# Iteratively Refined Early Interaction Alignment for Subgraph Matching and Retrieval

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Graph retrieval based on subgraph isomorphism has several real-world applications such as scene graph retrieval, molecular fingerprint detection and circuit design. We present EINSMATCH, an early interaction graph neural network (GNN) tailored for this task, supervised by pairwise preference between graphs instead of explicit alignments. We propose several technical innovations in the design of EINSMATCH. First, we compute embeddings of all nodes by passing messages within and across the two input graphs, guided by an *injective alignment* between their nodes. Second, we update this alignment in a lazy fashion over multiple *rounds*. Within each round, we run a layerwise GNN from scratch, based on the current state of the alignment. After the completion of one round of GNN, we use the last-layer embeddings to update the alignments, and proceed to the next round. Third, EINSMATCH incorporates a novel notion of node-pair partner interaction. Traditional early interaction computes attention between a node and its potential partners in the other graph, the attention then controlling messages passed across graphs. In contrast, we consider *node pairs* (not single nodes) as potential partners. Existence of an edge between the nodes in one graph and non-existence in the other provide vital signals for refining the alignment. Our experiments on several datasets show that the alignments get progressively refined with successive rounds, resulting in significantly better retrieval performance than existing methods. We demonstrate that all three innovations contribute to the enhanced accuracy.

## 1 Introduction

In graph retrieval based on subgraph isomorphism, the goal is to identify a subset of graphs from a corpus, denoted $\{G_c\}$, wherein each retrieved graph contains a subgraph isomorphic to a given query graph $G_q$. Numerous real-life applications, *e.g.*, molecular fingerprint detection [6], scene graph retrieval [16], circuit design [29] and frequent subgraph mining [43], can be formulated using subgraph isomorphism. Akin to other retrieval systems, the key challenge is to efficiently score corpus graphs against queries.

Recent work on neural graph retrieval [1, 2, 11, 22, 23, 35, 31, 46] has shown significant promise. Among them, Lou et al. [23, Neuromatch] and Roy et al. [35, IsoNet] focus specifically on subgraph isomorphism. They employ graph neural networks (GNNs) to obtain embeddings of query and corpus graphs and compute the relevance score using a form of order embedding [39]. In addition, IsoNet also approximates an *injective alignment* between the query and corpus graphs. These two models operate in a *late interaction* paradigm, where the representations of the query and corpus graphs are computed independent of each other. In contrast, GMN [22] is a powerful *early interaction* network for graph matching, where GNNs running on $G_q$ and $G_c$ interact with each other at every layer.

Conventional wisdom suggests that early interaction is more accurate (even if slower) than late interaction, but GMN was outperformed by IsoNet. This is because of the following reasons. (1) GMN does not explicitly infer any alignment between $G_q$ and $G_c$. The graphs are encoded by two

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

GNNs that interact with each other at every layer, mediated by attentions from each node in one graph on nodes in the other. These attentions are functions of node embeddings, so they change from layer to layer. While these attentions may be interpreted as approximate alignments, they induce at best non-injective mappings between nodes. (2) In principle, one wishes to propose a consistent alignment across all layers. However, GMN's attention based 'alignment' is updated in every layer. (3) GMN uses a standard GNN that is known to be an over-smoother [36, 40]. Due to this, the attention weights (which depend on the over-smoothed node representations) also suffer from oversmoothing. These limitations raise the possibility of a *third* approach based on early interaction networks, enabled with explicit alignment structures, that have the potential to outperform both GMN and IsoNet.

### 1.1 Our contributions

We present EINSMATCH, an early interaction network for subgraph matching that maintains a chain of explicit, iteratively refined, injective, approximate alignments between the two graphs.

**Early interaction GNNs with alignment refinement** We design early interaction networks for scoring graph pairs, that ensure the node embeddings of one graph are influenced by both its paired graph and the alignment map between them. In contrast to existing works, we model alignments as an explicit "data structure". An alignment can be defined between either nodes or edges. This leads us to develop two variants of our model: EINSMATCH (Node) and EINSMATCH (Edge). Within EINSMATCH, we maintain a sequence of such alignments and refine them using GNNs acting on the two graphs. These alignments mediate the interaction between the two GNNs. In our work, we realize the alignments as a doubly stochastic approximation to a permutation matrix, which is an injective correspondence by design.

**Eager or lazy alignment updates** In our work, we view the updates to the alignment maps as a form of gradient-based updates in a specific quadratic assignment problem or asymmetric Gromov-Wasserstein (GW) distance minimization [30, 41]. The general form of EINSMATCH allows updates that proceed lockstep with GNN layers (*eager* layer-wise updates), but it also allows *lazy* updates. Specifically, EINSMATCH can perform $T$ *rounds* of updates to the alignment, each round including $K$ *layers* of GNN message passing. During each round, the alignment is held fixed across all propagation layers in GNN. At the end of each round, we update the alignment by feeding the node embeddings into a neural Gumbel-Sinkhorn soft permutation generator [10, 26, 37].

**Node-pair partner interaction between graphs** The existing remedies to counter oversmoothing [8, 33, 40] entail extra computation; but they may be expensive in an early-interaction setting. In contrast to existing works [22] which perform node partner interaction, we perform node-pair partner interaction. Specifically, when computing the message on the edge $(u, v) \in G_q$, we borrow the representation of $(u', v') \in G_c$ Consequently, the embedding of node $u$ explicitly captures signals from nodes in $G_c$, that share soft correspondences with the *neighbors* of $u$ in $G_q$.

**Experiments** The design components of EINSMATCH and their implications are subtle — we report on extensive experiments that tease out their effects. Our experiments on real world datasets show that, EINSMATCH outperforms several state-of-the-art methods for graph retrieval by a substantial margin. Moreover, our results suggest that capturing information directly from node-pair partners can improve representation learning, as compared to taking information only from node partner.

## 2 Preliminaries

**Notation** Given graph $G = (V, E)$, we use $\text{nbr}(u)$ to denote the neighbors of a node $u \in V$. We use $u \to v$ to indicate a message flow from node $u$ to node $v$. Given a set of corpus graphs $C = \{G_c\}$ and a query graph $G_q$, we denote $y(G_c \,|\, G_q)$ as the binary relevance label of $G_c$ for $G_q$. Motivated by several real life applications like substructure search in molecular graphs [12], object search in scene graphs [16], and text entailment [20], we consider subgraph isomorphism to significantly influence the relevance label, similar to previous works [23, 35]. Specifically, $y(G_c \,|\, G_q) = 1$ when $G_q$ is a subgraph of $G_c$, and 0 otherwise. We define $C_{q+} \subseteq C$ as the set of corpus graphs that are relevant to $G_q$ and set $C_{q-} = C \backslash C_{q+}$. Mildly overloading notation, we use $\boldsymbol{P}$ to indicate a 'hard' (0/1) permutation matrix or its 'soft' doubly-stochastic relaxation. $\mathcal{B}_n$ denotes the set of all $n \times n$ doubly stochastic matrices, and $\Pi_n$ denotes the set of all $n \times n$ permutation matrices.

**Graph neural network [14, 18, 21, 22, 38, 42] (GNN)** Given a graph $G = (V, E)$, a GNN initializes node representations $\{\boldsymbol{h}_0(u) : u \in V\}$ using node-local features. Then, messages are

passed between neighboring nodes in $K$ *propagation layers*. In the $k$th layer, a node $u$ receives messages from its neighbors, aggregates them, and then combines the result with its state after the $(k-1)$th layer:

$$\boldsymbol{h}_k(u) = \mathrm{comb}_\theta\left(\boldsymbol{h}_{k-1}(u), \textstyle\sum_{v\in \mathrm{nbr}(u)} \{\mathrm{msg}_\theta(\boldsymbol{h}_{k-1}(u), \boldsymbol{h}_{k-1}(v))\}\right) \tag{1}$$

Here, $\mathrm{msg}_\theta(\cdot)$ and $\mathrm{comb}_\theta(\cdot,\cdot)$ are suitable networks with parameters collectively called $\theta$. Edges may also be featurized and influence the messages that are aggregated [24]. The node representations at the final propagation layer $K$ can be collected into the matrix $\boldsymbol{H} = \{\boldsymbol{h}_K(u) \,|\, u \in V\}$. Given a node $u \in G_q$ and a node $u' \in G_c$, we denote the embeddings of $u$ and $u'$ after the propagation layer $k$ as $\boldsymbol{h}_k^{(q)}(u)$ and $\boldsymbol{h}_k^{(c)}(u')$ respectively. $\boldsymbol{H}^{(q)}$ and $\boldsymbol{H}^{(c)}$ denote the $K$th-layer node embeddings of $G_q$ and $G_c$, collected into matrices.

Using Eq. (1) on $G_q$ and $G_c$ separately, we can formulate a **late** interaction network, where we first compute the set of vectors $\boldsymbol{H}^{(q)}$ and $\boldsymbol{H}^{(c)}$ independent of the other graph, and then compare these sets following the general pattern $\hat{y}(G_c \,|\, G_q) = \mathrm{sim}\left(\boldsymbol{H}^{(c)} \,|\, \boldsymbol{H}^{(q)}\right)$. Since subgraph isomorphism defines an asymmetric relevance, $\mathrm{sim}\left(\boldsymbol{H}^{(c)} \,|\, \boldsymbol{H}^{(q)}\right) \neq \mathrm{sim}\left(\boldsymbol{H}^{(q)} \,|\, \boldsymbol{H}^{(c)}\right)$. We may also define a distance $\Delta(\boldsymbol{H}^{(c)} \,|\, \boldsymbol{H}^{(q)})$ which is inversely related to $\mathrm{sim}\left(\boldsymbol{H}^{(c)} \,|\, \boldsymbol{H}^{(q)}\right)$.

In an **early** interaction network, $\boldsymbol{H}^{(q)}$ depends on $G_c$ and $\boldsymbol{H}^{(c)}$ depends on $G_q$ for any given $(G_q, G_c)$ pair. Formally, one should write $\boldsymbol{H}^{(q\,|\,c)}$ and $\boldsymbol{H}^{(c\,|\,q)}$ instead of $\boldsymbol{H}^{(q)}$ and $\boldsymbol{H}^{(c)}$ respectively for an early interaction network, but for simplicity, we will continue using $\boldsymbol{H}^{(q)}$ and $\boldsymbol{H}^{(c)}$.

**Our goal** Given a set of corpus graphs $C = \{G_c \,|\, c \in [|C|]\}$, our high-level goal is to build a graph retrieval model so that, given a query $G_q$, it can return the corpus graphs $\{G_c\}$ which are relevant to $G_q$. To that end, we seek to develop (1) a GNN-based early interaction model, and (2) an appropriate distance measure $\Delta(\cdot \,|\, \cdot)$, so that $\Delta(\boldsymbol{H}^{(c)} \,|\, \boldsymbol{H}^{(q)})$ is an accurate predictor of $y(G_c \,|\, G_q)$, at least to the extent that $\Delta(\cdot|\cdot)$ is effective for ranking candidate corpus graphs in response to a query graph.

# 3 Proposed early-interaction GNN with multi-round alignment refinement

In this section, we first write down the subgraph isomorphism task as an instance of the quadratic assignment problem (QAP) or the Gromov-Wasserstein (GW) distance optimization task. Then, we design EINSMATCH, by building upon this formulation.

## 3.1 Subgraph isomorphism as Gromov-Wasserstein distance optimization

**QAP or GW formulation with asymmetric cost** We are given a graph pair $G_q$ and $G_c$ padded with appropriate number of nodes to ensure $|V_q| = |V_c| = n$ (say). Let their adjacency matrices be $\boldsymbol{A}_q, \boldsymbol{A}_c \in \{0,1\}^{n\times n}$. Consider the family of hard permutation matrices $\boldsymbol{P} \in \Pi_n$ where $\boldsymbol{P}[u,u'] = 1$ indicates $u \in V_q$ is "matched" to $u' \in V_c$. Then, $G_q$ is a subgraph of $G_c$, if for some permutation matrix $\boldsymbol{P}$, the matrix $\boldsymbol{A}_q$ is covered by $\boldsymbol{P}\boldsymbol{A}_c\boldsymbol{P}^\top$, *i.e.*, for each pair $(u,v)$, whenever we have $\boldsymbol{A}_q[u,v] = 1$, we will also have $\boldsymbol{P}\boldsymbol{A}_c\boldsymbol{P}^\top[u,v] = 1$. This condition can be written as $\boldsymbol{A}_q \leq \boldsymbol{P}\boldsymbol{A}_c\boldsymbol{P}^\top$. We can regard a deficit in coverage as a cost or distance:

$$\mathrm{cost}(\boldsymbol{P}; \boldsymbol{A}_q, \boldsymbol{A}_c) = \textstyle\sum_{u\in[n], v\in[n]} \left[\left(\boldsymbol{A}_q - \boldsymbol{P}\boldsymbol{A}_c\boldsymbol{P}^\top\right)_+\right][u,v] \tag{2}$$

$$= \textstyle\sum_{u,v\in[n]} \sum_{u',v'\in[n]} (\boldsymbol{A}_q[u,v] - \boldsymbol{A}_c[u',v'])_+ \, \boldsymbol{P}[u,u'] \, \boldsymbol{P}[v,v'] \tag{3}$$

Here, $[\cdot]_+ = \max\{\cdot, 0\}$ is the ReLU function, applied elementwise. The function $\mathrm{cost}(\boldsymbol{P}; \boldsymbol{A}_q, \boldsymbol{A}_c)$ can be driven down to zero using a suitable choice of $\boldsymbol{P}$ iff $G_q$ is a subgraph of $G_c$. This naturally suggests the relevance distance

$$\Delta(G_c \,|\, G_q) = \min_{\boldsymbol{P}\in\Pi_n} \mathrm{cost}(\boldsymbol{P}; \boldsymbol{A}_q, \boldsymbol{A}_c) \tag{4}$$

Xu et al. [41] demonstrate that this QAP is a realization of the Gromov-Wassterstein distance minimization in a graph setting.

**Updating $\boldsymbol{P}$ with projected gradient descent** As shown in Benamou et al. [3], Peyré et al. [30], Xu et al. [41], one approach is to first relax $\boldsymbol{P}$ into a doubly stochastic matrix, which serves as a continuous approximation of the discrete permutation, and then update it using projected gradient descent (PGD). Here, the soft permutation $\boldsymbol{P}_{t-1}$ is updated to $\boldsymbol{P}_t$ at time-step $t$ by solving the following linear optimal

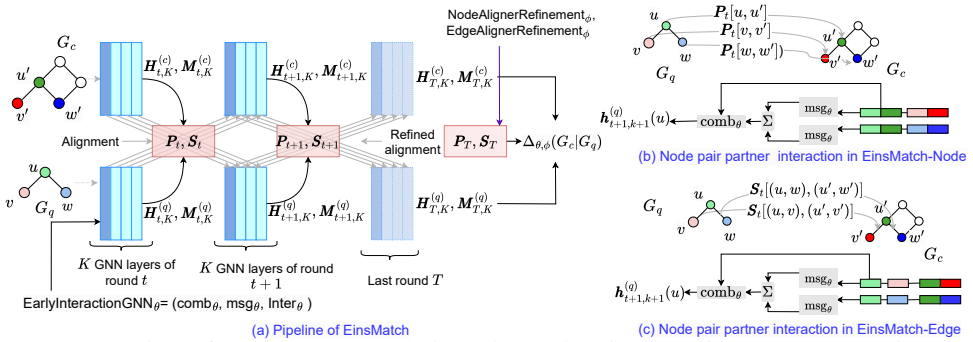



Figure 1: Overview of EINSMATCH. Panel (a) shows the pipeline of EINSMATCH. Given a graph pair $(G_q, G_c)$, we execute $T$ *rounds*, each consisting of $K$ GNN *layer* propagations. After a round $t$, we use the node embeddings to update the node alignment $\boldsymbol{P} = \boldsymbol{P}_t$ from its previous estimate $\boldsymbol{P} = \boldsymbol{P}_{t-1}$. Within each round $t \in [T]$, we compute the node embeddings of $G_q$ by gathering signals from $G_c$ and vice-versa, using GNN embeddings in the previous round and the node-alignment map $\boldsymbol{P}_t$. The alignment $\boldsymbol{P}_t$ remains consistent across all propagation layers $k \in [K]$ and is updated at the end of round $t$. Panel (b) shows our proposed node pair partner interaction in EINSMATCH (Node). When computing the message value of the node pair $(u, v)$, we also feed the node embeddings of the partners $u'$ and $v'$ in addition to the embeddings of the pairs $(u, v)$, where $u'$ and $v'$ is approximately aligned with $u$ and $v$, respectively. Panel (c) shows the node pair partner interaction in EINSMATCH (Edge). In contrast to EINSMATCH (Node), here we feed the information from the message value of the partner pair $(u', v')$ instead of their node embeddings into the message passing network $\mathrm{msg}_\theta$.

transport (OT) problem, regularized with the entropy of $\{\boldsymbol{P}[u, v] \mid u, v \in [n]\}$ with a temperature $\tau$.

$$\boldsymbol{P}_t \leftarrow \operatorname*{arg\,min}_{\boldsymbol{P} \in \mathcal{B}_n} \mathrm{Trace}\left(\boldsymbol{P}^\top \nabla_{\boldsymbol{P}} \mathrm{cost}(\boldsymbol{P}; \boldsymbol{A}_q, \boldsymbol{A}_c)\big|_{\boldsymbol{P}=\boldsymbol{P}_{t-1}}\right) + \tau \sum_{u,v} \boldsymbol{P}[u, v] \cdot \log \boldsymbol{P}[u, v]. \tag{5}$$

Such an OT problem is solved using the iterative Sinkhorn-Knopp algorithm [10, 37, 26]. Similar to other combinatorial optimization problems on graphs, a QAP (3) does not capture the coverage cost in the presence of dense node or edge features, where two nodes or edges may exhibit graded degrees of similarity represented by continuous values. Furthermore, the binary values of the adjacency matrices result in inadequate gradient signals in $\nabla_{\boldsymbol{P}} \mathrm{cost}(\cdot)$. Additionally, the computational bottleneck of solving a fresh OT problem in each PGD step introduces a significant overhead, especially given the large number of pairwise evaluations required in typical learning-to-rank setups.

### 3.2 Design of EINSMATCH (Node)

Building upon the insights from the above GW minimization (2) and the successive refinement step (5), we build EINSMATCH (Node), the first variant of our proposed early interaction model.

**Node-pair partner interactions between graphs** For simpler exposition, we begin by describing a synthetic scenario, where $\boldsymbol{P}$ is a hard node permutation matrix, which induces the alignment map as a bijection $\pi : V_q \to V_c$, so that $\pi(a) = b$ if $\boldsymbol{P}[a, b] = 1$. We first initialize layer $k = 0$ embeddings as $\boldsymbol{h}_0^{(q)}(u) = \mathrm{Init}_\theta(\mathrm{feature}(u))$ using a neural network $\mathrm{Init}_\theta$. (Throughout, $\boldsymbol{h}_k^{(c)}(u)$ are treated likewise.) Under the given alignment map $\pi$, a simple early interaction model would update the node embeddings as follows:

$$\boldsymbol{h}_{k+1}^{(q)}(u) = \mathrm{comb}_\theta\left(\boldsymbol{h}_k^{(q)}(u),\ \textstyle\sum_{v \in \mathrm{nbr}(u)} \mathrm{msg}_\theta(\boldsymbol{h}_k^{(q)}(u), \boldsymbol{h}_k^{(q)}(v)),\ \boldsymbol{h}_k^{(c)}(\pi(u))\right) \tag{6}$$

In the above expression, the update layer uses representation of the partner node $u' \in V_c$ during the message passing step, to compute $\boldsymbol{h}_{k+1}^{(q)}(u)$, the embedding of node $u \in V_q$. Li et al. [22] use a similar update protocol, by approximating $\boldsymbol{h}_k^{(c)}(\pi(u)) = \sum_{u' \in V_c} a_{u' \to u}^{(k)} \boldsymbol{h}_k^{(c)}(u')$, where $a_{u' \to u}^{(k)}$ is the $k$th layer attention from $u \in V_q$ to potential partner $u' \in V_c$, with $\sum_{u' \in V_c} a_{u' \to u}^{(k)} = 1$. Instead of regarding only nodes as potential partners, EINSMATCH will regard *node pairs* as partners. Given $(u, v) \in E_q$, the partners $(\pi(u), \pi(v)) \in E_c$ should then greatly influence the intensity of assimilation of $\boldsymbol{h}_k^{(c)}(u')$ into $\boldsymbol{h}_{k+1}^{(c)}(u)$. The first key innovation in EINSMATCH is to replace (6) to recognize and

implement this insight:

$$\boldsymbol{h}^{(q)}_{k+1}(u) = \mathrm{comb}_\theta\Big([\boldsymbol{h}^{(q)}_k(u), \boldsymbol{h}^{(c)}_k(\pi(u))], \sum\nolimits_{v\in\mathrm{nbr}(u)} \mathrm{msg}_\theta\left([\boldsymbol{h}^{(q)}_k(u), \boldsymbol{h}^{(c)}_k(\pi(u))], [\boldsymbol{h}^{(q)}_k(v), \boldsymbol{h}^{(c)}_k(\pi(v))]\right)\Big) \quad (7)$$

Embeddings $\boldsymbol{h}^{(c)}_{k+1}(u')$ for nodes $u' \in V_c$ are updated likewise in a symmetric manner. The network $\mathrm{msg}_\theta$ is provided embeddings from partners $\pi(u), \pi(v)$ of $u, v \in V_q$ — this allows $\boldsymbol{h}^{(\bullet)}_{k+1}(u)$ to capture information from all nodes in the paired graph, that match with the $(k+1)$-hop neighbors of $u$.

**Multi-round lazy refinement of node alignment** In reality, we are not given any alignment map $\pi$. This motivates our second key innovation beyond prior models [1, 22, 23, 35], where we decouple GNN layer propagation from updates to $\boldsymbol{P}$. To achieve this, EINSMATCH (Node) executes $T$ *rounds*, each consisting of $K$ *layer* propagations in both GNNs. At the end of each round $t$, we refine the earlier alignment $\boldsymbol{P}_{t-1}$ to the next estimate $\boldsymbol{P}_t$, which will be used in the next round. Henceforth, we will use the double subscript $t, k$ instead of the single subscript $k$ as in traditional GNNs. We denote the node embeddings at layer $k$ and round $t$ by $\boldsymbol{h}^{(q)}_{t,k}(u), \boldsymbol{h}^{(c)}_{t,k}(u') \in \mathbb{R}^{\dim_h}$ for $u \in V_q$ and $u' \in V_c$, which are (re-)initialized with node features $\boldsymbol{h}^{\bullet}_{t,0}$ for each round $t$. We gather these into matrices

$$\boldsymbol{H}^{(q)}_{t,k} = [\boldsymbol{h}^{(q)}_{t,k}(u) \mid u \in V_q] \in \mathbb{R}^{n\times \dim_h} \quad \text{and} \quad \boldsymbol{H}^{(c)}_{t,k} = [\boldsymbol{h}^{(c)}_{t,k}(u') \mid u' \in V_c] \in \mathbb{R}^{n\times \dim_h}. \quad (8)$$

$\boldsymbol{P}$ no longer remains an oracular hard permutation matrix, but becomes a doubly stochastic matrix indexed by rounds, written as $\boldsymbol{P}_t$. At the end of round $t$, a differentiable *aligner* module takes $\boldsymbol{H}^{(q)}_{t,K}$ and $\boldsymbol{H}^{(c)}_{t,K}$ as inputs and outputs a doubly stochastic node alignment (relaxed permutation) matrix $\boldsymbol{P}_t$ as follows:

$$\boldsymbol{P}_t = \mathrm{NodeAlignerRefinement}_\phi\left(\boldsymbol{H}^{(q)}_{t,K}, \boldsymbol{H}^{(c)}_{t,K}\right) \quad (9)$$

$$= \mathrm{GumbelSinkhorn}\left(\mathrm{LRL}_\phi(\boldsymbol{H}^{(q)}_{t,K})\,\mathrm{LRL}_\phi(\boldsymbol{H}^{(c)}_{t,K})^\top\right) \in \mathcal{B}_n \quad (10)$$

In the above expression, $\mathrm{GumbelSinkhorn}(\bullet)$ performs iterative Sinkhorn normalization on the input matrix added with Gumbel noise [26]; $\mathrm{LRL}_\phi$ is a neural module consisting of two linear layers with a ReLU activation after the first layer. As we shall see next, $\boldsymbol{P}_t$ is used to gate messages flowing *across* from one graph to the other during round $t+1$, i.e., while computing $\boldsymbol{H}^{(q)}_{t+1,1:K}$ and $\boldsymbol{H}^{(c)}_{t+1,1:K}$. The soft alignment $\boldsymbol{P}_t$ is kept frozen for the duration of all layers in round $t+1$. $\boldsymbol{P}_t[u, u']$ may be interpreted as the probability that $u$ is assigned to $u'$, which naturally requires that $\boldsymbol{P}_t$ should be row-equivariant (column equivariant) to the shuffling of the node indices of $G_q$ ($G_c$). As shown in Appendix D, the above design choice (10) ensures this property.

**Updating node representation using early-interaction GNN** Here, we describe the early interaction GNN for the query graph $G_q$. The GNN on the corpus graph $G_c$ follows the exact same design and is deferred to Appendix E.1. In the initial round ($t = 1$), since there is no prior alignment estimate $\boldsymbol{P}_{t=0}$, we employ the traditional late interaction GNN (1) to compute all layers $\boldsymbol{H}^{(q)}_{1,1:K}$ and $\boldsymbol{H}^{(c)}_{1,1:K}$ separately. These embeddings are then used to estimate $\boldsymbol{P}_{t=1}$ using Eq. (10). For subsequent rounds ($t > 1$), given embeddings $\boldsymbol{H}^{(q)}_{t,1:K}$, and the alignment estimate matrix $\boldsymbol{P}_t$, we run an early interaction GNN from scratch. We start with a fresh initialization of the node embeddings as before; i.e., $\boldsymbol{h}^{(q)}_{t+1,0}(u) = \mathrm{Init}_\theta(\mathrm{feature}(u))$. For each subsequent propagation layer $k+1$ ($k \in [0, K-1]$), we approximate (7) as follows. We read previous-round, same-layer embeddings $\boldsymbol{h}^{(c)}_{t,k}(u')$ of nodes $u'$ from the other graph $G_c$, incorporate the alignment strength $\boldsymbol{P}_t[u, u']$, and aggregate these to get an intermediate representation of $u$ that is sensitive to $\boldsymbol{P}_t$ and $G_c$.

$$\boldsymbol{z}^{(q)}_{t+1,k}(u) = \mathrm{inter}_\theta\left(\boldsymbol{h}^{(q)}_{t+1,k}(u), \textstyle\sum_{u'\in V_c} \boldsymbol{h}^{(c)}_{t,k}(u')\boldsymbol{P}_t[u, u']\right) \quad (11)$$

Here, $\mathrm{inter}_\theta$ is a neural network that computes interaction between the graph pairs; $\boldsymbol{z}^{(q)}_{t+1,k}(u)$ provides a soft alignment guided representation of $[\boldsymbol{h}^{(q)}_k(u), \boldsymbol{h}^{(c)}_k(\pi(u))]$ in Eq. (7), which can now be relaxed to the form

$$\boldsymbol{h}^{(q)}_{t+1,k+1}(u) = \mathrm{comb}_\theta\left(\boldsymbol{z}^{(q)}_{t+1,k}(u), \textstyle\sum_{v\in\mathrm{nbr}(u)} \mathrm{msg}_\theta(\boldsymbol{z}^{(q)}_{t+1,k}(u), \boldsymbol{z}^{(q)}_{t+1,k}(v))\right) \quad (12)$$

In the above expression, we explicitly feed $\boldsymbol{z}^{(q)}_{t+1,k}(v), v \in \text{nbr}(u)$ in the $\text{msg}_\theta$ network, capturing embeddings of nodes in the corpus $G_c$ aligned with the *neighbors* of node $u \in V_q$ in $\boldsymbol{h}^{(q)}_{t+1,k+1}(u)$. This allows the model to perform node-pair partner interaction. Instead, if we were to feed only $\boldsymbol{h}^{(q)}_{t+1,k}(u)$ into the $\text{msg}_\theta$ network, then it would only perform node partner interaction. In this case, the computed embedding for $u$ would be based solely on signals from nodes in the paired graph that directly correspond to $u$, therefore missing additional context from other neighbourhood nodes.

**Distant supervision of alignment** Finally, at the end of $T$ rounds, we express the relevance distance $\Delta(G_c \,|\, G_q)$ as a soft distance between the set $\boldsymbol{H}^{(q)}_{T,K} = [\boldsymbol{h}^{(q)}_{T,K}(u) \,|\, u \in V_q]$ and $\boldsymbol{H}^{(c)}_{T,K} = [\boldsymbol{h}^{(c)}_{T,K}(u') \,|\, u' \in V_c]$, measured as

$$\Delta_{\theta,\phi}(G_c \,|\, G_q) = \textstyle\sum_u \sum_d \text{ReLU}(\boldsymbol{H}^{(q)}_{T,K}[u,d] - (\boldsymbol{P}_T \boldsymbol{H}^{(c)}_{T,K})[u,d]) \tag{13}$$

Our focus is on graph retrieval applications. It is unrealistic to assume direct supervision from a gold alignment map $\boldsymbol{P}^*$. Instead, training query instances are associated with pairwise preferences between two corpus graphs, in the form $\langle G_q, G_{c+}, G_{c-} \rangle$, meaning that, ideally, we want $\Delta_{\theta,\phi}(G_{c-}|G_q) \geq \gamma + \Delta_{\theta,\phi}(G_{c+}|G_q)$, where $\gamma > 0$ is a margin hyperparameter. This suggests a minimization of the standard hinge loss as follows:

$$\min_{\theta,\phi} \textstyle\sum_{q \in Q} \sum_{c+ \in C_{q+}, c- \in C_{q-}} [\gamma + \Delta_{\theta,\phi}(G_{c+} \,|\, G_q) - \Delta_{\theta,\phi}(G_{c-} \,|\, G_q)]_+ \tag{14}$$

This loss is back-propagated to train model weights $\theta$ in $\text{comb}_\theta, \text{inter}_\theta, \text{msg}_\theta$ and weights $\phi$ in the Gumbel-Sinkhorn network.

**Multi-layer eager alignment variant** Having set up the general multi-round framework of EINSMATCH, we introduce a structurally simpler variant that updates $\boldsymbol{P}$ eagerly after every layer, eliminating the need to re-initialize node embeddings every time we update $\boldsymbol{P}$. The eager variant retains the benefits of node-pair partner interactions, while ablating EINSMATCH toward GMN. Updating $\boldsymbol{P}$ via Sinkhorn iterations is expensive compared to a single GNN layer. In practice, we see a non-trivial tradeoff between computation cost, end task accuracy, and the quality of our injective alignments, depending on the value of $K$ for eager updates, and the values $(T, K)$ for lazy updates.

### 3.3 Extension of EINSMATCH (Node) to EINSMATCH (Edge)

We now extend EINSMATCH (Node) to EINSMATCH (Edge) which uses explicit edge alignment for interaction across GNN and relevance distance surrogate, starting with the multi-round refinement protocol for edge alignment.

**Multi-round refinement of edge alignment** In EINSMATCH (Edge), we maintain a soft edge permutation matrix $\boldsymbol{S}$ which is frozen at $\boldsymbol{S} = \boldsymbol{S}_{t-1}$ within each round $t \in [T]$ and gets refined after every round $t$ as $\boldsymbol{S}_{t-1} \to \boldsymbol{S}_t$. Similar to EINSMATCH (Node), within each round $t$, GNN runs from scratch: it propagates messages across layers $k \in [K]$ and $\boldsymbol{S}_{t-1}$ assists it to capture cross-graph signals. Here, in addition to node embeddings $\boldsymbol{h}^{(\bullet)}_{t,k}$, we also use edge embeddings $\boldsymbol{m}^{(q)}_{t,k}(e),\ \boldsymbol{m}^{(c)}_{t,k}(e') \in \mathbb{R}^{\dim_m}$ at each layer $k$ and each round $t$, which capture the information about the subgraph $k \leq K$ hop away from the edges $e$ and $e'$. Similar to Eqn. (8), we define $\boldsymbol{M}^{(q)}_{t,k} = [\boldsymbol{m}^{(q)}_{t,k}(e)]_{e \in E_q}$, and $\boldsymbol{M}^{(c)}_{t,k} = [\boldsymbol{m}^{(c)}_{t,k}(e')]_{e' \in E_c}$. $\boldsymbol{M}^{(\bullet)}_{t,0}$ are initialized using the features of the nodes connected by the edges, and possibly local edge features. Given the embeddings $\boldsymbol{M}^{(q)}_{t,K}$ and $\boldsymbol{M}^{(c)}_{t,K}$ computed at the end of round $t$, an edge aligner module takes these embedding matrices as input and outputs a soft edge permutation matrix $\boldsymbol{S}_t$, similar to the update of $\boldsymbol{P}_t$ in Eq. (10).

$$\boldsymbol{S}_t = \text{EdgeAlignerRefinement}_\phi \left( \boldsymbol{M}^{(q)}_{t,K}, \boldsymbol{M}^{(c)}_{t,K} \right) \tag{15}$$

$$= \text{GumbelSinkhorn}(\text{LRL}_\phi(\boldsymbol{M}^{(q)}_{t,K})\ \text{LRL}_\phi(\boldsymbol{M}^{(c)}_{t,K})^\top) \tag{16}$$

Here, $\boldsymbol{M}^{(\bullet)}_{t,K}$ are appropriately padded to ensure that they have the same number of rows.

**Edge alignment-induced early interaction GNN** For $t = 1$, we start with a late interaction model using vanilla GNN (1) and obtain $\boldsymbol{S}_{t=1}$ using Eq. (16). Having computed the edge embeddings $\boldsymbol{m}^{(\bullet)}_{t,1:K}(\bullet)$ and node embeddings $\boldsymbol{h}^{(\bullet)}_{t,1:K}(\bullet)$ upto round $t$, we compute $\boldsymbol{S}_t$ and use it to build a fresh early interaction GNN for round $t+1$. To this end, we adapt the GNN guided by $\boldsymbol{P}_t$ in Eqs. (11)–

(12),to the GNN guided by $\boldsymbol{S}_t$. We overload the notations for neural modules and different embedding vectors from EINSMATCH (Node), whenever their roles are similar.

Starting with the same initialization as in EINSMATCH (Node), we perform the cross-graph interaction guided by the soft edge permutation matrix $\boldsymbol{S}_t$, similar to Eq. (11). Specifically, we use the embeddings of edges $\{e' = (u', v')\} \in E_c$, computed at layer $k$ at round $t$, which share soft alignments with an edge $e = (u, v) \in E_q$, to compute $\boldsymbol{z}^{(q)}_{t+1,k}(e)$ and $\boldsymbol{z}^{(q)}_{t+1,k}(e')$ as follows:

$$\boldsymbol{z}^{(q)}_{t+1,k}(e) = \text{inter}_\theta \left( \boldsymbol{m}^{(q)}_{t+1,k}(e), \textstyle\sum_{e' \in E_c} \boldsymbol{m}^{(c)}_{t,k}(e') \boldsymbol{S}_t[e, e'] \right) \tag{17}$$

Finally, we update the node embeddings $\boldsymbol{h}^{(\bullet)}_{t+1,k+1}$ for propagation layer $k+1$ as

$$\boldsymbol{h}^{(q)}_{t+1,k+1}(u) = \text{comb}_\theta \left( \boldsymbol{h}^{(q)}_{t+1,k}(u), \textstyle\sum_{a \in \text{nbr}(u)} \text{msg}_\theta(\boldsymbol{h}^{(q)}_{t+1,k}(u), \boldsymbol{h}^{(q)}_{t+1,k}(a), \boldsymbol{z}^{(q)}_{t+1,k}((u, a))) \right) \tag{18}$$

In this case, we perform the cross-graph interaction at the edge level rather than the node level. Hence, $\text{msg}_\theta$ acquires cross-edge signals separately as $\boldsymbol{z}^{(\bullet)}_{t+1,k}$. Finally, we use $\boldsymbol{h}^{(\bullet)}_{t+1,k+1}$ and $\boldsymbol{z}^{(\bullet)}_{t+1,k+1}$ to update $\boldsymbol{m}^{(\bullet)}_{t+1,k+1}$ as follows:

$$\boldsymbol{m}^{(q)}_{t+1,k+1}\big((u, v)\big) = \text{msg}_\theta \left( \boldsymbol{h}^{(q)}_{t+1,k+1}(u), \boldsymbol{h}^{(q)}_{t+1,k+1}(v), \boldsymbol{z}^{(q)}_{t+1,k}((u, v)) \right) \tag{19}$$

Likewise, we develop $\boldsymbol{m}^{(c)}_{t+1,k+1}$ for corpus graph $G_c$. Note that $\boldsymbol{m}^{(q)}_{t+1,k+1}((u, v))$ captures signals not only from the matched pair $(u', v')$, but also signals from the nodes in $G_c$ which share correspondences with the neighbor nodes of $u$ and $v$. Finally, we pad zero vectors to $[\boldsymbol{m}^{(q)}_{T,K}(e)]_{e \in E_q}$ and $[\boldsymbol{m}^{(c)}_{T,K}(e')]_{e' \in E_c}$ to build the matrices $\boldsymbol{M}^{(q)}_{T,K}$ and $\boldsymbol{M}^{(c)}_{T,K}$ with same number of rows, which are finally used to compute the relevance distance

$$\Delta_{\theta,\phi}(G_c \,|\, G_q) = \textstyle\sum_u \sum_d \text{ReLU}(\boldsymbol{M}^{(q)}_{T,K}[e, d] - (\boldsymbol{S}_T \boldsymbol{M}^{(c)}_{T,K})[e, d]). \tag{20}$$

# 4 Experiments

We report on a comprehensive evaluation of EINSMATCH on six real datasets and analyze the efficacy of the key novel design choices. In Appendix G, we provide results of additional experiments.

## 4.1 Experimental setup

**Datasets** We use six real world datasets in our experiments, *viz.*, AIDS, Mutag, PTC-FM (FM), PTC-FR (FR), PTC-MM (MM) and PTC-MR (MR), which were also used in [27, 35]. Appendix F provides the details about dataset generation and their statistics.

**State-of-the-art baselines** We compare our method against eleven state-of-the-art methods, *viz.*, (1) GraphSim [2] (2) GOTSim [11], (3) SimGNN [1], (4) EGSC [31], (5) H2MN [45], (6) Neuromatch [23], (7) GREED [32], (8) GEN [22], (9) GMN [22] (10) IsoNet (Node) [35], and (11) IsoNet (Edge) [35]. Among them, Neuromatch, GREED, IsoNet (Node) and IsoNet (Edge) apply asymmetric hinge distances between query and corpus embeddings for $\Delta(G_c \,|\, G_q)$, specifically catered towards subgraph matching, similar to our method in Eqs. (13) and (20). GMN and GEN use symmetric Euclidean distance between their (whole-) graph embeddings $\boldsymbol{g}^{(q)}$ (for query) and $\boldsymbol{g}^{(c)}$ (for corpus) as $||\boldsymbol{g}^{(q)} - \boldsymbol{g}^{(c)}||$ in their paper [22], which is not suitable for subgraph matching and therefore, results in poor performance. Hence, we change it to $\Delta(G_c \,|\, G_q) = [\boldsymbol{g}^{(q)} - \boldsymbol{g}^{(c)}]_+$. The other methods first compute the graph embeddings, then fuse them using a neural network and finally apply a nonlinear function on the fused embeddings to obtain the relevance score.

**Training and evaluation protocol** Given a fixed corpus set $C$, we split the query set $Q$ into $60\%$ training, $15\%$ validation and $25\%$ test set. We train all the models on the training set by minimizing a ranking loss (14). During the training of each model, we use five random seeds. Given a test query $q'$, we rank the corpus graphs $C$ in the decreasing order of $\Delta_{\theta,\phi}(G_c \,|\, G_{q'})$ computed using the trained model. We evaluate the quality of the ranking by measuring Average Precision (AP) and HITS@20, described in Appendix F. Finally, we report mean average precision (MAP) and mean HITS@20, across all the test queries. By default, we set the number of rounds $T = 3$, the number of propagation layers in GNN $K = 5$. In Appendix F, we discuss the baselines, hyperparameter setup and the evaluation metrics in more detail.

| Metrics → | Mean Average Precision (MAP) | | | | | | HITS @ 20 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AIDS | Mutag | FM | FR | MM | MR | AIDS | Mutag | FM | FR | MM | MR |
| GraphSim [2] | 0.356 | 0.472 | 0.477 | 0.423 | 0.415 | 0.453 | 0.145 | 0.257 | 0.261 | 0.227 | 0.212 | 0.23 |
| GOTSim [11] | 0.324 | 0.272 | 0.355 | 0.373 | 0.323 | 0.317 | 0.112 | 0.088 | 0.147 | 0.166 | 0.119 | 0.116 |
| SimGNN [1] | 0.341 | 0.283 | 0.473 | 0.341 | 0.298 | 0.379 | 0.138 | 0.087 | 0.235 | 0.155 | 0.111 | 0.160 |
| EGSC [31] | 0.505 | 0.476 | 0.609 | 0.607 | 0.586 | 0.58 | 0.267 | 0.243 | 0.364 | 0.382 | 0.348 | 0.325 |
| H2MN [45] | 0.267 | 0.276 | 0.436 | 0.412 | 0.312 | 0.243 | 0.076 | 0.084 | 0.200 | 0.189 | 0.119 | 0.069 |
| Neuromatch [23] | 0.489 | 0.576 | 0.615 | 0.559 | 0.519 | 0.606 | 0.262 | 0.376 | 0.389 | 0.350 | 0.282 | 0.385 |
| GREED [32] | 0.472 | 0.567 | 0.558 | 0.512 | 0.546 | 0.528 | 0.245 | 0.371 | 0.316 | 0.287 | 0.311 | 0.277 |
| GEN [22] | 0.557 | 0.605 | 0.661 | 0.575 | 0.539 | 0.631 | 0.321 | 0.429 | 0.448 | 0.368 | 0.292 | 0.391 |
| GMN [22] | 0.622 | 0.710 | 0.730 | 0.662 | 0.655 | 0.708 | 0.397 | 0.544 | 0.537 | 0.45 | 0.423 | 0.49 |
| IsoNet (Node) [35] | 0.659 | 0.697 | 0.729 | 0.68 | 0.708 | 0.738 | 0.438 | 0.509 | 0.525 | 0.475 | 0.493 | 0.532 |
| IsoNet (Edge) [35] | 0.690 | 0.706 | 0.783 | 0.722 | 0.753 | 0.774 | 0.479 | 0.529 | 0.613 | 0.538 | 0.571 | 0.601 |
| EINSM. (Node) | 0.825 | 0.851 | 0.888 | 0.855 | 0.838 | 0.874 | 0.672 | 0.732 | 0.797 | 0.737 | 0.702 | 0.755 |
| EINSM. (Edge) | 0.847 | 0.858 | 0.902 | 0.875 | 0.902 | 0.902 | 0.705 | 0.749 | 0.813 | 0.769 | 0.809 | 0.803 |

Table 2: Comparison of the two variants of EINSMATCH (EINSMATCH (Node) and EINSMATCH (Edge)) against all the state-of-the-art graph retrieval methods, across all six datasets. Performance is measured in terms average precision (MAP) and mean HITS@20. In all cases, we used 60% training, 15% validation and 25% test sets. The numbers highlighted with green and yellow indicate the best, second best method respectively, whereas the numbers with blue indicate the best method among the baselines. (MAP values for EINSMATCH (Edge) across FM, MM and MR were verified to be not exactly the same, but they match up to the third decimal place.)

| | | AIDS | Mutag | FM | FR | MM | MR |
|---|---|---|---|---|---|---|---|
| Node | Eager | 0.756 | 0.81 | 0.859 | 0.802 | 0.827 | 0.841 |
| | Lazy | 0.825 | 0.851 | 0.888 | 0.855 | 0.838 | 0.874 |
| Edge | Eager | 0.795 | 0.805 | 0.883 | 0.812 | 0.862 | 0.886 |
| | Lazy | 0.847 | 0.858 | 0.902 | 0.875 | 0.902 | 0.902 |

Table 3: Lazy multi-round vs. eager multi-layer. First (Last) two rows report MAP for EINSMATCH (Node) (EINSMATCH (Edge)). Green shows the best method

| | | AIDS | Mutag | FM | FR | MM | MR |
|---|---|---|---|---|---|---|---|
| Lazy | Node partner | 0.776 | 0.829 | 0.851 | 0.819 | 0.844 | 0.84 |
| | EINSM. (Node) | 0.825 | 0.851 | 0.888 | 0.855 | 0.838 | 0.874 |
| Eager | Node partner | 0.668 | 0.783 | 0.821 | 0.752 | 0.753 | 0.794 |
| | EINSM. (Node) | 0.756 | 0.81 | 0.859 | 0.802 | 0.827 | 0.841 |

Table 4: Node partner vs. node pair partner interaction. First (Last) two rows report MAP for multi-round (multi-layer) update. Green shows the best method.

## 4.2 Results

**Comparison with baselines** First, we compare EINSMATCH (Node) and EINSMATCH (Edge) against all the baselines, across all datasets. In Table 2, we report the results. The key observations are as follows: **(1)** EINSMATCH (Node) and EINSMATCH (Edge) outperform all the baselines by significant margins across all datasets. EINSMATCH (Edge) consistently outperforms EINSMATCH (Node). This is because edge alignment allows us to compare the graph pairs more effectively than node alignment. A similar effect was seen for IsoNet (Edge) vs. IsoNet (Node) [35]. **(2)** Among all state-of-the-art competitors, IsoNet (Edge) performs the best followed by IsoNet (Node). Similar to us, they also use edge and node alignments respectively. However, IsoNet does not perform any interaction between the graph pairs and the alignment is computed once only during the computation of $\Delta(G_c \,|\, G_q)$. This results in modest performance compared to EINSMATCH. **(3)** GMN uses "attention" to estimate the alignment between graph pairs, which induces a non-injective mapping. Therefore, despite being an early interaction model, it is mostly outperformed by IsoNet, which uses injective alignments.

**Lazy vs. eager updates** In lazy multi-round updates, the alignment matrices remain unchanged across all propagation layers and are updated only after the GNN completes its $K$-layer message propagations. To evaluate its effectiveness, we compare it against the eager multi-*layer* update (described at the end of Section 3.2), where the GNN executes its $K$-layer message propagations only once; the alignment map is updated across $K$ layers; and, the alignment at $k$th layer is used to compute the embeddings at $(k+1)$th layer. In Table 3, we compare the performance in terms MAP, which shows that lazy multi-round updates significantly outperform multi-layer updates.

**Node partner vs. node-pair partner interaction** To understand the benefits of node-pair partner interaction, we contrast EINSMATCH (Node) against another variant of our method, which performs *node partner* interaction rather than node pair partner interaction, similar to Eq. (6). For lazy multi-round updates, we compute the embeddings as follows:

$$\boldsymbol{h}^{(q)}_{t+1,k+1}(u) = \mathrm{comb}_\theta(\boldsymbol{h}^{(q)}_{t+1,k}(u),\ \textstyle\sum_{v\in \mathrm{nbr}(u)} \mathrm{msg}_\theta(\boldsymbol{h}^{(q)}_{t,k}(u), \boldsymbol{h}^{(q)}_{t,k}(v)),\ \sum_{u'\in V_c} \boldsymbol{P}_t[u,u']\boldsymbol{h}^{(c)}_{t,k}(u'))$$

For eager multi-layer updates, we compute the embeddings as:

$$\boldsymbol{h}^{(q)}_{k+1}(u) = \mathrm{comb}_\theta(\boldsymbol{h}^{(q)}_{k}(u),\ \textstyle\sum_{v\in \mathrm{nbr}(u)} \mathrm{msg}_\theta(\boldsymbol{h}^{(q)}_{k}(u), \boldsymbol{h}^{(q)}_{k}(v)),\ \sum_{u'\in V_c} \boldsymbol{P}_k[u,u']\boldsymbol{h}^{(c)}_{k}(u'))$$

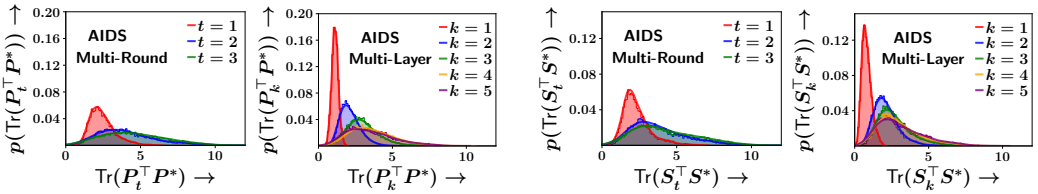
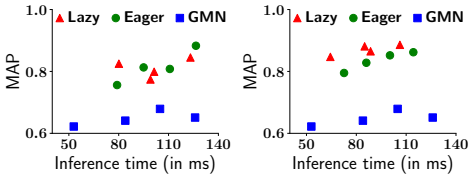



Figure 5: Empirical probability density of similarity between the estimated alignments and the true alignments $\boldsymbol{P}^*, \boldsymbol{S}^*$ for both multi-round and multi-layer update strategies across different stages of updates ($t$ for multi-round and $k$ for multi-layer), for AIDS. Similarity is measured using $p(\text{Tr}(\boldsymbol{P}_t^\top \boldsymbol{P}^*)), p(\text{Tr}(\boldsymbol{S}_t^\top \boldsymbol{S}^*))$ for multi-round lazy updates and $p(\text{Tr}(\boldsymbol{P}_k^\top \boldsymbol{P}^*)), p(\text{Tr}(\boldsymbol{S}_k^\top \boldsymbol{S}^*))$ for multi-layer eager updates.

Table 4 summarizes the results, which shows that EINSMATCH (Node) (node partner pair) performs significantly better than Node partner for both multi-round lazy updates (top-two rows) and multi-layer eager updates (bottom tow rows).

**Quality of injective alignments** Next we compare between multi-round and multi-layer update strategies in terms of their ability to refine the alignment matrices, as the number of updates of these matrices increases. For multi-round (layer) updates, we instrument the alignments $\boldsymbol{P}_t$ and $\boldsymbol{S}_t$ ($\boldsymbol{P}_k$ and $\boldsymbol{S}_k$) for different rounds $t \in [T]$ (layers $k \in [K]$). Specifically, we look into the distribution of the similarity between the learned alignments $\boldsymbol{P}_t, \boldsymbol{S}_t$ and the correct alignments $\boldsymbol{P}^*, \boldsymbol{S}^*$ (using combinatorial routine), measured using the inner products $\text{Tr}(\boldsymbol{P}_t^\top \boldsymbol{P}^*)$ and $\text{Tr}(\boldsymbol{S}_t^\top \boldsymbol{S}^*)$ for different $t$. Similarly, we compute $\text{Tr}(\boldsymbol{P}_k^\top \boldsymbol{P}^*)$ and $\text{Tr}(\boldsymbol{S}_k^\top \boldsymbol{S}^*)$ for different $k \in [K]$. Figure 5 summarizes the results, which shows that **(1)** as $t$ or $k$ increases, the learned alignments become closer to the gold alignments; **(2)** multi-round updates refine the alignments approximately twice as faster than the multi-layer variant. The distribution of $\text{Tr}(\boldsymbol{P}_t^\top \boldsymbol{P}^*)$ at $t = 1$ in multi-round strategy is almost always close to $\text{Tr}(\boldsymbol{P}_k^\top \boldsymbol{P}^*)$ for $k = 2$. Note that, our aligner networks learn to refine the $\boldsymbol{P}_t$ and $\boldsymbol{S}_t$ through end-to-end training, without using any form of supervision from true alignments or the gradient computed in Eq. (5).

**Accuracy-inference time trade-off** Here, we analyze the accuracy and inference time trade-off. We vary $T$ and $K$ for EINSMATCH's lazy multi-round variant, and vary $K$ for EINSMATCH's eager multi-layer variant and for GMN. Figure 6 summarizes the results. Notably, the eager multi-layer variant achieves the highest accuracy for $K = 8$ on the AIDS dataset, despite the known issue of oversmoothing in GNNs for large $K$. This unexpected result may be due to our message passing components, which involve terms like $\sum_{u'} \boldsymbol{P}[u, u']\boldsymbol{h}(u')$, effectively acting as a convolution between alignment scores and embedding vectors. This likely enables $\boldsymbol{P}$ to function as a filter, countering the oversmoothing effect.




(a) Node, AIDS (b) Edge, AIDS

Figure 6: Trade-off between MAP and inference time (batch size=128).

## 5 Conclusion

We introduce EINSMATCH as an early-interaction network for estimating subgraph isomorphism. EINSMATCH learns to identify explicit alignments between query and corpus graphs despite having access to only pairwise preferences and not explicit alignments during training. We design a GNN that uses an alignment estimate to propagate messages, then uses the GNN's output representations to refine the alignment. Experiments across several datasets confirm that alignment refinement is achieved over several rounds. Design choices such as using node-pair partner interaction (instead of node partner) and lazy updates (over eager) boost the performance of our architecture, making it the state-of-the-art in subgraph isomorphism based subgraph retrieval. We also demonstrate the accuracy v/s inference time trade offs for EINSMATCH, which show how different knobs can be tuned to utilize our models under regimes with varied time constraints.

This study can be extended to graph retrieval problems which use different graph similarity measures, such as maximum common subgraph or graph edit distance. Extracting information from node-pairs is an exciting idea and can be widely used to improve graph neural networks working on multiple graphs at once.

# References


[1] Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 384–392, 2019.

[2] 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*, volume 34, pages 3219–3226, 2020.

[3] J.-D. Benamou, G. Carlier, M. Cuturi, L. Nenna, and G. Peyré. Iterative bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 37(2):A1111–A1138, 2015.

[4] Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J.-P. Vert, and F. Bach. Learning with differentiable pertubed optimizers. *Advances in neural information processing systems*, 33:9508–9519, 2020.

[5] L. Biewald. Experiment tracking with weights and biases, 2020. URL `https://www.wandb.com/`. Software available from wandb.com.

[6] A. Cereto-Massagué, M. J. Ojeda, C. Valls, M. Mulero, S. Garcia-Vallvé, and G. Pujadas. Molecular fingerprint similarity search in virtual screening. *Methods*, 71:58–63, 2015.

[7] D. Chen, L. O'Bray, and K. Borgwardt. Structure-aware transformer for graph representation learning. *ICML*, 2022.

[8] E. Cohen-Karlik, A. B. David, and A. Globerson. Regularizing towards permutation invariance in recurrent models. In *NeurIPS*. Curran Associates Inc., 2020. ISBN 9781713829546. URL `https://arxiv.org/abs/2010.13055`.

[9] 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*, 26(10): 1367–1372, 2004.

[10] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300, 2013.

[11] K. D. Doan, S. Manchanda, S. Mahapatra, and C. K. Reddy. Interpretable graph similarity computation via differentiable optimal alignment of node embeddings. pages 665–674, 2021.

[12] H.-C. Ehrlich and M. Rarey. Systematic benchmark of substructure search in molecular graphs-from ullmann to vf2. *Journal of Cheminformatics*, 4:1–17, 2012.

[13] X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13(1):113–129, 2010.

[14] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[15] A. Hagberg, P. Swart, and D. S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[16] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3668–3678, 2015.

[17] N. Karalias and A. Loukas. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. *Advances in Neural Information Processing Systems*, 33: 6659–6672, 2020.

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

[19] J. Kotary, F. Fioretto, P. Van Hentenryck, and B. Wilder. End-to-end constrained optimization learning: A survey. *arXiv preprint arXiv:2103.16378*, 2021.

[20] A. Lai and J. Hockenmaier. Learning to predict denotational probabilities for modeling entailment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 721–730, 2017. URL https://www.aclweb.org/anthology/E17-1068.pdf.

[21] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[22] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli. Graph matching networks for learning the similarity of graph structured objects. In *International conference on machine learning*, pages 3835–3845. PMLR, 2019. URL https://arxiv.org/abs/1904.12787.

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

[24] D. Marcheggiani and I. Titov. Encoding sentences with graph convolutional networks for semantic role labeling. In M. Palmer, R. Hwa, and S. Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1159. URL https://aclanthology.org/D17-1159.

[25] B. McFee and G. R. G. Lanckriet. Partial order embedding with multiple kernels. In *International Conference on Machine Learning*, 2009. URL https://api.semanticscholar.org/CorpusID:699292.

[26] G. Mena, D. Belanger, S. Linderman, and J. Snoek. Learning latent permutations with gumbel-sinkhorn networks. *arXiv preprint arXiv:1802.08665*, 2018. URL https://arxiv.org/pdf/1802.08665.pdf.

[27] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL www.graphlearning.io.

[28] R. Myers, R. Wison, and E. R. Hancock. Bayesian graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):628–635, 2000.

[29] M. Ohlrich, C. Ebeling, E. Ginting, and L. Sather. Subgemini: Identifying subcircuits using a fast subgraph isomorphism algorithm. In *Proceedings of the 30th International Design Automation Conference*, pages 31–37, 1993.

[30] G. Peyré, M. Cuturi, and J. Solomon. Gromov-wasserstein averaging of kernel and distance matrices. In *International conference on machine learning*, pages 2664–2672. PMLR, 2016.

[31] C. Qin, H. Zhao, L. Wang, H. Wang, Y. Zhang, and Y. Fu. Slow learning and fast inference: Efficient graph similarity computation via knowledge distillation. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[32] R. Ranjan, S. Grover, S. Medya, V. Chakaravarthy, Y. Sabharwal, and S. Ranu. Greed: A neural framework for learning graph distance functions. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, November 29-Decemer 1, 2022*, 2022.

[33] I. Roy, A. De, and S. Chakrabarti. Adversarial permutation guided node representations for link prediction. In *AAAI Conference*, 2021. URL https://arxiv.org/abs/2012.08974.

[34] I. Roy, S. Chakrabarti, and A. De. Maximum common subgraph guided graph retrieval: Late and early interaction networks. In *NeurIPS*, 2022. URL https://openreview.net/forum?id=COAcbu3_k4U.

[35] I. Roy, V. S. Velugoti, S. Chakrabarti, and A. De. Interpretable neural subgraph matching for graph retrieval. In *AAAI Conference*, 2022. URL https://indradyumna.github.io/pdfs/IsoNet_main.pdf.

[36] T. K. Rusch, M. M. Bronstein, and S. Mishra. A survey on oversmoothing in graph neural networks. Preprint, 2023. URL https://arxiv.org/abs/2303.10993.

[37] R. Sinkhorn and P. Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

[38] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[39] I. Vendrov, R. Kiros, S. Fidler, and R. Urtasun. Order-embeddings of images and language. *arXiv preprint arXiv:1511.06361*, 2015. URL https://arxiv.org/pdf/1511.06361.

[40] F. Wenkel, Y. Min, M. Hirn, M. Perlmutter, and G. Wolf. Overcoming oversmoothness in graph convolutional networks via hybrid scattering networks, 2022. URL https://arxiv.org/abs/2201.08932.

[41] H. Xu, D. Luo, H. Zha, and L. C. Duke. Gromov-wasserstein learning for graph matching and node embedding. In *International conference on machine learning*, pages 6932–6941. PMLR, 2019.

[42] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[43] X. Yan, P. S. Yu, and J. Han. Graph indexing: A frequent structure-based approach. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 335–346, 2004.

[44] Z. Zeng, A. K. Tung, J. Wang, J. Feng, and L. Zhou. Comparing stars: On approximating graph edit distance. *Proceedings of the VLDB Endowment*, 2(1):25–36, 2009.

[45] Z. Zhang, J. Bu, M. Ester, Z. Li, C. Yao, Z. Yu, and C. Wang. H2mn: Graph similarity learning with hierarchical hypergraph matching networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2274–2284, 2021.

[46] W. Zhuo and G. Tan. Efficient graph similarity computation with alignment regularization. *Advances in Neural Information Processing Systems*, 35:30181–30193, 2022.

# Iteratively Refined Early Interaction Alignment for Subgraph Matching and Retrieval (Appendix)

## A Limitations

We find two limitations of our method each of which could form the basis of detailed future studies.

1. Retrieval systems greatly benefit from the similarity function being hashable. This can improve the inference time multi-fold while losing very little, if at all any, performance, making the approach ready for production environments working under tight time constraints. The design of a hash function for an early interaction network like ours is unknown and seemingly difficult. In fact, such a hashing procedure is not known even for predecessors like IsoNet (Edge) or GMN, and this is an exciting future direction.
2. Our approach does not explicitly differentiate between nodes or edges that may belong to different classes. This can be counterproductive when there exist constraints that prevent the alignment of two nodes or edges with different labels. While the network is designed to process node and edge features, it might not be enough to rule out alignments that violate the said constraint. Such constraints could also exist for node-pairs, such as in knowledge graphs with hierarchical relationships between entity types, and are not taken into account by our model. Extending our work to handle such restrictions is an interesting problem to consider.

## B Related work

In this section, we discuss different streams of work that are related to and have influenced the study.

### B.1 Graph Representation Learning

Graph neural networks (GNN) [14, 22, 21, 18, 42, 38] have emerged as a widely applicable approach for graph representation learning. A graph neural network computes the embedding of a node by aggregating the representations of its neighbors across $K$ steps of message passing, effectively combining information from $K$-hop neighbors. GNNs were first used for graph similarity computation by Li et al. [22], who enriched the architecture with attention to predict isomorphism between two graphs. Attention acts as a mechanism to transfer information from the representation of one graph to that of the other, thus boosting the performance of the approach. Chen et al. [7] enriched the representation of graphs by capturing the subgraph around a node effectively through a structure aware transformer architecture.

### B.2 Differentiable combinatorial solvers

We utilize a differentiable gadget to compute an injective alignment, which is a doubly stochastic matrix. The differentiability is crucial to the training procedure as it enables us to backpropagate through the alignments. The $\mathrm{GumbelSinkhorn}$ operator, which performs alternating normalizations across rows and columns, was first proposed by Sinkhorn and Knopp [37] and later used for the Optimal Transport problem by Cuturi [10]. Other methods to achieve differentiability include adding random noise to the inputs to discrete solvers [4] and designing probabilistic loss functions [17]. A compilation of such approaches towards constrained optimization on graphs through neural techniques is presented in [19].

### B.3 Graph Similarity Computation and Retrieval

Several different underlying measures have been proposed for graph similarity computation, including full graph isomorphism [22], subgraph isomorphism [23, 35], graph edit distance (GED) [2, 11, 13, 28, 44] and maximum common subgraph (MCS) [2, 11, 34]. Bai et al. [2] proposed GraphSim towards the GED and MCS problems, using convolutional neural network based scoring on top of graph similarity matrices. GOTSim [11] explicitly computes the alignment between the two graphs by studying the optimal transformation cost. GraphSim [2] utilizes both graph-level and node-level signals to compute a graph similarity score. NeuroMatch [23] evaluates, for each node pair across the two graphs, if the neighborhood of one node is contained in the neighborhood of another using order embeddings [25]. GREED [32] proposed a Siamese graph isomorphism network, a late interaction

model to tackle the GED problem and provided supporting theoretical guarantees. Zhang et al. [45] propose an early interaction model, using hypergraphs to learn higher order node similarity. Each hypergraph convolution contains a subgraph matching module to learn cross graph similarity. Qin et al. [31] trained a slower attention-based network on multi-level features from a GNN and distilled its knowledge into a faster student model. Roy et al. [35] used the $\mathrm{GumbelSinkhorn}$ operator as a differentiable gadget to compute alignments in a backpropagation-friendly fashion and also demonstrated the utility of computing alignments for edges instead of nodes.

# C Broader Impact

This work can be directly applied to numerous practical applications, such as drug discovery and circuit design, which are enormously beneficial for the society and continue to garner interest from researchers and practitioners worldwide. The ideas introduced in this paper have benefitted from and can benefit the information retrieval community as well, beyond the domain of graphs. However, malicious parties could use this technology for deceitful purposes, such as identifying and targeting specific social circles on online social networks (which can be represented as graphs). Such pros and cons are characteristic of every scientific study and the authors consider the positives to far outweigh the negatives.

# D Network architecture of different components of EINSMATCH

EINSMATCH models consist of three components - an encoder, a message-passing network and a node/edge aligner. We provide details about each of these components below. For convenience, we represent a linear layer with input dimension $a$ and output dimension $b$ as $\mathrm{Linear}(a, b)$ and a linear-ReLU-linear network with $\mathrm{Linear}(a, b),\ \mathrm{Linear}(b, c)$ layers with ReLU activation in the middle as $\mathrm{LRL}(a, b, c)$.

## D.1 Encoder

The encoder transforms input node/edge features before they are fed into the message-passing network. For models centred around node alignment like EINSMATCH (Node), the encoder refers to $\mathrm{Init}_\theta$ and is implemented as a $\mathrm{Linear}(1, 10)$ layer. The edge vectors are not encoded and passed as-is down to the message-passing network. For edge-based models like EINSMATCH (Edge), the encoder refers to both $\mathrm{Init}_{\theta,\mathrm{node}}$ and $\mathrm{Init}_{\theta,\mathrm{edge}}$, which are implemented as $\mathrm{Linear}(1, 10)$ and $\mathrm{Linear}(1, 20)$ layers respectively.

## D.2 GNN

Within the message-passing framework, we use node embeddings of size $\dim_h = 10$ and edge embeddings of size $\dim_m = 20$. We specify each component of the GNN below.

- $\mathrm{inter}_\theta$ combines the representation of the current node/edge ($\boldsymbol{h}_\bullet$) with that from the other graph, which are together fed to the network by concatenation. For node-based and edge-based models, it is implemented as $\mathrm{LRL}(20, 20, 10)$ and $\mathrm{LRL}(40, 40, 20)$ networks respectively. In particular, we ensure that the input dimension is twice the size of the output dimension, which in turn equals the intermediate embedding dimension $\dim(\boldsymbol{z})$.
- $\mathrm{msg}_\theta$ is used to compute messages by combining intermediate embeddings $\boldsymbol{z}_\bullet$ of nodes across an edge with the representation of that edge. For node-based models, the edge vector is a fixed vector of size 1 while the intermediate node embeddings $\boldsymbol{z}_\bullet$ are vectors of dimension 10, resulting in the network being a $\mathrm{Linear}(21, 20)$ layer. For edge-based models, the edge embedding is the $\boldsymbol{m}$ vector of size 20 which requires $\mathrm{msg}_\theta$ to be a $\mathrm{Linear}(40, 20)$ layer. Note that the message-passing network is applied twice, once to the ordered pair $(u, v)$ and then to $(v, u)$ and the outputs thus obtained are added up. This is to ensure node order invariance for undirected edges by design.
- $\mathrm{comb}_\theta$ combines the representation of a node $\boldsymbol{z}_\bullet$ with aggregated messages received by it from all its neighbors. It is modelled as a $\mathrm{GRU}$ where the node representation (of size 10) is the initial hidden state and the aggregated message vector (of size 20) is the only element of an input sequence which updates the hidden state to give us the final node embedding $\boldsymbol{h}_\bullet$.

### D.3 Node aligner

The node aligner takes as input two sets of node vectors $\boldsymbol{H}^{(q)} \in \mathbb{R}^{n\times 10}$ and $\boldsymbol{H}^{(c)} \in \mathbb{R}^{n\times 10}$ representing $G_q$ and $G_c$ respectively. $n$ refers to the number of nodes in the corpus graph (the query graph is padded to meet this node count). We use $\mathrm{LRL}_\phi$ as a $\mathrm{LRL}(10, 16, 16)$ network (refer Eq. 10).

### D.4 Edge aligner

The design of the edge aligner is similar to the node aligner described above in Section D.3, except that its inputs are sets of edge vectors $\boldsymbol{M}^{(q)} \in \mathbb{R}^{e\times 20}$ and $\boldsymbol{M}^{(c)} \in \mathbb{R}^{e\times 20}$. $e$ refers to the number of edges in the corpus graph (the query graph is padded to meet this edge count). We use $\mathrm{LRL}_\phi$ as a $\mathrm{LRL}(20, 16, 16)$ network (refer Eq. 16).

### D.5 $\mathrm{GumbelSinkhorn}$ operator

The $\mathrm{GumbelSinkhorn}$ operator consists of the following operations -

$$\boldsymbol{D}_0 = \exp(\boldsymbol{D}_{\text{in}}/\tau) \tag{21}$$

$$\boldsymbol{D}_{t+1} = \mathrm{RowNorm}\left(\mathrm{ColumnNorm}(\boldsymbol{D}_t)\right) \tag{22}$$

$$\boldsymbol{D}_{\text{out}} = \lim_{t\to\infty} \boldsymbol{D}_t \tag{23}$$

The matrix $\boldsymbol{D}_{\text{out}}$ obtained after this set of operations will be a doubly-stochastic matrix. The input $\boldsymbol{D}_{\text{in}}$ in our case is the matrix containing the dot product of the node/edge embeddings of the query and corpus graphs respectively. $\tau$ represents the temperature and is fixed to $0.1$ in all our experiments.

**Theorem** Equation 10 results in a permutation matrix that is row-equivariant (column-) to the shuffling of nodes in $G_q$ ($G_c$).

**Proof** To prove the equivariance of Eq. 10, we need to show that given a shuffling (permutation) of query nodes $Z \in \Pi_n$ which modifies the node embedding matrix to $Z\dot{\boldsymbol{H}}^{(q)}_{t,K}$, the resulting output of said equation would change to $Z\boldsymbol{P}_t$. Below, we consider any matrices with $Z$ in the suffix as being an intermediate expression in the computation of $\mathrm{NodeAlignerRefinement}_\phi(Z\boldsymbol{H}^{(q)}_{t,K}, \boldsymbol{H}^{(c)}_{t,K})$.

It is easy to observe that the operators $\mathrm{LRL}_\phi$ (a linear-ReLU-linear network applied to a matrix), $\mathrm{RowNorm}$, $\mathrm{ColumnNorm}$ and element-wise exponentiation ($\exp$), division are all permutation-equivariant since a shuffling of the vectors fed into these will trivially result in the output vectors getting shuffled in the same order. Thus, we get the following sequence of operations

$$\boldsymbol{D}_{\text{in},Z} = \mathrm{LRL}_\phi(Z\boldsymbol{H}^{(q)}_{t,K})\,\mathrm{LRL}_\phi(\boldsymbol{H}^{(c)}_{t,K})^\top = Z\cdot\mathrm{LRL}_\phi(\boldsymbol{H}^{(q)}_{t,K})\,\mathrm{LRL}_\phi(\boldsymbol{H}^{(c)}_{t,K})^\top\boldsymbol{D}_{\text{in}} = Z\boldsymbol{D}_{\text{in}} \tag{24}$$

$\boldsymbol{D}_{0,Z}$ equals $\exp(\boldsymbol{D}_{\text{in},Z}/\tau)$, which according to above equation would lead to $\boldsymbol{D}_{0,Z} = Z\boldsymbol{D}_0$. We can then inductively show using Eq. 22 and the equivariance of row/column normalization, assuming the following holds till $t$, that

$$\boldsymbol{D}_{t+1,Z} = \mathrm{RowNorm}\left(\mathrm{ColumnNorm}(\boldsymbol{D}_{t,Z})\right) = \mathrm{RowNorm}\left(\mathrm{ColumnNorm}(Z\boldsymbol{D}_t)\right) \tag{25}$$

$$= \mathrm{RowNorm}\left(Z\cdot\mathrm{ColumnNorm}(\boldsymbol{D}_t)\right) = Z\cdot\mathrm{RowNorm}\left(\mathrm{ColumnNorm}(\boldsymbol{D}_t)\right) = Z\boldsymbol{D}_{t+1} \tag{26}$$

The above equivariance would also hold in the limit, resulting in the doubly stochastic matrix $\boldsymbol{D}_{\text{out},Z} = Z\boldsymbol{D}_{\text{out}}$, which concludes the proof. ■

A similar proof can be followed to show column equivariance for a shuffling in the corpus nodes.

## E Variants of our models and GMN, used in the experiments

### E.1 Multi-round refinement of EINSMATCH (Node) for the corpus graph

- Initialize:

$$\boldsymbol{h}^{(c)}_0(u') = \mathrm{Init}_\theta(\mathrm{feature}(u')), \tag{27}$$

- Update the GNN embeddings as follows:

$$\boldsymbol{z}^{(c)}_{t+1,k}(u') = \mathrm{inter}_\theta\left(\boldsymbol{h}^{(c)}_{t+1,k}(u'), \textstyle\sum_{u\in V_q}\boldsymbol{h}^{(q)}_{t,k}(u)\boldsymbol{P}_t^\top[u',u]\right), \tag{28}$$

$$\boldsymbol{h}^{(c)}_{t+1,k+1}(u') = \mathrm{comb}_\theta\left(\boldsymbol{z}^{(c)}_{t+1,k}(u'), \textstyle\sum_{v'\in\mathrm{nbr}(u')}\mathrm{msg}_\theta(\boldsymbol{z}^{(c)}_{t+1,k}(u'), \boldsymbol{z}^{(c)}_{t+1,k}(v'))\right) \tag{29}$$

### E.2 Multi-layer refinement of EINSMATCH (Node)

- Initialize:

$$\boldsymbol{h}_0^{(q)}(u) = \text{Init}_\theta(\text{feature}(u)), \tag{30}$$

- The node alignment $\boldsymbol{P}_k$ is updated across layers. $\boldsymbol{P}_0$ is set to a matrix of zeros. For $k > 0$, the following equation is used:

$$\boldsymbol{P}_k = \text{NodeAlignerRefinement}_\phi\left(\boldsymbol{H}_k^{(q)}, \boldsymbol{H}_k^{(c)}\right) \tag{31}$$

$$= \text{GumbelSinkhorn}\left(\text{LRL}_\phi(\boldsymbol{H}_k^{(q)})\,\text{LRL}_\phi(\boldsymbol{H}_k^{(c)})^\top\right) \tag{32}$$

- We update the GNN embeddings as follows:

$$\boldsymbol{z}_k^{(q)}(u) = \text{inter}_\theta\left(\boldsymbol{h}_k^{(q)}(u), \textstyle\sum_{u' \in V_c} \boldsymbol{h}_k^{(c)}(u')\boldsymbol{P}_k[u, u']\right), \tag{33}$$

$$\boldsymbol{h}_{k+1}^{(q)}(u) = \text{comb}_\theta\left(\boldsymbol{z}_k^{(q)}(u), \textstyle\sum_{v \in \text{nbr}(u)} \text{msg}_\theta(\boldsymbol{z}_k^{(q)}(u), \boldsymbol{z}_k^{(q)}(v))\right) \tag{34}$$

### E.3 Multi-layer refinement of EINSMATCH (Edge)

- Initialize:

$$\boldsymbol{h}_0^{(q)}(u) = \text{Init}_{\theta,\text{node}}(\text{feature}(u)), \tag{35}$$

$$\boldsymbol{m}_0^{(q)}(e) = \text{Init}_{\theta,\text{edge}}(\text{feature}(e)), \tag{36}$$

- The edge alignment is updated across layers. $\boldsymbol{S}_0$ is set to a matrix of zeros. For $k > 0$, the following equation is used:

$$\boldsymbol{S}_k = \text{EdgeAlignerRefinement}_\phi\left(\boldsymbol{M}_k^{(q)}, \boldsymbol{M}_k^{(c)}\right) \tag{37}$$

$$= \text{GumbelSinkhorn}\left(\text{LRL}_\phi(\boldsymbol{M}_k^{(q)})\,\text{LRL}_\phi(\boldsymbol{M}_k^{(c)})^\top\right) \tag{38}$$

- We update the GNN node and edge embeddings as follows:

$$\boldsymbol{z}_k^{(q)}(e) = \text{inter}_\theta\left(\boldsymbol{m}_k^{(q)}(e), \textstyle\sum_{e' \in E_c} \boldsymbol{m}_k^{(c)}(e')\boldsymbol{S}_k[e, e']\right), \tag{39}$$

$$\boldsymbol{h}_{k+1}^{(q)}(u) = \text{comb}_\theta\left(\boldsymbol{h}_k^{(q)}(u), \textstyle\sum_{a \in \text{nbr}(u)} \text{msg}_\theta(\boldsymbol{h}_k^{(q)}(u), \boldsymbol{h}_k^{(q)}(a), \boldsymbol{z}_k^{(q)}((u, a)))\right) \tag{40}$$

$$\boldsymbol{m}_{k+1}^{(q)}((u, v)) = \text{msg}_\theta(\boldsymbol{h}_{k+1}^{(q)}(u), \boldsymbol{h}_{k+1}^{(q)}(v), \boldsymbol{z}_k^{(q)}((u, v))) \tag{41}$$

### E.4 Node partner (with additional MLP) variant of EINSMATCH (Node)

Initialization and transition between rounds is same as in EINSMATCH (Node). Below, we note the change in GNN update equations:

$$\boldsymbol{z}_{t+1,k}^{(q)}(u) = \text{inter}_\theta\left(\boldsymbol{h}_{t+1,k}^{(q)}(u), \textstyle\sum_{u' \in V_c} \boldsymbol{h}_{t,k}^{(c)}(u')\boldsymbol{P}_t[u, u']\right) \tag{42}$$

$$\boldsymbol{h}_{t+1,k+1}^{(q)}(u) = \text{comb}_\theta\left(\boldsymbol{z}_{t+1,k}^{(q)}(u), \textstyle\sum_{v \in \text{nbr}(u)} \text{msg}_\theta(\boldsymbol{h}_{t+1,k}^{(q)}(u), \boldsymbol{h}_{t+1,k}^{(q)}(v))\right) \tag{43}$$

Note that Eq. 42 is the same as Eq. 11. The node embedding enriched with node pair information is passed only as a seeding vector to the $\text{comb}_\theta$ network and not into the $\text{msg}_\theta$ network. This helps establish whether including information from node-pair partners in the messages is crucial towards EINSMATCH (Node) or not. As pointed out through the additional experiments in Appendix G.4, using $\boldsymbol{h}^{(q)}$ instead of $\boldsymbol{z}^{(q)}$ in the $\text{msg}_\theta$ network harms the performance of the model, making it clear that the representations of messages greatly benefit from node-pair partner information.

### E.5 Node pair partner ($\text{msg}$ only) variant of EINSMATCH (Node)

Initialization and transition between rounds is same as in EINSMATCH (Node). Below, we note the change in GNN update equations:

$$\boldsymbol{z}_{t+1,k}^{(q)}(u) = \text{inter}_\theta\left(\boldsymbol{h}_{t+1,k}^{(q)}(u), \textstyle\sum_{u' \in V_c} \boldsymbol{h}_{t,k}^{(c)}(u')\boldsymbol{P}_t[u, u']\right) \tag{44}$$

$$\boldsymbol{h}^{(q)}_{t+1,k+1}(u) = \mathrm{comb}_\theta\left(\boldsymbol{h}^{(q)}_{t+1,k}(u), \textstyle\sum_{v\in \mathrm{nbr}(u)} \mathrm{msg}_\theta(\boldsymbol{z}^{(q)}_{t+1,k}(u), \boldsymbol{z}^{(q)}_{t+1,k}(v))\right) \tag{45}$$

Note that Eq. 42 is the same as Eq. 11. The node embedding enriched with node pair information is passed only to the $\mathrm{msg}_\theta$ network and not as the seeding vector in the $\mathrm{comb}_\theta$ network. This helps pinpoint the exact mechanism through which node pair partner interaction is assisting EINSMATCH (Node). As pointed out through the additional experiments in Appendix G.4, using $\boldsymbol{h}^{(q)}$ instead of $\boldsymbol{z}^{(q)}$ as seeding vector in the $\mathrm{comb}_\theta$ network does not harm the performance of the model by a lot, highlighting that including node-pair partner information in the message representations is reasonably good for improving performance.

### E.6 Variants of GMN

**GMN ($K$-layer early interaction):** In GMN, two graphs $G_q$ and $G_c$ are matched via GNN-style message passing. Two kinds of messages are passed in each layer $k$: *within* each graph, for $(j, i) \in E$,

$$m_{j\to i}(k) = \mathrm{msg}_{\text{within}}(h_j(k), h_i(k), x_{ji}) \tag{46}$$

and *across* graphs ($i \in V_q, j' \in V_c$ or vice versa),

$$\mu_{j'\to i}(k) = \mathrm{msg}_{\text{across}}(h_{j'}(k), h_i(k)) \tag{47}$$

Embeddings of nodes in $V_q$ are updated as

$$h^{(q)}_u(k+1) = \mathrm{comb}\left(h^{(q)}_u(k), \underset{(v,u)\in E_q}{\mathrm{aggr}}(m(k, v\to u)), \underset{v\in V_c}{\mathrm{aggr}}(\mu(k, v\to u))\right) \tag{48}$$

and vice versa from nodes in $V_q$ to nodes in $V_c$. These node embeddings can be collected into $H^{(q)}(k)$ and $H^{(c)}(k)$. We will elide superscipts $(q), (c)$ when there is no risk of confusion.

After the last layer, vector sets $H^{(q)}(K)$ and $H^{(c)}(K)$ can be compared as before to implement early-interaction graph matching.

# F Additional details about experimental setup

## F.1 Datasets

We use six datasets from the TUDatasets collection [27] for benchmarking our methods with respect to existing baselines. Lou et al. [23] devised a method to sample query and corpus graphs from the graphs present in these datasets to create their training data. We adopt it for the task of subgraph matching. In particular, we choose a node $u \in G$ as the center of a Breadth First Search (BFS) and run the algorithm till $|V|$ nodes are traversed, where the range of $|V|$ is listed in Table 7 (refer to the Min and Max columns for $|V_q|$ and $|V_c|$). This process is independently performed for the query and corpus splits (with different ranges for graph size) to obtain $300$ query graphs and $800$ corpus graphs. The set of query graphs is split into train, validation and test splits of $180$ $(60\%)$, $45$ $(15\%)$ and $75$ $(25\%)$ graphs respectively. Ground truth labels are computed for each query-corpus graph pair using the VF2 algorithm [9, 15, 23] implemented in the Networkx library. Various statistics about the datasets are listed in Table 7. $\text{pairs}(y)$ denotes the number of pairs in the dataset with gold label $y$, where $y \in \{0, 1\}$.

| | Mean $\lvert V_q\rvert$ | Min $\lvert V_q\rvert$ | Max $\lvert V_q\rvert$ | Mean $\lvert E_q\rvert$ | Mean $\lvert V_c\rvert$ | Min $\lvert V_c\rvert$ | Max $\lvert V_c\rvert$ | Mean $\lvert E_c\rvert$ | pairs(1) | pairs(0) | $\frac{\text{pairs}(1)}{\text{pairs}(0)}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| AIDS | 11.25 | 6 | 14 | 11.25 | 18.87 | 16 | 23 | 18.87 | 41001 | 198999 | 0.2118 |
| Mutag | 13.27 | 6 | 17 | 13.27 | 19.89 | 16 | 24 | 19.89 | 42495 | 197505 | 0.2209 |
| FM | 11.35 | 6 | 14 | 11.35 | 18.81 | 16 | 24 | 18.81 | 40516 | 199484 | 0.2085 |
| FR | 11.39 | 6 | 14 | 11.39 | 18.79 | 16 | 24 | 18.79 | 39829 | 200171 | 0.2043 |
| MM | 11.37 | 6 | 14 | 11.37 | 18.79 | 16 | 24 | 18.79 | 40069 | 199931 | 0.2056 |
| MR | 11.49 | 6 | 14 | 11.49 | 18.78 | 16 | 24 | 18.78 | 40982 | 199018 | 0.2119 |

Table 7: Statistics for the 6 datasets borrowed from the TUDatasets collection [27]

## F.2 Baselines

**GraphSim, GOTSim, SimGNN, Neuromatch, GEN, GMN, IsoNet (Node), IsoNet (Edge)**: We utilized the code from official implementation of [35] [1]. Some *for loops* were vectorized to improve the running time of GMN.
**EGSC**: The official implementation [2] is refactored and integrated into our code.
**H2MN**: We use the official code from [3].
**GREED**: We use the official code from [4]. The model is adapted from the graph edit distance (GED) task to the subgraph isomorphism task, using a hinge scoring layer.

The number of parameters involved in all models (our methods and baselines) are reported in Table 8.

| | Number of parameters |
|---|---|
| GraphSim [2] | 3909 |
| GOTSim [11] | 304 |
| SimGNN [1] | 1671 |
| EGSC [31] | 3948 |
| H2MN [45] | 2974 |
| Neuromatch [23] | 3463 |
| GREED [32] | 1840 |
| GEN [22] | 1750 |
| GMN [22] | 2050 |
| IsoNet (Node) [35] | 1868 |
| IsoNet (Edge) [35] | 2028 |
| EINSMATCH (Node) | 2498 |
| EINSMATCH (Edge) | 4908 |

Table 8: Number of parameters for all models used in comparison

[1] https://github.com/Indradyumna/ISONET/
[2] https://github.com/canqin001/Efficient_Graph_Similarity_Computation
[3] https://github.com/cszhangzhen/H2MN
[4] https://github.com/idea-iitd/greed

### F.3 Calculation of Metrics: Mean Average Precision (MAP), HITS@K, Precision@K and Mean Reciprocal Rank (MRR)

Given a ranked list of corpus graphs $C = \{G_c\}$ for a test query $G_q$, sorted in the decreasing order of $\Delta_{\theta,\phi}(G_c|G_q)$, let us assume that the $c_+^{\text{th}}$ relevant graph is placed at position $\text{pos}(c_+) \in \{1, ..., |C|\}$ in the ranked list. Then Average Precision (AP) is computed as:

$$\text{AP}(q) = \frac{1}{|C_{q+}|} \sum_{c_+ \in [|C_{q+}|]} \frac{c_+}{\text{pos}(c_+)} \tag{49}$$

Mean average precision is defined as $\sum_{q \in Q} \text{AP}(q)/|Q|$.

Precision@$K(q) = \frac{1}{K}$ # relevant graphs corresponding to $G_q$ till rank $K$. Finally we report the mean of Precision@$K(q)$ across queries.

Reciprocal rank or RR$(q)$ is the inverse of the rank of the topmost relevant corpus graph corresponding to $G_q$ in the ranked list. Mean reciprocal rank (MRR) is average of RR$(q)$ across queries.

HITS@$K$ for a query $G_q$ is defined as the fraction of positively labeled corpus graphs that appear before the $K^{\text{th}}$ negatively labeled corpus graph. Finally, we report the average of HITS@$K$ across queries.

Note that HITS@K is a significantly aggressive metric compared to Precision@K and MRR, as can be seen in Tables 11 and 12.

### F.4 Details about hyperparameters

All models were trained using early stopping with MAP score on the validation split as a stopping criterion. For early stopping, we used a patience of $50$ with a tolerance of $10^{-4}$. We used the Adam optimizer with the learning rate as $10^{-3}$ and the weight decay parameter as $5 \cdot 10^{-4}$. We set batch size to $128$ and maximum number of epochs to $1000$.

**Seed Selection and Reproducibility** Five integer seeds were chosen uniformly at random from the range $[0, 10^4]$ resulting in the set $\{1704, 4929, 7366, 7474, 7762\}$. EINSMATCH (Node), GMN and IsoNet (Edge) were trained on each of these $5$ seeds for all $6$ datasets. Note that these seeds do not control the training-dev-test splits but only control the initialization. Since the overall problem is non-convex, in principle, one should choose the best initial conditions leading to local minima. Hence, for all models, we choose the best seed, based on validation MAP score, is shown in Table 9.

| | AIDS | Mutag | FM | FR | MM | MR |
|---|---|---|---|---|---|---|
| GraphSim [2] | 7762 | 4929 | 7762 | 7366 | 4929 | 7474 |
| GOTSim [11] | 7762 | 7366 | 1704 | 7762 | 1704 | 7366 |
| SimGNN [1] | 7762 | 7474 | 1704 | 4929 | 4929 | 7762 |
| EGSC [31] | 4929 | 1704 | 7762 | 4929 | 4929 | 7366 |
| H2MN [45] | 7762 | 4929 | 7366 | 1704 | 4929 | 7474 |
| Neuromatch [23] | 7366 | 4929 | 7762 | 7762 | 1704 | 7366 |
| GREED [32] | 7762 | 1704 | 1704 | 7474 | 1704 | 1704 |
| GEN [22] | 1704 | 4929 | 7474 | 7762 | 1704 | 1704 |
| GMN [22] | 7366 | 4929 | 7366 | 7474 | 7474 | 7366 |
| IsoNet (Node) [35] | 7474 | 7474 | 7474 | 1704 | 4929 | 1704 |
| IsoNet (Edge) [35] | 7474 | 7474 | 7474 | 1704 | 4929 | 1704 |
| GMN [22] | 7366 | 4929 | 7366 | 7474 | 7474 | 7366 |
| EINSMATCH (Node) | 7762 | 7762 | 7474 | 7762 | 7762 | 7366 |

Table 9: Best seeds for all models. For IsoNet (Edge), GMN and EINSMATCH (Node), these are computed based on MAP score on the validation split at convergence. For other models, the identification occurs after $10$ epochs of training.

EINSMATCH (Edge) and all ablations on top of EINSMATCH (Node) were trained using the best seeds for EINSMATCH (Node) (as in Tables 3, 4 and 15). Ablations of GMN were trained with the best GMN seeds.

For baselines excluding IsoNet (Edge), models were trained on all 5 seeds for 10 epochs and the MAP scores on the validation split were considered. Full training with early stopping was resumed only for the best seed per dataset. This approach was adopted to reduce the computational requirements for benchmarking.

**Margin Selection** For GraphSim, GOTSim, SimGNN, Neuromatch, GEN, GMN and IsoNet (Edge), we use the margins determined by Roy et al. [35] for each dataset. For IsoNet (Node), the margins prescribed for IsoNet (Edge) were used for standardization. For EINSMATCH (Node), EINSMATCH (Edge) and ablations, a fixed margin of 0.5 is used.

Procedure for baselines **EGSC, GREED, H2MN**: They are trained on five seeds with a margin of 0.5 for 10 epochs and the best seed is chosen using the validation MAP score at this point. This seed is also used to train a model with a margin of 0.1 for 10 epochs. The better of these models, again using MAP score on the validation split, is identified and retrained till completion using early stopping.

| | AIDS | Mutag | FM | FR | MM | MR |
|---|---|---|---|---|---|---|
| GraphSim [2] | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| GOTSim [11] | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| SimGNN [1] | 0.5 | 0.1 | 0.5 | 0.1 | 0.5 | 0.5 |
| EGSC [31] | 0.1 | 0.5 | 0.1 | 0.5 | 0.1 | 0.5 |
| H2MN [45] | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.1 |
| Neuromatch [23] | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| GREED [32] | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| GEN [22] | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| GMN [22] | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| IsoNet (Node) [35] | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| IsoNet (Edge) [35] | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |

Table 10: Best margin for baselines used in comparison.

### F.5 Software and Hardware

All experiments were run with Python 3.10.13 and PyTorch 2.1.2. EINSMATCH (Node), EINSMATCH (Edge), GMN, IsoNet (Edge) and ablations on top of these were trained on Nvidia RTX A6000 (48 GB) GPUs while other baselines like GraphSim, GOTSim etc. were trained on Nvidia A100 (80 GB) GPUs.

As an estimate of training time, we typically spawn 3 training runs of EINSMATCH (Node) or EINSMATCH (Edge) on one Nvidia RTX A6000 GPU, each of which takes 300 epochs to conclude on average, with an average of 6-12 minutes per epoch. This amounts to 2 days of training. Overloading the GPUs by spawning 6 training runs per GPU increases the training time marginally to 2.5 days.

Additionally, we use wandb [5] to manage and monitor the experiments.

### F.6 License

GEN, GMN, GOTSim, GREED and EGSC are available under the MIT license, while SimGNN is public under the GNU license. The licenses for GraphSim, H2MN, IsoNet (Node), IsoNet (Edge), Neuromatch could not be identified. The authors were unable to identify the license of the TUDatasets repository [27], which was used to compile the 6 datasets used in this paper.

# G Additional experiments

## G.1 Comparison against baselines

In Tables 11 and 12, we report the Mean Average Precision (MAP), HITS@20, MRR and Precision@20 scores for several baselines as well as the four approaches discussed in our paper - multi-layer and multi-round variants of EINSMATCH (Node) and EINSMATCH (Edge). Multi-round EINSMATCH (Edge) outperforms all other models with respect to all metrics, closely followed by multi-round EINSMATCH (Node) and multi-layer EINSMATCH (Edge) respectively. Among the baselines, IsoNet (Edge) is the best-performing model, closely followed by IsoNet (Node) and GMN.

For MRR, Precision@20, the comparisons are less indicative of the significant boost in performance obtained by EINSMATCH, since these are not aggressive metrics from the point of view of information retrieval.

| Mean Average Precision (MAP) | | | | | | |
|---|---|---|---|---|---|---|
| | AIDS | Mutag | FM | FR | MM | MR |
| GraphSim [2] | 0.356 ± 0.016 | 0.472 ± 0.027 | 0.477 ± 0.016 | 0.423 ± 0.019 | 0.415 ± 0.017 | 0.453 ± 0.018 |
| GOTSim [11] | 0.324 ± 0.015 | 0.272 ± 0.012 | 0.355 ± 0.014 | 0.373 ± 0.018 | 0.323 ± 0.015 | 0.317 ± 0.013 |
| SimGNN [1] | 0.341 ± 0.019 | 0.283 ± 0.012 | 0.473 ± 0.016 | 0.341 ± 0.015 | 0.298 ± 0.012 | 0.379 ± 0.015 |
| EGSC [31] | 0.505 ± 0.02 | 0.476 ± 0.022 | 0.609 ± 0.018 | 0.607 ± 0.019 | 0.586 ± 0.019 | 0.58 ± 0.018 |
| H2MN [45] | 0.267 ± 0.014 | 0.276 ± 0.012 | 0.436 ± 0.015 | 0.412 ± 0.016 | 0.312 ± 0.014 | 0.243 ± 0.008 |
| Neuromatch [23] | 0.489 ± 0.024 | 0.576 ± 0.029 | 0.615 ± 0.019 | 0.559 ± 0.024 | 0.519 ± 0.02 | 0.606 ± 0.021 |
| GREED [32] | 0.472 ± 0.021 | 0.567 ± 0.027 | 0.558 ± 0.02 | 0.512 ± 0.021 | 0.546 ± 0.021 | 0.528 ± 0.019 |
| GEN [22] | 0.557 ± 0.021 | 0.605 ± 0.028 | 0.661 ± 0.021 | 0.575 ± 0.02 | 0.539 ± 0.02 | 0.631 ± 0.018 |
| GMN [22] | 0.622 ± 0.02 | 0.710 ± 0.025 | 0.730 ± 0.018 | 0.662 ± 0.02 | 0.655 ± 0.02 | 0.708 ± 0.017 |
| IsoNet (Node) [35] | 0.659 ± 0.022 | 0.697 ± 0.026 | 0.729 ± 0.018 | 0.68 ± 0.022 | 0.708 ± 0.016 | 0.738 ± 0.017 |
| IsoNet (Edge) [35] | 0.690 ± 0.02 | 0.706 ± 0.026 | 0.783 ± 0.017 | 0.722 ± 0.02 | 0.753 ± 0.015 | 0.774 ± 0.016 |
| multi-layer EINSM. (Node) | 0.756 ± 0.019 | 0.81 ± 0.021 | 0.859 ± 0.015 | 0.802 ± 0.018 | 0.827 ± 0.015 | 0.841 ± 0.013 |
| multi-layer EINSM. (Edge) | 0.795 ± 0.018 | 0.805 ± 0.022 | 0.883 ± 0.013 | 0.812 ± 0.016 | 0.862 ± 0.013 | 0.886 ± 0.011 |
| multi-round EINSM. (Node) | 0.825 ± 0.016 | 0.851 ± 0.018 | 0.888 ± 0.012 | 0.855 ± 0.015 | 0.838 ± 0.015 | 0.874 ± 0.011 |
| multi-round EINSM. (Edge) | 0.847 ± 0.016 | 0.858 ± 0.019 | 0.902 ± 0.012 | 0.875 ± 0.014 | 0.902 ± 0.01 | 0.902 ± 0.01 |

| HITS@20 | | | | | | |
|---|---|---|---|---|---|---|
| | AIDS | Mutag | FM | FR | MM | MR |
| GraphSim [2] | 0.145 ± 0.011 | 0.257 ± 0.027 | 0.261 ± 0.015 | 0.227 ± 0.015 | 0.212 ± 0.014 | 0.23 ± 0.015 |
| GOTSim [11] | 0.112 ± 0.011 | 0.088 ± 0.009 | 0.147 ± 0.011 | 0.166 ± 0.014 | 0.119 ± 0.011 | 0.116 ± 0.011 |
| SimGNN [1] | 0.138 ± 0.016 | 0.087 ± 0.008 | 0.235 ± 0.015 | 0.155 ± 0.013 | 0.111 ± 0.009 | 0.160 ± 0.013 |
| EGSC [31] | 0.267 ±0.023 | 0.243 ±0.02 | 0.364 ±0.02 | 0.382 ±0.024 | 0.348 ±0.023 | 0.325 ±0.021 |
| H2MN [45] | 0.076 ± 0.009 | 0.084 ± 0.007 | 0.200 ± 0.012 | 0.189 ± 0.013 | 0.119 ± 0.011 | 0.069 ± 0.004 |
| Neuromatch [23] | 0.262 ± 0.025 | 0.376 ± 0.034 | 0.389 ± 0.022 | 0.350 ± 0.025 | 0.282 ± 0.019 | 0.385 ± 0.025 |
| GREED [32] | 0.245 ± 0.025 | 0.371 ± 0.034 | 0.316 ± 0.027 | 0.287 ± 0.019 | 0.311 ± 0.024 | 0.277 ± 0.023 |
| GEN [22] | 0.321 ± 0.026 | 0.429 ± 0.035 | 0.448 ± 0.03 | 0.368 ± 0.026 | 0.292 ± 0.024 | 0.391 ± 0.025 |
| GMN [22] | 0.397 ± 0.029 | 0.544 ± 0.035 | 0.537 ± 0.027 | 0.45 ± 0.027 | 0.423 ± 0.025 | 0.49 ± 0.026 |
| IsoNet (Node) [35] | 0.438 ± 0.028 | 0.509 ± 0.034 | 0.525 ± 0.026 | 0.475 ± 0.03 | 0.493 ± 0.023 | 0.532 ± 0.025 |
| IsoNet (Edge) [35] | 0.479 ± 0.029 | 0.529 ± 0.035 | 0.613 ± 0.026 | 0.538 ± 0.029 | 0.571 ± 0.023 | 0.601 ± 0.027 |
| multi-layer EINSM. (Node) | 0.57 ± 0.029 | 0.672 ± 0.033 | 0.744 ± 0.027 | 0.657 ± 0.031 | 0.68 ± 0.025 | 0.707 ± 0.024 |
| multi-layer EINSM. (Edge) | 0.626 ± 0.029 | 0.671 ± 0.035 | 0.775 ± 0.026 | 0.67 ± 0.028 | 0.743 ± 0.024 | 0.776 ± 0.021 |
| multi-round EINSM. (Node) | 0.672 ± 0.027 | 0.732 ± 0.03 | 0.797 ± 0.024 | 0.737 ± 0.026 | 0.702 ± 0.025 | 0.755 ± 0.022 |
| multi-round EINSM. (Edge) | 0.705 ± 0.028 | 0.749 ± 0.032 | 0.813 ± 0.023 | 0.769 ± 0.026 | 0.809 ± 0.019 | 0.803 ± 0.02 |

Table 11: Replication of Table 2 with standard error. Comparison of the two variants of EINSMATCH (EINSMATCH (Node) and EINSMATCH (Edge)) against all the state-of-the-art graph retrieval methods, across all six datasets. Performance is measured in terms average precision MAP and HITS@20. In all cases, we used 60% training, 15% validation and 25% test sets. The first five methods apply a neural network on the fused graph-pair representations. The next six methods apply asymmetric hinge distance between the query and corpus embeddings similar to our method. The numbers with green and yellow indicate the best, second best method respectively, whereas the numbers with blue indicate the best method among the baselines. (MAP values for EINSMATCH (Edge) across FM, MM and MR are verified to be not exactly same, but they take the same value until the third decimal).

| Mean Reciprocal Rank (MRR) | | | | | | |
|---|---|---|---|---|---|---|
| | AIDS | Mutag | FM | FR | MM | MR |
| GraphSim [2] | 0.71 ±0.039 | 0.795 ±0.037 | 0.885 ±0.029 | 0.817 ±0.032 | 0.818 ±0.034 | 0.789 ±0.037 |
| GOTSim [11] | 0.568 ±0.038 | 0.584 ±0.037 | 0.775 ±0.037 | 0.716 ±0.042 | 0.459 ±0.045 | 0.525 ±0.047 |
| SimGNN [1] | 0.533 ±0.038 | 0.644 ±0.043 | 0.866 ±0.031 | 0.753 ±0.038 | 0.669 ±0.04 | 0.638 ±0.046 |
| EGSC [31] | 0.894 ±0.026 | 0.75 ±0.041 | 0.943 ±0.021 | 0.909 ±0.023 | 0.904 ±0.025 | 0.932 ±0.022 |
| H2MN [45] | 0.46 ±0.047 | 0.565 ±0.042 | 0.822 ±0.035 | 0.817 ±0.034 | 0.386 ±0.039 | 0.62 ±0.041 |
| Neuromatch [23] | 0.823 ±0.035 | 0.855 ±0.035 | 0.88 ±0.028 | 0.929 ±0.022 | 0.87 ±0.027 | 0.895 ±0.026 |
| GREED [32] | 0.789 ±0.035 | 0.805 ±0.034 | 0.834 ±0.033 | 0.834 ±0.032 | 0.894 ±0.028 | 0.759 ±0.039 |
| GEN [22] | 0.865 ±0.028 | 0.895 ±0.029 | 0.889 ±0.026 | 0.878 ±0.028 | 0.814 ±0.034 | 0.878 ±0.026 |
| GMN [22] | 0.877 ±0.027 | 0.923 ±0.023 | 0.949 ±0.019 | 0.947 ±0.019 | 0.928 ±0.023 | 0.922 ±0.022 |
| IsoNet (Node) [35] | 0.916 ±0.024 | 0.887 ±0.029 | 0.977 ±0.013 | 0.954 ±0.018 | 0.956 ±0.018 | 0.954 ±0.018 |
| IsoNet (Edge) [35] | 0.949 ±0.02 | 0.926 ±0.026 | 0.973 ±0.013 | 0.956 ±0.018 | 0.98 ±0.011 | 0.948 ±0.019 |
| multi-layer EINSM. (Node) | 0.956 ±0.018 | 0.954 ±0.018 | 1.0 ±0.0 | 0.978 ±0.013 | 0.98 ±0.011 | 1.0 ±0.0 |
| multi-layer EINSM. (Edge) | 0.984 ±0.011 | 0.976 ±0.014 | 0.991 ±0.009 | 0.987 ±0.009 | 0.987 ±0.009 | 0.993 ±0.007 |
| multi-round EINSM. (Node) | 0.993 ±0.007 | 0.971 ±0.014 | 1.0 ±0.0 | 0.993 ±0.007 | 0.993 ±0.007 | 0.993 ±0.007 |
| multi-round EINSM. (Edge) | 1.0 ±0.0 | 0.983 ±0.012 | 0.991 ±0.009 | 1.0 ±0.0 | 1.0 ±0.0 | 1.0 ±0.0 |

| Precision@20 | | | | | | |
|---|---|---|---|---|---|---|
| | AIDS | Mutag | FM | FR | MM | MR |
| GraphSim [2] | 0.474 ±0.025 | 0.577 ±0.033 | 0.679 ±0.023 | 0.617 ±0.028 | 0.604 ±0.028 | 0.638 ±0.026 |
| GOTSim [11] | 0.386 ±0.024 | 0.325 ±0.021 | 0.479 ±0.027 | 0.519 ±0.03 | 0.409 ±0.027 | 0.421 ±0.03 |
| SimGNN [1] | 0.44 ±0.026 | 0.33 ±0.022 | 0.626 ±0.026 | 0.471 ±0.029 | 0.414 ±0.026 | 0.512 ±0.032 |
| EGSC [31] | 0.646 ±0.023 | 0.608 ±0.034 | 0.79 ±0.022 | 0.766 ±0.021 | 0.739 ±0.023 | 0.74 ±0.021 |
| H2MN [45] | 0.28 ±0.026 | 0.34 ±0.023 | 0.587 ±0.024 | 0.563 ±0.026 | 0.399 ±0.028 | 0.308 ±0.017 |
| Neuromatch [23] | 0.615 ±0.03 | 0.689 ±0.032 | 0.809 ±0.022 | 0.725 ±0.027 | 0.694 ±0.027 | 0.751 ±0.023 |
| GREED [32] | 0.591 ±0.024 | 0.661 ±0.03 | 0.689 ±0.026 | 0.642 ±0.028 | 0.699 ±0.028 | 0.624 ±0.029 |
| GEN [22] | 0.674 ±0.024 | 0.721 ±0.03 | 0.783 ±0.023 | 0.678 ±0.022 | 0.64 ±0.027 | 0.759 ±0.021 |
| GMN [22] | 0.751 ±0.022 | 0.82 ±0.023 | 0.852 ±0.02 | 0.809 ±0.019 | 0.783 ±0.022 | 0.832 ±0.018 |
| IsoNet (Node) [35] | 0.791 ±0.022 | 0.803 ±0.029 | 0.866 ±0.018 | 0.803 ±0.022 | 0.844 ±0.015 | 0.863 ±0.016 |
| IsoNet (Edge) [35] | 0.822 ±0.022 | 0.812 ±0.028 | 0.896 ±0.016 | 0.851 ±0.017 | 0.877 ±0.014 | 0.875 ±0.017 |
| multi-layer EINSM. (Node) | 0.873 ±0.018 | 0.897 ±0.018 | 0.935 ±0.012 | 0.917 ±0.012 | 0.93 ±0.013 | 0.931 ±0.012 |
| multi-layer EINSM. (Edge) | 0.905 ±0.015 | 0.883 ±0.021 | 0.958 ±0.01 | 0.93 ±0.01 | 0.953 ±0.01 | 0.976 ±0.005 |
| multi-round EINSM. (Node) | 0.932 ±0.012 | 0.943 ±0.011 | 0.957 ±0.01 | 0.961 ±0.008 | 0.949 ±0.011 | 0.963 ±0.008 |
| multi-round EINSM. (Edge) | 0.946 ±0.012 | 0.931 ±0.014 | 0.973 ±0.007 | 0.963 ±0.008 | 0.98 ±0.005 | 0.987 ±0.003 |

Table 12: MRR and Precision@20 of corresponding models from Table 2 with standard error. Comparison of the two variants of EINSMATCH (EINSMATCH (Node) and EINSMATCH (Edge)) against all the state-of-the-art graph retrieval methods, across all six datasets. Performance is measured in terms MRR and Precision@20. In all cases, we used 60% training, 15% validation and 25% test sets. The first five methods apply a neural network on the fused graph-pair representations. The next six methods apply asymmetric hinge distance between the query and corpus embeddings similar to our method. The numbers with green and yellow indicate the best, second best method respectively, whereas the numbers with blue indicate the best method among the baselines.

### G.2 HITS@20, MRR and Precision@20 for multi-round EINSMATCH and multi-layer EINSMATCH

Table 13 compares multi-round and multi-layer EINSMATCH with respect to different metrics. We observe that multi-round EINSMATCH outperforms multi-layer EINSMATCH by a significant margin when it comes to all metrics, both when the models are node-based or edge-based. This reinforces the observations from MAP scores noted earlier in Table 3. Note that a minor exception occurs for MRR but the scores are already so close to 1 that this particular metric can be discounted and our key observation above still stands.

### G.3 Refinement of alignment matrix across rounds and layers in multi-round EINSMATCH and multi-layer EINSMATCH

The node (edge) alignment calculated after round $t$ is denoted as $\boldsymbol{P}_t$ ($\boldsymbol{S}_t$). We accumulate such alignments across multiple rounds. This also includes $\boldsymbol{P}_T$ ($\boldsymbol{S}_T$) which is used to compute the

| | | HITS@20 | | | | | |
|---|---|---|---|---|---|---|---|
| | | AIDS | Mutag | FM | FR | MM | MR |
| Node | Multi-layer | 0.57 | 0.672 | 0.744 | 0.657 | 0.68 | 0.707 |
| | Multi-round | 0.672 | 0.732 | 0.797 | 0.737 | 0.702 | 0.755 |
| Edge | Multi-layer | 0.626 | 0.671 | 0.775 | 0.67 | 0.743 | 0.776 |
| | Multi-round | 0.705 | 0.749 | 0.813 | 0.769 | 0.809 | 0.803 |

| | | Mean Reciprocal Rank (MRR) | | | | | |
|---|---|---|---|---|---|---|---|
| | | AIDS | Mutag | FM | FR | MM | MR |
| Node | Multi-layer | 0.956 | 0.954 | 1.0 | 0.978 | 0.98 | 1.0 |
| | Multi-round | 0.993 | 0.971 | 1.0 | 0.993 | 0.993 | 0.993 |
| Edge | Multi-layer | 0.984 | 0.976 | 0.991 | 0.987 | 0.987 | 0.993 |
| | Multi-round | 1.0 | 0.983 | 0.991 | 1.0 | 1.0 | 1.0 |

| | | Precision@20 | | | | | |
|---|---|---|---|---|---|---|---|
| | | AIDS | Mutag | FM | FR | MM | MR |
| Node | Multi-layer | 0.873 | 0.897 | 0.935 | 0.917 | 0.93 | 0.931 |
| | Multi-round | 0.932 | 0.943 | 0.957 | 0.961 | 0.949 | 0.963 |
| Edge | Multi-layer | 0.905 | 0.883 | 0.958 | 0.93 | 0.953 | 0.976 |
| | Multi-round | 0.946 | 0.931 | 0.973 | 0.963 | 0.98 | 0.987 |

Table 13: Multi-round vs. multi-layer refinement. First and the last two rows of each table report HITS@20, MRR and Precision@20 for EINSMATCH (Node) and EINSMATCH (Edge) respectively. Rows colored green and yellow indicate the best and second best methods respectively.

relevance distance in Eq. 13 (Eq. 20). We wish to compare the predicted alignments with ground truth alignments. We expect our final alignment matrix $\boldsymbol{P}_t$ ($\boldsymbol{S}_t$) to be one of them. We determine the closest ground truth matrices $\boldsymbol{P}^*$ and $\boldsymbol{S}^*$ by computing $\max_{\boldsymbol{P}} \mathrm{Tr}(\boldsymbol{P}_T^\top \boldsymbol{P})$ and $\max_{\boldsymbol{S}} \mathrm{Tr}(\boldsymbol{S}_T^\top \boldsymbol{S})$ for EINSMATCH (Node) and EINSMATCH (Edge) respectively. We now use the closest ground-truth alignment $\boldsymbol{P}^*$, to compute $\mathrm{Tr}(\boldsymbol{P}_t^\top \boldsymbol{P}^*)$ for $t \in [T]$. For each $t$, we plot a histogram with bin width 0.1 that denotes the density estimate $\mathrm{p}(\mathrm{Tr}(\boldsymbol{P}_t^\top \boldsymbol{P}^*))$. The same procedure is adopted for edges, with $\boldsymbol{S}^*$ used instead of $\boldsymbol{P}^*$. The histograms are depicted in Figure 14. We observe that the plots shift rightward with increasing $t$. The frequency of graph pairs with misaligned $\boldsymbol{P}_t$ ($\boldsymbol{S}_t$) decreases with rounds $t$ while that with well-aligned $\boldsymbol{P}_t$ ($\boldsymbol{S}_t$) increases.

Here, we also study alignments obtained through multi-layer refinement. We adopt the same procedure as in Section G.3. One key difference is that the node/edge alignments are computed after every layer $k$ and are accumulated across layers $k \in [K]$. In Figure 14, we observe that the plots, in general, shift rightward with increasing $k$. The frequency of graph pairs with misaligned $\boldsymbol{P}_t$ ($\boldsymbol{S}_t$) decreases with rounds $k$ while that with well-aligned $\boldsymbol{P}_k$ ($\boldsymbol{S}_k$) increases.

### G.4 Comparison across alternatives of multi-layer EINSMATCH (Node) and multi-round EINSMATCH (Node)

In Table 15, we compare different alternatives to the multi-round and multi-layer variants of EINSMATCH (Node). In particular, we consider four alternatives - Node partner (equation shown in Section 4), Node partner (with additional MLP) [Appendix E.4], Node pair partner ($\mathrm{msg}$ only) [Appendix E.5] and EINSMATCH (Node). We observe that for all metrics, EINSMATCH (Node) and Node pair partner ($\mathrm{msg}$ only) dominate the other alternatives in most cases. This highlights the importance of node pair partner interaction for determining the subgraph isomorphism relationship between two graphs. For the multi-round variant, EINSMATCH (Node) outperforms Node pair partner ($\mathrm{msg}$ only) in four of the datasets and is comparable / slightly worse in the other two. Once again, comparisons based on MRR break down because it does not cause a strong differentiation between the approaches.

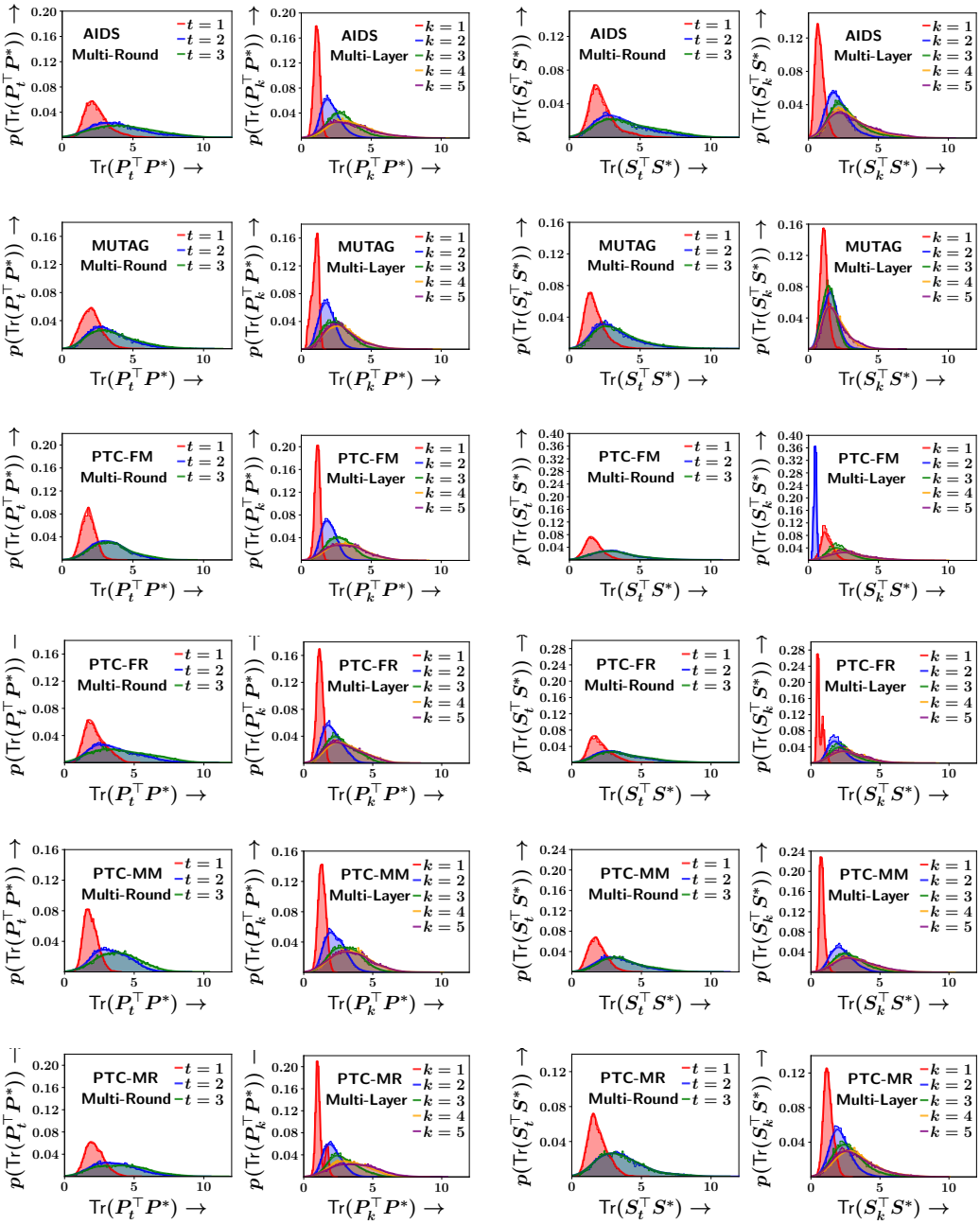



Figure 14: Similar to Figure 5, we plot empirical probability density of $p(\mathrm{Tr}(\boldsymbol{P}_t^\top \boldsymbol{P}^*))$ and $p(\mathrm{Tr}(\boldsymbol{S}_t^\top \boldsymbol{S}^*))$ for different values of $t$ lazy multi round updates and $p(\mathrm{Tr}(\boldsymbol{P}_k^\top \boldsymbol{P}^*))$ and $p(\mathrm{Tr}(\boldsymbol{S}_k^\top \boldsymbol{S}^*))$ for different values of $k$ for eager multi layer updates. The first (last) two plots in the left (right) of each row are for multi-round EINSMATCH (Node) (multi-round EINSMATCH (Edge)).

### G.5 Comparison of GMN with EINSMATCH alternative for multi-layer and multi-round

In Table 16, we modify the GMN architecture to include node pair partner interaction in the message-passing layer. Based on the reported metrics, we observe that there is no substantial improvement upon including information from node pairs in GMN, which is driven by a non-injective mapping (attention). This indicates that injectivity of the doubly stochastic matrix in our formulation is crucial towards the boost in performance obtained from node pair partner interaction as well.

| | Mean Average Precision (MAP) | | | | | | |
|---|---|---|---|---|---|---|---|
| | | AIDS | Mutag | FM | FR | MM | MR |
| **Multi-Layer** | Node partner (with additional MLP) | 0.692 | 0.782 | 0.822 | 0.776 | 0.777 | 0.803 |
| | Node pair partner (msg only) | 0.765 | 0.792 | 0.876 | 0.823 | 0.843 | 0.848 |
| | Node partner | 0.668 | 0.783 | 0.821 | 0.752 | 0.753 | 0.794 |
| | EINSMATCH (Node) | 0.756 | 0.81 | 0.859 | 0.802 | 0.827 | 0.841 |
| **Multi-Round** | Node partner (with additional MLP) | 0.815 | 0.844 | 0.868 | 0.852 | 0.818 | 0.858 |
| | Node pair partner (msg only) | 0.818 | 0.833 | 0.897 | 0.831 | 0.852 | 0.871 |
| | Node partner | 0.776 | 0.829 | 0.851 | 0.819 | 0.844 | 0.84 |
| | EINSMATCH (Node) | 0.825 | 0.851 | 0.888 | 0.855 | 0.838 | 0.874 |

| | HITS@20 | | | | | | |
|---|---|---|---|---|---|---|---|
| | | AIDS | Mutag | FM | FR | MM | MR |
| **Multi-Layer** | Node partner (with additional MLP) | 0.479 | 0.634 | 0.677 | 0.611 | 0.608 | 0.64 |
| | Node pair partner (msg only) | 0.577 | 0.651 | 0.775 | 0.682 | 0.719 | 0.703 |
| | Node partner | 0.433 | 0.639 | 0.678 | 0.58 | 0.571 | 0.624 |
| | EINSMATCH (Node) | 0.57 | 0.672 | 0.744 | 0.657 | 0.68 | 0.707 |
| **Multi-Round** | Node partner (with additional MLP) | 0.658 | 0.727 | 0.756 | 0.738 | 0.667 | 0.743 |
| | Node pair partner (msg only) | 0.671 | 0.717 | 0.807 | 0.696 | 0.728 | 0.753 |
| | Node partner | 0.603 | 0.702 | 0.736 | 0.686 | 0.721 | 0.695 |
| | EINSMATCH (Node) | 0.672 | 0.732 | 0.797 | 0.737 | 0.702 | 0.755 |

| | Mean Reciprocal Rank (MRR) | | | | | | |
|---|---|---|---|---|---|---|---|
| | | AIDS | Mutag | FM | FR | MM | MR |
| **Multi-Layer** | Node partner (with additional MLP) | 0.909 | 0.941 | 0.965 | 0.964 | 0.966 | 0.984 |
| | Node pair partner (msg only) | 0.97 | 0.956 | 0.964 | 0.993 | 0.978 | 1.0 |
| | Node partner | 0.917 | 0.945 | 0.964 | 0.987 | 0.958 | 0.969 |
| | EINSMATCH (Node) | 0.956 | 0.954 | 1.0 | 0.978 | 0.98 | 1.0 |
| **Multi-Round** | Node partner (with additional MLP) | 0.987 | 0.944 | 0.993 | 0.987 | 0.963 | 0.983 |
| | Node pair partner (msg only) | 0.984 | 0.958 | 0.993 | 0.98 | 0.984 | 0.984 |
| | Node partner | 0.984 | 0.949 | 0.993 | 0.978 | 0.978 | 0.97 |
| | EINSMATCH (Node) | 0.993 | 0.971 | 1.0 | 0.993 | 0.993 | 0.993 |

| | Precision@20 | | | | | | |
|---|---|---|---|---|---|---|---|
| | | AIDS | Mutag | FM | FR | MM | MR |
| **Multi-Layer** | Node partner (with additional MLP) | 0.817 | 0.867 | 0.913 | 0.913 | 0.883 | 0.914 |
| | Node pair partner (msg only) | 0.871 | 0.886 | 0.957 | 0.937 | 0.927 | 0.937 |
| | Node partner | 0.799 | 0.866 | 0.919 | 0.877 | 0.873 | 0.885 |
| | EINSMATCH (Node) | 0.873 | 0.897 | 0.935 | 0.917 | 0.93 | 0.931 |
| **Multi-Round** | Node partner (with additional MLP) | 0.921 | 0.917 | 0.936 | 0.951 | 0.921 | 0.945 |
| | Node pair partner (msg only) | 0.923 | 0.913 | 0.969 | 0.951 | 0.957 | 0.957 |
| | Node partner | 0.875 | 0.921 | 0.933 | 0.942 | 0.939 | 0.941 |
| | EINSMATCH (Node) | 0.932 | 0.943 | 0.957 | 0.961 | 0.949 | 0.963 |

Table 15: Effect of node pair partner interaction in EINSMATCH (Node). Table shows the comparison of EINSMATCH (Node) with three different alternatives. The first table reports MAP values, second reports HITS@20, third reports MRR and fourth reports Precision@20. In each table, the first two rows report metrics for multi-layer refinement and the second two rows report metrics for multi-round refinement. Rows colored green and yellow indicate the best and second best methods in their respective sections.

| | | Mean Average Precision (MAP) | | | | | |
|---|---|---|---|---|---|---|---|
| | | AIDS | Mutag | FM | FR | MM | MR |
| **Multi-Layer** | GMN | 0.622 | 0.71 | 0.73 | 0.662 | 0.655 | 0.708 |
| | Node pair partner | 0.579 | 0.732 | 0.74 | 0.677 | 0.641 | 0.713 |
| **Multi-Round** | GMN | 0.629 | 0.699 | 0.757 | 0.697 | 0.653 | 0.714 |
| | Node pair partner | 0.579 | 0.693 | 0.729 | 0.69 | 0.665 | 0.705 |

| | | HITS@20 | | | | | |
|---|---|---|---|---|---|---|---|
| | | AIDS | Mutag | FM | FR | MM | MR |
| **Multi-Layer** | GMN | 0.397 | 0.544 | 0.537 | 0.45 | 0.423 | 0.49 |
| | Node pair partner | 0.346 | 0.567 | 0.551 | 0.476 | 0.411 | 0.5 |
| **Multi-Round** | GMN | 0.403 | 0.533 | 0.562 | 0.494 | 0.431 | 0.502 |
| | Node pair partner | 0.344 | 0.528 | 0.54 | 0.502 | 0.462 | 0.506 |

| | | Mean Reciprocal Rank (MRR) | | | | | |
|---|---|---|---|---|---|---|---|
| | | AIDS | Mutag | FM | FR | MM | MR |
| **Multi-Layer** | GMN | 0.877 | 0.923 | 0.949 | 0.947 | 0.928 | 0.922 |
| | Node pair partner | 0.827 | 0.897 | 0.958 | 0.877 | 0.918 | 0.92 |
| **Multi-Round** | GMN | 0.905 | 0.862 | 0.958 | 0.956 | 0.906 | 0.921 |
| | Node pair partner | 0.811 | 0.901 | 0.907 | 0.908 | 0.964 | 0.92 |

| | | Precision@20 | | | | | |
|---|---|---|---|---|---|---|---|
| | | AIDS | Mutag | FM | FR | MM | MR |
| **Multi-Layer** | GMN | 0.751 | 0.82 | 0.852 | 0.809 | 0.783 | 0.832 |
| | Node pair partner | 0.7 | 0.833 | 0.861 | 0.797 | 0.792 | 0.846 |
| **Multi-Round** | GMN | 0.753 | 0.795 | 0.885 | 0.829 | 0.792 | 0.842 |
| | Node pair partner | 0.694 | 0.794 | 0.847 | 0.835 | 0.802 | 0.825 |

Table 16: Effect of node pair partner interaction in GMN. The tables compare GMN with its EINSMATCH alternative. The first table reports MAP values, the second table reports HITS@20 values, the third table reports MRR values and the fourth table reports Precision@20. In each table, the first two rows report metrics for multi-layer refinement and the second two rows report metrics for multi-round refinement. Rows colored green and yellow indicate the best and second best methods according to the respective metrics.

### G.6 Variation of EINSMATCH (Node) and EINSMATCH (Edge) with different $T$ and $K$

In this section, we analyze the accuracy and inference time trade-off of multi-round lazy and multi-layer eager variants of EINSMATCH (Node) and EINSMATCH (Edge). In the following tables, we show the MAP and inference time. Additionally, we also analyze the trade-off of GMN and IsoNet (Edge). The $T, K$ parameters for different models are so chosen that they can be compared against each other while fixing the inference time to be roughly similar. For instance, multi-round lazy EINSMATCH (Node) with $T = 5, K = 5$ maps to multi-layer eager EINSMATCH (Node) with $K = 8$, allowing for a direct comparison of performance without caring much about different compute. Note that in below tables, models are listed in order of increasing inference time (i.e. increasing $K$ or $T$).

In tables 18 and 19, we show variations for multi-round lazy EINSMATCH (Node) for fixed $T$ and fixed $K$ respectively. We observe that with fixed $T$, increasing $K$ from 5 to 10 doesn't improve the model significantly. For fixed $K$, performance (in terms of MAP) improves notably when increasing $T$ from 3 to 5.

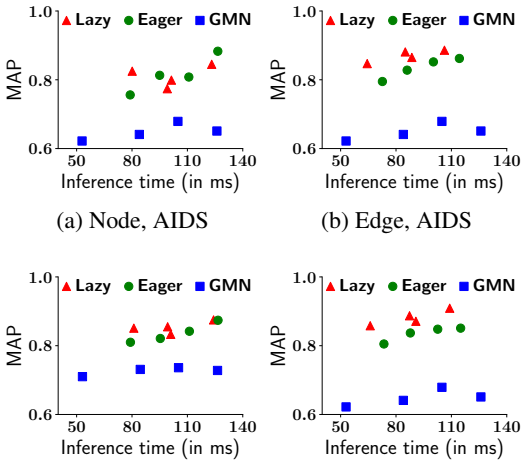
In table 20, we show variations for multi-layer eager EINSMATCH (Node) for varying $K$. We observe that except for a drop at $K = 7$, the performance of the model improves as we increase $K$. In fact, at $K = 8$, the performance is surprisingly good, even outperforming the similarly timed $T = 5, K = 5$ variant of lazy multi-round EINSMATCH (Node) on both AIDS and Mutag.

In tables 21 and 22, we compare variants of multi-round lazy EINSMATCH (Edge) with fixed $T$ and fixed $K$ respectively. We observe that when $T$ is fixed and $K$ is increased, the gain is marginal. We observe a significant gain When $K$ is fixed and $T$ is increased from 3 to 4.

In table 23, we study the trade-off for multi-layer eager EINSMATCH (Edge) for varying $K$. We observe that with increasing $K$, the performance continues to improve and peaks at $K = 8$. Note that even at this $K$, the performance of multi-layer eager EINSMATCH (Edge) is worse than a similarly timed variant ($T = 5, K = 5$) of multi-round EINSMATCH (Edge).

In table 24, we show variations for GMN for varying $K$. We observe marginal gains while increasing $K$. From $K = 10$ to $K = 12$, the performance drops.

In table 25, we show how performance varies for IsoNet (Edge) for varying $K$. We observe that the model does not improve with increasing $K$.




(a) Node, AIDS (b) Edge, AIDS

(c) Node, Mutag (d) Edge, Mutag

Figure 17: Trade off between MAP and inference time (batch size=128).

| Mean Average Precision (MAP) | | | | | | |
|---|---|---|---|---|---|---|
| | AIDS | Mutag | FM | FR | MM | MR |
| $T = 3, K = 5$ | 0.825 | 0.851 | 0.888 | 0.855 | 0.838 | 0.874 |
| $T = 3, K = 10$ | 0.774 | 0.855 | 0.898 | 0.811 | 0.855 | 0.882 |

| Inference time (in ms) | | | | | | |
|---|---|---|---|---|---|---|
| | AIDS | Mutag | FM | FR | MM | MR |
| $T = 3, K = 5$ | 80.11 | 80.99 | 81.01 | 81.24 | 80.94 | 80.25 |
| $T = 3, K = 10$ | 99.11 | 99.31 | 99.28 | 99.48 | 99.37 | 99.36 |

Table 18: MAP and inference time trade-off of variants of multi-round lazy EINSMATCH (Node) with fixed $T$. Rows colored green indicate the best $K$ according to the MAP score.

| Mean Average Precision (MAP) | | | | | | |
|---|---|---|---|---|---|---|
| | AIDS | Mutag | FM | FR | MM | MR |
| $T = 3, K = 5$ | 0.825 | 0.851 | 0.888 | 0.855 | 0.838 | 0.874 |
| $T = 4, K = 5$ | 0.799 | 0.833 | 0.892 | 0.858 | 0.867 | 0.891 |
| $T = 5, K = 5$ | 0.845 | 0.875 | 0.919 | 0.883 | 0.894 | 0.897 |

| Inference time (in ms) | | | | | | |
|---|---|---|---|---|---|---|
| | AIDS | Mutag | FM | FR | MM | MR |
| $T = 3, K = 5$ | 80.11 | 80.99 | 81.01 | 81.24 | 80.94 | 80.25 |
| $T = 4, K = 5$ | 101.33 | 100.99 | 100.95 | 100.46 | 100.59 | 100.87 |
| $T = 5, K = 5$ | 123.18 | 124.19 | 123.61 | 122.79 | 123.33 | 122.74 |

Table 19: MAP and inference time trade-off of variants of multi-round lazy EINSMATCH (Node) with fixed $K$. Rows colored green and yellow indicate the best and second best $T$ according to the MAP score.

| Mean Average Precision (MAP) | | |
|---|---|---|
| | AIDS | Mutag |
| $K = 5$ | 0.756 | 0.81 |
| $K = 6$ | 0.813 | 0.821 |
| $K = 7$ | 0.808 | 0.842 |
| $K = 8$ | 0.883 | 0.874 |

| Inference time (in ms) | | |
|---|---|---|
| | AIDS | Mutag |
| $K = 5$ | 79.02 | 79.15 |
| $K = 6$ | 94.99 | 95.33 |
| $K = 7$ | 110.78 | 111.09 |
| $K = 8$ | 126.48 | 126.6 |

Table 20: MAP and inference time trade-off of variants of multi-layer eager EINSMATCH (Node) with increasing $K$. Rows colored green and yellow indicate the best and second best $K$ according to the MAP score.

| Mean Average Precision (MAP) | | |
|---|---|---|
| | AIDS | Mutag |
| $T = 3, K = 5$ | 0.847 | 0.858 |
| $T = 3, K = 10$ | 0.865 | 0.871 |

| Inference time (in ms) | | |
|---|---|---|
| | AIDS | Mutag |
| $T = 3, K = 5$ | 64.39 | 66.03 |
| $T = 3, K = 10$ | 88.59 | 90.76 |

Table 21: MAP and inference time trade-off of variants of multi-round lazy EINSMATCH (Edge) with fixed $T$. Rows colored green indicate the best $K$ according to the MAP score.

| Mean Average Precision (MAP) | | |
|---|---|---|
| | AIDS | Mutag |
| $T = 3, K = 5$ | 0.847 | 0.858 |
| $T = 4, K = 5$ | 0.881 | 0.887 |
| $T = 5, K = 5$ | 0.886 | 0.909 |

| Inference time (in ms) | | |
|---|---|---|
| | AIDS | Mutag |
| $T = 3, K = 5$ | 64.39 | 66.03 |
| $T = 4, K = 5$ | 85.02 | 87.33 |
| $T = 5, K = 5$ | 106.24 | 109.1 |

Table 22: MAP and inference time trade-off of variants of multi-round lazy EINSMATCH (Edge) with fixed $K$. Rows colored green and yellow indicate the best and second best $T$ according to the MAP score.

| Mean Average Precision (MAP) | | |
|---|---|---|
| | AIDS | Mutag |
| $K = 5$ | 0.795 | 0.805 |
| $K = 6$ | 0.828 | 0.837 |
| $K = 7$ | 0.852 | 0.848 |
| $K = 8$ | 0.862 | 0.851 |

| Inference time (in ms) | | |
|---|---|---|
| | AIDS | Mutag |
| $K = 5$ | 72.63 | 73.46 |
| $K = 6$ | 86.03 | 87.77 |
| $K = 7$ | 100.26 | 102.6 |
| $K = 8$ | 114.33 | 115.01 |

Table 23: MAP and inference time trade-off of variants of multi-layer eager EINSMATCH (Edge) with increasing $K$. Rows colored green and yellow indicate the best and second best $K$ according to the MAP score.

| Mean Average Precision (MAP) | | | | | | |
|---|---|---|---|---|---|---|
| | AIDS | Mutag | FM | FR | MM | MR |
| $K = 5$ | 0.622 | 0.710 | 0.730 | 0.662 | 0.655 | 0.708 |
| $K = 8$ | 0.641 | 0.731 | 0.745 | 0.701 | 0.658 | 0.711 |
| $K = 10$ | 0.679 | 0.736 | 0.741 | 0.712 | 0.691 | 0.74 |
| $K = 12$ | 0.651 | 0.728 | 0.743 | 0.697 | 0.687 | 0.699 |

| Inference time (in ms) | | | | | | |
|---|---|---|---|---|---|---|
| | AIDS | Mutag | FM | FR | MM | MR |
| $K = 5$ | 52.94 | 53.16 | 53.23 | 53.12 | 53.32 | 53.34 |
| $K = 8$ | 83.97 | 84.47 | 84.64 | 84.38 | 85.41 | 84.51 |
| $K = 10$ | 104.87 | 105.21 | 105.72 | 105.33 | 105.66 | 105.73 |
| $K = 12$ | 125.99 | 126.33 | 126.53 | 126.39 | 126.79 | 126.59 |

Table 24: MAP and inference time trade-off of variants of GMN with increasing $K$. Rows colored green and yellow indicate the best and second best $K$ according to the MAP score.

| | AIDS | Inference time (in ms) |
|---|---|---|
| $K = 5$ | 0.69 | 19.77 |
| $K = 6$ | 0.717 | 20.83 |
| $K = 7$ | 0.697 | 21.96 |
| $K = 8$ | 0.709 | 23.02 |

Table 25: MAP and inference time trade-off of variants of IsoNet (Edge) with increasing $K$. Rows colored green and yellow indicate the best and second best $T$ according to the MAP score.

# NeurIPS Paper Checklist

1. **Claims**
   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?
   Answer: [Yes]
   Justification: Section 1.1 discusses the paper's contributions.
   Guidelines:
   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.
2. **Limitations**
   Question: Does the paper discuss the limitations of the work performed by the authors?
   Answer: [Yes]
   Justification: Appendix A discusses the limitations of our work. Details about computational efficiency are included in the main paper as well as in Appendix G.6, expressed explicitly as the running time of each approach.
   Guidelines:
   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.
3. **Theory Assumptions and Proofs**
   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?
   Answer: [Yes]
   Justification: The paper includes one theorem, which is noted and proved in Appendix D.5.
   Guidelines:
   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.

- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper introduces new architectures and the designs of each of these are discussed in the main paper under Sections 3.2, 3.3 and Appendices D, E. Hyperparameters, training procedure, hardware and random seeds for all experiments are noted in Appendix F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The anonymized code is made available as a supplementary material with the submission.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Important parameters that are required to understand and appreciate the results are noted as and when required. Hyperparameters, training procedure, hardware and random seeds for all experiments are noted in Appendix F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Appendix G.1 includes standard error for the metrics reported in the paper, which includes Mean Average Precision (MAP) and HITS@20, computed across each query graph in the test split. The section also reports the method of calculation of the standard error.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**
   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?
   Answer: [Yes]
   Justification: Types of GPUs and inference time are reported in Appendices F.5 and G.6 respectively. Other details about the training setting are mentioned point-wise in Appendix F.
   Guidelines:
   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).
9. **Code Of Ethics**
   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?
   Answer: [Yes]
   Justification: The authors have studied the ethics guidelines and find the work to conform well to them.
   Guidelines:
   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).
10. **Broader Impacts**
    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?
    Answer: [Yes]
    Justification: Broader societal impacts (both positive and negative) are discussed in Appendix C.
    Guidelines:
    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).
11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?
Answer: [NA]
Justification: The datasets used in the paper are derived from public graph datasets widely used by researchers and the authors are not aware of any risks for misuse posed by them.
Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**
Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?
Answer: [Yes]
Justification: The related work and datasets applicable to this work have been cited in Appendices B and F.1 respectively. Best effort was made by the authors to determine the licenses of the datasets and existing architectures, and are included in Appendix F.6.
Guidelines:
    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.
13. **New Assets**
Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?
Answer: [Yes]
Justification: Details of the models and datasets are discussed in Appendices D and F.1 respectively. Datasets used in the paper are publicly accessible. The code is submitted with the paper as a supplementary item and is anonymized & well-documented.
Guidelines:
    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.
14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The experiments performed in this paper do not involve human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The experiments performed in this paper do not involve human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.