

A Hidden Challenge of Link Prediction: Which Pairs to Check?

Caleb Belth
University of Michigan
 Ann Arbor, MI, USA
 cbelth@umich.edu

Alican Büyükcakir
University of Michigan
 Ann Arbor, MI, USA
 alicanb@umich.edu

Danai Koutra
University of Michigan
 Ann Arbor, MI, USA
 dkoutra@umich.edu

Abstract—The traditional setup of link prediction in networks assumes that a test set of node pairs, which is usually balanced, is available over which to predict the presence of links. However, in practice, there is no test set: the ground-truth is not known, so the number of possible pairs to predict over is quadratic in the number of nodes in the graph. Moreover, because graphs are sparse, most of these possible pairs will not be links. Thus, link prediction methods, which often rely on proximity-preserving embeddings or heuristic notions of node similarity, face a vast search space, with many pairs that are in close proximity, but that should not be linked. To mitigate this issue, we introduce LINKWALDO, a framework for choosing from this quadratic, massively-skewed search space of node pairs, a concise set of candidate pairs that, in addition to being in close proximity, also *structurally resemble the observed edges*. This allows it to ignore some high-proximity but low-resemblance pairs, and also identify high-resemblance, lower-proximity pairs. Our framework is built on a model that theoretically combines Stochastic Block Models (SBMs) with node proximity models. The block structure of the SBM maps out *where* in the search space new links are expected to fall, and the proximity identifies the most plausible links within these blocks, using locality sensitive hashing to avoid expensive exhaustive search. LINKWALDO can use any node representation learning or heuristic definition of proximity, and can generate candidate pairs for any link prediction method, allowing the representation power of current and future methods to be realized for link prediction *in practice*. We evaluate LINKWALDO on 13 networks across multiple domains, and show that on average it returns candidate sets containing 7-33% more missing and future links than both embedding-based and heuristic baselines’ sets.

I. INTRODUCTION

Link prediction is a long-studied problem that attempts to predict either missing links in an incomplete graph, or links that are likely to form in the future. This has applications in discovering unknown protein interactions to speed up the discovery of new drugs, friend recommendation in social networks, knowledge graph completion, and more [1], [15], [16], [25]. Techniques range from heuristics, such as predicting links based on the number of common neighbors between a pair of nodes, to machine learning techniques, which formulate the link prediction problem as a binary classification problem over node pairs [7], [29].

Link prediction is often evaluated via a *ranking*, where pairs of nodes that are not currently linked are sorted based on the “likelihood” score given by the method being evaluated [16]. To construct the ranking, a “ground-truth” test set of node pairs

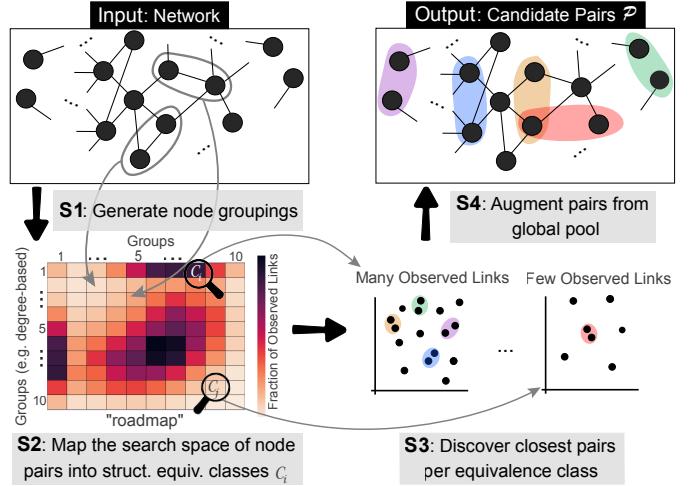


Fig. 1: Our proposed framework LINKWALDO chooses candidate pairs from the quadratic, highly-skewed search space of possible links by first constructing a *roadmap*, which partitions the search space into structural equivalence classes of node pairs to capture how much pairs in each location resemble the *observed* links. This roadmap tells LINKWALDO how closely to look in each section of the search space. LINKWALDO follows the roadmap, selecting from each equivalence class the node pairs in closest proximity.

is constructed by either (1) removing a certain percentage of links from a graph at random or (2) removing the newest links that formed in the graph, if edges have timestamps. These removed edges form the test positives, and the same number of unlinked pairs are generated at random as test negatives. The methods are then evaluated on how well they are able to rank the test positives higher than the test negatives.

However, when link prediction is applied in practice, these ground truth labels are not known, since that is the very question that link prediction is attempting to answer. Instead, *any* pair of nodes that are not currently linked could link in the future. Thus, to identify likely missing or future links, a link prediction method would need to consider $O(n^2)$ node pairs for a graph with n nodes; most of which in sparse, real-world networks would turn out to not link. Proximity, on its own, is only a weak signal, sufficient to rank pairs in a balanced test set, but likely to turn up many false positives in an asymptotically skewed space, leaving discovering the relatively small number of missing or future links a challenging problem.

Proximity-based link prediction heuristics [15], such as

Common Neighbors, could ignore some of the search space, such as nodes that are farther than two hops from each other, but this would not extend to other notions of proximity, like proximity-preserving embeddings. Duan *et al.* studied the problem of pruning the search space [5], but formulated it as top- k link prediction, which attempts to predict a small number of links, but misses a large number of missing links in the process, suffering from low recall.

The goal of this work is to develop a principled approach to choose, from the quadratic and skewed space of possible links, a set of candidate pairs for a link prediction method to make decisions about. We envision that this will allow current and future developments to be realized for link prediction in practice, where no ground-truth set is available.

Problem 1. *Given a graph and a proximity function between nodes, we seek to return a candidate set of node pairs for a link predictor to make decisions about, such that the set is significantly smaller than the quadratic search space, but contains many of the missing and future links.*

Our insight to handle the vast number of negatives is to consider not just the proximity of nodes, but also their *structural resemblance* to observed links. We measure resemblance as the fraction of observed links that fall in inferred, graph-structural equivalence classes of node pairs. For example, Fig. 1 shows one possible grouping of nodes based on their degrees, where the resulting structural equivalence classes (the cells in the “roadmap”) capture what fraction of observed links form between nodes of different degrees. Based on the roadmap, equivalence classes with a high fraction of observed edges are expected to contain more unlinked pairs than those with lower resemblance. We then employ node proximity within equivalence classes, rather than globally, which decreases false positives that are in close proximity, but do not resemble observed links, and decreases false negatives that are farther away in the graph, but resemble many observed edges. Moreover, to avoid computing proximities for all pairs of nodes within each equivalence class, we extend self-tuning locality sensitive hashing (LSH). Our main **contributions** are:

- **Formulation & Theoretical Connections.** Going beyond the heuristic of proximity between nodes, we model the plausibility of a node pair being linked as *both* their proximity *and* their structural resemblance to observed links. Based on this insight, we propose Future Link Location Models (FLLM), which combine Proximity Models and Stochastic Block Models; and we prove that Proximity Models are a naive special case. § III

- **Scalable Method.** We develop a scalable method, LINKWALDO (Fig. 1), which implements FLLM, and uses locality sensitive hashing to implicitly ignore unimportant pairs. § IV

- **Empirical Analysis.** We evaluate LINKWALDO on 13 diverse datasets from different domains, where it returns on average 22-33% more missing links than embedding-based models and 7-30% more than strong heuristics. § V

Our code is at <https://github.com/GemsLab/LinkWaldo>.

II. RELATED WORK

In this paper, we focus on the understudied problem of choosing candidate pairs from the quadratic space of possible links, for link prediction methods to make predictions about. Link prediction techniques range from heuristic definitions of similarity, such as Common Neighbors [15], Jaccard Similarity [15], and Adamic-Adar [1], to machine learning approaches, such as latent methods, which learn low-dimensional node representations that preserve graph-structural proximity in latent space [7], and GNN methods, which *learn* heuristics specific to each graph [29] or attempt to re-construct the observed adjacency matrix [10]. For detailed discussion of link prediction techniques, we refer readers to [15] and [16].

Selecting Candidate Pairs. The closest problem to ours is top- k link prediction [5], which attempts to take a particular link prediction method and prune its search space to directly return the k highest score pairs. One method [5] samples multiple subgraphs to form a bagging ensemble, and performs NMF on each subgraph, returning the nodes with the largest latent factor products from each, while leveraging early-stopping. The authors view their method’s output as predictions rather than candidates, and thus focus on high precision at small values of k relative to our setting. Another approach, Approximate Resistance Distance Link Predictor [21] generates spectral node embeddings by constructing a low-rank approximation of the graph’s *effective resistance* matrix, and applies a k -closest pairs algorithm on the embeddings, predicting these as links. However, this approach does not scale to moderate embedding dimensions (e.g., the dimensionality of 128 often-used used in embedding methods), and is often outperformed by the simple common neighbors heuristic.

A related problem is link-recommendation, which seeks to identify the k most relevant nodes to a query node. It has been studied in social networks for friend recommendation [26], and in knowledge graphs [9] to pick subgraphs that are likely to contain links to a given query entity. In contrast, we focus on candidate pairs globally, not specific to a query node.

III. THEORY

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph or network with $|\mathcal{V}| = n$ nodes and $|\mathcal{E}| = m$ edges, where $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. The adjacency matrix \mathbf{A} of \mathcal{G} is an $n \times n$ binary matrix with element $a_{ij} = 1$ if nodes i and j are linked, and 0 otherwise. The set of node v ’s neighbors is $\mathcal{N}(v) = \{u : (u, v) \in \mathcal{E}\}$. We summarize the key symbols used in this paper and their descriptions in Table I.

TABLE I: Description of major symbols.

Notation	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E}), \mathbf{A}$	Graph, nodes, edges, adjacency matrix
$ \mathcal{V} = n, \mathcal{E} = m$	Number of nodes resp. edges in \mathcal{G}
\mathcal{E}_{new}	Unobserved future or missing links
Γ, Π	Grouping of \mathcal{V} , Partition of $\mathcal{V} \times \mathcal{V}$
$\mathbf{x}_v \in \mathbf{X}, \boldsymbol{\mu}_v$	Node embedding and membership vector
\mathcal{C}_i	Equivalence class i
$\mathcal{P}, \tilde{\mathcal{P}}_G$	Pairs selected by LINKWALDO, global pool
k, κ	Budget for $ \mathcal{P} $, target for an equiv. class

We now formalize the problem that we seek to solve:

Problem 2. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a proximity function $\text{sim} : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}^+$ between nodes, and a budget $k \ll n^2$, return a set of plausible candidate node pairs $\mathcal{P} \subset \mathcal{V} \times \mathcal{V}$ of size $|\mathcal{P}| = k$ for a link predictor to make decisions about.

We describe next how to define resemblance in a principled way inspired by Stochastic Block Models, introduce a unified model for link prediction methods that use the proximity of nodes to rank pairs, and describe our model, which combines resemblance and proximity to solve Problem 2.

A. Stochastic Block Models

Stochastic Block Models (SBMs) are generative models of networks. They model the connectivity of graphs as emerging from the community or group membership of nodes [20].

Node Grouping. A node grouping Γ is a set of groups or subsets \mathcal{V}_i of the nodes that satisfies $\bigcup_{\mathcal{V}_i \in \Gamma} \mathcal{V}_i = \mathcal{V}$. It is called a *partition* if it satisfies $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset \forall \mathcal{V}_i \neq \mathcal{V}_j \in \Gamma$. Each node $v \in \mathcal{V}$ has a $|\Gamma|$ -dimensional binary membership vector $\boldsymbol{\mu}_v$, with element $\mu_{vi} = 1$ if v belongs to group \mathcal{V}_i .

A node grouping can capture community structure, but it can also capture other graph-structural properties, like the degrees of nodes, in which case the SBM captures the compatibility of nodes w.r.t degree—viz. degree assortativity.

Membership Indices. The membership indices $I_{u,v}$ of nodes u, v are the set of group ids (i, j) s.t. $u \in \mathcal{V}_i$ and $v \in \mathcal{V}_j$: $I_{u,v} \triangleq \{i : \mu_{u,i} = 1\} \times \{j : \mu_{v,j} = 1\}$, $i, j \in \{1, 2, \dots, |\Gamma|\}$.

Membership equivalence relation & classes. The membership indices form the equivalence relation \sim_I : $(u, v) \sim_I (u', v') \iff I_{u,v} = I_{u',v'}$. This induces a partition $\Pi = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_{|\Pi|}\}$ over all pairs of nodes $\mathcal{V} \times \mathcal{V}$ (both linked and unlinked), where the equivalence class \mathcal{C}_i contains all node pairs (u, v) with the same membership indices, i.e., $\boldsymbol{\mu}_u = \boldsymbol{\mu}$ and $\boldsymbol{\mu}_v = \boldsymbol{\mu}'$ for some $\boldsymbol{\mu}, \boldsymbol{\mu}' \in \{0, 1\}^{|\Gamma|}$. We denote the equivalence class of pair (u, v) as $[(u, v)]_{\sim_I}$.

Example 1. If nodes are grouped by their degrees to form Γ , then the membership indices $I_{u,v}$ of node pair (u, v) are determined by u and v 's respective degrees. For example, in Fig. 1, the upper circled node pair has degrees 3 and 5 respectively, which determines their equivalence class—in this case, the cell (3, 5) in the roadmap. Each cell of the roadmap corresponds to an equivalence class $\mathcal{C}_i \in \Pi$.

We can now formally define an SBM:

Definition 1 (Stochastic Block Model - SBM). Given a node grouping Γ and a $|\Gamma| \times |\Gamma|$ weight matrix \mathbf{W} specifying the propensity for links to form across groups, the probability that two nodes link given their group memberships is $\Pr(a_{uv} = 1 | \boldsymbol{\mu}_u, \boldsymbol{\mu}_v) = \sigma(\boldsymbol{\mu}_u^T \mathbf{W} \boldsymbol{\mu}_v)$, where function $\sigma(\cdot)$ converts the dot product to a probability (e.g., sigmoid) [18].

The vanilla SBM [20] assigns each node to one group (i.e., the grouping is a partition and membership vectors $\boldsymbol{\mu}$ are one-hot), in which case $\boldsymbol{\mu}_u^T \mathbf{W} \boldsymbol{\mu}_v = w_{I_{u,v}}$. The overlapping SBM

[18], [12] is a generalization that allows nodes to belong to multiple groups, in which case membership vectors may have multiple elements set to 1, and $\boldsymbol{\mu}_u^T \mathbf{W} \boldsymbol{\mu}_v = \sum_{i,j \in I_{u,v}} w_{ij}$.

Resemblance. Given an SBM with grouping Γ , we define the resemblance of node pair $(u, v) \in \mathcal{V} \times \mathcal{V}$ under the SBM as the percentage of the observed (training) edges that have the same group membership as (u, v) :

$$\rho(u, v) \triangleq \frac{|\{(v_1, v_2) \in \mathcal{E} : (v_1, v_2) \sim_I (u, v)\}|}{m}. \quad (1)$$

Example 2. In Figure 1, the resemblance $\rho(u, v)$ of node pair (u, v) corresponds to the density of the cell that it maps to. The high density in the border cells indicates that many low-degree nodes connect to high-degree nodes. The dense central cells indicate that mid-degree nodes connect to each other.

B. Proximity Models

Proximity-based link prediction models (PM) model the connectivity of graphs based on the proximity of nodes. Some methods define the proximity of nodes with a heuristic, such as Common Neighbors (CN), Jaccard Similarity (JS), and Adamic/Adar (AA). More recent approaches learn latent similarities between nodes, capturing the proximity in latent embeddings such that nodes that are in close proximity in the graph have similar latent embeddings (e.g., dot product) [7].

Node Embedding. A node embedding, $\mathbf{x}_v \in \mathbb{R}^d$, is a real-valued, d -dimensional vector representation of a node $v \in \mathcal{V}$. We denote all the node embeddings as matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$.

Definition 2 (Proximity Model - PM). Given a similarity or proximity function $\text{sim} : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}^+$ between nodes, the probability that nodes u and v link is an increasing function of their proximity: $\Pr(a_{uv} = 1 | \text{sim}(\cdot, \cdot)) = f(\text{sim}(u, v))$.

Instances of the PM include the Latent Proximity Model:

$$\text{sim}_{\text{LaPM}}(u, v) \triangleq \mathbf{x}_u^T \mathbf{x}_v, \quad (2)$$

where $\mathbf{x}_u, \mathbf{x}_v$ are the nodes' latent embeddings; and the Common Neighbors, Jaccard Similarity, and Adamic/Adar models:

$$\text{sim}_{\text{CN}}(u, v) \triangleq |\mathcal{N}(u) \cap \mathcal{N}(v)|, \quad (3)$$

$$\text{sim}_{\text{JS}}(u, v) \triangleq \frac{|\mathcal{N}(u) \cap \mathcal{N}(v)|}{|\mathcal{N}(u) \cup \mathcal{N}(v)|}, \text{ and} \quad (4)$$

$$\text{sim}_{\text{AA}}(u, v) \triangleq \sum_{v' \in \mathcal{N}(u) \cap \mathcal{N}(v)} \frac{1}{\log |\mathcal{N}(v')|}. \quad (5)$$

C. Proposed: Future Link Location Model

Unlike SBM and PM, our model, which we call the *Future Link Location Model* (FLLM), is not just modeling the probability of links, but rather *where* in the search space future links are likely to fall. To do so, FLLM uses a partition of the search space, and corresponding SBM, as a roadmap that gives the number of new edges expected to fall in each equivalence class. To formalize this idea, we first define two distributions:

New and Observed Distributions. The new link distribution $p_n(\mathcal{C}_i) \triangleq \Pr(\mathcal{C}_i | \mathcal{E}_{\text{new}})$ and the observed link distribution

$p_o(\mathcal{C}_i) \triangleq \Pr(\mathcal{C}_i | \mathcal{E})$ capture the fraction of new and observed edges that fall in equivalence class \mathcal{C}_i , respectively.

Definition 3 (Future Link Location Model - FLLM). *Given an overlapping SBM with grouping Γ , the expected number of new links in equivalence class \mathcal{C}_i is proportional to the number of observed links in \mathcal{C}_i , and the probability of node pair (u, v) linking is equal to the pair's resemblance times their proximity relative to other nodes in $[(u, v)]_{\sim_I}$:*

$$Pr(a_{uv} = 1 | \mu_u, \mu_v, sim(\cdot, \cdot)) = \rho(u, v) \cdot \frac{sim(u, v)}{\sum_{(u', v') \in [(u, v)]_{\sim_I}} sim(u', v')}.$$

FLLM employs the following theorem, which states that if $q\%$ of the observed links fall in equivalence class \mathcal{C}_i , then in expectation, $q\%$ of the unobserved links will fall in equivalence class \mathcal{C}_i . We initially assume that the unobserved future links follow the same distribution as the observed links—as generally assumed in machine learning—i.e., the relative fraction of links in each equivalence class will be the same for future links as observed links: $p_n = p_o$. In the next subsection, we show that for a fixed k , the error in this assumption is determined by the total variation distance between p_n and p_o , and hence is upper-bounded by a constant.

Theorem 1. *Given an overlapping SBM with grouping Γ inducing the partition Π of $\mathcal{V} \times \mathcal{V}$ for a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, out of k new (unobserved) links \mathcal{E}_{new} , the expected number that will fall in equivalence class \mathcal{C}_i and its variance are:*

$$\mathbb{E}[|\mathcal{C}_i \cap \mathcal{E}_{new}|] = \frac{k|\mathcal{C}_i \cap \mathcal{E}|}{m} \quad (6)$$

$$Var(|\mathcal{C}_i \cap \mathcal{E}_{new}|) = \frac{k|\mathcal{C}_i \cap \mathcal{E}||\mathcal{E} \setminus \mathcal{C}_i|}{m^2}. \quad (7)$$

Proof: Observe that the number of the k new edges that fall in equivalence class \mathcal{C}_i , i.e., $|\mathcal{C}_i \cap \mathcal{E}_{new}|$, is a binomial random variable over k trials, with success probability $\Pr(\mathcal{C}_i | \mathcal{E}_{new})$. Thus, the random variable's expected value is

$$\mathbb{E}[|\mathcal{C}_i \cap \mathcal{E}_{new}|] = k\Pr(\mathcal{C}_i | \mathcal{E}_{new}), \quad (8)$$

and its variance is

$$Var(|\mathcal{C}_i \cap \mathcal{E}_{new}|) = k\Pr(\mathcal{C}_i | \mathcal{E}_{new})(1 - \Pr(\mathcal{C}_i | \mathcal{E}_{new})). \quad (9)$$

We can derive $\Pr(\mathcal{C}_i | \mathcal{E}_{new})$ via $\Pr(\mathcal{C}_i | \mathcal{E})$ and Bayes' rule:

$$\Pr(\mathcal{C}_i | \mathcal{E}) = \frac{\Pr(\mathcal{E} | \mathcal{C}_i)\Pr(\mathcal{C}_i)}{\Pr(\mathcal{E})} = \frac{\frac{|\mathcal{C}_i \cap \mathcal{E}|}{|\mathcal{C}_i|} \frac{|\mathcal{C}_i|}{|\mathcal{V} \times \mathcal{V}|}}{\frac{m}{|\mathcal{V} \times \mathcal{V}|}} = \frac{|\mathcal{C}_i \cap \mathcal{E}|}{m}.$$

Combining the last equation with Eq. (8) results directly in Eq. (6), and by substituting into Eq. (9) we obtain:

$$Var(|\mathcal{C}_i \cap \mathcal{E}_{new}|) = \frac{k|\mathcal{C}_i \cap \mathcal{E}|}{m} \left(1 - \frac{|\mathcal{C}_i \cap \mathcal{E}|}{m}\right) = \frac{k|\mathcal{C}_i \cap \mathcal{E}||\mathcal{E} \setminus \mathcal{C}_i|}{m^2},$$

where we used the fact that $|\mathcal{E} \setminus \mathcal{C}_i| = |\mathcal{E}| - |\mathcal{C}_i \cap \mathcal{E}|$. ■

D. Guarantees on Error

While this derivation assumed that the future link distribution is the same as the observed link distribution, we now show that for a fixed k , the amount of error incurred when this assumption does not hold is entirely dependent on the total variation distance, and hence is upper-bounded by $2k$. *Total Variation Distance.* The total variation distance [27] between p_n and p_o , which is a metric, is defined as

$$d_{TV}(p_n, p_o) \triangleq \sup_{\mathcal{A} \subset \Pi} |p_n(\mathcal{A}) - p_o(\mathcal{A})|. \quad (10)$$

Total Error. The total error made in the approximation of $\mathbb{E}[|\mathcal{C}_i \cap \mathcal{E}_{new}|]$ using Eq. (6) is defined as

$$\begin{aligned} \xi &\triangleq \sum_{\mathcal{C}_i \in \Pi} |\hat{\mathbb{E}}[|\mathcal{C}_i \cap \mathcal{E}_{new}|] - \mathbb{E}[|\mathcal{C}_i \cap \mathcal{E}_{new}|]| \\ &= \sum_{\mathcal{C}_i \in \Pi} |kp_o(\mathcal{C}_i) - kp_n(\mathcal{C}_i)|, \end{aligned} \quad (11)$$

where $\hat{\mathbb{E}}[|\mathcal{C}_i \cap \mathcal{E}_{new}|]$ is the true expected value regardless of whether or not $p_n = p_o$ holds.

Theorem 2. *The total error incurred over Π in the computation of the expected number of new edges that fall in each $\mathcal{C}_i \in \Pi$ is an increasing function of the the number of new pairs k and the total variation distance between p_n and p_o . Furthermore, it has the following upper-bound:*

$$\xi = 2k d_{TV}(p_n, p_o) \leq \min(2k, 2k\sqrt{1/2D_{KL}(p_n || p_o)}). \quad (12)$$

Proof: From the definition of total error in Eq. (11), the first equality holds from [14]. The inequality holds based on the fact that $d_{TV}(\cdot, \cdot)$ ranges in $[0, 1]$, and Pinsker's inequality [27], which upper-bounds $d_{TV}(\cdot, \cdot)$ via KL-divergence. ■

E. Proximity Model as a Special Case of FLLM

The PM, defined in § III-B, is a special case of FLLM, where FLLM's grouping contains just one group $\Gamma = \{\mathcal{V}\}$. That is, if the nodes are not grouped, then the models give the same result. Thus, FLLM's improvement over LaPM is a result of using structurally-meaningful groupings over the graph. The following theorem states this result formally.

Theorem 3. *For a single node grouping $\Gamma = \{\mathcal{V}\}$, both PM and FLLM give the same ranking of pairs $(u, v) \in \mathcal{V} \times \mathcal{V}$:*

$$\begin{aligned} Pr_{PM}(a_{uv} = 1 | sim(\cdot, \cdot)) &> Pr_{PM}(a_{u'v'} = 1 | sim(\cdot, \cdot)) \iff \\ Pr_{FLLM}(a_{uv} = 1 | \mu_u, \mu_v, sim(\cdot, \cdot)) &> Pr_{FLLM}(a_{u'v'} = 1 | \mu_{u'}, \mu_{v'}, sim(\cdot, \cdot)). \end{aligned}$$

Proof: Since $\Gamma = \{\mathcal{V}\}$, $\Pi = \{\mathcal{V} \times \mathcal{V}\}$, and since $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, all observed edges fall in the lone equivalence class $\mathcal{C} = \mathcal{V} \times \mathcal{V}$. Thus $\rho(u, v) = 1 \forall (u, v) \in \mathcal{V} \times \mathcal{V}$. Since there is only one equivalence class, the denominator in Dfn. 3 is equal to a constant $c \triangleq \sum_{(u', v') \in [(u, v)]_{\sim_I}} sim(u', v') = \sum_{(u', v') \in \mathcal{V} \times \mathcal{V}} sim(u', v') \forall (u, v) \in \mathcal{V} \times \mathcal{V}$. Therefore, $\Pr_{FLLM}(a_{uv} = 1 | \mu_u, \mu_v, sim(\cdot, \cdot)) = \frac{1}{c} sim(u, v)$, and both models are increasing functions of $sim(\cdot, \cdot)$. ■

IV. METHOD

We solve Problem 2 by using our FLLM model in a new method, LINKWALDO, shown in Fig. 1, which has four steps:

- **S1:** Generate node groupings and equivalence classes.
- **S2:** Map the search space, deciding how many candidate pairs to return from each equivalence class.
- **S3:** Search each equivalence class, returning directly the highest-proximity pairs, and stashing some slightly lower-proximity pairs in a global pool.
- **S4:** Choose the best pairs from the global pool to augment those returned from each equivalence class.

We discuss these steps next, give pseudocode in Alg. 1, and discuss time complexity in the appendix.

A. Generating Node Groupings (**S1**)

In theory, we would like to infer the groupings that directly maximize the likelihood of the observed adjacency matrix. However, the techniques for inferring these groupings (and the corresponding node membership vectors) are computationally intensive, relying on Markov chain Monte Carlo (MCMC) methods [17]. Indeed, these methods are generally applied in networks with only up to a few hundred nodes [18]. In cases where n is large enough that considering all $O(n^2)$ node pairs would be computationally infeasible, so would be MCMC. Instead LINKWALDO uses a fixed grouping, though it is agnostic to how the nodes are grouped. We discuss a number of sensible groupings below, and discuss how to set the number of groups in § V-D. Any other grouping can be readily used within our framework, but should be carefully chosen to lead to strong results.

- *Log-binned Node Degree (DG).* This grouping captures degree assortativity [19], i.e., the extent to which low degree nodes link with other low degree nodes vs. high degree nodes, by creating uniform bins in log-space (e.g., Fig. 1; linear bins).
- *Structural Embedding Clusters (SG).* This grouping extends DG by clustering latent node embeddings that capture structural roles of nodes [24].
- *Communities (CG).* This grouping captures community structure by clustering proximity preserving latent embeddings or using community detection methods.
- *Multiple Groupings (MG).* Any subset of these groupings or any other groupings can be combined into a new grouping, by setting μ_v element(s) to 1 for v 's membership in each grouping, since nodes can have overlapping group memberships.

B. Mapping the Search Space (**S2**)

LINKWALDO's approach to mapping the search space (i.e., identifying how many pairs to return per class \mathcal{C}_i) follows directly from Thm. 1. LINKWALDO computes the expected number of pairs in each equivalence class based on Eq. (6) and its variance based on Eq. (7), as a measure of the uncertainty. When LINKWALDO searches each equivalence class \mathcal{C}_i , it returns the expected number of pairs *minus* a standard deviation *directly*, and adds more pairs, up to a standard deviation past the mean, to a global pool $\tilde{\mathcal{P}}_G$. Thus, LINKWALDO

Algorithm 1 LINKWALDO(\mathcal{G} , $sim(\cdot, \cdot)$, k , τ)

```

1: /* S1: Generating Node Groupings */
2: Generate node grouping  $\Gamma$  inducing partition  $\Pi$            ▷ § IV-A
3:  $\mathcal{P}, \tilde{\mathcal{P}}_G \leftarrow \emptyset, \emptyset$           ▷ Initialize pairs to return and global pool
4: for  $\mathcal{C}_i \in \Pi$  do          ▷ Search each equivalence class § IV-C
5:   /* S2: Mapping the Search Space */
6:    $\bar{\mu} \leftarrow \mathbb{E}[|\mathcal{C}_i \cap \mathcal{E}_{\text{new}}|]$           ▷ Eq. (6)
7:    $\sigma \leftarrow \sqrt{\text{Var}(|\mathcal{C}_i \cap \mathcal{E}_{\text{new}}|)}$           ▷ Eq. (7)
8:   /* S3: Discovering Closest Pairs per Equivalence Class */
9:   if  $|\mathcal{C}_i| < \tau$  then
10:    SELECTPAIRSEXACT( $\mathcal{P}, \tilde{\mathcal{P}}_G, \mathcal{C}_i, \bar{\mu}, \sigma$ )
11:   else
12:    SELECTPAIRSAPPROX( $\mathcal{P}, \tilde{\mathcal{P}}_G, \mathcal{C}_i, \bar{\mu}, \sigma$ )
13:   /* S4: Augmenting Pairs from Global Pool */
14:    $\mathcal{P} \leftarrow \mathcal{P} \cup \{\text{top } k - |\mathcal{P}| \text{ pairs from } \tilde{\mathcal{P}}_G\}$ 
15:   return  $\mathcal{P}$ 
16: procedure SELECTPAIRSEXACT( $\mathcal{P}, \tilde{\mathcal{P}}_G, \mathcal{C}_i, \bar{\mu}, \sigma$ )
17:   Sort pairs  $(u, v) \in \mathcal{C}_i$  in descending order on  $sim(u, v)$ 
18:    $\mathcal{P} \leftarrow \mathcal{P} \cup \{\text{top } \bar{\mu} - \sigma \text{ pairs}\}$ 
19:    $\tilde{\mathcal{P}}_G \leftarrow \tilde{\mathcal{P}}_G \cup \{\text{next } 2\sigma \text{ pairs}\}$ 
20: procedure SELECTPAIRSAPPROX( $\mathcal{P}, \tilde{\mathcal{P}}_G, \mathcal{C}_i, \bar{\mu}, \sigma$ )
21:   for  $i = 1, 2, \dots, r$  do          ▷ Create  $r$  trees
22:      $\mathcal{B} \leftarrow \{(\mathcal{V}_u, \mathcal{V}_v)\}$           ▷ buckets start off as the root
23:     while  $\text{Vol}(\mathcal{B}) > \kappa$  do          ▷ cf. Prob. 3 for  $\kappa$  definition
24:       Choose  $h(\cdot)$  at random from  $\mathcal{H}_{\text{rh}}$ 
25:        $\mathcal{B}' \leftarrow \emptyset$           ▷ Create new buckets
26:       for  $\beta \in \mathcal{B}$  do          ▷ Branch each leaf (bucket)
27:          $\beta'_{\text{left}} \leftarrow \{u \in \mathcal{V}_u^{(\beta)} : h(\mathbf{x}_u) < 0\}, \{v \in \mathcal{V}_v^{(\beta)} : h(\mathbf{x}_v) < 0\}$ 
28:          $\beta'_{\text{right}} \leftarrow \{u \in \mathcal{V}_u^{(\beta)} : h(\mathbf{x}_u) \geq 0\}, \{v \in \mathcal{V}_v^{(\beta)} : h(\mathbf{x}_v) \geq 0\}$ 
29:          $\mathcal{B}' \leftarrow \mathcal{B}' \cup \{\beta'_{\text{left}}, \beta'_{\text{right}}\}$ 
30:       if  $\text{Vol}(\mathcal{B}) \leq \kappa$  then  $\mathcal{B} \leftarrow \mathcal{B}'$ 
31:      $\mathcal{P} \leftarrow \mathcal{P} \cup \{\text{top } \bar{\mu} - \sigma \text{ pairs}\}$ 
32:      $\tilde{\mathcal{P}}_G \leftarrow \tilde{\mathcal{P}}_G \cup \{\text{next } 2\sigma \text{ pairs}\}$ 

```

adds into \mathcal{P} the $\mathbb{E}[|\mathcal{C}_i \cap \mathcal{E}_{\text{new}}|] - \sqrt{\text{Var}(|\mathcal{C}_i \cap \mathcal{E}_{\text{new}}|)}$ pairs in closest proximity in equivalence class \mathcal{C}_i , and the next $2\sqrt{\text{Var}(|\mathcal{C}_i \cap \mathcal{E}_{\text{new}}|)}$ closest pairs into the global pool $\tilde{\mathcal{P}}_G$ (both expressions are rounded to the nearest integer). Node pairs that are already linked are skipped.

C. Discovering Closest Pairs per Equivalence Class (**S3**)

We now discuss how LINKWALDO discovers the κ closest unlinked pairs *within* each equivalence class (Fig. 1), where κ is determined in step **S2** based on the expected number of pairs in the equivalence class, and variance (uncertainty).

Problem 3. Given an equivalence class \mathcal{C}_i , return the top- κ unlinked pairs in \mathcal{C}_i in closest proximity $sim(\cdot, \cdot)$, where $\kappa = \mathbb{E}[|\mathcal{C}_i \cap \mathcal{E}_{\text{new}}|] + \sqrt{\text{Var}(|\mathcal{C}_i \cap \mathcal{E}_{\text{new}}|)}$ (based on **S2**).

For equivalence classes smaller than some tolerance τ , it is feasible to search all pairs of nodes exhaustively. However, for $|\mathcal{C}_i| > \tau$, this should be avoided, to make the search practical. We first discuss this case when using the dot product similarity $sim_{\text{LaPM}}(\cdot, \cdot)$ in Eq. (2), and then discuss it for other similarity models (CN, JS, and AA) given by Eqs. (3)-(5). Finally, we introduce a refinement that improves the robustness of LINKWALDO against errors in proximity.

1) *Avoiding Exhaustive Search for Dot Product:* In the case of dot product, we use Locality Sensitive Hashing (LSH)

[28] to avoid searching all $|\mathcal{C}_i|$ pairs. LSH functions have the property that the probability of two items colliding is a function of their similarity. We use the following fact:

Fact 1. *The equivalence class \mathcal{C}_i can be decomposed into the Cartesian product of two sets $\mathcal{C}_i = \mathcal{V}_u \times \mathcal{V}_v$, where $\mathcal{V}_u \triangleq \{u : \mu_u = \mu\}$ and $\mathcal{V}_v = \{v : \mu_v = \mu'\}$.*

At a high level, to solve Prob. 3, we hash each node embedding of the nodes in \mathcal{V}_u and \mathcal{V}_v using a locality sensitive hash function. We design the hash function, described next, such that the number of pairs that map to the same bucket is greater than κ , but as small as possible, to maximally prune pairs. Once the embeddings are hashed, we search the pairs in each hash bucket for the κ closest. We normalize the embeddings so that dot product is equivalent to cosine similarity, and use the Random Hyperplane LSH family [4].

Definition 4 (Random Hyperplane Hash Family). *The random hyperplane hash family \mathcal{H}_{rh} is the set of hash functions $\mathcal{H}_{rh} \triangleq \{h : \mathbb{R}^d \rightarrow \{0, 1\}\}$, where \mathbf{r}_h is a random d -dimensional Gaussian unit vector and $h(\mathbf{x}) \triangleq \begin{cases} 1 & \text{if } \mathbf{r}_h^T \mathbf{x} \geq 0 \\ 0 & \text{if } \mathbf{r}_h^T \mathbf{x} < 0 \end{cases}$.*

This hash family is well-known to provide the property that the probability of two vectors colliding is a function of the degree of the angle between them [2]:

$$\Pr(h(\mathbf{x}_u) = h(\mathbf{x}_v)) = 1 - \frac{\theta(\mathbf{x}_u, \mathbf{x}_v)}{\pi} = 1 - \frac{\arccos(\mathbf{x}_u^T \mathbf{x}_v)}{\pi},$$

where the last equality holds due to normalized embeddings.

To lower the false positive rate, it is conventional to form a new hash function by sampling b hash functions from \mathcal{H}_{rh} and concatenating the hash codes: $g(\cdot) = (h_1(\cdot), h_2(\cdot), \dots, h_b(\cdot))$. The new hash function is from another LSH family:

Definition 5 (b -AND-Random Hyperplane Hash Family). *The b -AND-Random hyperplane hash family is the set of hash functions $\mathcal{H}_{and}^b \triangleq \{g : \mathbb{R}^d \rightarrow \{0, 1\}^b\}$, where $g(\mathbf{x}) = (h_1(\mathbf{x}), h_2(\mathbf{x}), \dots, h_b(\mathbf{x}))$ is formed by concatenating b randomly sampled hash functions $h(\cdot) \in \mathcal{H}_{rh}$ for some $b \in \mathbb{N}$.*

Since the hash functions are sampled randomly from \mathcal{H}_{rh} ,

$$\Pr(g(\mathbf{x}_u) = g(\mathbf{x}_v)) = \left(1 - \frac{\arccos(\mathbf{x}_u^T \mathbf{x}_v)}{\pi}\right)^b.$$

Only vectors that are not split by all b random hyperplanes end up with the same hash codes, so this process lowers the false positive rate. However, it also increases the false negative rate for the same reason. The conventional LSH-scheme then repeats the process r times, computing the dot product exactly over all pairs that match in at least one b -dim hash code, in order to lower the false negative rate. The challenge of this approach is determining how to set b . To do so, we first define the hash buckets of a hash function, and their volume.

Definition 6 (Hash Buckets and Volume). *Given an equivalence class $\mathcal{C}_i = \mathcal{V}_u \times \mathcal{V}_v$ and a hash function $g(\cdot) : \mathbb{R}^d \rightarrow$*

$\{0, 1\}^b$, after applying $g(\cdot)$ to all $v \in \mathcal{V}_u \cup \mathcal{V}_v$, a hash bucket

$$\beta = \{u \in \mathcal{V}_u, v \in \mathcal{V}_v : g(u) = g(v) = \beta_{hashcode}\}$$

consists of subsets $\mathcal{V}_u^{(\beta)} \subseteq \mathcal{V}_u, \mathcal{V}_v^{(\beta)} \subseteq \mathcal{V}_v$ of nodes that mapped to hashcode $\beta_{hashcode} \in \{0, 1\}^b$. The set of hash buckets $\mathcal{B}_g = \{\beta : |\beta| > 0\}$ consists of all non-empty buckets. We define the volume of the buckets as the number of pairs (u, v) where u and v landed in the same bucket:

$$Vol(\mathcal{B}_g) \triangleq |\{(u, v) : g(\mathbf{x}_u) = g(\mathbf{x}_v)\}| = \sum_{\beta \in \mathcal{B}_g} |\mathcal{V}_u^{(\beta)} \times \mathcal{V}_v^{(\beta)}|.$$

Since we are after the κ closest pairs, we want to find a hash function $g(\cdot)$ such that $Vol(\mathcal{B}_g) \geq \kappa$. But since we want to search as few pairs as possible, we seek the value of b that minimizes $Vol(\mathcal{B}_g)$ for some $g(\cdot) \in \mathcal{H}_{and}^b$ subject to the constraint that $Vol(\mathcal{B}_g) \geq \kappa$.

Any hash function $g \in \mathcal{H}_{and}^b$ corresponds to a binary prefix tree, like Fig. 2. Each level of the tree corresponds to one $h \in \mathcal{H}_{rh}$, and the leaves correspond to the buckets \mathcal{B}_g . Thus, to automatically identify the best value of b , we can recursively grow the tree, branching each leaf with a new random hyperplane hash function $h \in \mathcal{H}_{rh}$, until $Vol(\mathcal{B}_g) < \kappa$, then undo the last branch. At that point, the depth of the tree equals b , and is the largest value such that $Vol(\mathcal{B}_g) \geq \kappa$. To prevent this process from repeating indefinitely in edge cases, we halt the branching at a maximum depth b_{max} . This approach is closely related to LSH Forests [2], but with some key differences, which we discuss below.

Theorem 4. *Given a hash function $g \in \mathcal{H}_{and}^b$ the κ closest pairs in \mathcal{C}_i are the κ most likely pairs to be in the same bucket: $\Pr(g(\mathbf{x}_u) = g(\mathbf{x}_v)) > \Pr(g(\mathbf{x}_{u'}) = g(\mathbf{x}_{v'})) \iff \mathbf{x}_u^T \mathbf{x}_v > \mathbf{x}_{u'}^T \mathbf{x}_{v'}$.*

Proof: Since $\arccos(x)$ is a decreasing function of x , Eq. (13) shows that $\Pr(g(\mathbf{x}_u) = g(\mathbf{x}_v))$ is an increasing function of $\mathbf{x}_u^T \mathbf{x}_v$. The result follows from this. ■

While $\mathbf{x}_u^T \mathbf{x}_v > \mathbf{x}_{u'}^T \mathbf{x}_{v'}$ implies that (u, v) are more likely than (u', v') to be in the same bucket, it does not guarantee that this outcome will always happen. Thus, we repeat the process r times, creating r binary prefix trees and, searching the pairs that fall in the same bucket in any tree for the top κ . Setting the r parameter is considered of minor importance, as long as it is sufficiently large (e.g., 10) [2].

Differences from LSH Forests [2]. LSH Forests are designed for KNN -search, which seeks to return the nearest neighbors to a *query vector*. In contrast, our approach is designed for κ -closest-pairs search, which seeks to return the κ closest pairs in a set \mathcal{C}_i . LSH Forests grow each tree until each vector is in its own leaf. We grow each tree until we reach the target bucket volume κ . LSH Forests allow variable length hash codes, since the nearest neighbors of different query vectors may be at different relative distances. All our leaves are at the same depth so that the probability of (u, v) surviving together to the leaf is an increasing function of their dot product.

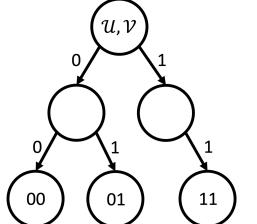


Fig. 2: LSH Tree

2) *Avoiding Exhaustive Search for Heuristics*: For the heuristic definitions of proximity in Eqs.(3)-(5), there are two approaches to solving Prob. 3. The first is to construct embeddings from the CN and AA scores (this does not apply to JS). For CN, if we let the node embeddings be their corresponding rows in the adjacency matrix, i.e., $\mathbf{X}_{\text{CN}} = \mathbf{A}$, then $\text{sim}_{\text{CN}}(u, v) = \mathbf{x}_u^T \mathbf{x}_v$. Similarly, $\mathbf{X}_{\text{AA}} = \mathbf{A} \cdot \mathbf{1} / \sqrt{\log(\mathbf{D})}$, yields $\text{sim}_{\text{AA}}(u, v) = \mathbf{x}_u^T \mathbf{x}_v$, where \mathbf{D} is a diagonal matrix recording the degree of each node. Thus, the LSH solution just described can be applied. The second approach uses the fact that all three heuristics are defined over the 1-hop neighborhoods of nodes (u, v) . Thus, to have nonzero proximity, (u, v) must be within 2-hops of each other, and any pairs not within 2-hops can implicitly be ignored.

3) *Bail Out Refinement*: To this point we have assumed that the proximity model used in LINKWALDO is highly informative and accurate. However, in reality, heuristics may not be informative for all equivalence classes, and even learned, latent proximity models, can fail to encode adequate information. For instance, it is challenging to learn high-quality representations for low-degree nodes. Thus, we introduce a refinement to LINKWALDO that automatically identifies when a proximity model is uninformative in an equivalence class, and allows it to *bail out* of searching that equivalence class.

Proximity Model Error. The error that a proximity model makes is the probability $\Pr(\text{sim}(u, v) < \text{sim}(u', v'))$ that it gives a higher proximity for some unlinked pair $(u', v') \notin \mathcal{E}$ than for some linked pair $(u, v) \in \mathcal{E}$.

By this definition of error, we expect strong proximity models to mostly assign higher proximity between observed edges than future or missing edges: $\Pr(\text{sim}(u, v) > \text{sim}(u', v')) \approx 1$ for some $(u, v) \in \mathcal{E}$ and $(u', v') \in \mathcal{E}_{\text{new}}$. Thus, on our way to finding the top- κ most similar (unlinked) pairs in an equivalence class (Problem 3), we expect to encounter a majority of the *observed* edges (linked pairs) $|\mathcal{E} \cap \mathcal{C}_i|$ that fall in that class. For a user-specified error tolerance ζ , LINKWALDO will bail out and return no pairs from any equivalence class where less than ζ fraction of its observed edges are encountered on the way to finding the κ most similar unlinked pairs. LINKWALDO keeps track of how many pairs were skipped by bailing out, and replaces them (after step S4) by adding to \mathcal{P} the top-ranked pairs of a heuristic (e.g., AA).

D. Augmenting Pairs from Global Pool (S4)

Since LINKWALDO returns a standard deviation below the expected number of new pairs in each equivalence class, it chooses the remaining pairs up to k from $\tilde{\mathcal{P}}_G$. To do so, it considers pairs in descending order on the input similarity function $\text{sim}(\cdot, \cdot)$, and greedily adds to \mathcal{P} until $|\mathcal{P}| = k$.

V. EVALUATION

We evaluate LINKWALDO on three research questions: **(RQ1)** Does the set \mathcal{P} returned by LINKWALDO have high recall and precision? **(RQ2)** Is LINKWALDO scalable? **(RQ3)** How do parameters affect performance?

TABLE II: Datasets statistics: if the graph is temporal or static, density, degree assortativity [19], and number of nodes and edges.

Graph	Time	Density	Assortativity	n	m
Yeast	-	0.41%	0.4539	2,375	11,693
DBLP	-	0.06%	-0.0458	12,595	49,638
Facebook1	-	1.08%	0.0636	4,041	88,235
MovieLens	✓	2.90%	-0.2268	2,627	100,000
HS-Protein	-	0.74%	0.2483	6,329	147,548
arXiv	-	0.11%	0.2051	18,772	198,110
MathOverflow	✓	0.06%	-0.1979	24,820	199,974
Enron	✓	0.01%	-0.1667	87,275	299,221
Reddit	✓	0.01%	-0.1278	67,180	309,667
Epinions	-	0.01%	-0.0406	75,881	405,741
Facebook2	✓	0.04%	0.1770	63,733	817,063
Digg	✓	< 0.01%	-0.0557	279,376	1,546,541
Protein-Soy	-	1.64%	-0.0192	45,116	16,691,679

A. Data & Setup

We evaluate LINKWALDO on a large, diverse set of networks: metabolic, social, communication, and information networks. Moreover, we include datasets to evaluate in both LP scenarios: (1) returning possible *missing* links in static graphs and (2) returning possible *future* links in temporal graphs. We treat all graphs as undirected.

Metabolic. Yeast [29], HS-Protein [11], and Protein-Soy [13] are metabolic protein networks, where edges denote known associations between proteins in different species. Yeast contains proteins in a species of yeast, HS-Protein in human beings, and Protein-Soy in Glycine max (soybeans).

Social. Facebook1 [13] and Facebook2 [11] capture friendships on Facebook, Reddit [13] encodes links between subreddits (topical discussion boards), edges in Epinions [11] connect users who trust each other's opinions, MathOverflow [13] captures comments and answers on math-related questions and comments (e.g., user u answered user v 's question), Digg [23] captures friendships among users.

Communication. Enron [11] is an email network, capturing emails sent during the collapse of the Enron energy company.

Information. DBLP [11] is a citation network, and arXiv [13] is a co-authorship network of Astrophysicists. MovieLens [11] is bipartite graph of users rating movies for the research project MovieLens. Edges encode users and the movies that they rated.

Training Graph and Ground Truth. While using LINKWALDO in practice does not require a test set, in order to know how effective it is, we must *evaluate* it on ground truth missing links. As ground truth, we remove 20% of the edges. In the static graphs, we remove 20% at random. In the temporal graphs, we remove the 20% of edges with the most recent timestamps. If either of the nodes in the removed edge is not present in the training graph, we discard the edge from the ground-truth. The graph with these edges removed is the training graph, which LINKWALDO and the baselines observe when choosing the set of unlinked pairs to return.

Setup. We discuss in § V-D how we choose which groupings to use, and how many groups in each. Whenever used, we implement SG and CG by clustering embeddings with KMeans: xNETMF [8] and NetMF [22] (window size 1), respectively. In LSH, we set the maximum tree depth dynamically based

on the size of an equivalence class: $b_{\max} = 12$ if $|\mathcal{C}_i| < 1B$, $b_{\max} = 15$ if $|\mathcal{C}_i| < 10B$, $b_{\max} = 20$ if $|\mathcal{C}_i| < 25B$, $b_{\max} = 30$ otherwise. We set the number of trees r based on the fraction of $|\mathcal{C}_i|$ that we seek to return: $r = 5$ if $\kappa/|\mathcal{C}_i| < 0.0001$, $r = 10$ if $\kappa/|\mathcal{C}_i| < 0.001$ and $r = 25$ otherwise.

B. Recall and Precision (**RQ1**)

Task Setup. We evaluate how effectively LINKWALDO returns in \mathcal{P} the ground-truth missing links, at values of k much smaller than n^2 . We report k , chosen based on dataset size, in Tab. III, and discuss effects of the choice in the appendix. We compare the set LINKWALDO returns to those of five baselines, and evaluate both LINKWALDO-D, which uses grouping DG, and LINKWALDO-M, which uses DG, SG, and CG together. In both LINKWALDO variants, we consider the following proximities (cf. III-B) as input, and report the results that are best: LaPM using NETMF [22] embeddings (window sizes 1 and 2), and AA, the best heuristic proximity. For the bipartite MovieLens, we use BiNE [6], an embedding method designed for bipartite graphs. We report the input proximity model for each dataset in Tab. V in the appendix. We set the exact-search and bailout tolerances to $\tau = 25M$ and $\zeta = 0.5$, which we determined via a parameter study in § V-D. Results are averages over five random seeds (§ V-A): for static graphs, the randomly-removed edges are different for each seed; for temporal graphs, the latest edges are always removed, so the LSH hash functions are the main source of randomness.

Metrics. We use Recall ($R@k$), the fraction of known missing/future links that are in the size- k set returned by the method, and Precision ($P@k$), the fraction of the k pairs that are known to be missing/future links. Recall is a more important metric, since (1) the returned set of pairs \mathcal{P} does not contain final predictions, but rather pairs for a LP method to make final decisions about, and (2) our real-world graphs are inherently incomplete, and thus pairs returned that are not *known* to be missing links, could nonetheless be missing in the original dataset prior to ground-truth removal (i.e., the open-world assumption [25]). We report both in Table III.

Baselines. We use five baselines. **NMF+BAG** [5] uses non-negative matrix factorization (NMF) and a bagging ensemble to return k pairs while pruning the search space. We use their reported strongest version: the *Biased Edge Bagging* version with *Node Uptake* and *Edge Filter* optimizations (*Biased(NMF+)*). We use the authors' recommended parameters when possible: $\epsilon = 1$, $\mu = 0.1$, $f = 0.1$, $\rho = 0.75$, number of latent factors $d = 50$, and ensemble size μ/f^2 . In some cases, these suggested parameters led to fewer than k pairs being returned, in which case we tweaked the values of ϵ, μ , and f until k were returned. We report these deviations in Tab. V in the appendix. We use our own implementation.

We also use four proximity models, which we showed to be special cases of FLLM in § III-E: **LaPM** ranks pairs *globally* based on the dot product of their embeddings, and returns the top k . To avoid searching all-pairs, we use the same LSH scheme that we introduce in § IV-C for LINKWALDO. We set

$r = 25$, and like LINKWALDO, use NETMF with a window size of 1 or 2, except for MovieLens, where we use BiNE. **JS**, **CN**, and **AA** are defined in III-B. We exploit the property described in IV-C2—i.e., all these scores are zero for nodes beyond two hops. We compute the scores for all nodes within two hops, and return the top k unlinked pairs.

Results. Across the 13 datasets, LINKWALDO is the best performing method on 10, in both recall and precision. The LINKWALDO-M variant is slightly stronger than LINKWALDO-D, but the small gap between the two demonstrates that even simple node groupings can lead to strong improvements over baselines. LINKWALDO generalizes well across the diverse types of networks. In contrast, the heuristics perform well on social networks, but not as well on, e.g., metabolic networks (Yeast, HS-Protein, and Protein-Soy). Furthermore, the heuristic baselines cannot extend to bipartite graphs like MovieLens, because fundamentally, all links form between nodes more than one hop away. These observations demonstrate the value of learning from the observed links, which LINKWALDO does via resemblance. We also observe that heuristic definitions of similarity, such as AA, outperform latent embeddings (LaPM) that capture proximity. We conjecture that the embedding methods are more sensitive to the massive skew of the data, because even random vectors in high-dimensional space can end up with some level of proximity, due to the curse of dimensionality. This suggests that the standard approach of evaluating on a balanced test set may artificially inflate results.

In the three datasets where LINKWALDO does not outperform AA, it is only outperformed by a small margin. Furthermore, the four datasets with the largest total Variation distances between p_n and p_o are MovieLens, MathOverflow, Enron, and Digg. Theorem 2 suggests that LINKWALDO may incur the most error in these datasets. Indeed, these are the only three datasets where LINKWALDO fails to outperform all other methods (with MovieLens being bipartite, as discussed above). While the performance on temporal networks is strong, the higher total Variation distance suggests that the assumption that $p_o = p_n$ may sometimes be violated due to *concept drift* [3]. Thus, a promising future research direction is to use the timestamps of observed edges to predict roadmap drift over time, in order to more accurately estimate the future roadmap.

C. Scalability (**RQ2**)

Task Setup. We evaluate how LINKWALDO scales with the number of edges, and the number of nodes in a graph by running LINKWALDO with fixed parameters on all datasets. We set $k = 1M$, use NETMF (window-size of 1) as $\text{sim}(\cdot, \cdot)$, and do not perform bailout ($\zeta = 0$). All other parameters are identical to **RQ1**. We use our Python implementation on an Intel(R) Xeon(R) CPU E5-2697 v3, 2.60GHz with 1TB RAM.

Results. The results in Fig. 3 demonstrate that in practice, LINKWALDO scales linearly on the number of edges, and sub-quadratically on the number of nodes.

TABLE III: On each dataset, we highlight the cell of the top-performing method with bold text and a gray background, and the second best with bold text only. An “**” denotes statistical significance at a 0.05 p -value in a paired t-test. The “***” means that the difference between the better performing variant of LINKWALDO was also significantly better than the other variant at the same p -value. Under each dataset, we give the percentage of the quadratic search space that the value of k corresponds to (usually < 1%). On average, LINKWALDO-M is the best-performing method, and LINKWALDO-D the second best.

Dataset	Metric	NMF+BAG [5]	LaPM	JS	CN	AA	LINKWALDO-D	LINKWALDO-M
Yeast 0.39%	R@10K	0.4078 ± 0.01	0.4400 ± 0.01	0.4766 ± 0.01	0.6142 ± 0.01	0.6590 ± 0.01	0.6762 ± 0.01	0.6926** ± 0.01
	P@10K	0.0898 ± 0.00	0.0969 ± 0.00	0.1049 ± 0.00	0.1352 ± 0.00	0.1451 ± 0.00	0.1489 ± 0.00	0.1525** ± 0.00
DBLP 0.15%	R@100K	0.2319 ± 0.00	0.0927 ± 0.00	0.0389 ± 0.00	0.3379 ± 0.00	0.3775 ± 0.00	0.4270 ± 0.00	0.4271* ± 0.00
	P@100K	0.0204 ± 0.00	0.0082 ± 0.00	0.0034 ± 0.00	0.0298 ± 0.00	0.0332 ± 0.00	0.0376* ± 0.00	0.0376* ± 0.00
Facebook1 1.34%	R@100K	0.4036 ± 0.02	0.8005 ± 0.00	0.8244 ± 0.00	0.8547 ± 0.00	0.8863 ± 0.00	0.8975 ± 0.00	0.9059*** ± 0.00
	P@100K	0.0711 ± 0.00	0.1410 ± 0.00	0.1453 ± 0.00	0.1506 ± 0.00	0.1562 ± 0.00	0.1581 ± 0.00	0.1596*** ± 0.00
MovieLens 3.57%	R@100K	0.1221 ± 0.02	0.2096 ± 0.01	0.0000 ± 0.00	0.0000 ± 0.00	0.0000 ± 0.00	0.1667 ± 0.01	0.3662** ± 0.01
	P@100K	0.0035 ± 0.00	0.0060 ± 0.00	0.0000 ± 0.00	0.0000 ± 0.00	0.0000 ± 0.00	0.0048 ± 0.00	0.0105** ± 0.00
HS-Protein 0.52%	R@100K	0.5127 ± 0.02	0.4033 ± 0.01	0.4768 ± 0.00	0.7998 ± 0.00	0.8429 ± 0.00	0.8878 ± 0.00	0.9038*** ± 0.00
	P@100K	0.1506 ± 0.00	0.1185 ± 0.00	0.1400 ± 0.00	0.2349 ± 0.00	0.2476 ± 0.00	0.2608 ± 0.00	0.2655** ± 0.00
arXiv 0.06%	R@100K	0.2877 ± 0.00	0.2584 ± 0.00	0.5149 ± 0.00	0.5576 ± 0.00	0.6539 ± 0.00	0.7004 ± 0.00	0.7032** ± 0.00
	P@100K	0.2017 ± 0.00	0.1812 ± 0.00	0.3610 ± 0.00	0.3909 ± 0.00	0.4585 ± 0.00	0.4911 ± 0.00	0.4930*** ± 0.00
MathOverflow 0.55%	R@1M	0.3901 ± 0.00	0.0279 ± 0.00	0.0084 ± 0.00	0.4201 ± 0.00	0.4333* ± 0.00	0.4227 ± 0.00	0.4233 ± 0.00
	P@1M	0.0070 ± 0.00	0.0005 ± 0.00	0.0001 ± 0.00	0.0075 ± 0.00	0.0078* ± 0.00	0.0076 ± 0.00	0.0076 ± 0.00
Enron 0.04%	R@1M	0.2551 ± 0.00	0.0008 ± 0.00	0.0004 ± 0.00	0.3071 ± 0.00	0.3209* ± 0.00	0.3166 ± 0.00	0.3080 ± 0.00
	P@1M	0.0089 ± 0.00	0.0000 ± 0.00	0.0000 ± 0.00	0.0107 ± 0.00	0.0112* ± 0.00	0.0111 ± 0.00	0.0108 ± 0.00
Reddit 0.06%	R@1M	0.3165 ± 0.00	0.0015 ± 0.00	0.0006 ± 0.00	0.3796 ± 0.00	0.4038 ± 0.00	0.4096 ± 0.00	0.4186** ± 0.00
	P@1M	0.0130 ± 0.00	0.0001 ± 0.00	0.0000 ± 0.00	0.0156 ± 0.00	0.0166 ± 0.00	0.0168 ± 0.00	0.0172** ± 0.00
Epinions 0.04%	R@1M	0.3312 ± 0.00	0.0371 ± 0.00	0.0383 ± 0.00	0.3871 ± 0.00	0.4291 ± 0.00	0.4292 ± 0.00	0.4323** ± 0.00
	P@1M	0.0242 ± 0.00	0.0027 ± 0.00	0.0028 ± 0.00	0.0283 ± 0.00	0.0313 ± 0.00	0.0313 ± 0.00	0.0316** ± 0.00
Facebook2 0.06%	R@1M	0.0948 ± 0.00	0.1947 ± 0.00	0.3605 ± 0.00	0.2439 ± 0.00	0.2832 ± 0.00	0.3796** ± 0.00	0.3776 ± 0.00
	P@1M	0.0145 ± 0.00	0.0298 ± 0.00	0.0551 ± 0.00	0.0373 ± 0.00	0.0433 ± 0.00	0.0580** ± 0.00	0.0577 ± 0.00
Digg 0.04%	R@10M	0.2459 ± 0.00	0.0035 ± 0.00	0.0015 ± 0.00	0.2952 ± 0.00	0.3066* ± 0.00	0.3053 ± 0.00	0.3052 ± 0.00
	P@10M	0.0032 ± 0.00	0.0000 ± 0.00	0.0000 ± 0.00	0.0038 ± 0.00	0.0040* ± 0.00	0.0040* ± 0.00	0.0040* ± 0.00
Protein-Soy 0.99%	R@10M	0.3624 ± 0.02	0.1225 ± 0.00	0.2792 ± 0.00	0.3573 ± 0.00	0.3636 ± 0.00	0.5781 ± 0.00	0.6016* ± 0.03
	P@10M	0.2178 ± 0.01	0.0736 ± 0.00	0.1678 ± 0.00	0.2147 ± 0.00	0.2185 ± 0.00	0.3473 ± 0.00	0.3615* ± 0.02
Avg R		0.3048	0.1994	0.2323	0.4273	0.4585	0.5074	0.5281
Avg P		0.0635	0.0506	0.0754	0.0969	0.1056	0.1213	0.1238

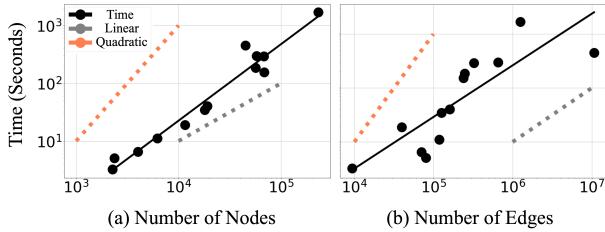


Fig. 3: LINKWALDO is sub-quadratic on the number of nodes (a) and linear on the number of edges (b).

D. Parameters (RQ3)

Setup. We evaluate the quality of different groupings (§ IV-A), and how the number of groups in each affects performance. On four graphs, Yeast, arXiv, Reddit, and Epinions, we run LINKWALDO with groupings DG, SG, and CG, varying the number of groups $|\Gamma| \in \{5, 10, 25, 50, 75, 100\}$. We also investigate pairs of groupings, and the combination of all three groupings, via grid search of the number of groupings in each. We also evaluated $\tau \in \{1M, 10M, 25M, 50M\}$, the tolerance for searching equivalence classes exactly vs. approximately with LSH, and $\zeta \in \{0, 0.1, 0.25, 1/3, 0.5, 2/3\}$, the fraction of training pairs we allow the proximity function to miss before we bailout of an equivalence class.

Results. The results for the individual groupings are shown in

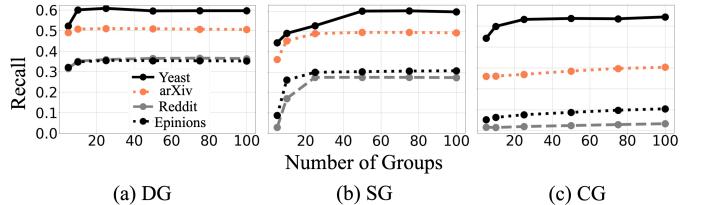


Fig. 4: Number of groups for groupings DG, SG, and CG

Fig. 4. Grouping by log-binning nodes based on their degree, (i.e., DG) is in general the strongest grouping. Across all three groupings, we find that $|\Gamma| = 25$ is a good number of groups. We found that using all three groupings was the best combination, with 25 log-bins, 5 structural clusters, and 5 communities (we omit the figures for brevity). For individual groupings, we observe diminishing returns, and in multiple groupings, slightly diminished performance when the number of groups in each grows large. We omit the figures for τ and ζ , but found that $\tau = 25M$ and $\zeta = 0.5$ were the best parameters.

VI. CONCLUSION

In this paper, we focus on the under-studied and challenging problem of identifying a moderately-sized set of node pairs for a link prediction method to make decisions about. We mitigate the vastness of the search-space, filled with mostly non-links,

by considering not just proximity, but also how much a pair of nodes *resembles* observed links. We formalize this idea in the Future Link Location Model, show its theoretical connections to stochastic block models and proximity models, and introduce an algorithm, **LINKWALDO**, that leverages it to return high-recall candidate sets, with only a tiny fraction of all pairs. Via our resemblance insight, **LINKWALDO**'s strong performance generalizes from social networks to protein networks. Future directions include investigating the directionality of links, since the roadmap can incorporate this information, and extending to heterogeneous graphs with many edge and node types, like knowledge graphs.

ACKNOWLEDGEMENTS

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

REFERENCES

- [1] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.
- [2] Mayank Bawa, Tyson Condie, and Prasanna Ganesan. Lsh forest: self-tuning indexes for similarity search. In *WWW*, pages 651–660, 2005.
- [3] Caleb Belth, Xinyi Zheng, and Danai Koutra. Mining persistent activity in continually evolving networks. In *KDD*, 2020.
- [4] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002.
- [5] Liang Duan, Shuai Ma, Charu Aggarwal, Tiejun Ma, and Jinpeng Huai. An ensemble approach to link prediction. *IEEE TKDE*, 29(11), 2017.
- [6] Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. Bine: Bipartite network embedding. In *SIGIR*, 2018.
- [7] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, 40(3):52–74, 2017.
- [8] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. REGAL: Representation learning-based graph alignment. In *CIKM*, 2018.
- [9] Unmesh Joshi and Jacopo Urbani. Searching for embeddings in a haystack: Link prediction on knowledge graphs with subgraph pruning. In *WebConf*, 2020.
- [10] Thomas N Kipf and Max Welling. Variational graph auto-encoders. In *NIPS Workshop on Bayesian Deep Learning*, 2016.
- [11] Jérôme Kunegis. Konect: the koblenz network collection. In *WWW*, 2013.
- [12] Pierre Latouche, Etienne Birmelé, Christophe Ambroise, et al. Overlapping stochastic block models with application to the french political blogosphere. *Annals App. Stat.*, 5(1):309–336, 2011.
- [13] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. <http://snap.stanford.edu/data>, June 2014.
- [14] David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.
- [15] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *AS/IS&T*, 58(7):1019–1031, 2007.
- [16] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *CSUR*, 49(4):1–33, 2016.
- [17] Nikhil Mehta, Lawrence Carin, and Piyush Rai. Stochastic blockmodels meet graph neural networks. In *ICML*, 2019.
- [18] Kurt Miller, Michael I Jordan, and Thomas L Griffiths. Nonparametric latent feature models for link prediction. In *NeurIPS*, 2009.
- [19] Mark EJ Newman. Mixing patterns in networks. *Phys. Rev. E*, 67(2), 2003.
- [20] Krzysztof Nowicki and Tom A B Snijders. Estimation and prediction for stochastic blockstructures. *AS/IS&T*, 96(455):1077–1087, 2001.
- [21] Benjamin Pachev and Benjamin Webb. Fast link prediction for large networks using spectral embedding. *J. Complex Netw.*, 6(1):79–94, 2018.
- [22] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*, pages 459–467, 2018.
- [23] Ryan Rossi and Nesreen Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.
- [24] Ryan A. Rossi, Di Jin, Sungchul Kim, Nesreen Ahmed, Danai Koutra, and John Boaz Lee. On proximity and structural role-based embeddings in networks: Misconceptions, techniques, and applications. *TKDD*, 2020.
- [25] Tara Safavi, Danai Koutra, and Edgar Meij. Evaluating the calibration of knowledge graph embeddings for trustworthy link prediction. In *EMNLP*, 2020.
- [26] Dongjin Song, David A Meyer, and Dacheng Tao. Top-k link recommendation in social networks. In *ICDM*, pages 389–398. IEEE, 2015.
- [27] Alexandre B Tsybakov. *Introduction to nonparametric estimation*. Springer Science & Business Media, 2008.
- [28] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014.
- [29] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *NeurIPS*, pages 5165–5175, 2018.

APPENDIX

A. Effect of k

Table IV gives results (avgs over 3 seeds) for multiple values of k . For reasonable values of k —roughly up to an order of magnitude greater than m —results are mostly stable. The main exception is for small k , where NMF+BAG performs well in some cases. This is consistent with its design: to return a small, accurate set of top- k predictions, rather than a candidate set.

B. Complexity Analysis

Let γ be the time complexity of the node grouping (**S1**). Computing the expected number of new edges in each cell (and variance), directly from the observed links, (**S2**) is $O(m)$. The complexity of searching equivalence classes (**S3**) comes from hashing each node in the decomposition $O(b_{\max})$ times, and finding the κ_i closest pairs in the $O(\kappa_i)$ pairs that land in the same bucket: $\sum_{C_i \in \Pi} O(|\mathcal{V}_u \cup \mathcal{V}_v| b_{\max} + \kappa_i) = O(nb_{\max} + k)$. This assumes that we do not encounter unrealistic scenarios, e.g., the embeddings being equivalent and hence inseparable, and that b_{\max} is set large enough that the volume of tree leaves is not asymptotically larger than $O(\kappa_i)$. Adding from the global pool (**S4**) takes $O(k)$ time, since $|\tilde{\mathcal{P}}_G| = O(k)$ and can be maintained in sorted order (similar to the merge in merge sort). Thus, the total time complexity is $O(\gamma + m + nb_{\max} + k)$.

TABLE IV: We report “ $< k$ ” if fewer than k pairs are returned. Black cells indicate values of k outside the scale of the dataset.

Metric	HS-Protein			Facebook2		
	NMF+BAG	AA	LINKWALDO-M	NMF+BAG	AA	LINKWALDO-M
R@10K	0.2629	0.1909	0.2251	0.0093	0.0103	0.0172
R@100K	$< k$	0.8443	0.9035	0.0422	0.0672	0.1106
R@1M	$< k$	0.9747	0.9847	0.0970	0.2832	0.3781
R@5M				$< k$	0.5561	0.5762

TABLE V: Input Proximity Model for LaPM and LINKWALDO, and parameter deviations from default for NMF+BAG.

Graph	LaPM	LINKWALDO-D	LINKWALDO-M	NMF+BAG
Yeast	NETMF-2	NETMF-2	NETMF-2	$\epsilon = 0.5$
DBLP	NETMF-2	NETMF-2	NETMF-2	Default
Facebook1	NETMF-1	AA	AA	$\epsilon = 0.1$
MovieLens	BiNE	BiNE	BiNE	$\epsilon = 0.05$
HS-Protein	NETMF-2	NETMF-2	NETMF-2	$\epsilon = 0.75$
arXiv	NETMF-2	AA	AA	Default
MathOverflow	NETMF-2	NETMF-1	NETMF-1	$\epsilon, \mu, f = 0.5, 0.3, 0.3$
Enron	NETMF-2	NETMF-1	AA	Default
Reddit	NETMF-2	AA	AA	Default
Epinions	NETMF-1	AA	AA	Default
Facebook2	NETMF-2	AA	AA	Default
Digg	NETMF-2	NETMF-1	NETMF-1	Default
Protein-Soy	NETMF-1	NETMF-2	NETMF-2	Default