Supplementary Material

A Additional Related Work

In this Appendix, we provide an overview of some additional related works, which complement the discussion in Section 2. Specifically, we discuss some more existing formulations and methods for solving the problem of aligning pairs of graphs. We also describe some existing measures of node similarity in more detail, as this concept is relevant to our work.

Pairwise Graph Alignment. One of the most common formulations for aligning two graphs G_1 and G_2, with adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, is to find a matrix $\mathbf{P}$ that permutes the nodes in $\mathbf{A}_1$ so that it best matches $\mathbf{A}_2$: $\min_{\mathbf{P}} ||\mathbf{P}\mathbf{A}_1\mathbf{P}^T - \mathbf{A}_2||_F^2$, where $||\bullet||_F$ is the Frobenius norm of the corresponding matrix. The original requirement is that $\mathbf{P}$ is a permutation matrix (i.e., a square binary matrix with exactly one entry 1 per row and column, and 0s elsewhere), but several convex or concave relaxations to this problem have been proposed, such as $\mathbf{P}$ being a (doubly) stochastic matrix [29].

The Hungarian method [28] is a well-known spectral method for solving this formulation, but it requires graphs of the same size n and is computationally expensive, $O(n^3)$, and thus inapplicable to large-scale networks. Other suggested solutions vary: clustering algorithms [26], graph edit distance [27], iterative HITS-inspired [20], and more. Like the Hungarian method, however, most of these methods align pairs of graphs and do not necessarily scale well to large-scale graph alignment; in general, they are designed for more limited uses cases than HASHALIGN.

Node Similarity. There are many different ways of computing similarity scores between nodes, most of which operate on a single network (e.g., Personalized Random Walks [23], SimRank [24], and Belief Propagation [25]). Across two graphs, [15] and [4] use (weighted) degree-based node similarities to initialize the alignment matrix. Zager and Verghese [20] compute the similarity between two graphs via a recursive method that couples the similarity scores of nodes and edges. Unlike our work, the alignment in [20] is found *after* computing the similarity between *all* pairs of nodes, as well as *all* pairs of edges.

B Euclidean Distance as an LSH family for HASHALIGN

In our main method (Sec.3.3), we introduce an LSH family based on cosine similarity. Here we also introduce EDHash-2G, which is based on Euclidean distance (ED) as the measure of (dis)similarity and is preferred when the dimensionality of the node vectors d is quite high. However, in our experiments, we report results using SimHash-2G since it ties or outperforms EDHash-2G (Sec. E).

At a high level, the corresponding hash functions compute the projections of the input vectors on random lines. More formally, given a real number w, the k^{th} hash function is constructed so that it maps a real-valued vector in $\mathbb{R}^d$ into an integer (which corresponds to a 'segment' ID):

$$h_{\mathbf{r}_k, b_k}(u) = \lfloor \frac{\mathbf{f}_{G_p}(u) \cdot \mathbf{r}_k + \beta_k}{w} \rfloor,$$

where $\mathbf{r}_k$ is the corresponding column-vector in $\mathbb{R}^d$ with entries chosen independently from a Gaussian distribution (which is 2-stable [7]) and β_k is positive real number chosen uniformly from $[0, w]$. If the nodes $u \in \mathcal{V}_1$ and $v \in \mathcal{V}_2$ map to the same w-long 'segment', they are considered similar, and their similarity is computed basd on the SKD-construction. The ED between two nodes is then proportional to the probability of them falling into the same 'segment'. Among the ways to convert a distance to similarity, we choose $sim = (1 + ED)^{-1}$, which gives bounded similarity scores in $[0, 1]$.

C Computational Complexity of HASHALIGN

In this section, we provide some more detail on the computational complexity of our proposed framework, HASHALIGN. The analysis is divided by step: **(1)** During the **node representation** phase, HASHALIGN extracts d-dimensional structural and node/edge attribute features in each of the l graphs. The complexity of this phase depends on the selected features (e.g., for G_p, the degree computation takes $O(m_p)$, while the betweenness centrality $O(n_p \cdot m_p)$). **(2)** The cost of finding the **center graph** from the l graphs to align is $O(l^2 \cdot d)$. The remaining steps can be parallelized per graph, so we report complexity analysis in terms of the size of the largest graph. **(3)** The hashing-based similarity computation consists of two parts: For K hashing functions and bucket size $|b_i|$, the complexity of **hashing** is $O(K \cdot \max_p n_p \cdot d)$ for graph G_p with n_p nodes. **(4)** For node **matching**, the similarity matrix must be constructed by computing the σ_{tot} scores over pairs of nodes in the same bucket in time $O(\max_i |b_i|^2)$. Then the greedy approach finds the maximum element for each of the n_p nodes in at most $O(n_p^2)$ time. In the case of multiple network alignment, we replace the expensive computation of pairwise similarity score and simply perform sparse matching matrix multiplications (with the center graph).

D Choice of Node Features for HASHALIGN

Since we believe that the inconsistency between graphs could be considered noise, the more stable the features are in the presence of noise, the better. Thus, we choose our node features after seeing that their distributions have little difference before and after adding noise to the graph. An example of a feature that is robust and one that is sensitive to noise is given in Fig. 6.

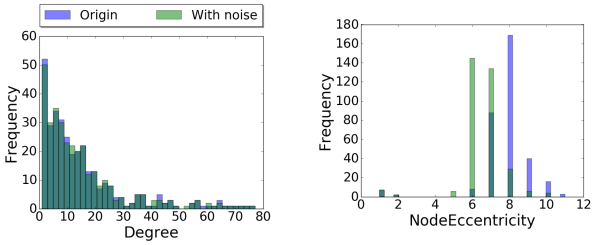
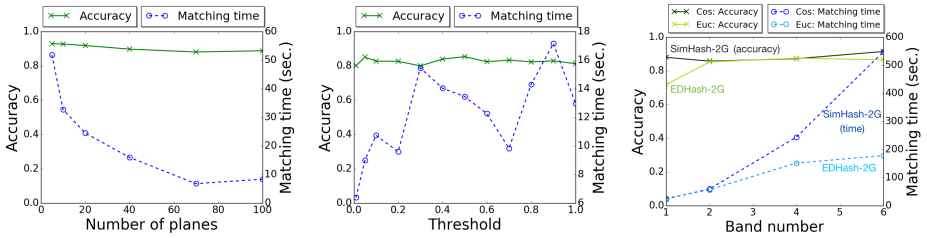


Fig. 6: Degree is robust, but eccentricity is sensitive to noise.

E HASHALIGN: Sensitivity Analysis

We provide additional analysis in this section to find out the patterns behind the various combinations of parameters in the synthetic graphs and LSH functions. The tuning



(a) Number of random hyperplanes K in SimHash-2G.

(b) Threshold α used in computing similar pairs.

(c) Efficiency with respect to the number of bands Z.

Fig. 7: Sensitivity analysis of HASHALIGN to parameters.

parameters for SimHash-2G / EDHash-2G are: the number of hash functions (or hyperplanes) K, the threshold α for computing the similarity of node pairs, and the number of bands Z. In Fig. 7, we present the results of a series of comprehensive experiments which vary these parameters. Based on these, we have the following observations:

- As shown in Fig. 7b, a higher threshold α leads to similar accuracy but potentially longer runtime. Computing pairwise similarity between 10 and 20% of the total number of nodes gives the best trade-off between runtime and accuracy.
- The number of node attributes or the length of feature vectors determines the performance of LSH. Given more features, LSH will generate more buckets, which decreases the runtime of filtering colliding nodes pairs and increases accuracy.
- In terms of choice of different types of LSHs, SimHash-2G (based on cosine similarity) tends to yield better randomness than EDHash-2G, while keeping runtime low. This is why we chose it for our other experiments. In Fig. 7c, we used segments of length $w = 4$ for EDHash-2G.
- The fewer bands, the better: more bands lead to roughly quadratic growth in runtime, since (1) there are more buckets from different bands to merge, and (2) there are fewer features in each band, making it easier for two nodes to collide (Fig. 7c).

References

23. T. H. Haveliwala. Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. *TKDE*, 15(4):784–796, 2003.
24. G. Jeh and J. Widom. SimRank: A Measure of Structural-Context Similarity. In *KDD*, pages 538–543. ACM, 2002.
25. D. Koutra, T.-Y. Ke, U. Kang, D. H. Chau, H.-K. K. Pao, and C. Faloutsos. Unifying Guilt-by-Association Approaches: Theorems and Fast Algorithms. In *ECML PKDD*, pages 245–260, 2011.
26. H. Qiu and E. R. Hancock. Graph Matching and Clustering Using Spectral Partitions. *IEEE TPAMI*, 39(1):22–34, 2006.
27. K. Riesen and H. Bunke. Approximate Graph Edit Distance Computation by Means of Bipartite Graph Matching. *Image and Vision Computing*, 27(7):950 – 959, 2009.
28. S. Umeyama. An Eigendecomposition Approach to Weighted Graph Matching Problems. *IEEE TPAMI*, 10(5):695–703, 1988.
29. M. Zaslavskiy, F. Bach, and J.-P. Vert. A Path Following Algorithm for the Graph Matching Problem. *IEEE TPAMI*, 31(12):2227–2242, Dec. 2009.