } i \\ -\infty, & \text{otherwise} \end{cases} \tag{10}$$

Interleaved Diffusion Attention Layer. Recent studies showed that data derived from many complex systems could show up to fifth-order dependencies [16]. After having the 1-hop attention matrix, we can calculate the K-hop attention matrix $\mathcal{A}^K$ to allow the effect of non-neighbor nodes by a procedure called *attention diffusion* [22] given in Eq. 11. As we adopt the idea of interleaved hop-by-hop schema [19], the K is interleaved (i.e., $K = 1, 3, 5, 7..$) to increase the receptive field without compromising the efficiency.

$$\mathcal{A}^K = \sum_{i=0}^{K} \theta_i \mathcal{A}^i \tag{11}$$

where $\sum_{i=1}^{K} \theta_i = 1$ and $\theta_i > 0$. In Eq. 11), θ_i is the attention decay factor in ensuring the further the nodes are, the less important they acquire. Next, for each node i, we perform the weighted sum itself with other nodes using the K-hop attention matrix.

To accelerate the learning process, we vectorise the features of all input nodes and feed them to the network at once rather than computing each node separately. As a result, estimating embeddings for all nodes can be performed efficiently using sparse matrix operation. The equivalent vector form for the batch input can be re-written as follows:

$$\text{InDiff}(G, H^\ell, \Theta) = \mathcal{A}^K \mathbf{LN}(H^\ell) \tag{12}$$

where H^ℓ is the vector form of all hidden node features and layer ℓ, and the Θ is the set of paratermers for interleaved attention computation, and **LN** is the layer normalisation. Since we calculate the attention diffusion recursively, the layer normalisation is added to help stabilise the recurrent computation procedure [25].

Finally, we apply Multi-head Attention [22] to get different features from many distinct perspectives. The vectors created by Multi-head Attention are then concatenated to create the final node refined feature vectors. The final output of this layer is defined as:

$$H^\ell = \left(\Big\Vert_{m=1}^{M} \delta.\text{InDiff}_{(m)} \right) \boldsymbol{W} = \left(\Big\Vert_{m=1}^{M} \delta \left(\mathcal{A}_{(m)}^K \mathbf{LN}(H^\ell) \right) \right) \boldsymbol{W} \tag{13}$$

where M is the number of heads of Multi-head Attention, $\mathcal{A}_{(m)}^K$ is the normalized K-hop attention matrix of head m, and $\boldsymbol{W}$ is a learnable weight matrix. We called the results computed from Eq. 13 is the output of a single interleaved diffusion attention (IDA) layer. The IDA layer is used as a building block to firm the matching network, which is presented in the next section.

C. Dual Matching Network with Interleaved Diffusion Attention

We design a subgraph matching network (**SMN**) to capture the relations between the pattern and the target graph. Each matching network relies on the IDA layer as the building block, and we stack multiple IDA layers to ensure the generalisability of the network (we empirically prove that four layers of IDA are adequate for the matching network in this study). In addition, as we can capture both intra-graph and cross-graph relations, we apply a dual matching network based on two different input representations (in Section IV), and their computations are as follows.

$$\begin{cases} H_{in}^\ell &= \mathbf{SMN}(\boldsymbol{X}, \boldsymbol{A}^{in}) \\ H_{cr}^\ell &= \mathbf{SMN}(\boldsymbol{X}, \boldsymbol{A}^{cr}) \end{cases} \tag{14}$$

Finally, as we desire to contrast the situation with and without cross-graph relations, the final output of the dual matching network is computed as:

$$H^\ell = H_{cr}^\ell - H_{in}^\ell \tag{15}$$

The final output H^ℓ will be aggregated and performed the matching task in the final stage.

VI. Embedding Aggregation

Toward the subgraph matching task, after being refined by the dual matching network, the node feature vectors from patterns are then aggregated to produce a representation vector. This vector is used to decide whether the input pattern is isomorphism to the target transaction or not through a classifier containing L_{FC} fully-connected layers. Eq. 16 gives way how to calculate the representation vector for the pattern, and equation Eq. 17 defines the formulations behind the classifier.

$$h_P^\ell = \sum_{i \in V_\mathcal{P}} h_i^\ell \tag{16}$$

Here, we employ a deep neural network as the classifier. More precisely, the computation is outlined as follows.

$$\begin{cases} x_P^{(i)} & = \sigma(\boldsymbol{W}_i x_P^{(i-1)} + b_i), i = \overline{1, L_{FC}-1} \\ \hat{y} & = \sigma(\boldsymbol{W}_y x_P^{(L_{FC}-1)} + b_y) \end{cases} \quad (17)$$

where $x_P^{(i)}$ is the feature of the i-th fully connected layer, and $x_P^{(0)}$ is assigned by h_P^{ℓ}. Finally, the cross-entropy loss is integrated, making `xDualSM` feasible as an end-to-end learning algorithm.

$$Loss = -\frac{1}{N}\sum_{k=1}^{N}(y_k \cdot \log(\hat{y}_k) + (1-y_k) \cdot \log(1-\hat{y}_k)) \quad (18)$$

where $\hat{y}_k$, y_k are the predicted output at k-layer and the ground truth, respectively. Note that our approach is generic and can incorporate any optimisation problem such as graph edit distance and graph similarity [7] (satisfying **R3**).

VII. Experiment

This section first describes evaluation protocols for `xDualSM`, and then conducts experiments on four real-world datasets to evaluate `xDualSM`'s performance. Experiments are designed to specifically address the following research questions (**RQs**): **RQ1**: Does `xDualSM` outperform the competing baselines on diverse metrics? **RQ2**: Is `xDualSM` efficient compared to approximate approaches? **RQ3**: How well does `xDualSM` approximate the exact approaches?

Before providing our empirical findings and results, we will first describe the experimental setup.

A. Experimental Datasets

To evaluate the effectiveness of `xDualSM`, we conduct experiments on four real-world datasets[1] that are commonly used in graph mining research [13], [26]–[29]. To evaluate the generalisation capability of our proposed approach, we choose datasets that spans a cross various domains including bioinformatics (KKI [27]), chemical (COX2 [26]), computer vision (MSRC [29]), and social networks (DBLP [28]). The summary statistics of four datasets are presented in Table II.

TABLE II: Statistics of the datasets.

Domain	Dataset	#Graph	#Node	#Edge
Bioinformatics	KKI [27]	83	2.2K	4.0K
Chemical	COX2 [26]	467	19.3K	20.3K
Computer Vision	MSRC [29]	563	43.64K	111.65K
Social Networks	DBLP [28]	19,456	203.90K	381.31K

B. Evaluation protocols

Given a collection of query pattern $\mathcal{P}$ to query and a collection of data graph D, we split $\mathcal{P}$ into 60%, 15%, and 25% for training, validation, and testing, respectively. We train `xDualSM` and the baselines using the training set and then use the validation set to select the corresponding parameters.

As it is common in the evaluation of algorithms for the subgraph matching problem [11]–[13], we examine *average execution time* of query sets, which exclude the data loading time from disks. Besides, we consider the additional five metrics for evaluating the subgraph search task as follows.

- *Precision:* The precision is calculated by dividing the number of true positive query samples by the number of positively classified queries.
- *Recall:* The recall is computed by dividing the number of true positive query samples by the number of positive ground truth samples.
- *Accuracy:* The accuracy is defined as the ratio of correctly classified query samples to the total number of samples, with true positives rewarded and false positives penalised.
- *Weighted F1-score:* The score is determined by taking the harmonic mean of Precision and Recall and weighting it by the number of true query instances in each class.
- *AUC score:* The AUC score is a metric for classification problems with different settings. The higher the AUC score the model achieves, the better the model is.

C. Baseline techniques

We compare our approach with both *exact* and *approximate* approaches. For *exact* methods, we implemented six different representative models, which are outlined below.

- *VF2 [8].* This is a state-of-the-art algorithm for exact matching that uses backtracking as its core.
- *QuickSI [9].* This is an algorithm that employs the infrequent-edge-first ordering technique.
- *GraphQL [10].* This is an algorithm that applies the neighborhood signature filter and left-deep join ordering.
- *TurboIso [3]* This is a strategy for subgraph isomorphism search that explores only candidate regions that could be matched with the query graph.
- *CECI [11].* Based on the Compact Embedding Cluster Index, this approach divides the data graph into many embedding clusters for parallel processing.
- *DAF [12].* This study provides a fast solution for subgraph matching that incorporates dynamic programming, directed acyclic graph (DAG) ordering, and early pruning.

For *approximate* methods, we adapt *NeuralMatch [13]*, which is the most recent neural approach for graph matching.

D. End-to-end comparison with neural approximate method

To answer the first question (**RQ1**) and (**RQ2**), we assess the performance of `xDualSM` in an end-to-end manner, focusing on six different metrics.

As we are in line with neural-based approaches for subgraph matching problems, we mainly compare our results with *NeuroMatch* [13]—the most recent and state-of-the-art neural approach for subgraph matching. The results are presented in Table III. From the results, we can see that `xDualSM` outperformed *NeuroMatch* in four datasets over all employed metrics. More precisely, while the query *excution time* is crucial for contemporary applications, `xDualSM` achieves an average improvement of $\times 8$ speedup compared to *NeuroMatch*. Moreover, we achieve the execution time that is $\times 12$ and $\times 13$ faster on COX2 and DBLP, respectively.

Considering the importance of matching accuracy, we also present the results of matching performance using all employed metrics. Overall, we see that `xDualSM` outperforms

[1]https://chrsmrrs.github.io/datasets/docs/datasets/

Datasets	Methods	Metrics					
		Execution time (s/query)	AUC score	Precision	Recall	Accuracy	F1-score
KKI	NeuroMatch	0.00196	0.8652	0.7713	0.8214	0.7956	0.7896
	xDualSM	**0.00051**	**0.9799**	**0.9687**	**0.9916**	**0.9799**	**0.9798**
COX2	NeuroMatch	0.00243	0.8837	0.8581	0.5097	0.6395	0.7127
	xDualSM	**0.00021**	**0.9835**	**0.9809**	**0.9861**	**0.9835**	**0.9835**
MSRC	NeuroMatch	0.00454	0.9851	0.9664	0.9914	0.9787	0.9784
	xDualSM	**0.00115**	**0.9956**	**0.9970**	**0.9971**	**0.9969**	**0.9970**
DBLP	NeuroMatch	0.00163	0.7357	0.6917	0.6929	0.6923	0.6921
	xDualSM	**0.00012**	**0.9663**	**0.9554**	**0.9781**	**0.9667**	**0.9662**

TABLE III: Comparison with the neural-based approximate approach.

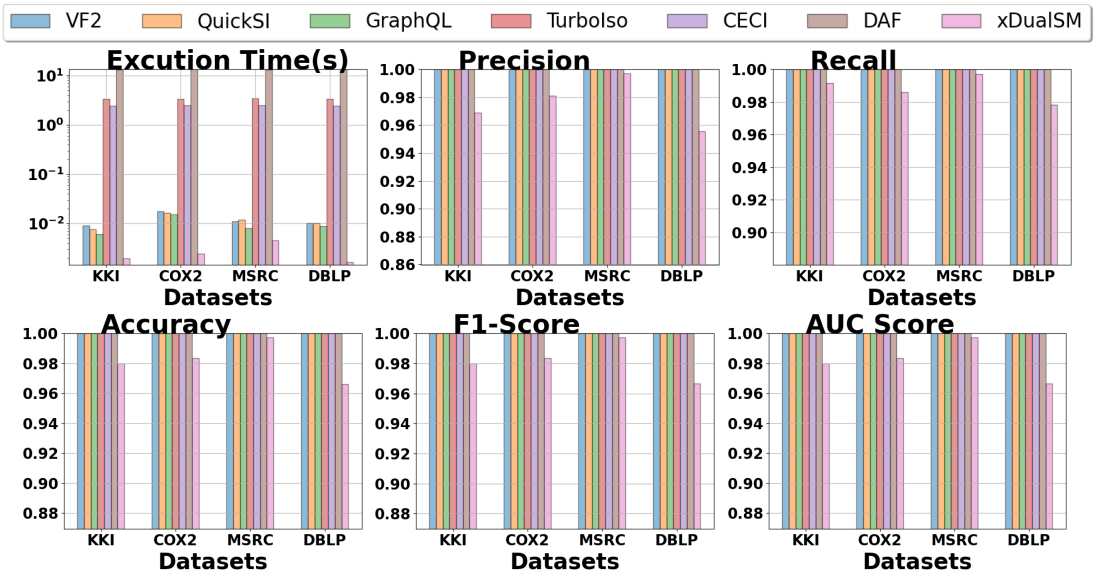

Fig. 2: Comparison with traditional exact approaches.

the competing baseline, indicating a solid capacity to match subgraphs efficiently. Although *NeuroMatch* presents a comparable performance to `xDualSM` on the MSRC dataset, its performance significantly drops in other datasets. The reason could be that without a high-order integration, *NeuroMatch* well represent a dense graph like MSRS, while it is less efficient in sparser datasets. Therefore, `xDualSM` achieved significant improvements of 47.64% in Recall and 34.39% in F1-score compared to *NeuroMatch* on COX2 dataset. For other metrics, we achieve an average of 10%-20% improvements to the baseline.

E. End-to-end comparison with exact methods

To answer the first question (**RQ1**) and (**RQ3**), we assess the performance of `xDualSM` in an end-to-end manner with exact methods, employing all metrics in the study.

In this section, we compare the performance of `xDualSM` to six representatively exact approaches, and the results are presented in Fig. 2. Overall, we see that `xDualSM` achieves the competitive performance compared to all exact methods over all datasets. However, our approach's execution improves significantly compared with other baselines. More precisely, it achieved average improvements of $\times 40$ faster than the best baselines (i.e., GraphQL). We explain the improvement in time complexity using parameters. Specifically, we define E_T as the number of edges in the target graph, E_P as the number of edges in the query pattern, V_T as the number of nodes in the target graph, and V_P as the number of nodes in the query pattern. Our approach takes $O(2K(|E_T|+|E_P|))$, where K is the number of GNN layers. In comparison, the best baseline (GraphQL) has a time complexity of $O(|V_T| \times |E_P|) + |V_T| \times |V_P|$ for the simplest pattern. Additionally, many other exact methods have exponential time complexity to complete the matching process [3], [9].

VIII. Related Works

This section briefly presents the related work on subgraph matching and graph neural networks.

Subgraph matching. The subgraph matching task is an NP-complete problem [30], and it has attracted a great deal of research efforts [3], [8]–[13] over the last decades. These approaches [8] aim to achieve effectiveness in matching orders and develop robust pruning procedures to reduce the number of intermediate candidates. QuickSI [9] implements the *infrequent-edge-first ordering* strategy, which sorts edges of the query pattern according to the frequencies with which they appear in the target graph. Similarly, GraphQL [10] employs a strategy, namely the *left-deep-join ordering* strategy, to model the candidate enumerating process as a joining problem. TurboIso [3]— a path-based ordering algorithm—provides a matching order based on the decomposition of the query

pattern into many pathways and sorting them by estimated embeddings. Compact Embedding Cluster Index (CECI [11]) is a new framework for listing subgraphs that partitions the target graph into many embedding clusters to facilitate parallel processing. DAF [12] provides a novel solution to the subgraph matching problem by constructing a queryDAG rather than a spanning tree.

With recent advances in graph neural networks, NeuroMatch [13] has been proposed to provide a trade-off between speed and accuracy. However, it relies on the oversimplifying assumption of first-order network dependency. Therefore, it neglects the generalisability of fully capturing graphs in varying sizes, which causes a significant performance drop in many datasets in various domains.

Graph Neural Network. Graph embedding vectorizes a graph's nodes to a numerical representation while preserving its structure [31]. Techniques for graph embedding can be classified into three categories: (i) *matrix-factorisation* techniques [32], which use the adjacency matrix to learn the representation through matrix decomposition directly; (ii) *random-walk* techniques [33], which do a random walk and learn the embedding in such a way that the decoder optimises the co-occurrences of nodes that appeared in the same walk; (iii) Deep learning techniques, such as graph convolutional networks [34], capture the attributes assigned to nodes and the graph's structure in a single model.

With `xDualSM`, we contribute a novel variant of GNN—a dual matching network based on diffusion attention [22]. Unlike existing models, it preserves intra-graph and cross-graph relations between the query pattern and the target graph. As a result, `xDualSM` fully captures high-order dependencies in graph data, thereby facilitating the accuracy of subgraph matching while reducing computational complexity.

IX. Conclusion

This paper proposes `xDualSM`, a dual matching network with interleaved diffusion attention that significantly boosts the subgraph matching speed. `xDualSM` comprises of three steps: *1) construction of proxy inputs*, *2) dual matching network with interleaved diffusion attention*, and *3) embedding aggregation*. Our experiments were conducted with four real-world datasets in various domains against seven different baselines, and the results illustrated that the proposed approach is:

- *Efficient*: `xDualSM` achieved the execution time 12× and 13× faster than the best approximate approach on COX2 and DBLP, respectively. In addition, we achieved average improvements of 40× faster than the best exact baseline.
- *Accurate:* `xDualSM` achieved significant improvements of 47.64% in Recall and 34.39% in F1-score compared to the best approximate approach on the COX2 dataset while achieving comparable performance with exact methods.

For future works, we intend to extend our framework to more universal settings such as graph similarity search, graph edit distance, and graph retrieval [7].

References

[1] S. Sahu, A. Mhedhbi, S. Salihoglu, J. Lin, and M. T. Özsu, "The ubiquity of large graphs and surprising challenges of graph processing: extended survey," *The VLDB journal*, vol. 29, no. 2, pp. 595–618, 2020.

[2] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the third annual ACM symposium on Theory of computing*, 1971, pp. 151–158.

[3] W.-S. Han, J. Lee, and J.-H. Lee, "Turboiso: towards ultrafast and robust subgraph isomorphism search in large graph databases," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 2013, pp. 337–348.

[4] C. Wang, X. Xu, Z. Wei, W. Zhang, and Z. Liu, "Lsgc: An interactive text matching model combined with enhanced encoding," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–7.

[5] Y. Wang, M. Xu, Y. Yan, T. Zhao, Y. Chen, and J. Yang, "Exploring topic supervision with bert for text matching," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–7.

[6] Y. Zuo, X. Peng, W. Lu, S. Wang, Z. Li, W. Zhang, and Y. Zhai, "Chinese sentence matching with multiple alignments and feature augmentation," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–8.

[7] R. J. Trudeau, *Introduction to graph theory*. Courier Corporation, 2013.

[8] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub) graph isomorphism algorithm for matching large graphs," *IEEE transactions on pattern analysis and machine intelligence*, 2004.

[9] H. Shang, Y. Zhang, X. Lin, and J. X. Yu, "Taming verification hardness: an efficient algorithm for testing subgraph isomorphism," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 364–375, 2008.

[10] H. He and A. K. Singh, "Graphs-at-a-time: query language and access methods for graph databases," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008.

[11] B. Bhattarai, H. Liu, and H. H. Huang, "Ceci: Compact embedding cluster index for scalable subgraph matching," in *Proceedings of the 2019 International Conference on Management of Data*, 2019.

[12] M. Han, H. Kim, G. Gu, K. Park, and W.-S. Han, "Efficient subgraph matching: Harmonizing dynamic programming, adaptive matching order, and failing set together," in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 1429–1446.

[13] Z. Lou, J. You, C. Wen, A. Canedo, J. Leskovec *et al.*, "Neural subgraph matching," *arXiv preprint arXiv:2007.03092*, 2020.

[14] J. Jiménez-Luna, F. Grisoni, and G. Schneider, "Drug discovery with explainable artificial intelligence," *Nature Machine Intelligence*, 2020.

[15] K. Xu, L. Wang, M. Yu, Y. Feng, Y. Song, Z. Wang, and D. Yu, "Cross-lingual knowledge graph alignment via graph matching neural network," *arXiv preprint arXiv:1905.11605*, 2019.

[16] J. Xu, T. L. Wickramarathne, and N. V. Chawla, "Representing higher-order dependencies in networks," *Science advances*, 2016.

[17] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," *arXiv preprint arXiv:1905.10947*, 2019.

[18] Y. Bai, H. Ding, K. Gu, Y. Sun, and W. Wang, "Learning-based efficient graph similarity computation via multi-scale convolutional set matching," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3219–3226.

[19] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks," in *IEEE Symposium on Security and Privacy*. IEEE, 2004.

[20] M. Kissel and K. Diepold, "Convolutional neural networks with analytically determined filters," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–7.

[21] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[22] G. Wang, R. Ying, J. Huang, and J. Leskovec, "Multi-hop attention graph neural network," *arXiv preprint arXiv:2009.14332*, 2020.

[23] S. Smyl, G. Dudek, and P. Pelka, "Es-drnn with dynamic attention for short-term load forecasting," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–8.

[24] Y. Song, H. Mao, and H. Li, "Spatio-temporal modeling for air quality prediction based on spectral graph convolutional network and attention mechanism," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–9.

[25] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.
[26] J. J. Sutherland, L. A. O'brien, and D. F. Weaver, "Spline-fitting with a genetic algorithm: A method for developing classification structure-activity relationships," *Journal of chemical information and computer sciences*, vol. 43, no. 6, pp. 1906–1915, 2003.
[27] S. Pan, J. Wu, X. Zhu, G. Long, and C. Zhang, "Task sensitive feature exploration and learning for multitask graph classification," *IEEE transactions on cybernetics*, vol. 47, no. 3, pp. 744–758, 2016.
[28] S. Pan, X. Zhu, C. Zhang, and S. Y. Philip, "Graph stream classification using labeled and unlabeled graphs," in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 2013, pp. 398–409.
[29] M. Neumann, R. Garnett, C. Bauckhage, and K. Kersting, "Propagation kernels: efficient graph kernels from propagated information," *Machine Learning*, vol. 102, no. 2, pp. 209–245, 2016.
[30] H. R. Lewis, "Michael r. πgarey and david s. johnson. computers and intractability. a guide to the theory of np-completeness. wh freeman and company, san francisco1979, x+ 338 pp." *The Journal of Symbolic Logic*, vol. 48, no. 2, pp. 498–500, 1983.
[31] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, 2018.
[32] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *KBS*, vol. 151, pp. 78–94, 2018.
[33] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD*, 2014, pp. 701–710.
[34] J. Zhou, Z. Wang, J. Meng, S. Liu, J. Zhang, and S. Chen, "Human interaction recognition with skeletal attention and shift graph convolution," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–8.