Pacific Symposium on Biocomputing 15:123-132(2010)

LOCAL OPTIMIZATION FOR GLOBAL ALIGNMENT OF PROTEIN INTERACTION NETWORKS

LEONID CHINDELEVITCH CHUNG-SHOU LIAO * BONNIE BERGER †

Department of Mathematics and Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139.
E-mail: {leonidus, shou, bab}@csail.mit.edu

We propose a novel algorithm, PISwap, for computing global pairwise alignments of protein interaction networks, based on a local optimization heuristic that has previously demonstrated its effectiveness for a variety of other NP-hard problems, such as the Traveling Salesman Problem. Our algorithm begins with a sequence-based network alignment and then iteratively adjusts the alignment by incorporating network structure information. It has a worst-case pseudo-polynomial running-time bound and is very efficient in practice. It is shown to produce improved alignments in several well-studied cases. In addition, the flexible nature of this algorithm makes it suitable for different applications of network alignments. Finally, this algorithm can yield interesting insights into the evolutionary history of the compared species.

Keywords: Network alignment; Local optimization; Functional orthology

1. Introduction

Ever since high-throughput experimental screening techniques such as yeast two-hybrid analysis,[1] mass spectrometry,[2] and TAP[3] made protein interaction networks available for several species, efforts have been made in the bioinformatics community to extract useful biological information from these networks. One important goal has been to produce accurate alignments of two or more of these networks, with the expectation that this would help in establishing the biological function of unknown proteins by exhibiting their correspondence with the proteins of another species with known biological function, and providing insight into evolutionary dynamics.

Algorithms for network alignment can be broadly separated into two distinct categories: *local* and *global* algorithms. The distinction is similar to the one made for sequence alignment algorithms. More specifically, local network alignment[4] is concerned with identifying a subnetwork of one species closely matching a subnetwork of another species (or having a given topology). Typically, multiple closely matching subnetworks are identified by such algorithms, which may be mutually inconsistent.[5] On the other hand, global network alignment algorithms[5] attempt to match two or more networks as a whole, and their output is a single mapping between the nodes of the networks. In the present paper, which deals with the global alignment problem, we view this mapping as a bipartite matching, where the nodes on one side of the matching are the proteins from one network, and the nodes on the other side, the ones from the other network.

1.1. *Contribution*

We propose a novel method, based on local optimization, for computing pairwise alignments of protein interaction networks. The method begins by identifying an optimal alignment based purely on sequence data, which correctly determines functionally orthologous proteins in many, but not all, cases. In order to adjust this initial alignment, the algorithm uses the intuition that conserved interactions can compensate for matching proteins whose sequences are not particularly similar to one another. In this way, the topology of the networks is taken into account, and information is propagated from each node to its neighbors.

Using the protein interaction networks available for three species: yeast, fly, and worm, we perform pairwise alignments on each pair. The results compare favorably to those produced by other methods. Furthermore, we suggest that the algorithm can be used to produce several alignments with almost no

*Also at the Institute of Information Science (IIS), Academia Sinica, Taipei 115, Taiwan.
†Corresponding Author

Pacific Symposium on Biocomputing 15:123-132(2010)

additional cost, or fine-tuned by combining it with other algorithms for partitioning or clustering protein interaction networks. Finally, we explain how specific information produced by this algorithm can produce insights into the evolutionary dynamics of protein interactions.

1.2. *Related Work*

A number of techniques for the problem of PPI network alignment exist. These include NetworkBLAST-M,[6] Graemlin 2.0,[7] IsoRank,[5] IsoRankN[8] and PATH,[9] though a number of other techniques exist as well.[4,10–15] Some of these methods can handle more than two networks. NetworkBLAST-M computes a local alignment by greedily finding regions of high local conservation based on inferred phylogeny. Graemlin 2.0, in contrast, computes a global alignment by training how to infer networks from phylogenetic relationships on a known set of alignments, then optimizing the learned objective function on the set of all networks. IsoRank uses spectral graph theory to first find pairwise alignment scores across all pairs of networks, obtained by a spectral method on a product graph; IsoRank then uses these scores in a greedy algorithm to produce the final alignment. The more recent IsoRankN uses a different method of spectral clustering on the induced graph of pairwise alignment scores. As pointed out in a recent paper,[9] one of the main difficulties faced by network alignment algorithms is the lack of an accurate and reliable gold standard for evaluation purposes.

2. Problem Formulation

We consider the global alignment of a pair of protein-protein-interaction (PPI) networks. Each network is represented by a graph whose vertices correspond to proteins, and there is an undirected edge between two vertices if the corresponding proteins are found to interact with each other.

Given a pair of PPI networks and a list of pairwise sequence similarities between proteins in the two networks computed according to some criterion, the specific aim of global alignment is to find a mapping between the proteins of the two networks that best represents conserved biological function. We formulate this network alignment problem as a graph-theoretical problem.

Let $G_X = (X, E_X)$ and $G_Y = (Y, E_Y)$ be two PPI networks in which $e_x = (x, x') \in E_X$ and $e_y = (y, y') \in E_Y$, with $x,\ x' \in X$ and $y,\ y' \in Y$, represent interaction between two proteins in G_X and G_Y respectively.

Suppose $G = (X \cup Y, E)$ is an edge-weighted bipartite graph in which each edge $e = (x, y) \in E$ is associated with a nonnegative edge weight $s(e)$, which represents the sequence similarity between $x \in X$ and $y \in Y$. That is, $s : E \to \mathrm{Z}^+$ denotes a sequence similarity function on the edges of G, where sequence similarity on an edge, a pair of proteins, could be, for instance, the BLAST Bit-value of the sequences as retrieved from Ensembl.[19]

A matching $M \subseteq E$ of G is defined to be a subset of edges such that no two edges in M share an endpoint. In addition, given a matching M, we let a function $t : E \to \mathrm{Z}^+$ be a topology similarity function with respect to M if for an edge $e = (x, y)$, $t(e)$ represents the topology similarities between the neighborhoods of $x \in X$ and $y \in Y$. The topology similarity function represents the number of edges in G_X and G_Y conserved by the matching M.

More precisely, for an edge $e = (x, y)$, $t(e)$ is the number of edges between the neighborhoods of x and y, $N(x)$ of G_X and $N(y)$ of G_Y respectively, which are also in the matching M, i.e. $t(e) = |\{(x', y') \in M | x' \in N(x) \text{ and } y' \in N(y)\}|$.

Our objective is to find a matching M that maximizes the following weight function w:

$$w(M) = \sum_{e \in M} \{\alpha t(e) + (1 - \alpha) s(e)\}, \tag{1}$$

where $\alpha \in [0, 1]$ is a parameter controlling the importance of the network topology similarities relative to sequence similarities. This is equivalent to the objective function introduced by Singh et al.[5]

Pacific Symposium on Biocomputing 15:123-132(2010)

The above weight function contains two terms: the topology similarity function t and the sequence similarity function s. Tuning the parameter α allows us to change the relative importance of PPI network data in finding the optimal global alignment. For instance, $\alpha = 0$ implies no network data will be used, while $\alpha = 1$ indicates only network data will be used.

We will use a local search approach to pairwise PPI network alignment in a manner similar to `2-Opt` as follows: when given a maximum weighted bipartite matching M^* in $G = (X \cup Y, E)$, we define a subset $\text{prefer}_Y(x)$ for each $x \in X$, which consists of the c highest-weighted neighbors of x in Y (where the weight of a neighbor of x is given by its sequence similarity to x). Similarly, for every $y \in Y$, a vertex subset $\text{prefer}_X(y) \subseteq X$ is similarly defined to consist of the c highest-weighted neighbors of y in X. Here, c is some relatively small integer chosen ahead of time. It can be shown[20] that $c = 20$ suffices for most practical applications.

Our aim is to repeatedly find a candidate $e' = (u, v)$, $v \in \text{prefer}_Y(x)$, $u \in \text{prefer}_X(y)$ to swap with $e = (x, y)$, where e, $e' \in M^*$, such that the weight of the new matching, $w((M^* \setminus \{e, e'\}) \cup \{e_1, e_2\})$, where $e_1 = (x, v)$ and $e_2 = (u, y)$ are the edges obtained by swapping e and e', is higher than $w(M^*)$.

Just as in IsoRank,[5] we seek to maximize an objective function consisting of two terms, one accounting for sequence similarity between the proteins that get matched to each other, and the other one, for the number of interactions preserved by the matching. Our search space is the set of all matchings. We use the parameter $\alpha \in [0, 1]$ to control the relative importance of the topology information with respect to the sequence information. This formulation of the problem is known to be NP-hard, and even APX-hard,[16] which means that it does not admit a polynomial-time approximation scheme unless P = NP. Since the size of the search space (i.e. the number of possible matchings) grows exponentially with the number of proteins in each network, we use a local search technique adapted from other NP-hard optimization problems, which is described in detail in the following section.

3. Algorithmic results

The main idea of our algorithm for the global alignment of pairwise PPI networks is a two-phase approach to searching for a matching that maximizes our objective function. The special case of our problem with $\alpha = 0$ (i.e. one where only sequence information is used) is a variant of the *linear assignment problem*, which is that of finding a maximum-weight matching in a bipartite graph.

Hence the first stage of our algorithm finds a maximum-weight matching in the bipartite graph obtained by joining pairs of proteins in the two networks, where the weight of an edge is given by the sequence similarity of the two proteins that it joins. This matching can be obtained by the well-known *Hungarian algorithm* in polynomial time[24] and sped up with extensive use of priority queues and decomposition techniques.[23]

In the second stage of our algorithm, we apply the local search method, which is widely used in the combinatorial optimization field, to iteratively improve the initial matching while taking into account both the sequence score and the topology score of the matching.

From a variety of local search methods, we make use of the `2-Opt` algorithm, which was first proposed by Croes.[18] The `2-Opt` algorithm is one of the most famous heuristics for the well-known *Traveling Salesman Problem*.[25] Given a set of cities, the Traveling Salesman Problem is to find an ordering of cities that minimizes the total length of the tour when visiting all the cities in some order and returning to the starting city. The basic concept of the `2-Opt` algorithm is simple. A move deletes two edges of the original tour, thus breaking the tour into two paths, and then reconnects those paths by swapping these edges.

Local search algorithms do not appear to perform well from a theoretical point of view. Papadimitriou and Steiglitz[27] have shown that no local search algorithm (like `2-Opt`) that takes polynomial time per move can guarantee a constant approximation ratio for TSP unless P = NP. In addition, it has been shown that a sequence of exponential moves might be required by `2-Opt` before halting[26] and an analogous result[17] has been extended to `3-Opt` and k-`Opt`.

Although the worst-case analysis of the `2-Opt` algorithm is pessimistic, the average-case analysis is considerably more optimistic. A significant discovery[17] has shown that the expected approximation ratio to

Pacific Symposium on Biocomputing 15:123-132(2010)

the optimum is bounded by a constant. A similar improvement with respect to running time has been obtained as well. That is, the expected number of moves is polynomially bounded. Furthermore, `2-Opt` outperformed almost all the local search and greedy algorithms in experimental results for TSP.[20] More precisely, `2-Opt` (or k-`Opt`) gave better final tours than other local search algorithms for TSPLIB instances[20] with respect to both approximation ratio and running time.

The technique of iterative edge swaps is, however, not limited to TSP. It is also the basis of an algorithm for graph randomization, which attempts to produce a random graph with a given degree distribution.[21] It can be shown[22] that the corresponding Markov chain converges to the uniform distribution on the set of all connected simple graphs with the given degree distribution, thus giving an exact algorithm.

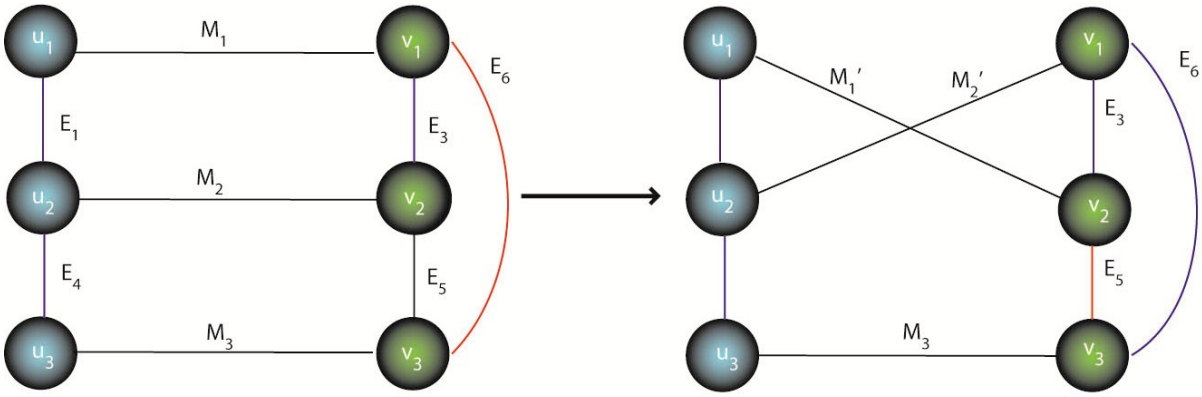

Fig. 1. The situation before and after a possible edge swap. The edges M_1 and M_2 of the matching are swapped out and replaced by $M_1^{'}$ and $M_2^{'}$. The matching is illustrated in black, conserved edges are in blue, and non-conserved edges are in red.

Figure 1 illustrates a matching on a pair of small networks transformed by an edge swap. It is important to note that an edge swap generally changes the conserved edges, provided these edges are incident to (i.e. share a vertex with) each of the two edges being swapped.

We now present the pseudocode of our algorithm. Its running-time is analyzed in the Appendix.

Algorithm 3.1. ***PISwap***

Given a weighted bipartite graph $G = (X \cup Y, E)$ with a maximum weighted matching M^* and parameters α and c,

(1) Compute topology similarities $t(e)$ and weight $w(e)$ for each edge $e \in M^*$, where the weight $w(e) = \alpha t(e) + (1-\alpha)s(e)$;

(2) Find a candidate set S consisting of every edge $e = (x, y) \in M^*$ which satisfies the following condition: There is $e' = (u, v) \in M^*$ with $v \in \text{prefer}_Y(x), u \in \text{prefer}_X(y)$ such that, for $e_1 = (x, v)$, $e_2 = (u, y)$,

$$swap(e, e') := \{w(e_1) + w(e_2) + \alpha(t(e_1) + t(e_2))\} - \{w(e) + w(e') + \alpha(t(e) + t(e'))\} > 0; \qquad (2)$$

(3) ***while*** *$S \neq \emptyset$*
do

3-1. Select the pair of edges $e = (x, y) \in S$ and $e' = (u, v)$ which achieve the maximum value of $swap(e, e')$;

3-2. Swap e, e' with $e_1 = (x, v)$, $e_2 = (u, y)$ to obtain a new matching $(M^* \setminus \{e, e'\}) \cup \{e_1, e_2\}$ and $S = S \setminus \{e, e'\}$;

3-3. Verify if the new inserted edges e_1, e_2 satisfy the condition (2) above and put them into S if necessary;

3-4. Update the topology similarities $t(e)$ for each edge $e \in M^*$ with one endpoint in $N(x) \cup N(u)$ and the other, in $N(y) \cup N(v)$ and modify $w(e)$ and $swap(e, e')$;

end while

(4) Output the final matching M^*.

Pacific Symposium on Biocomputing 15:123-132(2010)

4. Implementation details

The algorithm was implemented in Python 2.5.4[31] using the NetworkX[32] package, as well as Joris van Rantwijk's GPL implementation of the maximum-weight matching algorithm based on the blossom method for finding augmenting paths and the primal-dual method for finding a matching of maximum weight.[33] All experiments were performed on a Lenovo laptop with a 64-bit architecture running Windows Vista with an Intel Core 2 Duo T9600 CPU and 4 GB of RAM.

5. Experimental results

In spite of the analysis presented in the Appendix, the running-time of our algorithm is actually dominated by the preprocessing step, that of finding a maximum-weight bipartite matching. This is because the running time is cubic in the size of the input, whereas the number of iterations in the main loop of the algorithm was typically between 40 and 120. Thus, the upper bound presented in the analysis is overly pessimistic. We used the value $c = 200$ for our experiments, since this was close to the maximum degree Δ of the input networks (which was 190, 184 and 323 for C.elegans, D.melanogaster and S.cerevisiae, respectively).

In terms of actual time, the initial matching took between 5 and 10 minutes to compute on a single-processor laptop, but it was then saved and used with multiple values of α and c. Each of the runs of the second stage of the algorithm took no more than 15 to 20 seconds to produce the final mapping.

The ratio of α to $1 - \alpha$ can be thought of as giving the relative weight of sequence information to topology information. We found that choosing α to make this relative weight half of what it is for the initial mapping was a good rule of thumb. In other words, since the original mapping was based purely on sequence information, we chose α to favor the topology twice as much as the original mapping did, in all the reported experiments. We used both raw BLAST scores and normalized BLAST scores, retrieved from the IsoRank website.[34] The raw BLAST scores used were computed as $s(i, j) = B(i, j) + B(j, i)$, where $B(i, j)$ is the value given by BLAST on input i and j (this is because BLAST may occasionally produce "asymmetric" results). The normalized scores were computed as $\frac{s(i,j)}{\sqrt{s(i,i)s(j,j)}}$, and resulted in values between 0 and 1. Although our algorithm is described as dealing with integers, it is actually capable of handling fractional weights (although the running-time bound proved in the Appendix does not hold in that case). Also, it is important to note that since the initial matching was a maximum-weight matching, not all the proteins which had a non-zero similarity to some protein in the other species ended up in the matching — the size of the matching was roughly $0.9 \min(|V_1|, |V_2|)$, or 90% of the largest possible matching. We chose not to force matches between proteins with zero sequence similarity to all proteins in the other species, since topology alone does not suffice to create a reliable matching between them.[5]

Table 1 illustrates the results for the unnormalized input data (the results for the normalized data are qualitatively very similar). It clearly shows that PISwap compares favorably to other algorithms. The abbreviations denote the species used: CE = C.elegans, DM = D.melanogaster, SC = S.cerevisiae. The figures for the initial matching are provided for comparison purposes only, and it is the final alignments that are output by the algorithm.

The number of swaps required was 97 (102) for that between D.melanogaster and S.cerevisiae, 38 (54) for that between C.elegans and S.cerevisiae, and 71 (85) for the mapping between C.elegans and D.melanogaster. In each pair, the first number indicates the unnormalized case and the second, the normalized case. Note that these are significantly higher than the diameter (length of the path between the two most distant nodes) of the networks, which are 14, 11 and 9 for C.elegans, D.melanogaster and S.cerevisiae, respectively, meaning that information may have had a chance to propagate to all of the nodes.

To evaluate the output, we first used the HomoloGene database[35] which is thought to contain a reliable orthology mapping, though mainly based on sequence similarity between the proteins and the DNA regions that code for them.[35] We also looked at the number of conserved interactions. It is interesting to note that, compared to the initial matching, the final matching contains the same number of functional orthologs based on HomoloGene as the gold standard (except in the case of C.elegans and D.melanogaster, where the pair

Pacific Symposium on Biocomputing 15:123-132(2010)

Table 1. Evaluation of alignments based on unnnormalized sequence data

	DM-SC alignment		CE-SC alignment		CE-DM alignment	
	Initial	Final	Initial	Final	Initial	Final
Number of swaps	0	97	0	38	0	71
HomoloGene pairs	51	51	41	41	819	818
Conserved edges	272	398	106	150	80	154
Functional Coherence	N/A	0.510	N/A	0.172	N/A	0.355

(CO5C8.6, CG1826) becomes lost in the swapping process), while at the same time having significantly more conserved interactions (on average, 60% more). This shows that our algorithm is indeed performing its goal of achieving a high topology similarity while retaining an excellent sequence similarity. For comparison, let us mention that the D.melanogaster - S.cerevisiae alignment produced by the five algorithms (MP,[28] MRF,[29] IsoRank,[5] GA[9] and PATH[30]) studied by Zaslavskiy et al.[9] produce between 36 and 41 orthologous pairs from HomoloGene, compared to 51 for our algorithm (although the additional constraints imposed on the problem studied there make it difficult to perform a fair comparison).

In addition, we computed the Functional Coherence values, following the method outlined by Singh et al.[5] The method for computing Functional Coherence can be summarized as follows. First, the GO terms corresponding to each protein are collected. Then, each GO term is mapped to a subset of the so-called standardized GO terms, which in this case are its ancestors at a distance 5 from the root of the GO tree. Finally, the similarity between each pair of aligned proteins is computed as the median of the fractional overlaps of their corresponding sets of standardized GO terms. The Functional Coherence of an alignment is defined as the average pairwise Functional Coherence of the protein pairs that it matches. The values for the D.melanogaster-S.cerevisiae alignment produced by our algorithm was 0.510, comparable to the values of 0.519, 0.509 and 0.522 produced on the same input by IsoRank, GA and PATH, respectively.[9]

6. Evolutionary model

Although the edge-swapping technique was originally inspired by the field of combinatorial optimization, one can speculate that it can actually give us insights into the way two networks evolved from a common ancestor. If the networks of two closely related species are being analyzed, it is conceivable that at the outset, the proteins of the two networks are essentially identical in sequence, and hence, their correspondence can be determined exclusively on the basis of sequence information. Suppose, however, that as the two species evolve, a pair of proteins in one of the species have traded functions with one another. In that case, reconstructing the initial correspondence would require precisely an edge swap.

Comparing the network alignment problem to the (simpler) sequence alignment problem, one could say that edge swaps at the network level are the analog of compensatory mutations at the sequence level. One could then argue that, just as compensatory mutations can provide important clues for the evolutionary history of the sequences, function exchanges (represented by edge swaps) can provide important indications for the evolutionary history of the protein interaction networks. Unfortunately, function exchanges are much more difficult to detect than compensatory mutations, since network data is noisy, incomplete and unreliable.[5] Nevertheless, an algorithm such as PISwap could be adapted to estimating the number of function exchange events that have taken place during the evolutionary process.

While evolutionary events other than exchanges of function, such as duplications, insertions and deletions of proteins, certainly take place in biological networks,[14] this approach can still yield useful biological insights. In addition, the evolutionary distance between two species could in principle be computed from the number of evolutionary events (including function exchanges) that have taken place, and could perhaps provide a

more accurate estimate than the (appropriately defined and weighted) edit distance between two orthologous sequences present in those two species, since it would in some sense encompass all the protein sequences at once.

7. Discussion

We presented an algorithm for performing a global alignment of two protein interaction networks. In this algorithm, the parameter α plays an important role because it determines the relative importance of the topology data and the sequence data. Although our objective function is identical to that used in the IsoRank algorithm,[5] there are a number of important differences. IsoRank performs a random walk on the graph $G = G_X \otimes G_Y$, the tensor product of the two networks, where at each step, the walk is restarted with probability $1 - \alpha$ at a node $v = (x, y)$ in G chosen at random from the distribution proportional to the sequence similarity $s(x, y)$.[36] On the other hand, our algorithm can be thought of as performing a walk on the set of all matchings in the complete bipartite graph, and this walk is not random, but has the property that every step increases the value of the objective function. Another difference from IsoRank is that the output of IsoRank in terms of the pairwise alignment scores R_{ij} changes continuously with α, whereas in our algorithm, the set of possible matchings is discrete and it is clear that the interval $[0, 1]$ can be subdivided into non-overlapping subintervals such that on each one, the resulting matching is the same. Finally, our algorithm is not based on a spectral method, unlike both IsoRank and IsoRankN.

It is conceivable to obtain an "ensemble" or "fuzzy" mapping by using the mappings produced with several different values of α. The common part of these mappings would provide a more reliable set of functional orthologs, while the differences between the mappings could be used to evaluate the confidence of the assignments produced. In order to apply that strategy, it suffices to compute a single maximum-weight bipartite matching, which is the more time-intensive part of the algorithm, and save it in order to be able to reuse it with different values of the parameter α.

In order to speed up the algorithm without degrading its performance, it is possible to combine it with preprocessing by a graph-partitioning algorithm. For instance, if the two networks, G_X and G_Y, are each partitioned into two pieces (for instance, via a min-cut algorithm[37]), say, (G_X^1, G_X^2) and (G_Y^1, G_Y^2), one could apply the algorithm on each pair of pieces and then find the one that works best. Because there would be few edges between G_X^1 and G_X^2, as well as between G_Y^1 and G_Y^2, the difference in performance would not be significant. However, since the running time is $O(M^3)$, dominated in practice by the maximum-weight bipartite matching, if we assume that the number of vertices in both pieces is roughly equal, this would reduce the overall running time by a factor of 2. This strategy, of course, can also in principle be applied with other kinds of partitioning algorithms (such as community detection algorithms[38]).

Finally, as discussed in the previous section, this algorithm could yield new insights into the evolutionary history of the two species being analyzed.

Acknowledgments

The authors would like to thank Kelley Bailey, Mathieu Blanchette, Michael Schnall-Levin, Kanghao Lu, and Rohit Singh for assistance and useful discussion. A special thanks to Samujjal Purkayastha for creating Figure 1 and providing computational support. Leonid Chindelevitch was supported by the Postgraduate Award from the National Sciences and Engineering Research Council of Canada. Chung-Shou Liao was supported by the National Science Council under the Grants NSC95-2221-E-001-016-MY3.

References

1. T. Ito et al. A comprehensive two-hybrid analysis to explore the yeast protein interactome. In *Proceedings of the National Academy of Sciences USA*, **98(8)** (2001) pp. 4569-4574.
2. Y. Ho et al. Systematic identification of protein complexes in Saccharomyces cerevisae by mass spectrometry. *Nature* **415** (2002) pp. 180-183.

3. A.C. Gavin et al. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature* **415 (6868)** (2002) pp. 141-147.
4. R. Sharan et al. Conserved patterns of protein interaction in multiple species. In *Proceedings of the National Academy of Sciences USA*, **102** (2005) pp. 1974-1979.
5. R. Singh et al. Global alignment of multiple protein interaction networks with application to functional orthology detection. In *Proceedings of the National Academy of Sciences USA*, **105** (2008) pp. 12763–12768.
6. M. Kalaev et al. Fast and accurate alignment of multiple protein networks. In Research in Computational Molecular Biology. Vol. 4955, Lecture Notes in Computer Science, Springer, Berlin/Heidelberg (2008) pp. 246-256.
7. J. Flannick. et al. Automatic parameter learning for multiple network alignment. In *Research in Computational Molecular Biology*. Vol. 4955, Lecture Notes in Computer Science, Springer, Berlin/Heidelberg (2008) pp. 214-231.
8. C.-S. Liao, K. Lu, M. Baym, R. Singh and B. Berger. IsoRankN: spectral methods for global alignment of multiple protein networks. *Proceedings of the International Conference on Intelligent Systems in Molecular Biology, ISMB '09* (2009), pp. 253-258. doi:10.1093/bioinformatics/btp203
9. M. Zaslavskiy, F. Bach, J.-P. Vert. Global alignment of protein-protein interaction networks by graph matching methods. *Proceedings of the International Conference on Intelligent Systems in Molecular Biology, ISMB '09* (2009), pp. 259-267. doi:10.1093/bioinformatics/btp196
10. J. Berg and M. Lässig. Cross-species analysis of biological networks by Bayesian alignment. In *Proceedings of the National Academy of Sciences USA*, **103** (2006) pp. 10967-10972.
11. J. Dutkowski and J. Tiuryn. Identification of functional modules from conserved ancestral protein-protein interactions. *Bioinformatics*, **23** (2007) pp. 149-158.
12. B.P. Kelley et al. Conserved pathways within bacteria and yeast as revealed by global protein network alignment. In *Proceedings of the National Academy of Sciences USA*, **100** (2003) pp. 11394-11399.
13. B.P. Kelley et al. Pathblast: a tool for alignment of protein interaction networks. *Nucleic Acids Research*, **32** (2004) pp. 83-88.
14. M. Koyutürk et al. Pairwise Alignment of Protein Interaction Networks. *Journal of Computational Biology*, **13(2)** (2006) pp. 182–199.
15. B.S. Srinivasan et al. Integrated protein interaction networks for 11 microbes. In *Research in Computational Molecular Biology*. Vol. 3909, Lecture Notes in Computer Science, Springer, Berlin/Heidelberg (2006) pp. 1-14.
16. S. Sahni, T. Gonzales. P-complete approximation problems. *Journal of the Association for Computing Machinery*, **23** (1976) pp. 555-565.
17. B. Chandra, H. Karloff, and C. Tovey. New results on the old k-opt algorithm for the TSP. In *Proceedings of the 5th ACM-SIAM Symposium on Discrete Algorithm, SODA'94*, (1994) pp. 150–159.
18. G.A. Croes. A method for solving traveling salesman problems. *Operations Research*, **6** (1958) pp. 791–812.
19. T.J.P. Hubbard et al. Ensembl 2007. *Nucleoc Acids Research*, **35** (2007) pp. 610–617.
20. D.S. Johnson and L.A. McGeoch. The traveling salsman problem: a case study in local optimization. In *Local Search in Combinatorial Optimization*, John Wiley & Sons, London, (1997) pp. 215–310.
21. F. Viger, M. Latapy. Efficient and simple generation of random simple connected graphs with prescribed degree sequence. In *Proceedings of the International Computing and Combinatorics Conference, COCOON '05*, (2005) pp. 440–449.
22. R. Taylor. Constrained switchings in graphs. *Combinatorial Mathematics*, **8** (1980) pp. 314-336.
23. M.Y. Kao, T.W. Lam, W.K. Sung, and H.F. Ting. A decomposition theorem for maximum weight bipartite matchings. *SIAM Journal of Computing*, **31** (2001) pp. 18–26.
24. H.W. Kuhn. The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly*, **2** (1955) pp. 83–97.
25. E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. The Traveling Salesman Problem, John Wiley & Sons, Chichester (1985).
26. G. Lueker, manuscript, Princeton University, (1976).
27. C.H. Papadimitriou and K. Steiglitz. On the complexity of local search for the traveling salesman problem. *SIAM Journal of Computing*, **6** (1977) pp. 76–83.
28. Jordan, M. (ed.) Learning in Graphical Models. The MIT Press, Cambridge, (2001).
29. Bandyopadhyay, S. et al. Systematic identification of functional orthologs based on protein network comparison. *Genome Research*, **16** (2006), pp. 428-435.
30. Zaslavskiy, M. et al. A path following algorithm for graph matching. In *Image and Signal Processing, Proceedings of the 3rd International Conference, ICISP*, (2008), pp. 329-337.
31. Python Software Foundation. *http://www.python.org/psf/*
32. A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference, SciPy '08*, (2008) pp. 11–15.

33. Z. Galil, Efficient Algorithms for Finding Maximum Matchings in Graphs. *ACM Computing Surveys*, **18:1** (1986), pp. 23–38.
34. IsoRank and IsoRankN. *http://isorank.csail.mit.edu* Last checked on September 21, 2009.
35. HomoloGene. *http://www.ncbi.nlm.nih.gov/homologene* Last checked on September 21 13, 2009.
36. L. Chindelevitch. Thoughts on the IsoRank algorithm. Unpublished manuscript.
37. A. A. Tsay, W. S. Lovejoy, D. R. Karger. Random Sampling in Cut, Flow, and Network Design Problems. *Mathematics of Operations Research*, **24:2** (1999) pp. 383-413.
38. M. E. Newman. Modularity and community structure in networks. In *Proceedings of the National Academy of Sciences USA*, **103:23** (2006) pp. 8577-8582.

8. Appendix

This appendix presents a running-time bound for our algorithm, with proof.

8.1. *Running-time analysis*

In what follows, M^* always denotes the mapping at the current iteration.

First, note that only topology similarities $t(e)$ for each edge $e \in M^*$ incident to a node in $N(x)$, $N(y)$, $N(u)$, and $N(v)$ are changed when we swap the edges (x, y) and (u, v) in M^*. In addition, as we consider the weight difference $w(M^* \setminus \{e, e'\} \cup \{e_1, e_2\}) - w(M^*)$, removing the edge $e = (x, y) \in M^*$ causes the weight loss $w(e)$, but it also causes neighbors in $N(x)$ and $N(y)$ to lose $\alpha t(e)$. Removing the edge $e' = (u, v)$ produces an analogous effect. On the other hand, there is a weight gain $w(e_1)$ as well as $\alpha t(e_1)$ from inserting the edge $e_1 = (x, v)$. Inserting the edge $e_1 = (x, v)$ produces an analogous effect.

Therefore, the weight of the matching M^* increases by the quantity $swap(e, e')$, defined in (2). The inequality in (2) thus ensures that the objective function increases after the swap.

Theorem 8.1.

Given a a weighted bipartite graph $G = (X \cup Y, E)$ with a maximum weighted matching M^* and parameters α and c, the running time of PISwap *is pseudo-polynomial time bounded in the worst case.*

Proof.

It is readily seen that the cardinality of a maximum-weight matching M^* is $|M^*| \leq \min\{|X|, |Y|\}$. Note that the preprocessing of PISwap to obtain a maximum weighted matching M^* by the *Hungarian algorithm* takes $O(|M^*|^3)$ time.

Let Δ denote the largest degree of a vertex in G_X and G_Y, i.e. the largest number of neighbors a vertex in $X \cup Y$ can have. Let B denote the largest similarity value for two sequences, i.e. $B = \max_{x \in X, y \in Y}\{s(x, y)\}$.

In Step 1 we compute the topology similarities $t(e)$ and the weights $w(e)$ for each edge $e = (x, y) \in M^*$. Since we consider all possible pairwise combinations between neighbors of x and neighbors of y, this requires $O(|M^*| \times \Delta^2)$ time.

In Step 2 we find the candidate set S. We first compute the subsets $\text{prefer}_Y(x)$ and $\text{prefer}_X(y)$ for each vertex in $X \cup Y$. The running time is bounded by $O(|X| \times |Y|)$ since it requires $O(|Y|)$ (respectively $O(|X|)$) time to find the c highest-weighted neighbors in Y (respectively X) for each vertex $x \in X$ (respectively $y \in Y$), for any constant c.

We then find all the edges $e' \in M^*$ satisfying the condition (2) for every edge $e \in M^*$. For every edge $e = (x, y) \in M^*$, there are at most c^2 edges in M^* with one endpoint in $\text{prefer}_X(y)$ and the other endpoint in $\text{prefer}_Y(x)$ that e can be swapped with. The weight difference $swap(e, e')$ can be computed in constant time from the topology similarities and sequence similarities for every edge. Hence Step 2 takes $O(c^2|M^*|)$ time. Without loss of generality, we assume $c \leq \sqrt{|M^*|}$; otherwise we can check all pairs of edges in M^* to see if they are able to be swapped, at a cost of $O(|M^*|^2)$.

Step 3 is an iteration, and we first consider the time complexity of one iteration. The maximum value of $swap(e, e')$ can be found in constant time by using a priority queue. The swap operation also takes constant time. For the two new inserted edges of M^*, we verify if they satisfy the property (2) in $O(c^2)$ time as above.

Pacific Symposium on Biocomputing 15:123-132(2010)

The last step to update the values of $t(e)$, $w(e)$, and $swap(e, e')$. This takes $O(\Delta^2)$ time as above, since only the edges $e \in M^*$ with one endpoint in $N(x) \cup N(u)$ and the other, in $N(y) \cup N(v)$, are affected. Therefore, each iteration requires $O(\max\{c^2, \Delta^2\})$ time.

Finally, consider the number of iterations of the while loop. The total sequence score is an integer and varies between 0 and $|M^*| \times B$, and similarly, the total topology score is an integer and varies between 0 and $|M^*|^2$; the consecutive values of $w(M^*)$ form a strictly increasing sequence whose length is bounded by $(|M^*| \times B + 1) \times (|M^*|^2 + 1) \in O(|M^*|^3 \times B)$. Since step 3 is the dominating one, PISwap runs in $O(\max\{c^2, \Delta^2\} \times B \times |M^*|^3)$ time. □