Sampling Enclosing Subgraphs for Link Prediction

Paul Louis∗
Ontario Tech University
Oshawa, Ontario, Canada
paul.louis@ontariotechu.net

Shweta Ann Jacob∗
Ontario Tech University
Oshawa, Ontario, Canada
shweta.jacob@ontariotechu.net

Amirali Salehi-Abari
Ontario Tech University
Oshawa, Ontario, Canada
abari@ontariotechu.ca

ABSTRACT

Link prediction is a fundamental problem for graph-structured data (e.g., social networks, drug side-effect networks, etc.). Graph neural networks have offered robust solutions for this problem, specifically by learning the representation of the subgraph enclosing the target link (i.e., pair of nodes). However, these solutions do not scale well to large graphs as extraction and operation on enclosing subgraphs are computationally expensive. This paper presents a scalable link prediction solution, that we call ScaLed, which utilizes sparse enclosing subgraphs to make predictions. To extract sparse enclosing subgraphs, ScaLed takes multiple random walks from a target pair of nodes, then operates on the sampled enclosing subgraph induced by all visited nodes. By leveraging the smaller sampled enclosing subgraph, ScaLed can scale to larger graphs with much less overhead while maintaining high accuracy. Through comprehensive experiments, we have shown that ScaLed can produce comparable accuracy to those reported by the existing subgraph representation learning frameworks while being less computationally demanding.

CSCS CONCEPTS

• Computing methodologies → Neural networks; Learning latent representations.

KEYWORDS

Link Prediction, Graph Neural Networks, Subgraph Sampling

ACM Reference Format:
Paul Louis, Shweta Ann Jacob, and Amirali Salehi-Abari. 2022. Sampling Enclosing Subgraphs for Link Prediction. In Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM ’22), October 17–21, 2022, Atlanta, GA, USA. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557688

1 INTRODUCTION

Graph-structured data such as user interactions, collaborations, protein-protein interactions, drug-drug interactions are prevalent in natural and social sciences. Link prediction—a fundamental problem on graph-structured data—intends to quantify the likelihood of a link (or interaction) occurring between a pair of nodes (e.g., proteins, drugs, etc.). Link prediction has many diverse applications such as predicting drug side effects, drug-repurposing [11], understanding molecule interactions [16], and recommender systems [7, 38].

Many solutions to link prediction problem [22, 24–26, 34] has been proposed ranging from simple heuristics (e.g., common neighbors, Adamic-Adar [1]), Katz [17]) to graph neural networks (GNNs) [5, 6, 14, 19, 28, 31, 43]. Among these solutions, GNNs [12, 35, 46] have emerged as the promising solution for learning rich latent representations of graph data to tackle link prediction. The early GNNs focused on shallow encoders [10, 30] in which the latent nodes’ representations were first learnt through a sequence of random walks, and then a likelihood of a link is determined by combining its two-end nodes’ latent representations. However, these shallow encoders were limited by not incorporating nodal features and incompatibility with inductive settings. These two challenges were (partially) addressed with message-passing graph neural networks [13, 20, 36]. These advancements motivate the research on determining and extending the expressive power of GNNs [3, 9, 39–41, 44] for all downstream tasks of link prediction, node classification, and graph classification. For link prediction, subgraph-based representation learning (SGRL) methods [5, 6, 23, 28, 43]—by learning the enclosing subgraphs around the two-end nodes rather than independently learning two end-node’s embedding—have improved GNNs’ expressive power, and offered state-of-the-art solutions. However, these solutions suffer from the lack of scalability to large-scale graphs. This is primarily due to the computation overhead in extracting, preprocessing, and learning (large) enclosing subgraphs.

We introduce Sampling Enclosing Subgraphs for Link Prediction (ScaLed) to extend SGRL methods and enhance their scalability. ScaLed samples enclosing subgraphs using a sequence of random walks. This sampling reduces the computational overhead of large subgraphs while maintaining the key structural information. ScaLed can be integrated into any GNN, and also offers parallelizability and model compression that can be exploited for large-scale graphs. The two hyperparameters, walk length and number of walks, in ScaLed provide a way to control the trade-off between scalability and accuracy, if needed. Our extensive experiments on real-world datasets demonstrate that ScaLed produces comparable results to the state-of-the-art methods (e.g. SEAL [43]) in link prediction, but requiring magnitudes less training data, time, and memory.

2 LINK PREDICTION

We consider an undirected graph $G = (V, E, A)$ where $V = [n]$ is the set of $n$ nodes (e.g., individuals, proteins, etc), $E \subseteq V \times V$ represents the edge set (e.g., friendship relations or protein-to-protein interactions) and the tensor $A \in \mathbb{R}^{n \times m \times d}$ contains all nodes’ attributes (e.g., user profiles) and edges’ attributes (e.g. the strength or type of interactions). For each node $v \in V$, its attributes (if any) are stored in the diagonal component $A_{vv}$, while the off-diagonal

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CIKM ’22, October 17–21, 2022, Atlanta, GA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9236-5/22/10...
https://doi.org/10.1145/3511808.3557688
component $A_{uv}$ can have the attributes of an edge $(u, v)$ if $(u, v) \in E$; otherwise $A_{uv} = 0$.

**Link Prediction Problem.** Our goal in link prediction is to infer the presence or absence of an edge between a pair of target nodes given the observed tensor $A$. The learning problem is to find a likelihood (or scoring) function $f$ such that it assigns interaction likelihood (score) $\hat{A}_{uv}$ to each target pair of nodes $(u, v) \notin E$, whose relationships to each other are not observed. Larger $\hat{A}_{uv}$ indicates a higher chance of $(u, v)$ forming a link or missing a link. The function $f$ can be formulated as $\hat{A}_{uv} = f(u, v, A; \theta)$ with $\theta$ denoting the model parameters. Most link prediction methods differ from each other in the formulation of the likelihood function $f$ and its assumptions. The function $f$ can be some parameter-free predefined heuristics [1, 17, 27] or learned by a graph neural network [13, 20, 32, 36] or any other deep learning framework [37]. The likelihood function formulation also varies based on its computation requirement on the maximum hop of neighbors of target nodes. For example, first-order heuristics (e.g., common neighbors and preferential attachment [2]) only require the direct neighbors while graph neural networks methods [15, 19] and high-order heuristics (e.g., Katz [17], rooted PageRank [4]) require knowledge of the entire graph.

3 THE ScaLed MODEL

After describing the SEAL link prediction model and its variants, we detail how our proposed ScaLed model extends these models to maintain their prediction power but offer better scalability.

**SEAL and its variants.** Rather than learning the target nodes’ embeddings independently (as with Graph Convolutional Network [20] or GraphSAGE [13]), SEAL [43] focuses on learning the enclosing subgraph of a pair of target nodes to capture their relative positions to each other in the graph:

**Definition 1 (Enclosing Subgraph [43].** Given a graph $G$, the $h$-hop enclosing subgraph around target nodes $(u, v)$ is the subgraph $G_{uv}^h$ induced from $G$ with the set of nodes \( \{j | d(j, x) \leq h \text{ or } d(j, y) \leq h\} \), where $d(i, j)$ is the geodesic distance between node $i$ and $j$.

In SEAL, for each pair of the target nodes $(u, v)$, their enclosing subgraph $G_{uv}^h$ is found with two $h$-hop Breadth-First Search (BFS), where each BFS starts from $u$ and $v$. The nodes in the enclosing subgraph are also augmented with labels indicating their distances to the target pair of nodes using the Double-Radius Node Labeling (DRNL) hash function [43]:

\[
DRNL(x, G_{uv}^h) = 1 + \min(d_{ux}, d_{xu}) + \lfloor d'/2 \rfloor \lfloor d'/2 - 1 \rfloor,\]

where $x$ represents the nodes in the subgraph $G_{uv}^h$, $d_{ux}$ is the geodesic distance of $x$ to $u$ in $G_{uv}^h$, when node $v$ is removed, and $d' = d_{ux} + d_{xu}$. Note that the distance of $x$ to each target node $u$ is calculated in isolation by removing the other target node $v$ from the subgraph. The target nodes are given the label 1 and a node with $\infty$ distance to at least one of the target nodes is given the label 0. Each node label is then represented by its one-hot encoding, and expands the initial node features, if any. The subgraph $G_{uv}^h$ along with the augmented nodal features is fed into a graph neural network, which predicts the presence or absence of the edge. In SEAL, the link prediction is treated as a binary classification over the enclosing subgraphs by determining if the enclosing subgraph will be closed by a link between the pair of target nodes or not. Thus, SEAL uses a graph pooling mechanism (e.g., SortPooling [44]) to compute the enclosing subgraph representation for the classification task. Other variants of SEAL (e.g., DE-GNN [23] and WalkPool [28]) have replaced either its DRNL labeling method [23] or graph aggregation method [28] with other alternatives to improve its expressiveness power. However, SEAL and these variants suffer from the scalability issue as the subgraph size grows exponentially with the hop size $h$, and large-degree nodes (e.g., celebrities) possess large enclosing subgraphs even for a small $h$. To address these scalability issues, we propose Sampling Enclosing Subgraphs for Link Prediction (ScaLed).

**ScaLed.** Observing that the computational bottleneck of SEAL and its variants originates from the exponential growth and the size of enclosing subgraphs, we propose Sampled Enclosing Subgraphs with more tractable sizes:

**Definition 2 (Random Walk Sampled Enclosing Subgraph).** Given a graph $G$, the random-walk sampled $h$-hop enclosing subgraph around target nodes $(u, v)$ is the subgraph $G_{uv}^{h,k}$ induced from $G$ with the set of nodes $V_{uv}^{h,k} \in W_v^{h,k} \cup W_u^{h,k}$, where $W_{i}^{h,k}$ is the set of nodes visited by $k$ many $h$-length random-walk($s$) from node $i$.

Figure 1(b) illustrates sampled enclosing subgraph of the target pair $(u, v)$ for the original graph in Figure 1(a), where $h = 2$ and $k = 2$. Here, $W_v^{h,k} = \{a, d, e, f, g\}$ and $W_u^{h,k} = \{a, b, c, e, f, g, u, v\}$. The included subgraph in Figure 1(b) contains all nodes and edges between nodes in $V_{uv}^{h,k}$.

Comparing Definitions 1 and 2, a few important observations can be made: (i) the sampled enclosing subgraph $G_{uv}^{h,k}$ is the subgraph of the enclosing subgraph $G_{uv}^h$, as the $h$-length random walks cannot reach a node further than $h$-hop away from the starting node; (ii) the size of the sampled subgraph is bounded to $O(hk)$ and controlled by these two parameters compared to the exponential growth of enclosing subgraphs with $h$ in Definition 1. These two observations highlight that ScaLed, by replacing the dense enclosing subgraphs with their sparse (sub)graphs, offers scalability. ScaLed also offers flexibility to control the extent of sparsity and scalability with its sampling parameters $h$ and $k$.

The ScaLed model can use any labeling trick (e.g., DRNL, zero-one labeling, etc.) [45] to encode the distances between target nodes and other nodes in the sampled subgraphs; see Figure 1(b) for an example. Similar to SEAL, the one-hot encoding of the distance labels along with the nodal features (if any) of the nodes in the sampled subgraph are fed into a graph neural network with graph.
We run extensive experiments to compare the prediction accuracy. We leave further theoretical analysis as future work.

1

Table 1: The statistics of experimented datasets.

| Dataset   | # Nodes | # Edges | Avg. Deg. | # Features |
|-----------|---------|---------|-----------|------------|
| USAir     | 332     | 2126    | 12.81     | NA         |
| Celegans  | 297     | 2148    | 14.46     | NA         |
| NS        | 1461    | 2742    | 3.75      | NA         |
| Router    | 5022    | 6258    | 2.49      | NA         |
| Power     | 4941    | 6594    | 2.67      | NA         |
| Yeast     | 2375    | 11693   | 9.85      | NA         |
| Ecoli     | 1805    | 14660   | 16.24     | NA         |
| PB        | 1222    | 16714   | 27.36     | NA         |
| Cora      | 2708    | 5429    | 4         | 1433       |
| CiteSeer  | 3327    | 4732    | 2.84      | 3703       |

5

Table 2 reports the average AUC over five runs for each model in each run, we test it against testing data with those parameters which achieve highest AUC on validation data. For computational measurements, we also report average training plus inference time, allocated CUDA memory, model size, and number of parameters.

Results: Resource Consumption. Table 3 reports the average consumption of resources over five runs for ScaLed and SEAL. For all datasets, the average runtime of ScaLed is much lower for larger datasets (e.g., Ecoli and PB), but slightly lower for small datasets (USAir and Celegans). For Ecoli and PB, ScaLed gains speed up of 1.90× and 1.69× over SEAL, while using up to 20× less allocated GPU memory, model size, and parameters. The sampled subgraphs in ScaLed are sparser than that of SEAL (compare the number of nodes and edges in Table 3). ScaLed requires 7.86× and 5.17× less edges for Cora and CiteSeer, respectively. This compression can be up to 32.18× (see PB). The results in Tables 2 and 3 confirm our hypothesis that ScaLed is able to match the performance of SEAL with much less computational overhead. We even witness that ScaLed has outperformed SEAL for NS and Yeast while consuming 1.51× and 6.35× less edges in the sampled enclosing subgraphs. These results suggest that random walk based subgraph sampling is beneficial for the learning without compromising the accuracy. Random-walk sampling enables inclusion of both local and global neighborhoods around target nodes while keeping a low memory profile. Finally, the results on larger and denser datasets such as Ecoli and PB indicates that the largest computational efficiency gains are achieved by ScaLed on larger and denser datasets.

Results: Hyperparameter Sensitivity Analyses. We intend to understand how the walk length $h$ and the number of walks $k$ control the computational overhead in ScaLed. Thus, we conduct a sensitivity analysis of these two parameters on two of the largest and densest non-attributed datasets (i.e., Ecoli and PB) and on both of the attributed datasets (i.e., Cora and CiteSeer). We vary $h$ and $k$ from one-hot indicators. In MF, the nodal latent feature has 32 dimensions for each node. MF uses a 3-layered MLP with 32 hidden dimensions. For node2vec, we set sampling parameters $p = q = 1$ and a dimensionality of 32 for the node features. For SEAL, we set $h = 2$ for non-attributed datasets and $h = 3$ for attributed datasets. We also use a 3-layered DGCNN with a hidden dimensionality of 32 for all datasets. For ScaLed model, we set $k = 20$ while $h$ and all other hyperparameters are set the same as that of SEAL for fair comparison. The learning rate is set to 0.0001 for SEAL and ScaLed and 0.01 for node2vec, MF and GAE baselines. All learning models, for both attributed and non-attributed datasets, are trained for 50 epochs with a dropout of 0.5 (except for node2vec without dropout) and Adam [18] optimizer (except for node2vec with Sparse Adam). GAE baselines are trained by full-batch gradients; but others are trained with a batch size of 32.

Measurements. We report the mean of area under the curve (AUC) of the testing data over 5 runs with 5 random seeds. For each model, we intend to understand how the walk length $h$ and the number of walks $k$ control the computational overhead in ScaLed. Thus, we conduct a sensitivity analysis of these two parameters on two of the largest and densest non-attributed datasets (i.e., Ecoli and PB) and on both of the attributed datasets (i.e., Cora and CiteSeer). We vary $h$ and $k$ from one-hot indicators. In MF, the nodal latent feature has 32 dimensions for each node. MF uses a 3-layered MLP with 32 hidden dimensions. For node2vec, we set sampling parameters $p = q = 1$ and a dimensionality of 32 for the node features. For SEAL, we set $h = 2$ for non-attributed datasets and $h = 3$ for attributed datasets. We also use a 3-layered DGCNN with a hidden dimensionality of 32 for all datasets. For ScaLed model, we set $k = 20$ while $h$ and all other hyperparameters are set the same as that of SEAL for fair comparison. The learning rate is set to 0.0001 for SEAL and ScaLed and 0.01 for node2vec, MF and GAE baselines. All learning models, for both attributed and non-attributed datasets, are trained for 50 epochs with a dropout of 0.5 (except for node2vec without dropout) and Adam [18] optimizer (except for node2vec with Sparse Adam). GAE baselines are trained by full-batch gradients; but others are trained with a batch size of 32.

Measurements. We report the mean of area under the curve (AUC) of the testing data over 5 runs with 5 random seeds. For each model, we intend to understand how the walk length $h$ and the number of walks $k$ control the computational overhead in ScaLed. Thus, we conduct a sensitivity analysis of these two parameters on two of the largest and densest non-attributed datasets (i.e., Ecoli and PB) and on both of the attributed datasets (i.e., Cora and CiteSeer). We vary $h$ and $k$ from one-hot indicators. In MF, the nodal latent feature has 32 dimensions for each node. MF uses a 3-layered MLP with 32 hidden dimensions. For node2vec, we set sampling parameters $p = q = 1$ and a dimensionality of 32 for the node features. For SEAL, we set $h = 2$ for non-attributed datasets and $h = 3$ for attributed datasets. We also use a 3-layered DGCNN with a hidden dimensionality of 32 for all datasets. For ScaLed model, we set $k = 20$ while $h$ and all other hyperparameters are set the same as that of SEAL for fair comparison. The learning rate is set to 0.0001 for SEAL and ScaLed and 0.01 for node2vec, MF and GAE baselines. All learning models, for both attributed and non-attributed datasets, are trained for 50 epochs with a dropout of 0.5 (except for node2vec without dropout) and Adam [18] optimizer (except for node2vec with Sparse Adam). GAE baselines are trained by full-batch gradients; but others are trained with a batch size of 32.

Measurements. We report the mean of area under the curve (AUC) of the testing data over 5 runs with 5 random seeds. For each model, we intend to understand how the walk length $h$ and the number of walks $k$ control the computational overhead in ScaLed. Thus, we conduct a sensitivity analysis of these two parameters on two of the largest and densest non-attributed datasets (i.e., Ecoli and PB) and on both of the attributed datasets (i.e., Cora and CiteSeer). We vary $h$ and $k$ from one-hot indicators. In MF, the nodal latent feature has 32 dimensions for each node. MF uses a 3-layered MLP with 32 hidden dimensions. For node2vec, we set sampling parameters $p = q = 1$ and a dimensionality of 32 for the node features. For SEAL, we set $h = 2$ for non-attributed datasets and $h = 3$ for attributed datasets. We also use a 3-layered DGCNN with a hidden dimensionality of 32 for all datasets. For ScaLed model, we set $k = 20$ while $h$ and all other hyperparameters are set the same as that of SEAL for fair comparison. The learning rate is set to 0.0001 for SEAL and ScaLed and 0.01 for node2vec, MF and GAE baselines. All learning models, for both attributed and non-attributed datasets, are trained for 50 epochs with a dropout of 0.5 (except for node2vec without dropout) and Adam [18] optimizer (except for node2vec with Sparse Adam). GAE baselines are trained by full-batch gradients; but others are trained with a batch size of 32.

Measurements. We report the mean of area under the curve (AUC) of the testing data over 5 runs with 5 random seeds. For each model, we intend to understand how the walk length $h$ and the number of walks $k$ control the computational overhead in ScaLed. Thus, we conduct a sensitivity analysis of these two parameters on two of the largest and densest non-attributed datasets (i.e., Ecoli and PB) and on both of the attributed datasets (i.e., Cora and CiteSeer). We vary $h$ and $k$ from one-hot indicators. In MF, the nodal latent feature has 32 dimensions for each node. MF uses a 3-layered MLP with 32 hidden dimensions. For node2vec, we set sampling parameters $p = q = 1$ and a dimensionality of 32 for the node features. For SEAL, we set $h = 2$ for non-attributed datasets and $h = 3$ for attributed datasets. We also use a 3-layered DGCNN with a hidden dimensionality of 32 for all datasets. For ScaLed model, we set $k = 20$ while $h$ and all other hyperparameters are set the same as that of SEAL for fair comparison. The learning rate is set to 0.0001 for SEAL and ScaLed and 0.01 for node2vec, MF and GAE baselines. All learning models, for both attributed and non-attributed datasets, are trained for 50 epochs with a dropout of 0.5 (except for node2vec without dropout) and Adam [18] optimizer (except for node2vec with Sparse Adam). GAE baselines are trained by full-batch gradients; but others are trained with a batch size of 32.
### Table 2: Average AUCs for all datasets and models. The best and second best are shaded in dark and light gray respectively.

| Dataset | Model | Time | CUDA | Size | Params. | # Nodes | # Edges |
|---------|-------|------|------|------|---------|---------|---------|
| USAir   | SEAL  | 486  | 52 MB| 2.04 MB | 0.533 M | 207 | 2910 |
|         | ScaLed| 446  | 11 MB| 0.44 MB | 0.113 M | 40 | 2580 |
| Celegans| SEAL  | 473  | 47 MB| 1.84 MB | 0.480 M | 208 | 2482 |
|         | ScaLed| 453  | 7 MB | 0.43 MB | 0.117 M | 45 | 293 |
| NS      | SEAL  | 580  | 5 MB | 0.22 MB | 0.056 M | 17 | 83 |
|         | ScaLed| 572  | 3 MB | 0.19 MB | 0.048 M | 12 | 55 |
| Router  | SEAL  | 1330 | 38 MB| 0.55 MB | 0.144 M | 82 | 253 |
|         | ScaLed| 1342 | 5 MB | 0.27 MB | 0.068 M | 21 | 54 |
| Power   | SEAL  | 1394 | 4 MB | 0.22 MB | 0.056 M | 16 | 33 |
|         | ScaLed| 1404 | 3 MB | 0.20 MB | 0.052 M | 13 | 25 |
| Yeast   | SEAL  | 2695 | 65 MB| 1.38 MB | 0.362 M | 151 | 2438 |
|         | ScaLed| 2482 | 13 MB| 0.39 MB | 0.101 M | 35 | 584 |
| Ecoli   | SEAL  | 6044 | 331 MB| 10.22 MB| 2.68 M  | 1166 | 21075 |
|         | ScaLed| 3181 | 20 MB| 0.50 MB | 0.130 M | 46 | 790 |
| PB      | SEAL  | 6167 | 312 MB| 6.41 MB | 1.68 M  | 729 | 20952 |
|         | ScaLed| 3649 | 15 MB| 0.57 MB | 0.149 M | 57 | 652 |
| CiteSeer| SEAL  | 1941 | 221 MB| 0.79 MB | 0.233 M | 82 | 320 |
|         | ScaLed| 1944 | 49 MB| 0.72 MB | 0.187 M | 22 | 63 |
| Cora    | SEAL  | 1731 | 195 MB| 1.78 MB | 0.466 M | 202 | 692 |
|         | ScaLed| 1199 | 27 MB| 0.53 MB | 0.140 M | 32 | 88 |

### Table 3: Avg. computational consumption of SEAL vs. ScaLed over five runs: runtime in seconds, max allocated CUDA and model size in Megabytes (MB), number of parameters in Millions. Maximum ratio corresponds to the maximum of ratio of SEAL’s resource over ScaLed resource (in bold).

| Dataset | Model | Time | CUDA | Size | Params. | # Nodes | # Edges |
|---------|-------|------|------|------|---------|---------|---------|
| USAir   | SEAL  | 486  | 52 MB| 2.04 MB | 0.533 M | 207 | 2910 |
|         | ScaLed| 446  | 11 MB| 0.44 MB | 0.113 M | 40 | 2580 |
| Celegans| SEAL  | 473  | 47 MB| 1.84 MB | 0.480 M | 208 | 2482 |
|         | ScaLed| 453  | 7 MB | 0.43 MB | 0.117 M | 45 | 293 |
| NS      | SEAL  | 580  | 5 MB | 0.22 MB | 0.056 M | 17 | 83 |
|         | ScaLed| 572  | 3 MB | 0.19 MB | 0.048 M | 12 | 55 |
| Router  | SEAL  | 1330 | 38 MB| 0.55 MB | 0.144 M | 82 | 253 |
|         | ScaLed| 1342 | 5 MB | 0.27 MB | 0.068 M | 21 | 54 |
| Power   | SEAL  | 1394 | 4 MB | 0.22 MB | 0.056 M | 16 | 33 |
|         | ScaLed| 1404 | 3 MB | 0.20 MB | 0.052 M | 13 | 25 |
| Yeast   | SEAL  | 2695 | 65 MB| 1.38 MB | 0.362 M | 151 | 2438 |
|         | ScaLed| 2482 | 13 MB| 0.39 MB | 0.101 M | 35 | 584 |
| Ecoli   | SEAL  | 6044 | 331 MB| 10.22 MB| 2.68 M  | 1166 | 21075 |
|         | ScaLed| 3181 | 20 MB| 0.50 MB | 0.130 M | 46 | 790 |
| PB      | SEAL  | 6167 | 312 MB| 6.41 MB | 1.68 M  | 729 | 20952 |
|         | ScaLed| 3649 | 15 MB| 0.57 MB | 0.149 M | 57 | 652 |
| CiteSeer| SEAL  | 1941 | 221 MB| 0.79 MB | 0.233 M | 82 | 320 |
|         | ScaLed| 1944 | 49 MB| 0.72 MB | 0.187 M | 22 | 63 |
| Cora    | SEAL  | 1731 | 195 MB| 1.78 MB | 0.466 M | 202 | 692 |
|         | ScaLed| 1199 | 27 MB| 0.53 MB | 0.140 M | 32 | 88 |

### 5 CONCLUSION AND FUTURE WORKS

Link prediction is an important task for graph-structured data with applications spanning across multiple domains. Existing state-of-the-art link prediction methods use subgraph representation learning (SGRL), which learns the enriched embedding of the enclosing subgraphs around the pair of nodes. However, SGRL methods are not scalable to large real-world graphs. We proposed ScaLed to overcome this scalability shortcoming by exploiting random walks to sample sparse enclosing subgraphs. The main idea is to preserve the key structural information of subgraphs with less number of nodes and edges, thus yielding smaller computational graphs for GNNs which in turn reduces the runtime and memory consumption. Our extensive experiments demonstrate ScaLed can match the accuracy measures of the state-of-the-art link prediction while consuming order of magnitudes less resources. While larger datasets (e.g., OGB [15]) are not experimented, ScaLed could offer scalability by sparsifying their dense $h$-hop subgraphs. For future work, we plan to explore how to adaptively choose the length of the walks and the number of walks depending on the structural positions of two nodes. Another interesting research direction that could be explored is to apply graph augmentation techniques to the sampled subgraphs in ScaLed to further enhance its learning capabilities.
