Multiple Alignment of Biological Networks: A Flexible Approach

Yves-Pol Deniélou[1], Frédéric Boyer[2], Alain Viari[1], and Marie-France Sagot[1]

[1] INRIA Grenoble-Rhône-Alpes, projet BAMBOO, 655 avenue de l'Europe, 38334 Montbonnot Cedex, France
{yves-pol.denielou,marie-france.sagot,alain.viari}@inrialpes.fr
[2] CEA, iRTSV, Laboratoire Biologie, Informatique et Mathématiques, F-38054 Grenoble, France
frederic.boyer@cea.fr

Abstract. Recent experimental progress is once again producing a huge quantity of data in various areas of biology, in particular on protein interactions. In order to extract meaningful information from this data, researchers typically use a graph representation to which they apply network alignment tools. Because of the combinatorial difficulty of the network alignment problem, most of the algorithms developed so far are heuristics, and the exact ones are of no use in practice on large numbers of networks. In this paper, we propose a unified scheme on the question of network alignment and we present a new algorithm, C3Part-M, based on the work by Boyer *et al.* [2], that is much more efficient than the original one in the case of multiple networks. We compare it as concerns protein-protein interaction networks to a recently proposed alignment tool, NetworkBLAST-M [10], and show that we recover similar results, while using a different but exact approach.

Keywords: multiple graph alignment, biological network comparison, protein-protein interactions.

1 Introduction

The recent advances in high-throughput experiments have fueled the research for an automated characterisation of biological networks such as protein-protein interactions (PPI) networks [21], metabolic pathways [13], or gene regulation networks [1]. A central question to the analysis of this kind of data is to align the networks in order to extract conserved subnetworks across multiple species or data sources. This information is of primary importance to study both the evolution and function of the proteins.

The network alignment question shares some similarities with the classical case of sequence alignment. Indeed, network alignment approaches can be local [2, 7, 10, 11, 12, 16] or global [6, 17, 18], pairwise [11, 12, 17, 20] or multiple [6, 7, 10, 16, 18]. In addition, just like with sequence symbols, one must also address the question of defining a similarity between the vertices of different networks.

G. Kucherov and E. Ukkonen (Eds.): CPM 2009, LNCS 5577, pp. 263–273, 2009.

However, a specific question that arises only in network alignments is to define which topological similarity to enforce on the aligned subnetworks.

A key concept, used in most of the studies [2,11,12,15,16], is that of a merged representation of the networks, which is called network alignment graph [16] or correspondence multigraph [2] in the literature. Given n networks $G_1, G_2, \ldots G_n$, with $G_i = (V_i, E_i)$, and a correspondence relation S between the vertices of $\bigcup_i V_i$, the idea is to select n-tuples (alternatively called n-spines [10]) of vertices, one from each V_i, and to connect them with edges taken from the original E_i.

As we shall see later, an explicit construction of this representation is not always needed (nor desirable) but it turns out to be very useful to understand the differences between the algorithms or the biological questions to be solved. This can be summarised in three main questions:

- how to select the n-tuples (i.e. how to aggregate the vertices from the different sets V_i using S);
- which edges (from the different sets E_i) to keep;
- which topological condition(s) should be satisfied by the solution subgraphs.

Of course, the algorithmic difficulty of a network alignment greatly depends upon the answers to these questions and in most cases one has to resort to heuristics. In addition, several authors introduced scoring models at different levels of such heuristics. This will not be discussed here and we instead treat the question from

Table 1. Different methods for PPI network alignment and the corresponding assumptions. The *n-tuples* column gives the scheme used to gather homologuous proteins according to similarity edges. CC stands for connected components. The *edges* column details which edges from the PPI networks are taken into account. "conserved" means that only edges that appear in every single network are conserved. The *topology* column tells which kind of subgraphs the method is recovering, for instance the NetworkBLAST method recovers clusters and paths of the network alignment graph, whereas NetworkBLAST-M maximises a sum of scores, one for each network, ensuring the subgraphs are dense on each network.

Method	n-tuples construction	edges construction	topology	reference
PathBLAST	pairs (n=2)	conserved	paths	[11]
NetworkBLAST	CC ($n \leq 3$)	conserved	clusters / paths	[16]
NetworkBLAST-M	paths	all edges	dense clusters on each network	[10]
GRAEMLIN 1.0	non-overlap-ping CC	conserved	user-defined	[7]
CAPPI	non-overlap-ping CC	all edges > threshold	CC	[5]
HopeMap	pairs	conserved	CC	[20]
MaWISH	pairs	conserved	max-weight subgraph	[12]
PHUNKEE	pairs	all edges	max shared-edges ratio	[3]
C3PART	cliques or stars	all edges	common CC	[2]

a purely combinatorial point of view. We also focus on the more general question of a multiple alignment.

Table 1 summarises some important cases found in the literature.

For example, in the study by Sharan *et al.* [16] , the n-tuples are connected components for the correspondence relation S, the edges are those from E_i conserved in all the networks, and the enforced topologies are "dense" clusters, whereas in Boyer *et al.* [2] the n-tuples are cliques of S, all the edges in all the E_i's are kept (therefore giving rise to a multigraph instead of a graph) and the topological condition is simply the connectivity.

It is important to note that when the correspondence relation S is not one-to-one, the size of the network alignment graph may grow exponentially with both the number of vertices (n-tuples) and of edges, making it difficult to handle more than three networks. A general question thus appears: how to avoid the explicit construction of the network alignment graph ?

Several methods have been proposed to address this difficulty, among them the Graemlin algorithm [7] which avoids this construction by using a progressive alignment approach, and NetworkBLAST-M [10] which builds a set of n-tuple seeds with maximum score and then extends them greedily.

In this paper, we propose a non-heuristic approach to avoid the explicit construction of the network alignment graph and yet recover a pertinent and well-defined alignment. This work is based on the general framework proposed by Boyer *et al.* [2], which uses a correspondence multigraph formalism to extract connected components conserved in multiple networks. Our algorithm is an improvement of Boyer's original algorithm, C3Part, in order to avoid the explicit construction of the network alignment multigraph, and thus to be able to deal with a greater number of networks or a more degenerate correspondence relation across the networks. We illustrate this approach on the example of PPI networks, but all algorithmic concepts presented here can be applied on other kind of data as well.

The paper is organised as follows: Sections 2.1 to 2.3 define the layered data graph and the network alignment multigraph. Section 2.4 gives a brief overview of the C3Part algorithm and its limits. Section 2.5 presents our improvement of this algorithm, that builds on the fly the parts of the network alignment multigraph that are really needed for the alignment. And finally, Section 3 provides some results in the case of PPI networks, allowing us to compare the efficiency of our method with other similar approaches.

2 Methods

2.1 Layered Data Graph

The layered data graph (also called layered alignment graph in [10]) provides the simplest representation of the data at hand.

Definition 1. (adapted from [10]) *Given a set of n networks $G_i = (V_i, E_i)$, $i \in [|1, n|]$ (hereafter called* primary networks*) and a correspondence relation*

S between the elements of distinct sets V_i, the layered data graph *is the graph* $D = (V, E)$ *with*

- $V = \bigcup_i V_i$
- $E = (\bigcup_i E_i) \cup \{(u, v) \in V_i \times V_{j \neq i} / u \; S \; v\}$

Observe that there are two kinds of edges in E: one corresponds to the original sets E_i (hereafter called *intra-layer edges*) and the other one connects vertices from different layers (hereafter called *inter-layer edges*) (see Figure 1).

2.2 Correspondence n-Way, and n-Tuples

As mentioned before, for $n > 2$ networks, one has to define formerly how the layered data graph vertices are aggregated to form n-tuples.

Definition 2. *An* n-way correspondence *between elements of* $V_1, V_2, \ldots V_n$ *is defined as a restriction* $\mathcal{R}$ *of the cartesian product, denoted by* $\mathcal{R}(V_1 \times V_2 \times \ldots \times V_n)$.

There are three main interesting practical cases of such an aggregation.

1. The clique aggregator :
   $(v_1, \ldots v_n) \in \mathcal{R}(V_1 \times V_2 \times \ldots \times V_n) \Leftrightarrow \forall i, j \in [|1, n|], \; v_i \; S \; v_j$
   ie we require that all elements of the tuple are pairwise related.
2. The centered-star aggregator :
   $(v_1, \ldots v_n) \in \mathcal{R}(V_1 \times V_2 \times \ldots \times V_n) \Leftrightarrow \exists i \in [|1, n|] / \forall j \in [|1, n|], \; j \neq i, \; v_i \; S \; v_j$
   ie we require a "star" topology of the relations between elements of the tuple.
3. The connected component aggregator :
   $(v_1, \ldots v_n) \in \mathcal{R}(V_1 \times V_2 \times \ldots \times V_n) \Leftrightarrow (v_1, \ldots v_n)$ is a connected component of S.
   A particular and important case of connected component aggregator is the path aggregator [16] which requires the vertices to be connected by a path. Among the path aggregators, the tree-guided path aggregator [10] requires the path to be compatible with a given phylogenetic tree.

2.3 Network Alignment MultiGraph

A network alignment multigraph is a graph data structure that summarises both the n-way correspondence and the connectivity in the primary networks.

Definition 3. *The* network alignment multigraph *is the multigraph* $M = (V, E'_1, \ldots, E'_n)$ *such that:*

- $V = \mathcal{R}(V_1 \times V_2 \times \ldots \times V_n)$
- $\forall u = (u_1, u_2, \ldots, u_n) \in V$ *and* $v = (v_1, v_2, \ldots, v_n) \in V$,
  $(u, v) \in E'_i \Leftrightarrow (u_i, v_i) \in E_i \vee (v_i = u_i)$

In other words, the vertices of the multigraph are n-tuples and there is an edge between two vertices if the i^{th} elements (vertices) of the tuples are connected in the primary network G_i, for all $i \in [|1, n|]$. In the following, we refer to such an edge as an *edge of colour* i.

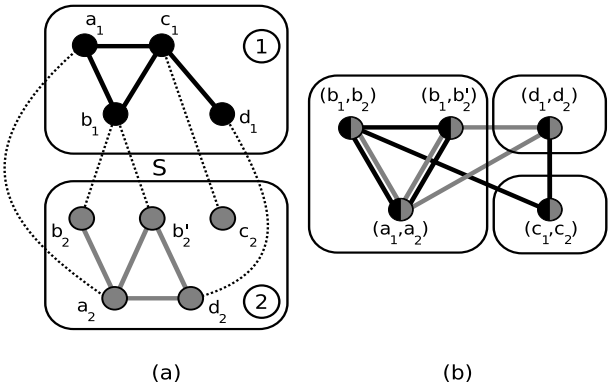

Fig. 1. Example of a layered data graph (a) and the corresponding network alignment multigraph (b) with 3 connectons. The correspondence relation S is represented by dotted lines. The restriction R is defined by the couples of vertices linked by S edges. Observe that the whole multigraph is not a connecton since (c_1, c_2) is not reachable by both colours. Also observe that connectons do not correspond to the intersection of the connected components.

2.4 Defining Connectons in the Network Alignment Multigraph

With this definition of the network alignment multigraph at hand, there are several possible definitions of the property we want to look for (see Table 1), and therefore of the conserved subgraphs we want to recover. Here we choose the definition formalised by Boyer *et al.* [2] which presents several advantages that will be discussed later.

Definition 4. *A* connecton *is a maximal set of vertices in the network alignment multigraph which are connected components for each* E'_i, $i \in [|1, n|]$.

An example of connecton is given in Figure 1. An important property is that the sets of connectons form a partition of the vertices of the multigraph. This property, which is usually not satisfied with other definitions (e.g. dense clusters [10]), allows the use of exact algorithms instead of heuristics to enumerate them. Moreover, the connecton condition is weaker than would be a condition on dense clusters, and, provided that connectons are not too numerous, they can be further post-processed to satisfy a stronger constraint. Also observe that to speed up the algorithm, we can restrict ourselves in the definition of connectons to connected components above a fixed size.

The algorithm used by Boyer *et al.* [2] to find connectons in the network alignment multigraph computes iteratively the partition, starting with a single class containing all the vertices and refining the partition at each step. The refinement procedure used in C3Part works as follows:

1. start with one single class containing all the vertices in the network alignment multigraph;

2. compute $\bigcap_i(CC_i)$, the intersection of all connected components on all colours, this gives rise to a new partition where each class is a potential connecton;
3. iterate -2- on each class until the partition does not change.

The worst-case time complexity is $\mathcal{O}((N + M) \times N)$ where N is the number of vertices and M the number of edges in the multigraph. This comes from the fact that each iteration requires $\mathcal{O}(N+M)$ operations to compute the connected components and that, in the worst-case one will have to perform $\mathcal{O}(N)$ iterations (this corresponds to the case where there are $\mathcal{O}(N)$ classes at the end and each class has been extracted at each iteration).

This complexity can actually be improved. In 2003, Gai *et al.* [8] proposed a more sophisticated algorithm combining the dynamic maintenance of a spanning forest together with an Hopcroft-like partitioning approach. This algorithm achieves an $\mathcal{O}((N + M \times logN) \times logN)$ complexity. Furthermore, in the case of interval graphs, the complexity reduces to $\mathcal{O}((N + M) \times logN)$ [9]. This latter particular case is important to handle questions related to chromosomal syntenies [14] where the primary networks are interval graphs.

In practice, neither C3Part nor those algorithms are usable on large numbers of networks, since their prerequisite is the construction of the network alignment multigraph, whose size is exponential with the number of networks when the correspondence relation S is not a one-to-one correspondence.

The objective of this work is to apply the same approach but to avoid the initial construction of the multigraph. The general idea is to build the multigraph on the fly, starting with a connected component on the first primary graph, expanding it on the second one, then splitting it on those two colours, and expanding recursively the results on the third graph, etc.

2.5 On the Fly Construction of the Network Alignment Multigraph

Although the original C3Part algorithm adopted a Breadth-First-Search presentation, is is easy to transform it into a Depth-First-Search by observing that the split of a class can actually be conducted on each colour in turn. This is made possible by the fact that if two vertices are disconnected by two or more colours, only one is actually needed to split the class. With a DFS approach, all classes are therefore refined independently. The next observation is that, since colours are now considered in turn, when we split a class on a colour i, we may not need the information about the $(n - i)$ remaining colours that will be used later on. This therefore makes it possible to add the new colours only when necessary.

The new algorithm, C3Part-M, will use two different operations.

- $SPLIT_{1-i}$ that splits a class on colours 1 to i;
- $EXPAND_{i+1}$ that adds the $(i+1)^{th}$ colour to the current network alignment multigraph.

A *stable class* is a class C such that $SPLIT_{1-n}(C) = C$ and is therefore a connecton.

The $SPLIT_{1-i}$ operation computes the connected components on each colour in turn. If, for a colour, the class is split, then it returns the split parts.

When a class C is such that $SPLIT_{1-i}(C) = C$ then it is stable for colours 1 to i and needs expansion to the $(i+1)^{th}$ colour.

The pseudocode of the algorithm is given hereafter.

Algorithm: C3PART-M.
Input: *Set_of_vertices class /* class to refine: initialised with all vertices from the first network */*
Colour_Index i / current colour index: initialised to 1 */*
Variables: *Partition_of_vertices split*

```
(1)   begin
(2)     split ← SPLIT_{1→i}(class);
(3)     if (|split| ≠ 1) then
(4)       for s ∈ split do
(5)         C3PART-M(s, i)
(6)       end for
(7)     else if (i ≠ maxcolour) then
(8)       newclass ← EXPAND(class, i + 1);
(9)       C3PART-M(newclass, i + 1);
(10)    else
(11)      /* class is stable */
(12)      PRINT(class)
(13)    end if
(14)  end
```

The expansion is done by the $EXPAND$ operation that works in two steps. Starting from the current class C (i.e. a reduced multigraph defined on colours 1 to i), we:

1. expand each vertex $v = (v_1, v_2, \ldots v_i)$ of size i to new vertices of size $i+1$ ($EXPAND\_VERTEX$);
2. add edges between these new vertices ($EXPAND\_EDGES$).

$EXPAND\_VERTEX$ works as follows. For each vertex $v = (v_1, v_2, \ldots v_i)$ in the current class C, we first collect all vertices v_{i+1} of the $(i+1)^{th}$ primary graph such that $(v_1, v_2, \ldots, v_i, v_{i+1}) \in \mathcal{R}(V_1 \times V_2 \times \ldots \times V_i \times V_{i+1})$. These vertices are called the *terminals* of v. Then for each of these terminals, a new vertex $v' = (v_1, v_2, \ldots v_{i+1})$ is created. This new vertex v' is hereafter called a *son* of v and v is called its *father*.

Observe that how all terminals of a given vertex v are collected depends upon the chosen aggregator. For the clique aggregator (resp. star aggregator), this is a simple task since, by construction $v = (v_1, v_2, \ldots v_i)$ is already a clique (resp. star) of S, then one has just to collect the terminals v_{i+1} that are S-connected to each v_i (resp. to the center star). For the connected component aggregator, the task is more demanding since vertices v_{i+1} may form a connected

component with $v = (v_1, v_2, \ldots v_i)$ only once all colours have been considered (i.e. colours greater than $i + 1$). The complexity therefore strongly depends upon the degeneracy of S.

Once all the new vertices have been added, one should add the new edges as well ($EXPAND\_EDGES$). There are actually three kinds of such edges, connecting two new vertices $(u_1, u_2, \ldots u_i, u_{i+1})$ and $(v_1, v_2, \ldots v_i, v_{i+1})$. The first kind is simply the edges already existing in the class C, i.e. connecting fathers, that should be restored with their previous colours. The second kind is edges connecting sons of the same father. Those edges have all colours from 1 to i since the father is the same ($(u_1, u_2, \ldots u_i) = (v_1, v_2, \ldots v_i)$). Finally, one has to add new edges of colour $i + 1$ corresponding to the terminals u_{i+1} and v_{i+1}, i.e. connecting all new vertices such that $(u_{i+1}, v_{i+1}) \in E_i$ (or $u_{i+1} = v_{i+1}$).

The worst case complexity of this new algorithm (C3Part-M) is the same as the one of C3Part. It corresponds to the case where there is no split on the first $n-1$ colours, thus giving rise to the full alignment multigraph that is eventually splitted on the last n^{th} colour. For all practical cases we have tested so far, this never happens and the new version turns out to be several order of magnitude faster than the previous one (see [4] for an example on syntenies). Moreover, one can observe that since the result does not depend upon the order in which the colours are chosen, it is possible to avoid this worst case either by choosing an order more likely to be favourable at the beginning or, better, by reordering the colours dynamically during the recursive split. Several heuristic optimisations have been tested but will not be further described in this extended abstract.

3 Results and Discussion

In order to evaluate the new algorithm, we selected the benchmark set of 10 microbial PPI networks used in previous similar studies [7, 10]. Those networks are not experimental but were actually produced by an inference algorithm (called SRINI) described in [19]. Briefly, SRINI generates a probabilistic interaction network by integrating several sources of information such as co-expression, co-evolution or chromosomal co-location. Unlike experimental PPIs, the output of SRINI is a complete graph where each pair of vertices is labelled by an interaction probability. In order to be used in any algorithm, these networks should therefore be thresholded.

As for the correspondence relation S, we used the same data as in [7, 10], *i.e.* a sequence similarity relation determined by BlastP. We selected a Blast threshold of 10^{-10} and limited the number of hits per protein to 5. The restriction R was defined by cliques of S (clique aggregator *i.e.* all proteins are similar one to the other).

The sizes of the obtained layered data graphs are given in Table 2 for different numbers of selected species (3, 5, 7 and 10; the selected species are the same as in [10]).

The first observation we can make is that C3Part-M was able to cope with these large networks whereas the explicit computation of the alignment multigraph is intractable by C3Part (and NetworkBLAST) for more than 3 species.

Table 2. Comparison of C3Part-M and NetworkBlast-M on 10 microbial species. NetworkBlast-M running times are given both for the relaxed mode (1) and for the tree-guided-path mode (2) (comparisons of other results are given for the relaxed mode). *#Spine* corresponds to the number of different n-tuples found by the various algorithms. *#Prot* corresponds to the number of different proteins involved in these n-tuples.

n	PPI Thresh.	# Prot	#Edges Simil.	#Edges PPI	Time (s) NBM (1)	Time (s) NBM (2)	Time (s) C3P-M	#Spine NBM	#Spine C3P-M	#Spine Common	#Prot NBM	#Prot C3P-M	#Prot Common
3	0.7	3751	1526	7450	1	1	4	91	194	76(84%)	235	339	207(88%)
	0.5	5322	2739	18511	2	2	6	169	457	128(76%)	413	628	354(86%)
	0.3	7826	4606	66484	9	5	12	291	972	195(67%)	679	1128	557(82%)
5	0.7	5910	4444	13061	12	2	6	82	261	58(71%)	337	370	266(79%)
	0.5	8169	7422	38645	34	3	9	168	566	73(43%)	599	656	390(65%)
	0.3	11404	11805	134437	175	12	19	201	1432	104(52%)	794	1231	562(71%)
7	0.7	8416	8721	18707	166	2	7	31	190	19(61%)	206	252	156(76%)
	0.5	12354	16897	60517	478	5	12	119	531	54(45%)	602	640	394(65%)
	0.3	16579	26452	211368	2832	17	38	193	2500	77(40%)	1002	1284	601(60%)
10	0.7	15411	21995	47340	9038	3	60	24	853	18(75%)	240	292	210(88%)
	0.5	23370	50758	173881	39644	10	74	111	2062	45(41%)	798	729	472(59%)
	0.3	30534	74126	616307	N/A	33	1143	N/A	13250	N/A	N/A	1613	N/A

We then compared our results with those obtained using NetworkBlast-M [10], that is the multiple version of the NetworkBlast algorithm [16]. As mentioned before, NetworkBlast-M is a heuristic algorithm that relies on different assumptions concerning the construction of the vertices and edges of the alignement multigraph and on the required topology of the subnetworks (see Table 2). We ran NetworkBlast-M both in "relaxed" mode (where n-spines are paths) and in the tree-guided-path mode that was shown to be much quicker [10]. The running time of C3Part-M compares well with the tree-guided-path mode and could in most case retrieve the connectons within a few minutes.

A more interesting comparison with NetworkBlast-M is in terms of the results found. Since the assumptions are not the same, we cannot compare directly the complexes found. We therefore decided to compare the results in terms of the different constructed n-tuples and different proteins involved. Interestingly, despite their different asumptions, it turns out that the two algorithms recover essentially the same objects, both in terms of spines and proteins. For 3 species, this was somehow expected since the clique and the path conditions are quite similar, and up to 84% of the spines and 88% of the proteins reported by NetworkBlast-M were also found by C3Part-M. Of course, this overlap decreases with the number of species but remains remarkably high even for 10 species. An explanation for this comes from the observation done by Kalaev *et al.* [10] that most of the recovered spines actually are cliques.

4 Conclusion

We addressed the problem of multiple network alignment with an exact and generic approach based on the work by Boyer *et al.* [2]. By avoiding the explicit construction of the network alignment multigraph, we were able to deal with a large number of networks.

A comparison of our algorithm to NetworkBlast-M [10] led to very similar results, with reasonable execution times. Furthermore, our definition of connectons as subgraphs corresponding to connected components on each network allows us to cleanly separate the heuristic choices of biologically-relevant scoring functions from the alignment procedure itself. Our approach can consequently be used as a pre-filter to other more specialised tools.

Many important challenges remain. For instance the introduction of weaker aggregators would help to recover conserved subgraphs in the case of missing proteins. One idea would be to introduce a species quorum *i.e.* to look for spines not necessarily containing all the species.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive comments and suggestions.

References

1. Babu, M.M., Luscombe, N.M., Aravind, L., Gerstein, M., Teichmann, S.A.: Structure and evolution of transcriptional regulatory networks. Curr. Opin. Struct. Biol. 14(3), 283–291 (2004)
2. Boyer, F., Morgat, A., Labarre, L., Pothier, J., Viari, A.: Syntons, metabolons and interactons: an exact graph-theoretical approach for exploring neighbourhood between genomic and functional data. Bioinformatics 21(23), 4209–4215 (2005)
3. Cootes, A.P., Muggleton, S.H., Sternberg, M.J.: The identification of similarities between biological networks: Application to the metabolome and interactome. Journal of Molecular Biology 369(4), 1126–1139 (2007)
4. Denielou, Y.-P., Boyer, F., Sagot, M.-F., Viari, A.: Recovering isofunctional genes: a synteny-based approach. In: JOBIM, pp. 11–16 (2008)
5. Dutkowsky, J., Tiuryn, J.: Identification of functional modules from conserved ancestral protein protein interactions. Bioinformatics 23(13) (2007)
6. Flannick, J.A., Novak, A.F., Do, C.B., Srinivasan, B.S., Batzoglou, S.: Automatic parameter learning for multiple network alignment. In: Vingron, M., Wong, L. (eds.) RECOMB 2008. LNCS (LNBI), vol. 4955, pp. 214–231. Springer, Heidelberg (2008)
7. Flannick, J., Novak, A., Srinivasan, B.S., McAdams, H.H., Batzoglou, S.: Græmlin: general and robust alignment of multiple large interaction networks. Genome Res. 16(9), 1169–1181 (2006)
8. Gai, A.T., Habib, M., Paul, C., Raffinot, M.: Identifying Common Connected Components of Graphs. Technical report, LIRMM (2003)
9. Habib, M., Paul, C., Raffinot, M.: Maximal Common Connected Sets of Interval Graphs. In: Sahinalp, S.C., Muthukrishnan, S.M., Dogrusoz, U. (eds.) CPM 2004. LNCS, vol. 3109, pp. 347–358. Springer, Heidelberg (2004)
10. Kalaev, M., Bafna, V., Sharan, R.: Fast and accurate alignment of multiple protein networks. In: Vingron, M., Wong, L. (eds.) RECOMB 2008. LNCS (LNBI), vol. 4955, pp. 246–256. Springer, Heidelberg (2008)

11. Kelley, B.P., Sharan, R., Karp, R.M., Sittler, T., Root, D.E., Stockwell, B.R., Ideker, T.: Conserved pathways within bacteria and yeast as revealed by global protein network alignment. Proc. Natl. Acad. Sci. USA 100(20), 11394–11399 (2003)
12. Koyutürk, M., Kim, Y., Topkara, U., Subramaniam, S., Szpankowski, W., Grama, A.: Pairwise alignment of protein interaction networks. J. Comput. Biol. 13(2), 182–199 (2006)
13. Papin, J.A., Price, N.D., Wiback, S.J., Fell, D.A., Palsson, B.O.: Metabolic pathways in the post-genome era. Trends Biochem. Sci. 28(5), 250–258 (2003)
14. Pasek, S., Bergeron, A., Risler, J.L., Louis, A., Ollivier, E., Raffinot, M.: Identification of genomic features using microsyntenies of domains: Domain teams. Genome Res (2005)
15. Sharan, R., Ideker, T., Kelley, B., Shamir, R., Karp, R.M.: Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. J. Comput. Biol. 12(6), 835–846 (2005)
16. Sharan, R., Suthram, S., Kelley, R.M., Kuhn, T., McCuine, S., Uetz, P., Sittler, T., Karp, R.M., Ideker, T.: From the cover: Conserved patterns of protein interaction in multiple species. Proc. Natl. Acad. Sci. USA 102(6), 1974–1979 (2005)
17. Singh, R., Xu, J., Berger, B.: Pairwise global alignment of protein interaction networks by matching neighborhood topology. In: Speed, T., Huang, H. (eds.) RECOMB 2007. LNCS (LNBI), vol. 4453, pp. 16–31. Springer, Heidelberg (2007)
18. Singh, R., Xu, J., Berger, B.: Global alignment of multiple protein interaction networks with application to functional orthology detection. Proceedings of the National Academy of Sciences 105(35), 12763–12768 (2008)
19. Srinivasan, B.S., Novak, A.F., Flannick, J.A., Batzoglou, S., McAdams, H.H.: Integrated protein interaction networks for 11 microbes. In: Apostolico, A., Guerra, C., Istrail, S., Pevzner, P.A., Waterman, M. (eds.) RECOMB 2006. LNCS (LNBI), vol. 3909, pp. 1–14. Springer, Heidelberg (2006)
20. Tian, W., Samatova, N.F.: Pairwise alignment of interaction networks by fast identification of maximal conserved patterns. In: PSB 2009, pp. 99–110 (2009)
21. Tucker, C.L., Gera, J.F., Uetz, P.: Towards an understanding of complex protein networks. Trends in Cell Biology 11(3), 102–106 (2001)