Bioinformatics, 33(4), 2017, 537–544
doi: 10.1093/bioinformatics/btw655
Advance Access Publication Date: 14 November 2016
Original Paper

Systems biology

RedNemo: topology-based PPI network reconstruction via repeated diffusion with neighborhood modifications

Ferhat Alkan[1,2] and Cesim Erten[3,*]

[1]Center for Non-coding RNA in Technology and Health, [2]Department of Veterinary Clinical and Animal Sciences, University of Copenhagen, Grønnegardsvej 3, Frederiksberg, DK1870, Denmark and [3]Department of Computer Engineering, Kadir Has University, Cibali, 34083 Istanbul, Turkey

*To whom correspondence should be addressed.

Associate Editor: Jonathan Wren

Received on April 19, 2016; revised on October 4, 2016; editorial decision on October 10, 2016; accepted on October 12, 2016

Abstract

Motivation: Analysis of protein–protein interaction (PPI) networks provides invaluable insight into several systems biology problems. High-throughput experimental techniques together with computational methods provide large-scale PPI networks. However, a major issue with these networks is their erroneous nature; they contain false-positive interactions and usually many more false-negatives. Recently, several computational methods have been proposed for network reconstruction based on topology, where given an input PPI network the goal is to reconstruct the network by identifying false-positives/-negatives as correctly as possible.

Results: We observe that the existing topology-based network reconstruction algorithms suffer several shortcomings. An important issue is regarding the scalability of their computational requirements, especially in terms of execution times, with the network sizes. They have only been tested on small-scale networks thus far and when applied on large-scale networks of popular PPI databases, the executions require unreasonable amounts of time, or may even crash without producing any output for some instances even after several months of execution. We provide an algorithm, RedNemo, for the topology-based network reconstruction problem. It provides more accurate networks than the alternatives as far as biological qualities measured in terms of most metrics based on gene ontology annotations. The recovery of a high-confidence network modified via random edge removals and rewirings is also better with RedNemo than with the alternatives under most of the experimented removal/rewiring ratios. Furthermore, through extensive tests on databases of varying sizes, we show that RedNemo achieves these results with much better running time performances.

Availability and Implementation: Supplementary material including source code, useful scripts, experimental data and the results are available at http://webprs.khas.edu.tr/~cesim/RedNemo.tar.gz

Contact: cesim@khas.edu.tr

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Proteins and their interactions constitute the core of almost every biological process. In protein–protein interaction (PPI) networks nodes represent the proteins and the edges correspond to interactions between pairs of proteins. Analysis of this specific type of biological network is quite central in the study of several systems biology problems that include understanding cell regulatory mechanisms, extracting protein functions, constructing pathways or protein complexes and predicting evolutionary patterns.

Several high-throughput experimental techniques including the yeast two-hybrid system (Finley and Brent, 1994) and co-immunoprecipitation coupled mass spectrometry (Aebersold and Mann, 2003) gave rise to extraction of large-scale PPI networks for many organisms. Although experimental techniques for the prediction of PPI networks are quite useful in producing massive data, they are usually time-consuming and expensive. Thus, many approaches based on a wide range of computational techniques have been suggested for the problem of PPI network construction (Aebersold and Mann, 2003; Goh and Cohen, 2002; Marcotte *et al.*, 1999). Existing computational methods vary depending on the type of information they employ for predictions. These include methods based on genomic context, structure, domain or sequence information; see Skrabanek *et al.* (2008); Xia *et al.* (2010) for useful reviews.

A crucial problem with the constructed networks is that the extracted set of interactions may provide erroneous results in terms of false-positives and false-negatives. With regards to the experimental techniques this is mostly due to the significant levels of noise, whereas the computational methods suffer both from the employed heuristics that may not solve the defined computational problems optimally and also from the noisy data, since one way or another they all employ some kind of experimental data. Therefore several computational methods have been developed for network reconstructions where given an erroneous network as input, the goal is to identify false-positive and false-negative interactions as correctly as possible, and reconstruct a novel network.

Several network reconstruction methods are based solely on network topology (Cannistraci *et al.*, 2013; Hulovatyy *et al.*, 2014; Lei and Ruan, 2013). Such methods usually rely on making new interaction predictions or identifying spurious interactions by computing a topology-based *similarity score* for all pairs of nodes of the network. The topology information employed in the construction of these similarity scores might be local or global. Appropriate measures defined on common neighborhoods or extended neighborhoods such as CN (common neighbors), AA (Adamic/Adar) and RA (resource allocation) are based on local topology (Zhu and Xia, 2015), whereas diffusion-based distances including RWR (random walk with restart) (Tong *et al.*, 2006), RWS (random walk with resistance) (Lei and Ruan, 2013) or distances based on geometric embedding (Kuchaiev *et al.*, 2009) are sample methods employing global topological information. A common feature of these reconstruction methods is that once a scoring measure is determined, the network is modified with respect to a global evaluation of all the similarity scores. In addition to topology-based methods, PPI network reconstruction algorithms making use of sequence information, 3D structural data, or GO term associations have also been proposed (Mosca *et al.*, 2014; Segura *et al.*, 2015; Singh *et al.*, 2006; Yerneni *et al.*, 2015).

We propose a novel topology-based network reconstruction method REpeated Diffusion with NEighborhood Modifications (RedNemo). One main novelty of RedNemo is its iterative nature. In addition, although some version of topological similarity scoring is employed in previous methods as well, we depart from previous work after this point and employ a local neighborhood evaluation of these scores for network modifications. We show that both of these key points work with each other well and produce networks that are biologically more relevant than or at least comparable to those of two recent alternative topology-based reconstruction algorithms, RWS (Lei and Ruan, 2013) and GDV (Hulovatyy *et al.*, 2014). Considering network recovery when the reconstruction is applied on a randomly modified high-confidence network as another performance metric, we show that RedNemo yields better results than the alternatives in most settings. Furthermore, we show that RedNemo not only provides higher quality networks, but also achieves this with much better time efficiency than the benchmark methods, due to its novel features.

2 Methods

The traditional network reconstruction algorithms can be identified by their *global-modifications* (GM) nature; once a proximity matrix denoting the *closeness* of pairs of nodes is computed based on the network topology (local or global), this matrix is employed as a whole to reconstruct the modified network globally. The choice to add new edges or remove existing edges is based solely on how large or small values recorded in this scoring matrix are, without any regard for the localities of these achieved low/high scores. Several issues reminiscent of the GM approaches are handled by the novel key points of RedNemo. The pseudocode is provided in Algorithm 1 and is described in detail in the following subsections.

2.1 General framework of RedNemo

The traditional GM approaches aim at one-shot network reconstructions; once a scoring is computed, the final network is reconstructed accordingly once and for all. On the other hand, RedNemo employs the simple idea that if a network reconstruction algorithm is asserted to be successful, a succeeding execution of the same algorithm on the reconstructed network should yield an even better reconstruction, which should hold after repeated executions until convergence. Thus, the general framework of RedNemo is iterative by nature. Interestingly, reconstruction qualities decrease considerably when the GM-based benchmark algorithms RWS and GDV are employed iteratively. At each iteration of RedNemo, a relatively small number of adjustments on the network are made, allowing the following iterations to work on a more reliable network and modify it with small adjustments each time. RedNemo has two main steps, the first of which constructs diffusion-based proximities of all pairs of nodes. Based on these proximity scores, the second step computes correlation scores between node pairs within local neighborhoods and modifies the network in its localities. These steps are repeated iteratively by assigning the network reconstructed at the end of an iteration as the input for the next iteration until the final output network is generated.

2.2 Diffusion-based proximities

Lines 5 – 15 of Algorithm 1 describe the first main step of computing a diffusion-based proximity matrix M. This is implemented via simulating the probabilistic traversal of a random walker moving between neighboring nodes throughout several time steps. We note that similar diffusion-based methods that mimic the flow of information in a network via random walks have been employed in many previous PPI network analysis studies (Cao *et al.*, 2014; Leiserson

Algorithm 1. RedNemo

Input: Network $G = (V, E)$, positive integer values r, k, positive insertion/deletion ratio Ω
Output: Reconstructed network G
repeat
  //Diffusion-based Proximities
  for $\forall u \in V$ **do**
    initialize $M'[u, v]$ for $\forall v \in V$
    repeat
      for $\forall v \in V$ **do**

$$M[u, v] = \sum_{s \in N^+(v)} max(0, \frac{M'[u, s]}{deg(s) + 1} - \epsilon)$$

      end for
      normalize $M[u, v]$ for $\forall v \in V$

$$totaldiff = \sum_{v \in V} - - |M[u, v] - M'[u, v]|$$

      $M'[u, v] = M[u, v]$ for $\forall v \in V$
    until $totaldiff < threshold$
  end for
  //Neighborhood Modifications
  $\mathcal{R} = \varnothing$
  for $\forall u \in V$ **do**
    Create subgraph $G_u = (N_u, E_u)$
    Construct sorted lists I_u, D_u
    for corresponding $(p, q) \in I_u, (p', q') \in D_u$ **do**
      if $\mathcal{P}(p, q) > 0$ **and** $\mathcal{P}(p, q) > \mathcal{P}(p', q')$ **then**
        add pair $\prec (p, q), (p', q') \succ$ into $\mathcal{R}$
      end if
    end for
  end for
  sort $\prec i, d \succ$ pairs in $\mathcal{R}$ in descending order of $\mathcal{P}(i) - \mathcal{P}(d)$
  commit first $\frac{|\mathcal{R}|}{r \times k}$ modifications of $\mathcal{R}$ on G
  if |insertions| / |deletions| $> \Omega$
    undo final insertions to balance
  else if |insertions| / |deletions| $< \Omega$
    undo final deletions to balance
  $cc = cc +$ *min of committed insertions or deletions*
until *convergence or* $cc \geq$ *modification amount*

et al., 2015; Wang and Qian, 2014). For $v \in V$, let $N^+(v)$ denote the set of neighbors of v together with v itself and $deg(v)$ denote the degree of v in G, that is $deg(v) = |N^+(v)| - 1$. Assuming the origin of the walk is node u, let $M'[u, v]$ denote the probability that the random walker is at node v after a certain number of time steps and $M[u, v]$ denote the same probability after one more time step. Initially, $M'[u, u] = 1$, $M'[u, v] = 0$ for $v \neq u$. $M[u, v]$ is computed from $M'[u, s]$ for $s \in N^+(v)$. Moving from a node through any incident edge in the next time step has the same probability. Thus, the contribution of a neighbor s of v to $M[u, v]$ is $\frac{M'[u,s]}{deg(s)+1}$. Similar to the teleportation idea of the *random walk with restart* (Lei and Ruan, 2013; Tong *et al.*, 2006), a small constant ϵ is decremented from this contribution to increase the chances of the walker remaining close to the origin. Once $M[u, v]$ is computed for every v, each such probability is normalized by dividing it with $\sum_{v \in V} M[u, v]$. This process of computing the probability of random walker being at node v after the current time step, from the probabilities of the walker being at the neighboring nodes in the previous time step is repeated until the total difference of the computed probabilities converge, that is the sum of the differences of probabilities with those of the previous time step does not exceed a predefined constant *threshold*. The constants ϵ and *threshold* are set to $\frac{|V|}{(2|E|)^2}$ and $\frac{|V|}{2|E|}$, respectively.

2.3 Neighborhood modifications

The proximity scores matrix M computed in the previous step is employed in scoring and selecting *bad* pairs among existing edges and *good* pairs among non-existing edges in this step. The bad pairs are candidates for deletion whereas the good pairs are candidates for insertion as new edges. Although the proximity computations of the previous step are global in the sense that the proximity between a pair of nodes is affected by all nodes of the network that are in the same connected component as the pair, the candidate modifications computed in this step are restricted to local neighborhoods.

Lines 17 – 33 of Algorithm 1 provide a description of this step. Corresponding to each node $u \in V$, first a neighborhood set N_u is created. It consists of the nodes with a graph-theoretical distance of at most r to u in G, that is $v \in N_u$ if and only if the shortest path distance of u to v in G is less than or equal to r. Next the subgraph $G_u = (N_u, E_u)$ of G induced by the node set N_u is created. From G_u and the proximity matrix M computed in the previous step, we construct insertion candidates list, I_u, and deletions candidates list, D_u. The former consists of the *best* k node pairs $p, q \in N_u$ such that $(p, q) \notin E_u$ whereas the latter consists of the *worst* k pairs $p, q \in N_u$ such that $(p, q) \in E_u$. The goodness of a pair p, q is proportional to the average of the proximities in both directions, that is $\frac{M[p,q]+M[q,p]}{2}$. Here k is a user-defined parameter that controls the maximum number of modifications (insertions/deletions) allowed in a neighborhood N_u. Although our implementation is general so that any value can be assigned for r, k the default values employed in all the experimental evaluations are 1, 2, respectively.

Next, the lists I_u, D_u are sorted with respect to the average proximity values, the former in descending and the latter in ascending order, respectively. The indices of the pairs in these sorted lists provide a matching between the pairs in I_u, D_u such that the highest proximity pair in I_u matches the lowest proximity pair in D_u, so on and so forth. Thus, the best non-edge (insertion candidate) is matched with the worst edge (deletion candidate). Starting with the smallest index and traversing the two lists simultaneously we create a candidate list $\mathcal{R}$ of replacement pairs. For $(p, q) \in I_u$ and the matching pair $(p', q') \in D_u$ at the same index, we place $\prec (p, q), (p', q') \succ$ into $\mathcal{R}$ if $\mathcal{P}(p, q) > 0$ and $\mathcal{P}(p, q) > \mathcal{P}(p', q')$, where $\mathcal{P}$ denotes the Pearson correlation coefficient of the column vectors of M corresponding to the input node pair. In other words, the replacement, that is deletion of the existing edge (p', q') and insertion of new edge (p, q) is a candidate modification, if the proximity vectors of p, q yield a positive correlation and they are better correlated than the proximity vectors of p', q'. Once candidate replacements from all neighborhoods are collected in $\mathcal{R}$, the replacements are sorted in descending order of their *priorities*, where the priority of a replacement in the form of an insertion/deletion pair $\prec i, d \succ$ is $\mathcal{P}(i) - \mathcal{P}(d)$. Among all replacements in $\mathcal{R}$ we commit the best $\frac{|\mathcal{R}|}{r \times k}$ replacements. User provided Ω determines the desired ratio of the number of committed insertions to the number of committed deletions at each iteration. An appropriate number of committed insertions or deletions are undone to preserve this ratio. We note that

even if $\Omega = 1$, due to possible neighborhood subgraph overlaps the number of committed insertions may not be equal to the number of deletions. Thus, the excess insertions or deletions committed last may need to be uncommitted at a certain iteration in this setting as well.

Although we do not provide a theoretical proof of convergence, at which point no further modifications are committed, we note that the algorithm, when forced to execute until convergence, converges for all the networks under study. With the exception of very few instances, the number of modifications committed at each iteration decreases monotonically; a large portion of all the modifications are committed during the first few iterations of the algorithm. Moreover, the time spent for the rest of the iterations does not seem to be justified by the biological qualities of the produced networks. In contrast, the networks produced when the algorithm is executed until convergence are slightly worse than those produced when the iterations are stopped after the first few iterations. Therefore, we introduce an extra heuristic stopping condition; the iterations stop when the algorithm converges or when the total number of replacements committed throughout all iterations, denoted with *cc* in the algorithm, exceeds the *modification amount* set by the algorithm. This amount is the size of the replacement set $\mathcal{R}$ computed right after the first iteration. Further details regarding this discussion can be found at the Supplementary Material.

Note that the traditional GM approaches may lead to insertions/deletions cluttered in certain parts of the network; an interaction might be predicted at the expense of an existing interaction at a completely unrelated part of the network. With RedNemo, every committed interaction insertion (deletion) has a corresponding deletion (insertion) within some locality; if an existing interaction is considered highly false-positive, it is replaced with a novel interaction predicted to be highly false-negative within the same local neighborhood. Nevertheless, this locality-based modifications approach does not necessarily imply blind one-to-one insertion/deletion modifications within defined neighborhoods. Overlaps between the interactions of the constructed neighborhoods provide an indirect voting scheme. A given neighborhood subgraph may end up with a single insertion and several deletions at the end of an iteration for instance; see Figure 1(i). Furthermore with the traditional GM approaches, there is no control over the distance of newly connected pairs of nodes; predictions between pairs of nodes that are too far apart in the original network are quite possible. RedNemo handles this issue by allowing modifications to be committed only within neighborhood subgraphs. However, this is not a hard constraint. Due to the iterative nature of the algorithm, the neighborhood subgraphs are not static and may grow or shrink throughout iterations which provides an opportunity to make novel predictions between distant pairs as well, only with smaller chances; see Figure 1(ii). Finally, it should be emphasized that the locality-based modifications of RedNemo provide huge gains in terms of the required execution times as compared to the traditional GM approaches. A comparative evaluation of the running times can be found at the end of the following section.

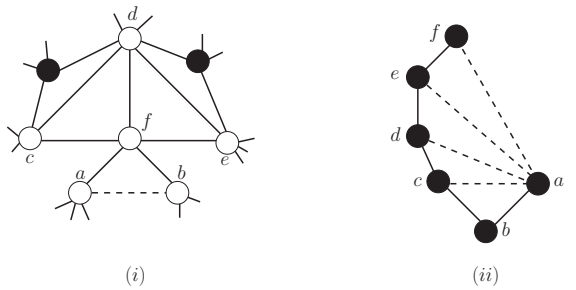

Fig. 1. Let $r = k = 1$. (i) Depiction of the neighborhood overlaps. N_f consists of white nodes. Let $\prec (a, b), (d, f) \succ$ be the replacement pair of G_f. (*c, d*) or (*e, d*) may also be eliminated from G_f if the black node on the left (or right) assigns it as a deletion in the replacement pair of its own neighborhood graph. (ii) Depiction of predictions between distant pairs via the iterative nature of RedNemo. $a, c \in N_b$, thus (*a, c*) may be inserted in G_b after iteration 1. Now $a, d \in N_c$, thus (*a, d*) may be inserted in G_c after iteration 2, so on and so forth. After four iterations, it is possible to connect *a, f* which originally had a distance of 5.

3 Discussion of results

We implemented the RedNemo algorithm in C++ using the LEDA library (Mehlhorn and Naher, 1999). The source code, executables, useful scripts for evaluations and all the input data are freely available as part of the supplementary material. We employ PPI networks of several databases for a comparative evaluation of RedNemo against the benchmark methods RWS and GDV. For RWS, we use the suggested parameter settings of $\epsilon = |V|/|E|^2$ and $\beta = 1/|E|$, and for GDV we use the default settings provided by the executables of Hulovatyy *et al.* (2014). Note that for a given input network, both of these benchmark algorithms produce a matrix of similarity scores and extract the desired number of top scoring pairs from this matrix as edges in the reconstructed network. The databases are categorized as small, medium and large. First, as part of small-scale networks, we evaluate the algorithms on three yeast datasets: one produced via Y2H (Solava *et al.*, 2012; Yu *et al.*, 2008), one via AP/MS (Solava *et al.*, 2012; Yu *et al.*, 2008) and one obtained from multiple sources (Collins *et al.*, 2007). Note that these are the datasets used by GDV (Hulovatyy *et al.*, 2014). The medium-sized networks involve those obtained from the IsoBase database (Park *et al.*, 2011). This dataset has recently been used in many PPI network analysis studies (Aladağ and Erten, 2013; Alkan and Erten, 2014, 2015; Park *et al.*, 2011; Sahraeian and Yoon, 2013). The networks under study from this database are those of *C. elegans*, *D. melanogaster*, *H. sapiens* and *S. cerevisiae*. Finally, as part of large-scale networks, we employ those of the same species from the October, 2015 version of the IntAct database (Orchard *et al.*, 2014). The networks are filtered to include only protein–protein interactions. Note that the results presented in Sections 3.1 and 3.2 apply to the scenario where the input networks and the corresponding reconstructions are of the same size in terms of the number of edges, that is Ω is set to 1.

3.1 Analysis of network properties

Table 1 provides network properties of the original networks and those that are reconstructed by the three algorithms. The columns $|V|, |E|$ respectively indicate the number of nodes (those with degree more than 0) and edges in the network. Since the number of edges of a reconstructed network is the same as the corresponding original network, $|E|$ is not shown for a reconstructed network. Instead we provide *DP* (*difference percentage*) value, which is the percentage of the edges of the network that are different from those of the corresponding original network. The *CC* column indicates the number of connected components and the *BCC* column indicates the number of biconnected components of a network. An interesting comparison is regarding the DP values of the networks reconstructed by GDV, RWS and RedNemo. Note that the amount of change proposed by each algorithm is determined internally and thus each algorithm not only reconstructs a network but also indirectly provides a measure

Table 1. Properties of networks under consideration

		Original Network				GDV Output				RWS Output				RedNemo Output			
DB	Net	$\|V\|$	$\|E\|$	CC	BCC	$\|V\|$	DP	CC	BCC	$\|V\|$	DP	CC	BCC	$\|V\|$	DP	CC	BCC
Various	Y2H	1647	2518	1	925	368	95	1	122	1534	45	271	693	1647	18	29	792
Yeast	APMS	1004	8319	1	155	203	60	1	16	916	35	90	128	999	7	36	162
Databases	Collins	1004	8323	1	155	202	60	1	15	916	36	90	131	999	7	36	161
	ce	2974	4827	123	1682	496	92	1	70	2597	55	472	1030	2974	16	140	1296
	dm	7387	24937	57	2229	1446	91	1	229	7224	65	234	1063	7387	20	57	924
IsoBase	hs	10296	54654	62	2219					9099	73	567	1833	10296	16	62	1295
	sc	5523	82656	1	170					1348	88	111	217	5523	2	1	171
	ce	5102	11829	123	2547	415	84	1	46	4304	65	862	1692	5102	16	135	1833
	dm	11213	40813	67	3430					9412	76	792	2809	11213	15	68	2129
IntAct	hs	16434	99379	97	4156					10539	85	1385	2656	16434	11	97	2955
	sc	6055	76742	6	479					3829	85	210	713	6055	3	6	477

Columns *DB* and *Net* provide the name of the database and the network, respectively. The abbreviations *ce, dm, hs, sc* stand, respectively, for the *C.elegans, D.melanogaster, H.sapiens*, and the *S.cerevisiae* networks. Each row provides the measured values for the network under the *Net* column. No results could be obtained for GDV on instances where values are left blank.

of its prediction of false-positive rates through the amount of change it suggests. Both GDV and RWS provide large DP values; even for each of the relatively high-confidence yeast data of APMS and Collins networks, GDV predicts 60%, RWS predicts almost 35% of the original network interactions to be false. Respective values of the RedNemo reconstructions of APMS and Collins networks are 7%. Although yeast is a well-studied organism and thus yeast networks in general are assumed to have higher confidence than those of the other species, considering the multi species databases of IsoBase and IntAct, RWS reconstructions provide the largest DP values for the *S. cerevisiae* networks: 88% for the IsoBase network and 85% for the IntAct network. Respective DP values of the RedNemo reconstructions are the smallest among networks of all species, 2 and 3.

The issue of clustering in certain areas at the expense of disconnecting various unrelated regions of the network discussed at the end of the previous section becomes evident with a simultaneous inspection of $|V|$, *CC*, and *BCC* values. Comparing the $|V|$ values of the original networks and those of the GDV, it is clear that in most of the cases GDV appears to distribute all the network edges at only one fifth of the nodes, disconnecting all the rest. Although not as drastic, RWS also suffers from this issue, especially for the Isobase *S. cerevisiae* network and the Intact *H. Sapiens* and *S. cerevisiae* networks. The same problem affects the *CC* and *BCC* values as well; since a large portion of the network becomes very sparse the number of connected components increases dramatically, whereas edge concentration in certain regions which were previously less densely populated decreases the number of biconnected components. Note that although *CC* seems to be 1 for the GDV networks, considering the differences in $|V|$ values of the reconstructions and those of the originals, the actual *CC* values are much larger. For instance the actual *CC* values for the Y2H networks are 1280 and 384, for the GDV and RWS reconstructions respectively. The difference between the topological properties of the network reconstructions of RWS and GDV may be due to several design choices the two approaches have. However, it is worth noting that one of the main differences between RWS and GDV is that the former, similar to RedNemo, has a scoring computation based on global topology, whereas the latter constructs the scoring matrix based on local topological information. The correlation of a pair of vectors, each recording random walk-based distances from a node to all network nodes, is an indication of 'similarity' in RWS. For GDV, on the other hand, the similarity is sought in the correlation of vectors that record local information in the form of existence/absence of subgraphs isomorphic to predetermined structures in local neighborhoods of the nodes.

3.2 Analysis of biological qualities

Two databases are employed in setting up the evaluation metrics; the Gene Ontology (GO) database (Ashburner *et al.*, 2000) and the STRING database (Szklarczyk *et al.*, 2015). The GO database annotates proteins from several species with appropriate GO categories organized as a directed acyclic graph (DAG) (Ashburner *et al.*, 2000). In order to standardize the GO annotations of proteins, similar to the evaluation methods of Singh *et al.* (2008), Liao *et al.* (2009) and Aladağ and Erten (2013), we restrict the protein annotations to level 5 of the GO DAG by ignoring the higher-level annotations and replacing the deeper-level category annotations with their ancestors at the restricted level. Note that only experimentally determined annotations are employed. For a node $u \in V$, let $GO(u)$ indicate the set of experimentally determined GO annotations of the protein corresponding to u. An edge (x, y) is considered *annotated*, if $GO(x) \neq \emptyset$ and $GO(y) \neq \emptyset$. Our GO-based evaluations consider only the annotated edges. An annotated edge (x, y) is considered *consistent* if $GO(x) \cap GO(y) \neq \emptyset$. As part of the GO-based evaluations we employ three metrics. *NCE* represents the number of consistent edges, whereas *CER* denotes the ratio of number of consistent edges to the number of all annotated edges. Finally, *GOC* is defined as the sum of $|GO(x) \cap GO(y)|/|GO(x) \cup GO(y)|$ over all annotated edges (x, y). The GO-based evaluations of the algorithms on all the networks are shown in Figure 2. For the Y2H network, RedNemo provides the largest NCE while being able to provide a quite large ratio in terms of CER. It also provides the largest GOC value for this instance. For the APMS and Collins datasets, both RWS and RedNemo provide the largest NCE values accompanied with quite large CER values, although GOC value of RWS is slightly better than that of RedNemo. For all these three network instances, GDV performs quite poorly in all metrics. Note that among the three networks Y2H is the low confidence one and RedNemo reconstruction performing even better than the original network in all metrics is notable. For the Isobase *C. elegans* network RedNemo performs the best in terms of NCE accompanied with a large CER, whereas RWS is better in terms of the GOC score. GDV performs the poorest. On the other hand, for the IntAct network of the same

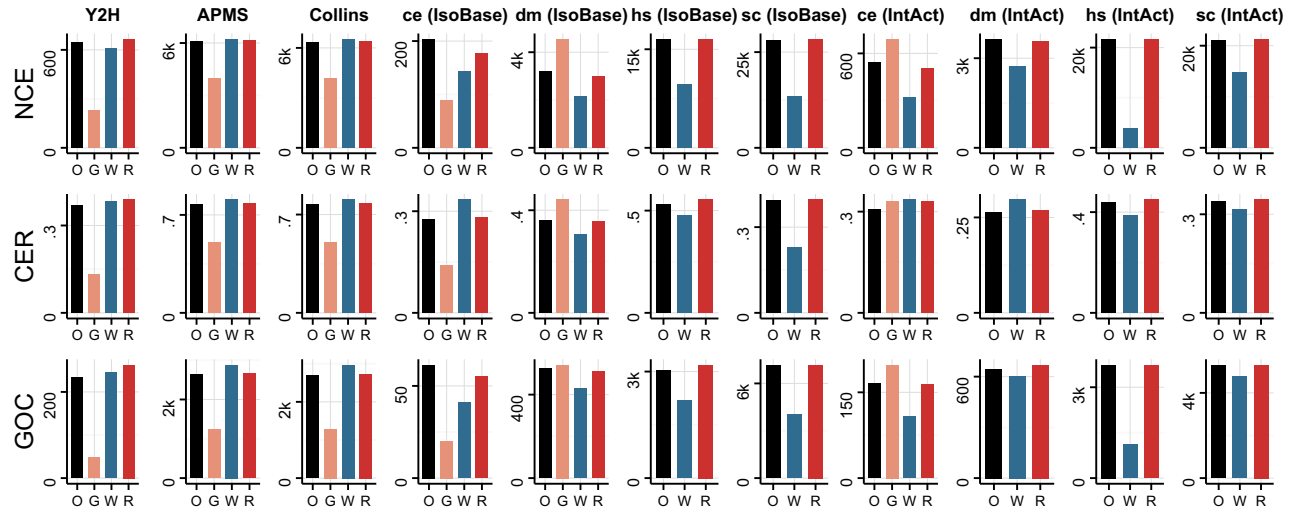

Fig. 2. GO-based evaluations of the reconstructed networks. O, G, W, R, respectively, stand for the original network and the network reconstructions of GDV, RWS and RedNemo. The abbreviations *ce, dm, hs, sc* are the same as those in Table 1 (Color version of this figure is available at *Bioinformatics* online.)

species GDV performs the best, followed by RedNemo which has slightly better scores than RWS. Same holds for the IsoBase *D. melanogaster* network; GDV is the best and RWS is the worst performer. For all the rest of the networks, namely the IntAct *D. melanogaster* network, the IsoBase and IntAct *H. sapiens* networks, and the Isobase and IntAct *S. cerevisiae* networks RedNemo performs the best by a fair margin in all metrics. We note that we implemented one additional GO-based metric, *cGOC* (complementary GOC), with a definition similar to that of GOC on the complement of the output network. The performances of the reconstructions by alternative methods were more or less similar to those obtained via GOC. Details regarding these tests can be found in the Supplementary Material.

Regarding the STRING-based evaluations, we make use of the metrics *neighborhood (Ngh)*, *fusion (Fus)*, *cooccurence (Coo)*, *coexpression (Coe)*, *experimental (Exp)*, *database (Dat)*, *textmining (Txt)* and *combined (Com)* as defined in von Mering *et al.* (2005). Given a protein–protein interaction, the STRING database provides for each metric a confidence score for the interaction. For a PPI network, for each metric we compute an average of the scores over all interactions. Due to space limitations the STRING-based evaluations of the algorithms can be found in the Supplementary Material. With respect to three important metrics, *Exp*, *Dat* and *Com*, RedNemo provides the best scores in all network instances. To summarize the complete results with all of the eight metrics, for the Y2H network, RWS yields best scores in two metrics, RedNemo in four metrics and they have a tie in two, whereas GDV provides the worst scores in all metrics. Considering the APMS and Collins networks together, GDV yields the best scores in four instances, RWS is the best in two, and RedNemo is the best in ten instances. Considering the IsoBase and IntAct databases together for four networks of each database and eight different metrics we have 32 instances in total. For all these 32 instances, RWS networks provide the best scores in 12 and RedNemo scores the best in the remaining 20 instances.

3.3 Analysis of network growth qualities

In addition to network reconstructions aiming at producing a network with the same number of edges, we employed tests to evaluate the success of the algorithms when the network size increased by setting Ω to certain values greater than one. For all the networks under consideration, the relative performances of the algorithms in terms of GO-based and STRING-based scores are the same as those of the same-size reconstructions to a large extent. Due to space limitations, detailed results are given in the Supplementary Material.

3.4 Recovery of random removals/rewirings

In another evaluation context, we performed random removals and rewirings on the high confidence Collins network and we compared the performances of the alternative methods in recovering the original Collins network. We compared the methods using three evaluation metrics, area under ROC curve (AUROC) and area under precision-recall curve (AUPR) for the performances of reconstructions on networks with different levels of edge removals, and true positive rate (TPR) for the evaluations of same size reconstructions on networks with different levels of random rewirings. AUROC and AUPR measures are employed in Hulovatyy *et al.* (2014) as well for benchmarking purposes. However, since RedNemo does not aim to provide a scoring matrix as output, we used a slightly different approach for a fair comparison between alternative methods. For each removal ratio (from 0.05 to 0.5 with increments of 0.05) and randomization (total of 10), we generated the corresponding input networks from high confidence Collins network. For each network, we performed 16 network reconstructions with RedNemo and GDV-scored-RedNemo setting the Ω parameter to 0.4, 0.6, 0.8, 1, 1.1, 1.25, 1.5, 2, 2.5, 3, 5, 10, 15, 20, 25, 50 and 100. At each instance, the size of the RedNemo output network is set as the reference size; that many of the best scoring entries of the GDV and RWS scoring matrices constitute the output networks of GDV and RWS. Using the original Collins network as ground data, we determined the TPR, false-positive rate (FPR) and positive predictive value (PPV) statistics for all resulting networks and we averaged these statistics over ten randomizations. At the end, for each alternative method and removal ratio, we plotted the ROC (FPR versus TPR), precision-recall (TPR versus PPV) curves and we computed the area under them which correspond to AUROC and AUPR, respectively; see Figures 3 and 4. RedNemo clearly outperforms RWS and GDV when reconstructing the Collins network with randomly removed edges. Due to space limitations, the original curves are presented in the Supplementary Material.

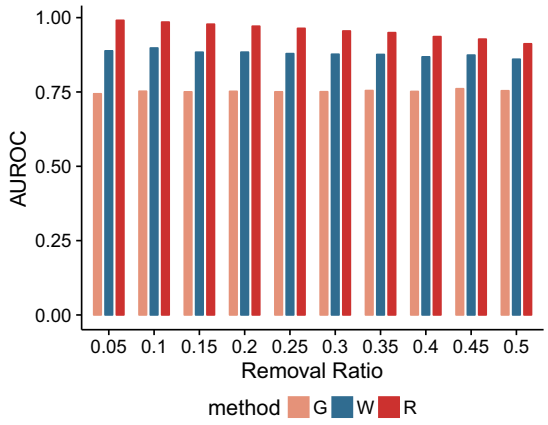

Fig. 3. AUROC performances of GDV, RWS and RedNemo when recovering the original networks from flawed networks with different edge removal ratios (Color version of this figure is available at *Bioinformatics* online.)

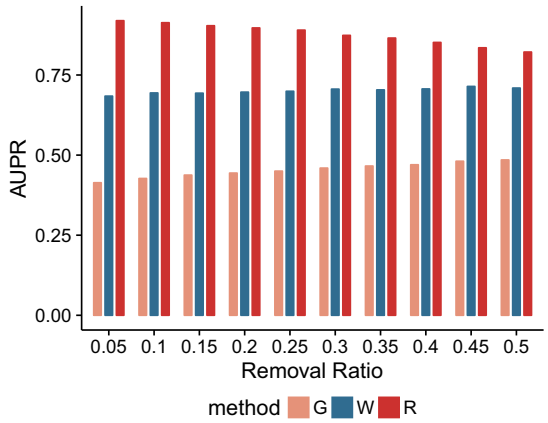

Fig. 4. AUPR performances of GDV, RWS and RedNemo when recovering the original networks from flawed networks with different edge removal ratios (Color version of this figure is available at *Bioinformatics* online.)

In the random rewiring case, we evaluated the same size reconstruction performances of all methods by only looking at the TPR measure. From the high confidence Collins network, we generated 10 randomly rewired networks for each rewiring ratio and we reconstructed them by applying the three methods. Using the original Collins network as ground data, we plotted the TPRs for each method and rewiring ratio in box plot format to show the distributions over 10 randomizations; see Figure 5. RedNemo succeeded in reconstructing the original network for all rewiring ratios above 0.1 and it performed much better than other methods for rewiring ratios under 0.4.

3.5 Required execution times

A major drawback of the benchmark algorithms GDV and RWS is in terms of the execution times they require. We note that GDV could not be executed until completion on the medium-sized IsoBase *H. sapiens, S. cerevisiae* networks and the large-scale IntAct *D. melanogaster, H. sapiens, S. cerevisiae* networks. The program crashed before completion after several weeks of execution. Note that the tests for computation times were conducted on powerful cluster node with eight dedicated CPUs (x86_64, GenuineIntel) and 120 GB of memory. The largest network for which GDV provides reconstructions is the IsoBase *D. melanogaster* network. We note

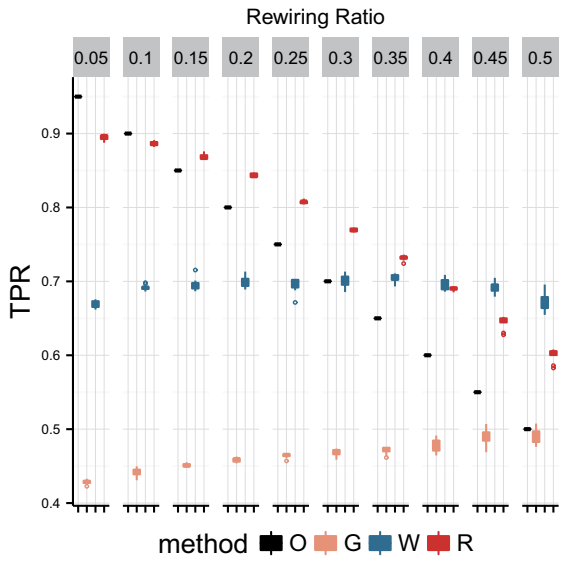

Fig. 5. True positive rates when reconstructing the randomly rewired Collins networks with different noise ratios. Results are given in box plots showing the distribution of TPR over 10 different randomizations (Color version of this figure is available at *Bioinformatics* online.)

that for this network GDV required 408 min, RWS required 124 min, whereas RedNemo executed in just 8 min. For the largest dataset of the IntAct, *H. sapiens* network RWS execution took 1415 min whereas RedNemo execution took 111 min. We note that the number of iterations of RedNemo is usually a small constant and for the employed r, k values, the running time of RedNemo is bounded by $O(|V|(\Delta^2 + \log|V|))$, where Δ denotes the maximum degree of G. A detailed running time analysis together with execution times on all the network instances can be found in the Supplementary Material.

4 Conclusion

We provided an algorithm, RedNemo, for the topology-based reconstruction of PPI networks. The novelties of RedNemo include iterative network refinements and the modifications localized in small neighborhoods. Associating a proposed edge prediction with an edge deletion and the indirect voting mechanism via neighborhood overlaps provides a balance between the number of proposed modifications and their concentrations in different network regions. We showed that RedNemo provides better performances than the alternatives in most cases when metrics based on GO or STRING data are considered. It also outperforms the alternatives in reconstructing the original network when random rewirings/removals are introduced under most rewiring/removal ratio settings. Furthermore, it has much better running time performance than the alternatives. For the construction of the proximity scores, we tested different diffusion-based scorings including random walk, random walk with restart and random walk with resistance (also employed in RWS). The results were more or less similar with those of the restart version slightly better than the others. To further put the neighborhood modifications concept of RedNemo into test, we implemented a version of RedNemo where the scores employed in this step are taken from scoring matrix produced by GDV. In almost all the cases, it outperformed the original GDV algorithm and in some cases it even performed better than all algorithms including the original RedNemo which is based on random walks-based scoring.

However, a limitation of employing GDV-scoring within the overall RedNemo framework is the absence of the iterative improvements step, due to the large computational requirements of GDV-scoring. This further limits the network growth scenario as the RedNemo framework depends on iterative improvements for network growth. Results in the Supplementary material include those obtained with this GDV-scoring based RedNemo version when applicable. Employing proximity scorings from different genres such as those based on graph-theoretical distances or on geometric embeddings within the general framework of RedNemo is part of future work.

Acknowledgement

This work was carried out while C. Erten was visiting the Bioinformatics group at CIPF, Centro de Investigación Príncipe Felipe.

Funding

The author thanks CIPF for their hospitality and acknowledges TUBITAK-BIDEB, grant no. 1059B191501053 which partially funded this research.

Conflict of Interest: none declared.

References

Aebersold,R. and Mann,M. (2003) Mass spectrometry-based proteomics. *Nature*, **422**, 198–207.

Aladağ,A.E. and Erten,C. (2013) Spinal: scalable protein interaction network alignment. *Bioinformatics*, **29**, 917–924.

Alkan,F. and Erten,C. (2014) Beams: backbone extraction and merge strategy for the global many-to-many alignment of multiple PPI networks. *Bioinformatics*, **30**, 531–539.

Alkan,F. and Erten,C. (2015) Sipan: simultaneous prediction and alignment of protein-protein interaction networks. *Bioinformatics*, **31**, 2356–2363.

Ashburner,M. *et al*. (2000) Gene Ontology: tool for the unification of biology. *Nat. Genet*., **25**, 25–29.

Cannistraci,C.V. *et al*. (2013) Minimum curvilinearity to enhance topological prediction of protein interactions by network embedding. *Bioinformatics*, **29**, i199–i209.

Cao,M. *et al*. (2014) New directions for diffusion-based network prediction of protein function: incorporating pathways with confidence. *Bioinformatics*, **30**, 219–227.

Collins,S.R. *et al*. (2007) Toward a comprehensive atlas of the physical interactome of *Saccharomyces cerevisiae*. *Mol. Cell Proteomics*, **6**, 439–450.

Finley,R.L. and Brent,R. (1994) Interaction mating reveals binary and ternary connections between drosophila cell cycle regulators. *Proc. Natl. Acad. Sci. USA*, **91**, 12980–12984.

Goh,C.S. and Cohen,F.E. (2002) Co-evolutionary analysis reveals insights into protein-protein interactions. *J. Mol. Biol*., **324**, 177–192.

Hulovatyy,Y. *et al*. (2014) Revealing missing parts of the interactome via link prediction. *PLoS One*, **9**, 1–11.

Kuchaiev,O. *et al*. (2009) Geometric de-noising of protein-protein interaction networks. *PLoS Comput. Biol*., **5**, e1000454.

Lei,C. and Ruan,J. (2013) A novel link prediction algorithm for reconstructing protein-protein interaction networks by topological similarity. *Bioinformatics*, **29**, 355–364.

Leiserson,M.D. *et al*. (2015) Pan-cancer network analysis identifies combinations of rare somatic mutations across pathways and protein complexes. *Nat. Genet*., **47**, 106–114.

Liao,C.S. *et al*. (2009) Isorankn: spectral methods for global alignment of multiple protein networks. *Bioinformatics*, **25**, i253–i258.

Marcotte,E.M. *et al*. (1999) Detecting protein function and protein-protein interactions from genome sequences. *Science (New York, N.Y.)*, **285**, 751–753.

Mehlhorn,K. and Naher,S. (1999). *Leda: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge.

Mosca,R. *et al*. (2014) 3did: a catalog of domain-based interactions of known three-dimensional structure. *Nucleic Acids Res.*, **42(Database-Issue)**, 374–379.

Orchard,S. *et al*. (2014) The mintact project–intact as a common curation platform for 11 molecular interaction databases. *Nucleic Acids Res*., **42(Database issue)**, D358–D363.

Park,D. *et al*. (2011) Isobase: a database of functionally related proteins across PPI networks. *Nucleic Acids Res*., **39(Database-Issue)**, 295–300.

Sahraeian,S.M.E. and Yoon,B.J. (2013) Smetana: accurate and scalable algorithm for probabilistic alignment of large-scale biological networks. *PLoS One*, **8**, e67995.

Segura,J. *et al*. (2015) Using neighborhood cohesiveness to infer interactions between protein domains. *Bioinformatics*, **31**, 2545–2552.

Singh,R. *et al*. (2006). Struct2net: Integrating structure into protein-protein interaction prediction. In: *Proceedings of the Pacific Symposium on Biocomputing 2006, Maui, Hawaii, USA, 3–7 January 2006*, pp. 403–414.

Singh,R. *et al*. (2008). Global alignment of multiple protein interaction networks. *Pac. Symp. Biocomput*. 303–314.

Skrabanek,L. *et al*. (2008) Computational prediction of protein-protein interactions. *Mol. Biotechnol*., **38**, 1–17.

Solava,R.W. *et al*. (2012) Graphlet-based edge clustering reveals pathogen-interacting proteins. *Bioinformatics*, **28**, 480–486.

Szklarczyk,D. *et al*. (2015) STRING v10: protein-protein interaction networks, integrated over the tree of life. *Nucleic Acids Res*., **43(Database issue)**, 447–452.

Tong,H. *et al*. (2006). Fast random walk with restart and its applications. In: *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pp. 613–622.

von Mering,C. *et al*. (2005) String: known and predicted protein-protein associations, integrated and transferred across organisms. *Nucleic Acids Res*., **33(Database-Issue)**, 433–437.

Wang,Y. and Qian,X. (2014) Functional module identification in protein interaction networks by interaction patterns. *Bioinformatics*, **30**, 81–93.

Xia,J. *et al*. (2010) Computational methods for the prediction of protein-protein interactions. *Protein Pept. Lett*., **9**, 1069–1078.

Yerneni,S. *et al*. (2015) IAS: Interaction specific go term associations for predicting protein-protein interaction networks. *IEEE/ACM Trans. Comput. Biol. Bioinform*., **99**, 1.

Yu,H. *et al*. (2008) High-quality binary protein interaction map of the yeast interactome network. *Science*, **322**, 104–110.

Zhu,B. and Xia,Y. (2015) An information-theoretic model for link prediction in complex networks. *Nat. Sci. Rep*., **5**, 1–11.