Are under-studied proteins under-represented? How to fairly evaluate link prediction algorithms in network biology

Serhan Yılmaz [1,*], Kaan Yorgancioglu [1,*], Mehmet Koyutürk [1, 2]

[1]Department of Computer and Data Sciences, Case Western Reserve University

[2]Center for Proteomics and Bioinformatics, Case Western Reserve University

[*]Both authors contributed equally to this research.

serhan.yilmaz@case.edu, kaan.yorgancioglu@case.edu

Abstract

In the context of biomedical applications, new link prediction algorithms are continuously being developed and these algorithms are typically evaluated computationally, using test sets generated by sampling the edges uniformly at random. However, as we demonstrate, this creates a bias in the evaluation towards "the rich nodes", i.e., those with higher degrees in the network. More concerningly, we demonstrate that this bias is prevalent even when different snapshots of the network are used for evaluation as recommended in the machine learning community. This leads to a cycle in research where newly developed algorithms generate more knowledge on well-studied biological entities while the under-studied entities are commonly ignored. To overcome this issue, we propose a weighted validation setting focusing on under-studied entities and present strategies to facilitate bias-aware evaluation of link prediction algorithms. These strategies can help researchers gain better insights from computational evaluations and promote the development of new algorithms focusing on novel findings and under-studied proteins. We provide a web tool to assess the bias in evaluation data at: https://yilmazs.shinyapps.io/colipe/

Keywords: Benchmarking, Link prediction, Systems biology, Bias in machine learning, Graph embeddings, Fair evaluation, Under-studied proteins, Matthew's effect, Rich gets richer, Protein-Protein interactions, Preferential attachment, Under-represented proteins in evaluation

Acknowledgments: This work was supported in part by the National Library Of Medicine of the National Institutes of Health under award number R01-LM012980.
Conflict of interest statement: None declared.

Code and data availability: All materials (code and data) to reproduce the analyses and figures in the paper is available in figshare (doi:10.6084/m9.figshare.21330429). The code for the evaluation framework implementing the proposed strategies is available at github[†].

[†] https://github.com/serhan-yilmaz/colipe

Introduction

Background and related literature. In the context of network biology, link prediction is commonly applied to discover previously unknown associations or interactions [1]. Many biomedical prediction tasks are formulated as link prediction problems, including prediction of drug–disease associations (DDAs) [2], drug response prediction [3], disease gene prioritization [4], prediction of drug-drug interactions (DDI) [5], protein-protein interactions (PPIs) [6], transcription factor regulatory relationships [7], kinase-substrate associations [8], and kinase-kinase interactions [9].

Early research on link prediction focused on computing a score to assess the likelihood of the existence of an edge between two nodes [10]. These include local measures based on guilt-by-association, including common neighbors and preferential attachment [11]. Global approaches, such as random walks, generalize this principle to the notion that nodes that are "proximate" are likely to acquire an edge [12]. More recently, graph embedding models, which map each node to a vector in a lower-dimensional embedding space, allow machine learning methods to be utilized seamlessly in link prediction [13, 14]. With the availability of various types of omic data, along with rapid advances in machine learning, more sophisticated learning algorithms, including graph convolutional networks, are increasingly applied to link prediction problems in systems biology [15, 16, 17].

Evaluation of link prediction algorithms. For evaluating link prediction algorithms, a recommended strategy is to perform the validation on an independent test dataset, using different snapshots of the network (e.g., taken from different data sources or different points in time) as training and test sets [18, 19]. In the absence of multiple snapshots, the evaluation is typically performed by generating training and test instances from a single network, sampling the edges to be removed from the network uniformly at random [14, 20, 21]. With the availability of more link prediction algorithms with ever-increasing sophistication, research on the evaluation of algorithms has also gained attention [1, 19, 22]. Although there has been significant attention to algorithmic bias and fairness [23] as well as the reproducibility and comparability of the results [24] in graph machine learning, studies investigating fairness and sources of bias in the evaluation of link prediction algorithms are relatively scarce, particularly in the context of network biology [25].

Motivation and significance in systems biology. For biological knowledge discovery, *fairness* can be considered as the ability to identify biological entities that are relatively less studied (e.g., when a scientist is looking to identify a kinase that phosphorylates a specific phosphorylation site they discovered, does the algorithm give equal consideration to all kinases regardless of how well-studied they are?). Matthew's effect (also known as "rich gets richer") is quite pronounced in biology - according to the Understudied Protein Initative that was announced in May 2022 [26], "95% of all life science publications focus on a group of 5,000 particularly well-studied human proteins". This effect is also a critical source of *bias* during the evaluation of link prediction algorithms in biology.

We [4, 27] and other groups [25] have documented the degree bias in biological networks and its consequences in the context of specific applications in network biology. However, little attention is paid to the effect of bias in evaluating new link prediction algorithms, leading to the development of algorithms that continuously reinforce what is already known about well-studied proteins [28]. In this paper, we show that *both the benchmarking data and standard evaluation techniques for link prediction favor well-studied biological entities.* Specifically, we demonstrate that (i) randomly sampling edges to generate a test set creates bias in which edges that connect high-degree nodes are over-represented, (ii) this bias also exists in settings that utilize different snapshots of a network as training/test sets as opposed to a randomized sampling. In turn, link prediction algorithms that make biased predictions are disproportionately rewarded for favoring high-degree nodes. This results in a serious barrier in making new discoveries involving under-represented biological entities.

Contributions of this study. We argue that successful prediction of interactions and associations that involve low-degree nodes can be more valuable as they can offer more insight about the biological mechanisms under study [26]. Therefore, the evaluation of a link-prediction algorithm in biology needs to account for degree bias throughout analysis. Motivated by this consideration, using prediction of protein-protein interactions (PPIs) as a case example, we first investigate the typical evaluation settings used in the literature. We demonstrate how the current evaluation settings incentivize algorithms to bring forward well-studied proteins in their predictions. To address this issue and faciliate the bias-aware evaluation of link prediction algorithms, we propose multiple strategies organized in five views: (i) quantifying the bias in predictions, (ii) quantifying bias in benchmarking data (and the incentive toward high-bias predictors), (iii) a weighted validation setting that aims to ensure that under-studied proteins are not under-represented in the evaluation, (iv) a stratified analysis that decomposes the prediction performance based on how well-studied the nodes are, and (v) a summary view to outline the main characteristics of an algorithm.

Finally, we survey additional problems to show that the issues we demonstrate in the context of PPI prediction generalize to other link prediction problems in biology: 1) kinase-substrate associations, 2) transcription factor-target interactions, 3) drug-drug interactions, 4) drug-disease associations. These results suggest that, for a broad range of problems in network biology, under-studied entities are severely under-represented in traditional evaluation settings. The proposed framework can be helpful to perform a balanced evaluation, facilitating the development of algorithms focusing on novel findings and new interactions between under-studied biological entities.

Results

Experimental setting. In this work, to bring to light some issues in standard evaluation settings that are a result of a severe imbalance in the gathered knowledge for biological networks, and to demonstrate the strategies we propose to resolve these issues, we primarily focus on the problem of PPI predictions and the human PPI network obtained from Biogrid [29]. As a final part of our analysis, we analyze a broad range of networks and problems in the context of biomedical applications to show that our observations on PPI network are generalizable to other domains. For simplicity, we mainly consider and refer to the nodes in a protein context, although the developed techniques are not specific to proteins or PPI network. For the evaluation, we obtain the required training/test sets either by random sampling of the edges or by utilizing multiple versions of the Biogrid network (taken at 2020 and 2022). Link prediction algorithms use these training portions of the network to produce prediction scores for pairs of nodes and to obtain a ranking for pairs that are most likely to have an interaction between. These rankings are then compared against the known interactions in the test set to evaluate the prediction performance of the algorithms.

Selected algorithms for the analysis. To select the link prediction algorithms (Figure 1(a)) for inclusion in our analysis, we consider two criteria: (i) to include representative methods for different classes of algorithms (e.g., scoring metrics, network propagation methods, embedding/machine learning based methods), and (ii) to include algorithms with differing levels of bias towards high-degree nodes. Namely, we include at least two versions from each category: One version that prioritizes high-degree nodes (high-bias methods) and another, normalized version with lower bias. For example, in the scoring metrics category, common neighbors is the high-bias version, whereas Jaccard index (intersection divided by union) is the normalized, lower-bias version. In both cases, the information source is the same (number of shared interactions), the only difference is the normalization based on node degrees, and therefore, disposition of the method toward high-degree nodes. For Deepwalk [13], a low-bias embedding algorithm that generates and uses feature vectors with a low correlation to the node degrees (S. Figure 1), we create a biased version (Deepwalk-withdegree) by adding node degree to the embeddings as an additional feature. Similarly, since L3 [6] is a

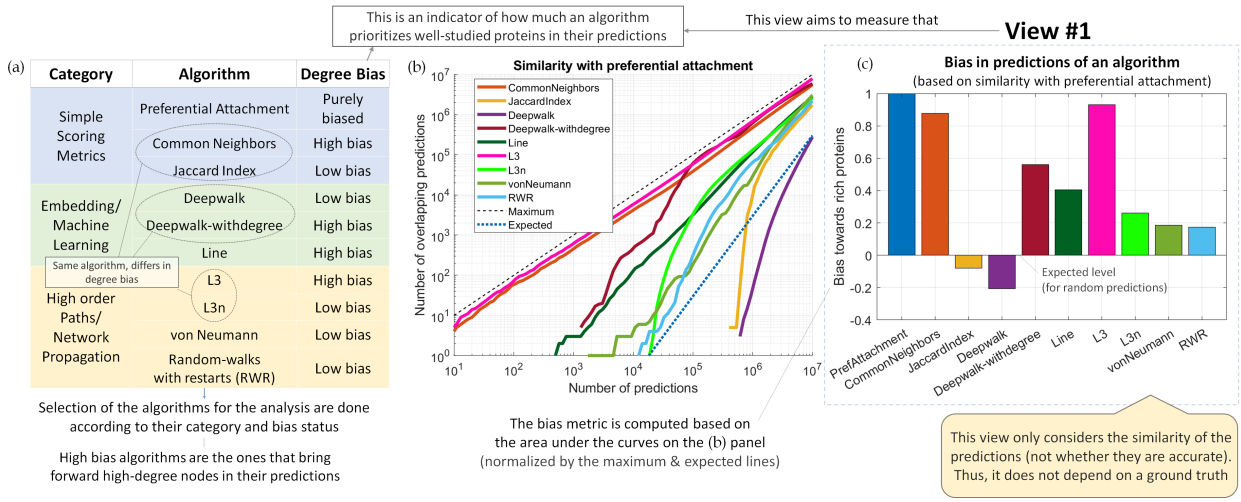


Figure 1: Quantifying the bias towards well-studied proteins in the predictions of an algorithm based on similarity with preferential attachment model on Biogrid PPI network. (a) The algorithms selected for the analysis, their categorization and affinities towards degree bias. (b) Overlap of the predictions of the algorithms with preferential attachment. (c) The quantified bias of the algorithms.

high bias algorithm (as it counts the paths of length 3 and only applies a soft normalization), we create a lower-bias version, L3n, by applying a stronger normalization based on node degrees. Besides these, we also consider preferential attachment (a purely-biased baseline that only considers node-degree information), LINE [30] a neural-network based embedding algorithm with high bias (since its learning process captures the node degree information in the embeddings, S. Figure 2), two network propagation algorithms von Neumann [31] and random walks with restarts (RWR) [32] (both with low-bias due to strong normalization based on node degrees in their formulation). For embedding algorithms, we train a logistic regression model to obtain the predictions.

Note that, our aim for the analysis (and the selection of algorithms) is not to determine the best performing method among the state-of-the-art methods for PPI prediction problem. Instead, our aim is to (i) investigate the benchmark data and evaluation process as a function of degree distribution, elucidating the effect of the imbalance in the network on commonly used evaluation settings, and (ii) demonstrate how these evaluation settings can reward the development of algorithms that are biased toward high-degree proteins, which often correspond to well-studied proteins [33].

View #1: Bias of link prediction algorithms toward high-degree proteins

- **Proposed strategy:** To understand the disposition of an algorithm toward well-studied proteins, measure its similarity with preferential attachment (biased baseline).
- Provides information about how much an algorithm prioritizes high-degree nodes. Node degree is considered an indicator of how well-studied a protein is.

Here, we aim to investigate and quantify the bias in the predictions of the algorithms toward well-studied proteins. For this purpose, we use preferential attachment as a *biased baseline* (since it scores pairs of nodes by multiplying their degrees) and quantify the similarity in the predictions of the algorithms with that of preferential attachment by measuring the overlaps for k predictions (for varying k, Figure 1(b)). To obtain a normalized score (where +1/0/-1 indicates bias towards high degrees/no bias/anti-bias towards low degrees), we compute the area under these functions for each algorithm in log-log scale (so that top predictions are given emphasis) and normalize the area according to the maximum possible overlap (k) and the expected overlap (for random predictions). The results of this analysis (Figure 1(c)) are mostly as expected: Common neighbors and L3 exhibit the highest bias, followed by Deepwalk-withdegree and Line. Other algorithms exhibit relatively lower bias, while Jaccard index and Deepwalk are slightly biased toward low-degree nodes.

Standard settings for evaluating link prediction algorithms

Here, our aim is to investigate the standard evaluation settings and demonstrate how they can favor bias in predictions towards well-studied entities and how this can lead to conclusions that are counter-productive to the goals of algorithm development in the context network biology and proteomics [26]. For this purpose, following the recommendations of the machine learning community [18, 19], we consider two ways to generate train/test splits: (i) Edge-Uniform: We randomly sample the edges in the network (in 2020 version) uniformly at random and include 10% of the edges in the test set and use the remaining as the training set (ii) Across-Time: We use a more recent, 2022 snapshot of the network as the test set and the older 2020 version as the training set.

In Figure 2, the precision-recall curve is shown for all algorithms for both benchmarking datasets. Here, precision is scaled so that the precision of random prediction is 1 (e.g., an algorithm having a scaled precision of 100 indicates 100× more precise predictions compared to random). We also compute two metrics: The area under precision-recall curve (AUPR) and area under precision-recall curve in log-log scale (AUlogPR). Since link prediction problems involve a large background set (i.e., possible node pairs for n nodes is $\Theta(n^2)$), even 10% recall corresponds to a very high number of predictions (in the order of $\approx 10^5/10^6$ for edge-uniform/across-time data; S. Figure 3). Thus, AUPR in linear scale, whose more than 90% of effective region consists of high number of predictions ($> 10^5$), can be considered a measure of *late curve predictivity*. AUlogPR, on the other hand, puts more emphasis on lower recall values (in logarithmic intervals) corresponding to lower number of predictions, thus providing a measure of *early curve predictivity*. While other metrics like early precision are used in recent literature to evaluate early curve predictivity [34], an advantage of AUlogPR over early precision is that it does not require a fixed threshold that defines "early".

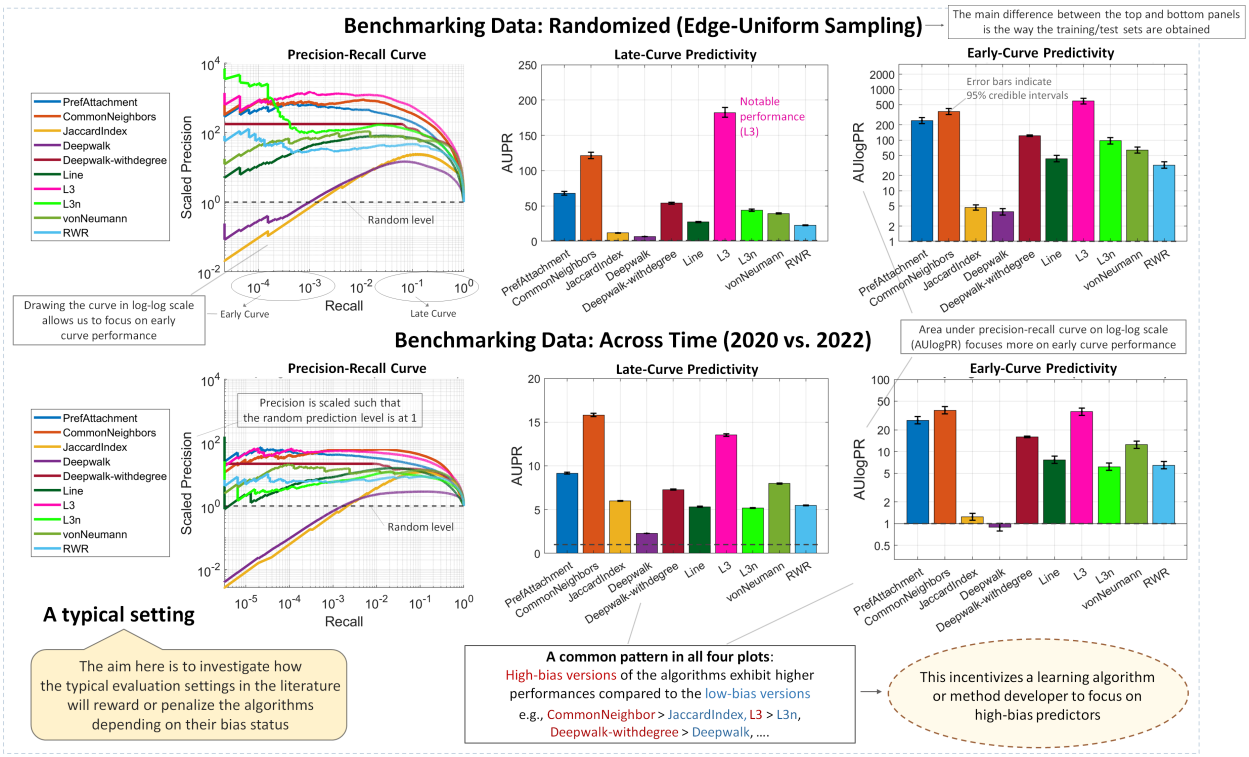


Figure 2: Results of a typical evaluation setting investigating the prediction performance of the link prediction algorithms in the context of PPI predictions. Two types of benchmarking data is considered: (Top panel) Randomized edge-uniform sampling and (Bottom panel) different snapshots of the network across time are used to generate the training and test instances.

As seen in Figure 2, the algorithms that are biased toward high-degree nodes seem to outperform other algorithms according to this evaluation setting, where the high-bias versions of the algorithms (CommonNeighbors, L3, Deepwalk-withdegree) exhibit considerably higher prediction performance compared to their low-bias versions (JaccardIndex, L3n, Deepwalk). The differences based on the degree bias seem more pronounced in the early curve (AUlogPR), i.e., the algorithms are typically penalized more strictly if they rank low-degree node pairs higher in their predictions. Overall, these results show that the standard evaluation settings for evaluating link prediction algorithms can incentivize an algorithm or method developer to focus on high-bias predictors that bring forward well-studied biological entities at the expense of the under-studied ones.

View #2: Bias in benchmarking data and evaluation framework

- **Aim:** To assess bias in an evaluation setting (e.g., training/test sets) in terms of the incentive it provides toward high-bias predictors (that prioritize well-studied entities).
- **Proposed strategy:** Measure the informedness of node degree information in distinguishing the positives from negatives in the test set, using preferential attachment as a representative model for node degree information.

Having observed that the standard evaluation setting favors algorithms that are biased toward high-degree nodes, we next aim to understand the reasons that underlie this observation. For this purpose, we assess the imbalance in a given network or benchmarking data (i.e., train/test splits) in terms of the degree distribution and quantify the predictive power provided by this imbalance for separating the "positives" (known interactions hidden from the algorithms) from the "negatives" (set of possible node pairs without a known interaction). For this purpose, we start by categorizing the nodes based on their connectivity in the PPI network (Figure 3(a)): Poor nodes (<= 20 interactions), Moderate nodes (degree between 20 and 100), and Rich nodes (> 100 interactions). Note that, we assign these categories by considering the cumulative degree distribution so that Poor and Rich nodes roughly comprise 50% and 15% of all nodes in the network.

Once nodes are categorized into three groups, we categorize the interactions in the network into nine (3x3) groups involving all possible combinations of categories of the incident nodes. We report the number of edges in each of these nine categories (Figure 3(b)). This analysis highlights the

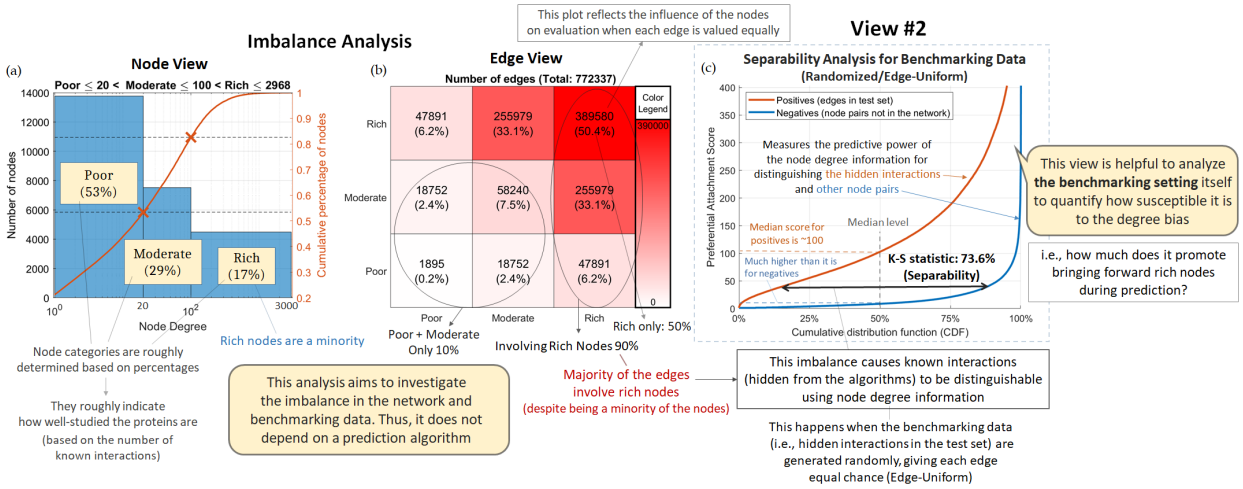


Figure 3: Investigating the imbalance in the benchmarking data and the incentive towards high-bias predictors by quantifying the predictive power of node degree information in distinguishing the known interactions on Biogrid PPI network. (a) Assigned node categories indicating how well-studied a protein is based on node degree information. (b) The distribution of the edges in the network across these categories. (c) Separability analysis for the randomized/edge-uniform setting.

drastic imbalance in the distribution of the edges between different node groups: Although Poor and Moderate nodes together comprise about 85% of all nodes in the network, 50% of all edges are between two Rich nodes and 90% of the edges in the network involve at least one Rich node. A concerning consequence of this imbalance is that, when all edges are valued equally in the evaluation metrics (as typically the case in standard settings) and when the edges in test set are sampled uniformly at random (Edge-Uniform), this guarantees most of the attention in the evaluation to be given to high-degree-nodes (70% expected influence for Rich nodes). In other words, the evaluation setting pays almost no attention to the ability of algorithms to predict interactions that involve low-degree nodes (5% influence for Poor nodes despite being a 53% majority of all nodes). This situation makes it lucrative for the algorithms to prioritize prediction of new interactions for well-studied proteins at the expense of under-studied ones, even though uncovering a new interaction between under-studied proteins may very well be more beneficial for biological knowledge generation [26].

To quantify the degree to which the algorithms are incentivized to prioritize high-degree nodes, we perform an analysis that we refer as separability analysis and assess the predictive power provided by node degree information. For this purpose, we compare the cumulative distribution functions (CDFs) of the preferential attachment scores for the positive and negative sets (known interactions in test set vs. other node pairs) and use the Kolmogorov-Smirnov (K-S) statistic to quantify the separability of the CDFs (which corresponds to the informedness [35] of the preferential attachment model at its best possible prediction point). Figure 3(c) shows that the edges in the positive set generated by random (edge-uniform) sampling are largely distinguishable from negative pairs using the node degrees (K-S statistic: 73.6%). When multiple snapshots (across time) of the network are used for evaluation, the positive edges are still largely separable from negative pairs (K-S statistic: 59.3%, S. Figure 4), though to a lesser degree than it is for edge-uniform sampling. This suggests that using a different snapshot of the network as a test set instead of a randomly sampled test set *does not address* the issue of favoring algorithms that make biased predictions. In contrast, this observation reinforces the notion that research continues to generate knowledge that involves well-studied proteins [26], as the nodes that gain interactions over time are those that have high-degree in the earlier network. Thus, we conclude that an alternative evaluation style is needed to prevent the under-representation of the under-studied proteins on evaluation. For this purpose, in the next view, we focus on a simple idea: Valuing each node equally, as opposed to each edge.

View #3: Weighted evaluation setting focusing on under-studied entities

- **Aim:** To ensure that under-studied proteins are not under-represented while assessing the prediction performance for the link prediction algorithms.
- **Proposed strategy:** Apply weights to explicitly value the importance of discovering different interactions based on the degree of the involved nodes. The weights are optimized to balance the influence of the nodes on evaluation.

As we demonstrate in the previous section, standard evaluation settings provide little information on an algorithm's ability to make successful predictions involving under-studied entities, even though making successful predictions involving under-studied entities are at least as important as making successful predictions involving well-studied entities [26]. To fill this important gap in the evaluation pipeline, we propose a weighted setting that aims to balance the influence of each node on evaluation to be roughly equal (hence *node-uniform evaluation*, as opposed to each edge in standard settings). To obtain such weights, we formulate this as an optimization problem, where the objective is to make the weighted node degrees as close as possible to an input degree distribution (which we set as uniform distribution). We develop an iterative algorithm (Algorithm 1) to solve this optimization problem and assign the optimized weights to edges as instance weights during the computation of evaluation metrics (an alternative option to this, that we do not tackle in detail here, is to

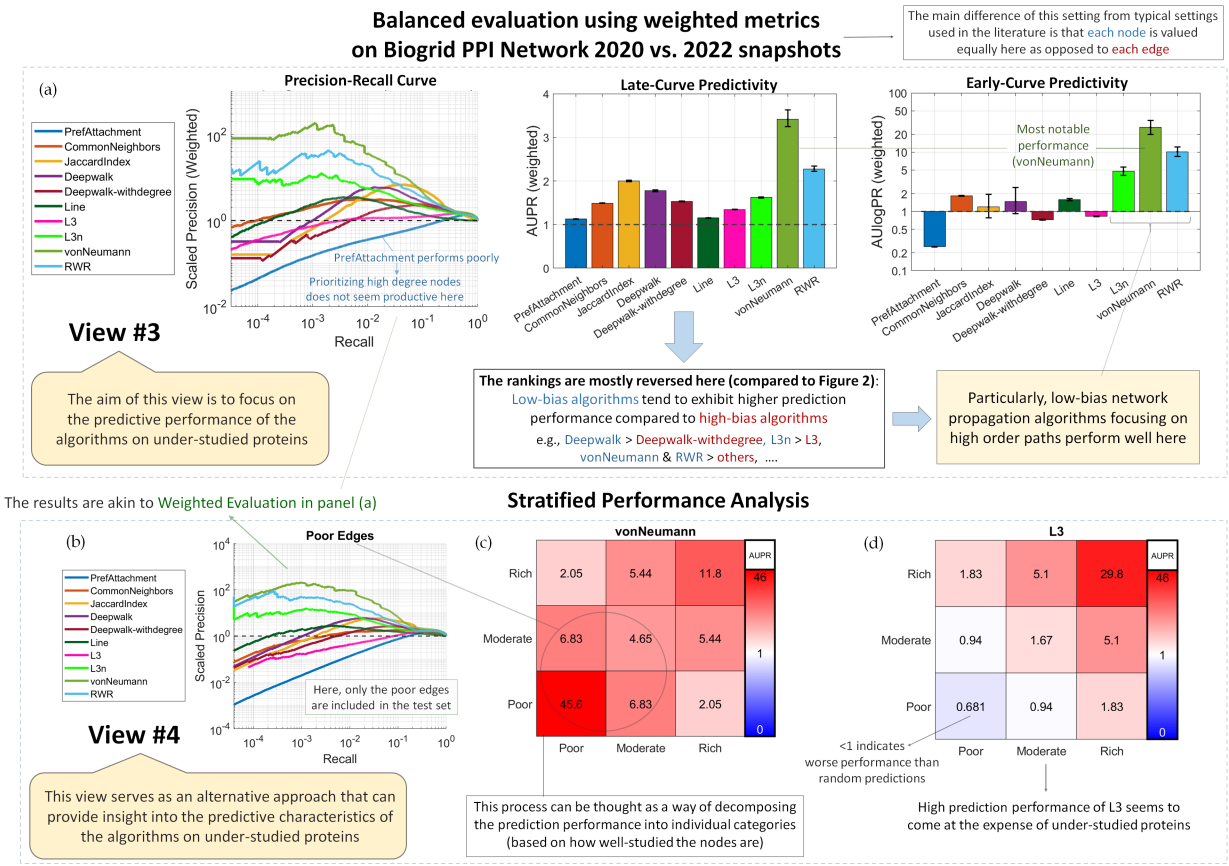


Figure 4: Balanced evaluation setting focusing on the prediction performance of the algorithms on under-studied proteins with the use of weighted metrics (node-uniform) and stratified analysis on Biogrid PPI Across-Time (2020 vs. 2022) data. Prediction performance of the algorithms for (a) weighted analysis and (b) stratified analysis. (c, d) Late curve predictivity (AUPR) stratified by node categories for von Neumann and L3 algorithms.

use the weights as probabilities to generate node-uniform sampled test sets for evaluation). We show that these weighted metrics can roughly balance the influence of the nodes in the evaluation process (S. Figure 5, 40%/35%/25% expected influence for Poor/Moderate/Rich nodes in weighted setting) and mitigate the degree bias in the benchmarking data by reducing the predictive power of node degree information (S. Figure 5(d), K-S statistic for pref. attachment is 33.9%/18.4% for edge-uniform/across-time data in weighted setting).

The evaluation of the link prediction algorithms using the weighted metrics is shown in Figure 4 for across-time data (results for sampled data are given in S. Figure 6). As seen in the figure, the performance comparisons suggested by this setting is quite different from that suggested by the standard settings (Figure 2) and low-bias versions of the algorithms tend to exhibit higher performances compared to high-bias versions here. Note that, while biased algorithms are not favored in this setting, anti-biased algorithms (that bring forward low-degree nodes indiscriminately) are not favored either. For example, anti-preferential attachment model (i.e., ranking the pairs in the opposite order for preferential attachment) performs just as worse as preferential attachment in this setting (S. Figure 7), which suggests that the weighting mitigates the degree bias in the evaluation without causing an anti-bias by inflating the weights of low-degree nodes beyond necessary. Overall, we observe that the best performing algorithms according to this setting are low-biased network propagation algorithms, von Neumann and RWR (whose performance levels are mostly consistent

with the standard, unweighted setting, while the other algorithms' have dropped).

View #4: Stratified analysis to focus on under-studied proteins

- **Aim:** To assess the prediction performance of the algorithms for uncovering new interactions depending on how well-studied the involved proteins are.
- **Proposed strategy:** Stratify the prediction performance into individual edge categories (based on the degrees of incident nodes) by keeping only the interactions from one category in the test set during evaluation.

An alternate and perhaps more direct way to investigate the prediction performance of the algorithms for discovering new interactions on under-studied proteins is to decompose the prediction performance into individual categories based on node degrees, measuring the predictivity in each category by only keeping the edges in that category in the test set. For this purpose, we stratify the edges into 3x3 categories based on node degrees (similar to how it is done in Figure 3b) and further group them into two categories: Poor edges (edges between Poor+Moderate nodes), and Rich edges (remaining edges involved with a rich node). As seen in Figure 4(b) for across-time data, the precision-recall curves for Poor edges is quite similar to the ones obtained by weighted analysis in Figure 4 although quite different from the standard setting in Figure 2 (this is not surprising since poor edges are given 54% influence in the weighted setting as opposed to 8% in unweighted setting, S. Figure 8). Similarly, the curves for Rich edges (S. Figure 9) are akin to the ones in standard setting (as these edges were given 92% influence there). In Figure 4(c) and (d), we show the results of 3x3 stratified analysis for the best performing algorithms, vonNeumann and L3, respectively for Poor and Rich edges. Here, we observe that vonNeumann's predictivity is relatively balanced across different edge categories, whereas the high predictive performance achieved by L3 on Rich-Rich and Rich-Moderate interactions seems to come at the cost of severely diminished predictivity for edges involving under-studied proteins.

View #5: Simple but comprehensive summary for prediction performance

- **Aim:** To make a comprehensive and bias-aware evaluation in a simple manner.
- **Proposed strategy:** Measure five aspects, early/late curve prediction performance for under-studied/well-studied nodes, and the disposition of an algorithm regarding degree bias.

While inspecting the performance curves (as in Figure 2(a), and Figure 4(a)) or the results of stratified analysis (as in Figure 4(c) and (d)) are in general more informative than looking at individual summary metrics, the use of such metrics is still critical for making quick assessments and comparisons. As a reasonable compromise between simplicity and comprehensiveness, we propose the use of five-metrics to summarize the prediction characteristics of a given algorithm in a bias-aware manner (Figure 5), measuring: fairness of the predictions (defined as 1 minus absolute value of the bias metric), early and late curve predictivity (measured by AUPR and AUlogPR respectively) for well-studied entities (results of the standard, unweighted evaluation setting valuing each edge equally) and the same for under-studied entities (results of the weighted setting, valuing each node equally).

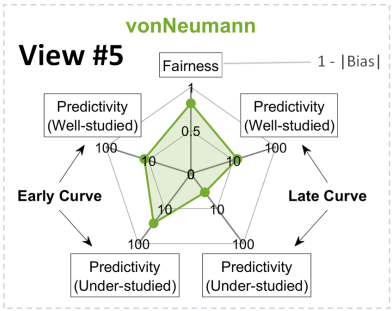


Figure 5: 5-metric summary for von Neumann algorithm.

Bias in benchmarking data for other link prediction problems in biology

The results presented so far demonstrate the bias toward well-studied proteins in benchmarking data and the evaluation setting in the context of PPI predictions using the Biogrid network. To assess the generalizibility of our conclusions and motivate the application of the proposed strategies

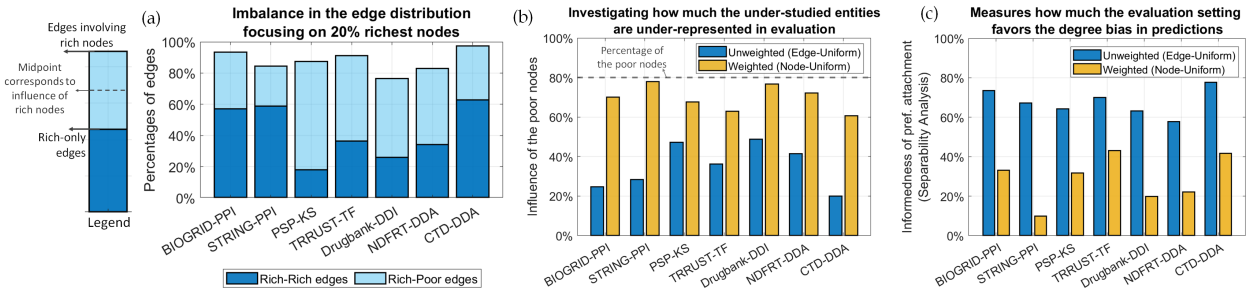


Figure 6: Investigating different datasets and link prediction problems in the context of network biology in terms of the imbalance in the network, under-representation of the under-studied entities in the evaluation, and the incentive towards high-bias predictors. (a) The percentage of edges involved with the 20% richest nodes in the network. (b) Influence of the under-studied (i.e., 80% the lowest degree) nodes in the evaluation for weighted (node-uniform) and unweighted (edge-uniform) settings. (c) Predictive power of the node degree information (measured by the separability analysis). Please visit https://yilmazs.shinyapps.io/colipe to inspect the imbalance in these datasets in an interactive manner.

to a broader range of problems, we here analyze the bias in benchmarking data for additional PPI datasets and other prominent link prediction problems in biomedical applications for the weighted (node-uniform) and unweighted (standard, edge-uniform) settings (Figure 6). Specifically, we investigate the STRING human PPI network [36], PhosphositePlus kinase-substrate interactions (PSP-KS) [37], TRRUST transcription factor regulatory interactions [38], DrugBank drug-drug interactions (Drugbank-DDI) [39], and drug-disease association (NDRFT-DDA [1] and CDA-DDA [40]) networks. For each of these datasets, we examine the imbalance in the edge distribution for the top 20% of the nodes with highest degrees (Figure 6a), quantify how much the under-studied entities are under-represented in the evaluation (by measuring the expected influence of 80% of the nodes with lowest degrees, Figure 6b), and the incentive provided by the evaluation setting towards high-bias predictors (by measuring the predictive power of node degree information based on separability analysis, Figure 6c). Overall, we observe that for a broad range of network datasets that are commonly used for benchmarking link prediction algorithms, there is large degree of imbalance in the edge distributions, and as a result, standard settings that equally value each edge in the evaluation reward algorithms that bring forward high-degree, well-studied entities in their predictions. To overcome this issue, node-uniform weighting can help balance the influence of the nodes and prevent the under-studied proteins from being under-represented in the evaluation.

Conclusions

Overall, to facilitate bias-aware evaluation of link prediction algorithms and to promote the discovery of new interactions involving under-studied entities, we suggest the following approaches to developers and evaluators of link prediction algorithms:

- **Bias in algorithms:** Investigate the disposition of the algorithms towards well-studied nodes based on similarity with preferential attachment model (View 1).
- **Bias in benchmarking data:** Examine the training/test splits (and the benchmarking setting itself) to see how much they incentivize high-bias predictors by quantifying the predictive power of node degree information (View 2).
- **Evaluating the prediction performance:** Adopt a weighted setting, valuing each node as opposed to each edge equally (View 3), or perform a stratified analysis (View 4) to assess the prediction performance on under-studied proteins.
- **Summarize the findings:** Consider five aspects to give an outline for the main characteristics of an algorithm, regarding the early curve/late curve predictivity, well-studied/under-proteins, as well as the bias in predictions (View 5).

Methods

In this section, we briefly describe the methodology we propose, focusing on evaluation metrics and the proposed weighted validation setting. Technical details, formal descriptions, and other information regarding the methods used in this work are provided as Supplementary Materials.

Evaluation metrics. To measure the late-curve prediction performance, we utilize the area under the precision-recall curve (AUPR). For the early-curve performance, we compute the area under the precision-recall curve in log-log scale (AUlogPR) through numerical integration (after normalizing the logarithmic x-axis such that the resulting unit for AUlogPR is precision). Note that, to make the evaluation results comparable with different networks or settings, we scale both metrics to have an expected value of 1 for random predictions. To account for the variance in the estimation of these meaures, we construct 95% credible intervals following a Bayesian approach [41].

Optimization algorithm for node-uniform edge weights. To obtain a set of edge weights (denoted as $\mathbf{W}$ matrix) that establishes node-wise uniformity (i.e., for the row and column sums of $\mathbf{W}$ to be equal for all nodes), we formulate this as an optimization problem and develop an algorithm that iteratively performs multiplicative updates (ensuring the uniformity of the rows in one step, and for the columns in another) until the uniformity of the row and column sums are established simultaneously at an acceptable level. The formulation of the optimization problem and a simplified pseudo-code of the developed algorithm (omitting some details) is given in Algorithm 1. A more detailed description of the algorithm (including a complete pseudo-code) specifying the technical details (e.g., regarding matrix initialization, termination conditions, controlling the step size during updates and so on) are provided in the Supplementary Methods. Note that, we denote $\mathbf{Q_r}$ and $\mathbf{Q_c}$ to indicate the desired weights for rows and columns instead of assuming uniformity for the sake of generalizability (i.e., node-uniform when $\mathbf{Q}_r = \mathbf{Q}_c = \mathbf{1}$).

Algorithm 1 Optimization Problem and Algorithm (Simplified Pseudo-Code)

Input: Graph $\mathbf{G} = \{\mathcal{V}, \mathcal{E}\}$, desired node weights $\mathbf{Q_r}$ and $\mathbf{Q_c}$ for rows and columns, step size $\alpha \leq 1$
Output: Edge weighting matrix $\mathbf{W}$ such that $\mathbf{W}_{ij} = 0$ if $(i,j) \notin \mathcal{E}$ and $\mathbf{W}_{ij} \geq 0\ \forall (i,j) \in \mathcal{E}$
  Initialize $\mathbf{W} \leftarrow \mathbf{W}_{init}$ and Normalize $\mathbf{W}$ to sum up to 1
  while maximum iteration limit is not reached **do**
    $\mathbf{D_r} \leftarrow$ row sums of $\mathbf{W}$
    $\mathbf{D_c} \leftarrow$ column sums of $\mathbf{W}$
    $\mathbf{W} \leftarrow \mathbf{W} \oslash (\mathbf{D_r} \oslash \mathbf{Q_r})^{\alpha}$ ▷ $\oslash$ indicates Hadamard (element-wise) division
    $\mathbf{W} \leftarrow \mathbf{W} \oslash (\mathbf{D_c} \oslash \mathbf{Q_c})^{\alpha}$
    Normalize $\mathbf{W}$ to sum up to 1
  end while
Optimization Problem: Compute $\mathbf{W}$ to minimize $||\mathbf{Q_r} - \mathbf{D_r}||_2 + ||\mathbf{Q_c} - \mathbf{D_c}||_2$

Weighted evaluation. In the weighted evaluation setting, we use the optimized edge weights (the matrix $\mathbf{W}$ computed by the algorithm above) as the weight for each positive instance (existing edge in the test set). For this purpose, we generalize the computation of AUPR and AUlogPR to assign weights to instances (positives in the test set) while counting the number of true positives (TPs) and false negatives (FNs). For example, an edge in the test set that is weighted worth of 3 unweighted edges, if included in the predictions of an algorithm, would increase the number of TPs by 3 as opposed to 1. Performance measures are then computed based on these weighted counts.

Influence of a node category on evaluation. We quantify the influence of a node category (rich, moderate, or poor) on evaluation as the total weight of the edges (percentage of edges for standard evaluation) that are incident to the nodes in that category, counting between-category edges as half such that total influence for all categories adds up to 1.

References

[1] Yue, X. *et al.* Graph embedding on biomedical networks: methods, applications and evaluations. *Bioinformatics* **36**, 1241–1251 (2020).

[2] Liang, X. *et al.* Lrssl: predict and interpret drug–disease associations based on data integration using sparse subspace learning. *Bioinformatics* **33**, 1187–1196 (2017).

[3] Stanfield, Z., Coşkun, M. & Koyutürk, M. Drug response prediction as a link prediction problem. *Scientific reports* **7**, 1–13 (2017).

[4] Erten, S., Bebek, G., Ewing, R. M. & Koyutürk, M. Da da: degree-aware algorithms for network-based disease gene prioritization. *BioData mining* **4**, 1–20 (2011).

[5] Zhang, W., Chen, Y., Li, D. & Yue, X. Manifold regularized matrix factorization for drug-drug interaction prediction. *Journal of biomedical informatics* **88**, 90–97 (2018).

[6] Kovács, I. A. *et al.* Network-based prediction of protein interactions. *Nature communications* **10**, 1–8 (2019).

[7] Li, H. *et al.* Inferring transcription factor regulatory networks from single-cell atac-seq data based on graph neural networks. *Nature Machine Intelligence* **4**, 389–400 (2022).

[8] Ayati, M. *et al.* Cophosk: A method for comprehensive kinase substrate annotation using co-phosphorylation analysis. *PLoS computational biology* **15**, e1006678 (2019).

[9] Invergo, B. M. *et al.* Prediction of signed protein kinase regulatory circuits. *Cell systems* **10**, 384–396 (2020).

[10] Zhou, T. Progresses and challenges in link prediction. *Iscience* **24**, 103217 (2021).

[11] Liben-Nowell, D. & Kleinberg, J. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, 556–559 (2003).

[12] Lei, C. & Ruan, J. A novel link prediction algorithm for reconstructing protein–protein interaction networks by topological similarity. *Bioinformatics* **29**, 355–364 (2013).

[13] Perozzi, B., Al-Rfou, R. & Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of KDD*, 701–710 (2014).

[14] Grover, A. & Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of KDD*, 855–864 (2016).

[15] Su, C., Tong, J., Zhu, Y., Cui, P. & Wang, F. Network embedding in biomedical data science. *Briefings in bioinformatics* **21**, 182–197 (2020).

[16] Deznabi, I., Arabaci, B., Koyutürk, M. & Tastan, O. Deepkinzero: zero-shot learning for predicting kinase–phosphosite associations involving understudied kinases. *Bioinformatics* **36**, 3652–3661 (2020).

[17] Yu, B. *et al.* Prediction of protein–protein interactions based on elastic net and deep forest. *Expert Systems with Applications* **176**, 114876 (2021).

[18] Yang, Y., Lichtenwalter, R. N. & Chawla, N. V. Evaluating link prediction methods. *Knowledge and Information Systems* **45**, 751–782 (2015).

[19] Mara, A. C., Lijffijt, J. & De Bie, T. Benchmarking network embedding models for link prediction: Are we making progress? In *2020 IEEE 7th International conference on data science and advanced analytics (DSAA)*, 138–147 (IEEE, 2020).

[20] Lai, Y.-A., Hsu, C.-C., Chen, W. H., Yeh, M.-Y. & Lin, S.-D. Prune: Preserving proximity and global ranking for network embedding. *Advances in neural information processing systems* **30** (2017).

[21] Gao, M., Chen, L., He, X. & Zhou, A. Bine: Bipartite network embedding. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, 715–724 (2018).

[22] Nelson, W. *et al.* To embed or not: network embedding as a paradigm in computational biology. *Frontiers in genetics* **10**, 381 (2019).

[23] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K. & Galstyan, A. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)* **54**, 1–35 (2021).

[24] Mara, A., Lijffijt, J. & De Bie, T. Evalne: A framework for evaluating network embeddings on link prediction. *arXiv preprint arXiv:1901.09691* (2019).

[25] Eid, F.-E. *et al.* Systematic auditing is essential to debiasing machine learning in biology. *Communications biology* **4**, 1–9 (2021).

[26] Kustatscher, G. *et al.* Understudied proteins: opportunities and challenges for functional proteomics. *Nature Methods* 1–6 (2022).

[27] Yılmaz, S. *et al.* Robust inference of kinase activity using functional networks. *Nature communications* **12**, 1–12 (2021).

[28] Jones, D. T. Setting the standards for machine learning in biology. *Nature Reviews Molecular Cell Biology* **20**, 659–660 (2019).

[29] Oughtred, R. *et al.* The biogrid database: A comprehensive biomedical resource of curated protein, genetic, and chemical interactions. *Protein Science* **30**, 187–200 (2021).

[30] Tang, J. *et al.* Line: Large-scale information network embedding. In *Proceedings of WWW*, 1067–1077 (2015).

[31] Kunegis, J., Luca, E. W. D. & Albayrak, S. The link prediction problem in bipartite networks. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems*, 380–389 (Springer, 2010).

[32] Tong, H., Faloutsos, C. & Pan, J.-Y. Random walk with restart: fast solutions and applications. *Knowledge and Information Systems* **14**, 327–346 (2008).

[33] Gillis, J., Ballouz, S. & Pavlidis, P. Bias tradeoffs in the creation and analysis of protein–protein interaction networks. *Journal of proteomics* **100**, 44–54 (2014).

[34] Pratapa, A., Jalihal, A. P., Law, J. N., Bharadwaj, A. & Murali, T. Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nature methods* **17**, 147–154 (2020).

[35] Youden, W. J. Index for rating diagnostic tests. *Cancer* **3**, 32–35 (1950).

[36] Szklarczyk, D. *et al.* String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic acids research* **47**, D607–D613 (2019).

[37] Hornbeck, P. V. *et al.* Phosphositeplus, 2014: mutations, ptms and recalibrations. *Nucleic acids research* **43**, D512–D520 (2015).

[38] Han, H. *et al.* Trrust v2: an expanded reference database of human and mouse transcriptional regulatory interactions. *Nucleic acids research* **46**, D380–D386 (2018).

[39] Knox, C. *et al.* Drugbank 3.0: a comprehensive resource for 'omics' research on drugs. *Nucleic acids research* **39**, D1035–D1041 (2010).

[40] Davis, A. P. *et al.* The comparative toxicogenomics database: update 2019. *Nucleic acids research* **47**, D948–D954 (2019).

[41] Jaynes, E. T. & Kempthorne, O. Confidence intervals vs bayesian intervals. In *Foundations of probability theory, statistical inference, and statistical theories of science*, 175–257 (Springer, 1976).

[42] Mikolov, T., Chen, K., Corrado, G. & Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[43] Openne: An open source toolkit for network embedding. `https://github.com/thunlp/OpenNE` (2019).

Are under-studied proteins under-represented?
How to fairly evaluate link prediction algorithms in network biology

Supplementary Materials

Table of Contents for Supplementary Materials

Supplementary Methods **2**
- Link Prediction Algorithms - Verbal Descriptions . . . 2
- Link Prediction Algorithms - Mathematical Formulations . . . 2
  - Methods based on simple scoring metrics . . . 2
  - Higher-order paths and Network propagation based methods . . . 3
  - Embedding-based methods . . . 3
- Evaluation Metrics for Prediction Performance . . . 4
  - Computing the area under precision-recall curve (AUPR) . . . 4
  - Interpolating the curve during numerical integration for computing the area under . 5
  - Early-curve performance, the area under log-scale precision-recall curve (AUlogPR) . 6
  - Computing credible intervals for the variance in estimation . . . 6
- Weighted Validation Setting Focusing on Under-Studied Entities . . . 7
  - Optimization algorithm for obtaining edge weights based on node valuations . . . 7
  - Weighted evaluation metrics . . . 9
  - Computing credible intervals for the weighted metrics . . . 9
- Quantifying the bias towards high-degree nodes in method predictions . . . 10
- Separability analysis quantifying the imbalance in benchmarking data . . . 10
- Computing the influence of the nodes on evaluation . . . 10
- Performing stratified analysis . . . 10
- Datasets used in this work . . . 11

List of Figures

1 S. Figure 1 - Embedding dimensions of Deepwalk and their association with node degrees. . . . 12
2 S. Figure 2 - Embedding dimensions of LINE and their association with node degrees. 13
3 S. Figure 3 - Number of predictions required to reach a particular recall threshold. . 14
4 S. Figure 4 - Separability analysis for Biogrid PPI across time. . . . 14
5 S. Figure 5 - Mitigating degree bias in the evaluation of link prediction algorithms by assigning weights to edges . . . 15
6 S. Figure 6 - Balanced/weighted evaluation results on randomized (sampled) benchmarking data . . . 15
7 S. Figure 7 - Comparison of preferential attachment and anti-preferential attachment on different evaluation settings. . . . 16
8 S. Figure 8 - Influence by node and edge categories . . . 17
9 S. Figure 9 - Stratified performance analysis for Rich edges connnected to well-studied nodes . . . 17

Supplementary Methods

Link Prediction Algorithms - Verbal Descriptions

Methods based on scoring metrics: We consider the preferential attachment model to represent a purely biased model (where node pairs are ranked based on the product of the degrees of the endpoints). Common neighbors represents a high-bias algorithm that considers paths of length 2 (high bias since the number of paths in correlated with the node degrees). Jaccard Index represents a low bias version of common neighbors where a normalization is applied based on node degrees.

High order paths/Network propagation algorithms: L3 is a method that counts the paths of length 3 to make predictions. For this purpose, in this work, we consider the formulation given in [6] that applies a soft normalization (based on square root of degrees, this is what we consider the high-bias version). We also introduce a low bias version of it, L3-Normalized (L3n) that applies a stronger normalization based on node degrees. Whereas, von Neumann [31] and random walks with restarts (RWR) [32] are network propagation algorithms that consider a weighted combination of paths of different lengths. Formulation of both algorithms involve a strong normalization based on node degrees (the main difference between them is the style of the normalization, whether it is done symmetrically or based on column normalization). Thus, we consider both to be low-bias algorithms.

Embedding/Learning Methods: We consider two types of embedding methods: Deepwalk [13] (Random walk based) and Line [30] (Neural network based). For each of these methods, we train a logistic regression model using the embeddings as features. Here, deepwalk represents a low-bias algorithm (since the embedding dimensions are uncorrelated with node degrees, likely by design, S. Figure 1) and Line is a higher bias algorithm (since its embeddings pick up the node degree info during learning, S. Figure 2). For deepwalk-withdegree, we include the node degrees as an additional dimension (as if it is part of the embeddings matrix) to construct a high-bias version of the deepwalk algorithm.

Link Prediction Algorithms - Mathematical Formulations

Methods based on simple scoring metrics

Preferential Attachment:

$$\sigma_{PA}(u,v) = \sqrt{|\Gamma(u)||\Gamma(v)|} \tag{1}$$

where $\Gamma(u)$ denotes the set containing the neighbors of u. In matrix form, the preferential attachment score is equal to:

$$\sigma_{PA} = \sqrt{D_r \odot D_c} \tag{2}$$

where $\odot$ indicates the element-wise (Hadamard) product and D_r and D_c are respectively row and column degrees in matrix form:

$$\begin{aligned} D_r(u,v) &= |\Gamma(u)| \\ D_c(u,v) &= |\Gamma(v)| \end{aligned} \tag{3}$$

Common Neighbors:

$$\sigma_{AA}(u,v) = |\Gamma(u) \cap \Gamma(v)| \tag{4}$$

In matrix form, this is simply equal to:

$$\sigma_{AA} = A^2 \tag{5}$$

where A is the adjacency matrix of the network.
Jaccard Index:

$$\sigma_{JI}(u,v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|} \tag{6}$$

In matrix form,

$$\begin{aligned} \sigma_{JI} &= A^2 \oslash N \\ N &= D_r + D_c - A^2 \end{aligned} \tag{7}$$

where $\oslash$ indicates the element-wise (Hadamard) divide operation.

Higher-order paths and Network propagation based methods

L3: In matrix form,

$$\begin{aligned} \sigma_{L3} &= A' \times A' \times A \\ A' &= A \oslash \sqrt{D_r} \end{aligned} \tag{8}$$

L3-Normalized (L3n):

$$\begin{aligned} \sigma_{L3n} &= A_n^3 \\ A_n &= A \oslash D_r \end{aligned} \tag{9}$$

von Neumann:

$$\begin{aligned} A_s &= A \oslash \sqrt{D_r \odot D_c} \\ \sigma_{VN} &= \alpha A_s + \alpha^2 A_s^2 + \alpha^3 A_s^3 \ldots \\ &= \sum_{i=1}^{l} \alpha^i A_s^i \end{aligned} \tag{10}$$

Here, we use $\alpha = 0.5$ and go up to path lengths of $l = 4$ for computational efficiency reasons.
Random walks with restarts (RWR):

$$\begin{aligned} A_n &= A \oslash D_r \\ \sigma_{RWR} &= \alpha A_n + \alpha^2 A_n^2 + \alpha^3 A_n^3 \ldots \\ &= \sum_{i=1}^{l} \alpha^i A_n^i \end{aligned} \tag{11}$$

Similar to von-Neumann method, we use $\alpha = 0.5$ and go up to $l = 4$.

Embedding-based methods

In addition to these methods that compute a single score for each candidate pair, recent link prediction algorithms commonly use node embeddings to facilitate supervised learning. Node embeddings map the nodes in a network to a lower-dimensional embedding space, such that adjacent nodes are mapped to points that are close to each other in this embedding space [14]. Subsequently, using these embeddings as feature vectors and existing edges as training data, machine learning models are trained to predict new edges. We consider two embedding methods that are representative of common approaches to the computation of node embeddings.

Deepwalk (Random walk based node embedding): Deepwalk[13] uses random walks to generate a list of paths in the network as its corpus and then uses Word2Vec[42], a natural language processing algorithm for word embedding, to compute node embeddings by treating the list of paths as text and nodes as words. In our experiments, we use the implementation used in BioNEV[1] repository (which is based on OpenNE [43]) for all embedding methods.

LINE (Neural network based embedding): LINE [30] is one of the earliest algorithms to incorporate neural networks into the computation of node embeddings. It uses a single layer MLP to estimate first and second order proximity of nodes and produces the embedding vectors using a variational auto-encoder.

Unless otherwise specified, we use the default value of 128 in the OpenNE implementation as the embedding dimension (i.e., number of embeddings) for both embedding methods.

In addition to the above two, we consider a version of deepwalk (***Deepwalk-withdegree***) where the node degree information is appended as an additional dimension to the embedding matrix.

Logistic regression as prediction model: For each of three embedding approaches described above, we train a logistic regression using the embeddings as features. For each embedding dimension $x^{(i)}$, we add the following three terms to the logistic regression model corresponding to the prediction for edge (u,v):

$$\text{logit}(Y_{uv}) \propto \sum_i \left(\beta_r^{(i)} x_u^{(i)} + \beta_c^{(i)} x_v^{(i)} + \beta_{rc}^{(i)} x_u^{(i)} x_v^{(i)} \right) \tag{12}$$

While training the model, to ensure a balanced training set, we randomly sample the edges with negative labels (i.e., not in the training set) to have the same size as the edges with positive labels.

Evaluation Metrics for Prediction Performance

Precision and Scaled Precision:

$$\text{Precision} = r = \frac{TP}{TP + FP} \tag{13}$$

where TP denotes the number of true positives, and FP the number of false positives. The expected precision for random predictions is equal to the prevalence of positive labels:

$$E[\text{Precision}] = \text{Prevalence} = \frac{N_P}{N_{\text{total}}} \tag{14}$$

where N_P is the number of positive labels and N_{total} is the total number of edges that are to be predicted (approximately $O(N^2)$).

$$\text{Scaled Precision} = \frac{\text{Precision}}{E[\text{Precision}]} = \frac{TP}{TP + FP} \frac{N_{\text{total}}}{N_P} \tag{15}$$

Recall:

$$\text{Recall} = \frac{TP}{N_P} \tag{16}$$

Computing the area under precision-recall curve (AUPR)

To compute the area under the precision-recall (PR) curve, we use numerical integration. Suppose we have m measurement points. Let TP_i denote the number of true positives, FP_i the number of false positives, $N_i = TP_i + FP_i$ the number of predictions, X_i the recall, and Y_i the precision

corresponding to the ith measurement point. In general, m is less than N_{total} since edges having the same prediction score (e.g., because the link prediction method uses discrete scoring like common neighbors) correspond to a single measurement point. Also, without loss of generality, consider that the first point is the $TP = 0$ and $FP = 0$ point and all points are sorted by the number of predictions ($TP_i + FP_i$) in ascending order. With these in mind, we compute the area under precision-recall curve through numerical integration as follows:

$$\text{AUPR} = \frac{\sum_{i=1}^{m-1} \Delta X_i f Y_i}{\sum_{i=1}^{m-1} \Delta X_i} \tag{17}$$

where ΔX_i is the gap between two consecutive points:

$$\Delta X_i = |X_{i+1} - X_i| \tag{18}$$

Whereas, fY_i is an interpolating function that returns the normalized area under two consecutive points Y_i and Y_{i+1} (thus, it is a type of averaging for two given points and is always between $[Y_i,\ Y_{i+1}]$). For example, a simple function for this purpose can be $\frac{Y_i + Y_{i+1}}{2}$ (interpolating the precision values linearly). However, this type of interpolation suffers from inaccuracy when there are large gaps between two consecutive points X_i and X_{i+1}, which is particularly relevant for link prediction methods with discrete scoring. To demonstrate the inaccuracy, suppose the first point is at 1/1 ($TP = 1$, $FP = 0$) with precision 1 and the next point is in 100/10000 ($TP = 100$, $FP = 9900$) with precision 0.01. In this example, although linear interpolation suggests that the average precision would be ≈0.5, observing one TP in the beginning hardly gives any evidence that the precision will be ≈0.5 at the $TP = 5000$ point. To overcome this type of inaccuracy, we use an interpolation function tailored for the precision-recall curve detailed below:

Interpolating the curve during numerical integration for computing the area under

For the intermediate points between two consecutive points X_i and X_{i+1}, we assume that both the true positives and false positives are scaled linearly:

$$\begin{aligned} TP_x &= TP_i + x(TP_{i+1} - TP_i) \\ FP_x &= FP_i + x(FP_{i+1} - FP_i) \end{aligned} \tag{19}$$

where x is a normalized variable between [0, 1] indicating which endpoint the point is closest to (e.g., 1 indicates the point is right on the i+1th point). Thus, the precision for the intermediate points is given by the ratio $r_i(x)$:

$$\begin{aligned} r_i(x) &= \frac{TP_i + x(TP_{i+1} - TP_i)}{TP_i + x(TP_{i+1} - TP_i) + FP_i + x(FP_{i+1} - FP_i)} \\ &= \frac{TP_i + x(TP_{i+1} - TP_i)}{N_i + x(N_{i+1} - N_i)} \end{aligned} \tag{20}$$

To compute the area under this curve (denoted fY), we need the integral:

$$fY_i = \int_0^1 r_i(x)dx = \int_0^1 \frac{TP_i + x(TP_{i+1} - TP_i)}{N_i + x(N_{i+1} - N_i)} dx \tag{21}$$

Solving this integral gives:

$$fY_i = \frac{(TP_i N_{i+1} - TP_{i+1} N_i) \log\left(\frac{N_{i+1}}{N_i}\right) + (N_{i+1} - N_i)(TP_{i+1} - TP_i)}{(N_{i+1} - N_i)^2} \tag{22}$$

Thus, we use the above function for interpolating while computing the area under the PR curve. To give some insight into what this function results in: For the example before (one point at $TP/N = 1/1$ while the other is at 100/10000), this integral results in ≈ 0.011 precision which is much closer to the latter point (as it should be).

Note that, although this integral (and Equation 22) is not defined at $N_i = 0$ point (since precision is not defined at 0 predictions), the limit from above converges to $\frac{TP_{i+1}}{N_{i+1}} = Y_{i+1}$. Thus, as the first interpolated area, we use:

$$fY_1 = Y_2 \tag{23}$$

where Y_2 (i.e., the second point) corresponds to the first measured precision value (since the 0/0 point is specified in the $i = 1$th point in this notation).

Overall, this interpolation is helpful for reducing the inaccuracy when there are large gaps in between, which is particularly relevant for methods with discrete scoring or for computing the area under the PR curve in logarithmic scale.

Early-curve performance, the area under log-scale precision-recall curve (AUlogPR)

For computing the area under log-log scale PR curve, the process is similar. We use the interpolating function fY given in Equation 22 and perform numerical integration as follows:

$$\begin{aligned} \text{AUlogPR} &= \exp\left(\frac{\sum_{i=1}^{m-1} \Delta X'_i \log fY_i}{\sum_{i=1}^{m-1} \Delta X'_i}\right) \\ \Delta X'_i &= \log X_{i+1} - \log X_i \end{aligned} \tag{24}$$

Note that, while computing AUlogPR, we start the curve at $TP = 10$ point to reduce the variance in the estimation (since the initial points between TP=[1, 10] are considerably volatile).

Note that, for both AUPR and AUlogPR metrics, after computing the area under the curve, we scale them to have an expected value of 1 (for random predictions) by dividing with the prevalence of the positive labels (Equation 14), similar to how it is done in Equation 15.

Computing credible intervals for the variance in estimation

We follow a Bayesian approach to estimate the expected variance in the evaluation metrics (e.g., precision and AUPR). Our view here is akin to the "checking whether a coin is fair" problem. We assume that there is an unknown, but fixed probability r (corresponding to precision). Based on this probability, we suppose that we have made k trials (corresponding to predictions) and observed TP number of hits and FP number of misses. Now, we ask the question "Based on these observations, what can we say about the posterior probability of the ratio r?"

If we assume uniform prior (i.e., all r values in [0, 1] are equally likely), the answer to the above question is specified by the beta distribution:

$$r \sim \text{Beta}(TP + 1, FP + 1) \tag{25}$$

Thus, we obtain the distribution for the posterior probability of the ratio r (i.e., precision) after k predictions. Based on this distribution, we can easily construct a credible interval containing the 95% of the variance using the inverse cumulative distribution function Beta^{-1}. Note that, in general, there is not a single credible interval unique to a given posterior distribution. Thus, among the alternatives, we choose the equal-tailed interval where the probability of being below the interval is as likely as being above it.

This process gives us a 95% interval for the precision at fixed number of predictions k point. To obtain 95% intervals for the area under metrics (AUPR and AUlogPR), we simply construct the intervals for all k points and compute the area under the precision-recall curves formed by the maximum/minimum bounds.

Weighted Validation Setting Focusing on Under-Studied Entities

Optimization algorithm for obtaining edge weights based on node valuations

We formulate this problem as follows: Suppose we are given as set of node valuations q. Let $q_r(u)$ and $q_c(u)$ denote the desired expected number of edges coming into and going out of $u \in \mathcal{V}$ (i.e., the desired row and column sums). Let $\mathbf{W}$ represent the weights of the edges as a sparse matrix where $\mathbf{W}_{ij} = 0$ if $(i, j) \notin \mathcal{E}$. Here, our aim is to estimate a set of edge weights/values $\mathbf{W}$ such that the row and column sums of $\mathbf{W}$ are respectively equal to v_r and v_c. For this purpose, we will use an expectation-maximization based optimization algorithm with multiplicative steps.

Algorithm 2 Optimization Algorithm for Edge Weights

Input: Node valuations $\mathbf{Q_r}$ and $\mathbf{Q_c}$, max. number of iterations, convergence threshold $\epsilon_{convergence}$, maximum step size α_{max}, step size increment $\gamma_{increment}$, step size decrement $\gamma_{decrement}$
Output: Edge weighting matrix $\mathbf{W}$

```
Initialize W ← W_init
Normalize W to sum up to 1
Set initial step size α ← α_max
W_best ← W
bestError ← ∞
while maximum iteration limit is not reached do
    D_r ← row sums of W
    D_c ← column sums of W
    error ← sum of squared error for W
    Measure Δ_change and Δ_improvement
    if Δ_improvement > 0 then                                   ▷
        W_best ← W
        bestError ← error
        Increase step size α ← min(α γ_increase, α_max)
    else
        Restore W ← W_best
        Decrease step size α ← α γ_decrease
    end if
    if (Δ_change + Δ_improvement) ≤ ε_convergence then
        Stop the optimization and return W_best
    end if
    W ← W ⊘ (D_r ⊘ Q_r)^α          ▷ ⊘ indicates Hadamard (element-wise) division
    W ← W ⊘ (D_c ⊘ Q_c)^α
    Normalize W to sum up to 1
end while
```

The pseudo-code of the algorithm is given in Algorithm 2. Below, we describe each step of the algorithm:

Initialization: In the beginning of the algorithm, we set $\mathbf{W}$ to be equal to an initial, approx-

imate solution:

$$\mathbf{W}_{init}(i,j) = \left\{ \begin{array}{ll} \sqrt{\dfrac{q_r(i)q_c(j)}{d_r(i)d_c(j)}} & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{array} \right\} \tag{26}$$

where $d_r(i)$ and $d_c(j)$ are the row and column degrees of nodes i and j in the network respectively. After setting $\mathbf{W} = \mathbf{W}_{init}$, we normalize the weights $\mathbf{W}$ to sum up to 1 as follows:

$$\mathbf{W} \leftarrow \frac{\mathbf{W}}{\sum_{i,j} \mathbf{W}(i,j)} \tag{27}$$

Update steps of the algorithm: Here, to ensure that the updated weights remain positive, we use multiplicative update steps based on row/column normalizations:

$$\begin{aligned} \mathbf{W} &\leftarrow \mathbf{W} \oslash (\mathbf{D_r} \oslash \mathbf{Q_r})^{\alpha} \\ \mathbf{W} &\leftarrow \mathbf{W} \oslash (\mathbf{D_c} \oslash \mathbf{Q_c})^{\alpha} \end{aligned} \tag{28}$$

where α is a multiplicative step size parameter and $\mathbf{D_r}$ and $\mathbf{D_c}$ are row/column sum matrices of $\mathbf{W}$ respectively:

$$\mathbf{D_r}(i,j) = \sum_j \mathbf{W}(i,j) \quad \text{and} \quad \mathbf{D_c}(i,j) = \sum_i \mathbf{W}(i,j) \tag{29}$$

Similarly, input node valuation vectors v_r and v_c are organized as matrices $\mathbf{V_r}$ and $\mathbf{V_c}$ after being normalized:

$$\mathbf{Q_r}(i,j) = \frac{q_r(i)n_r}{\sum_{i'} q_r(i')} \quad \text{and} \quad \mathbf{Q_c}(i,j) = \frac{q_c(j)n_c}{\sum_{j'} q_c(j')} \tag{30}$$

where n_r and n_c are scalars indicating the number of rows and columns in the network. In each step, after updating $\mathbf{W}$ according to Equation 28, $\mathbf{W}$ is normalized again to sum up to 1 as in Equation 27.

Termination of the algorithm: To determine the convergence of the algorithm, we look at two criteria. The first one focuses on the amount of change in $\mathbf{W}$:

$$\Delta_{change} = \frac{||\mathbf{W} - \mathbf{W}_{best}||_2}{||\mathbf{W}_{best}||_2} \tag{31}$$

The second one focuses on the amount of improvement. For this purpose, we first quantify the error using sum of squares:

$$error(\mathbf{W}') = \sum_i \left(Q_r(i) - \sum_j \mathbf{W}'(i,j) \right)^2 + \sum_j \left(Q_c(j) - \sum_i \mathbf{W}'(i,j) \right)^2 \tag{32}$$

Thus, at each step, we measure the improvement $\mathbf{W}$ provides over $\mathbf{W}_{best}$ as follows:

$$\Delta_{improvement} = \max\left(1 - \frac{error(\mathbf{W})}{error(\mathbf{W}_{best})}, 0\right) \tag{33}$$

Overall, we terminate the algorithm when the amount of change plus the improvement is less than a predefined threshold $\epsilon_{convergence}$:

$$\text{Terminate if } \Delta_{change} + \Delta_{improvement} \leq \epsilon_{convergence} \tag{34}$$

In addition to the convergence threshold, we terminate the optimization if the maximum iteration limit is reached. Unless otherwise specified, we use $\epsilon = 10^{-2}$ and 100 maximum iterations for the termination of the algorithm.

Updating the step sizes: When there is no improvement at any point (i.e., $\Delta_{improvement} \leq 0$), we conclude that step size is too large and need to be reduced. For this purpose, we restore $\mathbf{W}$ to $\mathbf{W}_{best}$ (i.e., the best weights with lowest error up to this point) and decrease step size α by a factor of $\gamma_{decrease}$:

$$\alpha \leftarrow \alpha\gamma_{decrease} \tag{35}$$

Conversely, when $\Delta_{improvement} > 0$, we restore α by increasing it with a factor $\gamma_{increase}$ and truncating it to α_{max}:

$$\alpha \leftarrow \min\left(\alpha\gamma_{increase}, \alpha_{max}\right) \tag{36}$$

Unless otherwise specified, we set $\alpha_{max} = 0.999$, $\gamma_{decrease} = 0.6$, and $\gamma_{increase} = 1.25$.

Note about sparse matrices and efficiency: Here, we have described the update steps (Equation 28) in terms of $\mathbf{D_r}/\mathbf{D_c}$ and $\mathbf{Q_r}/\mathbf{Q_c}$ in matrix format for the sake of brevity and clarity. While implementing the algorithm, the element-wise divide ($\oslash$) operation can be efficiently applied on vectors and sparse matrices without ever storing the full matrices.

Weighted evaluation metrics

After obtaining the weighting matrix $\mathbf{W}$ using the optimization algorithm, let weighting vector $w \in \mathbb{R}^{N_{total}\times 1}$ be organized in such a way that w_i indicates the edge weight corresponding to the ith prediction (after all edges are sorted based on the prediction scores of a method). Using this vector, we can compute the weighted true positives for k predictions as follows:

$$TP_w = \frac{\sum_{i=1}^{k} w_i I_i}{w_{\text{norm}}} \tag{37}$$

where I_i is an indicator variable that is equal to 1 if the ith prediction is a true positive and is equal to 0 otherwise and w_{norm} is a normalization factor:

$$w_{\text{norm}} = \frac{\sum_{i=1}^{N_{\text{total}}} w_i I_i}{N_P} \tag{38}$$

where N_P is the number of positive labels in the test set.

After obtaining the weighted true positives, the weighted versions of the AUPR and AUlogPR metrics are computed as described in the previous sections (this time using TP_w instead of TP).

$$\text{Precision (Weighted)} = r_w = \frac{TP_w}{TP_w + FP} \tag{39}$$

Computing credible intervals for the weighted metrics

Previously in *"Computing intervals for the variance in estimation"* section, we obtained the posterior distribution of the ratio r corresponding to unweighted precision (Equation 25). Here, we will transform this for the weighted precision. For this purpose, we start by defining a weighting factor w_f equal to the ratio of weighted and unweighted true positives:

$$w_f = \frac{TP_w}{TP} \tag{40}$$

Using this, we can write the equation for weighted precision in terms of the unweighted ratio r:

$$\begin{aligned} r_w = \frac{TP_w}{TP_w + FP} &= \frac{w_f TP}{w_f TP + FP} \\ = \frac{\frac{w_f TP}{TP + FP}}{\frac{w_f TP + FP}{TP + FP}} &= \frac{w_f r}{(w_f - 1)r + 1} \end{aligned} \tag{41}$$

Thus, we transform the distribution given in Equation 25 according to the above equation to obtain the posterior distribution of the weighted ratio r_w. After that, we compute the 95% credible intervals as detailed before.

Quantifying the bias towards high-degree nodes in method predictions

Here, for each algorithm, we count the number of overlapping edges with the predictions from preferential attachment (i.e., the number of pairs that are predicted as positive by both algorithms) for varying values of k (where k indicates the number of predictions for both algorithms). Next, we compute the bias metric based on the area under this curves and normalize it with respect to the maximum & expected values to confine in [-1, 1] region (the maximum is equal to k, whereas the expected value for random predictions is k^2 divided by the total number of pairs to be predicted).

Separability analysis quantifying the imbalance in benchmarking data

Here, we first compute the preferential attachment scores (using the node degrees in training data) for the positives (the hidden interactions in test set) and negative node pairs (pairs without a known interaction). Next, we make use of the kolmogorov-smirnov statistic (which corresponds to the maximum distance in the cumulative distribution functions of) to quantify the predictive power of node degree information. Note that, this way of quantifying the separability is equivalent to computing informedness at the best prediction point (i.e., the maximum vertical difference in the ROC curve and the diagonal line corresponding to random predictions) for the predictions of preferential attachment model. For the weighted version of the separability analysis, we simply use the optimized edge weights as instance weights while estimating the cumulative distribution function (CDF) for the positives and compute the kolmogorov-smirnov statistic as usual (this time using the CDF for the weighted positives).

Computing the influence of the nodes on evaluation

Influence of the node categories on the evaluation is computed based on the percentages given in Figure 3(b) and S. Figure 5(b) (i.e., based on the number of edges or the weights of edges). Here, the influence of mixed category edges are counted as as half for each category (e.g., a Poor-Rich edge provides half of its weight as influence to poor category, and the other half to rich category). Note that, this way of computing the node influence ensures that their total adds up to 1.

Performing stratified analysis

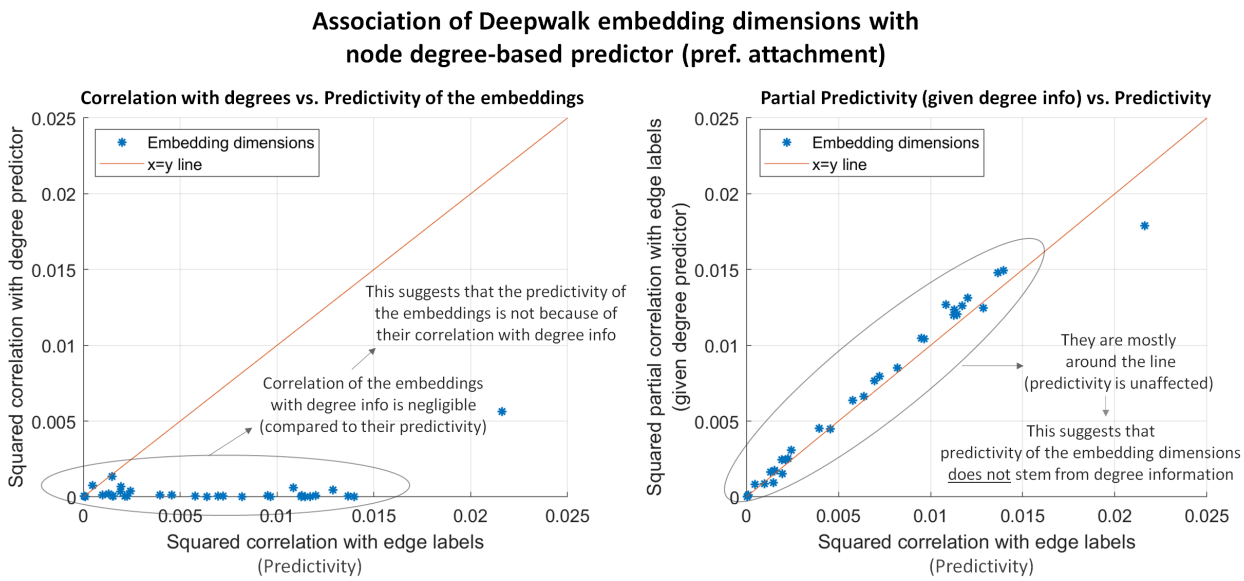
For the stratified analysis, we first obtain the node categories (Poor, Moderate, Rich) as shown in Figure 3a. Next, we assign each edge in the test set into one of six categories (e.g., Rich-Rich, Poor-Rich, Poor-Moderate and so on). For each of these categories separately, we repeat the evaluation (and compute performance metrics like AUPR), keeping only the edges in the corresponding category in the test set (in other words, considering the prediction of only the edges in that category to be true positives). Note that, the background set of possible node pairs is not affected by this stratification (i.e., the negative set includes pairs from all categories).

Datasets used in this work

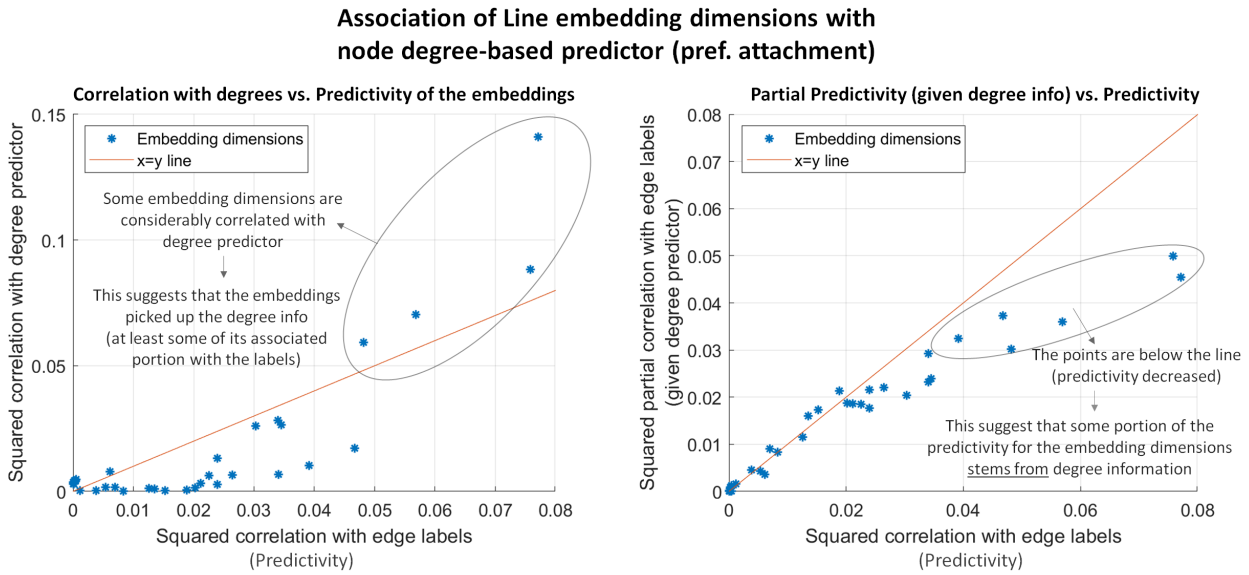
The bulk of the experiments done in this paper uses BioGRID [29] Human Protein-Protein Interaction network for two versions obtained at different times: (i) 2020 version (v4.0.189) contains 464,003 interactions between 25,776 proteins, (ii) 2022 version (v4.4.210) contains 784,774 interactions between 27,408 proteins. For constructing the training/test sets across time (2020 for training, new edges in 2022 for testing), we filter for the proteins that exist in the 2020 version and use the 308,334 new interactions for 16305 proteins in 2022 version as the test set (for a total of 772,337 interactions between 25,776 proteins, training & test sets combined).

The final part of our analysis includes six other networks listed below. Some of them were obtained and parsed from the source databases directly, while others are taken from BioNEV [1] repository as pre-processed edgelists.

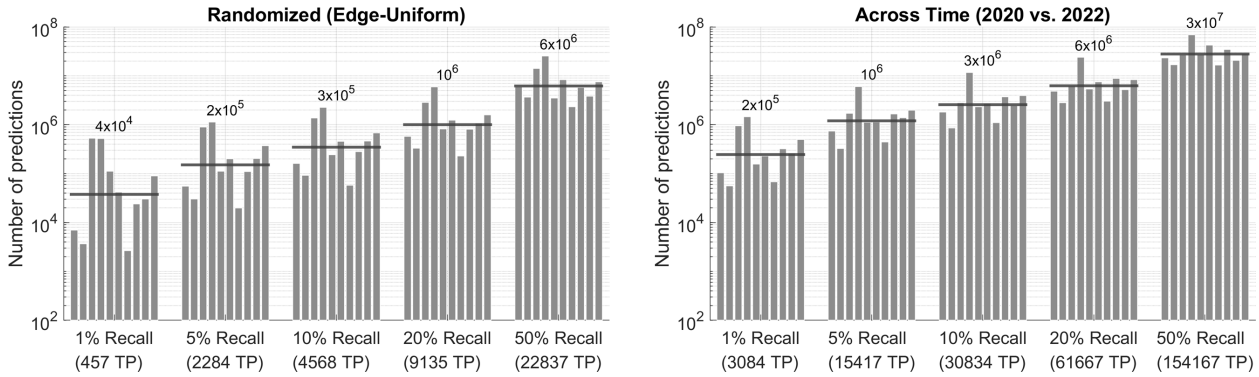
- STRING PPI [36] contains 359,776 interaction between 15,131 proteins. Taken from BioNEV repository as an unweighted undirected network.
- PhosphoSitePlus Kinase-Substrate (PSP-KS) dataset [37] contains 13,664 Kinase-phosphosite pairings. Taken from source (PhosphoSitePlus) and parsed by us as an unweighted undirected heterogeneous bipartite network. Filtered only to contain pairs observed in Human tissue.
- TRRUST [38] (Transcriptional Regulatory Relationships Unraveled by Sentence based Text mining) dataset contains 3,149 transcription-factor relationships between 1,621 genes. Taken from source and parsed by us as an unweighted undirected network. Filtered only to contain Activation relationships (as opposed to repression or unknown).
- Drugbank[39] Drug-Drug Interaction dataset contains 242,027 interactions between 2,191 drugs. Taken from BioNEV repository as an unweighted undirected network.
- NDFRT is a Disease-drug association dataset containing 56,515 associations between 13,545 diseases and drugs. Taken from BioNEV repository as an unweighted undirected heterogeneous bipartite network.
- CTD [40] is a Disease-drug association dataset containing 92,813 associations between 12,765 diseases and drugs. Taken from BioNEV repository as a pre-processed unweighted undirected heterogeneous bipartite network.



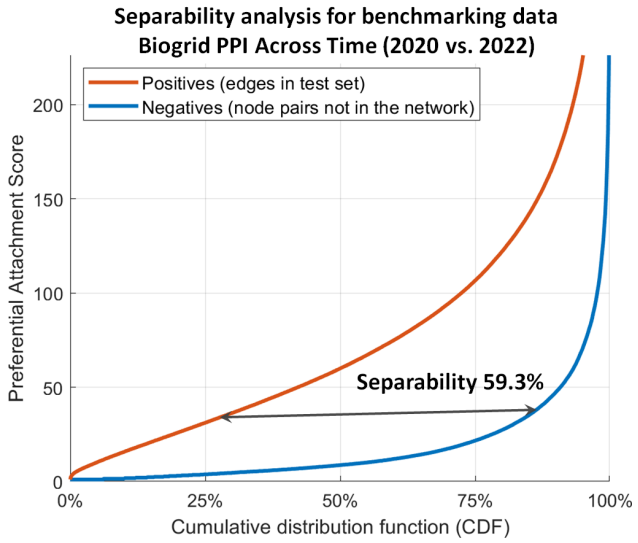
S. Figure 1: Investigating the embedding dimensions of Deepwalk in terms of their association with node degrees. The analysis suggests that the embeddings of Deepwalk does not depend on the degree information.



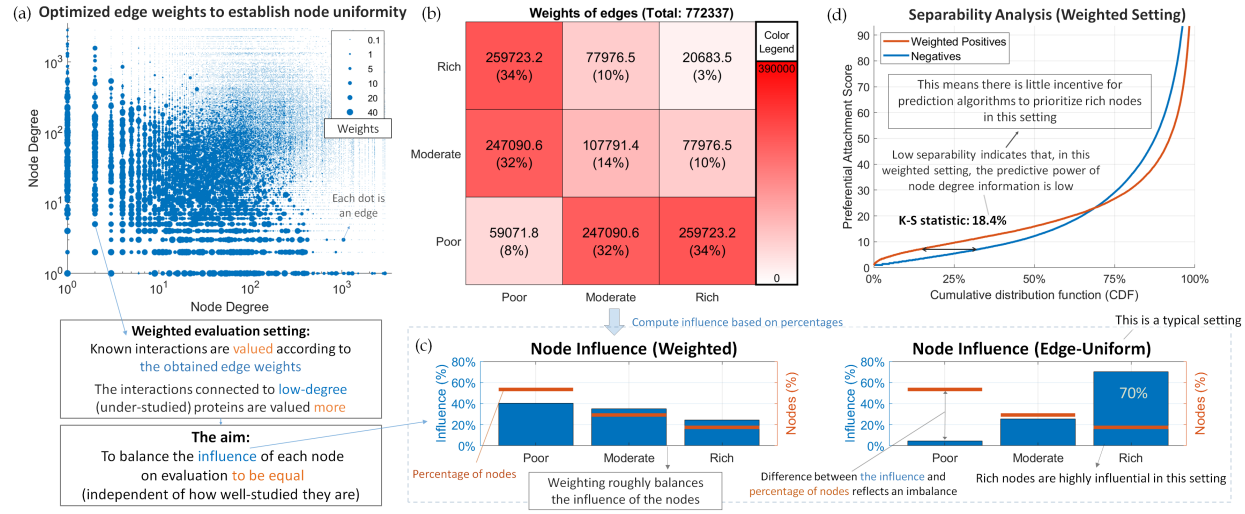
S. Figure 2: Investigating the embedding dimensions of Line in terms of their association with node degrees. The analysis suggests that Line embeddings picked up the node degree information and the predictivity of some of its embeddings dimensions stems from their correlation with node degrees.



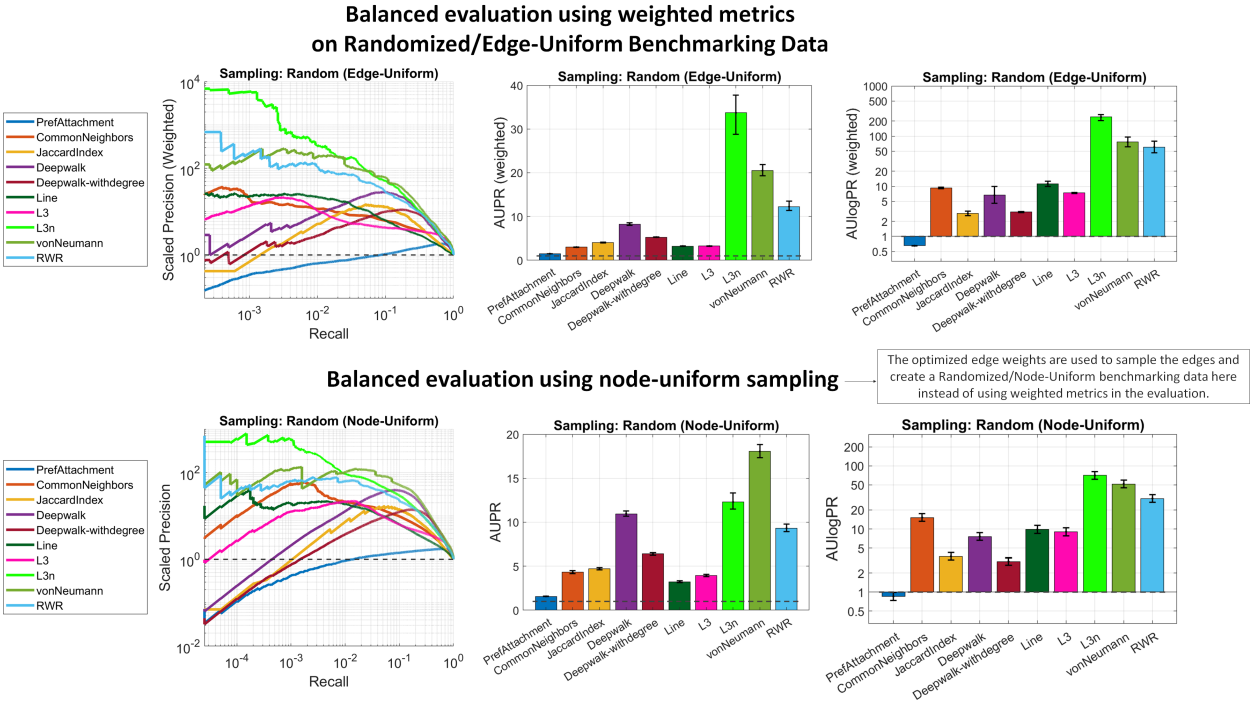
S. Figure 3: Number of predictions required to reach a particular recall threshold for Biogrid PPI predictions. (Left) The randomized (edge-uniform) evaluation. (Right) The across time evaluation setting. For both panels, the bars represent different link prediction algorithms. The horizontal lines and the numbers on the top indicate the geometric average of the number of predictions for each recall threshold.



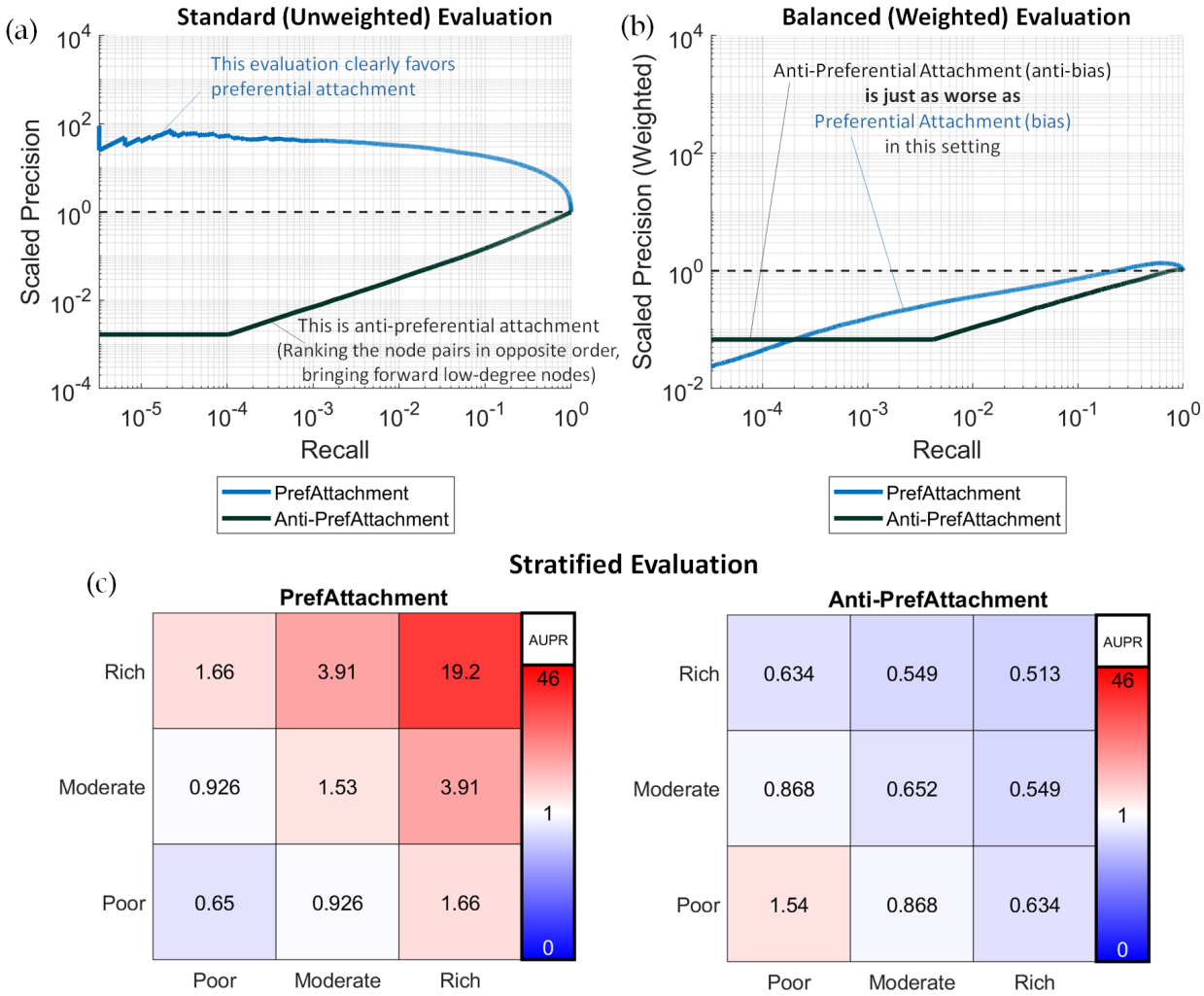
S. Figure 4: Separability analysis investigating the informedness of node degree information for Biogrid PPI across time (2020 vs. 2022) setting.



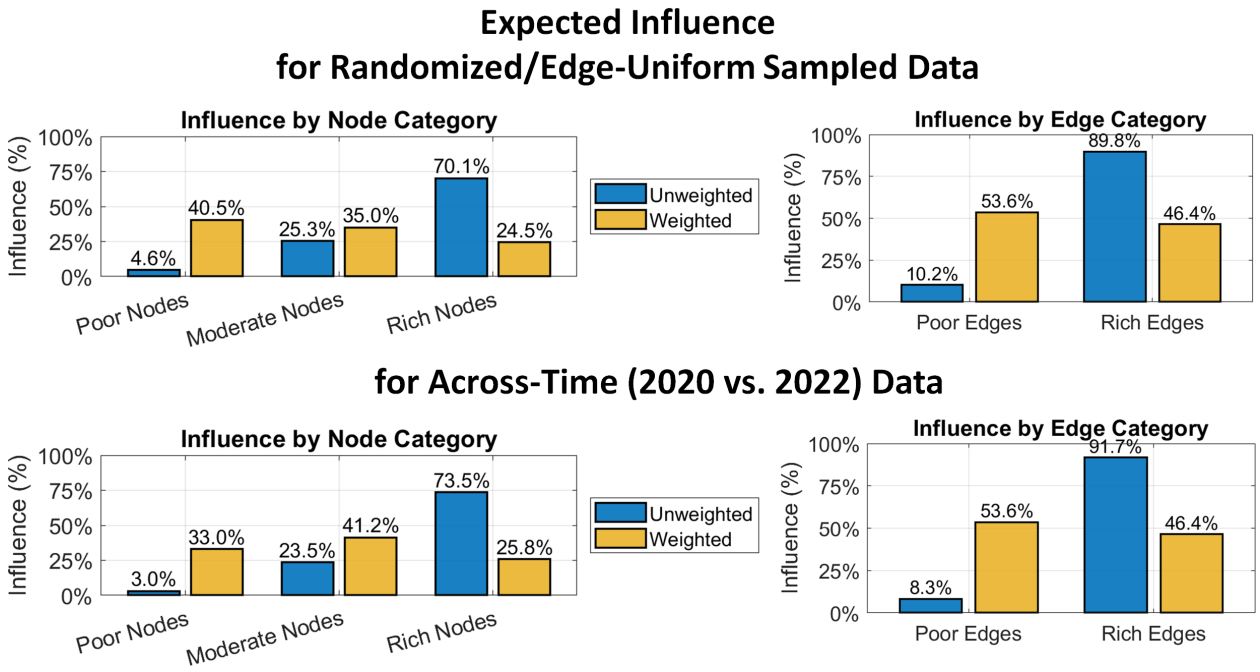
S. Figure 5: Mitigating degree bias in the evaluation of link prediction algorithms by assigning weights to edges during evaluation. Assignment of optimized edge weights establishes node-uniformity and balances the influence of nodes on evaluation. (a) Visualization of the optimized edge weights with respect to the degrees of incident nodes. The size of each point reflects the assigned weight of the corresponding edge. (b) The total weight of the edges by node category. (c) Influence of the nodes on evaluation (shown as bars) with respect to the node categories for weighted (node-uniform) and unweighted (edge-uniform) settings. (c) Separability analysis for the weighted (node-uniform) evaluation setting.



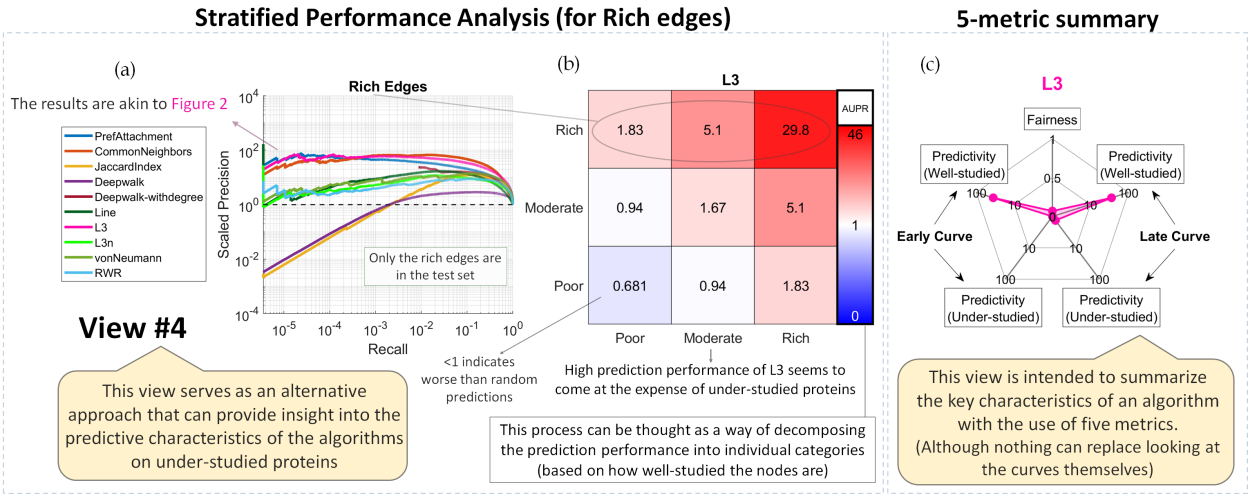
S. Figure 6: Balanced/Weighted evaluation results on randomized (sampled) benchmarking data for Biogrid PPI predictions. (Top) Balanced evaluation using weighted metrics. (Bottom) Balanced evaluation via node-uniform sampling (using the weights as sampling probabilities)



S. Figure 7: Comparison of preferential attachment (biased baseline) and anti-preferential attachment (anti-biased baseline) in different evaluation settings on Biogrid PPI predictions. Across-time (2020 vs. 2022) snapshots of the network are used as the benchmarking data (i.e., train/test splits) in this analysis. (a & b) Precision-Recall curves for the preferential attachment and anti-preferential attachment models respectively for standard (unweighted) and balanced (weighted) evaluation settings. (c) Stratified performance analysis results for preferential attachment and anti-preferential attachment algorithms. Each cell indicates the prediction performance of the algorithms for the corresponding edge category (e.g., for Poor-Rich category, only the edges that are between poor and rich nodes are included in the test set).



S. Figure 8: Expected influence for different categories of nodes or edges based on node degrees for randomized/edge-uniform and across-time bencharking data.



S. Figure 9: Stratified performance analysis for Rich edges connnected to well-studied nodes and the 5-metric summary for the best performing method on rich edges. (a) Precision-Recall performance curves for rich edges in log-log scale. (b) Late curve prediction performance (AUPR) stratified by node categories for L3 algorithm. (c) 5-metric summary for L3.