Message-Passing Algorithms for Sparse Network Alignment

MOHSEN BAYATI, Stanford University
DAVID F. GLEICH, Purdue University
AMIN SABERI, Stanford University
YING WANG, Google

Network alignment generalizes and unifies several approaches for forming a matching or alignment between the vertices of two graphs. We study a mathematical programming framework for network alignment problem and a sparse variation of it where only a small number of matches between the vertices of the two graphs are possible. We propose a new message passing algorithm that allows us to compute, very efficiently, approximate solutions to the sparse network alignment problems with graph sizes as large as hundreds of thousands of vertices. We also provide extensive simulations comparing our algorithms with two of the best solvers for network alignment problems on two synthetic matching problems, two bioinformatics problems, and three large ontology alignment problems including a multilingual problem with a known labeled alignment.

Categories and Subject Descriptors: H.4.0 [**Information Systems Applications**]: General

General Terms: Algorithms

Additional Key Words and Phrases: Network alignment, graph matching, belief propagation, message-passing

ACM Reference Format:
Bayati, M., Gleich, D. F., Saberi, A., and Wang, Y. 2013. Message-passing algorithms for sparse network alignment. *ACM Trans. Knowl. Discov. Data* 7, 1, Article 3 (March 2013), 31 pages.
DOI: http://dx.doi.org/10.1145/2435209.2435212

1. INTRODUCTION

The focus of this article is to find approximate isomorphisms, or alignments, between large graphs. This problem is motivated by applications in several areas including biology, computer vision, and natural language processing. For example, the study of protein interactions across different species has made network alignment a common topic in computational biology [Flannick et al. 2006, 2008; Klau 2009; Kuchaiev et al. 2009; Singh et al. 2007, 2008]. In computer vision, network alignment is used for matching images [Conte et al. 2004; Schellewald and Schnörr 2005], and in the ontology alignment, it is used for finding correspondence between different representations of a database [Lacoste-Julien et al. 2006; Melnik et al. 2002; Šváb 2007].

This work was partially supported by NSF grants DBI-0850203, HRD-0833093, and DMS-0915110, and DHS grants 2009-ST-062-00016 and 2010-ST-062-000039. The Library of Congress also funded this work.
Authors' addresses: M. Bayati, Stanford University, Graduate School of Business, 450 Serra Mall, Stanford, CA 94305; email: bayati_mohsen@gsb.stanford.edu; D. F. Gleich, Purdue University, Computer Science Department, 101 N. Grant Street #110, West Lafayette, IN 47907; email: dgleich@purdue.edu; A. Saberi, Stanford University, Management Science and Engineering Department, 450 Serra Mall, Stanford, CA 94305; email: saberi@stanford.edu; Y. Wang, Google, 1600 Ampitheatre Pkwy, Mountain View, CA; email: yingmwang@google.com.

DOI: http://dx.doi.org/10.1145/2435209.2435212

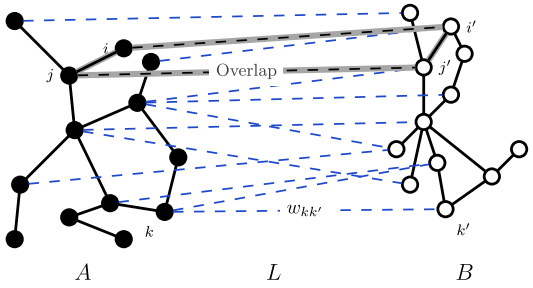


Fig. 1. The setup for network alignment problem. The goal is to maximize the number of overlaps in any matching subset of L and the weight of the matching.

The formulation of the problem studied in this article is a variation of classic algorithmic problems: graph isomorphism, maximum common subgraph, and the quadratic assignment problem. Because of the intractability of the problem, our focus will be on practical heuristics. We will give a quick review of the existing results and their applications. Then, we will present two message passing algorithms that yield near optimal results in practice determined by comparison to an upper bound from a linear program. Both algorithms easily work on graphs with 100,000–1,000,000 vertices. Because our algorithms use message passing, they can be parallelized on MapReduce and bulk-synchronous processing architectures for even larger problems.

1.1. Problem Definition

Consider two sets of vertices $V_A = \{1, 2, \ldots, n\}$ and $V_B = \{1', 2', \ldots, m'\}$. Let $A = (V_A, E_A)$ and $B = (V_B, E_B)$ be two undirected graphs with their respective vertex and edge sets. Let L be a bipartite graph between the vertices of A and B, formally $L = (V_A \cup V_B, E_L)$. Our overall goal is to find a matching between A and B using only edges from L. In other words, we seek a subset of E_L such that no two edges share a common endpoint. Under such a matching M, we say that an edge $(i,j) \in E_A$ is *overlapped* with $(i',j') \in E_B$ if (i, i') and (j,j') belong to M. See Figure 1 for an illustration.

More generally, and following Singh et al. [2007], we will study the case where the edges between A and B are weighted. That is, each edge $e = (k, k') \in E_L$ has a nonnegative weight w_e indicating a measure of similarity between vertices k and k'. In these cases, a matching has a weight that is equal to the sum of the weights of edges in the matching.

Definition 1.1. Given graphs A, B and L, as well as the weight function w, find a matching M maximizing a linear combination of the matching weight and the number of overlapped edges.

This problem is a generalization of several NP-complete problems including the densest subgraph problem as well as the maximum common subgraph problem. The latter is also known to be APX-hard.

1.2. Our Contribution

In this article, we provide

(1) two novel message passing algorithms: NetAlignMP and NetAlignMP++ for the problem based on max-product belief propagation (Section 5);
(2) an extensive comparison between NetAlignMP, NetAlignMP++ and two of the best existing algorithms on two synthetic matching problems (Section 7), two

Table I. Notation for the Paper

A, S	capital letters are sets and graphs
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{S}$	bold capitals are matrices
$\boldsymbol{x}, \boldsymbol{w}$	lowercase bold letters are vectors
$A_{ij}, \boldsymbol{S}[ii', jj']$	subscripts or brackets denote matrix entries
$(i,i'), ii'$	edges in L
$ii' \square jj'$	squares in $E_L \times E_L$
$\mathbf{1}_n$	n by 1 vector of all ones
$\boldsymbol{Ax}, \boldsymbol{AB}$	standard vector and matrix products
$\boldsymbol{A} \bullet \boldsymbol{B} = \sum_{ij} \boldsymbol{A}_{ij} \boldsymbol{B}_{ij}$	matrix inner product

bioinformatics problems, and three large ontology alignment problems (Section 8) including a multilingual problem with a known alignment.

We will show that our algorithms are fast, robust, and yield near-optimal[1] objective values for a large family of graphs, including real datasets. In one of the cases where there is a known alignment produced by experts, our algorithms recover a large fraction of the correct matches quickly, without any tweaking.

In evaluating our algorithms on synthetic datasets, we will observe that for both sparse and dense cases, our algorithms produce near-optimal solution (using the theoretical upper bounds). As the number of edges in L increases, our approaches outperform existing methods by a factor of 2 or more. On the other hand, when there are only a small number of potential matches in L, our results nearly ties with Klau's algorithm [Klau 2009].

All of our algorithms are implemented in MATLAB and the software and datasets for this paper are available to public from the web page `http://www.cs.purdue.edu/homes/dgleich/codes/netalign`.

We also include all the experimental code to reproduce the figures in this paper. Using our network alignment MATLAB package, solving the network alignment problem in Figure 1 is done with the following code.

```
load('data/example_overlap.mat');   % load the data
[S,w,li,lj]=netalign_setup(A,B,L);  % setup the quadratic program
x=netalignmp(S,w,1,1,li,lj);        % call our netalignmp
[mi ma mb]=mwmround(x,li,lj);       % round to matching
```

This manuscript is an extension of our previous paper [Bayati et al. 2009] and includes a full derivation of our belief propagation algorithm, as well as a new algorithm. It also includes a more thorough experimental evaluation.

2. A MATHEMATICAL PROGRAM FOR NETWORK ALIGNMENT

In this section, we adapt standard mathematical programming ideas to formulate the network alignment as a quadratic program (QP). Let us start by introducing some notation in Table I.

Given $A = (V_A, E_A)$, $B = (V_B, E_B)$, and $L = (V_A \cup V_B, E_L)$, our goal is to produce a matching M to maximize a linear combination of overlap and matching weight. For each edge of E_L we will use the notations (i,i') and ii' interchangeably. Each matching in E_L is represented by a zero-one vector by assigning a variable $x_{ii'}$ to each $ii' \in E_L$,

[1]For the families of graphs that we study, we have a theoretical upper bound provided by linear programming for the objective function. Hence, we can check the quality of all algorithms' solutions.

which is equal to 1 if ii' is in the matching or 0 if it is not. For convenience of notation, we define an ordering $\mathcal{O}_L$ over the set E_L. We will use the same ordering in vector representation of the edges. Note that $x_{ii'}$ is *only defined* for edges in L, and $|E_L| \ll |V_A| \cdot |V_B|$. This differs from many other formulations of the problem where the set L is implicitly the full bipartite collection.

Next, we define a zero-one matrix $\boldsymbol{S}$ of size $|E_L| \times |E_L|$ indexed by edges of E_L. Denote the entry at row ii' and column jj' by $\boldsymbol{S}[ii',jj']$, where

$$\boldsymbol{S}[ii',jj'] = \begin{cases} 1 & \text{if } (i,j) \in E_A \text{ and } (i',j') \in E_B \\ 0 & \text{otherwise.} \end{cases}$$

We also say that two edges ii' and jj' in E_L form a *square* if $\boldsymbol{S}[ii',jj'] = 1$ and denote it by $ii' \square jj'$. In other words, $\boldsymbol{S}$ is the indicator matrix of all squares.

Let $\boldsymbol{x}$ be the indicator vector for a matching. The total number of overlapped edges is

$$(1/2)\boldsymbol{x}^T\boldsymbol{S}\boldsymbol{x} = \sum_{ii' \square jj'} x_{ii'}x_{jj'}.$$

Moreover, let $w_{ii'}$ be the weight of each ii' and denote the vector of all weights by $\boldsymbol{w}$. The constraint that $\boldsymbol{x}$ must be a valid matching can be written by set of linear inequalities. For all vertices $(i,i') \in L$,

$$\sum_{j':(ij') \in E_L} x_{ij'} \le 1, \qquad \sum_{j:(ji') \in E_L} x_{ji'} \le 1 \qquad x_{ii'} \in \{0,1\}.$$

To write these constraints more compactly, define $\boldsymbol{C}$ to be the binary incidence matrix of graph L of dimensions $|V_L| \times |E_L|$. Then the matching constraints can be written as $\boldsymbol{C}\boldsymbol{x} \le \boldsymbol{1}_{|V_L|}$.

Using these definitions, the network alignment problem is an integer quadratic program (QP)

$$\begin{aligned} &\underset{\boldsymbol{x}}{\text{maximize}} \;\; \alpha\boldsymbol{w}^T\boldsymbol{x} + \beta/2\boldsymbol{x}^T\boldsymbol{S}\boldsymbol{x} \\ &\text{subject to} \;\; \boldsymbol{C}\boldsymbol{x} \le \boldsymbol{1}_{n+m}, \qquad x_{ii'} \in \{0,1\}, \end{aligned} \tag{NAQP}$$

where α and β are arbitrarily chosen nonnegative constants that define the trade-off between the similarity and overlap objectives. When $\alpha = 0$ and $\beta = 1$, then the program solves a special case we call the overlap graph matching problem or the pure overlap problem. When $\alpha = 1$ and $\beta = 0$, it solves the maximum weight matching problem. Figure 2 shows an example of the network alignment problem and an explicit construction of the matrices $\boldsymbol{S}$ and $\boldsymbol{C}$.

A general integer QP is an NP-hard problem. In fact, even a real-valued QP with an indefinite Hessian matrix is an NP-hard problem. Because the matrix $\boldsymbol{S}$ is indefinite, we cannot easily find the global maximizer even after relaxing the constraints.

This derivation is closely related to what is done by Klau [2009], and also related to standard statement of a quadratic assignment problem [Burkard et al. 2012]. The primary difference is that we specialize on the case when only a small subset of possible matches is present. We derive a few additional relationships to the standard case in the discussion of the IsoRank algorithm (Section 4.1).

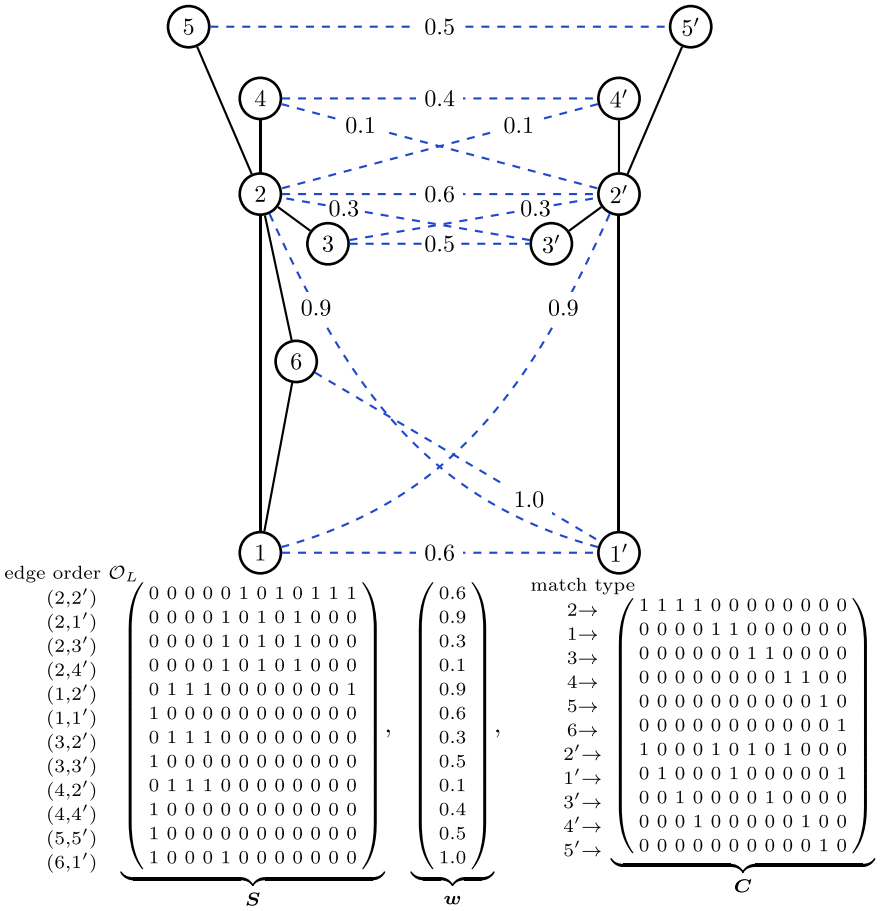


Fig. 2. A small sample problem and the data for the QP formulation.

3. APPLICATIONS

Network alignment is deeply intertwined with many classical computational problems such as graph isomorphism, quadratic assignment, maximum common subgraph, and maximum clique. For a survey of these connections, see Conte et al. [2004]. In this section, we briefly highlight the key applications of network alignment that appear in pattern recognition, ontology alignment and bioinformatics.

3.1. Pattern Recognition

Network alignment for pattern recognition involves identifying a small model graph within a large scene graph. The model graph typically represents the desired pattern—a rooftop, a face, a person—and the scene graph describes the entire space—possibly a picture. The assumptions are often that the data are noisy and the goal is not an exact subgraph isomorphism. Again, Conte et al. [2004] is a good starting point to explore this literature. Recent work includes trying to *learn* good scores $\boldsymbol{w}$ to avoid using the quadratic matching formulations [Caetano et al. 2009].

3.2. Ontology Matching

An ontology is a set of statements, which connect subjects to objects with verbs. An elementary example is an ontology describing the authors of this article:

subject	verb	object
David Gleich	*wrote*	Sparse Network Alignment
Mohsen Bayati	*wrote*	Sparse Network Alignment
Sparse Network Alignment	*is an*	Academic Manuscript
Academic Manuscript	*is a*	Paper

and so on. An ontology is a flexible data description format, and a fundamental problem is how to align two ontologies about the same data. Suppose that Citeseer and DBLP expose their networks of papers as an ontology, the problem of ontology alignment is to figure out the correspondence between Citeseer papers and DBLP papers. This problem has been studied extensively. See Hu et al. [2005, 2008], Ehrig and Staab [2004], and Blondel et al. [2004] for a few different approaches to these problems. All of these approaches utilize some heuristic approach for a network alignment problem.

3.3. Finding Common Pathways in Biological Networks

Network alignment is becoming a small industry within bioinformatics. Broadly speaking, this emergence is due to the rapid increase in high quality data about protein interactions. A protein-protein interaction (PPI) graph has proteins as vertices and edges that connect proteins known to interact. Suppose that A and B are two PPI networks, and we compute an alignment between them. The alignment produces a one-to-one mapping between proteins in A and proteins in B. If the proteins are from two different species, then the alignment hints at similar functions for the two proteins, or two groups of proteins. Alternatively, we may know information about proteins in A. An alignment with B suggests what information about A might apply to the proteins in B.

Due to the wide interest in this problem, several tools have been developed for aligning protein-protein interaction networks. These include NetworkBLAST [Sharan et al. 2005], MAWISH [Koyutürk et al. 2006] NetAlign [Liang et al. 2006], Græmlin [Flannick et al. 2006, 2008], IsoRank [Singh et al. 2007, 2008], GRAAL [Kuchaiev et al. 2010], Natalie [Klau 2009], Natalie 2.0 [El-Kebir et al. 2011] and the algorithm of Bradde et al. [2010]. Some of these tools have extensions for aligning more than two networks, but we focus on the two network case here. We review the IsoRank algorithm in detail in Section 4.1. Alternative approaches are proposed by Berg and Lässig [2006] and Kuchaiev et al. [2009].

4. EXISTING ALGORITHMS FOR NETWORK ALIGNMENT

In this section, we review existing algorithms that produce good solutions for the network alignment.

4.1. IsoRank Algorithm

Singh et al. [2007] proposed IsoRank to approximately solve NAQP when L is a complete bipartite graph. In this section, we present IsoRank and our variation SpaIsoRank, which is more efficient when L is sparse.

The main idea of IsoRank algorithm is to approximate the objective of NAQP without direct concern for the matching constraints. Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be the adjacency matrices for graphs A and B, and also let $\boldsymbol{D}_A$ and $\boldsymbol{D}_B$ be the diagonal matrices of their degrees, respectively. IsoRank solves for the matrix $\boldsymbol{Z}$ that satisfies:

$$\gamma \underbrace{\boldsymbol{A}^T\boldsymbol{D}_A}_{\boldsymbol{P}^T} \boldsymbol{Z} \underbrace{\boldsymbol{D}_B\boldsymbol{B}}_{\boldsymbol{Q}} + (1-\gamma)\boldsymbol{W} = \boldsymbol{Z}.$$

Here $W_{i,j} = w_{i,j}$ is the weight function on edges in L represented as a matrix when L is the complete bipartite graph. The intuition is that each entry $Z_{i,i'}$ is a real number based on a weighted average of all neighboring values $Z_{j,j'}$ where $(i,j) \in E_A$ and $(i',j') \in E_B$. With this heuristic solution $\boldsymbol{Z}$, they compute a binary solution $\boldsymbol{X}$ by solving a maximum weight matching problem where the weights are from $\boldsymbol{Z}$. We discuss rounding schemes in more detail in Section 6.

Now we discuss our extensions of IsoRank algorithm for the case when L is sparse. This extension rests on the Kronecker product. Recall that the Kronecker product of an $m \times n$ matrix $\boldsymbol{A}$ and another matrix $\boldsymbol{B}$ is defined by

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{pmatrix} A_{11}\boldsymbol{B} & \cdots & A_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ A_{m1}\boldsymbol{B} & \cdots & A_{mn}\boldsymbol{B} \end{pmatrix}. \tag{1}$$

When L is the complete bipartite graph, then the matrix indicating potential overlaps, or squares, is $\boldsymbol{S} = \boldsymbol{B} \otimes \boldsymbol{A}$ (up to an arbitrary permutation based on the edge order $\mathcal{O}_L$). Moreover, the mixed product property states that

$$\boldsymbol{P}^T\boldsymbol{Z}\boldsymbol{Q} = (\boldsymbol{Q} \otimes \boldsymbol{P})\,\mathrm{vec}(\boldsymbol{Z}),$$

where $\mathrm{vec}(\boldsymbol{Z})$ is a column-wise vector representation of a matrix:

$$\mathrm{vec}(\boldsymbol{Z}) = \begin{bmatrix} \boldsymbol{Z}\boldsymbol{e}_1 \\ \boldsymbol{Z}\boldsymbol{e}_2 \\ \vdots \\ \boldsymbol{Z}\boldsymbol{e}_n \end{bmatrix}.$$

Note that $\boldsymbol{Q} \otimes \boldsymbol{P} = \mathrm{diag}[\boldsymbol{S}\boldsymbol{1}_{|E_L|}]\,(\boldsymbol{B} \otimes \boldsymbol{A})$.

Thus, the following PageRank problem is equivalent to IsoRank when L is the complete bipartite graph, but handles sparse L as well:

$$\gamma \boldsymbol{D}_S\boldsymbol{S}^T\boldsymbol{z} + (1-\gamma)\boldsymbol{w} = \boldsymbol{z}.$$

We compute $\boldsymbol{z}$ using a standard algorithm for PageRank. At each iteration, we employ one of two rounding schemes to produce a matching. The first just uses the vector $\boldsymbol{z}^{(k)}$ as the weight on each edge in L and solves a bipartite max-weight matching problem. The second uses the vector $\alpha\boldsymbol{w} + (\beta/2)\boldsymbol{S}\boldsymbol{z}^{(k)}$ as the weights on $\boldsymbol{L}$ and more closely mirrors the original objective function.

Algorithm SpaIsoRank

INPUT $\boldsymbol{S}$, L, damping (nonnegative) parameter $\gamma < 1$, ε, $\mathrm{n_{iter}}$, rounding_type $\in \{1, 2\}$

$\boldsymbol{v} = \boldsymbol{w} / \left(\mathbf{1}_{|E_L|}^T \boldsymbol{w}\right)$
$\boldsymbol{d} = \boldsymbol{S}\mathbf{1}_{|E_L|}$, $\boldsymbol{P} = \mathrm{diag}[\boldsymbol{d}]^{-1}\boldsymbol{S}$
$\boldsymbol{z}^{(0)} = \boldsymbol{v}$, $\delta = \varepsilon + 1$
for $k = 1$ to $\mathrm{n_{iter}}$ unless $\delta < \varepsilon$
  $\boldsymbol{z}^{(k)} = \gamma \boldsymbol{P}^T \boldsymbol{z}^{(k-1)} + (1-\gamma)\boldsymbol{v}$
  $\delta = \left\| \boldsymbol{z}^{(k)} - \boldsymbol{z}^{(k-1)} \right\|$
  if rounding_type is 1
    $\boldsymbol{x}^{(k)} = \mathrm{bipartite\_match}(\mathrm{L}, \boldsymbol{z}^{(k)})$
    $\mathrm{obj}^{(k)} = \mathrm{objective}(\boldsymbol{x}^{(k)})$
  else if rounding_type is 2
    $\boldsymbol{x}^{(k)} = \mathrm{bipartite\_match}(\mathrm{L}, \alpha\boldsymbol{w} + (\beta/2)\boldsymbol{S}\boldsymbol{z}^{(k)})$
    $\mathrm{obj}^{(k)} = \mathrm{objective}(\boldsymbol{x}^{(k)})$
  end
end
return $\boldsymbol{x}^{(k)}$ with the highest value of $\mathrm{obj}^{(k)}$

4.2. Linear Program Formulations

We now review a series of linear programming (LP) relaxations for network alignment. These ideas originated in mixed integer translations of the quadratic assignment problem [Lawler 1963], and subsequent tightened versions by that were originally described by Frieze and Yadegar [1983] and Adams and Johnson [1994]. The adaptation to network alignment appeared in Klau [2009].

In the first relaxation, Lawler [1963], converted NAQP into a mixed integer linear program. To do so, replace each product $x_{ii'}x_{jj'}$ with a new variable $y_{ii',jj'}$, and add constraints $y_{ii',jj'} \le x_{ii'}$, and $y_{ii',jj'} \le x_{jj'}$. These constraints enforce $y_{ii',jj'} \le x_{ii'}x_{jj'}$ when $x_{ii'}$ and $x_{jj'}$ are binary. We also add symmetry constraints $y_{ii',jj'} = y_{jj',ii'}$. Notice that with the symmetry constraints the constraints $y_{ii',jj'} \le x_{jj'}$ can be dropped.

Before writing the new integer program, let us define $\boldsymbol{Y}_{\boldsymbol{S}}$ to be a matrix with the same dimension as $\boldsymbol{S}$ where

$$\boldsymbol{Y}_{\boldsymbol{S}}[ii',jj'] = \begin{cases} y_{ii',jj'} & \text{if } \boldsymbol{S}[ii',jj'] = 1 \\ 0 & \text{Otherwise.} \end{cases}$$

Thus, we arrive at:

$$\begin{array}{lll} \underset{\boldsymbol{x},\boldsymbol{y}}{\text{maximize}} & \alpha\boldsymbol{w}^T\boldsymbol{x} + \frac{\beta}{2}\sum_{ii'}\sum_{ii'\square jj'} y_{ii',jj'} & \\ \text{subject to} & \boldsymbol{C}\boldsymbol{x} \le \mathbf{1}_{n+m}, & x_{ii'} \in \{0,1\}, \\ & y_{ii',jj'} \le x_{ii'} & \text{for all } ii' \square jj', \\ & \boldsymbol{Y}_{\boldsymbol{S}} = \boldsymbol{Y}_{\boldsymbol{S}}^T & \end{array} \qquad \text{(NAILP)}$$

as a mixed-integer linear program to solve the network alignment problem

In contrast with the quadratic program, we can relax the binary constraint on NAILP and get an efficient algorithm. After we write $\sum_{ii'} \sum_{ii' \square jj'} y_{ii',jj'}$ as $\boldsymbol{S} \bullet \boldsymbol{Y_S}$, the relaxed program is

$$\begin{array}{ll} \underset{\boldsymbol{x},\boldsymbol{y}}{\text{maximize}} & \alpha \boldsymbol{w}^T \boldsymbol{x} + \frac{\beta}{2} \boldsymbol{S} \bullet \boldsymbol{Y_S} \\ \text{subject to} & \boldsymbol{C}\boldsymbol{x} \leq \mathbf{1}_{n+m}, \qquad x_{ii'} \in [0,1], \\ & y_{ii',jj'} \leq x_{ii'} \qquad \text{for all } ii' \square jj', \\ & \boldsymbol{Y_S} = \boldsymbol{Y_S}^T \end{array} . \tag{NARLP}$$

It admits a polynomial-time solution with an appropriate linear program solver.

Remark 4.1. The relaxation NARLP is advantageous because it yields an upper bound on the objective value of the network alignment problem. Furthermore, solving NARLP with $\alpha = 0, \beta = 1$ allows us to get an upper bound on the maximum possible overlap between two networks.

4.3. Klau's Iterative Matching Relaxation

Klau [2009] constructed an iterative algorithm to approximate NAQP. The key components of this algorithm are a tighter LP relaxation of NAQP and the Lagrangian decomposition of the symmetry constraints. We first explain the Lagrangian decomposition for NARLP and then show the tightened LP. In the Lagrangian decomposition, we drop all the symmetry constraints $\boldsymbol{Y_S} = \boldsymbol{Y_S}^T$ by adding penalty terms of the form $u_{ii',jj'}(y_{ii',jj'} - y_{jj',ii'})$. Here $u_{ii',jj'}$'s are Lagrange multipliers, a set of $n^2 - n$ new variables. Following this idea, we arrive at

$$\begin{array}{ll} \underset{\boldsymbol{x},\boldsymbol{y}}{\text{maximize}} & \alpha \boldsymbol{w}^T \boldsymbol{x} + \frac{\beta}{2} \boldsymbol{S} \bullet \boldsymbol{Y_S} + \boldsymbol{U_S} \bullet (\boldsymbol{Y_S} - \boldsymbol{Y_S}^T) \\ \text{subject to} & \boldsymbol{C}\boldsymbol{x} \leq \mathbf{1}_{n+m}, \qquad x_{ii'} \in [0,1], \\ & y_{ii',jj'} \leq x_{ii'} \qquad \text{for all } ii' \square jj'. \end{array} \tag{NALLP}$$

When $\boldsymbol{Y_S} = \boldsymbol{Y_S}^T$, the two linear programs NARLP and NALLP are equivalent. Therefore, for any fixed $\boldsymbol{U_S}$ the optimum solution of NALLP is an upper bound for the objective of NARLP, which is itself an upper bound for the network alignment problem. Standard Lagrangian theory dictates that with the optimal Lagrange multipliers $\boldsymbol{U_S}$, the two LP's have the same optimum. The advantage of using NALLP is that the solution is integral for any fixed $\boldsymbol{U_S}$, and moreover, we can compute it by solving a max-weight matching problem. Let us explain why that happens. For a fixed $\boldsymbol{U}$, note that the objective decouples between $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\begin{aligned} &\alpha \sum_{ii'} w_{ii'} x_{ii'} + \frac{\beta}{2} \sum_{ii' \square jj'} y_{ii',jj'} + \sum_{ii \square jj'} u_{ii',jj'}(y_{ii',jj'} - y_{jj',ii'}) \\ &= \alpha \sum_{ii'} w_{ii'} x_{ii'} + \sum_{ii' \square jj'} y_{ii',jj'} \left(\frac{\beta}{2} + u_{ii',jj'} - u_{jj',ii'}\right). \end{aligned}$$

Because $y_{ii',jj'} \leq x_{ii'}$, the optimum is

$$y_{ii',jj'} = \begin{cases} 0 & \frac{\beta}{2} + u_{ii',jj'} - u_{jj',ii'} < 0 \\ x_{ii'} & \text{otherwise.} \end{cases}$$

Therefore, let

$$\bar{w}_{ii'} = \alpha w_{ii'} + \sum_{ii' \square jj'} \max\{0, \frac{\beta}{2} + u_{ii',jj'} - u_{jj',ii'}\}.$$

Then, the solution of NALLP can be found by solving the following max-weight-matching problem:

$$\begin{aligned} &\underset{\boldsymbol{x}}{\text{maximize}} \ \bar{\boldsymbol{w}}^T \boldsymbol{x} \\ &\text{subject to} \ \boldsymbol{C}\boldsymbol{x} \leq \mathbf{1}_{n+m}, \qquad x_{ii'} \in \{0,1\}. \end{aligned}$$

Thus, for any fixed Lagrange multipliers, we can solve NALLP as a single max-weight matching problem. In effect, we have grouped the objective function of NALLP into pieces where $y_{ii',jj'}$ is completely determined by $x_{ii'}$. Additionally, note that we if $u_{ii',jj'} - u_{jj',ii'} = 0$, then we get an especially simple mean of upper-bounding the overlap with a single max-weight matching.

While these relaxations give upper bounds on the objective, there is often a large gap between the upper bound and the integer solution. Frieze and Yadegar [1983] and Adams and Johnson [1994] propose tightened LPs for the quadratic assignment problem. Klau's algorithm adapts these improvements to the network alignment problem. Notice that in both NAILP and NARLP

$$\sum_j y_{ii',jj'} \leq \sum_j x_{jj'} \leq 1, \qquad \sum_{j'} y_{ii',jj'} \leq \sum_{j'} x_{jj'} \leq 1. \tag{2}$$

for any fixed ii'. This means that row ii' of $\boldsymbol{Y_S}$ (denoted by $\boldsymbol{Y_S}[ii',:]$) should satisfy the matching constraint $\boldsymbol{C}\big(\boldsymbol{Y_S}[ii',:]\big)^T \leq \mathbf{1}_{n+m}$. However, when the symmetry constraints are removed, the inequalities (2) may be violated. The tightened LP re-adds these constraints:

$$\begin{aligned} &\underset{\boldsymbol{x},\boldsymbol{y}}{\text{maximize}} \ \alpha \boldsymbol{w}^T \boldsymbol{x} + \tfrac{\beta}{2} \boldsymbol{S} \bullet \boldsymbol{Y_S} + \boldsymbol{U_S} \bullet (\boldsymbol{Y_S} - \boldsymbol{Y_S}^T) \\ &\text{subject to} \ \boldsymbol{C}\boldsymbol{x} \leq \mathbf{1}_{n+m}, \qquad x_{ii'} \in [0,1], \\ &\qquad\qquad y_{ii',jj'} \leq x_{ii'} \qquad \text{for all } ii' \square jj', \\ &\qquad\qquad \boldsymbol{C}\big(\boldsymbol{Y_S}[ii',:]\big)^T \leq \mathbf{1}_{n+m} \qquad \text{for all } ii' \end{aligned} \tag{NATLP}$$

This tighter LP still can be solved using a MWM algorithm. Like in NALLP, $y_{ii',jj'}$ can be grouped by ii'. Now the term

$$\max_{\boldsymbol{y}} \sum_{ii' \square jj'} y_{ii',jj'} (\frac{\beta}{2} + u_{ii',jj'} - u_{jj',ii'})$$

equals $x_{ii'}$ times the solution of a *small* MWM problem

$$\begin{aligned} &\underset{\boldsymbol{Y_S}[ii',:]}{\text{maximize}} \ \boldsymbol{Y_S}[ii',:] \left[\tfrac{\beta}{2} \mathbf{1}^T_{|E_L|} + \boldsymbol{U_S}[ii',:] - \boldsymbol{U_S}^T[ii',:] \right]^T \\ &\text{subject to} \ \boldsymbol{C}\big(\boldsymbol{Y_S}[ii',:]\big)^T \leq \mathbf{1}_{n+m}, \qquad y_{ii',jj'} \in \{0,1\} \end{aligned}. \tag{3}$$

Klau's final algorithm is an iterative procedure that uses a subgradient algorithm to optimize $\boldsymbol{U_S}$ (step 8 in the algorithm). Each step of the subgradient method involves solving NATLP for a new $\boldsymbol{U_S}$. These subproblems are solved by solving $|\boldsymbol{E}_L|$ small max-weight matching problems for (3) and then a single large max-weight matching

problem for NATLP. Because each iteration of this algorithm calls many max-weight matching functions, we call this algorithm the *matching relaxation* or MR for short.

To state the algorithm compactly, we need a small bit of new notation. First, let $a \leq b$. Define

$$\underset{a,b}{\text{bound}}\, z \equiv \min(b, \max(a, z)) = \begin{cases} a & z < a \\ z & a \leq z \leq b \\ b & z > b \end{cases}$$

and let both $\text{bound}_{a,b}\, \boldsymbol{x}$ and $\text{bound}_{a,b}\, \boldsymbol{A}$ be defined element-wise. Also define

$$\boldsymbol{d}, \boldsymbol{S_L} = \text{maxrowmatch}(\boldsymbol{S}, L),$$

where each entry ii' in $\boldsymbol{d}$ is the result of a MWM on all the other edges in row ii' of $\boldsymbol{S}$, with weights from the corresponding entries of $\boldsymbol{S}$. Written formally, $d_{ii'} = \text{bipartite\_match}(\{jj' : S[ii', jj'] = 1)\})$. The matrix $\boldsymbol{S_L}$ has a 1 for any edge used in the optimal solution of the bipartite matching problem for a row.

Algorithm NetAlignMR

INPUT $\boldsymbol{S}$, $\boldsymbol{w}$, nonnegative damping parameters $\gamma \leq 1$, n_{iter}, mstep, α, β

$\boldsymbol{U}^{(0)} = 0$
for $k = 1$ to n_{iter}
  $\boldsymbol{d}, \boldsymbol{S_L}$ = maxrowmatch$((\beta/2)\boldsymbol{S} + \boldsymbol{U} - \boldsymbol{U}^T, L)$
  $\bar{\boldsymbol{w}}^{(k)} = \alpha \boldsymbol{w} + \boldsymbol{d}$
  $\boldsymbol{x}^{(k)}$ = bipartite_match$(L, \bar{\boldsymbol{w}}^{(k)})$
  $\text{obj}^{(k)} = \alpha {\boldsymbol{x}^{(k)}}^T \boldsymbol{w} + \beta/2 {\boldsymbol{x}^{(k)}}^T \boldsymbol{S} \boldsymbol{x}^{(k)}$
  $\text{upper}^{(k)} = {\bar{\boldsymbol{w}}^{(k)}}^T \boldsymbol{x}^{(k)}$
  $\boldsymbol{F} = \boldsymbol{U}^{(k-1)} - \gamma \boldsymbol{X}^{(k)} \textbf{triu}(\boldsymbol{S_L}) + \gamma \textbf{tril}(\boldsymbol{S_L})^T \boldsymbol{X}^{(k)}$
  $\boldsymbol{U}^{(k)} = \underset{-0.5,0.5}{\text{bound}}\, \boldsymbol{F}$
  if $\text{upper}^{(k)}$ has not changed in mstep iterations,
    set $\gamma = \gamma/2$
  end
end
return $\boldsymbol{x}^{(k)}$ with the largest value of $\text{obj}^{(k)}$

Here $\textbf{triu}(\boldsymbol{S_L})$ ($\textbf{tril}(\boldsymbol{S_L})$) represents the upper (lower) triangular part of $\boldsymbol{S_L}$. We reduce the subgradient step-length γ on a schedule determined by the change in the upper-bound.

5. OUR RESULT: TWO MESSAGE-PASSING ALGORITHMS

In this section, we introduce two message-passing algorithms for network alignment. Message passing has been remarkably successful in coding theory [Gallager 1963], artificial intelligence [Pearl 1988], solving constraint satisfaction problems [Mezard and Zecchina 2002], structural biology [Yanover and Weiss 2002], computer vision [Tappen and Freemand 2003], data clustering [Frey and Dueck 2007], and compressed sensing [Donoho et al. 2009].

More precisely, we will use a Belief Propagation (BP) approach. In general, BP works by iteratively making local and greedy decisions. Decisions are updated by passing messages between neighboring entities (nodes of the graph).

In what follows, we provide a quick overview of related BP approaches for the matching problems, next we derive a BP-based algorithm that passes messages along the edges of graph L and also among the squares (Section 5.3). These messages have an intuitive representation, which we present in Section 5.4. Next, we state a matrix-based version of the same algorithm (Section 5.5) in order to elucidate the data organization and computation. Finally, we conclude by developing a more technical message passing algorithm that includes additional constraints from NATLP (Section 5.6).

5.1. Related Work on BP and Graph Matching

BP approaches have been shown to correctly find the optimum solution for a variety of optimization problems including maximum-weight matching [Bayati et al. 2005, 2007a; Sanghavi et al. 2011], and our algorithm for network alignment problem is inspired [Bayati et al. 2005, 2007a; Sanghavi et al. 2011]. The matching problem studied in Bayati et al. [2005, 2007a] and Sanghavi et al. [2011], is a very special case of the network alignment problem (when $\beta = 0$ in NAQP) that can be solved exactly in polynomial time. The quadratic term that appears when $\beta \neq 0$ is NP hard to maximize and requires special treatment, which we carry by defining a new factor graph on squares and edges of the graph L. Recently, and independently from our work, Bradde et al. [2010] introduced a completely different BP approach for aligning graphs in biology using a sub-graph isomorphism representation of the problem. Bradde et al. [2010] aims at finding an injective matching $\pi : A \to B$ (assuming A has less nodes than B). In particular, in one version, the factor graph is the dense complete graph on all nodes of A, which is not applicable to large graphs. In their second approach, Bradde et al. [2010] relax the matching constraint and add an extra parameter p and a term pN_π to the cost function where N_π is the number of matched nodes of B that are matched using π. As $p \to \infty$ the number of violations of the matching constraint goes to zero. Although both Bradde et al. [2010] and this work share the use of a BP approach, the factor graph and the algorithms are different.

5.2. A Factor Graph Representation

To use BP, it is standard to define a probability distribution on the space of all matchings in L that assigns the highest probability to the matching that maximizes NAQP. This matching is also called the *maximum a posteriori assignment* (MAP). We begin with this construction.

Let $V_A = \{1, \ldots, n\}$ and $V_B = \{1', \ldots, m'\}$. For any square formed by the two edges ii' and jj' of E_L, we create a new vertex $ii'jj'$, and denote the set of all such vertices by V_S, i.e.

$$V_S = \left\{ ii'jj' \mid ii' \text{ and } jj' \text{ form a square} \right\}.$$

Now, we assign a binary variable $x_{ii'}$ to each edge $ii' \in E_L$ and a binary variable $x_{ii'jj'}$ for each square $ii'jj' \in V_S$. We also use notation $\boldsymbol{x}_D$ for any subset $D \subset E_L \cup V_S$ to denote the vector $[x_d]_{d \in D}$. The set of neighbors of a node v in a graph G is denoted by ∂v.

Next, we define a new graph (factor graph) that has the following two types of nodes:

(i) *Variable Nodes.* $|E_L| + |V_S|$ nodes, one for each element of E_L and V_S. The binary variables assigned to these nodes are denoted by $(\boldsymbol{x}_{E_L}, \boldsymbol{x}_S) \in \{0,1\}^{|E_L|+|S|}$.

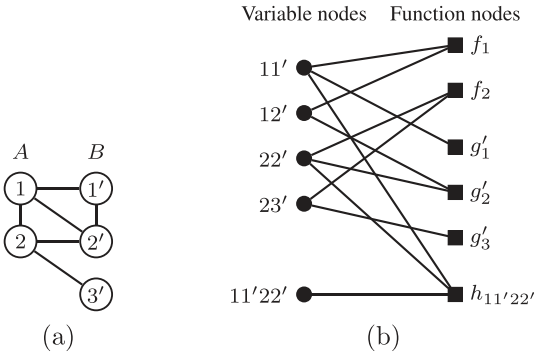


Fig. 3. The graph (b) is the factor-graph representation of the network alignment problem in (a).

(ii) *Function Nodes.* $|V_A| + |V_B| + |S|$ nodes of two types. One type is for enforcing the integer constraints. That is for each vertex $i \in V_A$ ($i' \in V_B$), we define a function node $f_i : \{0,1\}^{|E_L|+|S|} \to \mathbb{R}$ ($g_{i'} : \{0,1\}^{|E_L|+|S|} \to \mathbb{R}$) by:

$$f_i(\boldsymbol{x}_{\partial f_i}) = \begin{cases} 1 & \sum_{ii' \in E_L} x_{ii'} \le 1 \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i$$

$$g_{i'}(\boldsymbol{x}_{\partial g_{i'}}) = \begin{cases} 1 & \sum_{ii' \in E_L} x_{ii'} \le 1 \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i' .$$

The neighbor operation used to define the left-hand vector $\boldsymbol{x}_{\partial f_i}$ is implicitly defined by the set of variables used on the right-hand side of the equation. In words, the function node f_i ($g_{i'}$) enforces the matching constraint at i (i')
Another type of function nodes check the validity of squares. For each square $ii' \square jj'$ define a function node $h_{ii'jj'} : \{0,1\}^{|E_L|+|S|} \to \mathbb{R}$:

$$h_{ii'jj'}(\boldsymbol{x}_{\partial h_{ii'jj'}}) = \begin{cases} 1 & x_{ii'jj'} = x_{ii'} x_{jj'} \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } (ii', jj') \in V_S .$$

In other words, $h_{ii'jj'}$ guarantees that $x_{ii'jj'} = 1$ if and only if $x_{ii'} = x_{jj'} = 1$.

The edges of the factor graph are simply connecting each function node to the variable nodes it acts on. For example each f_i is connected to all variable nodes $ii' \in E_L$ and each $h_{ii'jj'}$ is connected to ii', jj' and $ii'jj'$ in $E_L \cup V_S$. Therefore, the factor graph is bipartite.

Figure 3 shows an example of a graph pair A, B and their factor-graph representation as described previously.

Now define the following probability distribution

$$p(\boldsymbol{x}_L, \boldsymbol{x}_S) = \frac{1}{Z} \left[\prod_{i=1}^{n} f_i(\boldsymbol{x}_{\partial f_i}) \prod_{j=1}^{m} g_j(\boldsymbol{x}_{\partial g_j}) \prod_{ijrs \in V_S} h_{ijrs}(\boldsymbol{x}_{\partial h_{ijrs}}) \right] e^{\alpha \boldsymbol{w}^T \boldsymbol{x}_L + \frac{\beta}{2} \mathbf{1}_{|S|}^T \boldsymbol{x}_S}, \tag{4}$$

where Z is just a normalization term to make $p(\boldsymbol{x}_L, \boldsymbol{x}_S)$ a probability distribution. In particular,

$$Z \equiv \sum_{(\boldsymbol{x}_L, \boldsymbol{x}_S) \in \{0,1\}^{|E_L|+|S|}} \left[\prod_{i=1}^{n} f_i(\boldsymbol{x}_{\partial f_i}) \prod_{j=1}^{m} g_j(\boldsymbol{x}_{\partial g_j}) \prod_{ijrs \in V_S} h_{ijrs}(\boldsymbol{x}_{\partial h_{ijrs}}) \right] e^{\alpha \boldsymbol{w}^T \boldsymbol{x}_L + \frac{\beta}{2} \mathbf{1}_{|S|}^T \boldsymbol{x}_S} .$$

Algorithm NetAlignMP

INPUT α, β, the set of squares V_S, and the weighted bipartite graph $L = (V_A \cup V_B, E_L)$, and a damping parameter γ.

(1) At times $t = 0, 1, \ldots$, each edge ii' sends two messages of the form $m^{(t)}_{ii' \to f_i}$ and $m^{(t)}_{ii' \to g_{i'}}$ and also sends one message of the form $m^{(t)}_{ii' \to h_{ii'jj'}}$ for any square $ii' \square jj'$.

(2) Initialize messages to 0.

(3) For $t \geq 1$, the messages in iteration t are obtained from the messages in iteration $t-1$. In particular for all $ii' \in E_L$

$$m^{(t)}_{ii' \to f_i} = \alpha w_{ii'} - \left(\max_{k \neq i} \left[m^{(t-1)}_{ki' \to g_{i'}} \right] \right)_+ + \sum_{jj' : ii' \square jj'} [(\frac{\beta}{2} + m^{(t-1)}_{jj' \to h_{ii'jj'}})_+ - (m^{(t-1)}_{jj' \to h_{ii'jj'}})_+]. \quad (5)$$

Here, notation $(x)_+$ represents $\max(0, x)$. The update rule for $m^{(t)}_{ii' \to g_{i'}}$ is similar to the update rule for $m^{(t)}_{ii' \to f_i}$ and

$$\begin{aligned} m^{(t)}_{ii' \to h_{ii'jj'}} = \alpha w_{ii'} &+ \sum_{ii'kk' \neq ii'jj'} [(\frac{\beta}{2} + m^{(t-1)}_{kk' \to h_{ii'kk'}})_+ - (m^{(t-1)}_{kk' \to h_{ii'kk'}})_+] \\ &- \left(\max_{k \neq i} \left[m^{(t-1)}_{ki' \to g_{i'}} \right] \right)_+ - \left(\max_{k' \neq i'} \left[m^{(t-1)}_{ik' \to f_i} \right] \right)_+ . \end{aligned} \quad (6)$$

(4) Apply damping on the message updates. (See possibilities in Section 5.4.1.)

(5) Round the solution (see possibilites in Section 6) and compute the objective function on the rounded messages.

(6) Repeat (3)-(5) for a fixed number of iterations unless the messages stop changing.

OUTPUT the rounded solution with the best objective value.

Note that, there is a 1–1 correspondence between the feasible solutions of NAQP and support of the probability distribution (4). The following lemma formalizes this observation.

LEMMA 5.1. *For any $(\boldsymbol{x}_L, \boldsymbol{x}_S) \in \{0,1\}^{|E_L|+|V_S|}$ with nonzero probability, the vector $\boldsymbol{x}_L$ satisfies the constraints of the integer program NAQP. Conversely, any feasible solution $\boldsymbol{x}_L$ to NAQP has a unique counterpart $(\boldsymbol{x}_L, \boldsymbol{x}_S)$ with nonzero probability $p(\boldsymbol{x}_L, \boldsymbol{x}_S) = e^{\alpha \boldsymbol{w}^T \boldsymbol{x} + (\beta/2) \mathbf{1}^T_{|S|} \boldsymbol{x}_S}$.*

PROOF. Any $(\boldsymbol{x}_L, \boldsymbol{x}_S) \in \{0,1\}^{|E_L|+|V_S|}$ with nonzero probability should satisfy the conditions dictated by function nodes f, g, h, which translates to $\boldsymbol{x}_L, \boldsymbol{x}_S$ being a feasible solution to NAQP. Conversely, for any feasible solution to NAQP the values of function nodes f, g, h are equal to 1 and hence the probability is nonzero. □

Moreover, any pair with maximum probability is an optimum solution to NAQP.

LEMMA 5.2. *The vector $(\boldsymbol{x}^*_L, \boldsymbol{x}^*_S)$ is equal to $\arg\max_{\boldsymbol{x}_L, \boldsymbol{x}_S} p(\boldsymbol{x}_L, \boldsymbol{x}_S)$ if and only if $\boldsymbol{x}^*_L$ is the optimum solution to NAQP and $\boldsymbol{x}^*_S$ is the vector of squares generated by it.*

PROOF. Proof immediately follows from Lemma 5.1. □

Using Lemma 5.2, it is known that a variant of BP algorithm (*max-product* or *min-sum*) can be used to find an approximate solution to *NAQP* [Mezard and Montanari 2009]. In this article, we use the notion BP to refer to this variant.

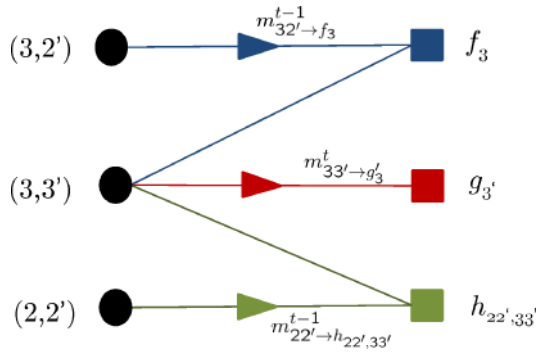


Fig. 4. Dependence of $m^t_{33'\to g_{3'}}$ to messages of time $t-1$ for the base example from Figure 2.

5.3. The Message Passing Algorithm

The standard BP messages for finding the optimum solution $\arg\max_{\boldsymbol{x}_L,\boldsymbol{x}_S} p(\boldsymbol{x}_L,\boldsymbol{x}_S)$ are vectors of numbers. However, for our problem, we show that the information contained in these vector messages can be compressed to a real number. Therefore, we can obtain a simple algorithm with a smaller running time that will be presented next. For completeness, we provide the derivation of this simplified version from the standard BP in Appendix 9. However, in Section 5.4, we provide a more intuitive description of the algorithm.

5.4. The Intuition Behind NetAlignMP

NetAlignMP exploits the fact that the constraints of NAQP are local. Suppose each edge of the graph L is an agent and each agent can talk to its neighbors. First, observe that, together, the agents can verify the feasibility of any solution to NAQP. The next step is to note that they can also calculate the merit of each solution ($\alpha \boldsymbol{w}^T\boldsymbol{x}+\beta/2\boldsymbol{x}^T\boldsymbol{S}\boldsymbol{x}$) locally.

Based on this intuition, each agent should communicate to the neighboring agents to control the matching constraints. Messages of the type $m^{(t)}_{ii'\to f_i}$ and $m^{(t)}_{ii'\to g_{i'}}$ serve this purpose. They also contain the information about the weights of the edges (term $\alpha\boldsymbol{w}^T\boldsymbol{x}$ in the cost function). Similarly, any two agents that form a square should communicate, so that we can calculate the term $\beta\boldsymbol{x}^T\boldsymbol{S}\boldsymbol{x}$ in the cost function. This information is passed by the messages of type $m^{(t)}_{ii'\to h_{ii'jj'}}$.

From a slightly different perspective, our algorithm can be seen as a form of dynamic programming generalized from trees to general graphs. In fact, it is instructive to consider the special case in which the factor graph (explained in Section 5.2) is indeed a forest. In that case, removing an edge (or agent) splits the tree component into two pieces. This means that the optimization problem NAQP could be solved independently on each component. The message of the form $m^{(t)}_{ii'\to f_i}$ carries the information about the component that contains i'. Figure 4 shows this type of message update. It also contains the information about all squares that contain ii'. Ideally, the message $m^{(t)}_{ii'\to f_i}$ should show the amount of change in the cost function (excluding the connected component containing i) by participation of the edge ii' in a solution. Similarly, each message of the type $m^{(t)}_{jj'\to h_{ii'jj'}}$ should be the change in the cost function by participation of jj' (restricted to the component the edges jj').

Algorithm NetAlignMP (Matrix-based)

INPUT $\boldsymbol{C} = \left(\boldsymbol{C}_A^T \ \boldsymbol{C}_B^T\right)^T$, $\boldsymbol{S}$, $\boldsymbol{w}$, damping parameter γ, $\mathrm{n_{iter}}$, damping_type

$\boldsymbol{y}^{(0)} = 0, \boldsymbol{z}^{(0)} = 0, \boldsymbol{S}^{(0)} = 0$
for t = 1 to $\mathrm{n_{iter}}$
  $\boldsymbol{F} = \mathrm{bound}_{0,\frac{\beta}{2}}(\boldsymbol{S}^{(t-1)^T} + \frac{\beta}{2}\boldsymbol{S})$
  $\boldsymbol{d}^{(t)} = \boldsymbol{F} \cdot \boldsymbol{e}$
  $\boldsymbol{y}^{(t)} = \alpha\boldsymbol{w} - \mathrm{bound}_{0,\infty}[\,(\boldsymbol{C}_A^T\boldsymbol{C}_A - \boldsymbol{I}) \boxdot \boldsymbol{z}^{(t-1)}] + \boldsymbol{d}^{(t)}$
  $\boldsymbol{z}^{(t)} = \alpha\boldsymbol{w} - \mathrm{bound}_{0,\infty}[\,(\boldsymbol{C}_B^T\boldsymbol{C}_B - \boldsymbol{I}) \boxdot \boldsymbol{y}^{(t-1)}] + \boldsymbol{d}^{(t)}$
  $\boldsymbol{S}^{(t)} = (\boldsymbol{Y}^{(t)} + \boldsymbol{Z}^{(t)} - \alpha\boldsymbol{W} - \boldsymbol{D}^{(t)}) \cdot \boldsymbol{S} - \boldsymbol{F}$
  if damping_type is 1
    $(\boldsymbol{y}^{(t)}, \boldsymbol{z}^{(t)}, \boldsymbol{S}^{(t)}) \leftarrow \gamma^t(\boldsymbol{y}^{(t)}, \boldsymbol{z}^{(t)}, \boldsymbol{S}^{(t)}) + (1-\gamma^t)(\boldsymbol{y}^{(t-1)}, \boldsymbol{z}^{(t-1)}, \boldsymbol{S}^{(t-1)})$
  else if damping_type is 2
    $\boldsymbol{p} = \boldsymbol{y}^{(t-1)} + \boldsymbol{z}^{(t-1)} - \alpha\boldsymbol{w} + \boldsymbol{d}^{(t-1)}$
    $(\boldsymbol{y}^{(t)}, \boldsymbol{z}^{(t)}, \boldsymbol{S}^{(t)}) \leftarrow (\boldsymbol{y}^{(t)}, \boldsymbol{z}^{(t)}, \boldsymbol{S}^{(t)}) + (1-\gamma^t)(\boldsymbol{p}, \boldsymbol{p}, \boldsymbol{S}^{(t-1)} + \boldsymbol{S}^{(t-1)^T} - \beta\boldsymbol{S})$
  else if damping_type is 3
    $\boldsymbol{p} = \boldsymbol{y}^{(t-1)} + \boldsymbol{z}^{(t-1)} - \alpha\boldsymbol{w} + \boldsymbol{d}^{(t-1)}$
    $(\boldsymbol{y}^{(t)}, \boldsymbol{z}^{(t)}, \boldsymbol{S}^{(t)}) \leftarrow \gamma^t(\boldsymbol{y}^{(t)}, \boldsymbol{z}^{(t)}, \boldsymbol{S}^{(t)}) + (1-\gamma^t)(\boldsymbol{p}, \boldsymbol{p}, \boldsymbol{S}^{(t-1)} + \boldsymbol{S}^{(t-1)^T} - \beta\boldsymbol{S})$
  end
  $\boldsymbol{x}^{(t)}$ = round_messages$(\boldsymbol{y}^{(t)}, \boldsymbol{z}^{(t)}, \boldsymbol{S}^{(t)})$
  obj$^{(t)}$ = objective$(\boldsymbol{x}^{(t)})$
end
return $\boldsymbol{x}^{(t)}$ with the largest value of obj$^{(t)}$

Now we give a rough derivation of Eq. (5) using this discussion. If ii' is present in the solution, then $\alpha w_{ii'}$ is added to the cost function. But none of the edges ki' ($k \neq i$) can now be in the matching. Thus, we should subtract their maximum contribution $\left(\max_{k\neq i}\left[m^{(t-1)}_{ki'\to g_{i'}}\right]\right)_+$. This explains the first two terms in the right hand side of Eq. (5). Moreover, we should add the number of squares that will be added by this edge. For each square $ii'jj'$ if the edge jj' is not present in the matching, then nothing is added. Otherwise, a $\beta/2$ plus the term $m^{(t)}_{jj'\to h_{ii'jj'}}$ should be added. This roughly explains the addition of the third term in (5). A similar explanation justifies (6) as well.

5.4.1. Convergence of NetAlignMP. We now elaborate on step (4) of NetAlignMP. Ideally, at the end of iteration t, each vertex i selects the edge ii' that sends the maximum incoming message $m^{(t)}_{ii'\to f_i}$ to it, and we denote the resulting matching by $M(t)$. We'd like to terminate the iteration when $M(t)$ converges. Unfortunately, picking edges with this rule does not always produce a matching, and also $M(t)$ may not converge. We discuss better approaches to picking a matching from the messages in Section 6. When $M(t)$ does not converge, it often oscillates between a few states. Therefore, we could terminate the algorithm when such an oscillation is observed, and use the current messages to find a matching using the recipe in Section 6. Another approach for resolving the oscillation is to use a damping factor $\gamma \in [0,1]$ [Braunstein and Zecchina 2006; Frey and Dueck 2007; Murphy et al. 1999]. Let $\mathbf{n}(t)$ be the vector of all messages at time t. That is $\mathbf{n}(t)$ is a fixed ordering of all messages at time t. Then, the

update Eqs. (5)–(6) can be rewritten as $\mathfrak{n}(t) = F(\mathfrak{n}(t-1))$ where F is an operator that is uniquely defined by Eqs. (5)–(6). Now, one can consider a new operator G defined by $G(\mathfrak{n}(t)) = (1-\gamma^t)\mathfrak{n}(t-1) + \gamma^t F(\mathfrak{n}(t-1))$ and update the messages using G instead of F. The new update equations will converge for $\gamma < 1$. We make the damping explicit in the matrix version of this algorithm in Section 5.5.

5.5. A Matrix Formulation

We now restate the NetAlignMP algorithm (from Section 5.3) using matrix notation. This helps clarify issues of data organization and computation. To begin, we again need another bit of notation. For $\boldsymbol{A} \in \mathbb{R}^{m,n}$ and $\boldsymbol{x} \in \mathbb{R}^n$, define

$$\boldsymbol{A} \boxdot \boldsymbol{x} \equiv \begin{pmatrix} \max_j a_{1,j} x_j \\ \max_j a_{2,j} x_j \\ \vdots \\ \max_j a_{m,j} x_j \end{pmatrix}.$$

This operator is just the regular matrix-vector product but with the summation $(\boldsymbol{Ax})_i = \sum_j a_{i,j} x_j$ replaced by maximization. (This is the matrix-vector product from the max-product algebra and is related to the max-plus algebra via logarithm/exponential transforms.) We also need to split the constraint matrix $\boldsymbol{C}$ into $\begin{pmatrix} \boldsymbol{C}_A^T & \boldsymbol{C}_B^T \end{pmatrix}^T$ corresponding to the matching constraints from graph $A \to B$ and graph $B \to A$, respectively.

5.6. Improved NetAlignMP

Recall that Klau's algorithm [Klau 2009] is obtained by tightening the linear program NALLP using combinatorial properties of the problem. Similarly, we can modify the factor graph representation of Section 5.2 to improve the solutions of NetAlignMP at the expense of increasing the running time. Here is a rough explanation of this modification. For each variable node ii' add function nodes $d_{ii',j}$ and $d_{ii,j'}$ for all jj' with $ii' \square jj'$. These function nodes are defined by:

$$d_{ii',j}([x_{jk'}]_{k':jk' \square ii'}) = \begin{cases} 1 & \sum_{k':jk' \square ii'} x_{jk'} \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

$$d_{ii',j'}([x_{kj'}]_{k:kj' \square ii'}) = \begin{cases} 1 & \sum_{k:kj' \square ii'} x_{kj'} \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$

After a similarly-lengthy-but-straightforward derivation like in Appendix 9, we arrive at the following extension of NetAlignMP.

6. ROUNDING STRATEGIES

All algorithms, except for Klau's matching relaxation, introduced so far rely on formulating the problem as a mathematical program, with the integer constraint relaxed. As a consequence, the computed solution is fractional for most instances. For IsoRank and SpaIsoRank, the fractional values are associated with edges in L and for NetAlignMP and NetAlignMP++, the values are on both edges and squares. The last step of each algorithm is to round this fractional solution to an integral solution, that is, a

matching. There are many ways of rounding, and as always, the best rounding scheme depends on the actual problem and the type of relaxation.

Algorithm NetAlignMP++

INPUT α, β, the set of squares V_S, and the weighted bipartite graph $L = (V_A \cup V_B, E_L)$, and a damping parameter γ.

(1) At times $t = 0, 1, \ldots$, each edge ii' sends two messages of the form $m^{(t)}_{ii' \to f_i}$ and $m^{(t)}_{ii' \to g_{i'}}$ and also sends one message of the form $m^{(t)}_{ii' \to h_{ii'jj'}}$ for any square $ii'jj' \in V_S$. Each square $ii'jj'$ sends a message of the form $m^{(t)}_{ii'jj' \to h_{ii'jj'}}$ and four messages of the type $m^{(t)}_{ii'jj' \to d_{ii'j'}}$ to $d_{ii'j}, d_{ii'j'}, d_{jj'i}$ and $d_{jj'i'}$.

(2) Messages are initialized by an arbitrary number (let us say 0).

(3) For $t \geq 1$, the messages in iteration t are obtained from the messages in iteration $t-1$ recursively. In particular for all $ii' \in E_L$

$$m^{(t)}_{ii' \to f_i} = \alpha w_{ii'} - \left(\max_{k \neq i} \left[m^{(t-1)}_{ki' \to g_{i'}} \right] \right)_+ + \sum_{ii'jj' \in V_S} \left[\left(m^{(t-1)}_{ii'jj' \to h_{ii'jj'}} + m^{(t-1)}_{jj' \to h_{ii'jj'}} \right)_+ - \left(m^{(t-1)}_{jj' \to h_{ii'jj'}} \right)_+ \right]. \quad (7)$$

The update rule for $m^{(t)}_{ii' \to g'_i}$ is similar to the update rule for $m^{(t)}_{ii' \to f_i}$ and

$$m^{(t)}_{ii' \to h_{ii'jj'}} = \alpha w_{ii'} + \sum_{\substack{kk' \neq jj' \\ ii'jj' \in V_S}} \left[\left(m^{(t-1)}_{ii'kk' \to h_{ii'kk'}} + m^{(t-1)}_{kk' \to h_{ii'kk'}} \right)_+ - (m^{(t-1)}_{kk' \to h_{ii'kk'}})_+ \right] - \left(\max_{k \neq i} \left[m^{(t-1)}_{ki' \to g_{i'}} \right] \right)_+ - \left(\max_{k' \neq i'} \left[m^{(t-1)}_{ik' \to f_i} \right] \right)_+ . \quad (8)$$

and

$$m^{(t)}_{ii'jj' \to h_{ii'jj'}} = \frac{\beta}{2} - \left(\max_{k \neq i} \left[m^{(t-1)}_{ki'jj' \to d_{jj'i'}} \right] \right)_+ - \left(\max_{k' \neq i'} \left[m^{(t-1)}_{ik'jj' \to d_{jj'i}} \right] \right)_+ - \left(\max_{k \neq j} \left[m^{(t-1)}_{ii'kj' \to d_{ii'j'}} \right] \right)_+ - \left(\max_{k' \neq j'} \left[m^{(t-1)}_{ii'jk' \to d_{ii'j}} \right] \right)_+ \quad (9)$$

and

$$m^{(t)}_{ii'jj' \to d_{ii'j}} = \frac{\beta}{2} - \left(\max_{k \neq i} \left[m^{(t-1)}_{ki'jj' \to d_{jj'i'}} \right] \right)_+ - \left(\max_{k' \neq i'} \left[m^{(t-1)}_{ik'jj' \to d_{jj'i}} \right] \right)_+ - \left(\max_{k \neq j} \left[m^{(t-1)}_{ii'kj' \to d_{ii'j'}} \right] \right)_+ + \min \left(m^{(t-1)}_{ii' \to h_{ii'jj'}} + m^{(t-1)}_{jj' \to h_{ii'jj'}},\ m^{(t-1)}_{ii' \to h_{ii'jj'}},\ m^{(t-1)}_{jj' \to h_{ii'jj'}} \right) \quad (10)$$

Equations for $m^{(t)}_{ii'jj' \to d_{ii'j'}}, m^{(t)}_{ii'jj' \to d_{jj'i}}$ and $m^{(t)}_{ii'jj' \to d_{jj'i'}}$ are similar to (10).

(4) Damp the messages using one of the schemes from NetAlignMP.

(5) Round the messages to an integer solution (see possibilities in Section 6) and compute the objective function on the rounded messages

OUTPUT the rounded solution with the best objective value.

The primary type of rounding used is based on using the fractional solution or the BP messages to construct a max-weight matching problem. Solving it produces a solution that then obeys the matching constraints. Specifically, we utilize the function:

Algorithm round_messages

INPUT messages from A to B $\boldsymbol{y}^{(t)}$, messages from B to A $\boldsymbol{z}^{(t)}$, messages on the squares $\boldsymbol{S}^{(t)}$

1 $\boldsymbol{x}_A^{(k)} = \text{bipartite\_match}(\text{L}, \boldsymbol{y}^{(k)})$
2 $\text{obj}_A^{(k)} = \text{objective}(\boldsymbol{x}_A^{(k)})$
3 $\boldsymbol{x}_B^{(k)} = \text{bipartite\_match}(\text{L}, \boldsymbol{z}^{(k)})$
4 $\text{obj}_B^{(k)} = \text{objective}(\boldsymbol{x}_B^{(k)}\}$
5 **return** $\boldsymbol{x}^{(k)}$ with the highest value of $\text{obj}^{(k)}$

This function rounds both types of messages and returns the best solution. Another alternative is to use a greedy matching scheme, where M starts as an empty matching, and we greedily add edges to M based on the largest values of $\boldsymbol{y}_{ii'}^{(k)}$ or $\boldsymbol{z}_{ii'}^{(k)}$ such that it stays a matching. Though computationally more expensive, MWM rounding yields the best result in most of our experiments. Therefore, results in Section 7 and 8 are all obtained using MWM rounding. For the BP algorithm, greedy rounding using messages on squares—using $\boldsymbol{S}^{(t)}$ above—yields similar performance as the MWM rounding. Note that Klau's algorithm explicitly generates an integer solution by solving a max-weight matching problem on each iteration.

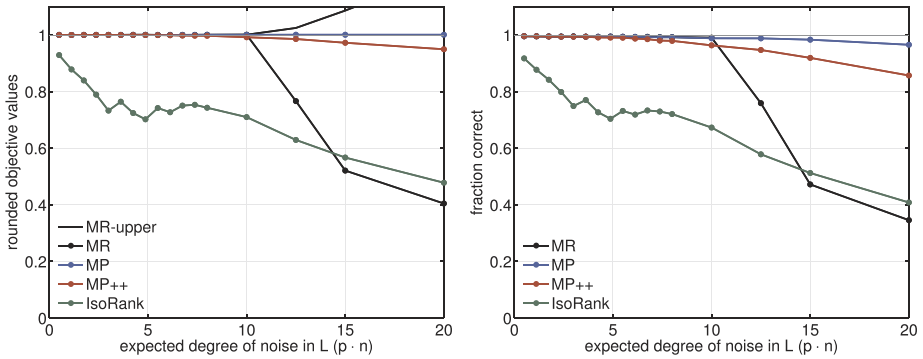
7. SYNTHETIC EXPERIMENTS

We first compare the belief propagation (BP) algorithm to existing algorithms on two synthetic matching problems. The first problem aligns two perturbed grids and the second aligns two perturbed power-law graphs.

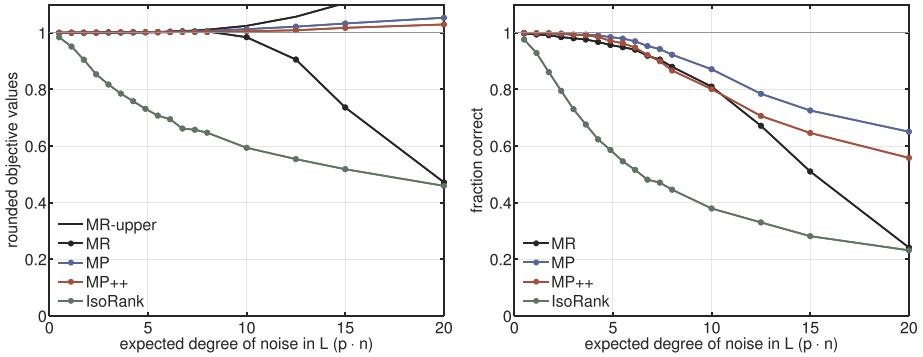
Let A and B be independent realizations of a perturbed $k \times k$ grid. The perturbation is a set of random edges generated with probability $q/d(u,v)^2$ where $d(u,v)$ is the graph distance between u and v. In these problems, the *ideal alignment* is known: match each vertex to its image in the other grid. Note that this ideal alignment does not necessarily maximize the objective function. Now we generate L by matching each grid vertex to its image and then add additional edges to L with probability p. This noise globally corrupts the alignment. We further disturb L by adding random edges within graph distance d of the end points of ideal alignment, sampled with probability proportional to the maximum number of paths. This step locally corrupts the alignment.

For the power-law graph test, we construct a reference graph from a power-law degree sequence with exponent θ and n vertices using the algorithm from Bayati et al. [2007b]. Again, let A and B be independent realizations of the power-law graph perturbed with the same noise as the grid in this article. Generate L in the same manner, but without additional distance based edges.

In our results, we compare all outputs to the reference matching between the graphs A and B. Figure 5 shows the average fraction of the reference matching obtained by each algorithm over 48 trials. The objective function is pure overlap and the dark lines in the figure show the ratio of the algorithm's overlap to the overlap of the reference solution. Each algorithm should be computing a good objective, and thus larger values are better. Indeed, the reference solution may not be the best solution when L is highly corrupted with a large expected degree. When this happens with the power-law graphs, we observe that the BP algorithm finds a matching with a higher overlap and thus the



(a) Grid graph ($k = 20, q = 2, d = 1$). Function values at left and correct matches at right.



(b) Power-law graphs ($\theta = 1.8, n = 400, q = 1$). Function values at left and correct matches at right. function value

Fig. 5. Upper bounds and correct solutions to synthetic problems on grid-graphs (a) and power-law graphs (b). The MR label is for the NetAlignMR algorithm and IsoRank refers to the SpaIsoRank procedure. The x-axis corresponds to expected degree that increases with p, the fraction of global mismatched edges in L, which we measure in the expected degree of the noise. Once the noise is large, the two message passing approaches show the best results. Section 7 for more information.

fraction is larger than 1. Similarly, the light lines show the fraction of correct matches from the the algorithms. These values track the objective values showing that the network alignment objective is a good surrogate for the number of correct matches objective.

When the amount of random noise in L exceeds an expected degree of 10 for the grid graphs and 8 for the power-law graphs, many of the algorithms are no longer able to obtain good solutions. In this regime, the MP and MP++ algorithms performs better than the MR algorithm.

We used the MP and MP++ algorithms with $\alpha = 1, \beta = 2$, the SpaIsoRank algorithm with $\gamma = 0.95$, and the MR algorithm with $\alpha = 0, \beta = 1$ for these experiments. These parameters are natural for the various algorithms. For example, MR requires $\alpha = 0, \beta = 1$ to produce an upper-bound on overlap. In the next section, we study the behavior of the algorithms for a wider variety of parameters.

8. REAL DATASETS

While we saw that the BP algorithm performed well on noisy synthetic problems in the previous section, in this section we investigate alignment problems from

Table II. Properties of the Real-World Test Problems

Problem	$\|V_A\|$	$\|E_A\|$	$\|V_B\|$	$\|E_B\|$	$\|E_L\|$
dmela-scere	9459	25636	5696	31261	34582
Mus M.-Homo S.	3247	2793	9695	32890	15810
lcsh2wiki-small	1919	1565	2000	3904	16952
lcsh2wiki-full	297266	248230	205948	382353	4971629

bioinformatics and ontology matching. For each algorithm, we explore a range of choices for all of the parameter values and summarize the results from the *best* choice in Table III. Note that Klau's algorithm uses two parameters γ and *st* to control the subgradient method.

8.1. Bioinformatics

The alignment of protein-protein interaction (PPI) networks of different species is an important problem in bioinformatics [Singh et al. 2007]. We consider aligning the PPI network of Drosophila melanogaster (fly) and Saccharomyces cerevisiae (yeast), and Homo sapiens (human) and Mus musculus (mouse). These PPI networks are available in several open databases and they are used in Singh et al. [2008] and Klau [2009], respectively. For each problem, we utilize the value of $\boldsymbol{w}$ from the original publication. While the results of the experiment are rich in biological information, we focus solely on the optimization problem.

Figure 6 shows the performance of the four algorithms—NetAlignMP, NetAlignMP++, NetAlignMR, SpaIsoRank—on these two alignments. For each algorithm, we perform a parameter sweep over the following parameters

SpaIsoRank	Damping	$\gamma \in \{0.3, 0.5, 0.85, 0.95\}$
	Rounding	type $\in \{1, 2\}$
NetAlignMP	Objective	$(\alpha, \beta) \in \{(10, 1), (2, 1), (1, 1), (1, 2), (1, 10)\}$
	Damping	$\gamma \in \{0.9, 0.99, 0.995, 0.999\}$
		type $\in \{2, 3\}$
NetAlignMP++	*same as* NetAlignMP	
NetAlignMR	Objective	$(\alpha, \beta) \in \{(10, 1), (2, 1), (1, 1), (1, 2), (1, 10)\}$
	Damping	$\gamma \in \{0.1, 0.4\}$
		mstep $\in \{5, 25, 50\}$

We run SpaIsoRank until convergence, and run the other approaches for a total of 500 iterations. On these instances, we record the best iterate ever generated and plot the overlap and weight of the alignments in the figure.

In both problems NetAlignMP, NetAlignMP++ and NetAlignMR manage to obtain near-optimal solutions. In terms of the largest overlap, our NetAlignMP does the best on the Mus M.-Homo S. alignment, whereas NetAlignMP++ does the best of the dmela-scere alignment. (See Table III for the parameters that produced the best overlap.)

8.2. Ontology

Our original motivation for investigating network alignment is aligning the network of subject headings from the Library of Congress with the categories from Wikipedia [Wikipedia 2007]. Each node in these networks has a small text label, and we use a Lucene search index [Hatcher and Gospodnetic 2004] (Version 2.2.0 from 2007) to quickly find potential matches between nodes based on the text. To score the

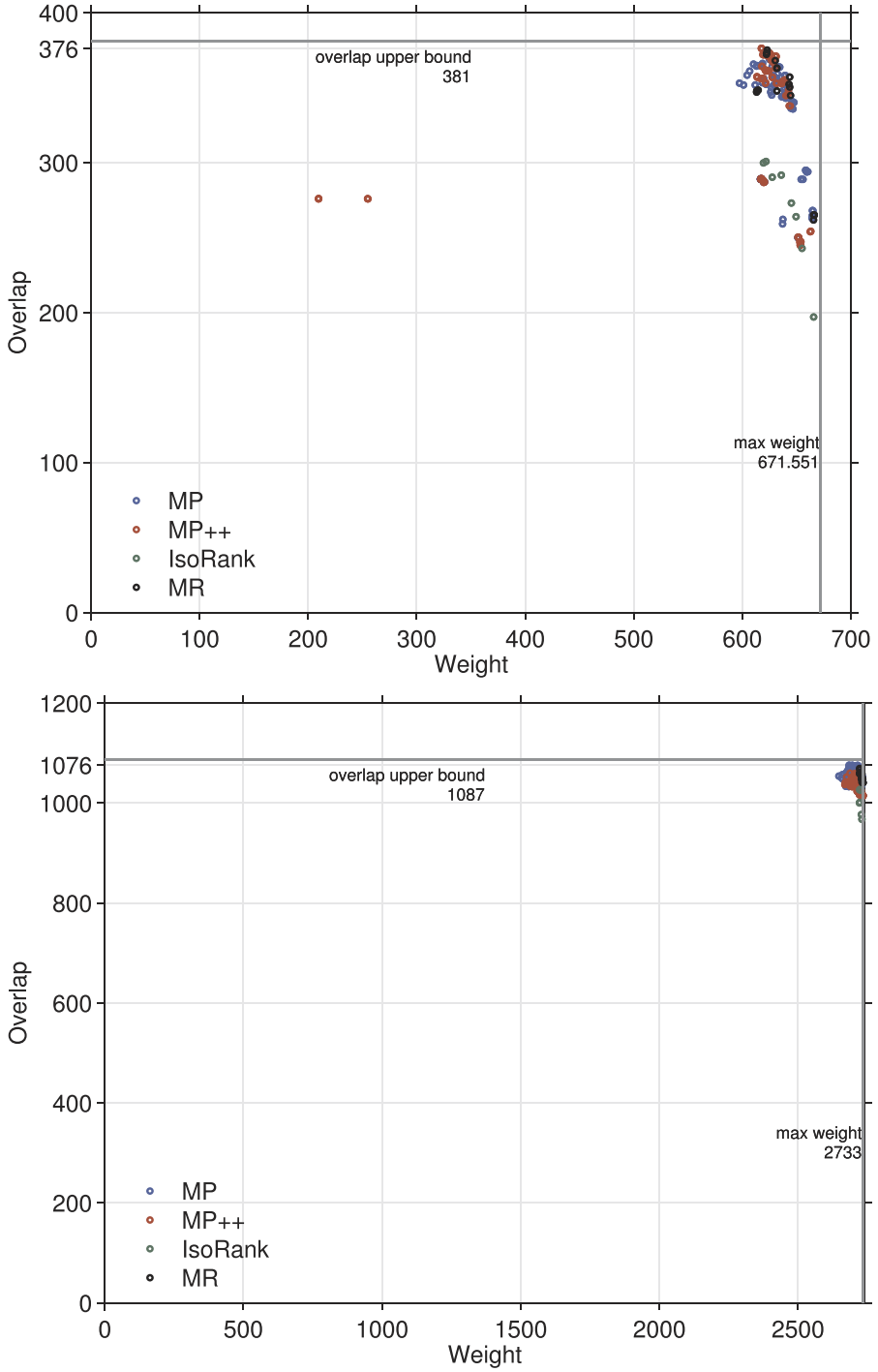


Fig. 6. Results of the three algorithms SpaIsoRank (IsoRank), NetAlignMP, NetAlignMP++, and NetAlignMR (MR) on the Mus. M.-Homo S. alignment (top) and dmela-scere alignment (bottom).

Table III. The Parameters Used to Produce the Results with the Highest Overlap from Figures 6 and 7

Alg.	Data		Overlap	Sol. Time	Total Time	Parameters
MWM	musm-homo	393	36.2%	—	—	
	dmela-scere	135	35.4%	—	—	
	lcsh-small	119	36.8%	—	—	
	lcsh2wiki	2346	13.3%	—	—	
Iso	musm-homo	1027	94.5%	0.0	0.4	$\gamma = 0.50$; r = 2
	dmela-scere	301	79.0%	3.7	10.7	$\gamma = 0.95$; r = 2
	lcsh-small	257	79.6%	0.0	0.7	$\gamma = 0.50$; r = 2
	lcsh2wiki	11732	66.6%	11.7	587.3	$\gamma = 0.95$; r = 2
MP	musm-homo	1076	99.0%	2.6	13.2	$\alpha = 2; \beta = 1; \gamma = 0.995; d = 3$
	dmela-scere	369	96.9%	26.7	34.9	$\alpha = 1; \beta = 2; \gamma = 0.999; d = 3$
	lcsh-small	316	97.8%	7.6	12.6	$\alpha = 1; \beta = 1; \gamma = 0.999; d = 3$
	lcsh2wiki	15974	90.7%	3795.3	4198.4	$\alpha = 1; \beta = 2; \gamma = 0.999; d = 2$
MP++	musm-homo	1062	97.7%	14.4	17.3	$\alpha = 1; \beta = 1; \gamma = 0.999; d = 3$
	dmela-scere	376	98.7%	28.7	33.3	$\alpha = 1; \beta = 10; \gamma = 0.999; d = 3$
	lcsh-small	318	98.5%	11.8	15.2	$\alpha = 1; \beta = 2; \gamma = 0.999; d = 3$
	lcsh2wiki	15771	89.6%	4103.8	4990.2	$\alpha = 1; \beta = 1; \gamma = 0.999; d = 3$
MR	musm-homo	1070	98.4%	12.5	12.6	$\alpha = 1; \beta = 10; \gamma = 0.400; st = 5$
	dmela-scere	375	98.4%	22.7	79.4	$\alpha = 1; \beta = 2; \gamma = 0.400; st = 5$
	lcsh-small	318	98.5%	4.1	16.8	$\alpha = 1; \beta = 2; \gamma = 0.400; st = 5$
	lcsh2wiki	16836	95.6%	4878.2	4988.0	$\alpha = 1; \beta = 2; \gamma = 0.400; st = 5$

We abbreviated lcsh2wiki-small as lcsh-small. The overlap score shows the highest overlap produced by that method on the problem and the percentage of the best upper-bound on the solution objective. All times are reported in seconds, and the Sol. Time column indicates the time taken to compute the best solution whereas the Total Time column indicates the total time for all iterations of the method. (Recall that all methods return the iterate with the best solution, which may not be the final iterate.) We organized the table to indicate the most successful parameter choices.

matches, we use the SoftTF-IDF scoring routine [Cohen et al. 2003]. These scores become the weights in $\boldsymbol{w}$. Our *real* problem is to match the entire graphs. From this problem we extract a small instance that should capture the most important nodes in the problem. (Node importance is either reference count (subject headings) or PageRank (Wikipedia categories).) The results are shown in Figure 7.

We repeated the parameter sweep from the previous section on these two problems as well. The best algorithm on these two problems is NetAlignMR, with NetAlignMP and NetAlignMP++ alternating for second place. In lcsh2wiki-small, the upper bound computed by NetAlignMR is 323. NetAlignMP achieves a lower bound of 318 and NetAlignMR achieves 321. In lcsh2wiki, we compute an upper bound of 17608 using a linear programming solver on NATLP with the full symmetry constraints instead of the Lagrange multipliers. Though not shown in the figure, NetAlignMP obtains a lower bound of 16204 with $\gamma = 0.9995$, $\alpha = 0.2$ and $\beta = 1$.

In all our real datasets L is quite sparse, making NetAlignMR more favorable. Still, NetAlignMP is closely following and has an advantage on running time – see the summary in Table III for information about runtime.

8.3. Multilingual Ontologies

For a final test, we evaluate automatically aligning two large networks where a correct alignment exists. The networks are the Library of Congress Subject Headings and its French analogue, Rameau. Both are similar ontologies and we expect a non-trivial

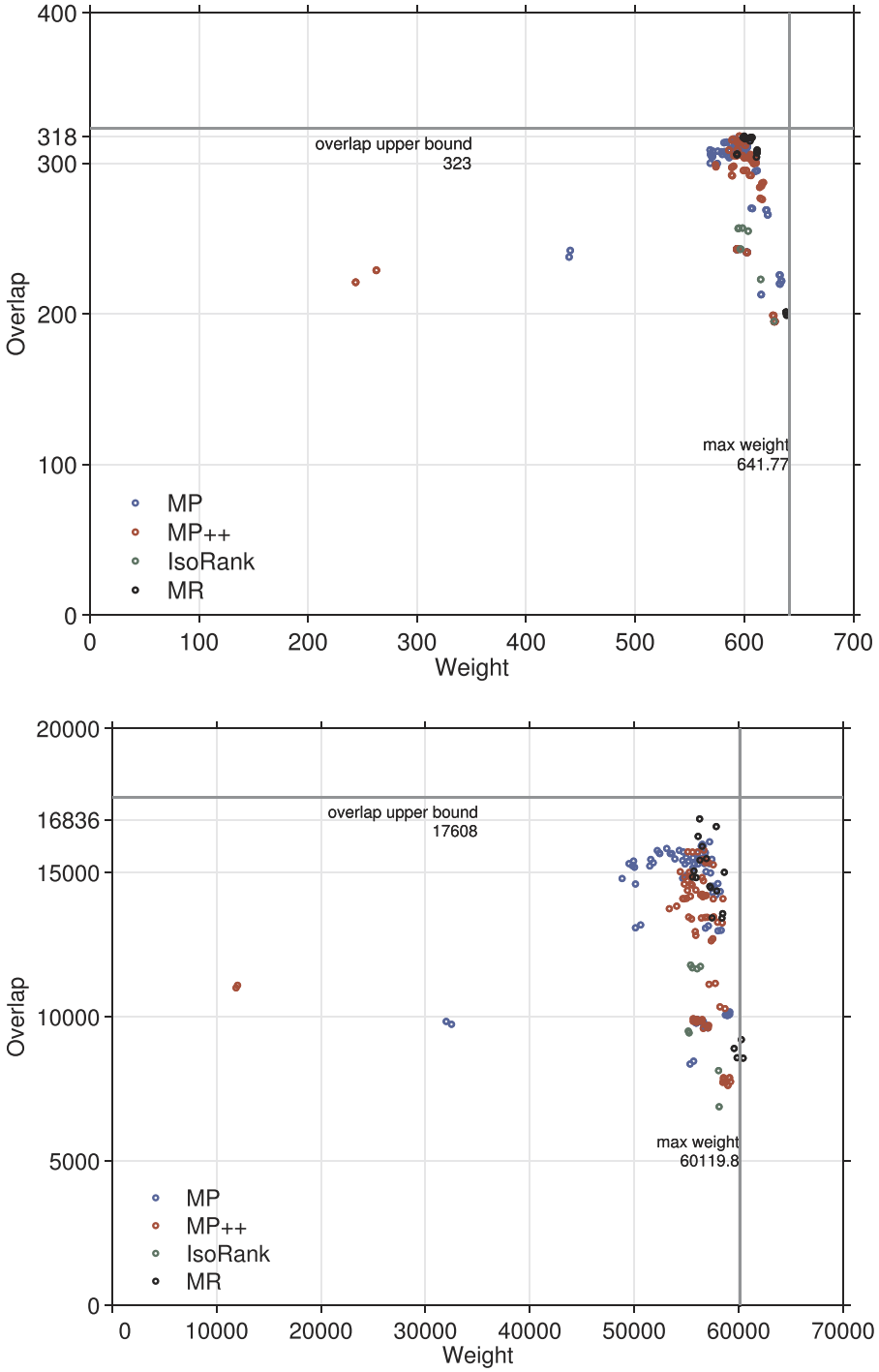


Fig. 7. Results of the four algorithms SpaIsoRank (IsoRank), NetAlignMP, NetAlignMP++, NetAlignMR (MR) on lcsh2wiki-small (top) and lcsh2wiki (bottom).

Table IV. The Alignment Results for LCSH and Rameau

Obj.	Alg.	Weight	Overlap	Time (s)	Correct	Rec.	Prec.	Triangles
	Sol.	36332.42	39847	—	57645	100%	100%	2073
	MWM	93279.0	16990	29.6	29098	50.5%	23.3%	350
$\alpha = 1, \beta = 1$	MP	84622.0	46400	23522.0	32585	56.5%	27.6%	1515
	MP++	85810.1	46942	27115.6	32857	57.0%	27.4%	1548
	MR	87588.6	48367	33366.9	33225	57.6%	27.0%	1617
$\alpha = 1, \beta = 2$	MP	81752.6	46569	23427.1	31724	55.0%	27.6%	1483
	MP++	84615.7	46656	26673.1	31952	55.4%	26.7%	1531
	MR	85438.4	48934	56961.6	32303	56.0%	26.3%	1604
$\alpha = 0, \beta = 1$	MP	60617.9	45247	14284.8	24794	43.0%	23.2%	1467
	MP++	60502.8	41592	13979.5	24498	42.5%	23.0%	1484
	MR	65994.2	46163	10384.4	25455	44.2%	21.5%	1602

The first set of results shows the statistics of the known alignment and the results from the max-weight matching algorithm. Next we show results from our algorithms for three objective parameters. The columns are: objective parameters, algorithms, matching weight, matching edge overlap, time, total correct, recall, precision, and matching triangle overlap.

alignment between the networks. The correct alignment between the networks is available from http://www.cs.vu.nl/STITCH/rameau/dump/. It contains 57,645 matches between the 154,974 nodes of Rameau and the 342,684 nodes of LCSH. (This experiment used a newer version of LCSH than the previous experiments, which is why the number of nodes changed).

To build the set of potential matches, we translate the French subject headings to English using Google Translate (translate.google.com), and translate the English headings to French also using Google Translate. Then, we use Lucene to compute a pairwise match between the strings and keep the top 25 matches. This produces up to 100 potential matches per node, 25 from LCSH $\rightarrow$ Rameau in English, 25 from Rameau $\rightarrow$ LCSH in English, and another 50 for the same sets in French. The weights are computed in the same way as in the previous section. In total, we had 20,883,500 possible edges between the graphs. Of these, only 42,215 of the correct matches appeared. The overlap induced by the correct set of matches is 39,749.

The results and running time from our algorithms are presented in Table IV. In summary, NetAlignMR computes the best results in terms of the optimization objective, but it also takes the most time. NetAlignMP++ is the runner-up and fills the gap in results and run-time between NetAlignMP and NetAlignMR. With respect to recall and precision, NetAlignMR has the highest recall (57.6%) with good precision (27.0%), but NetAlignMP and NetAlignMP++ always have slightly higher precision. Note that we performed no specific tuning to account for the differences in French and English. We did not test SpaIsoRank given its performance in the previous studies.

In the table, we also showed the number of triangles overlapped by a matching. This number appears to be indicative of the true matching performance. We believe these results demonstrate that including overlapped triangles into the objective may improve matching algorithms. In this article, we considered alignments where each node maps to at most one node in the other graph. In certain applications this constraint can be relaxed (each node can be matched to at most b nodes). It is not difficult to see that the factor graph representation of NetAlignMP can be extended to those settings as well, this can be done by updating the definition of function nodes f_i and $g_{i'}$. In fact when $\beta = 0$ this extension has been studied by Huang and Jebara [2007], Bayati et al. [2007a], and Sanghavi et al. [2011]. We plan to investigate these ideas in the future.

9. DISCUSSION

Let us recap. Network alignment is an important tool in a variety of applications including systems biology, computer vision and ontology matching. It is especially useful for comparing large datasets with inherent and related graph structures. Here, we explored matching protein-protein interaction networks and ontologies. In the future, we envision applications of these techniques in mapping large social network structure.

Of course, finding the best alignment between two networks is NP-hard. Thus far, we are limited to attacking the problem heuristically as there is no known approximation algorithm. Many different heuristics for the problem fit nicely within our quadratic programming framework for the problem. We studied several existing algorithms this framework and compared their performance on both synthetic and real data.

We find that the NetAlignMR from Klau [2009] produces the best results when a sparse set of potential matches between two graphs exist. Our two new message-passing algorithms, NetAlignMP and NetAlignMP++, were designed based on belief propagation ideas for solving the integer optimization problem directly. They are mildly faster than NetAlignMR (roughly 1.3% in our experiments) and their results nearly tie with NetAlignMR. Additionally, our algorithms produce better solutions when the set of potential matches is dense.

There are a number of avenues for future work we plan to investigate. First, because our algorithms use message passing, they should allow simple parallel implementations, including on MapReduce style architectures. Second, in each of the real data sets we used, the nodes of the two graphs had an informative label, which helped us to apply preprocessing to produce a sparse graph of potential matches between the two graphs. All of the previously discussed algorithms utilize this fact, except for IsoRank. We also plan to investigate aligning graphs without these initial "hints."

APPENDIX

A. DERIVATION OF NetAlignMP EQUATIONS

The belief propagation algorithm (and its max-product version) is an iterative procedure for passing messages along the edges of a factor graph [Pearl 1988]. We use the notation $t = 0, 1, \ldots$ to denote the messages after t message passing steps. The BP algorithm specifies the messages to pass for a general factor graph. For our factor-graph representation we obtain two types of *real-valued* BP messages. We denote these two types by ν and λ respectively.

(1) Messages from variable nodes to function nodes. Each variable node ii' sends the following messages

$$\nu^{(t+1)}_{ii'\to f_i}(x_{ii'}) = \lambda^{(t)}_{g_{i'}\to ii'}(x_{ii'}) \prod_{ii'jj'} \lambda^{(t)}_{h_{ii'jj'}\to ii'}(x_{ii'}), \tag{11}$$

$$\nu^{(t+1)}_{ii'\to g_{i'}}(x_{ii'}) = \lambda^{(t)}_{f_i\to ii'}(x_{ii'}) \prod_{ii'jj'} \lambda^{(t)}_{h_{ii'jj'}\to ii'}(x_{ii'}), \tag{12}$$

$$\nu^{(t+1)}_{ii'\to h_{ii'jj'}}(x_{ii'}) = \lambda^{(t)}_{f_i\to ii'}(x_{ii'})\lambda^{(t)}_{g_{i'}\to ii'}(x_{ii'}) \prod_{ii'kk'\neq ii'jj'} \lambda^{(t)}_{h_{ii'kk'}\to ii'}(x_{ii'}). \tag{13}$$

Note that each variable node $ii'jj'$ has only one neighbor. Hence, its message is always defined by $\nu^{(t+1)}_{ii'jj'\to h_{ii'jj'}}(x_{ii'jj'}) = 1$.

(2) The messages from function nodes to variable nodes are:

$$\begin{aligned}
\lambda^{(t)}_{f_i\to ii'}(x_{ii'}) &= \max_{\boldsymbol{x}_{\partial f_i\setminus\{ii'\}}} \left\{ e^{[\alpha\sum_{j'} w_{ij'}x_{ij'}]} f_i(\boldsymbol{x}_{\partial f_i}) \prod_{j'\neq i'} \nu^{(t)}_{ij'\to f_i}(x_{ij'}) \right\},\\
\lambda^{(t)}_{g_{i'}\to ii'}(x_{ii'}) &= \max_{\boldsymbol{x}_{\partial g_{i'}\setminus\{ii'\}}} \left\{ e^{[\alpha\sum_{j} w_{ji'}x_{ji'}]} g_{i'}(\boldsymbol{x}_{\partial g_{i'}}) \prod_{j\neq i} \nu^{(t)}_{ji'\to g_{i'}}(x_{ji'}) \right\},\\
\lambda^{(t)}_{h_{ii'jj'}\to ii'}(x_{ii'}) &= \max_{x_{jj'},x_{ii'jj'}} \left\{ e^{\frac{\beta}{2}x_{ii'jj'}} h_{ii'jj'}(\boldsymbol{x}_{\partial h_{ii'jj'}}) \nu^{(t)}_{jj'\to h_{ii'jj'}}(x_{jj'}) \right\},\\
\lambda^{(t)}_{h_{ii'jj'}\to jj'}(x_{jj'}) &= \max_{x_{ii'},x_{ii'jj'}} \left\{ e^{\frac{\beta}{2}x_{ii'jj'}} h_{ii'jj'}(\boldsymbol{x}_{\partial h_{ii'jj'}}) \nu^{(t)}_{ii'\to h_{ii'jj'}}(x_{ii'}) \right\},\\
\lambda^{(t)}_{h_{ii'jj'}\to ii'jj'}(x_{ii'jj'}) &= \max_{x_{ii'},x_{jj'}} \left\{ e^{\frac{\beta}{2}x_{ii'jj'}} h_{ii'jj'}(\boldsymbol{x}_{\partial h_{ii'jj'}}) \nu^{(t)}_{ii'\to h_{ii'jj'}}(x_{ii'}) \nu^{(t)}_{jj'\to h_{ii'jj'}}(x_{jj'}) \right\}.
\end{aligned} \tag{14}$$

At the end of each iteration t, each variable node $x_{ii'}$ ($x_{ii'jj'}$) is assigned a binary value as follows:

$$\begin{aligned}
x^{(t)}_{ii'} &= \arg\max_{x_{ii'}} \left\{ \lambda^{(t)}_{f_i\to ii'}(x_{ii'}) \lambda^{(t)}_{g_{i'}\to ii'}(x_{ii'}) \prod_{ii'jj'} \lambda^{(t)}_{h_{ii'jj'}\to ii'}(x_{ii'}) \right\},\\
x^{(t)}_{ii'jj'} &= \arg\max_{x_{ii'jj'}} \left\{ \prod_{ii'jj'} \lambda^{(t)}_{h_{ii'jj'}\to ii'jj'}(x_{ii'jj'}) \right\}.
\end{aligned}$$

In many applications as $t\to\infty$, the assigned values $x^{(t)}_{ii'}, x^{(t)}_{ii'jj'}$ converge to good approximate solutions.

It is possible to simplify the equations above by eliminating redundancies—for example, we already mentioned that the message $\nu^{(t+1)}_{ii'jj'\to h_{ii'jj'}}(x_{ii'jj'}) = 1$ always. We now simplify the above set of equations. Since the variables x_{ij} and x_{ijrs} are binary valued, we compress the messages by sending just the log-likelihood values $m^{(t)}_{ij\to f_i} = \log\left(\nu^{(t)}_{ij\to f_i}(1)/\nu^{(t)}_{ij\to f_i}(0)\right)$. Similarly, we define messages $m^{(t)}_{ij\to g_j}$, $m^{(t)}_{ij\to h_{ijrs}}$, and $m^{(t)}_{ijrs\to h_{ijrs}}$.

Next, we will carry out these calculations for $m^{(t)}_{ij\to f_i}$.

$$m^{(t+1)}_{ii'\to f_i} = \log\left(\frac{\lambda^{(t)}_{g_{i'}\to ii'}(1)}{\lambda^{(t)}_{g_{i'}\to ii'}(0)}\right) + \sum_{ii'jj'} \log\left(\frac{\lambda^{(t)}_{h_{ii'jj'}\to ii'}(1)}{\lambda^{(t)}_{h_{ii'jj'}\to ii'}(0)}\right).$$

Each log term here can be simplified because log is a monotone function, and hence, it commutes with max. For example,

$$\log(\lambda^{(t)}_{g_{i'}\to ii'}(1)) = \max_{\substack{\boldsymbol{x}_{\partial g_{i'}\setminus\{ii'\}}\\ x_{ii'}=1}} \left\{ \alpha w_{ii'} + \alpha \sum_{j\neq i} w_{ji'}x_{ji'} + \log g_{i'}(\boldsymbol{x}_{\partial g_{i'}}) + \sum_{j\neq i} \log \nu^{(t)}_{ji'\to g_{i'}}(x_{ji'}) \right\}$$
$$= \alpha w_{ii'} + \sum_{j\neq i} \log \nu^{(t)}_{ji'\to g_{i'}}(0)$$

where the last equality uses the matching constraint imposed by $g_{i'}$. Similarly,

$$\log(\lambda^{(t)}_{g_{i'}\to ii'}(0)) = \max\left\{ \sum_{j\neq i} \nu^{(t)}_{ji'\to g_{i'}}(0), \max_{k\neq i}\left[\alpha w_{ki'} + \sum_{j\neq i} \log \nu^{(t)}_{ji'\to g_{i'}}(0) + \log(\frac{\nu^{(t)}_{ki'\to g_{i'}}(1)}{\nu^{(t)}_{ki'\to g_{i'}}(0)}) \right] \right\}.$$

Therefore, we have

$$\log\left(\frac{\lambda^{(t)}_{g_{i'}\to ii'}(1)}{\lambda^{(t)}_{g_{i'}\to ii'}(0)}\right) = \alpha w_{ii'} - \left\{ \max_{k\neq i}(\alpha w_{ki'} + m^{(t)}_{ki'\to g_{i'}}) \right\}_+$$

where $(a)_+$ means $\max(a,0)$. Similar calculations for $\lambda^{(t)}_{h_{ii'jj'}\to ii'}$ yield

$$\log\left(\frac{\lambda^{(t)}_{h_{ii'jj'}\to ii'}(1)}{\lambda^{(t)}_{h_{ii'jj'}\to ii'}(0)}\right) = \max\left(\frac{\beta}{2} + \log \nu^{(t)}_{jj'\to h_{ii'jj'}}(1), \log \nu^{(t)}_{jj'\to h_{ii'jj'}}(0)\right)$$
$$- \max\left(\log \nu^{(t)}_{jj'\to h_{ii'jj'}}(1), \log \nu^{(t)}_{jj'\to h_{ii'jj'}}(0)\right)$$
$$= \left(\frac{\beta}{2} + m^{(t)}_{jj'\to h_{ii'jj'}}\right)_+ - \left(m^{(t)}_{jj'\to h_{ii'jj'}}\right)_+.$$

Summarizing, we obtain

$$m^{(t+1)}_{ii'\to f_i} = \alpha w_{ii'} - \left\{ \max_{k\neq i}(\alpha w_{ki'} + m^{(t)}_{ki'\to g_{i'}}) \right\}_+ + \sum_{ii'jj'} \left((\frac{\beta}{2} + m^{t}_{jj'\to h_{ii'jj'}})_+ - (m^{(t)}_{jj'\to h_{ii'jj'}})_+ \right).$$

By symmetry we obtain

$$m^{(t+1)}_{ii'\to g_{i'}} = \alpha w_{ii'} - \left\{ \max_{k'\neq i'}(\alpha w_{ik'} + m^{(t)}_{ik'\to f_i}) \right\}_+ + \sum_{ii'jj'} \left((\frac{\beta}{2} + m^{(t)}_{jj'\to h_{ii'jj'}})_+ - (m^{(t)}_{jj'\to h_{ii'jj'}})_+ \right).$$

and

$$m^{(t+1)}_{ii'\to h_{ii'jj'}} = 2\alpha w_{ii'} - \left\{ \max_{k\neq i}(\alpha w_{ki'} + m^{(t)}_{ki'\to g_{i'}}) \right\}_+ - \left\{ \max_{k'\neq i'}(\alpha w_{ik'} + m^{(t)}_{ik'\to f_i}) \right\}_+$$
$$+ \sum_{ii'kk'\neq ii'jj'} \left((\frac{\beta}{2} + m^{(t)}_{kk'\to h_{ii'kk'}})_+ - (m^{(t)}_{kk'\to h_{ii'kk'}})_+ \right).$$

We can simplify these equations further to prove Lemma A.1. This is achieved by defining $m^{(t)}_{ii'\to f_i} \equiv w_{ii'} + m^{(t)}_{ii'\to f_i}$ and $m^{(t)}_{ii'\to g_{i'}} \equiv \alpha w_{ii'} + m^{(t)}_{ii'\to g_{i'}}$ and replacing $\tilde{\beta}$ with $\beta/2$.

LEMMA A.1. *The max-product Eqs.* (11)–(14) *are equivalent to the simplified BP Eqs.* (5)–(6).

ACKNOWLEDGMENTS

We thank Margot Gerritsen for helping with an initial version of this manuscript. We also extend our heartfelt thanks to the computational approaches to digital stewardship group at Stanford. In particular, we'd like to thank Laura Campbell, Barbara Tillet, and Ed Summers for their own contributions. Thanks to Jure Leskovec for discussing the problem with us. Also, we thank Nathan Sakunkoo for implementing the Soft TF-IDF-based scoring algorithm.

REFERENCES

Adams, W. P. and Johnson, T. A. 1994. Improved linear programming based lower bounds for the quadratica ssignment problem. In *Quadratic Assignment and Related Problems*, P. Pardalos and H. Wolkowicz Eds., DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 16., AMS, Providence, R. I. 43–77.

Bayati, M., Shah, D., and Sharma, M. 2005. Maximum weight-matching via max-product belief propagation. In *Proceedings of the International Symposium on Information Theory*. 1763–1767.

Bayati, M., Borgs, C., Chayes, J., and Zecchina, R. 2007a. Belief-propagation for weighted b-matchings on arbitrary graphs and its relation to linear programs with integer solutions. arxiv.org/abs/0709.1190.

Bayati, M., Kim, J. H., and Saberi, A. 2007b. A sequential algorithm for generating random graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, vol. abs/cs/0702124. Springer Berlin, 326–340.

Bayati, M., Gerritsen, M., Gleich, D. F., Saberi, A., and Wang, Y. 2009. Algorithms for large, sparse network alignment problems. In *Proceedings of the IEEE International Conference on Data Mining*. 705–710. DOI: 10.1109/ICDM.2009.135.

Berg, J. and Lässig, M. 2006. Cross-species analysis of biological networks by bayesian alignment. *Proc. Nat. Acad. Sci. 103*, 29, 10967–10972.

Blondel, V. D., Gajardo, A., Heymans, M., Senellart, P., and Dooren, P. V. 2004. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM Rev. 46*, 4, 647–666.

Bradde, S., Braunstein, A., Mohoudi, H., Tria, F., Weigt, M., and Zecchina, R. 2010. Aligning graphs and finding substructures by a cavity approach. *Europhys. Lett. 89 37009*. DOI: 10.1209/0295-5075/89/37009.

Braunstein, A. and Zecchina, R. 2006. Learning by message passing in networks of discrete synapses. *Phys. Rev. Lett*.

Burkard, R., Dell'Amico, M., and Martello, S. 2012. *Assignment Problems* Revised Ed. SIAM.

Caetano, T., McAuley, J., Cheng, L., Le, Q., and Smola, A. 2009. Learning graph matching. *IEEE Trans. Patt. Anal. Mach. Intell. 31*, 6, 1048–1058.

Cohen, W. W., Ravikumar, P., and Fienberg, S. 2003. A comparison of string metrics for matching names and records. In *Proceedings of the KDD Workshop on Data Cleaning and Object Consolidation*. ACM, New York.

Conte, D., P. Foggia, P., Sansone, C., and Vento, M. 2004. Thirty years of graph matching in pattern recognition. *Int. J. Patt. Recog. Artif. Intell. 18*, 3, 265–298.

Donoho, D. L., Maleki, A., and Montanari, A. 2009. Message Passing Algorithms for Compressed Sensing. *Proc. Nat. Acad. Sci. 106*, 18914–18919.

Ehrig, M. and Staab, S. 2004. QOM - quick ontology mapping. In *Proceedings of the 3rd International Semantic Web Conference*. Lecture Notes in Computer Science, vol. 3298. Springer, Berlin, 683–697.

El-Kebir, M., Heringa, J., and Klau, G. W. 2011. Lagrangian relaxation applied to sparse global network alignment. In *Proceedings of the 6th IAPR International Conference on Pattern Recognition in Bioinformatics (PRIB'11)*. Lecture Notes in Computer Science, vol. 7036. Springer, 225–236.

Flannick, J., Novak, A., Do, C. B., Srinivasan, B. S., and Batzoglou, S. 2008. Automatic parameter learning for multiple network alignment. In *Proceedings of the 12th Annual International Conference on Computational Molecular Biology (RECOMB'08)*. 214–231.

Flannick, J., Novak, A., Srinivasan, B. S., McAdams, H. H., and Batzoglou, S. 2006. Græmlin: General and robust alignment of multiple large interaction networks. *Genome Res. 16*, 1169–1181.

Frey, B. J. and Dueck, D. 2007. Clustering by passing messages between data points. *Science 315*, 5814, 972–976.

Frieze, A. M. and Yadegar, J. 1983. On the quadratic assignment problem. *Disc. Appl. Math. 5*, 1, 89–98.

Gallager, R. G. 1963. *Low Density Parity Check Codes*. MIT Press, Cambridge MA.

Hatcher, E. and Gospodnetic, O. 2004. *Lucene in Action*. Manning Publications.

Hu, W., Jian, N., Qu, Y., and Wang, Y. 2005. GMO: A graph matching for ontologies. In *Proceedings of K-CAP Workshop on Integrating Ontologies*. 41–48.

Hu, W., Qu, Y., and Cheng, G. 2008. Matching large ontologies: A divide-and-conquer approach. *Data Knowl. Eng. 67*, 1, 140–160.

Huang, B. and Jebara, T. 2007. Loopy belief propagation for bipartite maximum weight b-matching. In *Proceedings of the Artificial Intelligence and Statistics (AISTATS)*.

Klau, G. 2009. A new graph-based method for pairwise global network alignment. *BMC Bioinformatics 10*, Suppl 1, S59.

Koyutürk, M., Kim, Y., Topkara, U., Subramaniam, S., Szpankowski, W., and Grama, A. 2006. Pairwise alignment of protein interaction networks. *J Comput. Biol. 13*, 182–199.

Kuchaiev, O., Milenkovic, T., Memisevic, V., Hayes, W., and Przulj, N. 2009. Topological network alignment uncovers biological function and phylogeny. arXiv 0810.3280.

Kuchaiev, O., Milenkovic, T., Memisevic, V., Hayes, W., and Przulj, N. 2010. Topological network alignment uncovers biological function and phylogeny. *J R Soc. Interface. 7*, 1341–1354.

Lacoste-Julien, S., Taskar, B., Klein, D., and Jordan, M. I. 2006. Word alignment via quadratic assignment. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL'06)*. Association for Computational Linguistics, Stroudsburg, PA, 112–119.

Lawler, E. L. 1963. The quadratic assignment problem. *Manage. Sci. 9*, 4, 586–599.

Liang, Z., Xu, M., Teng, M., and Niu, L. 2006. Netalign: A web-based tool for comparison of protein interaction networks. *Bioinformatics 22*, 17, 2175–2177.

Melnik, S., Garcia-Molina, H., and Rahm, E. 2002. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of the 18th International Conference on Data Engineering*. IEEE Computer Society, Los Alamitos, CA, 117.

Mezard, M. and Montanari, A. 2009. *Information, Physics, and Computation* (Oxford Graduate Texts). Oxford University Press.

Mezard, M. and Zecchina, R. 2002. Random k-satisfiability: From an analytic solution to a new efficient algorithm. *Phys. Rev. E 66*.

Murphy, K., Weiss, Y., and Jordan., M. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA.

Sanghavi, S., Malioutov, D., and Willsky, A. 2011. Belief propagation and LP relxation for weighted matching in general graphs. *IEEE Trans. Inf. Theory 57*, 4.

Schellewald, C. and Schnörr, C. 2005. Probabilistic subgraph matching based on convex relaxation. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, Springer, Berlin, 171–186.

Sharan, R., Ideker, T., Kelley, B., Shamir, R., and Karp, R. M. 2005. Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. *J. Comput. Biol. 12*, 835–846.

Singh, R., Xu, J., and Berger, B. 2007. Pairwise global alignment of protein interaction networks by matching neighborhood topology. In *Proceedings of the 11th Annual International Conference on Research in Computational Molecular Biology (RECOMB)*. Lecture Notes in Computer Science, vol. 4453, Springer, Berlin, 16–31.

Singh, R., Xu, J., and Berger, B. 2008. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proc. Nat. Acad. Sci. 105*, 35, 12763–12768.

Šváb, O. 2007. Exploiting patterns in ontology mapping. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007)*, Busan, South Korea, K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, and L. J. B. Nixon Eds., Lecture Notes in Computer Science, vol. 4825, Springer Verlag, Berlin, 950–954.

Tappen, M. and Freemand, W. 2003. Graph cuts and belief propagation for stereo, using identical mrf parameters. In *Proceedings of the ICCV*.

Wikipedia. 2007. Wikipedia XML database dump from April 2, 2007. `http://en.wikipedia.org/wiki/Wikipedia:Database_download`.

Yanover, C. and Weiss, Y. 2002. Approximate inference and protein folding. In *Advances in Neural Processing Systems*.

Received October 2011; revised May 2012; accepted June 2012