A Additional Related Work

Theoretical Algorithms Existing theoretical algorithms suffer from several limitations. First, graphs with different characteristics (e.g., Erdős-Rényi graphs versus power-law graphs) may require different features and carefully tuned parameters. In contrast, by learning from the training graphs, our SeedGNN can automatically choose the effective features. Second, these theoretical algorithms may not synthesize different features most effectively. For instance, the l-hop algorithm in (Mossel et al., 2019) only utilizes the witnesses at a specific hop l, but does not study how to combine witnesses at different hops.

Inductive Semi-supervised Learning on Graphs Our goal of using supervised learning for seeded graph matching shares some similarity with the work in Wen et al. (2021), which also aims to both perform inductive learning (i.e., learn transferrable knowledge from training graphs) and utilize a small amount of labeled data on the test graph. However, Wen et al. (2021) focuses on a node classification problem, which is quite different from seeded graph matching. In particular, Wen et al. (2021) uses node-based GNNs, which (as we discussed in Section 4.1) have more difficulty in effectively utilizing seed information than our proposed pair-wise GNN. Further, in order to transfer knowledge from the trained GNN to test graphs, Wen et al. (2021) scales all GNN weights by a common factor. It is unclear how this scaling will effectively transfer knowledge for seeded graph matching, e.g., how to best use different hops of witnesses. In contrast, our design of SeedGNN exploits the inherent structure of the seeded graph matching problem, and can be shown to generalize well to unseen graphs of sizes and types very different from the training set. For future work, it would be of interest to explore whether our SeedGNN can be further improved with a meta-learning component (Santoro et al., 2016).

Convex Relaxation Algorithms In addition to the theoretical algorithms and the GNN approaches, there is another class of algorithms based on convex relaxations of the quadratic assignment problem, which maximizes the total number of matched edges between two graphs subject to the seed constraint (Lyzinski et al., 2014; Fishkind et al., 2019). In (Fishkind et al., 2019), the authors describe a gradient ascent approach to solve this relaxed problem, which is called SGM. Compared to SeedGNN, SGM also has flavors of using witnesses and percolation ideas. Specifically, the gradient of the SGM algorithm is similar to a matrix counting 1-hop witnesses. However, using only 1-hop witnesses is known to be ineffective in sparse graphs (as there are very few 1-hop witnesses even for true pairs). Indeed, our experiments in Section 5 find that our SeedGNN often outperforms SGM, especially in sparse graphs.

Differences in Using Similarity Matrix and Masking We note that both the idea of using similarity matrix to refine higher-layer matching and the idea of masking have appeared in seedless matching. For example, some previously proposed node-based GNN architectures for *seedless* graph matching also compute the similarity matrix and use it to refine the node embedding in each layer (Wang et al., 2019; Fey et al., 2020). However, these approaches heavily rely on high-quality non-topological node features and does not clean up the "noisy" information as we carefully did. Yu et al. (2019) also uses the Hungarian algorithm for seedless graph matching, but they only clean up the matching result in their loss function. The results of intermediate layers are still very noisy. We use the Hungarian algorithm in each layer to filter out the misleading information, and thus the final result would be better.

Differences between Our Percolation Module and Previous Percolation Aalgorithms Unlike previous percolation algorithms (Yartseva et al., 2013), we allow SeedGNN to correct errors from ealier layers by re-matching nodes at each layer. Note that in many percolation algorithms, once a new pair of seeds is identified, it will be used as the correct matching until the end. This approach can be problematic if an incorrect pair is identified as seeds, whose impact will be lasting for many iterations down the road. In contrast, since our SeedGNN rematches nodes at each layer, even if some of the newly-identified seeds in the previous layer are incorrect, we can potentially correct these errors in the next layer, as long as the fraction of incorrect seeds is small. In other words, our design of SeedGNN takes advantage of the power of partially-correct (i.e., noisy) seeds (as theoretically verified in Yu et al. (2021a))

A.1 COMPARISON WITH NGM

The NGM architecture of Wang et al. (2021) shares some similarity with our SeedGNN, and it also uses a pair-wise GNN and uses the affinity matrix as input. However, note that Wang et al. (2021) focuses on seedless graph matching. Therefore, the NGM architecture in Wang et al. (2021) was not designed for seeded graph matching. For example, they do not aim to exploit important features such as witness. Further, the NGM algorithm has not been evaluated for seeded graph matching either. In Section 5.2, we transferred the NGM approach to seeded graph matching by modifying the affinity matrix to encode seed information, and compare its performance with our SeedGNN. Through these experiments, we find that the NGM algorithm in Wang et al. (2021) does not generalize well when the test graph is with much larger size and node-degree than the training graph. Below, we discuss the possible reasons.

Recall that we train both NGM and our SeedGNN on the same training set in Section 5.1, and test on correlated Erdős-Rényi graphs with $n = 500, s = 0.8, p = \{0.01, 0.2\}$. Note that this test graph size is larger than the training Erdős-Rényi graph size of $n = 100$. From the experimental results in Figure 5 in Section 5.2, we can observe that, although NGM performs close to our SeedGNN in larger sparse graphs ($p = 0.01$), it performs quite poorly in larger dense graphs ($p = 0.2$). One possible reason for this deterioration in the generalization power of NGM could be that, in the aggregation step, NGM normalizes each representation by the vertex degree of the association graph (which is roughly the square of the node degrees), but we do not. To see why this difference matters, note that according to known theoretical results on seeded graph matching, there exist algorithms that only need $\Omega(\log n)$ seeds to match all n nodes (Mossel et al., 2019). However, if the graph sparsity p is fixed, the node degree increases proportionally to n, and correspondingly the vertex degree of the association graph increases quadratically with n. As a result, when NGM divides the similarity of each node pair by the vertex degree, we expect that the resulting value ($\sim \frac{\log n}{n^2}$) will decrease close to zero as the graph size increases. Hence, it would be difficult for the sinkhorn step in NGM to distinguish the true pairs from the fake pairs in test graphs with larger size and node degrees than the training graphs. In contrast, since SeedGNN does not divide the similarity scores by the vertex degrees, the Hungarian algorithm step in our percolation (which can distinguish any absolute difference) will then be able to distinguish the true pairs from the fake pairs.

In contrast to Figure 5(b), for the experiment on the SHREC'16 dataset (Table 1), NGM has similar performance as our SeedGNN. This is because in this experiment, we train NGM also with the SHREC '16 dataset (same as other seedless GNNs in Table 1). Note that the node degrees of the graphs in the SHREC'16 dataset are all around 6. In other words, the test graphs and training graphs are with similar node degrees. As a result, the issue caused by dividing the similarity scores by the vertex degree of the associate graphs is not as critical for the SHREC'16 dataset.

B COMPLEXITY AND SCALABILITY

B.1 TIME AND SPACE COMPLEXITY

First, we analyze the computational complexity of our SeedGNN. In each layer, counting witnesses in (2) takes $O(n_1 n_2 d_{\text{mean}})$ time. The neural networks (3) and (5) take $O(n_1 n_2)$ time. The Hungarian algorithm takes $O(n_1 n_2^2)$ times (Crouse, 2016). Thus, the total time complexity is $O(n_1 n_2^2)$.

The space complexity of our SeedGNN is $O(n_1 n_2)$ since we need to store the representations of all $n_1 n_2$ node-pairs in each layer.

B.2 MAKING SEEDGNN MORE SCALABLE

For very large graphs, the step of the Hungarian algorithm may potentially become the computational bottleneck. We can use greedy max-weight matching (GMWM) in (Avis, 1983) instead, as the time complexity of GMWM is only $O(n_1 n_2 \log n_2)$. With this improvement, the total time-complexity is reduced to $O(n_1 n_2 \log n_2 + n_1 n_2 d_{\text{mean}})$. To the best of our knowledge, the best-known time complexity for GNN-based algorithms is $O(n_1 n_2)$ (Fey et al., 2020). Thus, the computational complexity of our SeedGNN is only moderately larger than the best known one. Our numerical

result shown in Table 1 has demonstrated that the run time of our SeedGNN is comparable to the best-known GNN-based algorithms.

C STUDYING THE INNER-WORKING OF SEEDGNN

In this section, we further investigate how the performance of SeedGNN varies as we change its inner working. First, to verify the effectiveness of our design choices for our SeedGNN method, we compare the performance of different architectural designs. Then, we investigate which sets of samples need to be included in our training set to obtain an effective trained model. Finally, we study the matching process of SeedGNN for different types of graphs. The results suggest that SeedGNN could potentially choose the appropriate features for different graphs based on the confidence level of new seeds.

C.1 STUDY OF THE DESIGN CHOICES

To verify the effectiveness of our design choices, we consider four variants of SeedGNN, which are:

1. **SeedGNN-x:** SeedGNN without convolution module. This variant aims to verify the importance of extracting witness-like information at a larger number of hops.
2. **SeedGNN-w:** SeedGNN without percolation module. This variant aims to verify the importance of the percolation module in SeedGNN.
3. **SeedGNN-p:** SeedGNN with percolation module but without the Hungarian matching algorithm (i.e., $z_l = \mathsf{unvec}(Y_l)$ in each GNN layer). This variant aims to verify the importance of the "cleaning" process in SeedGNN.
4. **SeedGNN-h:** SeedGNN with $z_l = \mathsf{unvec}(R_l)$ instead of (7) in each layer. This variant aims to verify that among the new seeds, it is still important to distinguish the high-confident one and low-confident one.

Finally, we use "SeedGNN" to denote the full design in Fig. 1. We train all these variants with the same training set $\mathcal{T}$ in Section 5.1.

In Figure 8, we show the performance of the above variants of SeedGNN on correlated Erdős-Rényi graph model. For test graphs, we increase θ from 0 to 0.05 while fixing $n = 500$, $p = 0.04$, $s = 0.8$. As illustrated in Figure 8, our SeedGNN with full design achieves the best performance among all variants, which shows the effectiveness of our design choices for the SeedGNN architecture. Further, among the variants, SeedGNN-w almost fails completely, which highlights the significant importance of using the percolation idea in SeedGNN for seeded graph matching. SeedGNNx does performs poorly, which demonstrates that it is also important to extract witness information at a larger number of hops instead of only 1-hop. We can observe that SeedGNN and SeedGNN-h both outperform SeedGNN-p and the improvement of SeedGNN is significantly bigger. This result verifies that it is not enough to only use the soft-correspondence (as in SeedGNN-p), and we need to combine both the matching result R_l of the Hungarian algorithm and the similarity Y_l as in (7) to achieve the best performance.

C.2 STUDY OF THE NECESSARY TRAINING SAMPLES FOR GENERALIZATION

Intuitively, in order to help our SeedGNN successfully learn useful knowledge that can be applied to never-seen graphs, the training set needs to contain graph pairs with different varieties, e.g., graph sparsity, graph correlation, and the size of seed set. However, a larger training set also increases the training time. To show which sets of graph pairs are necessary, we compare SeedGNN trained with different training sets, whose parameters are shown in Table 2. We use $\mathcal{T}$ to denote the training set that only includes the Erdős-Rényi graphs of the training set in Section 5.1. First, to show the necessity of training graph pairs with a wide range of sparsity, we train SeedGNN with $\mathcal{T}$, $\mathcal{T}_{p1}$ and $\mathcal{T}_{p2}$, and compare the performance of the trained models while increasing p from 0.02 to 0.2 and fixing $n = 500$, $s = 0.8$ and $\theta = 0.05$. Figure 9(a) shows that, if SeedGNN is only trained with $p = 0.1$, it performs well on sparse graphs but poorly on dense graphs. In contrast, if SeedGNN is only trained with $p = 0.5$, it performs well on dense graphs but poorly on sparse graphs. Thus, we should include both $p = 0.1$ and $p = 0.5$ in the training set to achieve good performance. Second, to

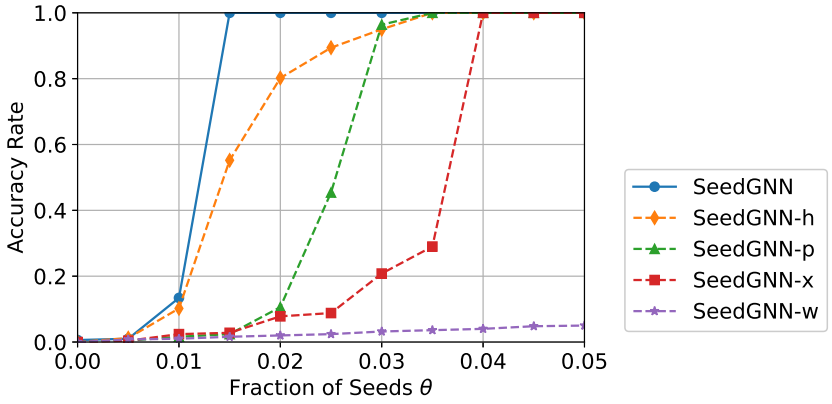
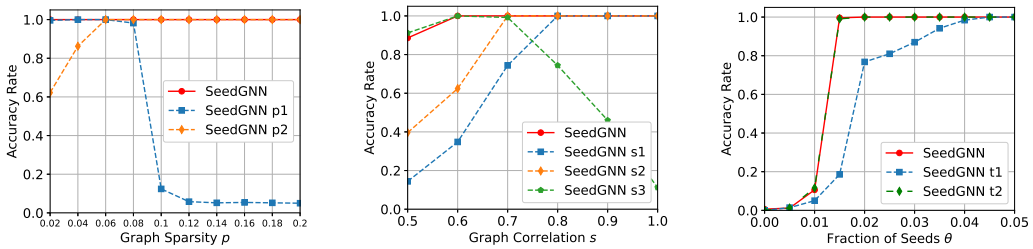


Figure 8: Performance comparison of our SeedGNN and four other variants on correlated Erdős-Rényi graph model with different θ. Fix $n = 500$, $p = 0.04$, $s = 0.8$.

show the necessity of training graph pairs with different correlations, we compare the performance of SeedGNN trained with $\mathcal{T}$, $\mathcal{T}_{s1}$, $\mathcal{T}_{s2}$ and $\mathcal{T}_{s3}$, and compare these models while increasing s from 0.5 to 1 and fixing $n = 500$, $p = 0.08$ and $\theta = 0.05$. Figure 9(b) shows that, if SeedGNN is only trained with $s = 0.6$, it performs well on moderately correlated graphs but poorly on highly correlated graphs. In contrast, if SeedGNN is only trained with $s = 0.8$ or $s = 1$, it performs well on highly correlated graphs but poorly on moderately correlated graphs. Thus, we should include different correlations in the training set to achieve good performance. Third, we compare the performance of SeedGNN trained with $\mathcal{T}$, $\mathcal{T}_{t1}$ and $\mathcal{T}_{t2}$, and compare these models while increasing θ from 0 to 0.05 and fixing $n = 500$, $p = 0.04$ and $s = 0.8$. Figure 9(c) shows that, if SeedGNN is only trained with $\theta = 0.1$ and $\theta \in \{0.1, 0.3\}$, it performs exactly the same. If SeedGNN is only trained with $\theta = 0.3$, it performs worse than the former two. Thus, we only need to include graph pairs with a relatively small seed set in the training set.

Table 2: Different Training Sets

Training Sets	p	s	θ
$\mathcal{T}_{p1}$	$\{0.1\}$	$\{0.6, 0.8, 1\}$	$\{0.05, 0.1\}$
$\mathcal{T}_{p2}$	$\{0.5\}$	$\{0.6, 0.8, 1\}$	$\{0.05, 0.1\}$
$\mathcal{T}_{s1}$	$\{0.1, 0.5\}$	$\{1\}$	$\{0.05, 0.1\}$
$\mathcal{T}_{s2}$	$\{0.1, 0.5\}$	$\{0.8\}$	$\{0.05, 0.1\}$
$\mathcal{T}_{s3}$	$\{0.1, 0.5\}$	$\{0.6\}$	$\{0.05, 0.1\}$
$\mathcal{T}_{t1}$	$\{0.1, 0.5\}$	$\{0.6, 0.8, 1\}$	$\{0.3\}$
$\mathcal{T}_{t2}$	$\{0.1, 0.5\}$	$\{0.6, 0.8, 1\}$	$\{0.1, 0.3\}$



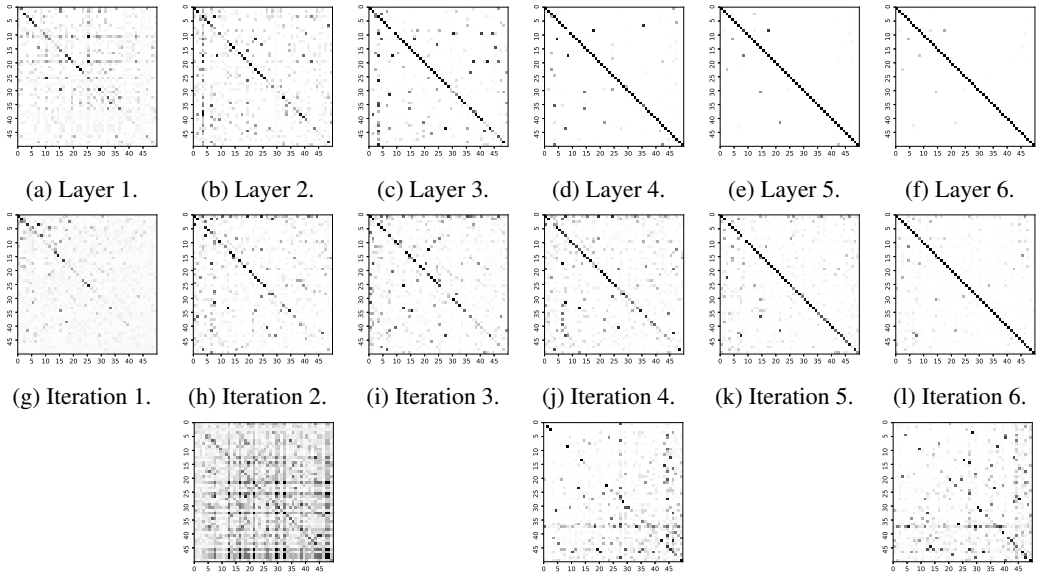
(a) $s = 0.8,\ \theta = 0.05$. (b) $p = 0.08,\ \theta = 0.05$. (c) $p = 0.04,\ s = 0.8$.

Figure 9: Performance comparison of SeedGNN trained with different training sets. Fix $n = 500$.

C.3 LAYER-WISE STUDY OF SEEDGNN

Recall from Section 4.3 that our design on the feature combination potentially enables SeedGNN to utilize various types of witness information adaptively, based on the confidence levels of new seeds z_l. In this section, we verify this capability through numerical results. To directly visualize z_l in the matching process, we present the similarity matrix Y_l of each layer of SeedGNN and compare it with the witness matrix of the iterative 1-hop and 2-hop algorithms at each iteration. We assume that the true mapping π is the identity permutation, i.e., $\pi(i) = i$.

First, we study the matching process in *dense* graphs. We fix a pair of correlated Erdős-Rényi graphs with $n = 50$, $p = 0.4$, $s = 0.8$ and $\theta = 0.1$. Then, we index the nodes from 0 to 49 in the descending order of the node degree in the parent graph $\mathcal{G}_0$. In Figure 10, we show the similarity matrix Y_l in each layer of our SeedGNN, and compare it with the witness matrix in each iteration using either the 1-hop or 2-hop algorithm. We can immediately see that the similarity matrices provided by SeedGNN are more similar to the witness matrices of the iterative 1-hop algorithm than that of the iterative 2-hop algorithm. Specifically, since the graphs are dense, the 1-hop witness information from the initial seeds can already generate new seeds with high confidence (see Figure 10(a) and 10(g), where there are many dark points on the diagonal (i.e., consistent with the underlying true mapping), while there are few dark points off the diagonal). The iterative 1-hop algorithm is known to use new 1-hop witnesses from these new seeds (see Figure 10(h)) in the next iteration. In contrast, the 2-hop witnesses from the initial seeds are much noisier (see Figure 10(m), where the darkness of the points on the diagonal cannot be differentiated from those off the diagonal). As we illustrated in Figure 4, these two types of witness information are both contained in the second layer of SeedGNN. By comparing Figure 10(b) with Figure 10(h) and Figure 10(m), we can observe that the second layer of SeedGNN produces a similarity matrix that is closer to the witness matrix of the 1-hop algorithm than that of the 2-hop algorithm. Thus, we infer that, for these dense graphs in which the new seeds are reliable, the SeedGNN relies more on witnesses computed from these new seeds.



(a) Layer 1. (b) Layer 2. (c) Layer 3. (d) Layer 4. (e) Layer 5. (f) Layer 6.

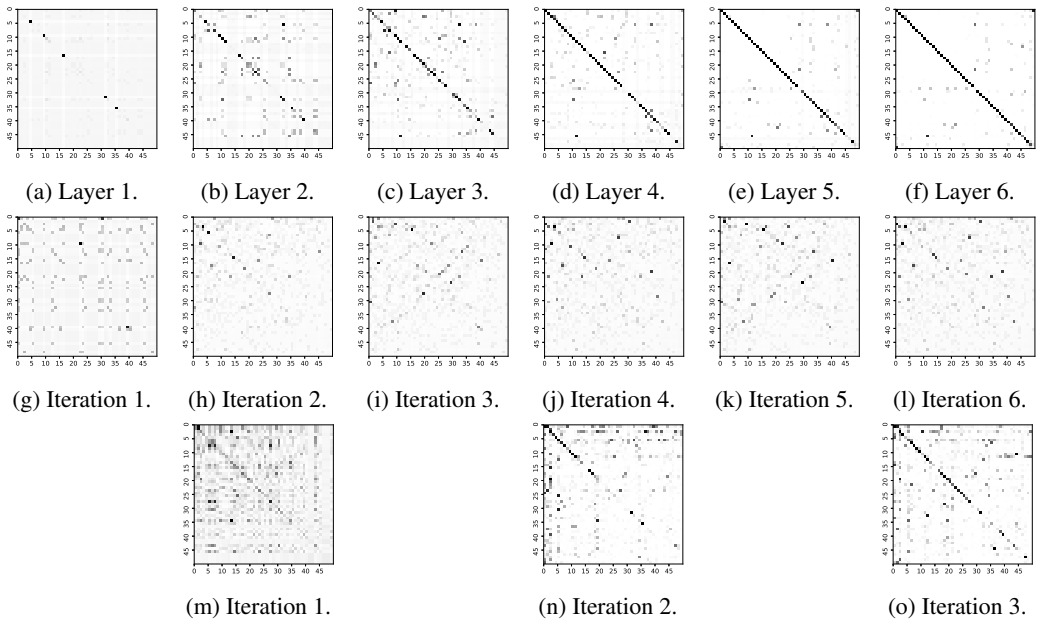
(g) Iteration 1. (h) Iteration 2. (i) Iteration 3. (j) Iteration 4. (k) Iteration 5. (l) Iteration 6.

(m) Iteration 1. (n) Iteration 2. (o) Iteration 3.

Figure 10: The similarity/witness matrices of the matching process on a fixed pair of dense correlated Erdős-Rényi graphs with $n = 50$, $p = 0.4$, $s = 0.8$ and $\theta = 0.1$. Darker points correspond to higher similarity (in Y_l) or a larger number of witnesses. Figure 10(a) — Figure 10(f) are the similarity matrix from each layer of SeedGNN. Figure 10(g) — Figure 10(l) are the witness matrix from each iteration of the iterative 1-hop algorithm. Figure 10(m) — Figure 10(o) are the witness matrix from each iteration of the iterative 2-hop algorithm.

Then, we study the matching process in *sparse* graphs. We fix a pair of correlated Erdős-Rényi graphs with $n = 50$, $p = 0.1$, $s = 0.8$ and $\theta = 0.1$. Then, we also index the nodes from 0 to 49 in

the descending order of the node degree in the parent graph $\mathcal{G}_0$. In Figure 11, we show the similarity matrix Y_l in each layer of our SeedGNN, and compare it with the witness matrix in each iteration using either the 1-hop or 2-hop algorithm. In contrast to Figure 10, in this case, we observe that the similarity matrices provided by SeedGNN are more similar to the witness matrices of the iterative 2-hop algorithm than those of the iterative 1-hop algorithm. Specifically, since the graphs are sparse, there are very few 1-hop witnesses even for true pairs. Thus, the 1-hop algorithm almost fails completely (see Figure 11(g) — Figure 11(l)). On the contrary, the 2-hop witnesses from the initial seeds are much more reliable (see Figure 11(m)). As a result, the iterative 2-hop algorithm produces much better results (see Figure 11(m) — Figure 11(o)). By comparing Figure 11(b) with Figure 11(h) and Figure 11(m), we can observe that the second layer of SeedGNN produces a similarity matrix that is closer to the witness matrix of the 2-hop algorithm than that of the 1-hop algorithm. Thus, we can infer that, for these sparse graphs in which the confidence levels of new seeds are low, SeedGNN utilizes 2-hop witness information from the initial seeds, and avoids using 1-hop witnesses based on these new seeds.



(a) Layer 1. (b) Layer 2. (c) Layer 3. (d) Layer 4. (e) Layer 5. (f) Layer 6.

(g) Iteration 1. (h) Iteration 2. (i) Iteration 3. (j) Iteration 4. (k) Iteration 5. (l) Iteration 6.

(m) Iteration 1. (n) Iteration 2. (o) Iteration 3.

Figure 11: The similarity/witness matrices of the matching process on a fixed pair of sparse correlated Erdős-Rényi graphs with $n = 50$, $p = 0.1$, $s = 0.8$ and $\theta = 0.1$. Darker points correspond to higher similarity (in Y_l) or a larger number of witnesses. Figure 11(a) — Figure 11(f) are the similarity matrix from each layer of SeedGNN. Figure 11(g) — Figure 11(l) are the witness matrix from each iteration of the iterative 1-hop algorithm. Figure 11(m) — Figure 11(o) are the witness matrix from each iteration of the iterative 2-hop algorithm.

In summary, from these two case studies, we conclude that our SeedGNN might be able to choose the appropriate features for different types of graphs according to the confidence level of new seeds. Further, we observe that the matching accuracy of SeedGNN is even higher than that of the 1-hop and 2-hop algorithms, the latter two of which have been theoretically proven to work well for dense graphs and sparse graphs, respectively (Mossel et al., 2019). Thus, this result suggests that SeedGNN may extract more valuable features, or learn more effective ways to synthesize witness information, than the theoretical algorithms.

D Performance Comparison with GNN Methods

In this section, we further compare the SeedGNN with several state-of-the-art deep graph matching networks, including semi-supervised learning methods (**PALE** (Man et al., 2016), **DeepLink** (Zhou et al., 2018), **dName** (Zhou et al., 2019), **CrossMNA** (Chu et al., 2019), **MGCN** (Chen et al., 2020)) and supervised learning methods (**GMN** (Zanfir et al., 2018),**PCA-GM** (Wang et al., 2019),

NGM (Wang et al., 2021), **IPCA-GM** (Wang et al., 2020a), **CIE** (Yu et al., 2019), **GLMNet** (Jiang et al., 2022), **LCS** (Wang et al., 2020b), **DGMC** (Fey et al., 2020), **BB-GM** (Rolínek et al., 2020), **DGM** (Gao et al., 2021), **DLGM** (Yu et al., 2021c)). We conduct experiments on Willow Object dataset (Cho et al., 2013), which consists of 256 images in 5 categories. The training set contains all categories of images, with 20 images of each category. We test the trained models on the rest images. Following the experimental setups in (Fey et al., 2020), we construct graphs via the Delaunay triangulation of keypoints, and the input features of keypoints are given by the concatenated output of relu4_2 and relu5_1 of a pre-trained VGG16 (Simonyan & Zisserman, 2015). Note that the resulting graphs only have 10 nodes. For semi-supervised methods, we randomly choose 5 true pairs as seeds. For supervised methods, they do not need seeds. For SeedGNN, we still directly use the model trained in Section 5.1. We generate the seeds in two ways. The first way is to apply the Hungarian algorithm on the similarities of non-topological node features. The second way is to use the matching result of the GNN methods for seedless graph matching (we choose DGMC). Note that for both ways, our SeedGNN does not utilize any training graph in the Willow Object dataset. For the semi-supervised algorithms, we use the publicly available implementations from their respective papers to generate the corresponding matching results. The performance values of the existing supervised algorithms are directly retrieved from their respective papers.

Since there are lack of sufficient training data for semi-supervised methods (there are only 5 seeds for each pair of graphs), it is difficult for them to learn to match the seeds effectively. As a result, we observe in Table 3 that SeedGNN significantly outperforms the semi-supervised methods. In contrast, the supervised methods learn from a large number of graph pairs. Further, the two images to be matched are of the same category. Therefore, the input node feature generated are similar and informative enough for correlating keypoints. Thus, the supervised methods have performed relatively well, and SeedGNN using seeds generated by the non-topological node features does not achieve performance gain. However, we can use SeedGNN to refine the output of seedless graph matching algorithms. We observe that SeedGNN consistently improves the matching performance of DGMC and achieves the best performance.

Table 3: Comparison of matching accuracy (%) on Willow Object dataset. The best results are marked as bold. The performance values of the existing supervised algorithms are directly retrieved from their respective papers.

	Method	face	mbike	car	duck	wbottle	Mean
Semi-Supervised	PALE (Man et al., 2016)	85.4	52.9	55.1	56.4	68.1	60.1
	DeepLink (Zhou et al., 2018)	86.1	55.8	63.7	62.0	72.3	66.0
	dName (Zhou et al., 2019)	86.9	58.3	65.3	66.0	77.7	68.8
	CrossMNA (Chu et al., 2019)	85.6	60.1	61.4	65.8	74.2	68.0
	MGCN (Chen et al., 2020)	87.2	63.0	67.5	67.2	78.1	72.6
Supervised	GMN (Zanfir et al., 2018)	98.1	65.0	72.9	74.3	70.5	76.2
	PCA-GM (Wang et al., 2019)	**100.0**	76.7	84.0	93.5	96.9	90.2
	NGM (Wang et al., 2021)	99.2	82.1	84.1	77.4	93.5	87.2
	IPCA-GM (Wang et al., 2020a)	**100.0**	77.7	90.2	84.9	95.2	89.6
	CIE (Yu et al., 2019)	**100.0**	90.0	82.2	81.2	97.6	90.2
	GLMNet (Jiang et al., 2022)	**100.0**	89.7	93.6	85.4	93.4	92.4
	LCS (Wang et al., 2020b)	**100.0**	99.4	91.2	86.2	97.9	94.9
	DGMC (Fey et al., 2020)	**100.0**	92.1	90.3	89.0	97.1	93.7
	BB-GM (Rolínek et al., 2020)	**100.0**	98.9	95.7	93.1	99.1	97.4
	DGM (Gao et al., 2021)	**100.0**	98.8	98.0	92.8	99.0	97.7
	DLGM (Yu et al., 2021c)	**100.0**	99.3	96.5	93.7	**99.3**	97.8
	SeedGNN (ours)	**100.0**	98.9	98.0	93.1	98.7	97.7
	DGMC+ SeedGNN	**100.0**	**99.6**	**100.0**	**99.7**	99.1	**99.5**