Algorithms for Large, Sparse Network Alignment Problems

Mohsen Bayati*, Margot Gerritsen†, David F. Gleich‡, Amin Saberi§ and Ying Wang¶
*Electrical Engineering Department, Stanford University, bayati@stanford.edu
†Energy Resources Engineering Department, Stanford University, margot.gerritsen@stanford.edu
‡ICME, Stanford University, dgleich@stanford.edu
§MS&E Department, Stanford University, saberi@stanford.edu
¶ICME, Stanford University, yw1984@stanford.edu

***Abstract*—We propose a new distributed algorithm for sparse variants of the network alignment problem, which occurs in a variety of data mining areas including systems biology, database matching, and computer vision. Our algorithm uses a belief propagation heuristic and provides near optimal solutions for this NP-hard combinatorial optimization problem. We show that our algorithm is faster and outperforms or ties existing algorithms on synthetic problems, a problem in bioinformatics, and a problem in ontology matching. We also provide a unified framework for studying and comparing all network alignment solvers.**

***Keywords*-network alignment; belief propagation; graph matching; message-passing**

I. Introduction

The focus of the network alignment problem is to find approximate isomorphisms, or alignments, between similar graphs. It is widely applied to problems in bioinformatics [1], [2], database schema matching [3], ontology matching [4], and computer vision [5], [6] (where it is also known as the weighted graph matching problem). Recent algorithms solve this problem approximately for graphs with around 10,000 vertices, however there are few comparisons among them. We propose a new algorithm based on belief propagation, survey and compare existing algorithms on real and synthetic data, and investigate a problem with over 200,000 vertices.

A. Network alignment

Consider two graphs $A = (V_A, E_A)$, $B = (V_B, E_B)$ with vertex sets $V_A = \{1, 2, \ldots, n\}$ and $V_B = \{1', 2', \ldots, m'\}$ and let L be a bipartite graph between the vertices of A and B, formally $L = (V_A \cup V_B, E_L)$. The goal is to find a matching between vertices in A and B where all possible matches must be from the edges of L. A *matching* in L is a subset of E_L such that no two edges share a common endpoint. Let M be such a matching. For a matching M, we say that an edge $(i, i') \in M$ forms a *square* with another edge $(j, j') \in M$ if $(i, j) \in E_A$ and $(i', j') \in E_B$. In such situation we also say that the edges $(i, j) \in E_A$ and $(i', j') \in E_B$ are *overlapped*. Denote the set of all squares by V_S. See Figure 1 for an illustration.

Definition 1. *Given A, B, and L as above, the overlap graph matching problem is to find a matching M that maximizes the total number of squares (overlapped edges).*

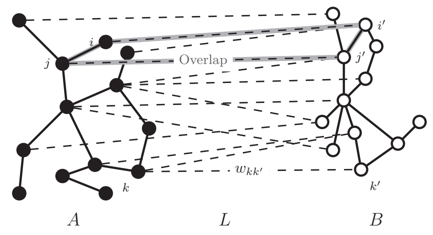


Figure 1. The setup for the network alignment problem. The goal is to maximize the number of squares in any matching while maximizing the weight of the matching as well.

When L is the complete bipartite graph, then overlap graph matching is the maximum common subgraph problem. Thus, it is NP-hard. When each edge (i, i') of L has a non-negative weight $w_{ii'}$, often the problem is generalized to finding a matching that is large in both weight and the number of squares. This new version is called the *network alignment problem* and is formally defined in Section II-A.

B. Our contribution

Most of the existing literature on the problem (except Klau [1]) assumes that L is the complete bipartite graph. We explicitly formulate the problem with only a sparse set of possible matches between A and B. This formulation easily handles sparse graphs with over 200,000 vertices and also expresses many existing algorithms. Using this formulation: 1) we provide the first unified framework for studying all algorithms; 2) we propose a fast algorithm based on max-product belief propagation; and 3) we compare the algorithms on two synthetic and three real datasets. These experiments show our belief propagation algorithm is fast and robust. It yields near-optimal solutions in the tests, and it outperforms or nearly ties with the best existing algorithm [1] in terms of quality of the solution on extremely sparse graphs. On slightly dense graphs, our algorithm performs better than others. Therefore, we see great potential for our approach in future network alignment applications. All of our algorithms are implemented in MATLAB and are available from http://www.stanford.edu/~dgleich/publications/2009/netalign. A longer version of this manuscript is available on arXiv [7].

C. Related Work

The network alignment problem has a rich and evolving literature. It has been studied within the context of database schema matching [3], protein interaction alignment [2], [8], ontology matching [4], and pattern recognition [5].

The maximum common subgraph problem is the oldest related literature. Most of the algorithms for this problem seek an exact solution and require exponential time [9]. Other heuristics include [10], Graemlin [11], the IsoRank algorithm [2], the closely related similarity flooding algorithm [3], and an SDP based algorithm [6]. Finally, Klau [1] formulates the problem as a quadratic program and proposes a series of linear programming relaxations. Among these algorithms only IsoRank (or similarity flooding), and Klau's algorithm can handle large graphs well. Independent from our work, [12] developed a different algorithm based on BP to study the maximum-clique and graph-alignment problems for random graphs.

II. Problem Formulation

In this section we investigate a QP and LP formulation of the problem and several relaxations.

A. Quadratic Program Formulation

First we introduce the notation ii' instead of (i,i') for an edge of L. For each edge $ii' \in E_L$, we assign a binary variable $x_{ii'} = 1$ if ii' is in the matching M or $x_{ii'} = 0$ if it is not. The total number of squares (overlapped edges) is $(1/2)\mathbf{x}^T \boldsymbol{S}\mathbf{x}$, where $\mathbf{x}$ is the $|E_L| \times 1$ vector of all $x_{ii'}$'s and $\boldsymbol{S}$ is a $0,1$ matrix of size $|E_L| \times |E_L|$ such that $S_{ii',jj'} = 1$ if and only if the corresponding edges ii' and jj' of L form a square (contribute an overlap). We construct both $\mathbf{x}$ and $\boldsymbol{S}$ according to a fixed ordering of the edges of L.

The vector $\mathbf{x}$ must be a valid matching, so for any vertex of L, the sum of $x_{ii'}$'s on the edges incident to it cannot exceed 1. We can write this as $\boldsymbol{A}\mathbf{x} \le \mathbf{1}$. Here $\mathbf{1} = \mathbf{1}_{n+m} \in \mathbb{R}^{(n+m)\times 1}$ is the vector of all ones. Using these definitions, the network alignment problem is an integer quadratic program (QP)

$$\begin{array}{ll} \underset{\mathbf{x}}{\text{maximize}} & \alpha \mathbf{w}^T\mathbf{x} + (\beta/2)\mathbf{x}^T\boldsymbol{S}\mathbf{x} \\ \text{subject to} & \boldsymbol{A}\mathbf{x} \le \mathbf{1}, \qquad x_{ii'} \in \{0,1\} \end{array} \tag{NAQP}$$

where α, β are arbitrarily chosen nonnegative constants to balance the matching and overlap objectives. Picking $\alpha = 0$ and $\beta = 1$ produces the overlap graph matching problem. See Figure 2 for an example.

B. Linear Program Formulations

Klau [1] originally described a linear programming relaxation of (NAQP). In the longer manuscript online, we show the relaxation in our sparse-alignment formulation [7]. The most efficient algorithm for the LP is a sequence of max-weight matching problems. This algorithm is called NetAlignMR in the remainder of this short paper.

III. A belief propagation algorithm

In this section we introduce a distributed algorithm that is inspired by a recent result on solving the maximum weight matching problem using the max-product version of belief propagation (BP) [13]. For simplicity, we refer to this algorithm by BP. We will show in Sections IV-V that the BP algorithm always produces better results than IsoRank, and compared to NetAlignMR, BP is either better or its solutions are close. But, BP has a better running time than NetAlignMR and IsoRank.

Recently, BP and its variations have been successful for the solution of random constraint satisfaction problems [14]. It has also been known for many years in the context of coding theory [15], artificial intelligence [16], and computer vision [17]. Recent applications can also be found in systems biology [18] and data clustering [19].

The goal of the BP algorithm that we develop here is to solve the integer program (NAQP) directly. We obtain a distributed algorithm that runs by passing messages along the edges of graph L and also along the squares. The main intuition behind this algorithm (and, indeed, all BP algorithms) is that each vertex of the graph assumes the graph has no cycles, and makes the best (greedy) decision based on this assumption.

In the interest of space, we leave a detailed derivation of the algorithm for the longer version of this paper [7]. Here is our BP algorithm for the network alignment problem.

Algorithm BP *Input* (V_A, V_B, E_L, V_S) *Output* $M \subset E_L$

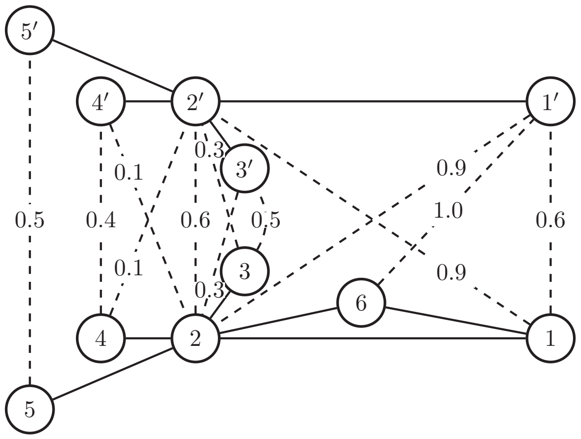
1) At times $t = 0, 1, \ldots$, each edge ii' sends two messages of the form $m^{(t)}_{ii'\to i}$ and $m^{(t)}_{ii'\to i'}$ and also sends one message of the form $m^{(t)}_{ii'\to ii'jj'}$ for any square $ii'jj'$. All messages of time $t = 0$ are initialized by an arbitrary number (let us say 0).
2) For $t \ge 1$, and for all $ii' \in E_L$

$$m^{(t)}_{ii'\to i} = \alpha w_{ii'} - \left(\max_{k\neq i}\left[m^{(t-1)}_{ki'\to i'}\right]\right)^+ + \sum_{ii'jj'\in V_S} \min\left(\frac{\beta}{2}, \max(0, \frac{\beta}{2} + m^{(t-1)}_{jj'\to ii'jj'})\right). \tag{1}$$

The update rule for $m^{(t)}_{ii'\to i'}$ is similar, and

$$m^{(t)}_{ii'\to ii'jj'} = \sum_{\substack{kk'\neq jj' \\ ii'kk'\in V_S}} \min\left(\frac{\beta}{2}, \max(0, m^{(t-1)}_{kk'\to ii'kk'} + \frac{\beta}{2})\right) + \alpha w_{ii'} - \left(\max_{k\neq i}\left[m^{(t-1)}_{ki'\to i'}\right]\right)^+ - \left(\max_{k'\neq i'}\left[m^{(t-1)}_{ik'\to i}\right]\right)^+ . \tag{2}$$

where $(a)^+$ means $\max(a,0)$.
3) At the end of iteration t each vertex i selects the edge ii' that sends the maximum incoming message $m^{(t)}_{ii'\to i}$ to it. Denote the set of these selected edges by $M(t)$. Now, repeat (2)-(3) until $M(t)$ converges.



edge order
(2,2′)
(2,1′)
(2,3′)
(2,4′)
(1,2′)
(1,1′)
(3,2′)
(3,3′)
(4,2′)
(4,4′)
(5,5′)
(6,1′)

$$\underbrace{\begin{pmatrix} 0&0&0&0&0&1&0&1&0&1&1&1\\ 0&0&0&0&1&0&1&0&1&0&0&0\\ 0&0&0&0&1&0&1&0&1&0&0&0\\ 0&0&0&0&1&0&1&0&1&0&0&0\\ 0&1&1&1&0&0&0&0&0&0&0&1\\ 1&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&1&1&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&1&1&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&0&0&0\\ 1&0&0&0&1&0&0&0&0&0&0&0 \end{pmatrix}}_{S}, \underbrace{\begin{pmatrix} 0.6\\0.9\\0.3\\0.1\\0.9\\0.6\\0.3\\0.5\\0.1\\0.4\\0.5\\1.0 \end{pmatrix}}_{w}$$

$$\underbrace{\begin{pmatrix} 1&1&1&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&0&1\\ 1&0&0&0&1&0&1&0&1&0&0&0\\ 0&1&0&0&0&1&0&0&0&0&0&1\\ 0&0&1&0&0&0&0&1&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&0 \end{pmatrix}}_{A}$$

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

A. *Convergence of BP*

In practice $M(t)$ does not necessarily converge, and in most cases it oscillates between a few states. Therefore, we can terminate the algorithm when such an oscillation is observed, and use the current messages of the edges as weights of a MWM problem to obtain an integer solution. Another approach for resolving the oscillation, is to use a damping factor $\gamma \in [0, 1]$. In this approach each messages at time t is a linear combination of its value at time $t-1$ and its new value is based on equations (1)-(2) with coefficients $(1-\gamma^t)$ and γ^t respectively. See for example [20], [21], [19] for various ways of damping BP messages.

B. *A matrix formulation*

For $\boldsymbol{A} \in \mathbb{R}^{m,n}$ and $\mathbf{x} \in \mathbb{R}^n$, define $\boldsymbol{A} \boxdot \mathbf{x}$ to be a vector in $\mathbb{R}^{m\times 1}$ where its r^{th} entry is $\max_j a_{r,j} x_j$. This operator is just the regular matrix-vector product but with the summation $(\boldsymbol{A}\mathbf{x})_i = \sum_j a_{i,j} x_j$ replaced by maximization.

Now we present a matrix formulation of the BP algorithm. We need to split the constraint matrix $\boldsymbol{A}$ into $[\boldsymbol{A}_r^T \; \boldsymbol{A}_c^T]^T$ corresponding to the matching constraints in graph A and graph B, respectively.

```
*Algorithm* NetAlignBP
*Input* $\boldsymbol{A} = [\boldsymbol{A}_r^T \; \boldsymbol{A}_c^T]^T$, $\boldsymbol{S}$, $\mathbf{w}$, damping parameter $\gamma$, niter
$\mathbf{y}^{(0)} = 0, \mathbf{z}^{(0)} = 0, \boldsymbol{S}^{(0)} = 0$
for $t$ = 1 to niter
    $\boldsymbol{F} = \text{bound}_{0,\frac{\beta}{2}}(\boldsymbol{S}^{(t-1)^T} + \frac{\beta}{2}\boldsymbol{S})$
    $\mathbf{d} = \boldsymbol{F} \cdot \mathbf{e}$
    $\mathbf{y}^{(t)} = \alpha\mathbf{w} - \text{bound}_{0,\infty}[(\boldsymbol{A}_r^T\boldsymbol{A}_r - \boldsymbol{I}) \boxdot \mathbf{z}^{(t-1)}] + \mathbf{d}$
    $\mathbf{z}^{(t)} = \alpha\mathbf{w} - \text{bound}_{0,\infty}[(\boldsymbol{A}_c^T\boldsymbol{A}_c - \boldsymbol{I}) \boxdot \mathbf{y}^{(t-1)}] + \mathbf{d}$
    $\boldsymbol{S}^{(t)} = (\text{diag}[\mathbf{y}^{(t)} + \mathbf{y}^{(t)} - \alpha\mathbf{w} - \mathbf{d}]) \cdot \boldsymbol{S} - \boldsymbol{F}$
    $(\mathbf{y}^{(t)}, \mathbf{z}^{(t)}, \boldsymbol{S}^{(t)}) \leftarrow$
        $\gamma^t(\mathbf{y}^{(t)}, \mathbf{z}^{(t)}, \boldsymbol{S}^{(t)}) + (1-\gamma^t)(\mathbf{y}^{(t-1)}, \mathbf{z}^{(t-1)}, \boldsymbol{S}^{(t-1)})$
end
```

To produce a binary output, NetAlignBP ends by solving two maximum weight matching problems weighted by the messages $\mathbf{y}^{(t)}$ and $\mathbf{z}^{(t)}$. It then outputs the solution with the best objective.

IV. Synthetic experiments

In this section we generate two types of synthetic network alignment data and compare our NetAlignBP algorithm with IsoRank (actually a sparse variant SpaIsoRank, see the full paper [7]) and NetAlignMR.

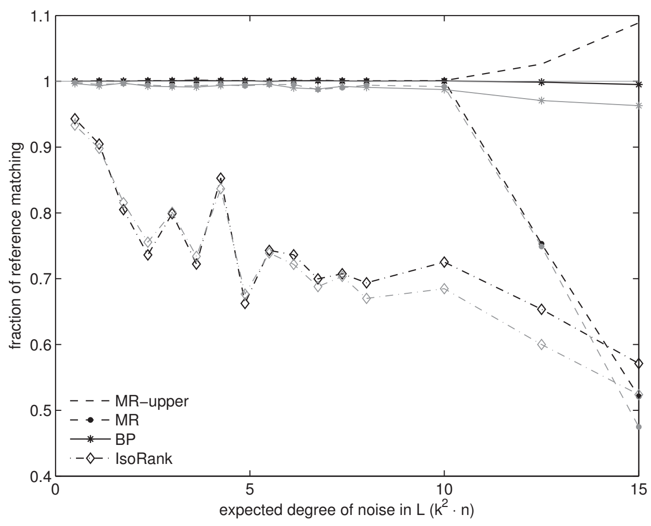
In the first class of data, we start by taking two copies of a $k \times k$ grid as A and B. Then for each vertex $i \in V_A$ and its corresponding copy in V_B we add the edge ii' to L. We call these $|V_A|$ edges *correct*. Then for any two random vertex pairs $i \in V_A$, $j' \in V_B$ we add the edge ij' to L, independently, with probability p for some fixed constant $0 \le p \le 1$.

In order to make the model more realistic, we need to perturb graphs A, B by adding an edge between any two random vertices $u, v \in V_A$ ($u', v' \in V_B$), independently, with probability proportional to $q/d_A(u,v)^2$ ($q/d_B(u',v')^2$), where $d_G(u,v)$ is the distance between u, v in G.

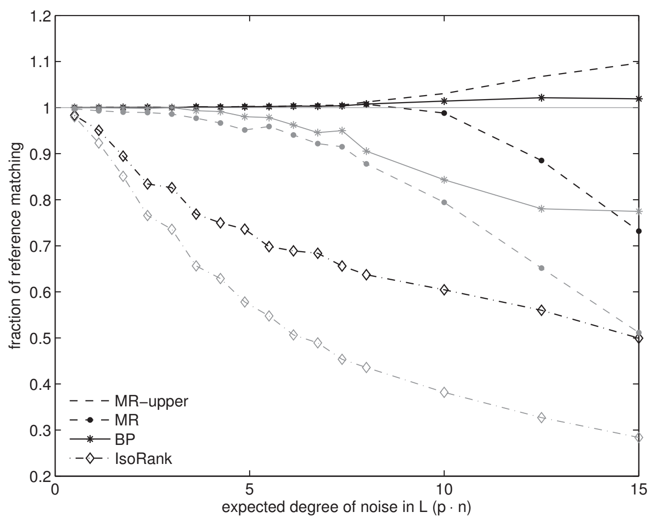
In these problems, the *ideal alignment* is known: the set of all correct edges ii'. In real-world networks, each correct edge ii' may be mistaken by many edges jj' where $d_A(i,j)$ and $d_B(i',j')$ are small. To capture this, we add some random edges to L within graph distance d of the end points of the ideal alignment.

In the second class, we let A, B be random power-law degree sequence graphs with exponent θ and n vertices. This means the fraction of vertices with degree k is proportion to $k^{-\theta}$. Similarly, we add correct and noisy edges to L and perturb graphs A, B. But we do not add the additional distance based noise to L.

In our results, we compare the outputs of all algorithms to the correct matching edges ii' between the graphs A and B. Figure IV shows the average fraction of the correct matching obtained by each algorithm over 10 trials. Here the objective value of the network alignment is the total number



(a) Grid graphs ($k = 20, q = 2, d = 1$)



(b) Power-law graphs ($\theta = 1.8, n = 400, q = 1$)

Figure 3. Upper bounds and solutions to synthetic problems on grid-graphs (a) and power-law graphs (b). Dark lines are values of the network alignment objective and light lines are the number of correct matches. The BP algorithm performs the best in terms of objective and correct matches. See Section IV for more information.

of squares, and the dark lines in the figure show the ratio of the algorithm's objective to the objective value when the correct matching is used.

Although the larger values for the objective tend to recover a higher fraction of the correct matching, the correct matching may not produce the best objective (highest number of squares) when L is highly corrupted with a large number of edges. This explains a few cases where the fraction is more than 1. Similarly, the light lines show the fraction of the correct matches that each algorithm has recovered. These values track the objective values showing that the network alignment objective is a good surrogate for the number of correct matches objective.

We see that BP and MR produce the best solutions. When the amount of random noise in L exceeds an expected degree ($n \cdot p$) 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 BP algorithm performs significantly better than the MR algorithm.

We used the BP algorithm with $\alpha = 1, \beta = 2$, the IsoRank algorithm with $\gamma = 0.95$, and the MR algorithm with $\alpha = 0, \beta = 1$ for these experiments.

V. 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 bioinformatics and ontology matching. Table I summarizes the properties of these problems.

Table I

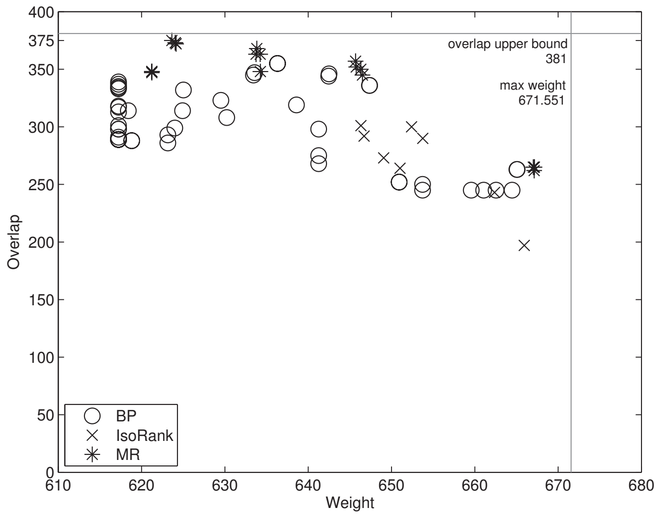
Problem	$\lvert V_A\rvert$	$\lvert E_A\rvert$	$\lvert V_B\rvert$	$\lvert E_B\rvert$	$\lvert E_L\rvert$
dmela-scere	9459	25636	5696	31261	34582
Mus M.-Homo S.	3247	2793	9695	32890	15810
lcsh2wiki-small	1919	1565	2000	3904	16952
lcsh2wiki-full	297266	248230	205948	382353	4971629

A. Bioinformatics

The alignment of protein-protein interaction (PPI) networks of different species is an important problem in bioinformatics [2]. We consider aligning the PPI network of Drosophila melanogaster (fly) with Saccharomyces cerevisiae (yeast), and Homo sapiens (human) with Mus musculus (mouse). These PPI networks are available in several open databases and they are used in [8] and [1], respectively. While the results of the experiment are rich in biological information, we focus our interest solely on the optimization problem.

Figure 4 shows the performance of the three algorithms (NetAlignBP, NetAlignMR, SpaIsoRank) on these two alignments. For each algorithm, we run it for 100 iterations (SpaIsoRank and NetAlignBP) or 500 iterations (NetAlignMR) with varied α and β parameters and various damping parameters and stepsizes. For each combination of parameters, we record the best iterate ever generated and plot the overlap and weight of the alignments in the figure.

In both problems NetAlignBP and NetAlignMR manage to obtain near-optimal solutions. NetAlignMR slightly outperforms NetAlignBP, and they both dominate IsoRank. NetAlignBP has a better running time instead.



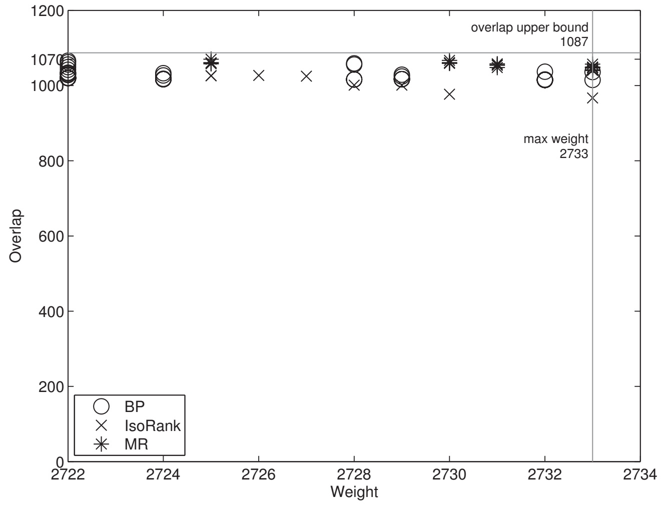
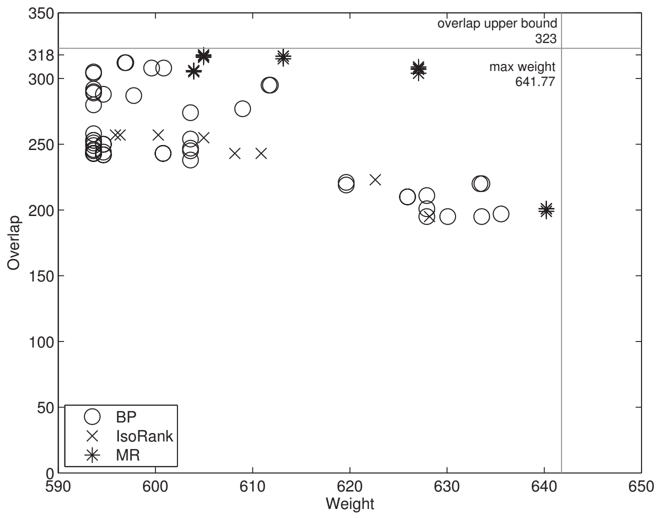


Figure 4. Results of the three algorithms NetAlignBP, SpaIsoRank, and NetAlignMR on the dmela-scere alignment from [8] (left), and on the Mus M.-Homo S. alignment from [1] (right).



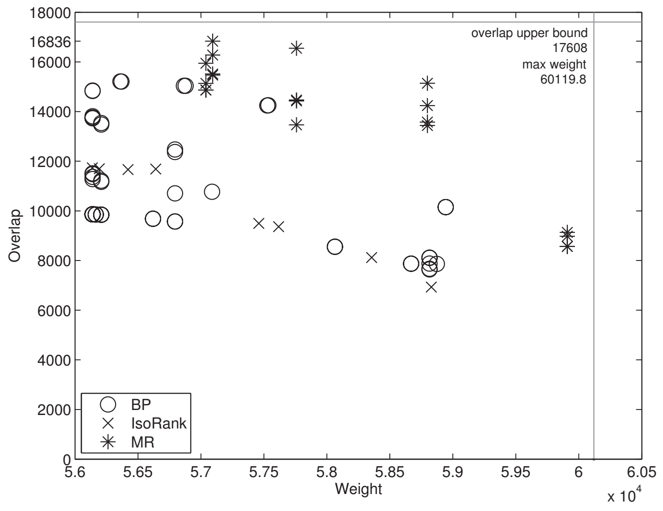


Figure 5. Results of the three algorithms NetAlignBP, SpaIsoRank, and NetAlignMR on lcsh2wiki-small (left), and on lcsh2wiki-full (right).

B. *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 [22]. Each node in these networks has a small text label, and we use a Lucene search index to quickly find potential matches between nodes based on the text. To score the matches, we use the SoftTF-IDF scoring routine [23]. 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 5. We leave a detailed description of this data-set and implications of the network alignment to the longer version of the paper [7].

We saw similar behaviors as in Section V-A. NetAlignMR and BP are close in overlap and they outperform IsoRank. Though not shown in the figure, BP obtains a lower bound of 16204 in lcsh2wiki-full with $\gamma = 0.9995$, $\alpha = 0.2$ and $\beta = 1$.

In all our real datasets L is quite sparse, making NetAlignMR more favorable. Still, BP produces closely comparable results and has an advantage on running time. We leave evaluating the accuracy and precision of the matches to a future paper. IsoRank is outperformed in these experiments, but it was originally designed for multi-way matching, i.e. between more than two PPI networks.

VI. Conclusions

In all of our experiments NetAlignBP yields near-optimal solutions, and it significantly outperforms other algorithms when L is dense. The experiments also show that results of NetAlignBP and NetAlignMR are withing 95% of the upper bound in large graphs with 5,000,000 potential edges and hundreds of thousands of vertices, which is promising since the problem is APX-hard. Our study shows that the current network alignment algorithms are fast and good enough to handle large-scale real world problems.

When the set of potential matches (L) is dense, all algorithms studied in this paper are impractical for large graphs. NetAlignBP suffers from a large storage demand ($|E_A| \times |E_B|$). NetAlignMR has similar storage limitations and becomes slow since it is solving a large number of MWM instances per iteration. When L is the complete bipartite graph, IsoRank does not need to form the matrix $\boldsymbol{S}$ explicitly, thus greatly saving on storage. However, the solution it produces is not satisfying. One future direction will be finding an algorithm that performs well when L is dense.

NetAlignMR is based on a tightened LP where the key idea is solving a sub matching problem for each row of $\boldsymbol{S}$. It is a tempting idea to combine this technique with NetAlignBP. Furthermore, our experiments show that NetAlignMR corresponds to an LP with a small integrality gap in most cases. Finding another LP relaxation with a smaller integrality gap is an interesting problem.

References

[1] G. Klau, "A new graph-based method for pairwise global network alignment," *BMC Bioinformatics*, vol. 10, no. S1, p. S59, January 2009.

[2] R. Singh, J. Xu, and B. Berger, "Pairwise global alignment of protein interaction networks by matching neighborhood topology," in *Proc. of the 11th Annual Intl. Conf. on Research in Computational Molecular Biology (RECOMB)*, ser. LNCS, vol. 4453. Springer, 2007, pp. 16–31.

[3] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity flooding: A versatile graph matching algorithm and its application to schema matching," in *Proc. of the 18th Intl. Conf. on Data Engineering*. IEEE Computer Society, 2002, p. 117.

[4] O. Šváb, "Exploiting patterns in ontology mapping," in *Proc. of the 6th Intl. Semantic Web Conf.*, ser. LNCS, K. A. et al., Ed., vol. 4825. Berlin, Heidelberg: Springer, November 2007, pp. 950–954.

[5] D. Conte, P. P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," *Intl. J. of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 3, pp. 265–298, May 2004.

[6] C. Schellewald and C. Schnörr, "Probabilistic subgraph matching based on convex relaxation," in *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer, 2005, pp. 171–186.

[7] M. Bayati, M. Gerritsen, D. F. Gleich, A. Saberi, and Y. Wang, "Algorithms for large, sparse network alignment problems," *arXiv*, vol. 0907.3338, 2009.

[8] R. Singh, J. Xu, and B. Berger, "Global alignment of multiple protein interaction networks with application to functional orthology detection," *PNAS*, vol. 105, no. 35, pp. 12 763–12 768, September 2008.

[9] D. Conte, P. Foggia, and M. Vento, "Challenging complexity of maximum common subgraph detection algorithms: A performance analysis of three algorithms on a wide database of graphs," *J. Graph Algorithms Appl.*, vol. 11, no. 1, pp. 99–143, 2007.

[10] J. Berg and M. Lässig, "Cross-species analysis of biological networks by bayesian alignment," *PNAS*, vol. 103, no. 29, pp. 10 967–10 972, 2006.

[11] J. Flannick, A. Novak, B. S. Srinivasan, H. H. McAdams, and S. Batzoglou, "Græmlin: General and robust alignment of multiple large interaction networks," *Genome Research*, vol. 16, pp. 1169–1181, August 2006.

[12] S. Bradde, A. Braunstein, H. Mahmoudi, F. Tria, M. Weigt, and R. Zecchina, "Aligning graphs and finding substructures by message passing," *arXiv*, vol. 0905.1893, 2009.

[13] M. Bayati, D. Shah, and M. Sharma, "Maximum weight matching via max-product belief propagation," in *Intl. Symposium on Information Theory*, 2005, pp. 1763–1767.

[14] M. Mezard and R. Zecchina, "Random k-satisfiability: from an analytic solution to a new efficient algorithm," *Phys.Rev. E*, vol. 66, 2002.

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

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

[17] M. Tappen and W. Freemand, "Graph cuts and belief propagation for stereo, using identical mrf parameters," in *ICCV*, 2003.

[18] C. Yanover and Y. Weiss, "Approximate inference and protein folding," in *NIPS*, 2002.

[19] B. J. Frey and D. Dueck, "Clustering by passing messages between data points." *Science*, vol. 315, no. 5814, pp. 972–976, February 2007.

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

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

[22] Various, "Wikipedia XML database dump." http://en.wikipedia.org/wiki/Wikipedia:Database_download, Apr. 2007.

[23] W. W. Cohen, P. Ravikumar, and S. Fienberg, "A comparison of string metrics for matching names and records," in *Proc. of the KDD Workshop on Data Cleaning and Object Consolidation*, 2003.