Refining Network Alignment to Improve Matched Neighborhood Consistency

Mark Heimann*† Xiyuan Chen*‡ Fatemeh Vahedian* Danai Koutra*

Abstract

Network alignment, or the task of finding meaningful node correspondences between nodes in different graphs, is an important graph mining task with many scientific and industrial applications. An important principle for network alignment is *matched neighborhood consistency* (MNC): nodes that are close in one graph should be matched to nodes that are close in the other graph. We theoretically demonstrate a close relationship between MNC and alignment accuracy. As many existing network alignment methods struggle to preserve topological consistency in difficult scenarios, we show how to refine their solutions by improving their MNC. Our refinement method, RefiNA, is straightforward to implement, admits scalable sparse approximation, and can be paired *post hoc* with *any* network alignment method. Extensive experiments show that RefiNA increases the accuracy of diverse unsupervised network alignment methods by up to 90%, making them robust enough to align graphs that are 5× more topologically different than were considered in prior work.

1 Introduction

Network alignment, or the task of finding correspondences between the nodes of multiple networks, is a fundamental graph mining task with applications to user identity linkage [20], discovery of novel biological functions [15], computer vision [10], schema matching in databases [21], and other problems of academic and industrial interest. Methods to solve this problem vary widely in techniques, ranging from nonlinear relaxations of a computationally hard optimization problem [1], to belief propagation [2], genetic algorithms [25], spectral methods [26], node embedding [13], and more.

Matching the topological structure of graphs is difficult, closely related to the canonical graph isomorphism problem [1]. As a result, many network alignment approaches rely heavily on node or edge side information [32, 22] (in some cases ignoring the graph structure altogether), or on known ground-truth alignments—used as anchor links to connect the two graphs [20, 34] or to supervise the training of deep neural networks [9]. However, in many settings [13, 5, 8], side information or anchor links may not be available.

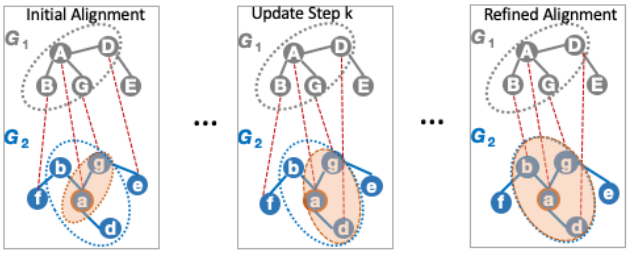


Figure 1: RefiNA refines an initial network alignment solution, which maps node A and its neighbors in G_1 far apart in G_2. The refined network alignment solution has higher *matched neighborhood consistency*: neighbors of A are aligned to neighbors of a, to which A itself is aligned.

In this paper, we focus on the challenging problem of unsupervised topological network alignment. With neither anchor links to seed the alignment process nor side information to guide it, the main objective for this task is to preserve some kind of topological consistency in the alignment solution. We theoretically analyze the principle of *matched neighborhood consistency* (MNC), or how well a node's neighborhood maps onto the neighborhood of its counterpart in the other graph (illustrated in Fig. 1), and show its connection to alignment accuracy. On the other hand, we find that when network alignment methods are inaccurate, the MNC of their solutions breaks down (e.g., Fig. 1 left).

To address this, we introduce RefiNA, a method for refining network alignment solutions *post hoc* by iteratively updating nodes' correspondences to improve their MNC. By strategically limiting the possible correspondences per node to update in each iteration, we can sparsify the computations to make RefiNA scalable to large graphs. Experimentally, we show that RefiNA signifi-

*Computer Science & Engineering, University of Michigan. Email: {mheimann, shinech, vfatemeh, dkoutra}@umich.edu
†Now at Lawrence Livermore National Laboratory.
‡Now at Stanford University.

cantly improves a variety of network alignment methods on many highly challenging datasets, even when starting a network alignment solution with limited accuracy. Also, it can be succinctly expressed as matrix operations in a few lines of code, making it easy for practitioners to adopt. In this compact formulation, we incorporate several useful insights for network alignment, and our techniques have interesting connections to other successful graph-based methods.

Our contributions are as follows:

- **New Algorithm**: We propose RefiNA, a post-processing step that can be applied to the output of any network alignment method. Its compact design incorporates several important insights for network alignment, and it permits a sparse approximation that is scalable to large graphs.

- **Theoretical Connections**: We show a rigorous connection between *matched neighborhood consistency*, the property that RefiNA improves, and alignment accuracy. We also provide network alignment insights justifying each of RefiNA's design choices and technical connections to other graph-based methods.

- **Experiments**: We conduct thorough experiments on real and simulated network alignment tasks and show that RefiNA improves the accuracy of many methodologically diverse network alignment methods by up to 90%, making them robust enough to recover matchings in 5× noisier datasets than those considered in prior work. We extensively drill down RefiNA to justify the insights that inspire its design.

We provide our code and additional supplementary material at `https://github.com/GemsLab/RefiNA`.

2 Related Work

Network alignment has numerous applications from matching schemas in databases [21] and objects in images [10], to revealing biological functions shared by different organisms [15], to integration of multiple data sources to create a holistic worldview network [6]. as been widely studied and several methods have been proposed to solve this problem. For example, the intuition of NetAlign [2] as a message-passing algorithm is to "complete squares" by aligning two nodes that share an edge in one graph to two nodes that share an edge in another graph. Similarly, FINAL [32] has an objective of preserving topological consistency between the graphs that may be augmented with node and edge attribute information, if available. MAGNA [25] is a genetic algorithm that can evolve network populations to maximize topological consistency. Recent works leverage kernel methods [35] or optimal transport [31] but suffer from high (cubic) computational complexity.

Networks can also be aligned by comparing nodes directly. Early works hand-engineered node features: GRAAL [15] computes a graphlet degree signature, while GHOST [23] defines a multiscale spectral signature. UniAlign [14] and HashAlign [11] extract features for each node from graph statistics such as degree and various node centralities. Recent works instead leverage node embeddings that are very expressive, but must be comparable across networks [13]. REGAL [13] computes node embeddings that capture structural roles that are not specific to a particular network. Other methods align the embedding spaces of different networks using techniques from unsupervised machine translation: DANA [5] uses adversarial training [17], and CONE-Align [4] uses non-convex alternating optimization methods. All these approaches find match nodes with embedding-based nearest-neighbor search, which does not enforce alignment consistency and can benefit from our refinement. Our contributions are orthogonal to a recent effort to accelerate embedding-based node matching via graph compression [24].

Some graph neural matching network models, often in specific applications like social networks [19], computer vision [9], or knowledge graphs [30], use objectives that enforce matching consistency between neighboring nodes. This is similar to our approach RefiNA; indeed, in the supplementary §B we show interesting technical connections between RefiNA and graph neural networks. However, RefiNA, like Simple Graph Convolution [29] is much faster and simpler to implement and apply than these graph neural networks, and it does not need known node matchings to supervise training.

3 Theoretical Analysis

We first introduce key alignment concepts and notation. Then, we theoretically justify the topological consistency principle that is the basis of our refinement approach, RefiNA (§4).

3.1 Preliminaries

3.1.1 Graphs. Following the network alignment literature [13, 25], we consider two unweighted and undirected graphs $G_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2)$ with their corresponding nodesets $\mathcal{V}_1, \mathcal{V}_2$ and edgesets $\mathcal{E}_1, \mathcal{E}_2$. We denote their adjacency matrices as $\mathbf{A}_1$ and $\mathbf{A}_2$. Since they are symmetric, $\mathbf{A}_1^\top = \mathbf{A}_1$ and $\mathbf{A}_2^\top = \mathbf{A}_2$, and we simplify our notation below.

3.1.2 Alignment. A node alignment is a function $\pi : \mathcal{V}_1 \to \mathcal{V}_2$ that maps the nodes of G_1 to those of

Table 1: Major symbols and definitions.

Symbols	Definitions
$G_\ell = (\mathcal{V}_\ell, \mathcal{E}_\ell)$	ℓ^{th} graph with nodeset $\mathcal{V}_\ell$, edgeset $\mathcal{E}_\ell$
$\mathbf{A}_\ell$	Adjacency matrix of G_ℓ
$n_\ell, \bar{d}^\ell$	Number of nodes and average degree in G_ℓ
$\pi(\cdot)$	An alignment between graphs G_1 and G_2; a function mapping a node in $\mathcal{V}_1$ to a node in $\mathcal{V}_2$
$\mathbf{M}$	$n_1 \times n_2$ matrix specifying correspondences of nodes in $\mathcal{V}_1$ to those in $\mathcal{V}_2$
$\mathcal{N}_{G_\ell}(u)$	Neighbors of node i in graph G_ℓ
$\tilde{\mathcal{N}}^\pi_{G_2}(u)$	"Mapped neighborhood" of node i; counterparts in G_2 (mapped by π) of nodes in $\mathcal{N}_{G_1}(i)$

G_2. It is also commonly represented as a $|\mathcal{V}_1| \times |\mathcal{V}_2|$ alignment matrix $\mathbf{M}$, where $\mathbf{M}_{ij}$ is the (real-valued or binary) similarity between node i in G_1 and node j in G_2. $\mathbf{M}$ can be used to encode a mapping π, e.g., greedy alignment $\pi(i) = \arg\max_j \mathbf{M}_{ij}$. We note that alignment between two graphs should be sought *if* the nodes of the two graphs meaningfully correspond.

3.1.3 Neighborhood and Consistency. Let $\mathcal{N}_{G_1}(i) = \{j \in \mathcal{V}_1 : (i,j) \in \mathcal{E}_1\}$ be the neighbors of node i in G_1, i.e., the set of all nodes with which i shares an edge. We define node i's "**mapped neighborhood**" in G_2 as the set of nodes onto which π maps i's neighbors: $\tilde{\mathcal{N}}^\pi_{G_2}(i) = \{j \in \mathcal{V}_2 : \exists k \in \mathcal{N}_{G_1}(i) \text{ s.t. } \pi(k) = j\}$. For example, in Fig. 1 (first panel), node A's neighbors in G_1 are $B, G,$ and D, which are respectively mapped to nodes $f, g,$ and e, so $\tilde{\mathcal{N}}^\pi_{G_2}(A) = \{f, g, e\}$.

We call the neighbors of i's counterpart $\mathcal{N}_{G_2}\big(\pi(i)\big)$. In Fig. 1, node A's counterpart is node a, whose neighbors are $b, g,$ and d. Thus, $\mathcal{N}_{G_2}\big(\pi(A)\big) = \{b, g, d\}$.

The **matched neighborhood consistency** (MNC) [4] of node i in G_1 and node j in G_2 is the Jaccard similarity of the sets $\tilde{\mathcal{N}}^\pi_{G_2}(i)$ and $\mathcal{N}_{G_2}(j)$:

$$\text{MNC}(i,j) = \frac{|\tilde{\mathcal{N}}^\pi_{G_2}(i) \cap \mathcal{N}_{G_2}(j)|}{|\tilde{\mathcal{N}}^\pi_{G_2}(i) \cup \mathcal{N}_{G_2}(j)|}. \tag{3.1}$$

3.2 Theoretical Justification of MNC. Several unsupervised network alignment algorithms attempt to enforce some notion of topological consistency in their objective functions (§2). We justify this intuition by showing that a specific form of topological consistency, matched neighborhood consistency or MNC, has a close relationship with alignment accuracy.

Unsupervised network alignment is commonly evaluated on graphs that are isomorphic up to noisy or missing edges [13, 5, 25, 14, 32, 33]. When edges are removed from one graph independently with probability p, we show that accurate alignment entails high MNC.

THEOREM 3.1. *For isomorphic graphs $G_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2)$, let $\pi(\cdot)$ be the isomorphism. Let $\overline{G}_2 = (\mathcal{V}_2, \overline{\mathcal{E}}_2)$ be a noisy version of G_2 created by removing each edge from $\mathcal{E}_2$ independently with probability p. Then for any node i in G_1 and its counterpart $\pi(i)$ in $\overline{G}_2$, $\mathbb{E}\big(MNC(i, \pi(i))\big) = 1 - p$.*

However, this does not prove that a solution with perfect MNC will have perfect accuracy. In fact, we can construct counterexamples, such as two "star" graphs, each consisting of one central node connected to $n-1$ peripheral nodes (of degree one). Whatever the true correspondence of the peripheral nodes, aligning them to each other in any order would lead to perfect MNC. Prior network alignment work [14] has observed a few such special cases and in fact gives up on trying to distinguish them from the graph topology. We formalize this concept of *structural indistinguishability*:

DEFINITION 1. *Let $\mathcal{N}_k(u)$ be the subgraph induced by all nodes that are k or fewer hops/steps away from node u. Two nodes u and v are structurally indistinguishable if for all k, $\mathcal{N}_k(u)$ and $\mathcal{N}_k(v)$ are isomorphic.*

Our next result proves that for isomorphic graphs, structurally indistinguishable nodes are the only possible failure case for a solution with perfect MNC.

THEOREM 3.2. *For isomorphic graphs $G_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2)$, suppose there exists $\pi(\cdot)$ that yields MNC = 1 for all nodes. Then, if π misaligns a node v to some node v^* instead of the true counterpart v', it is because v^* is structurally indistinguishable from v'.*

Proof idea. We give all the proofs in the supplementary §A. At a high level, MNC measures the extent to which the alignment preserves edges in a node's neighborhood, and isomorphic graphs have a (not necessarily unique) perfectly edge-preserving node matching.

Formalizing the intuitions of prior work: MNC was proposed as *intuition* for modeling intra-graph node proximities when performing cross-graph matching [4]. It was also used (not by that name) as a *heuristic* in embedding-based network alignment [8]. Our analysis provides theoretical justification for both works.

4 Method

We consider an unsupervised network alignment setting, where an initial solution, $\mathbf{M}_0$, is provided by any network alignment method (§2). We do *not* take any of these initial node alignments as ground truth; on the contrary, we seek to improve their correctness by leveraging insights from our theory in §3.2. Thus, our problem differs from semi-supervised network alignment that is seeded with known ground-truth node correspondences [20, 34]. Formally, we state it as:

PROBLEM 1. *Given a sparse initial alignment matrix* $\mathbf{M}_0$ *between the nodes of two graphs* G_1 *and* G_2*, we seek to* refine *this initial solution into a new, real-valued matrix* $\mathbf{M}$ *of refined similarity scores that encodes a more accurate alignment.*

4.1 RefiNA and Connections to MNC. Our theoretical results pave a path to solving Problem 1 by increasing matched neighborhood consistency. While our results characterize "ideal" cases (perfect accuracy or MNC), *heuristic* solutions in prior works have found that increasing MNC tends to increase accuracy [4, 8]. Given an alignment matrix returned by a network alignment method, how can we improve its MNC in a *principled* way? We first derive a matrix-based form for MNC, which we prove in the supplementary §A:

THEOREM 4.1. *The MNC of a binary alignment matrix* $\mathbf{M}$ *can be written as a matrix* $\mathbf{S}^{MNC}$ *such that* $MNC(i,j) = \mathbf{S}_{ij}^{MNC}$ *as:*

$$\mathbf{S}^{MNC} = \mathbf{A}_1\mathbf{M}\mathbf{A}_2 \oslash (\mathbf{A}_1\mathbf{M}\mathbf{1}^{n_2} \otimes \mathbf{1}^{n_2} + \mathbf{1}^{n_1} \otimes \mathbf{A}_2\mathbf{1}^{n_2} - \mathbf{A}_1\mathbf{M}\mathbf{A}_2) \tag{4.2}$$

where $\oslash$ *is elementwise division and* $\otimes$ *is outer product.*

We can then compute refined alignments $\mathbf{M}'$ by multiplicative updating each node's alignment score (in $\mathbf{M}$) with its matched neighborhood consistency:

$$\mathbf{M}' = \mathbf{M} \circ \mathbf{S}^{\text{MNC}} \tag{4.3}$$

where $\circ$ denotes Hadamard product. Repeating over several iterations can take advantage of an improving alignment solution. The high-level idea of our proposed refinement scheme is to **iteratively increase alignment scores for nodes that have high MNC**.

While we could just repeatedly iterate Eq. (4.3), we can introduce some mechanisms leverage important insights without increasing complexity:

• `I1`**: Prioritize high-degree nodes**. Higher degree nodes are easier to align [14], and so it is desirable to give them higher scores particularly in early iterations. To do so, we use only the (elementwise) numerator of Eq. (4.1), which counts nodes' number of matched neighbors (excluding the normalizing denominator increases the score for high degree nodes). Thus, instead of using Eq. (4.3), we simplify our update rule to:

$$\mathbf{M}' = \mathbf{M} \circ \mathbf{A}_1\mathbf{M}\mathbf{A}_2 \tag{4.4}$$

Additionally, this spares us the extra matrix operations required to compute the denominator of Eq. (4.1).

• `I2`**: Do not overly rely on the initial solution**. At every iteration, we add a small ϵ to every element of $\mathbf{M}$. This gives each pair of nodes a token match score whether or not the initial alignment algorithm identified them as matches, which gives us a chance to correct the initial solution's false negatives.

• `I3`**: Allow convergence.** Finally, to keep the scale of the values of $\mathbf{M}$ from exploding, we we row-normalize $\mathbf{M}$ followed by column-normalizing it at every iteration. Previous iterative graph matching methods such as graduated assignment [10] require full normalization per iteration using the Sinkhorn algorithm [27] to produce a doubly stochastic matrix. In contrast, with RefiNA a single round of normalization suffices, avoiding large computational expense (§5).

Putting it all together, we give the pseudocode of our method RefiNA in Algorithm 1. RefiNA is powerful yet conceptually simple and straightforward to implement. It requires only a few lines of code, with each line implementing an important insight.

Algorithm 1 RefiNA $(\mathbf{A}_1, \mathbf{A}_2, \mathbf{M}_0, K, \epsilon)$

Input: adjacency matrices $\mathbf{A}_1, \mathbf{A}_2$, initial alignment matrix $\mathbf{M}_0$, number of iterations K, token match score ϵ
for $k = 1 \to K$ **do** ▷ Refinement iterations
  $\mathbf{M}_k = \mathbf{M}_{k-1} \circ \mathbf{A}_1\mathbf{M}_{k-1}\mathbf{A}_2$ ▷ MNC update
  $\mathbf{M}_k = \mathbf{M}_k + \epsilon$ ▷ Add token match scores
  $\mathbf{M}_k = \text{Normalize}(\mathbf{M}_k)$ ▷ By row then column
end for
return $\mathbf{M}_K$

4.2 Connections to Other Graph Problems. In the supplementary §B, we further justify RefiNA conceptually by comparing and contrasting it to several other graph techniques. We find that RefiNA's update can be viewed as a more flexible version of the seed-and-extend node matching heuristic [15, 23], and that it performs a graph filtering operation akin to a graph neural network [29]. In particular, a similar neighborhood aggregation operation is used by graph neural tangent kernels [7] for graph-level comparison. This further highlights the connection between the individual node similarities found in graph matching [32, 13] and aggregated node similarities in graph kernels [32, 12].

4.3 Optimizations: Sparse RefiNA. The dense matrix updates lead to quadratic computational complexity in the number of nodes. To scale RefiNA to large graphs, we sparsify it by updating only a small number of alignment scores for each node. Intuitively, we forgo updating scores of likely non-aligned node pairs (these updates would be small anyway). Concretely, sparse

RefiNA replaces Line 3 of Algorithm 1 with

$$\mathbf{M}_{k|\mathbf{U}_k} = \mathbf{M}_{k-1|\mathbf{U}_k} \circ \mathbf{U}_k$$

where the update matrix $\mathbf{U}_k = \text{top-}\alpha(\mathbf{A}_1\mathbf{M}\mathbf{A}_2)$ is a sparse version of $\mathbf{A}_1\mathbf{M}\mathbf{A}_2$ containing only the largest α entries per row. $\mathbf{M}_{k|\mathbf{U}_k}$ selects the elements of $\mathbf{M}_k$ (pairs of nodes) corresponding to nonzero elements in $\mathbf{U}_k$. These are the only elements on which we perform an MNC-based update, and the only ones to receive a token match score (Line 4): $\mathbf{M}_{k|\mathbf{U}_k} = \mathbf{M}_{k|\mathbf{U}_k} + \epsilon$.

As the elements to update are selected by the size of their update scores, which are computed using the previous solution $\mathbf{M}$, sparse RefiNA relies somewhat more strongly on the initial solution. However, we still comply with `I2` by updating $\alpha > 1$ possible alignments for each node. This has sub-quadratic time and space requirements (cf. supplemental §C) and accuracy comparable to dense updates (§ 5.4).

4.4 Assumption and Limitations. Network alignment typically relies on some kind of topological consistency assumption (§2). Likewise, RefiNA assumes that improving alignment MNC will improve its quality. Our theory (§3.2) shows this assumption is well-founded for the quasi-isomorphic graphs studied by many prior works [13, 5, 25, 14, 32, 33], and our experiments (§5) support RefiNA's efficacy in even more use cases. However, all assumptions have limitations, so we discuss two limitations of ours. First, as Theorem 3.2 notes, misaligned nodes can have high MNC. In practice, we find that this is comparatively rare, and we shed insight into when it may occur (§5.3.1). Second, the correct alignment may have low MNC, as is the case in some datasets consisting of multiple social networks sharing a limited number of common users [32, 20]. Recent work characterizes these datasets as having low *structural credibility*: topological consistency widely breaks down, and supervision and/or rich attribute information is generally needed for successful network alignment [28]. Future work could extend RefiNA to use such information.

5 Experiments

In this section, we first demonstrate RefiNA's ability to improve diverse unsupervised network alignment methods in a variety of challenging scenarios at reasonable computational cost. We next perform a deep study of RefiNA that verifies its various design insights.

5.1 Experimental Setup

5.1.1 Data. We choose network datasets from a variety of domains (e.g. biological, social) (Tab. 2). We consider two scenarios for network alignment with **ground truth node correspondence**: **(1)** ***Simulated noise.*** For each network with adjacency matrix $\mathbf{A}$, we create a permuted copy $\tilde{\mathbf{A}} = \mathbf{P}\mathbf{A}\mathbf{P}^\top$, where $\mathbf{P}$ is a random permutation matrix, and add noise by removing each edge with probability $p \in [0.05, 0.10, 0.15, 0.20, 0.25]$. The task is to align each network to its *noisy* permuted copy $\tilde{\mathbf{A}}^{(p)}$, with the ground truth alignments being given by $\mathbf{P}$. **(2)** ***Real noise.*** Our PPI-Y dataset is commonly studied in biological network alignment [25]. The task is to align the largest connected component of the yeast (*S.cerevisiae*) PPI network to its copies augmented with 5, 10, 15, 20, and 25 percent additional low-confidence PPIs (added in order of their confidence) [25].

Table 2: Description of the datasets used.

Name	Nodes	Edges	Description
Arenas Email [16]	1 133	5 451	communication
Hamsterster [16]	2 426	16 613	social
PPI-H [3]	3 890	76 584	PPI (human)
Facebook [18]	4 039	88 234	social
PPI-Y [25]	1 004	8 323	PPI (yeast)
LiveMocha [16]	104 103	2 193 083	social
SC & DM [26]	5 926	88 779	PPI (yeast)
	1 124	9 763	PPI (fruit fly)

We also show (§5.2.2) that RefiNA helps find more meaningful alignments even for networks defined on differing underlying nodes, by aligning graphs with **no ground truth node correspondence**: the PPI networks of **(3)** ***separate species*** (SC & DM). We align a yeast network (*S.cerevisiae*, a larger network than PPI-Y) to a fruit fly network (*D.melanogaster*).

5.1.2 Metrics. For graphs with ground-truth node correspondence, our main measure of alignment success is **accuracy**: the proportion of correctly aligned nodes. While this is the primary goal of network alignment, we also give **matched neighborhood consistency** (MNC, Eq. (3.1)) as a secondary metric (e.g. in Fig. 5a-b), in light of its importance in our analysis (§3.2).

When the networks have no ground-truth node correspondence, accuracy is not defined, so we assess two properties of the conserved network $\overline{\mathbf{A}} = \mathbf{M}^\top\mathbf{A}_1\mathbf{M} \circ \mathbf{A}_2$ (the overlap of the aligned adjacency matrices):

- **Normalized Overlap (N-OV)**: the percentage of conserved edges: $100 * \frac{\text{nnz}(\overline{\mathbf{A}})}{\max(\text{nnz}(\mathbf{A}_1),\text{nnz}(\mathbf{A}_2))}$
- **Largest Conserved Connected Component (LCCC)**: the number of edges in the largest connected component of $\overline{\mathbf{A}}$.

These respectively measure the alignment solution's ability to conserve edges and large substructures, which in biological networks may correspond to functions that

are shared across species [26]. Thus, larger values may indicate a more biologically interesting alignment.

5.1.3 Base Network Alignment Methods. To illustrate RefiNA's flexibility, we apply it to a set of base network alignment methods using a diverse range of techniques (belief propagation, spectral methods, genetic algorithms, and node embeddings): **(1) NetAlign** [2], **(2) FINAL** [32], **(3) REGAL** [13], **(4) CONE-Align** [4], and **(5) MAGNA** [25].

• *Base Method Settings.* We configure all base methods following the literature (cf. the supplementary §D).

• *RefiNA Settings.* By default, we use $K = 100$ refinement iterations (cf. supplementary §E) and token match score $\epsilon = \frac{1}{10^{\overline{p}}}$ where $\overline{p} = \min\{p \in \mathbb{N} : 10^p > n\}$, so that a node's token match scores will sum to less than 1. For sparse refinement, we update $\alpha = 10$ entries per node. We justify these choices by sensitivity analysis.

5.2 Analysis of Alignment Performance

5.2.1 Finding Known Correspondences. In Fig. 2 we plot the accuracy of all methods with (solid/-dashed lines) and without (dotted lines) refinement. For simulated experiments (2a-2d) we report average and standard deviation of accuracy over five trials.

Results. We see dramatic improvements in alignment accuracy for *all* methods with refinement. A striking example is NetAlign on the PPI-H dataset with 5% noise. Initially finding just 4% of alignments, with RefiNA it achieves 94% accuracy–going from a nearly completely incorrect to nearly completely correct. Similarly, CONE-Align achieves well over 90% accuracy on Arenas and PPI-H with refinement *and* sees only a slight drop even at the highest noise levels.

Observation 1. *RefiNA greatly improves the accuracy of diverse network alignment methods across datasets.*

We are also able to align networks that are much noisier than previous works had considered: our *lowest* noise level is 5%, the *highest* noise level in [13]. RefiNA's benefit is appreciable at up to 10% noise on most datasets for FINAL, 15% noise for NetAlign, 20% for REGAL, and the full 25% noise for CONE-Align.

Observation 2. *Refinement can make network alignment methods considerably more robust to noise.*

Fig. 2 and the supplementary Tab. 3 (§F) show that network alignment methods vary in how much their accuracy increases from RefiNA, and also the maximum noise level at which they appreciably improve. This variation shows that RefiNA is not ignoring the initial solution. Precisely characterizing the "refinability" of initial solutions is an interesting question for future work. Here, our goal is not to rank "better" or "worse" existing methods; instead, comparing all methods to *their own* unrefined solutions shows the value of RefiNA.

Observation 3. *Different methods benefit differently from refinement, but all benefit considerably.*

Compared to dense refinement, sparse refinement is slightly less accurate overall. However, the gap between the two variants decreases as the noise level increases, and in some cases (e.g., REGAL on Arenas and Hamsterster at high noise levels) sparse refinement even performs better, possibly indicating a regularizing effect of forgoing updates of low-confidence node pairs. Sparse refinement is faster than dense refinement, as Fig. 3 shows, but both offer a reasonable overhead compared to the initial alignment time. In general, they are modestly slower than NetAlign and REGAL, two famously scalable methods, on par with CONE-Align, faster than FINAL, and much faster than MAGNA.[1]

Observation 4. *RefiNA has a manageable computational overhead, particularly with sparse updates.*

5.2.2 Conserving Meaningful Structures. We now use RefiNA to help REGAL, NetAlign, and FINAL [2] align the SC-DM dataset consisting of the PPI networks of different species: graphs whose nodes do not necessarily have an underlying correspondence.

Results. Fig. 4 shows that using RefiNA, both measures of structural conservation increase significantly for all methods, indicating more successful alignments. In fact, these results are on par with ones obtained using amino acid sequence information in addition to graph topology [26]. Interestingly, sparse refinement brings even greater improvements than dense.

Observation 5. *RefiNA is useful even for graphs with no underlying node correspondence.*

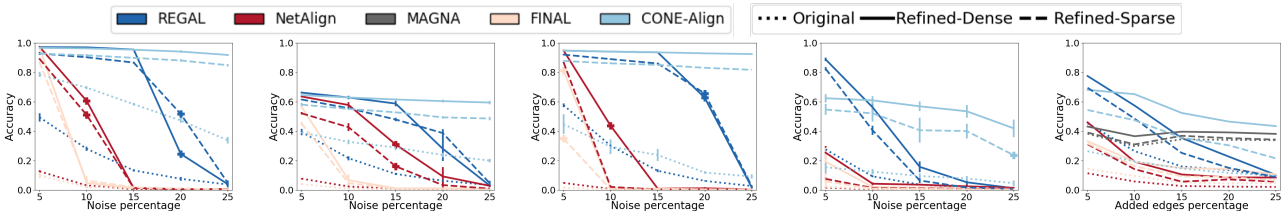
5.3 Drilldown: Network Alignment Insights.

In this section, we perform a drilldown of (dense) RefiNA that gives a deeper understanding into its specific design choices in light of the three insights we laid out in §4.

5.3.1 Aligning High Degree Nodes (`I1`). In Figs. 5a-b, we analyze the claim that high-degree nodes are easier to align, which inspires RefiNA's design, for NetAlign on the Arenas dataset for brevity.

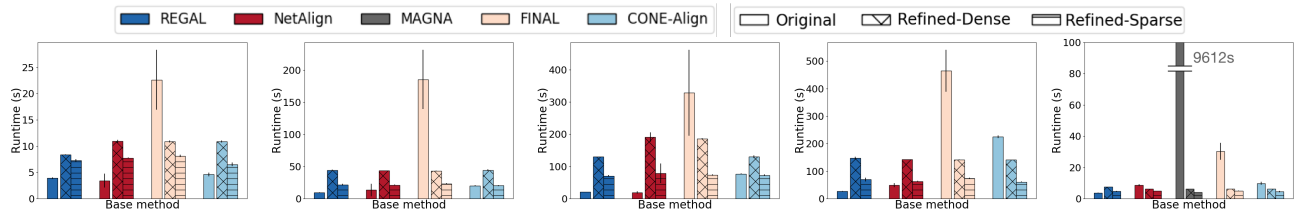
[1] On account of MAGNA's high runtime, we only run it on PPI-Y (where it takes 9612 seconds on average.)

[2] CONE-Align's subspace alignment operates on embeddings for the same number of nodes, which these graphs do not have.



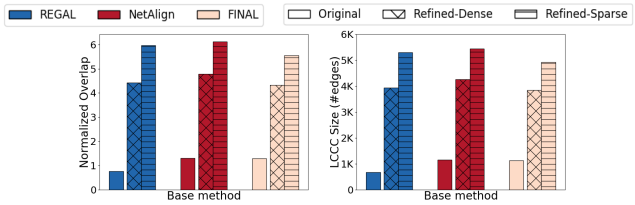
(a) Arenas (b) Hamsterster (c) PPI-H (d) Facebook (e) PPI-Y

Figure 2: Alignment accuracy vs graphs' topological difference. Both sparse and dense refinement with RefiNA improve all base methods' alignment accuracy and robustness, often quite dramatically. (We run MAGNA only on PPI-Y, the dataset with real noise, due to its high runtime—cf. Fig. 3(e).)



(a) Arenas (b) Hamsterster (c) PPI-H (d) Facebook (e) PPI-Y

Figure 3: Runtime for different refinement variations. Sparse refinement is appreciably faster than dense refinement. Both refinements offer a modest computational overhead, often on par with or faster than the time taken to perform the original network alignment.



(a) Normalized Overlap (b) Size of LCCC

Figure 4: Alignment of PPI networks of different species with no ground truth node alignment. RefiNA helps base methods find potentially more biologically interesting solutions by multiple metrics.

We split the nodes into three groups by degree: $[0, \frac{d_{\max}}{3}), [\frac{d_{\max}}{3}, \frac{2d_{\max}}{3}), [\frac{2d_{\max}}{3}, d_{\max}]$ ($d_{\max}$ is the maximum node degree) and plot the distribution of MNC in each group among correct and incorrectly aligned nodes.

Results. High-degree nodes are rarely misaligned even before refinement, and the few nodes that are still misaligned afterward have low degrees. These often still have a high MNC, probably because it is easier for smaller node neighborhoods to be (nearly) isomorphic, resulting in (near) structural indistinguishability (the challenge indicated by Theorem 3.2.)

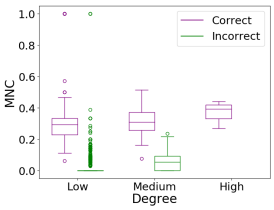
Observation 6. *It is easier to align high-degree nodes, verifying* `I1` *that motivates RefiNA's update formulation to encourage early alignment of high-degree nodes.*

5.3.2 Token Match Score (`I2`). We now study the effects of the token match score ϵ, used to overcome some limitations of an erroneous initial solution (our second network alignment insight in §4). We use the Arenas dataset with 5% noise averaged over five trials (we observe similar trends on other datasets), varying ϵ from 10^{-2} to 10^{-6} and $\epsilon = 0$ (no token match score).

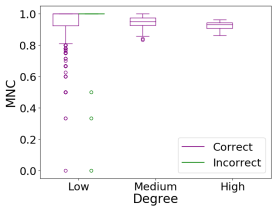
Results. In Fig. 5c, we see that the performance can dramatically drop with too large ϵ, where the token match scores overwhelm the alignment information. We need a positive token match score for the multiplicative update rule to work: with $\epsilon = 0$, RefiNA fails to discover new alignments and only achieves the initial solutions' accuracy (cf. Fig. 2a). However, RefiNA works with a wide range of small positive choices of ϵ, including $\epsilon = 10^{-4}$ as recommended by our criterion in § 5.1.

Observation 7. *RefiNA is robust to the token match score, as long as it is not so large that it would drown out the actual node similarity scores.*

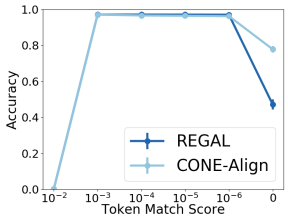
5.3.3 Alignment Matrix Normalization (`I3`). RefiNA normalizes the rows and columns of the alignment matrix, per our third insight in §4. Sinkhorn's algorithm iteratively repeats this process, converging to a doubly stochastic matrix [27]. Methods such as graduated assignment [10] nest iterative Sinkhorn normalization at every iteration of optimization. We refine NetAlign's solution on the Hamsterster dataset with 5% noise, both as proposed and using Sinkhorn's algorithm (up to 1000 iterations or a tolerance of 10^{-2}).



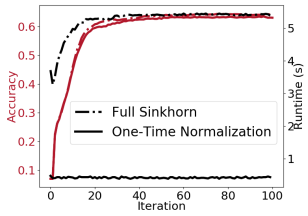
(a) Distribution of MNC and accuracy by node degree *before* refinement: NetAlign, Arenas 5% noise.



(b) Distribution of MNC and accuracy by node degree *after* refinement: NetAlign, Arenas 5% noise..

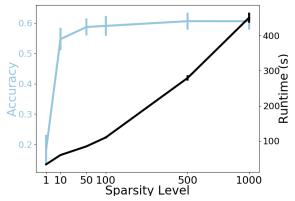


(c) Accuracy with varying token match score ϵ: Arenas, 5% noise. The methods almost overlap for $\epsilon > 0$.

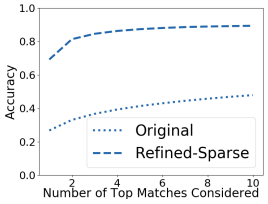


(d) Accuracy/runtime per iteration with limited vs full Sinkhorn normalization: NetAlign, Hamsterster 5% noise.

Figure 5: Drilldown of RefiNA in terms of our three insights that inspired its design. (a)-(b) For `I1`, before and after alignment, high degree nodes are more likely to have high MNC and be correctly aligned. (c) For `I2`, our token match score admits a wide range of values that yield good refinement performance. (d) For `I3`, our proposed normalization effectively avoids the computational expense of full Sinkhorn normalization.



(a) Accuracy/runtime vs sparsity level α: CONE-Align, Facebook 5% noise.



(b) Top-k accuracy on a time budget: REGAL, LiveMocha 5% noise.

Figure 6: Sparse RefiNA. (a) Varying the update sparsity allows us to trade off accuracy and runtime. (b) On extremely large graphs, RefiNA uncovers additional alignment (and near-alignments).

Results. We see in Fig. 5d that Sinkhorn's algorithm takes longer to converge (particularly as refinement continues) and dominates the running time. Meanwhile, our proposed normalization yields virtually the same accuracy, while keeping computation time low ($\ll$ 1 second per iteration) and relatively fixed.

Observation 8. *RefiNA is advantageous by being able to eschew the full Sinkhorn procedure.*

5.4 Sparse Updates and Scalability

5.4.1 Sparsity Level of Refinement. By varying α, the number of updates per node, we can interpolate between the sparse and dense versions of RefiNA. We study this on Facebook with 5% noise.

Results. Updating just one alignment per node leads to poor performance, as per `I2`: we should not blindly trust the initial solution by using only its top choice. However, the initial solution provides enough information that we can use only the top few choices, with marginal accuracy returns compared to the extra runtime required by using more top choices. Thus, sparse RefiNA offers a favorable balance of accuracy and speed.

5.4.2 "Soft" Alignments, Large Graphs. To utilize the scalability of sparse RefiNA, we use it on a large dataset: LiveMocha, which has over 100K nodes and a *million* edges. We simulate an alignment scenario with 5% noise. We only use REGAL, the most scalable base alignment method we consider, together with sparse refinement (dense refinement runs out of memory). We consider a *budgeted computation* scenario, running RefiNA for as long as REGAL takes (2600s). Meanwhile, we explore the top-k accuracy: the proportion of correct alignments in the top k choices per node [13] ranked by the real-valued entries of $\mathbf{M}$.

Results. In Fig. 6b, we see that RefiNA scales to large graphs and more than doubles the accuracy of REGAL in the same amount of time it took REGAL to obtain its initial solution. The top-k scores also increase as we consider more top matches per node (up to the sparsity parameter 10), which may be useful for some applications [13].

Observation 9. *Sparse refinement offers a favorable efficiency tradeoff and scales to large graphs.*

6 Conclusion

We have proposed RefiNA, a powerful technique for refining existing network alignment methods that greatly improves accuracy and robustness. RefiNA is simple to implement and apply *post hoc* to a variety of methods. Its compact formulation encodes several insights for network alignment, supported by our extensive theoretical and empirical analysis, and is highly scalable with sparse computation. We hope that all these positive qualities will make it attractive to practitioners.

Acknowledgements

This work is supported by NSF Grant No. IIS 1845491, Army Young Investigator Award No. W9-11NF1810397, and Adobe, Amazon, Facebook, and Google faculty awards.

References

[1] Yonathan Aflalo, Alexander Bronstein, and Ron Kimmel. On convex relaxation of graph isomorphism. *PNAS*, 2015.

[2] Mohsen Bayati, Margot Gerritsen, David F Gleich, Amin Saberi, and Ying Wang. Algorithms for large, sparse network alignment problems. In *ICDM*, 2009.

[3] Bobby-Joe Breitkreutz, Chris Stark, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, Michael Livstone, Rose Oughtred, Daniel H Lackner, Jürg Bähler, Valerie Wood, et al. The biogrid interaction database: 2008 update. *Nucleic acids research*, 2008.

[4] Xiyuan Chen, Mark Heimann, Fatemeh Vahedian, and Danai Koutra. Cone-align: Consistent network alignment with proximity-preserving node embedding. In *CIKM*, 2020.

[5] Tyler Derr, Hamid Karimi, Xiaorui Liu, Jiejun Xu, and Jiliang Tang. Deep adversarial network alignment. *arXiv preprint arXiv:1902.10307*, 2019.

[6] Joel Douglas, Ben Zimmerman, Alexei Kopylov, Jiejun Xu, Daniel Sussman, and Vince Lyzinski. Metrics for evaluating network alignment. *GTA3 at WSDM*, 2018.

[7] Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *NeurIPS*, 2019.

[8] Xingbo Du, Junchi Yan, and Hongyuan Zha. Joint link prediction and network alignment via cross-graph embedding. In *IJCAI*, 2019.

[9] Matthias Fey, Jan E Lenssen, Christopher Morris, Jonathan Masci, and Nils M Kriege. Deep graph matching consensus. In *ICLR*, 2020.

[10] Steven Gold and Anand Rangarajan. A graduated assignment algorithm for graph matching. *TPAMI*, 1996.

[11] Mark Heimann, Wei Lee, Shengjie Pan, Kuan-Yu Chen, and Danai Koutra. Hashalign: Hash-based alignment of multiple graphs. In *PAKDD*, 2018.

[12] Mark Heimann, Tara Safavi, and Danai Koutra. Distribution of node embeddings as multiresolution features for graphs. In *ICDM*, 2019.

[13] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. Regal: Representation learning-based graph alignment. In *CIKM*, 2018.

[14] Danai Koutra, Hanghang Tong, and David Lubensky. Big-align: Fast bipartite graph alignment. In *ICDM*, 2013.

[15] Oleksii Kuchaiev, Tijana Milenković, Vesna Memišević, Wayne Hayes, and Nataša Pržulj. Topological network alignment uncovers biological function and phylogeny. *Journal of the Royal Society Interface*, 2010.

[16] Jérôme Kunegis. Konect: the koblenz network collection. In *WWW*. ACM, 2013.

[17] Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. In *ICLR*, 2018.

[18] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[19] Chaozhuo Li, Senzhang Wang, Yukun Wang, Philip Yu, Yanbo Liang, Yun Liu, and Zhoujun Li. Adversarial learning for weakly-supervised social network alignment. In *AAAI*, 2019.

[20] Li Liu, William K Cheung, Xin Li, and Lejian Liao. Aligning users across social networks using network embedding. In *IJCAI*, 2016.

[21] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, 2002.

[22] Lei Meng, Joseph Crawford, Aaron Striegel, and Tijana Milenkovic. Igloo: Integrating global and local biological network alignment. In *MLG at KDD*, 2016.

[23] Rob Patro and Carl Kingsford. Global network alignment using multiscale spectral signatures. *Bioinformatics*, 2012.

[24] Kyle K Qin, Flora D Salim, Yongli Ren, Wei Shao, Mark Heimann, and Danai Koutra. G-crewe: Graph compression with embedding for network alignment. In *CIKM*, 2020.

[25] Vikram Saraph and Tijana Milenković. Magna: maximizing accuracy in global network alignment. *Bioinformatics*, 2014.

[26] Rohit Singh, Jinbo Xu, and Bonnie Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *PNAS*, 2008.

[27] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *Ann. Math. Stat.*, 1964.

[28] Chenxu Wang, Yang Wang, Zhiyuan Zhao, Dong Qin, Xiapu Luo, and Tao Qin. Credible seed identification for large-scale structural network alignment. *DAMI*, 2020.

[29] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *ICML*, 2019.

[30] Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, and Dongyan Zhao. Neighborhood matching network for entity alignment. In *ACL*, 2020.

[31] Hongteng Xu, Dixin Luo, Hongyuan Zha, and Lawrence Carin Duke. Gromov-wasserstein learning for graph matching and node embedding. In *ICML*, 2019.

[32] Si Zhang and Hanghang Tong. Final: Fast attributed network alignment. In *KDD*, 2016.

[33] Si Zhang, Hanghang Tong, Jie Tang, Jiejun Xu, and Wei Fan. ineat: Incomplete network alignment. In *ICDM*, 2017.

[34] Si Zhang, Hanghang Tong, Jiejun Xu, Yifan Hu, and Ross Maciejewski. Origin: Non-rigid network alignment. In *Big Data*, 2019.

[35] Zhen Zhang, Yijian Xiang, Lingfei Wu, Bing Xue, and Arye Nehorai. KerGM: Kernelized Graph Matching. In *NeurIPS*, 2019.

A Proofs of Theorems in § 3.2-4

Proof. (**Theorem 3.1: Perfect alignment accuracy implies high MNC**). By Eq. (3.1), $\text{MNC}(i,\pi(i)) = \frac{|\tilde{\mathcal{N}}^{\pi}_{\overline{G}_2}(i)\cap\mathcal{N}_{\overline{G}_2}(\pi(i))|}{|\tilde{\mathcal{N}}^{\pi}_{\overline{G}_2}(i)\cup\mathcal{N}_{\overline{G}_2}(\pi(i))|}$. By definition, $\tilde{\mathcal{N}}^{\pi}_{G_2}(i) = \tilde{\mathcal{N}}^{\pi}_{\overline{G}_2}(i)$; it does not change as neither π nor G_1's adjacency matrix $\mathbf{A}_1$ is affected by the noise. However, $\mathcal{N}_{\overline{G}_2}(\pi(i)) \subseteq \mathcal{N}_{G_2}(\pi(i))$, since under edge removal $\pi(i)$ can only lose neighbors in $\overline{G}_2$ compared to G_2.

Now $\tilde{\mathcal{N}}^{\pi}_{G_2}(i) = \mathcal{N}_{G_2}(\pi(i))$ since by definition an isomorphism is edge preserving, and so $\mathcal{N}_{G_2}(\pi(i)) = \tilde{\mathcal{N}}^{\pi}_{G_2}(i)$, which is the same as $\tilde{\mathcal{N}}^{\pi}_{\overline{G}_2}(i)$. Thus, $\mathcal{N}_{\overline{G}_2}(\pi(i)) \subseteq \tilde{\mathcal{N}}^{\pi}_{\overline{G}_2}(i)$. We can simplify $\text{MNC}(i,\pi(i)) = \frac{|\mathcal{N}_{\overline{G}_2}(\pi(i))|}{|\tilde{\mathcal{N}}^{\pi}_{\overline{G}_2}(i)|} = \frac{|\mathcal{N}_{\overline{G}_2}(\pi(i))|}{|\mathcal{N}_{G_2}(\pi(i))|}$. However, every node $j' \in \mathcal{N}_{G_2}(\pi(i))$ is also in $\mathcal{N}_{\overline{G}_2}(\pi(i))$ as long as the edge $\big(\pi(i), j'\big) \in \mathcal{E}_2$ has not been removed from $\tilde{\mathcal{E}}_2$, which happens with probability p. So, $\mathbb{E}\Big(\frac{|\mathcal{N}_{\overline{G}_2}(\pi(i))|}{|\mathcal{N}_{G_2}(\pi(i))|}\Big) = \mathbb{E}\big(\text{MNC}(i,\pi(i))\big) = 1-p$. ∎

Proof. (**Theorem 3.2: Perfect MNC implies perfect accuracy except for structurally indistinguishable nodes**). Since for isomorphic graphs, a node v is structurally indistinguishable from its true counterpart v', and since graph isomorphism is transitive, it suffices to show that v^* is also structurally indistinguishable from v. Suppose for some k, $\mathcal{N}_k(v)$ is not isomorphic to $\mathcal{N}_k(v^*)$. Then by definition there exists neighboring nodes $a, b \in \mathcal{N}_k(v)$ where either $\pi(a)$ or $\pi(b)$ is not in $\mathcal{N}_k(v^*)$, or $\pi(a)$ and $\pi(b)$ do not share an edge.

In case 1, without loss of generality $\pi(b) \notin \mathcal{N}_k(v^*)$. Then no bijective mapping exists between a shortest path between v^* and $\pi(b)$ and a shortest path from v^* to $\pi(b)$. There will thus be neighbors on one path whose counterparts are not neighbors on the other path, making MNC less than 1: a contradiction.

In case 2, since $\pi(b)$ is the counterpart of a neighbor of a, it must also be a neighbor of the counterpart of a, which is a contradiction of the assumption that $\pi(a)$ and $\pi(b)$ do not share an edge, or else $\text{MNC}(a,\pi(a)) < 1$, another contradiction. Thus, we conclude that the k-hop neighborhoods are isomorphic, that is, v and v^* are structurally indistinguishable. ∎

Proof. (**Theorem 4.1: Matrix form of MNC**). $\tilde{\mathcal{N}}^{\pi}_{G_2}(i) = \{\ell : \exists k \in \mathcal{V}_1 \text{ s.t. } \mathbf{A}_{1_{ik}}\mathbf{M}_{k\ell} \neq 0\}$, and of course $\mathcal{N}_{G_2}(j) = \{\ell : \mathbf{A}_{2_{j\ell}} \neq 0\}$. Since the product of two numbers is nonzero if and only if both numbers are nonzero, $\tilde{\mathcal{N}}^{\pi}_{G_2}(i) \cap \mathcal{N}_{G_2}(j) = \{\ell : \mathbf{A}_{1_{ik}}\mathbf{M}_{k\ell}\mathbf{A}_{2_{j\ell}} \neq 0\}$. For binary $\mathbf{A}_1, \mathbf{A}_2$, and $\mathbf{M}$, the cardinality of this set, which is the numerator of Eq. (3.1), is $\sum_{k\in\mathcal{V}_1,\ell\in\mathcal{V}_2} \mathbf{A}_{1_{ik}}\mathbf{M}_{k\ell}\mathbf{A}_{2_{j\ell}} = (\mathbf{A}_1\mathbf{M}\mathbf{A}_2)_{ij}$. Meanwhile, the denominator of Eq. (3.1) is $|\tilde{\mathcal{N}}^{\pi}_{G_2}(i) \cup \mathcal{N}_{G_2}(j)| = |\tilde{\mathcal{N}}^{\pi}_{G_2}(i)| + |\mathcal{N}_{G_2}(j)| - |\tilde{\mathcal{N}}^{\pi}_{G_2}(i) \cap \mathcal{N}_{G_2}(j)|$. Plugging in for each individual term, we obtain $\sum_{k\in\mathcal{V}_1}\sum_{\ell\in\mathcal{V}_2}\mathbf{A}_{1_{ik}}\mathbf{M}_{k\ell} + \sum_{\ell\in\mathcal{V}_2}\mathbf{A}_{2_{j\ell}} - \sum_{k\in\mathcal{V}_1}\sum_{\ell\in\mathcal{V}_2}\mathbf{A}_{1_{ik}}\mathbf{M}_{k\ell}\mathbf{A}_{2_{j\ell}}$. Substituting matrix products, this becomes $\sum_{\ell\in\mathcal{V}_2}(\mathbf{A}_1\mathbf{M})_{i\ell} + \sum_{\ell\in\mathcal{V}_2}\mathbf{A}_{2_{j\ell}} - (\mathbf{A}_1\mathbf{M}\mathbf{A}_2)_{ij}$. Using all-1 vectors to sum over columns, this is $(\mathbf{A}_1\mathbf{M}\mathbf{1}^{n_2})_i + (\mathbf{A}_2\mathbf{1}^{n_2})_j - (\mathbf{A}_1\mathbf{M}\mathbf{A}_2)_{ij}$. Then, expanding the two left vectors into matrices with outer product: $(\mathbf{A}_1\mathbf{M}\mathbf{1}^{n_2}\otimes\mathbf{1}^{n_2})_{ij} + (\mathbf{1}^{n_1}\otimes\mathbf{A}_2\mathbf{1}^{n_2})_{ij} - (\mathbf{A}_1\mathbf{M}\mathbf{A}_2)_{ij}$. Adding everything together, the denominator is the ij-th entry of the matrix $\mathbf{A}_1\mathbf{M}\mathbf{1}^{n_2}\otimes\mathbf{1}^{n_2} + \mathbf{1}^{n_1}\otimes\mathbf{A}_2\mathbf{1}^{n_2} - \mathbf{A}_1\mathbf{M}\mathbf{A}_2$. ∎

B Connections to Other Graph Methods

We show additional connections between RefiNA and other diverse graph methods: first, *seed-and-extend* as an alignment strategy, and second a graph filtering perspective on RefiNA's update rule similar to the analysis of graph neural networks.

Seed-and-extend alignment heuristic. Many global network alignment methods [15, 23] use this heuristic to find node correspondences between two or multiple networks. Given initial pairwise similarities (or alignment costs) between nodes of the compared graphs as $\mathbf{M}$, a pair of nodes i and j with high probability to be aligned (e.g., whose similarity according to $\mathbf{M}$ is above some confidence threshold) are set as the seed regions of the alignment. After the seed (i,j) is selected, the r–hop neighborhoods of i and j (i.e., $\mathcal{N}_{r,G_1}(i)$ and $\mathcal{N}_{r,G_2}(j)$) in their respective graphs are built. Next, the selected seed (i,j) is extended for the final alignment $\mathbf{M}'$ by greedily matching nodes in $\mathcal{N}_{r,G_1}(i)$ and $\mathcal{N}_{r,G_2}(j)$, searching for the pairs $(i',j') : i' \in \mathcal{N}_{r,G_1}(i)$ and $j' \in \mathcal{N}_{r,G_2}(j)$ that are not already aligned and can be aligned with the maximum value of similarity according to $\mathbf{M}$. The process can be written as $\forall k \in \mathcal{N}_{r,G_1}(i), \mathbf{M}'_{k\ell} \neq 0$ if $\ell = \arg\max_{\ell\in\mathcal{N}_{r,G_2}(j)}\mathbf{M}_{k\ell}$.

By setting $r = 1$ to consider each seed's direct neighbors, and $\mathbf{M}'_{k^*\ell^*} \neq 0$ if $k^*, \ell^* = \arg\max_{k\in\mathcal{V}_1,\ell\in\mathcal{V}_2}\mathbf{A}_{1_{ik}}\mathbf{M}_{k\ell}\mathbf{A}_{2_{j\ell}}$, we see that the seed-and-extend heuristic analyzes the same set of elements used to compute the update in RefiNA (Eq. (4.4)). However, instead of summing them to update the similarity of seed nodes i and j, it takes the argmax over them to adaptively select the next pair of alignments. Thus, seed-and-extend aligns less well with `I1` by relying heavily on a correct initial solution, as the early alignments are irrevocable and used to restrict the scope of subsequent alignments.

Graph Filtering. The matching matrix $\mathbf{M}$ can also be interpreted as a high-dimensional feature matrix. For example, for each node in G_1, a row of $\mathbf{M}$ may be regarded as an n_2-dimensional feature vector consisting of the node correspondences to each of the nodes in

G_2, and similarly the $n_2 \times n_1$ matrix $\mathbf{M}^\top$ contains n_1-dimensional cross-network correspondence features for each node in G_2. For challenging alignment scenarios, these are likely highly noisy features. However, recent works have shown that multiplying a node feature matrix by the graph's adjacency matrix corresponds to a low-pass filtering operation, which is of interest in explaining the mechanisms of graph neural networks [29].

We can write our update rule in Eq. (4.4) as a feature matrix left multiplied by adjacency matrices: $\mathbf{A}_1\mathbf{M}\mathbf{A}_2 = \mathbf{A}_1(\mathbf{A}_2\mathbf{M}^\top)^\top$ (for undirected graphs, $\mathbf{A}_2^\top = \mathbf{A}_2$), where $\mathbf{A}_2\mathbf{M}^\top$ produces a filtered set of n_1-dimensional features. By taking the transpose, these may be interpreted as n_2-dimensional features for each node of G_1, which are then filtered again by left multiplication with $\mathbf{A}_1$.[3]

Interpreting RefiNA's updates as graph filtering explains its strong performance, as well as the success of recent supervised graph matching work [34, 9] using graph neural networks. Of course RefiNA does not have the learnable parameters and nonlinearities of a graph neural network. However, just as SGC [29] recently compares favorably to graph neural networks by replacing their deep nonlinear feature extraction with repeated multiplication by the adjacency matrix, we find that that unsupervised "alignment filtering" is highly effective.

Graph Kernels. Just as nodes can be matched across networks using suitable cross-network node similarities, entire graphs can be compared (for the purposes of performing machine learning tasks on graph-structured data points) by aggregating their nodes' similarities [12, 7]. Some base network alignment methods we studied are known to have close connections to graph kernels. For example, FINAL's pooled cross-network node similarities are closely related to a graph kernel based on random walks [32]; likewise, the xNetMF node embeddings that REGAL uses to match nodes one by one [13] can also be used to construct a feature map for an entire graph, where the dot product between two graphs' feature maps approximates the mean-pooled (kernelized) node embedding similarities [12]. We now show that the update rule of RefiNA is also related to the graph neural tangent kernel (GNTK) [7], a graph kernel designed to achieve the effect of an infinitely wide graph neural network trained by gradient descent. GNTK starts by computing a cross-network node similarity matrix, the same as our matrix $\mathbf{M}$. (As proposed in [7], this matrix is computed using input node features that are common in graph classification benchmark datasets, but the formulation can be more general.) To achieve the effect of a graph neural network's feature aggregation, the node similarities are iteratively propagated across graphs:

$$\mathbf{M}'_{ij} = c_i c_j \sum_{u \in \{\mathcal{N}_{G_1}(i) \cup i\}} \sum_{v \in \{\mathcal{N}_{G_2}(j) \cup j\}} \mathbf{M}_{uv}$$

Up to scaling factors c_i and c_j designed to model the effects of particular graph neural network architectures, this term is elementwise equivalent to our update $\mathbf{M}' = \mathbf{A}_1\mathbf{M}\mathbf{A}_2$, where the graphs' adjacency matrices have been augmented with self-loops as is commonplace for graph neural networks. At each round of propagation, GNTK post-processes $\mathbf{M}$ by performing R transformations corresponding to R fully-connected neural network layers with ReLU activation, whereas we simply add token match scores and normalize.

The analysis for GNTK connects the training of kernel machines (the final kernel value is obtained by applying a READOUT operation to the refined node similarities, a simple example being to sum them all) with that of graph neural networks, in either case using graph-level supervision [7]. While this is of course a quite different setup from the unsupervised node-level matching problem RefiNA tackles, a very interesting direction for future work is to see if RefiNA and/or GNTK can benefit from further methodological exchange (at minimum, our sparse refinement techniques could approximate the computation of GNTK between large graphs). Such analysis may also reveal more about the general connection between network alignment based on individual node-level similarities and network classification based on aggregated node-level similarities.

C Complexity Analysis of RefiNA

Here, we analyze the precise computational complexity of RefiNA with both sparse and dense refinement.

C.1 Complexity of Dense Refinement. To simplify notation, we assume that both graphs have n nodes [13]. For K iterations, our algorithm computes the left and right multiplication of a dense $n \times n$ matching matrix with two adjacency matrices of graphs with average degree (number of nonzero entries per row of the adjacency matrix) $\bar{d}_1$ and $\bar{d}_2$, respectively. Thus, the time complexity of this update step is $O(n^2(\bar{d}_1 + \bar{d}_2))$. Normalizing the matrix at each iteration and adding in token match scores requires $O(n^2)$ time. Therefore, the overall time complexity is $O\Big(Kn^2(\bar{d}_1 + \bar{d}_2)\Big)$. While the number of iterations and average node degree are in practice constant or asymptotically smaller than the

[3] Graph convolutional networks use the augmented normalized adjacency matrix $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, which we have not found helpful.

graph size, the quadratic time and space complexity of RefiNA's dense matrix operations makes it harder to apply to large graphs.

C.2 Complexity of Sparse Refinement. For K iterations, we compute a sparse update matrix with $O(n\alpha)$ nonzero entries by multiplying matrices with $O(n\bar{d}_1)$, $O(Kn\alpha)$, and $O(n\bar{d}_2)$ nonzero entries, respectively. It takes $O(nK\alpha\bar{d}_1)$ time to compute $\tilde{\mathbf{A}}_1 = \mathbf{A}_1\mathbf{M}_{k-1}$ and $O(nK\alpha\bar{d}_1\bar{d}_2)$ time to compute $\tilde{\mathbf{A}}_1\mathbf{A}_2$. We then compute $\mathbf{M}_k$ by updating $O(n\alpha)$ entries in $\mathbf{M}_{k-1}$ per iteration. Thus, $\mathbf{M}_k$ may have $O(Kn\alpha)$ nonzero entries and requires $O(Kn\alpha)$ time to update and normalize. Altogether, the runtime is now $O(nK^2\alpha\bar{d}_1\bar{d}_2)$, i.e. linear in the number of nodes. (This is a worst-case analysis; in practice, the runtime scales close to linearly with K.) We can also avoid storing a dense matching matrix, leading to subquadratic space complexity.

D Baseline Hyperparameter Settings

For REGAL's own xNetMF embeddings, we used default embedding dimension $\lfloor 10\log_2(n_1+n_2)\rfloor$ [13], maximum neighborhood distance 2, neighborhood distance discount factor $\delta = 0.1$, and resolution parameter $\gamma_{\text{struc}} = 1$, all recommended parameters. For the embeddings used in CONE-Align, we set embedding dimension $d = 128$, context window size $w = 10$, and negative sampling parameter $\alpha = 1$. We used $n_0 = 10$ iterations and regularization parameter $\lambda_0 = 1.0$ for the convex initialization of the subspace alignment, which we performed with $T = 50$ iterations of Wasserstein Procrustes optimization with batch size $b = 10$, learning rate $\eta = 1.0$, and regularization parameter $\lambda = 0.05$ as were suggested by the authors [4]. For REGAL and CONE-Align, we computed embedding similarity with dot product followed by softmax normalization [9], using k-d trees to perform fast 10-nearest neighbor search for REGAL on LiveMocha [13].

NetAlign and FINAL require a matrix of prior alignment information, which we computed from pairwise node degree similarity. Then following [13, 5], we constructed this matrix by taking the top $k = \lfloor \log_2\big((n_1 + n_2)/2\big)\rfloor$) entries for each node in G_1; that is, the top k most similar nodes in G_2 by degree.

For MAGNA, starting from a random initial population with size 15000, we simulated the evolution for 2000 steps following [25] using edge correctness as the optimizing measure as it is similar to the objectives of our other methods. We used 3 threads to execute the alignment procedure.

We consider the output of all methods to be a binary matrix $\mathbf{M}$ consisting of the "hard" (one-to-one) alignments they find, to treat methods consistently and to show that RefiNA can refine the most general network alignment solution. It is worth noting that some methods (e.g. REGAL, CONE-Align) can also produce "soft" alignments (real-valued node similarity scores) and our formulation is capable of using those.

Computing Environment. We performed all experiments on an Intel(R) Xeon(R) CPU E5-1650 at 3.50GHz, 256GB RAM.

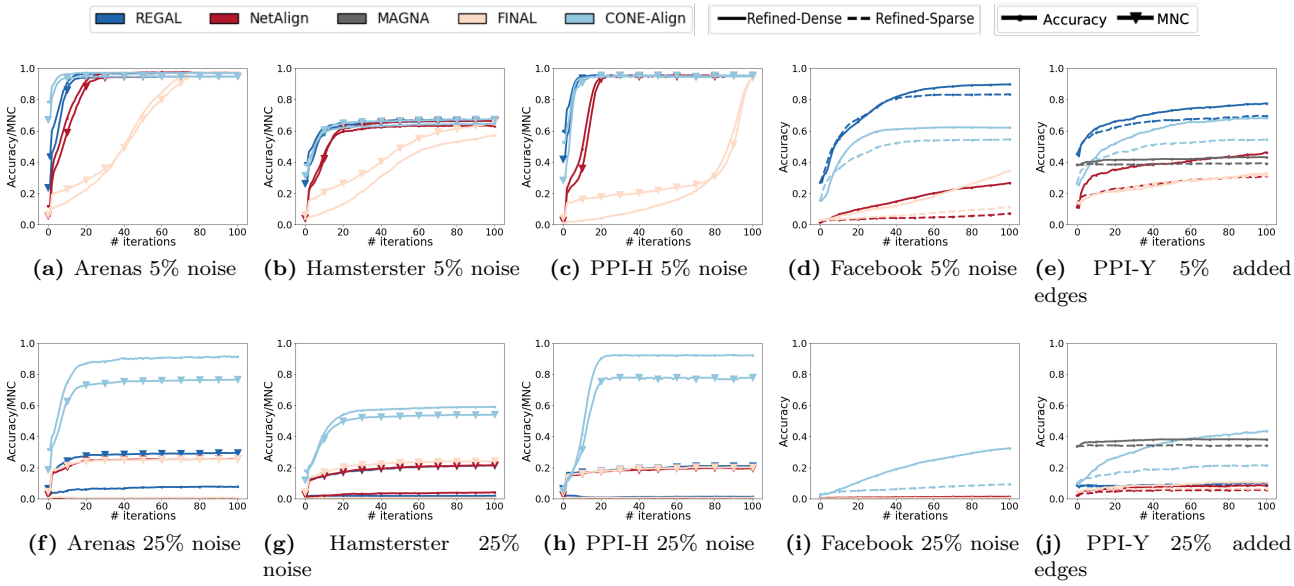
E Convergence Analysis: Accuracy & MNC

One of the parameters of RefiNA is K, the number of iterations for which the initial matching is refined. In Fig. 7, we plot the performance at each iteration, up to our maximum value of 100, for all methods and datasets (at lowest and highest noise levels, or number of added low-confidence PPIs in the case of PPI-Y). For brevity, we show accuracy and MNC for dense refinement only on the first three datasets (Arenas, Hamsterster, and PPI-H), and the per-iteration accuracy only for sparse and dense refinement on the remaining datasets.

Results. We see that accuracy and MNC tend to trend similarly, as do sparse and dense refinement. In both cases, we see similar trends for the same colored curves. As for RefiNA variations, dense refinement of CONE-Align grows in accuracy slightly more steeply than sparse refinement. For MNC, we see that accuracy and MNC tend to have very similar values at 5% noise (thus the lines of the same color are close to overlapping). At 25% noise, MNC is lower than accuracy for highly accurate methods like CONE-Align—this is to be expected because Theorem 3.1 proved that the expected average MNC under even for a perfect alignment solution is bounded by the noise ratio. For the remaining methods which struggle to achieve high accuracy, MNC is higher than accuracy. As Theorem 3.2 showed, it is possible to find a high-MNC alignment that is still not perfectly accurate (due to structural indistinguishability). Indeed, here we see this happening in some especially challenging settings starting from less favorable initial solutions. Nevertheless, in most cases, improving MNC is a successful strategy for accurate network alignment.

Convergence rates differ for different methods and datasets. For example, FINAL is generally the slowest method to converge, taking close to 70 iterations to fully converge on the Arenas dataset, with NetAlign taking around 20, REGAL around 10, and CONE-Align around 5. On the PPI-H dataset with 5% noise, FINAL sees slow progress for nearly the full 100 iterations before making rapid progress at the end.

In general, we find that a modest number of iterations is generally sufficient. While we allow all methods 100 iterations of refinement in our experiments, the elbow-shaped curves indicate that in practice, conver-



(a) Arenas 5% noise **(b)** Hamsterster 5% noise **(c)** PPI-H 5% noise **(d)** Facebook 5% noise **(e)** PPI-Y 5% added edges

(f) Arenas 25% noise **(g)** Hamsterster 25% noise **(h)** PPI-H 25% noise **(i)** Facebook 25% noise **(j)** PPI-Y 25% added edges

Figure 7: Analysis of RefiNA as a function of number of iterations (0 = performance of base method before refinement). Arenas, PPI-H, and Hamsterster datasets show accuracy and MNC, and Facebook and PPI-Y datasets show accuracy of sparse and dense RefiNA versions. Convergence rates are different for different methods, but often happen well before 100 iterations for both sparse and dense RefiNA. Accuracy and MNC consistently increase and follow similar trends, as per their theoretical connection.

Table 3: Maximum improvement thanks to RefiNA for each base method on each dataset (with abbreviated name), and maximum noise level at which RefiNA brings noticeable improvement. For all methods and all datasets, RefiNA brings dramatic increases in accuracy and can often bring significant improvement even at very high noise levels.

	REGAL	NetAlign	FINAL	CONE-Align	MAGNA
AR	82.18% *20% noise*	84.63% *10% noise*	86.93% *5% noise*	58.02% *25% noise*	N/A
P-H	80.53% *20% noise*	90.02% *10% noise*	79.57% *5% noise*	83.28% *25% noise*	N/A
HA	47.86% *20% noise*	55.85% *15% noise*	52.68% *10% noise*	39.50% *25% noise*	N/A
FB	61.22% *15% noise*	23.76% *15% noise*	17.03% *5% noise*	48.51% *25% noise*	N/A
P-Y	32.17% *20% noise*	34.67% *25% noise*	18.83% *25% noise*	45.72% *25% noise*	6.77% *25% noise*

gence often happens in far fewer than 100 iterations (with some exceptions for a few methods on the PPI-Y and Facebook datasets). In practice, early convergence can be ascertained by small changes in the discovered alignments.

Observation 10. *Accuracy and MNC, sparse and dense refinement tend to trend similarly on each method on each dataset. Although convergence rates differ for different methods and datasets, convergence is quite fast in practice.*

F Improvement thanks to RefiNA

To further illustrate that different methods benefit differently from refinement, we summarize our results from Fig. 2 in tabular format in Tab. 3, where we show the maximum improvement in mean accuracy thanks to RefiNA for each base method on each dataset (across refinement types and noise levels, averaged over five trials). We also show the maximum noise level at which RefiNA is able to yield a noticeable improvement (3% or more). As discussed in §5, while all network alignment methods benefit from RefiNA, this does not eliminate the effect of the initial solution. We have also observed that RefiNA poorly refines a randomly initialized alignment matrix on our datasets. Again, this indicates that our contributions complement rather than replace existing network alignment solutions.