CS224W: Project Writeup

Pierre Kreitmann

December 11, 2011

1 Introduction

Network Alignment is the problem of uncovering the relationship that exists in the nodes from two graphs which model the same phenomenon. Simply put, given two graphs $G = (V, E)$ and $G' = (V', E')$, the goal is to find a one-to-one matching $f : V \to V'$ such that for $v \in V$, $v' \in V'$, $f(v) = v'$ iff. v and v' represent the same entity.

Common interesting applications of the Network Alignment problem include the discovery of similarity in the ways protein interact in different species (a matching is found between protein-interaction networks from different species, see [Berg and Lassig, 2004]), or de-anonymizing social networks (by matching an anonymized social network with another social network for which the identities are public, see [Narayanan and Shmatikov, 2009]).

Another implication is the field of ontology matching, where one tries to build one large ontology given several overlapping ontologies. Each ontology is modeled by a graph of relationships between concepts, and thus the problem is in fact equivalent to graph alignment.

In this paper, I first give an overview of the previous work in Network Alignment. Then I explain the reasons that led me to create a new dataset to benchmark Network Alignment algorithms, and the properties of the dataset I have created. I then describe my main contribution, which is a scalable algorithm that outperforms previous work in the domain. The last section gives a detailed overview of the results obtained, and explores the reasons that caused this performance.

2 Previous Work

Historically, the first algorithms for Network Alignment have been developed for matching protein-interaction networks. There are two different types of approach: IsoRank and Belief Propagation.

2.1 IsoRank

IsoRank ([Singh et al., 2008]) builds on the PageRank algorithm to recursively define a similarity measure between nodes in V and nodes in V'. The idea is that two nodes $u \in V$ and $u' \in V'$ are similar if there are several pairs $(v, v') \in V \times V'$ such that $(u, v) \in E$ (ie. u and v are neighbors in G), $(u', v') \in E'$, and v and v' are very similar.

More precisely, we can define the similarity function R by:

$$R_{u,u'} = \sum_{v \in N(u)} \sum_{v' \in N(u')} \frac{R_{v,v'}}{|N(v)||N(v')|}$$

The relationship to the PageRank algorithm algorithm is made more clear if one defines R as a vector indexed by pairs (u, u'). Then, we can define a two-dimensional matrix A by:

$$A[u, u'][v, v'] = \begin{cases} \frac{1}{|N(v)||N(v')|} & \text{if } (u, v) \in E \text{ and } (u', v') \in E' \\ 0 & \text{otherwise} \end{cases}$$

Then the vector R is defined by $R = AR$. As in PageRank, R is then obtained by computing the principal eigenvalue of the matrix A, by the power method: R is initialized randomly, and is update with the rule $R \leftarrow \frac{AR}{|AR|}$ until convergence is reached (note that A is of size $nn' \times nn'$ here, with $n = |E|$ and $n' = |E'|$).

2.2 Belief Propagation

The Belief Propagation algorithm ([Bay, 2005]) transforms the Network Alignment problem into a maximization of a factored probability distribution. It then uses classical techniques from the field of Probabilistic Graphical Models to efficiently maximize the probability function.

It first assumes that $|E| = |E'| = n$ and that for each pair $(u_i, u'_j) \in V \times V'$, there is a weight w_{ij} that represents the prior similarity of the nodes u_i and u'_j.

For each node $u_i \in V$, it defines a factor $\phi_{u_i}(r) = e^{w_{ir}}$; for each node in $u'_j \in V'$ a factor $\phi_{u'_j}(r) = e^{w_{rj}}$. Furthermore, for each pair $(u, u') \in V \times V'$ it defines a factor ψ by:

$$psi_{u_i,u'_j}(r, s) = \begin{cases} 1 & \text{if } i = s \text{ and } j = r \\ 1 & \text{if } i \neq s \text{ and } j \neq r \\ 0 & \text{otherwise} \end{cases}$$

It then defines a probability distribution P over $2n$ variables $X = (x_1, \dots x_n)$, $Y = (y_1, \dots y_n)$ by:

$$P(X, Y) = \frac{1}{Z} \prod_{i,j} \psi_{u_i,u'_j}(x_i, y_j) \prod_i \phi_{u_i}(x_i) \phi_{u'_i}(y_i)$$

This probability distribution is maximal if the two matchings $\{(u_1, u'_{x_1}), \dots (u_n, u'_{x_n})\}$ and $\{(u_{y_1}, u'_1,) \dots (u_{y_n}, u'_n)\}$ are both one-to-one matchings. Furthermore, when it is non-zero, it is equal to

$$\frac{1}{Z} \exp\left(\sum_i w_{ix_i}\right)$$

Thus, maximizing P is equivalent to finding a matching of maximal weight. The detailed algorithm is fairly complex, and so far in my work I haven't used it extensively, which is why I think it is better to refer the reader directly to [Bay, 2005], where it is very clearly explained.

[Bayati et al., 2009] shows how this algorithm can be used to match protein-interaction networks with very high precision. However, it is important to note that the running time of the algorithm here is $\mathcal{O}(n^3)$. It is thus impossible to apply to networks of more that a thousand nodes.

2.3 Other previous work

[Kollias et al., 2011] introduces a variant of IsoRank that allows the algorithm to run much faster, while still being a good approximation of the original algorithm. The algorithm, called Network Similarity Decomposition (NSD), first uses a decomposition of the matrix A (like SVD) to approximate the power iteration. However, we'll see later that it still suffer from the same drawback as IsoRank, in that it is impossible to run for large networks.

[Narayanan and Shmatikov, 2009] tries to solve the Network Alignment problem on much larger networks than what was previously done. Specifically, they try to de-anonymize social networks: the perform a matching between an anonymized version of Twitter and a non-anonymized version of Flickr to re-identify users of the Twitter network. Given the size of the networks, they use a simple greedy approach to perform their matching: starting from a set of nodes that they have identified already, they identify the next node by choosing the match that maximizes the number of squares (a square denotes a square made by an edge of G, a matching between G and G', and edge of G', and another matching between G and G'). Their matching achieves some interesting results, but is far from a perfect accuracy.

3 Contribution

The contribution of this paper is two-fold: first I extracted a rich dataset from publicly available data that allows for reliable and flexible benchmarking of network alignment algorithms. Second, I created and implemented a scalable algorithm from graph alignment that outperforms previous work on this dataset.

3.1 New dataset

Since most of the previous work focused on protein-network alignment which tend to be small (at most 10k nodes), there is not a wide variety of large datasets to benchmark Network Alignment algorithm. Furthermore, for most of the datasets the true matching is unknown, and thus it is very hard to measure the accuracy of an algorithm.

To palliate this problem, I created two networks of Wikipedia articles with the links from articles to others, one network representing the articles in German and the other the articles in French. A matching between two nodes then represents the fact that one article is the translation of another. Then, I also collected information about the translations of each article, and built the original matching between articles.

Each of those networks has more than a million nodes. However, it is often useful to test algorithms on smaller networks to iterate more quickly. To that effect, it is possible to greedily select a set of nodes in both networks such that the original matching is still a one-to-one matching when restricted on those two subgraphs, and such that each of those networks are densely connected.

This has two big advantages: first, one can select sets of various sizes depending on the needs. Second, there is a huge variety of such sets for each size, which makes this dataset very suitable to competitions for example. One can for example create different training and test sets so that the different teams can test their algorithm on the training sets, and then their algorithm can be tested against a test set kept private.

In fact, this is exactly this approach that allowed to create the dataset for the CS224W competition. In this occasion, the dataset has proven its utility: multiple students implemented their algorithm, whose accuracy were tested against the original matching.

The dataset is not yet available online, as some work is done to include textual information. However, it is available on request.

3.2 Scalable algorithm

Here I describe the nature of my algorithm for network alignment. The algorithm has two stages: first, I create an initial matching from the partial matching by selecting nodes of high degree in one graph and matching them greedily in the other. Then, I use a Simulated Annealing optimization to further maximize the number of squares formed by the matching.

Greedy initial solution First, I search for a greedy solution using the following algorithm: at each iteration, I consider the set V of nodes in the French Wikipedia that already have a match in the German Wikipedia. At the initial step, V is the set of matches that are provided. I compute the set V' of nodes that are neighbors of V (either because of in-edges or out-edges), and I pick the node of highest degree in V' (this proves to be more efficient than picking a random node).

Then, I select the best match for this node in the German Wikipedia. This is the node that adds the largest number of squares to the current matching. I then repeat until all nodes are matched. This approach is very similar to the approach proposed by [Narayanan and Shmatikov, 2009], the difference being that they pick a random node in the French graph while I pick the nodes with highest degree. The intuition behind that choice is that nodes of highest degree have the potential to add more squares, and setting them early avoid for them to be poorly matched because their real match has already been selected for another node with a lower number of squares.

Simulated Annealing I then use a Simulated Annealing algorithm to improve the solution found. The elementary operation is a swap between two edges (i, i') and (j, j') in the matching, which results in two new edges (i, j') and (j, i'). Note that the result of the greedy approach might not in fact be a local maxima. Indeed, in the greedy phase we match each French node to the best possible German node (the German node being chosen greedily), but nothing guarantees that no swap could improve the solution. In fact, it turns out that greedily swapping edges instead of performing a simulated annealing gives almost as good results as the simulated annealing, but takes much longer however.

My Simulated Annealing performs 5000 potential swaps at each iteration. It performs each swap if it increases the quality of the solution, or if a certain acceptance criteria is met. More precisely, is δ_{ij} denotes the increase in number of squares obtained by swapping i and j, the swap is accepted if $\delta_{ij} > 0$ or if

$$r \leqslant \exp\left(\frac{\delta}{k_b T}\right)$$

where r is a random number picked uniformly between 0 and 1.

The temperature T decreases over time. I perform several runs with various values for the initial parameter k_b. This corresponds to the Boltzmann constant: if k_b is big, then the algorithm starts by swapping a lot of edges even if this decreases the number of squares. If k_b is small, the probability to swap an edge for which $\delta_{ij} < 0$ is small. The algorithm stops when too many iteration didn't lead to an improvement of the number of squares.

4 Results

4.1 Evaluation of the Simulated Annealing approach

4.1.1 The Initial Greedy Step

The greedy step of the algorithm is very similar to the algorithm described in [Narayanan and Shmatikov, 2009]; the only difference is that I trye to match the nodes nodes with high degrees first, while [Narayanan and Shmatikov, 2009] matches nodes in random order.

My approach proves significantly better in practice: the random strategy creates a matching with about 38000 squares, while my approach's resulting matching has 76000 squares. As a comparison, the initial partial matching has about 4000 squares, and the original matching has about 100000 squares. Furthermore, my approach leads to 1317 correct edges, while choosing random edges to match only produces 1175 correct edges.

4.1.2 The Complete Pipeline

To evaluate my algorithm, I used IsoRank as a baseline. The matrix A is of size $n^2 \times n^2$, which is prohibitive. It is sparse, but the number of non-zero coefficients is about $n^2 d^2$, where d is the average degree. For $n = 4000$ and $d \simeq 170$, this is impossible to store in memory. Thus, my implementation doesn't use the matrix formulation, but rather updates $R_{uu'}$ for all pairs (u, u'), using the formula:

$$R_{u,u'} = \sum_{v \in N(u)} \sum_{v' \in N(u')} \frac{R_{v,v'}}{|N(v)||N(v')|}$$

Furthermore, my implementation runs those updates in parallel to speed-up the computation. I then use a computer with 64 parallel threads to run the algorithm, which gives a 55× improvement over the sequential implementation. Indeed, one can easily assign each node u to a thread, which will update all the values $R_{u,u'}$, for all $u' \in V'$. I use a double buffer to store $R_{u,u'}$: one for reading and one for writing. The buffers are swapped after each iteration. Thus, no complex synchronization is required, as the threads read concurrently but write at different locations that do not influence the reads.

In order to use the partial matching, I only update $R_{uu'}$ if neither u nor u' are in the partial matching. If one of them is, then the coefficient $R_{uu'}$ is set to 1 of u and u' are a match, and 0 otherwise. In order for the normalization to remain relevant with those fixed values, I choose to normalize the vector R with the $\| \cdot \|_0$ norm, which is simply the largest coefficient.

	Number of Squares	Accuracy (%)	Time (s.)
IsoRank	70101	28.47	18448
First Greedy Step	76000	16.16	6
Sim-Ann	103189	**76.97**	8885
Original Matching	100448	100	

Table 1: Results for IsoRank and Simulated Annealing

Comparison of the accuracies Table 1 shows the results in terms of accuracy (number of unknown edges that are correct), number of squares, and computational time, for IsoRank, the Simulated Algorithm, and also for the

first greedy step of the Simulated Annealing. The original matching is given as a reference.

First, as expected Simulated Annealing is the best in terms of number of squares. This is not surprising, as it explicitly tries to maximize this quantity. More surprising is the results in accuracy: IsoRank has a very low accuracy, while Simulated Annealing performs very well with almost 77% of the unknown edges that were recovered.

Furthermore, in terms of computational time Simulated Annealing is slightly faster than IsoRank. However, it is important to consider that IsoRank was parallelized on 64 threads while Simulated-Annealing was implemented sequentially. A parallel implementation would increase the difference.

Convergence It is also interesting to note that both IsoRank and Simulated Annealing can be stopped before convergence. For larger graphs, this is a useful feature as it will often take too long to converge, and if the algorithm is able to find a good enough matching early in the computation it can be stopped earlier.

Therefore, we measure the accuracy and the number of squares of the matching during the computation for the two algorithms. Figure 1 shows the evolution of the accuracy and the number of squares during IsoRank. After an initial period where the two quantities oscillate, the evolution becomes smoother, and converges after about 25 iterations.

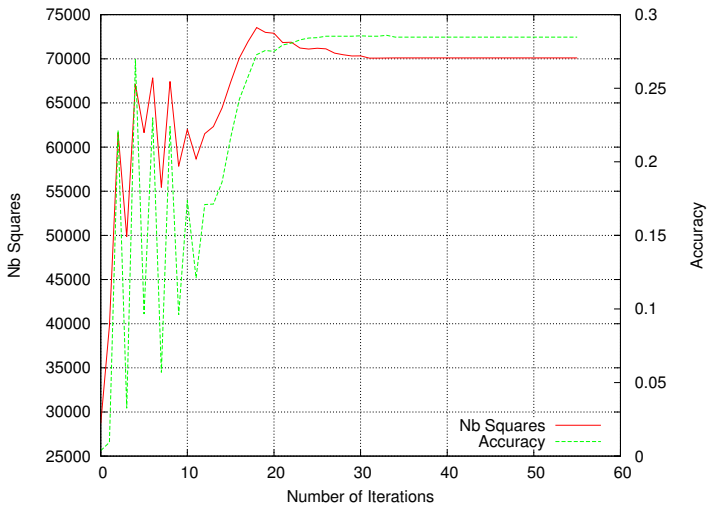

Figure 1: Evolution Number of squares and Accuracy (% of preserved edges) during the iterations of IsoRank

Figure 2 shows the same data for the Simulated Annealing algorithm. Keep in mind that that the number of iterations have no common scale in the two

algorithms: while each iteration takes more than 5 minutes for IsoRank, Simulated Annealing's iterations take less that a second.

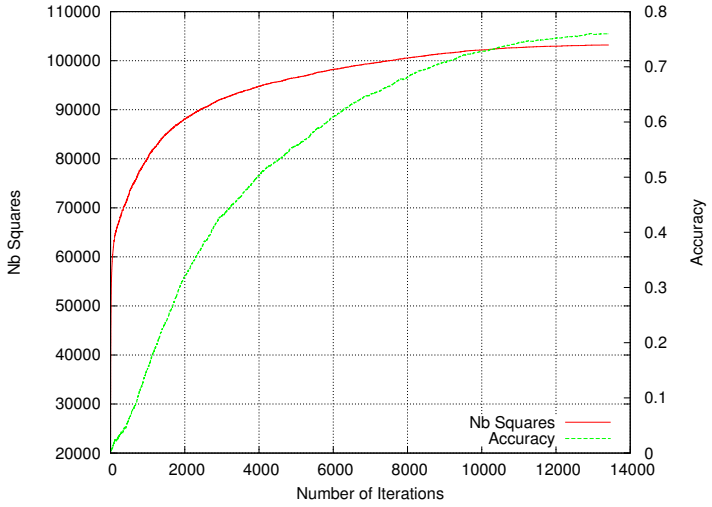

Figure 2: Evolution Number of squares and Accuracy (% of preserved edges) during the iterations of Simulated Annealing

During Simulated Annealing, the evolution of the number of squares and of the accuracy are very smooth: they slowly converge to their maximum without iterating. This means that the algorithm can be used even if the time needed to converge is arbitrarily big: it stars improving the accuracy after the first iteration. This is particularly important for very large graphs, where waiting for convergence would be impossible. While IsoRank performs very badly during the first iterations, Simulated Annealing is more reliable.

4.2 Evaluation of the Number of Squares heuristic

4.3 Overfitting

Table 1 shows that the Simulated Annealing algorithm, which tries to maximizes the number of squares, finds a matching with more squares than the original matching. While it seems intuitive that maximizing the number of squares is a good heuristic, which is why it is used as a measure of performance in previous work, the relationship between the number of squares and the accuracy is unclear.

Since our dataset contains the original matching, we can measure the accuracy of the matchings obtained at each iteration of both IsoRank and Simulated Annealing.

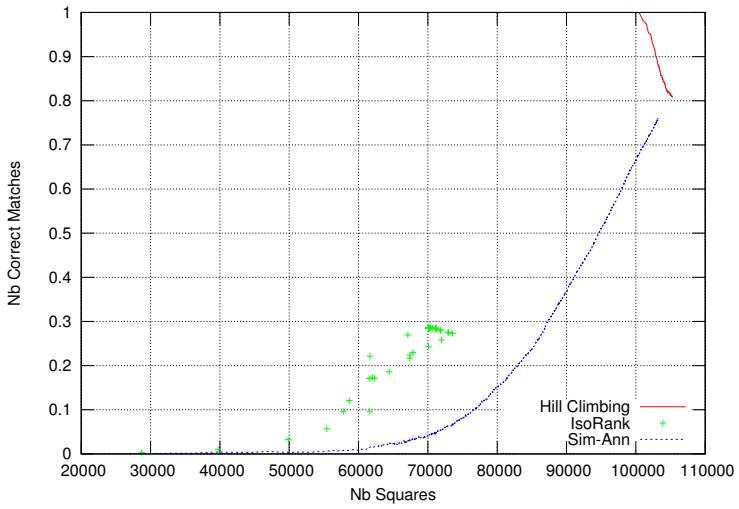

Figure 3: Accuracy (% of correct matches) as a function of the number of squares. Each dot corresponds to a matching, the color represents the algorithm that found the matching.

Figure 3 shows the relationship between the number of squares and the proportion of correct matches. It contains the matchings obtained during the iterations of Simulated Annealing and of IsoRank. Also, we take the original matching and use it as a starting point for a greedy search, during which we swap all pairs (u, u') such that $\delta(u, u') > 0$ until no such pair exists (this is equivalent to our Simulated Annealing with $k_b T = 0$).

This indicates that algorithms that try to maximize the number of squares risk to overfit, ie. to produce matchings with a very high number of squares but a low accuracy. However, Figure 2 shows that the accuracy increases at each iteration for Simulated Annealing. Thus, it seems that even though it is theoretically possible that the algorithm starts to overfit after some time, in practice the search is stuck in a local maxima area where the most accurate matching is also the biggest number of squares.

This is not the case for IsoRank: for equal number of squares, IsoRank's matchings are much more accurate than the ones obtained with Simulated Annealing. This is because IsoRank is designed to find a matching that has a meaning, rather than to maximize the number of squers.

In fact, Figure 1 shows that during the iterations of the algorithms, the number of squares stops increasing after iteration 18, while the accuracy keeps increasing after that. In that regard, it seems that IsoRank is less susceptible to overfitting.

4.4 Local structure of the squares

So far we have studied the heuristic of squares globally, ie. on a whole matching. Here, we study the distribution of the squares on the network matched: we study which nodes create the biggest number of squares, depending on their degree. We compare this distribution on the original matching, on random matchings and on matchings obtained with the Simulated Annealing algorithm.

Figure 4 shows various plots which all represent the fraction $\frac{s}{d}$ versus d, where s is the number of squares created by a node, and d is the degree of the node. Let us denote this fraction by $\tilde{s}$: $\tilde{s} = \frac{s}{d}$. Each plot corresponds to a different matching or to different French and German graphs (see below).

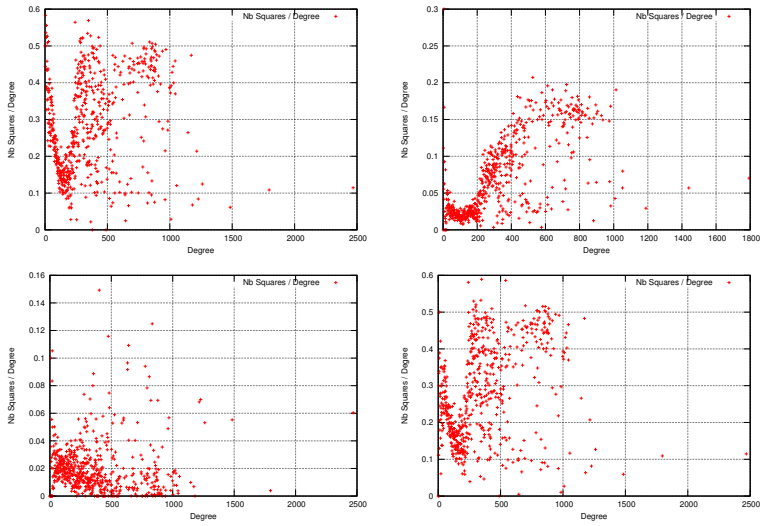

Figure 4: **All figures**: $\tilde{s} = \frac{s}{d}$ as a function of d. Each point is a node in the French graph. **Top-Left**: original matching. **Top-Right**: After random shuffling of the edges of the French and the German graph. **Bottom-Left**: Original French and German graph, random shuffling on the matching. **Bottom-Right**: Matching obtained with Simulated Annealing.

On the top-left, the original matching. There are two different trends: first, a clear relationship that the fraction $\frac{s}{d}$ grows with d for degrees above 200. Second, for $d < 200$ there is an opposite relationship: $\tilde{s}$ decreases as d increases. In fact, the shape of the curve is roughly similar to $\frac{1}{d}$, which means that s is a constant. This means that roughly, for edges with small degrees the number of squares is a constant that doesn't vary much with the degree.

On the top-right plot, we shuffle the edges of the French graph and of the German graph before plotting the data (each node keeps the same in and out-degree, only the edges are shuffled). Even though one could expect a random scatter plot, it seems that the structure is roughly preserved. The quantity $\tilde{s}$

is of course lower than for the original matching, but the two trends are still present. This means that the degree of each node is in itself an important quantity for the matching, and that the edges are less relevant.

The bottom-left plot confirms that intuition: it is plotted after keeping the original French and German networks, but choosing a random matching. Here, we see a truly random set of points, and the previous trends are almost non-existent.

Finally, the bottom-right plot uses the matching computed by the Simulated Annealing algorithm. The similarity with the original matching is striking. In fact, for nodes of degree higher than 500 only a few points differ in the two plots. The major difference is in fact for the nodes of small degrees. While, in the original, they had a high $\tilde{s}$ (above 0.4), for the Simulated Annealing matching they always have $\tilde{s} < 0.4$.

This means that the Simulated Annealing is excellent at finding matches between nodes of high degree, but does not perform well on low-degree nodes. Since the degree distribution follows a power law (not shown here, but it does), this penalizes the Simulated Annealing very much in terms of accuracy. However in terms of number of squares this is not penalizing, since nodes of small degree account for a small minority of the squares. This is why our algorithm is able to achieve a great number of squares while still being far from the perfect accuracy.

Conclusion

The Wikipedia dataset has already proven very useful in two contexts: during CS224W 2011 annual competition, where it was used to benchmark student's algorithms, and in the context of this study, where its unique feature – the fact that the original matching exists and is available – allowed me to explore the relationship between number of square and accuracy based on data.

The Simulated Annealing approach, even though very simple in its implementation, performs very well on the Wikipedia dataset. Furthermore, it is designed with scalability in mind, and can easily be adapted to deal with much larger datasets. It can also be used as a post-processing tool, without waiting for full convergence, as it improves the quality of a given matching only after a few iterations. Finally, it isn't very susceptible to overfitting, as its limit is in fact that it tends to stay stuck in local maxima.

The limits of this approach is that it relies on the heuristic of the number of squares. While this proved successful on this dataset, it is unclear if all datasets exhibit this property. Furthermore, it fails to match nodes with low degree, which are very abundant in graphs where the degree distribution follows a power law.

I believe that further research should focus on using Simulated Annealing in conjunction with features of the nodes themselves, or a-priori weights of the matches (u, u') for $u \in V$ and $u' \in V'$.

References

[Bay, 2005] (2005). Maximum weight matching via max-product belief propagation.

[Bayati et al., 2009] Bayati, M., Gerritsen, M., Gleich, D. F., Saberi, A. and Wang, Y. (2009). Algorithms for Large, Sparse Network Alignment Problems. Data Mining, IEEE International Conference on *0*, 705–710.

[Berg and Lassig, 2004] Berg, J. and Lassig, M. (2004). Local graph alignment and motif search in biological networks. Proc Natl Acad Sci U S A *101*, 14689–94.

[Kollias et al., 2011] Kollias, G., Mohammadi, S. and Grama, A. (2011). Network Similarity Decomposition (NSD): A Fast and Scalable Approach to Network Alignment. IEEE Transactions on Knowledge and Data Engineering *99*.

[Narayanan and Shmatikov, 2009] Narayanan, A. and Shmatikov, V. (2009). De-anonymizing Social Networks. Security and Privacy, IEEE Symposium on *0*, 173–187.

[Singh et al., 2008] Singh, R., Xu, J. and Berger, B. (2008). Global alignment of multiple protein interaction networks with application to functional orthology detection. Proceedings of the National Academy of Sciences of the United States of America *105*, 12763–12768.