ELRUHNA: Elimination Rule-based Hypergraph Alignment

Cameron Ibrahim[1], S M Ferdous[2] Ilya Safro[1], Marco Minutoli[2], and Mahantesh Halappanavar[2]

[1] University of Delaware, Newark, DE 19716, USA,
{cibrahim,isafro}@udel.edu,
[2] Pacific Northwest National Laboratory, Richland WA 99354, USA,
{sm.ferdous, marco.minutoli, hala}@pnnl.gov

Abstract. Hypergraph alignment is a well-known NP-hard problem with numerous practical applications across domains such as bioinformatics, social network analysis, and computer vision. Despite its computational complexity, practical and scalable solutions are urgently needed to enable pattern discovery and entity correspondence in high-order relational data. The problem remains understudied in contrast to its graph based counterpart. In this paper, we propose ELRUHNA, an elimination rule-based framework for unsupervised hypergraph alignment that operates on the bipartite representation of hypergraphs. We introduce the incidence alignment formulation, a binary quadratic optimization approach that jointly aligns vertices and hyperedges. ELRUHNA employs a novel similarity propagation scheme using local matching and cooling rules, supported by an initialization strategy based on generalized eigenvector centrality for incidence matrices. Through extensive experiments on real-world datasets, we demonstrate that ELRUHNA achieves higher alignment accuracy compared to state-of-the-art algorithms, while scaling effectively to large hypergraphs.
Reproducibility: Code and data are available at https://github.com/cameton/ASONAM_ELRUHNA.

1 Introduction

Given hypergraphs $H_Q \coloneqq (V_Q, E_Q)$ and $H_D \coloneqq (V_D, E_D)$ known as the query hypergraph and data hypergraph respectively, the *hypergraph alignment* problem aims to find an injection $\sigma \colon V_Q \hookrightarrow V_D$ maximizing some measure of overlap between H_Q and H_D. Typical applications of the hypergraph alignment include finding patterns common between a pair of hypergraphs, or identifying vertices which represent the same entity in the query and data hypergraphs. This problem extends the *graph alignment* with the goal to capture higher-order relationships between nodes. Hypergraph alignment is of considerable interest in fields such as machine learning [9], bioinformatics [1,18], computer vision [14], and natural language processing [16].

For example, in machine learning and computer vision, the hypergraph alignment methods are commonly used to perform feature matching [9]. Given two

datasets, we attempt to represent a set of features as the vertices of a hypergraph. Hypergraph alignment can then be used as a heuristic method for identifying features which are common between the two datasets. Another common framing of this problem in machine learning is to attempt to identify user correspondence across social networks[22]. In this setting, these social networks are represented as hypergraphs with users as vertices, groups of friends/co-authors/coworkers/etc as hyperedges. As a result of privacy concerns, methods in this area may consist of supervised and unsupervised approaches [9].

In this paper, we focus on finding global solutions to the hypergraph alignment problem in an unsupervised setting. In this context, the goal is to find a mapping σ which maximizes the hyperedge correctness metric. In short, σ is chosen to maximize the number of hyperedges e in the query hypergraph such that $\{\sigma(v) \mid v \in e\}$ is also an hyperedge in the data hypergraph [14]. Maximizing hyperedge correctness for general (hyper)graphs is NP-hard [14].

Our Contribution: We introduce a number of variants on the hyperedge correctness metric for hypergraph alignment which we refer to collectively as *incidence alignment* (Eq. 3) and discuss the benefits of this formulation. We provide and efficient heuristic for maximizing incidence alignment (Algorithm 1) and introduce a novel hypergraph alignment solver ELRUHNA that exhibits superior quality-runtime trade-off. We compare the results obtained by ELRUHNA against state of the art methods, and demonstrate an improvement in quality of up to 25%, and solve problem instances with tens of thousands of vertices in the bipartite representation.

2 Background & Notation

An undirected graph $G = (V, E)$ consists of a vertex set V and an edge set $E \subseteq \{e \in 2^V \mid |e| = 2\}$. The higher order generalization of an undirected graph is a hypergraph $H = (V, E)$, whose set of hyperedges is a subset $E \subseteq 2^V$. The rank of a hypergraph is the maximum size of any of its edges. A vertex v is incident to a hyperedge e if $v \in e$. Two vertices v, u are adjacent if there exists a hyperedge e which is incident to both u and v. A graph $G = (V, E)$ is bipartite if $V = V_1 \cup V_2$ such that $V_1 \cap V_2 = \emptyset$ and $uv \in E$ implies $u \in V_1, v \in V_2$ or vice versa. V_1 and V_2 may be referred to as the left and right parts respectively. The bipartite representation of a hypergraph $H = (V, E)$ is a bipartite graph $G' = (V \cup E, E')$ such that the left and right parts are sets V and E, respectively, and a vertex $v \in V$ is adjacent to a vertex that represents a hyperedge $e \in E$ in G' if and only if $v \in e$ in H, i.e., there is an edge $ve \in E'$. The clique expansion of a hypergraph $H = (V, E)$ is an undirected graph $G' \coloneqq (V, E')$ defined so that u and v are adjacent in G' if and only if u and v are adjacent in H, i.e., each hyperedge in a hypergraph is represented as a clique (that contains all its nodes) in the graph.

Throughout this paper, we make use of the indicator function $\mathbb{1}[\phi] \coloneqq 1$ if ϕ is true and 0 otherwise. For a hypergraph $H = (V, E)$ with $n \coloneqq |V|$ vertices and $m \coloneqq |E|$ hyperedges, we can define an $n \times n$ matrix A and an $n \times m$ matrix

B as $A_{uv} \coloneqq \mathbb{1}[uv \in E]$ and $B_{ue} \coloneqq \mathbb{1}[u \in e,\ e \in E]$. A and B are known as the adjacency and incidence matrices of H, respectively. Notably, A is also the adjacency matrix of the clique expansion of H, and so elides any higher order information. Weighted variants $\tilde{A}, \tilde{B}$ of these matrices may be created by replacing any 1 with a positive real number.

2.1 Hypergraph Alignment

Broadly, hypergraph alignment aims to map a given query hypergraph $H_Q \coloneqq (V_Q, E_Q)$, onto a known data hypergraph $H_D \coloneqq (V_D, E_D)$. To do so, we attempt to find an injective function $\sigma \colon V_Q \hookrightarrow V_D$ which maximizes some reward function R

$$\max_{\sigma \colon V_D \hookrightarrow V_Q} R(H_D, H_Q, \sigma).$$

If the ground truth σ^* is known for some subset $S \subseteq V_Q$, we may attempt to maximize accuracy $R_{acc}(\sigma) = \frac{1}{|S|} \sum_{v \in S} \mathbb{1}\left[\sigma(v) = \sigma^*(v)\right]$. In this paper, we focus on the unsupervised case where the ground truth σ^* is not known during the optimization process [9,8].

A common metric used in both hypergraph and graph alignments literature is the hyperedge correctness

$$R_{EC}(\sigma) = \frac{1}{|E_Q|} \sum_{e \in E_Q} \mathbb{1}\left[\{\sigma(v) \mid v \in e\} \in E_D\right].$$

This gives the fraction of hyperedges in the query hypergraph which are present in the data hypergraph after mapping.

A mapping $\sigma \colon V_Q \hookrightarrow V_D$ can alternatively be represented by an $|V_Q| \times |V_D|$ binary matrix X whose rows and columns all sum to at most one [5]. This can be expressed succinctly in terms of the $\|X\|_1$ and $\|X\|_\infty$ norms, defined as $\|X\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |X_{ij}|$ and $\|X\|_\infty = \max_{1 \le i \le m} \sum_{j=1}^{n} |X_{ij}|$. respectively. As such, hyperedge correctness can also be generalized as the following binary optimization problem. Let $\mathsf{S}^{(k)}$ be a k dimensional tensor defined as

$$\mathsf{S}^{(k)}_{(v_1,u_1),\ldots,(v_k,u_k)} = \begin{cases} 1 & \{v_1, \ldots, v_k\} \in E_D \wedge \{u_1, \ldots, u_k\} \in E_Q, \\ 0 & \text{otherwise.} \end{cases}$$

If K is the rank of the hypergraph, the hypergraph alignment problem is then generalized by

$$\begin{aligned} \max_{X} \quad & \langle W, X \rangle_F + \beta \sum_{k}^{K} \mathsf{S}^{(k)} \mathrm{vec}(X)^{\otimes k} \\ \text{st} \quad & \|X\|_1, \|X\|_\infty \le 1 \\ & X \odot \mathit{\Pi} = X \\ & X \in \{0,1\}^{|V_Q| \times |V_D|}, \end{aligned}$$

where $\langle W, X \rangle_F = \text{Tr}[W^T X]$ is the Frobenius inner product and $X \odot \Pi = X$ is a constraint which can prohibit certain pairs from being matched. Setting $W = 0$ and $\beta = \frac{1}{|E_Q|}$ gives a result which maximizes R_{EC}.

For a general hypergraph with $K > 2$, this is an example of the NP-hard polynomial integer programming problem [10]. Using the equivalence $\mathsf{S}^{(2)} = A_Q \otimes A_D$ where $\otimes$ is the Kronecker product, hypergraph alignment with $K = 2$ is known as graph alignment and is an example of integer quadratic programming, for which there are a variety of successful heuristics [23,5,25].

$$\begin{aligned} \max_X \quad & \langle W, X \rangle_F + \frac{\beta}{2} \langle X A_Q, A_D X \rangle_F \\ \text{st} \quad & \|X\|_1, \|X\|_\infty \leq 1 \\ & X \odot \Pi = X \\ & X \in \{0,1\}^{|V_Q| \times |V_D|} \end{aligned} \tag{1}$$

Furthermore, when the data and query graph have an equal number of vertices there exists a constant C not dependent on X such that the quadratic portion of the objective is equivalent to $\langle X A_Q, A_D X \rangle_F + C = \frac{1}{2} \|X^T A_Q X - A_D\|_F^2$ which is a common way of framing the edge correctness objective for graph alignment [12].

2.2 Related Work

Due to its range of applications, graph alignment has been subject to a broad array of approaches. While graph alignment has been extensively studied, hypergraph alignment remains relatively underexplored. In this section, we will provide an overview of relevant research in both problems. Further information regarding these methods can be found in [25].

A variant of graph alignment for bipartite graphs has also been discussed in [12] with the algorithm BiG-Align. In this context, both G_Q and G_D are bipartite, and the aim is to match the left (resp. right) part of G_Q to the left (resp. right) part of G_D. To do so, two matching matrices X, Y are found to maximize $\|X^T B_Q Y - B_D\|_F^2$. In [9], the authors have made use of this formulation using the equivalency of hypergraphs to bipartite graphs in order to create a heuristic for hypergraph alignment. We utilize a similar formulation in our method.

One may attempt to solve a graph or hypergraph alignment problem by directly trying to solve the integer programming formulations; however, because of the computational cost of solving these problems with conventional solvers, a number of approximate methods have arisen. Big-Align is one of several methods which attempt to solve the graph alignment problem by relaxing the integer constraint on the optimization problem. Doing so produces a matrix which may be interpreted as a set of probabilities indicating how likely it is to match pair of vertices. Approximate matching algorithms such as those described in [23,11] may then be used in order to round these to a solution. In NetAlign, a belief propogation algorithm is applied to find an approximate solution to the integer

quadratic programming problem of graph alignment [4]. In Adaptive Discrete Hypergraph Mathching, the integer polynomial programming representation of the hypergraph matching problem is solved via iterative linear approximations [24].

Of particular importance to this paper is the ELRUNA algorithm [20]. ELRUNA is an algorithm for graph alignment based purely on the topology of the input graphs. In ELRUNA, a set of initial similarities between vertices in the query and data graphs are iteratively smoothed according to a number of elimination rules. These rules govern which similarities are allowed to propagate through the graph by tying each update to the result of a local matching problem in the region of a pair. An approximation algorithm is then again used in the matrix of similarities in order to achieve a result.

Node embeddings are commonly used in (hyper)graph alignment to determine similarities between pairs of nodes and determine candidate pairs for the final alignment. The Cone-align algorithm embeds each graph separately, the attempts to align each embedding space in order to find a unified embedding for both graphs [8]. This technique is extended by the cuAlign algorithm, which includes sparsification of candidate pairs to reduce computational complexity, then solves the resulting sparsified problem using a combination of belief propagation and an approximate weighted matching solver [23].

Node embeddings have also been seen in a hypergraph context with approaches such as HyperAlign [9] and HNN-HM [15]. In the former, features for each node are extracted from each hypergraph, then the resulting data is embedded using contrastive learning. The node emeddings are then used to determine similarities between nodes, which are utilized to sparsify the inputs as in cuAlign. In HNN-HM, the data and query hypergraphs are combined into a single association hypergraph, which is then embedded as a single object before being decoded and classified.

3 Algorithm ELRUHNA: Elimination Rule-based Hypergraph Network Alignment

In this section, we will introduce two variations of the hypergraph alignment optimization problem which we refer to as *incidence alignment*. The aim of incidence alignment is not only to align the vertices of the data and query hypergraphs, but to align the hyperedges as well in order to maintain the incidence of the aligned hypergraphs as much as possible.

After this, we will introduce the Elimination Rule-based Hypergraph Network Alignment algorithm, or ELRUHNA. The full algorithm is presented in Algorithm 1. ELRUHNA utilizes an iterative strategy to propagate similarities throughout the hypergraph using a pair of elimination rules described in Sect. 3.3. Strategies for determining initial similarities are discussed in Sect. 3.2.

Algorithm 1: Elimination Rule Based Hypergraph Alignment

Input: Incidence matrices B_Q, B_D with sizes n_Q, m_Q and n_D, m_D
Output: Matrices X, Y representing σ_V, σ_E
$X, Y \leftarrow \texttt{zeros}(n_Q, n_D), \texttt{zeros}(m_Q, m_D)$;
while *has_unmatched*(X, Y) **do**
  $X^*, Y^* \leftarrow \texttt{matched}(X, Y)$;
  $W_V, W_E \leftarrow \tilde{B}_Q^T Y^* \tilde{B}_D, \tilde{B}_Q X^* \tilde{B}_D^T$, Equation 5;
  $B'_Q, B'_D, W'_V, W'_E \leftarrow \texttt{get\_unmatched}(B_Q, B_D, W_V, W_E, X^*, Y^*)$;
  $X \leftarrow \texttt{initialize\_similarity}(\tilde{B}'_Q, \tilde{B}'_D, W'_V, W'_E)$, Sect. 3.2;
  $Y \leftarrow W'_E + \tilde{B}'_Q X' (\tilde{B}'_D)^T$;
  Propagate similarities, Algorithm 2;
  $Y' \leftarrow \texttt{dominant\_match}(Y')$;
  $X' \leftarrow \texttt{dominant\_match}(W_V + (\tilde{B}'_Q)^T Y' \tilde{B}'_D)$;
  $X, Y \leftarrow \texttt{update}(X, Y, X', Y')$;
end

3.1 Incidence Alignment

In practical contexts, the goal of hypergraph alignment is often to try to identify patterns shared across the data and query instances, H_Q and H_D in order to identify vertices representing entities present in both hypergraphs. In real world scenarios, these patterns may be prone to noise: for example, an individual may be a member of a friend group in a social network represented by H_Q, but not in the network represented by H_D. As a result, if one were to consider the hyperedge e representing this friend group, it would be impossible to find a σ such that mapping e would contribute to the hyperedge correctness objective.

Instead, it may be more sensible to maximize the greatest overlap a hyperedge in H_Q has with that in H_D. To do so, we simultaneously try to find an alignment σ_V of the vertices of H_Q to those of H_D and an alignment σ_E of the hyperedges of H_Q to those of H_D. This is represented by the hyperedge overlap objective

$$R_O(\sigma_V, \sigma_E) \coloneqq \sum_{e \in E_Q} |\{\sigma_V(v) \mid v \in e\} \cap \sigma_E(e)|. \tag{2}$$

The hyperedge overlap objective as presented here is then resistant to small differences between a hyperedge in H_Q and some corresponding ground truth hyperedge in H_D.

Expanding out the intersection terms, we see that hyperedge overlap objective is equivalent to

$$R_O(\sigma_V, \sigma_E) = \sum_{e \in E_Q} \sum_{v \in V_Q} \mathbb{1}[v \in e \wedge \sigma_V(v) \in \sigma_E(e)].$$

As such, another interpretation of the hyperedge overlap objective is that we are attempting to align the vertices and hyperedges of H_Q and H_D in order

to maximize the number of incident pairs (u, e) which are present in H_D as $(\sigma(u), \sigma(e))$.

To improve our setup further, note that $|\{\sigma_V(v) \mid v \in e\} \cap \sigma_E(e)|$ has a value of at most $\min(|e|, \sigma_E(e))$. As a result, $\{\sigma_V(v) \mid v \in e\} = \sigma_E(e)$ contributes just as much to the objective as $\{\sigma_V(v) \mid v \in e\} \subset \sigma_E(e)$. In order to encourage matching hyperedges of similar cardinalates, we introduce the hyperedge incidence objective

$$R_I(\sigma_V, \sigma_E) = \frac{1}{|E_Q|} \sum_{e \in E_Q} \underbrace{\frac{|\{\sigma_V(v) \mid v \in e\} \cap \sigma_E(e)|}{\sqrt{|e||\sigma_E(e)|}}}_{\text{overlap term}}.$$

Each overlap term is equal to one if and only if e aligns perfectly to $\sigma(e)$. As a result, for any σ_V there exists σ_E such that $R_{EC}(\sigma_V) \leq R_I(\sigma_V, \sigma_E)$ and $R_I(\sigma_V, \sigma_E) = 1$ if and only if $R_{EC}(\sigma_V) = 1$.

Given an incidence matrix B of a hypergraph H, let $\tilde{B}$ be the hyperedge normalized incidence matrix of H, defined as $\tilde{B}_{ue} = \frac{B_{ue}}{\sqrt{|e|}}$. If we utilize X, Y as the matrix representations of σ_V and σ_E respectively, we can rewrite this objective as $\langle \tilde{B}_Q Y, X \tilde{B}_D \rangle_F$. As a result, we may define the more general incidence alignment problem as

$$\begin{aligned} \max_{X,Y} \quad & \langle W_V, X \rangle_F + \langle W_E, Y \rangle_F + \beta \langle \tilde{B}_Q Y, X \tilde{B}_D \rangle_F \\ & \|X\|_1, \|X\|_\infty \leq 1, \qquad \|Y\|_1, \|Y\|_\infty \leq 1 \\ \text{such that} \quad & X \odot \Pi_V = X, \qquad Y \odot \Pi_E = Y \\ & X \in \{0,1\}^{|V_Q| \times |V_D|}, \quad Y \in \{0,1\}^{|E_Q| \times |E_D|} \end{aligned} \tag{3}$$

Notably, this formulation for $k = 2$ is a quadratic program rather than a polynomial program. Furthermore, for $W_V = W_E = 0$, $|V_Q| \leq |V_D|$, and $|E_Q| \leq |E_D|$, Eq. 3 functions as a weighted extension of the objective used by BiG-Align as described in Sect. 2.2.

It may be the case that we want to allow multiple hyperedges in the query hypergraph to be mapped to the same hyperedge in the data hypergraph; for example, two overlapping friend groups represented in your query hypergraph correspond to a single overall friend group in the data hypergraph. In this case, we could once again focus primarily on vertex alignment and introduce the non-exclusive overlap objective

$$R_{XO}(\sigma) = \frac{1}{|E_Q|} \sum_{e \in E_Q} \max_{e' \in E_D} \frac{|\{\sigma(v) \mid v \in e\} \cap e'|}{\sqrt{|e||e'|}}.$$

This itself corresponds to a more relaxed version of the incidence alignment problem where the injection constraint $\|Y\|_1 \leq 1$ on the edge matching has been removed from Eq. 3.

3.2 Similarity Initialization

For the purposes of similarity propagation, the computation of an initial similarity is important on multiple fronts. For one, it's what will be utilized to compute a sparse set of candidate pairs for large instances where considering all pairs for the final alignment would be computationally inefficient. What's more, a poor initial similarity may take a much longer time to converge (if ever) when running similarity propagation. As a result, it is very important to start with a good initial guess as to the similarity.

Previous works on similarity-based hypergraph alignments have utilized symmetric comparison functions of the form $f(a, b)$ where $f(a, b) \leq f(a, a)$ for all b. This comparison function is then fed an attribute or vector of attributes associated with a given vertex in order to compute an initial similarity. For example, given vectors x, y one could set an initial similarity as $S_{ij} = f(x_i, y_j)$.

For topology based graph alignment, degree alone has seen success as an input attribute for determining similarity [20,12]. With this approach, one would use the attribute vectors $x_v = \sum_{e \in E} B_{ve}$ and $y_v = \sum_{e \in E} B_{ve}$, with an analogous definition using cardinality to compute the similarity between pairs of edges. One natural option for an initial similarity for solving the incidence alignment problem in Eq. 3 would be to instead use a weighted degree based on the edge normalized adjacency $\tilde{B}$. Here, the attribute would be $x_v = \sum_{e \in E} \tilde{B}_{ve}$ with an analogous definition for hyperedges.

However, the question remains as to how to incorporate the importance matrices W_V, W_E. To do so, we utilize a technique analogous to the idea of the eigenvector centrality for graphs. The eigenvector centrality of a graph G is given by the eigenvector corresponding to the largest eigenvalue λ_1 of the adjacency matrix of G [17]. The eigenvector centrality of a vertex is equal to the value at its position in this eigenvector, and is measure of the importance of this vertex relative to the rest of the graph; furthermore, the eigenvector centrality is related to the degree of the given vertex as well as the value of the eigenvector centrality for its neighbors [17]. If scalability is a concern for certain applications, the eigenvector computation can be replaced by non-convergent but stabilized Gauss-Seidel relaxation in the spirit of algebraic distance [7,21].

For this method, we utilize the left and right singular vectors ℓ, r corresponding to the largest singular value λ of a given normalized incidence matrix $\tilde{B}$. Then $\lambda\ell$ and λr encode a notion of the importance of each vertex and each hyperedge respectively and can be used as attributes for initializing similarity. This can be extended to include the importance matrices W_V, W_E by embedding each of these matrices in to a single block and computing the left and right singular vectors of this block matrix

$$\underset{\substack{\ell_Q, r_Q, \ell_D, r_D \\ \|r_Q\|_2^2 + \|r_D\|_2^2 = \|\ell_Q\|_2^2 + \|\ell_D\|_2^2 = 1}}{\operatorname{argmax}} \begin{bmatrix} u_Q^T & r_D^T \end{bmatrix} \begin{bmatrix} \frac{1}{\beta} W_V & \tilde{B}_Q \\ \tilde{B}_D^T & \frac{1}{\beta} W_E^T \end{bmatrix} \begin{bmatrix} \ell_D \\ r_Q \end{bmatrix}. \tag{4}$$

The vectors r_Q, ℓ_Q, r_D, ℓ_D then not only act as a representation of the importance of a vertex or hyperedge, but they are also influenced by which pairs

are strongly encouraged by W_V, W_E i.e. if $(W_V)_{ij}$ is large, $(\ell_Q)_i$ and $(\ell_D)_j$ are also encouraged to be large.

A particularly useful option for W_V and W_E can be acquired by fixing part of the solution as X^*. Because the Frobenius inner product is bilinear, we get

$$\langle \tilde{B}_Q Y, (X + X^*)\tilde{B}_D \rangle_F = \langle W_E, Y \rangle_F + \langle \tilde{B}_Q Y, X\tilde{B}_D \rangle_F, \quad W_E = \tilde{B}_Q^T X^* B_D. \quad (5)$$

Analogously, we can define $W_V = \tilde{B}_Q Y^* B_D^T$ for some known matching between hyperedges. This allows for an iterative strategy, where if a complete match isn't found, the optimization may be run again with a fixed portion of the solution.

From here, we use the comparison function $f(a, b) = \frac{\min(a,b)}{\max(a,b)}$, as was used in the ELRUNA algorithm [20]. When a sparse initial similarity is needed, a KD-Tree can be used to efficiently get the top k most similar nodes [8].

3.3 Similarity Propagation

In this section, we present the two primary rules which we use to propagate similarities throughout the hypergraph, smoothing these values so that a pair with high similarity boosts the similarity of neighboring pairs. Likewise, if those neighboring pairs have a low similarity, this should instead decrease the similarity of the original pair. In this way, we can reshape the similarity matrix in order to better reflect the topological structure of the hypergraph.

Algorithm 2: Propagate Similarities

Input: Similarity matrices X, Y; incidence matrices B_Q, B_D
for $\_ \in \{1, \ldots, n_{iter}\}$ **do**
  for $(i, j) \in$ *nonzero(Y)* **do**
    Match $N(e_i)$ to $N(e_j)$, update Y_{ij} as in Eq. 6;
  end
  for $(i, j) \in$ *nonzero(X)* **do**
    Match $N(v_i)$ to $N(v_j)$, update X_{ij} as in Eq. 6;
  end
  Decay similarities as in Eq. 7;
end

Rule 1: Local Matching A naive implementation of similarity smoothing for hypergraphs punishes pairs of hyperedges that connect pairs of vertices that are not proportionally similar to one another. Instead, we instead attempt to find a local matching in the neighborhood of the pair of vertices whose similarities we're updating.

Finding a local matching prevents any one pair from contributing its similarity multiple times, preventing any one pair with high similarity from dominating

all of its neighbors. Likewise, in the event that most pairs are dissimilar, it allows for the few pairs which are actually relevant to have a greater impact on the update. A depiction of the local matching is given in Fig. 1.

Letting b_v and b_e be the greatest similarity that the vertex v or hyperedge e has with another node, the update is then

$$X_{ij} = \frac{\texttt{dominant\_match}(Y, \tilde{B}_Q, \tilde{B}_D, i, j)}{\max\left(\sum_{u \in N(i)} b_u, \sum_{u \in N(j)} b_u\right)}, \tag{6}$$

Where `dominant_match` is the locally dominant matching algorithm described here [11], acting on the submatrix of Y described by the i-th column of B_Q and the j-th column for B_D. The update for Y is defined analogously.

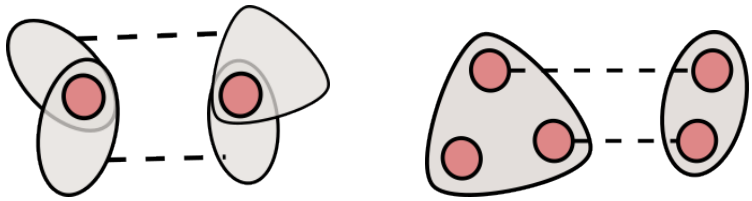

Fig. 1: An update to the similarity for a pair of vertices is accomplished by aligning sets of hyperedges incident to those vertices, and analogously for the similarity of a pair of hyperedges. In this way, a given pair will have high similarity if and only if there is a matching with a high score in their neighborhood.

Rule 2: Similarity Cooling In order to encourage convergence, we also include a cooling rule for the similarity propagation process. This rule penalizes pairs which are not locally dominant, pulling them down towards a lower threshold below which will be rounded to zero.

Let $c_v = \frac{b_v}{T_v}$ and $c_u = \frac{b_u}{T_u}$. Then the cooling strategy utilized in ELRUHNA is defined by

$$\text{decay}(s, t_1, t_2) = \begin{cases} s & \max(t_1, t_2) \leq s \\ s\sqrt{\frac{\min(t_1,t_2)}{\max(t_1,t_2)}} & \min(t_1, t_2) \leq s \\ 0 & \text{otherwise} \end{cases}$$

$$X^*_{ab} = \text{decay}\left(X^*_{ab}, \frac{b_a}{T_a}, \frac{b_b}{T_b}\right) \qquad Y^*_{cd} = \text{decay}\left(Y^*_{cd}, \frac{b_c}{T_c}, \frac{b_d}{T_d}\right) \tag{7}$$

Pairs whose similarity falls between the dominant match for the two vertices will be scaled downwards at a rate proportional to the difference between the two values.

4 Evaluation

In this section, we will examine the performance of the ELRUHNA algorithm in comparison related methods such as BiG-Align, and Cone-Align [12,8]. While HyperAlign [9] is relevant, we are unable to compare against it at this time due to reproducibility issues. Likewise, methods such as HNN-HM are specialized towards an image processing domain and are not appropriate for the test cases available here.

We will focus on a number of instances from the HyperAlign paper [9] and the XGI repositories of hypergraph datasets [13]. This is a collection of hypergraphs from domains such as gene-disease association, co-authorship patterns, and social network analysis [6,19,3]. For each hypergraph used in this paper, we follow the approach used in [9] and compute the 2 core of the bipartite representation of the hypergraph; that is, we take the maximal induced subgraph such that every vertex has degree at least 2 and every hyperedge has a size of at least 2. This is because it is often infeasible to differentiate degree 1 vertices using only topological information. The hypergraphs used range from several hundred vertices to tens of thousands after taking this core. Methods utilizing the bipartite representation and dense similarity matrices are constrained primarily by the maximum of $|V|$ and $|E|$: a summary of these sizes is presented in Table 1.

We focus primarily on aligning a given hypergraph with a noisy version of itself, which is a common downstream task to evaluate the alignment. In order to add noise to these hypergraphs, we utilize a probabilistic hypergraph model described in [2]. For each random hyperedge, we assume that each vertex has a small independent probability p of being in that hyperedge. The size of this random hyperedge is then provided by a binomial distribution with $|V|$ events and probability p. We generally assume we are working with hypergraphs whose rank and maximum degree are significantly less than $|V|$. As such, $p << |V|$, and the size of each random hyperedge is approximately Poisson distributed with rate $\lambda = |V|p$. As such, the size of each hyperedge is determined by sampling from a Poisson distribution, then sampling that many vertices without replacement from V. Throughout this paper, we choose $\lambda = \bar{k}$, where $\bar{k}$ is the average size of hyperedges in the hypergraph.

4.1 Qualitative Assessment

Let us compare the quality of ELRUHNA with competing methods on a number of small hypergraph examples which are suitable for dense methods. In particular, we compare against BiG-Align, which similarly works on the bipartite representation of the hypergraph structure. Additionally, we compare these results to that of Cone-Align, which is a state of the art method acting on the clique expansion of the hypergraphs.

In particular, for these tests we compare the ground truth accuracy R_{acc} achieved by these methods on the provided hypergraphs. Each hypergraph is evaluated at 5 different noise levels across 10 different random seeds. We summarize the average of these trials in Fig. 2. This figure shows the average true

Table 1: Hypergraph instances used in this paper, together with the size of their bipartite representation $|V|+|E|$ and the average size of hyperedges in the sample.

Small bipartite representation

	$\lvert V\rvert$	$\lvert E\rvert$	$\bar{k}$
email-Enron	143	1457	3.1
NDC-classes	561	835	6.9
diseasome	114	152	2.5
house-committees	292	124	9.5
senate-committees	263	283	16.8

Large bipartite representation

	$\lvert V\rvert$	$\lvert E\rvert$	$\bar{k}$
coauth-Geology	9537	36659	3.1
contact-high-school	327	4858	2.3
coauth-History	14968	20160	2.4
contact-primary-school	242	12704	2.4
email-Eu	900	24319	3.5
threads-ask-ubuntu	7522	31567	2.3

accuracy over 10 runs as a function of the noise level of the input graph. On this collection, ELRUHNA was able to achieve over 90% true accuracy on average for some instances in a low noise setting with a standard deviation of only 0.01. This is as much as a 25% increase in true accuracy over BiG-Align. These results significantly outperformed the results found using the clique expansion of the hypergraph, which struggled even in low noise setting. However, while ELRUHNA fairly consistently performs better on average than the other methods tested, we did note a significant increase in standard deviation at higher noise levels, up to 0.3. for some instances. Addressing these instabilities in the presence of large amounts of noise is a topic of future work.

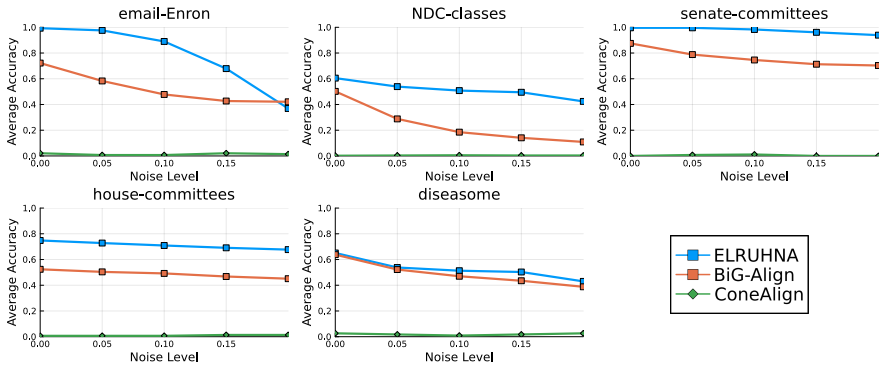

Fig. 2: A comparison of ELRUHNA, BiG-Align, and Cone-Align for a collection of small hypergraph samples. Each subfigure gives the accuracy of the returned permutation as a function of the noise level in the input.

Next, we examine the behavior of ELRUHNA in a sparse setting, where it is intractable to represent the whole similarity matrix in dense form. The experiment setup remains generally the same as that used with the smaller

hypergraph instances, although in this case we will be considering hyperedge correctness as a metric over true accuracy. when sparsifying, we consider only the top $\lceil \log_2(\max(|V|, |E|)) \rceil$ pairs. The results of these tests are shown in Fig. 3. Notably, for the noise free setting where the input hypergraph is only permuted, ELRUHNA is still able to achieve a near perfect solution for some instances despite the aggressive sparsification. In the presence of noise, ELRUHNA fairly consistently outputs results in the range of $15 - 25\%$ hyperedge correctness.

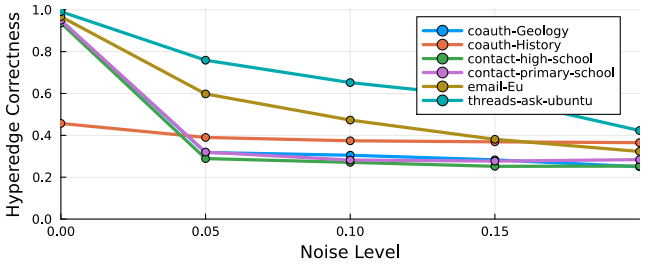

Fig. 3: The hyperedge correctness found for a collection of larger larger hypergraph inputs as a function of noise level using the sparse mode of ELRUHNA.

5 Conclusion

Hypergraph Alignment is an important NP-hard combinatorial optimization problem with applications in a wide variety of fields from biomedical to social sciences. In this paper, we have discussed the hyperedge overlap formulation of the hypergraph alignment problem in depth and provided an optimization formulation which connects this concept back to existing literature on alignment for bipartite graphs. What's more, we've provided an iterative framework for propagating similarities using the bipartite representation of the hypergraph using a pair of elimination rules. We've also discussed how to initialize these similarities by contrasting the importance of each pair using an extension of the eigenvector centrality to incidence matrices. Finally, we've explained how this can be extended to a sparse domain by iteratively updating the importance matrix W based on the current solution in order to guide a sparse local solution to a global one. We demonstrate an improvement of up to 25% in terms of true accuracy compared to relevant alignment methods and show that we can acquire reasonable solutions for hypergraphs with tens of thousands of vertices in the bipartite representation. *The source code and data are available at https://github.com/cameton/ASONAM_ELRUHNA.*

Acknowledgments: This research is based upon work supported by the U.S. Department of Energy (DOE) through the Exascale Computing Project (17-SC-20-SC) (ExaGraph) at the Pacific Northwest National Laboratory (PNNL) the

Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through the Advanced Graphic Intelligence Logical Computing Environment (AGILE) research program, contract number 77740, and Laboratory Directed Research and Development funds at PNNL.

References

1. Aladağ, A.E., Erten, C.: Spinal: scalable protein interaction network alignment. Bioinformatics **29**(7) (2013) 917–924
2. Barthelemy, M.: Class of models for random hypergraphs. Phys. Rev. E **106** (Dec 2022) 064310
3. Bastian, M., Heymann, S., Jacomy, M.: Gephi: an open source software for exploring and manipulating networks. In: Proceedings of the international AAAI conference on web and social media. Volume 3. (2009) 361–362
4. Bayati, M., Gerritsen, M., Gleich, D.F., Saberi, A., Wang, Y.: Algorithms for large, sparse network alignment problems. In: 2009 Ninth IEEE International Conference on Data Mining, IEEE (2009) 705–710
5. Bayati, M., Gleich, D.F., Saberi, A., Wang, Y.: Message-passing algorithms for sparse network alignment. ACM Trans. Knowl. Discov. Data **7**(1) (March 2013)
6. Benson, A.R., Abebe, R., Schaub, M.T., Jadbabaie, A., Kleinberg, J.: Simplicial closure and higher-order link prediction. Proceedings of the National Academy of Sciences **115**(48) (2018) E11221–E11230
7. Chen, J., Safro, I.: Algebraic distance on graphs. SIAM Journal on Scientific Computing **33**(6) (2011) 3468–3490
8. Chen, X., Heimann, M., Vahedian, F., Koutra, D.: CONE-Align: Consistent Network Alignment with Proximity-Preserving Node Embedding. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. (October 2020) 1985–1988
9. Do, M.T., Shin, K.: Unsupervised alignment of hypergraphs with different scales. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. (2024) 609–620
10. Friedland, S., Lasserre, J.B., Lim, L.H., Nie, J.: Special Issue: Polynomial and Tensor Optimization. Mathematical Programming **193**(2) (June 2022) 511–512
11. Khan, A.M., Gleich, D.F., Pothen, A., Halappanavar, M.: A multithreaded algorithm for network alignment via approximate matching. In: SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. (2012) 1–11
12. Koutra, D., Tong, H., Lubensky, D.: Big-align: Fast bipartite graph alignment. In: 2013 IEEE 13th international conference on data mining, IEEE (2013) 389–398
13. Landry, N.W., Lucas, M., Iacopini, I., Petri, G., Schwarze, A., Patania, A., Torres, L.: Xgi: A python package for higher-order interaction networks. Journal of Open Source Software **8**(85) (2023) 5162
14. Liao, X., Xu, Y., Ling, H.: Hypergraph neural networks for hypergraph matching. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). (2021) 1246–1255
15. Liao, X., Xu, Y., Ling, H.: Hypergraph Neural Networks for Hypergraph Matching. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2021) 1266–1275

16. Mao, X., Wang, W., Wu, Y., Lan, M.: From alignment to assignment: Frustratingly simple unsupervised entity alignment. In Moens, M.F., Huang, X., Specia, L., Yih, S.W.t., eds.: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Online and Punta Cana, Dominican Republic, Association for Computational Linguistics (November 2021) 2843–2853
17. Newman, M.E.: The mathematics of networks. The new palgrave encyclopedia of economics **2**(2008) (2008) 1–12
18. Neyshabur, B., Khadem, A., Hashemifar, S., Arab, S.S.: Netal: a new graph-based method for global alignment of protein–protein interaction networks. Bioinformatics **29**(13) (2013) 1654–1662
19. Piñero, J., Ramírez-Anguita, J.M., Saüch-Pitarch, J., Ronzano, F., Centeno, E., Sanz, F., Furlong, L.I.: The disgenet knowledge platform for disease genomics: 2019 update. Nucleic acids research **48**(D1) (2020) D845–D855
20. Qiu, Z., Shaydulin, R., Liu, X., Alexeev, Y., Henry, C.S., Safro, I.: ELRUNA: Elimination Rule-based Network Alignment. ACM J. Exp. Algorithmics **26** (December 2021) 1–32
21. Shaydulin, R., Chen, J., Safro, I.: Relaxation-based coarsening for multilevel hypergraph partitioning. Multiscale Modeling & Simulation **17**(1) (2019) 482–506
22. Tan, S., Guan, Z., Cai, D., Qin, X., Bu, J., Chen, C.: Mapping users across networks by manifold alignment on hypergraph. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence. AAAI'14, AAAI Press (2014) 159–165
23. Xiang, L., Khan, A., Ferdous, S.M., Aravind, S., Halappanavar, M.: cuAlign: Scalable Network Alignment on GPU Accelerators. In: Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis. SC-W '23, New York, NY, USA, Association for Computing Machinery (November 2023) 747–755
24. Yan, J., Li, C., Li, Y., Cao, G.: Adaptive Discrete Hypergraph Matching. IEEE Transactions on Cybernetics **48**(2) (February 2018) 765–779
25. Yan, J., Yin, X.C., Lin, W., Deng, C., Zha, H., Yang, X.: A short survey of recent advances in graph matching. In: Proceedings of the 2016 ACM on international conference on multimedia retrieval. (2016) 167–174