

---

**Novel Computational Approaches for Multi-network Analysis to Improve Protein Function Prediction****Shawn Gu****Publication Date**

03-02-2022

**License**

This work is made available under a All Rights Reserved license and should only be used in accordance with that license.

**Citation for this work (American Psychological Association 7th edition)**

Gu, S. (2022). *Novel Computational Approaches for Multi-network Analysis to Improve Protein Function Prediction* (Version 1). University of Notre Dame. <https://doi.org/10.7274/2227mp51r18>

This work was downloaded from CurateND, the University of Notre Dame's institutional repository.

For more information about this work, to report or an issue, or to preserve and share your original work, please contact the CurateND team for assistance at [curate@nd.edu](mailto:curate@nd.edu).

NOVEL COMPUTATIONAL APPROACHES FOR MULTI-NETWORK  
ANALYSIS TO IMPROVE PROTEIN FUNCTION PREDICTION

A Dissertation

Submitted to the Graduate School  
of the University of Notre Dame  
in Partial Fulfillment of the Requirements  
for the Degree of

Doctor of Philosophy

by

Shawn Gu

---

Tijana Milenković, Director

Graduate Program in Computer Science and Engineering

Notre Dame, Indiana

February 2022

# NOVEL COMPUTATIONAL APPROACHES FOR MULTI-NETWORK ANALYSIS TO IMPROVE PROTEIN FUNCTION PREDICTION

Abstract

by

Shawn Gu

Networks can be used to model complex real-world systems from many domains, including computational biology. A protein-protein interaction (PPI) network (PPIN), in which nodes are proteins and edges are PPIs, is a popular type of biological network. While PPIN data are becoming widely available thanks to biotechnological advancements, functions of many proteins remain unknown. As such, many computational techniques have been developed to analyze PPINs in order to gain insights into proteins' functions.

One such technique is biological network alignment (NA), which aims to find a node mapping between species' molecular networks that uncovers similar network regions, thus allowing for the transfer of functional knowledge between the aligned nodes. However, a major issue of NA methods is that often aligned nodes (proteins) do not actually share the same function. So we aim to address such challenges by introducing several novel computational advances, such as allowing for the alignment of heterogeneous biological networks for the first time, or by learning from -omics data what patterns of network topological relatedness (rather than similarity) correspond to functional relatedness between biological networks of different species. We show that the novel computational advances improve the accuracy of across-species protein functional prediction compared to existing NA methods.

One limitation of across-species NA is that it only considers biological networks at the same scale: PPINs. However, at a more fine-grained scale, a protein’s 3D structure has important implications for its function. Such structures have been modeled using protein structure networks (PSNs), where nodes are amino acids and edges join those that are close in the 3D crystal structure, to great success. Thus, we argue that PPIN and PSN data should be integrated as a “network of networks” (NoN). We aim to answer whether NoN-based data integration is effective, by evaluating whether NoN-based protein functional prediction, fusing the complementary PPIN and PSN information, is more accurate than single-scale functional prediction, using only PPIN or only PSN information. We show that NoN-based data integration has the potential to uncover novel biological knowledge compared to only considering a single scale, and thus is an exciting direction for future research.



# CONTENTS

Figures . . . . .	v
Tables . . . . .	xxxii
Chapter 1: Introduction . . . . .	1
1.1 Overview . . . . .	1
1.2 Network alignment . . . . .	4
1.2.1 Background . . . . .	4
1.2.2 Related work . . . . .	7
1.2.3 Research questions and our contributions . . . . .	8
1.3 Modeling multi-scale data via a network of networks . . . . .	14
1.3.1 Background . . . . .	14
1.3.2 Related work . . . . .	15
1.3.3 Research questions and our contributions . . . . .	16
1.4 Organization of the dissertation . . . . .	19
Chapter 2: Pairwise versus multiple network alignment . . . . .	20
2.1 Introduction . . . . .	20
2.1.1 Background and motivation . . . . .	20
2.1.2 Our contributions . . . . .	22
2.2 Methods . . . . .	25
2.2.1 Data . . . . .	25
2.2.2 Network alignment methods that we evaluate . . . . .	27
2.2.3 Alignment quality measures . . . . .	28
2.2.4 Evaluation framework . . . . .	32
2.3 Results and discussion . . . . .	37
2.3.1 Topology versus topology+sequence alignments . . . . .	37
2.3.2 Method comparison: evaluation details . . . . .	39
2.3.3 Method comparison: results in the pairwise evaluation framework	42
2.3.4 Method comparison: results in the multiple evaluation framework	45
2.3.5 Method comparison: focusing on accuracy of protein function prediction . . . . .	49
2.4 Conclusion . . . . .	51

Chapter 3: Heterogeneous network alignment . . . . .	55
3.1 Introduction . . . . .	55
3.1.1 Background and motivation . . . . .	55
3.1.2 Our contributions . . . . .	56
3.2 Results and discussion . . . . .	62
3.2.1 Evaluation . . . . .	63
3.2.2 Comparison of homogeneous and heterogeneous network alignment . . . . .	67
3.3 Methods . . . . .	78
3.3.1 Calculating node similarities, i.e., node conservation . . . . .	78
3.3.2 From homogeneous to heterogeneous node conservation . . . . .	79
3.3.3 From homogeneous to heterogeneous edge conservation . . . . .	81
3.3.4 From homogeneous to heterogeneous network alignment . . . . .	82
3.4 Conclusion . . . . .	85
Chapter 4: Data-driven network alignment . . . . .	87
4.1 TARA: Data-driven network alignment . . . . .	87
4.1.1 Introduction . . . . .	87
4.1.2 Methods . . . . .	95
4.1.3 Results and discussion . . . . .	103
4.1.4 Conclusion . . . . .	123
4.2 TARA++: Data-driven network alignment that integrates topology and sequence to predict function . . . . .	125
4.2.1 Introduction . . . . .	125
4.2.2 Methods . . . . .	131
4.2.3 Results and discussion . . . . .	141
4.2.4 Conclusion . . . . .	155
Chapter 5: Modeling multi-scale data via a network of networks . . . . .	157
5.1 Introduction . . . . .	157
5.1.1 Our contributions . . . . .	158
5.1.2 Related work . . . . .	160
5.2 Methods . . . . .	162
5.2.1 Network of networks definition . . . . .	162
5.2.2 Problem statement . . . . .	162
5.2.3 Data . . . . .	163
5.2.4 Approaches for label prediction . . . . .	168
5.2.5 Evaluation . . . . .	171
5.3 Results and discussion . . . . .	172
5.3.1 Accuracy on synthetic networks of networks . . . . .	172
5.3.2 Accuracy on the biological network of networks . . . . .	176
5.3.3 Running time analysis . . . . .	181
5.4 Conclusion . . . . .	182

Chapter 6: Concluding remarks . . . . .	185
Appendix A: Pairwise versus multiple network alignment . . . . .	190
A.1 Methods . . . . .	190
A.1.1 NA methods that we evaluate . . . . .	190
A.1.2 Alignment quality measures . . . . .	196
A.1.3 Evaluation framework . . . . .	202
A.1.4 T versus T+S alignments . . . . .	204
A.1.5 Method comparison in the ME framework: accuracy versus running time . . . . .	204
A.2 Results . . . . .	206
A.3 Supplementary files . . . . .	238
Appendix B: Heterogeneous network alignment . . . . .	239
B.1 Results . . . . .	239
Appendix C: Data-driven network alignment . . . . .	252
C.1 TARA: Data-driven network alignment . . . . .	252
C.1.1 Results . . . . .	252
C.1.2 Supplementary files . . . . .	265
C.2 Towards TARA++: Integrating topology and sequence to prediction function . . . . .	266
C.2.1 Methods . . . . .	266
C.2.2 Results . . . . .	268
Appendix D: Modeling multi-scale data via a network of networks . . . . .	291
D.1 Methods . . . . .	291
D.1.1 Data . . . . .	291
D.1.2 Existing approaches for label prediction . . . . .	294
D.1.3 Our integrative GCN approach . . . . .	296
D.1.4 Evaluation . . . . .	300
D.2 Results . . . . .	304
D.2.1 Synthetic NoNs . . . . .	304
D.2.2 Biological NoN . . . . .	304
D.2.3 Running times . . . . .	317
D.3 Supplementary files . . . . .	319
Bibliography . . . . .	320

## FIGURES

1.1	Illustration of alignments produced by <b>(a)</b> local and <b>(b)</b> global NA. Dashed lines are between nodes that are aligned to each other. This figure is adapted from [110]. . . . .	5
1.2	Illustration of a <b>(a)</b> pairwise one-to-one alignment; <b>(b)</b> multiple one-to-one alignment; <b>(c)</b> pairwise many-to-many alignment; and <b>(d)</b> multiple many-to-many alignment. In this toy example we only show three networks for multiple NA, but multiple NA can be used on more than three networks as well. . . . .	7
1.3	Illustration of two heterogeneous networks, each containing different node as well as edge types (or colors). In a given network, different node shapes represent different node types, and different line styles represent different edge types. If we do not consider the ovals with red edges (the bottom portion of the network), then we have a heterogeneous network with different node types, and thus implicitly different edge types. If we only consider the ovals with blue or red edges, then we have a heterogeneous network with different edge types but a single node type (also called multimodal networks with two edge modes). The goal of HetNA as we define it is to find a node mapping between heterogeneous networks that contain different node types, different edge types, or both. . . . .	10

1.4	Illustration of the existing notion of topological similarity versus our new notion of topological relatedness. Suppose that we are aligning PPI networks of two different species, where for simplicity, only parts of the whole networks are shown. Also, suppose that a color corresponds to the function that a node performs, in this case the “purple” function or the “orange” function. <b>(a)</b> An NA method based on topological similarity will produce an alignment with low functional quality on our example networks. Such a method will align nodes $d$ , $e$ , $f$ , and $g$ in species 1 to nodes 1, 2, 3, and 8 in species 2 because each set of nodes forms the same subgraph: a square with a diagonal (square-with-diagonal). However, the species 1 nodes perform the “orange” function, while the species 2 nodes perform the “purple” function – the nodes are not functionally related. <b>(b)</b> On the other hand, an NA method based on topological relatedness will produce an alignment with high functional quality on our example networks. This is because such a method will learn that 3-node paths in species 1 should be aligned to square-with-diagonals in species 2, since the 3-node path consisting of nodes $a$ , $b$ , and $c$ in species 1 performs the same function (“purple”) as the square-with-diagonal consisting of nodes 1, 2, 3, and 8 in species 2; and that square-with-diagonals in species 1 should be aligned to squares in species 2, since the square-with-diagonal consisting of nodes $d$ , $e$ , $f$ , and $g$ in species 1 performs the same function (“orange”) as the square consisting of nodes 4, 5, 6, and 7 in species 2. Using these learned patterns, the method will try to align the rest of the nodes between the networks (not shown in the figure), transferring the functions of 3-node paths to square-with-diagonals, and of squares-with-diagonals to squares. In essence, noisy data or evolutionary events can be captured by topological relatedness but not topological similarity. . . . .	13
1.5	Illustration of a two-level biological NoN. Level 2 nodes (proteins) in <b>(a)</b> the level 2 network (PPI network) are joined to their corresponding <b>(b)</b> level 1 networks (PSNs) by dotted lines. Only three level 1 networks are shown for simplicity, but generally every level 2 node can have a corresponding level 1 network. Nodes in the PSNs are colored based on their corresponding amino acids in the ribbon diagram and are not indicative of node labels. . . . .	15
2.1	Overview of our PNA versus MNA evaluation framework. . . . .	23
2.2	Illustration of different alignment types. . . . .	25
2.3	Illustration on a set of three networks ( $G_1$ , $G_2$ , and $G_3$ ) of how we convert: <b>(a)</b> a multiple alignment to pairwise alignments, <b>(b)</b> one-to-one pairwise alignments to a multiple alignment, and <b>(c)</b> many-to-many pairwise alignments to a multiple alignment. . . . .	35

2.4	Comparison of the quality of T alignments versus the corresponding T+S alignments, under each of the PE and ME frameworks. Each bar shows the number of cases (here, a case refers to a combination of NA method, a network pair/set, and an alignment quality measure) in which the T alignment is superior, the T+S alignment is superior, or the two alignments are tied (i.e., within 1% of each other's accuracy). The cases are separated into network pairs/sets with known true node mapping and network pairs/sets with unknown true node mapping. . . . .	38
2.5	Alignment category comparison results for each of the <b>PE</b> and <b>ME</b> frameworks over all evaluation tests for T+S alignments. The alignment categories (i.e., PE-P-P, etc.) are color-coded. <b>View I.</b> Overall ranking of the NA methods. The "Overall rank" column shows the rank of each method averaged over all evaluation tests, along with the corresponding standard deviation (in brackets). <b>View II.</b> Alternative view of ranking of the NA methods. Each pie chart shows the fraction of evaluation test ranks that fall into the 1–4, 5–8, and 9–12 rank bins out of all evaluation test ranks in the given alignment category. The pie charts are color-coded with respect to alignments of network pairs/sets with known and unknown node mapping, and TQ and FQ measures. <b>View III.</b> Overall ranking of an NA method versus its running time for the Y2H <sub>1</sub> network set. The size of each point visualizes the overall ranking of the corresponding method over all evaluation tests, corresponding to the "Overall rank" column in View I; the larger the point size, the better the method. . . . .	40
2.6	Comparison of protein function prediction accuracy between the new (approach 3) versus existing (approach 2) prediction approach for multiple alignments. Each bar on the left of the figure shows the number of cases (i.e., alignments) in which the new approach is superior, the existing approach is superior, or the two approaches are tied. Each table shows the precision, recall, and number of predictions averaged over all tests. In parentheses, we show standard deviations. The results are separated into network sets with known and unknown node mapping. . . . .	50
2.7	Comparison of protein function prediction accuracy under the the PE and ME frameworks. The figure can be interpreted the same way as Fig. 2.6. Here, we use new approach 3 for the ME framework. . . . .	52

3.1 Illustration of (a) node-colored and (b) edge-colored graphlets. (a) With the exhaustive approach for enumerating all possible heterogeneous graphlets corresponding to homogeneous graphlet  $G_1$ , i.e., a 3-node path, given two colors, there would be six heterogeneous graphlets, each accounting for both which colors are present in the graphlet and which node position has which color. On the other hand, with our approach, there are three possible colored graphlets, denoted by  $\{c_{n_1}\}$ ,  $\{c_{n_2}\}$ , and  $\{c_{n_1}, c_{n_2}\}$ , each accounting only for which colors are present in the graphlet, ignoring the node-specific color information. Consequently, with our approach, the last four graphlets on the right of the arrow, which all have the same two colors present in them, are treated as the same heterogeneous graphlet. We design our approach in this way primarily to reduce the time complexity of counting heterogeneous graphlets in a network (but consequently, we also reduce the space complexity compared to the exhaustive approach). Namely, with our approach, the computational time complexity of searching for a given colored graphlet in a heterogeneous network remains the same as that of searching for its homogeneous equivalent. This is because the former involves: 1) counting in the heterogeneous network all graphlets, independent of their colors (which is the same as counting homogeneous graphlets in the network), and 2) for each of the homogeneous graphlets found in the network, simply determining which node colors appear in it and thus which node-colored graphlet the non-colored graphlet corresponds to. Step 1 is the time consuming part of the node-colored graphlet counting process, unlike step 2, which is trivial (can be done in constant time). (b) We develop a similar approach for edge-colored graphlets. . . . . 59

3.2 Illustration of HomEC and HetEC for an alignment between networks  $G$  and  $H$ . Arrows represent one possible alignment (mapping) between the networks, i.e., their nodes. Note that this node mapping is not the best alignment possible with respect to HomEC, but we use it to illustrate the concepts involved. In the homogeneous case (i.e., if all nodes were of the same color), there exist four conserved edges: the one formed by  $(a, a)$  and  $(a', a')$  – because  $a$  is aligned to  $a'$ ,  $b$  is aligned to  $b'$ , and an edge exists both between  $a$  and  $b$  as well as between  $a'$  and  $b'$ ; the one formed by  $(a, c)$  and  $(a', c')$ ; the one formed by  $(c, d)$  and  $(c', d')$ ; and the one formed by  $(b, d)$  and  $(b', d')$ . On the other hand,  $(a, d)$  and  $(a', d')$  form a non-conserved edge, because while  $a$  is aligned to  $a'$  and  $d$  is aligned to  $d'$ , there is an edge between  $a$  and  $d$  but not between  $a'$  and  $d'$ . For a similar reason,  $(b, c)$  and  $(b', c')$  form another non-conserved edge. So, given the existence of four conserved edges and two non-conserved edges, homogeneous  $S^3$  is 
$$\frac{\# \text{ conserved edges}}{(\# \text{ conserved edges} + \# \text{ non-conserved edges})} = 4/(4 + 2) = 0.67.$$
 In the heterogeneous case, for an edge to be conserved, the homogeneous condition is still required. However, we also account for colors of the aligned end nodes of a conserved edge and penalize for color mismatches. Specifically,  $(a, b)$  and  $(a', b')$  are counted as a fully conserved edge (with conservation weight of 1), because in addition to the fact that this edge is conserved in the homogeneous case,  $a$  has the same color as  $a'$ , and  $b$  has the same color as  $b'$ .  $(a, c)$  and  $(a', c')$  are counted as a less conserved edge (with conservation weight of  $\frac{2}{3}$ ), because while  $a$  and  $a'$  have the same color,  $c$  and  $c'$  do not. Similarly,  $(b, d)$  and  $(b', d')$  form a partly conserved edge with conservation weight of  $\frac{2}{3}$ .  $(c, d)$  and  $(c', d')$  are counted as an even less conserved edge (with conservation weight of  $\frac{1}{3}$ ) because neither  $c$  and  $c'$  nor  $d$  and  $d'$  have the same color. Finally,  $(a, d)$  and  $(a', d')$  form a non-conserved edge, just as in the homogeneous case. Given the total edge conservation of  $1 + \frac{2}{3} + \frac{2}{3} + \frac{1}{3} = \frac{8}{3}$  and two non-conserved edges (the same ones as in the homogeneous case), heterogeneous  $S^3$  uses the same formula as  $S^3$  and is  $\frac{8/3}{(\frac{8}{3} + 2)} = 0.57.$  61



- 3.3 Summarized results regarding the effect of the **number of considered node colors** on alignment quality for (a) synthetic networks, (b) PPI networks, and (c) protein-GO networks. In panels (a) and (b), there are up to four considered node colors, while in panel (c), there are up to two considered node colors (see Section 3.2.1 for details). For each case (see below), we compare the different color levels (i.e., numbers of considered colors shown on  $x$ -axes) and rank them from the best (rank 1) to the worst (rank 4 in panels 1 and b, and rank 2 in panel c). Then, we compute the percentage or frequency of all cases (see below) in which the given color level is ranked as the first (rank 1), second (rank 2), third (rank 3), or fourth (rank 4) best among all considered color levels. In panel (a), there are 3 methods (WAVE, MAGNA++, SANA)  $\times$  2 networks (geometric, scale-free)  $\times$  5 noise levels (0%, 10%, 25%, 50%, 75%) = 30 cases. In panel (b), there are 2 methods (WAVE, SANA)  $\times$  4 networks (APMS-Expr, APMS-Seq, Y2H-Expr, Y2H-Seq)  $\times$  5 noise levels (0%, 10%, 25%, 50%, 75%) = 40 cases. In panel (c), there are 2 methods (WAVE, SANA)  $\times$  2 networks (protein-GO-APMS, protein-GO-Y2H)  $\times$  5 noise levels (0%, 10%, 25%, 50%, 75%) = 20 cases. Note that we analyzed an additional noise level (100%), but we leave the corresponding results from this summary figure, because at this level all cases are expected to result in the same (random) alignments (Section 3.2.1 – Creating noisy counterparts of a synthetic, PPI, or protein-GO network). Instead, we show the results for the noise level of 100% in the detailed figures (Figs. 3.4, 3.5, 3.6). Also, note that in this figure, for each case, we choose the best between HetNC-HomEC and HetNC-HetEC. . . . . 70
- 3.4 Detailed alignment quality results regarding the effect of the **number of node colors** on alignment quality as a function of noise level for **synthetic, specifically geometric**, networks, using (a) WAVE, (b) MAGNA++, and (c) SANA. Gray squares, light blue circles, dark blue triangles, and black stars indicate the aligned networks containing one, two, three, and four node colors, respectively. For two or more node colors, solid lines represent using HetNC-HomEC, and dashed lines represent using HetNC-HetEC. Equivalent results for the remaining synthetic, specifically scale-free, networks are shown in Supplementary Fig. B.2. . . . . 71
- 3.5 Detailed alignment quality results regarding the effect of the **number of node colors** on alignment quality as a function of noise level for **PPI, specifically APMS-Expr**, networks using (a) WAVE and (b) SANA. The figure can be interpreted in the same way as Fig. 3.4. Recall that for these larger networks, we have not run MAGNA++ due to its high computational complexity. Equivalent results for the remaining PPI, specifically APMS-Seq, Y2H-Expr, and Y2H-Seq, networks are shown in Supplementary Figs. B.4, B.5, and B.6. . . . . 71

3.6	Detailed alignment quality results regarding the effect of the <b>number of node colors</b> on alignment quality as a function of noise level for <b>protein-GO, specifically protein-GO-APMS</b> , networks using (a) WAVE and (b) SANA. The figure can be interpreted in the same way as Fig. 3.4. Recall that for these larger networks, we have not run MAGNA++ due to its high computational complexity. Equivalent results for the remaining protein-GO, specifically protein-GO-Y2H, networks are shown in Supplementary Fig. B.8. . . . .	72
3.7	Summarized results regarding the effect of using HetEC over HomEC (both with HetNC) on alignment quality for (a) synthetic networks, (b) PPI networks, and (c) protein-GO networks. In all panels, there are two evaluation scenarios (HetNC-HomEC and HetNC-HetEC). For each case (see below), we compare the two considered evaluation scenarios and rank them from the best (rank 1) to the worst (rank 2). Then, we compute the percentage or frequency of all cases (see below) in which the given scenario is ranked as the first (rank 1) and second (rank 2) best among the considered scenarios. In panel (a), there are 2 methods (MAGNA++, SANA) $\times$ 2 networks (geometric, scale-free) $\times$ 5 noise levels (0, 10, 25, 50, 75) $\times$ 3 colors (1 color does not have a HetEC counterpart) = 60 cases. In panel (b), there is 1 method (SANA) $\times$ 4 networks (APMS-Expr, APMS-Seq, Y2H-Expr, Y2H-Seq) $\times$ 5 noise levels (as before) $\times$ 3 colors (as before) = 60 cases. In panel (c), there is 1 method (SANA) $\times$ 2 networks (protein-GO-APMS, protein-GO-Y2H) $\times$ 5 noise levels (as before) $\times$ 1 color (maximum 2 colors, but 1 color does not have a HetEC counterpart) = 10 cases. Note that we analyzed an additional noise level (100%), but we leave the corresponding results from this summary figure, because at this level all cases are expected to result in the same (random) alignments (Section 3.2.1 – Creating noise counterparts of a synthetic, PPI, or protein-GO network). Instead, we show the results for the noise level of 100% in the detailed figures (Figs. 3.4, 3.5, 3.6). . . . .	73
3.8	Detailed alignment quality results regarding the effect of <b>HomNC-HetEC</b> compared to HomNC-HomEC, HetNC-HomEC, and HetNC-HetEC on alignment quality for the two considered case study evaluation tests: (a) geometric networks using MAGNA++ and (b) APMS-Expr networks using SANA. The figure can be interpreted in the same way as Fig. 3.4, except that now solid lines represent HetNC-HomEC, short-long dotted lines represent HomNC-HetEC, and finely dotted lines represent HetNC-HetEC. . . . .	74

3.9 Summarized results regarding the effect of the **alignment method** on alignment quality for (a) synthetic networks, (b) PPI networks, and (c) protein-GO networks. In panel (a), there are three considered alignment methods (WAVE, MAGNA++, and SANA). In panels (b) and (c), there are two considered alignment methods (WAVE and SANA; MAGNA++ was not tested because of its high computational complexity). For each case (see below), we compare the alignment methods and rank the different methods from best (rank 1) to worst (rank 3 in panel (a), and rank 2 in panels (b) and (c)). Then, we compute the percentage of all cases in which the given method is ranked as the first (rank 1), second (rank 2), or third (rank 3) best among all considered methods. In panel (a), there are 2 networks (geometric, scale-free)  $\times$  5 noise levels (0, 10, 25, 50, 75) = 10 cases. In panel (b), there are 4 networks (APMS-Expr, APMS-Seq, Y2H-Expr, Y2H-Seq)  $\times$  5 noise levels (as above) = 20 cases. In panel (c), there are 2 networks (protein-GO-APMS, protein-GO-Y2H)  $\times$  5 noise levels (as above) = 10 cases. Note that we analyzed an additional noise level (100%), but we leave the corresponding results from this summary figure, because at this level all cases are expected to result in the same (random) alignments (Section 3.2.1 – Creating noise counterparts of a synthetic, PPI, or protein-GO network). Instead, we show the results for the noise level of 100% in the detailed figures (Figs. 3.4, 3.5, 3.6). Also, note that in this figure, we give each method the best case advantage. That is, we show results for the best of HetNC-HomEC and HetNC-HetEC, and also only for the maximum node color level (four colors in panels (a) and (b), and two colors in panel (c)). We do the latter because of all color levels, it is the maximum color level at which the given method performs the best, for each method. Nonetheless, the results remain qualitatively the same if we account for all considered colored levels. . . . .

76

3.10 Summarized results comparing the **running times versus accuracy** of different methods for 25% and 50% noise on (a) synthetic, specifically geometric and scale-free, (b) PPI, specifically APMS-Expr and APMS-Seq, and (c) protein-GO, specifically APMS and Y2H, networks. The  $x$ -axis is the running time of the given method on the given network data at the given noise level, and the  $y$ -axis is the alignment quality score. Here we use different shapes to represent the different methods, different colors to represent the different noise levels, and solid or broken lines to represent the different network data. Lines are drawn between the different methods for the same noise level and network data, for easier comparison of the different methods. Detailed running time results for all other noise levels and network data are shown in Supplementary Figs. B.9–B.16. . . . .

77

4.1	Distribution of topological similarity (GDV similarity) between node pairs of a geometric random graph (i.e., a synthetic network) and its <b>(a)</b> 0% and <b>(b)</b> 25% randomly perturbed counterparts. We show three lines representing the distribution of topological similarity for matching (i.e., functionally related) node pairs (blue), for non-matching, i.e., functionally unrelated, node pairs (red), and for 10 random samples of the same size as the set of matching pairs, averaged (purple). Results are qualitatively similar for 50% random perturbation, scale-free random graphs (a different type of synthetic networks), and GHOST's and IsoRank's similarity measures. (Supplementary Figs. C.1-C.3).	104
4.2	Distribution of topological similarity (GDV similarity) versus sequence similarity (E-value) between yeast and human PPI networks of those yeast-human protein pairs that are <b>(a)</b> functionally related (i.e., share at least one biological process GO term such that the protein-GO term annotation was experimentally inferred) and <b>(b)</b> functionally unrelated (i.e., share zero GO terms). The color of a pixel represents how many node pairs have a given topological similarity and given sequence similarity. The red horizontal and vertical lines indicate the thresholds for topologically similar ( $y \geq 0.85$ ) or sequence similar ( $x \leq 10^{-10}$ ) pairs, and the percentages indicate the fraction of pairs that are in a given quadrant.	106
4.3	Average prediction accuracy of <b>(a)</b> 10-fold cross-validation and <b>(b)</b> percent training tests for a geometric network and its randomly perturbed counterparts. In panel <b>(b)</b> , different colored lines represent how much data is used for training; these colors do not apply to panel <b>(a)</b> . A dotted black line indicates the accuracy expected if the classifier makes random predictions. Qualitatively similar results for AUROC and for scale-free networks are shown in Supplementary Figs. C.4-C.5.	107
4.4	Average prediction accuracy of <b>(a)</b> 10-fold cross-validation and <b>(b)</b> percent training tests for real-world networks. In panel <b>(b)</b> , different colored lines represent how much data is used for training; these colors do not apply to panel <b>(a)</b> . A dotted black line indicates the accuracy expected if the classifier makes random predictions. Qualitatively similar results for AUROC are shown in Supplementary Figs. C.6-C.7.	109

4.5	Comparison of different TARA evaluation tests in the task of protein function prediction, for the ALL GO term rarity threshold. Different percent training tests, specifically 10, 50, and 90, are compared within each panel, and different ground truth datasets, specifically <b>(a)</b> atleast1-EXP, <b>(b)</b> atleast2-EXP, and <b>(c)</b> atleast3-EXP, are compared across panels. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method, averaged over the 10 instances we perform for each test, are shown on the top. For example, the alignment for TARA-90 for the atleast2-EXP dataset contains 1,327 aligned yeast-human protein pairs, and predicts 5,657 protein-GO term associations. Raw precision, recall, and F-score values are color-coded inside each panel. Complete results for the other rarity thresholds are shown in Supplementary Fig. C.8. . . . .	110
4.6	Comparison of the six considered NA methods for rarity thresholds <b>(a, d)</b> ALL, <b>(b, e)</b> 50, and <b>(c, f)</b> 25 using ground truth datasets <b>(a, b, c)</b> atleast1-EXP and <b>(d, e, f)</b> atleast2-EXP in the task of protein function prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method. For example, the alignment for TARA in panel <b>(a)</b> contains 27,155 aligned yeast-human protein pairs, and predicts 91,618 protein-GO term associations. Raw precision, recall, and F-score values are color-coded inside each panel. Results for atleast3-EXP are shown in Supplementary Fig. C.9. . . . .	115
4.7	Overlap of the functional predictions made by TARA and PrimAlign for atleast2-EXP at the 50 rarity threshold. Percentages are out of the total number of unique predictions made by both methods combined. Complete results for all methods and parameters are shown in Supplementary Fig. C.10 and Supplementary File C.1. . . . .	117
4.8	Distribution of TARA's redefined topological relatedness between node pairs of a geometric random graph (i.e., a synthetic network) and its (a) 0% and (b) 25% randomly perturbed counterparts. We show three lines representing the distribution of topological relatedness for matching (i.e., functionally related) node pairs (blue), for non-matching, i.e., functionally unrelated node pairs (red), and for 10 random samples of the same size as the set of matching pairs, averaged (purple). . . . .	120

4.9	Comparison of TARA on the 2017 versus 2020 networks for rarity thresholds <b>(a, d)</b> ALL, <b>(b, e)</b> 50, and <b>(c, f)</b> 25 using ground truth datasets <b>(a, b, c)</b> atleast1-EXP and <b>(d, e, f)</b> atleast2-EXP in the task of protein function prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method. For example, the alignment for TARA-2017 in panel <b>(a)</b> contains 27,155 aligned yeast-human protein pairs, and predicts 91,618 protein-GO term associations. Raw precision, recall, and F-score values are color-coded inside each panel. Results for atleast3-EXP are shown in Supplementary Fig. C.11. . . . .	124
4.10	Summary of TARA-TS and our evaluation framework. <b>(a)</b> TARA-TS aims to align two networks (in this study, yeast and human PPI networks). Besides the networks, TARA-TS also uses sequence similar yeast-human protein pairs as anchor links. See Section “4.2.2 – Data”. <b>(b)</b> From the networks and anchor links, TARA-TS builds an integrated yeast-human network and extracts integrated topology- and sequence-based features of node (protein) pairs. See Section “4.2.2 – TARA-TS’s feature extraction methodology”. <b>(c)</b> Given the features, TARA-TS trains a classifier on a training set to learn what features distinguish between functionally related and functionally unrelated node pairs, and then the classifier is evaluated on a testing set. To perform this classification, yeast-human node pairs are labeled. If the two nodes in a given pair are functionally related (intuitively, share GO terms), they are labeled with the positive class; if they are functionally unrelated, they are labeled with the negative class. See Section “4.2.2 – Data”. Then, the set of labeled node pairs is split into training and testing sets to perform the classification. Only if classification accuracy is high, i.e., if TARA-TS accurately predicts functionally (un)related nodes to be functionally (un)related, does it make sense to use TARA-TS to create an alignment for protein functional prediction. <b>(d)</b> Node pairs from the testing set that are predicted as functionally related are taken as TARA-TS’s alignment. Note that relying on testing data only to create an alignment avoids any circular argument. See Section “4.2.2 – TARA-TS’s classification and alignment generation”. <b>(e)</b> Any alignment, of TARA-TS or an existing NA method such as PrimAlign and TARA, can be given to a protein functional prediction framework to predict protein-GO term annotations. Then, the different methods’ alignments are evaluated in terms of their prediction accuracy (we also evaluate their running times). See Section “4.2.2 – Using an alignment for protein functional prediction”. . . . .	129

- 4.11 Comparison of the three TARA-TS versions and TARA. Comparison of the three TARA-TS versions and TARA for GO term rarity threshold 25 and ground truth dataset atleast1-EXP, in terms of: **(a)** classification accuracy, **(b)** protein functional prediction accuracy, **(c)** overlap between aligned yeast-human protein pairs, and **(d)** overlap between predicted protein-GO term associations. In panel (b), the alignment for e.g., TARA contains 1,716 aligned protein pairs and predicts 3,474 protein-GO term associations. In panels (c)-(d), the pairwise overlaps are measured via the Jaccard index. Panel (a) encompasses all  $y$  percent training tests. Panels (b)-(d) are for the 90% training test. Comparisons of different metapath choices for metapath2vec can be found in Supplementary Fig. C.12. Results for the other ground-truth rarity datasets and percent training tests are shown in Supplementary Figs. C.13–C.19. . . . . 143
- 4.12 Comparison of TARA-TS and TARA in terms of their alignment and prediction overlaps. Comparison of the selected TARA-TS version and TARA for GO term rarity threshold 50, ground truth dataset atleast1-EXP, and the 90% training test, in terms of overlap between their: **(a)** aligned yeast-human protein pairs and **(b)** predicted protein-GO term associations. In panel (b), precision and recall are shown for each of the three prediction sets captured by the Venn diagram; TARA++’s predictions are those in the overlap. The overlaps are for one of the 10 balanced datasets; so, the alignment size and prediction number of a method may differ from those in Fig. 4.11(b), where the statistics are averaged over all balanced datasets. Results for the other ground truth-rarity datasets are shown in Supplementary Figs. C.20–C.21. . 147
- 4.13 Comparison of TARA++ and three existing methods in the task of protein functional prediction. Comparison of TARA++ and three existing methods in the task of protein functional prediction, for rarity thresholds **(a)** 50 and **(b, c)** 25, and for ground truth datasets **(a, b)** atleast1-EXP and **(c)** atleast2-EXP. The alignment size (the number of aligned yeast-protein pairs) and number of functional predictions (predicted protein-GO term associations) are shown for each method, except that TARA++ does not have an alignment *per se.* i.e., TARA++ comes from the overlap of *predictions* made by TARA and TARA-TS; hence the “N/A”s. For TARA++ and TARA, results are averages over all balanced datasets; the standard deviations are small and thus invisible. Results for the other ground truth-rarity datasets are shown in Supplementary Fig. C.22. . . . . 149

4.14	Comparison of TARA++ and three existing methods when all make the same number of predictions. Representative results (for one ground truth-rarity dataset) comparing TARA++ and three existing methods in the same way as in Fig. 4.13(a) except that here all methods make the same number of predictions. The remaining results (for the other ground truth-rarity datasets) are shown in Supplementary Fig. C.23.	150
4.15	Comparison of TARA++ and PrimAlign in terms of their prediction overlaps. Representative results (for GO term rarity threshold 50 and ground truth dataset atleast1-EXP) comparing TARA++ and PrimAlign in the same way as TARA and TARA-TS are compared in Fig. 4.12(b). The remaining results (for the other ground truth-rarity datasets) are shown in Supplementary Fig. C.24.	151
4.16	Robustness of TARA++ to data noise. Robustness of TARA++ protein functional prediction accuracy as data noise increases from 0% to 100%, for GO term rarity threshold 25 and ground truth dataset atleast2-EXP.	153
5.1	A toy synthetic NoN generated from two random graph models. Large dotted circles represent level 2 node groups (originating from isolated NoNs) whose level 2 nodes are connected in a random geometric-(GEO) or scale-free-like (SF) fashion. Small solid circles represent level 2 nodes whose level 1 networks are of the random graph type indicated. Level 1 nodes and edges are not shown. Level 2 nodes are colored based on their label, i.e., their combination of level 1 and level 2 network topology ( $\{(GEO, GEO), (GEO, SF), (SF, GEO), \text{ and } (SF, SF)\}$ ).	166
5.2	Comparison of the nine considered approaches in the task of label prediction for synthetic NoNs with the following parameters: <b>(a)</b> 5% across-edge and 0% rewiring-noise amount, <b>(b)</b> 5% across-edge and 75% rewiring-noise amount, <b>(c)</b> 95% across-edge and 0% rewiring-noise amount, and <b>(d)</b> 95% across-edge and 75% rewiring-noise amount. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars. Standard deviations are indicated at the top of each bar; some have very small values and are thus not visible. We expect an approach that only uses a single level and does not capture clustering information to have around $\frac{\# \text{ of models}}{\# \text{ of labels}}$ , or 0.5, accuracy when both across-edge and rewiring-noise amount are low (Supplementary Section D.2.1). Results for other parameter combinations are shown in Supplementary Figs. D.2-D.6.	175



5.3	Summarized results of the eight considered approaches (as GCN-3 is not used for the biological NoN) in the task of protein functional prediction in terms of AUPR. For each GO term (out of the 131 total), we rank the eight approaches' from best (rank 1) to worst (rank 8). Then, we calculate the proportion of GO terms each approach achieves each rank. "Combined all" refers to L1 GDVM + L2 GDV + L1 Diff-Pool + L2 SIGN. Results for other evaluation measures are shown in Supplementary Fig. D.7 . . . . .	177
A.1	Clustering of NA methods, each with its T and T+S versions, using each of the <b>PE</b> and <b>ME</b> frameworks. Clustering is based on pairwise method similarities, which we compute as follows. The similarity between two NA methods is the mean of the Adjusted Rand Index (ARI; explained below) of each pair of corresponding alignments produced by the two NA methods, over all network pairs/sets. Each alignment of a network pair/set is a set of node groups, i.e., a partition of the nodes in all of the networks in the network pair/set, and we measure similarity between two alignments by comparing their partitions using ARI. ARI [167] is a widely used measure to calculate the similarity between two partitions. Given the similarities between all pairs of the NA methods, we cluster using complete linkage hierarchical clustering [50] and visualize the clustering using a dendrogram. The results shown in this figure rely on all alignments over all network sets (Yeast+%LC, PHY <sub>1</sub> , PHY <sub>2</sub> , Y2H <sub>1</sub> , and Y2H <sub>2</sub> ). Equivalent results broken down into results for networks with known node mapping and results for networks with unknown node mapping are shown in Supplementary Figs. A.2 and A.3, respectively. . . . .	227
A.2	Clustering of NA methods, each with its T and T+S versions, using all network sets with (a) <b>known node mapping</b> and (b) <b>unknown node mapping</b> in the <b>PE framework</b> . The figure can be interpreted the same way as Supplementary Fig. A.1. . . . .	228
A.3	Clustering of NA methods, each with its T and T+S versions, using all network sets with (a) <b>known node mapping</b> and (b) <b>unknown node mapping</b> in the <b>ME framework</b> . The figure can be interpreted the same way as Supplementary Fig. A.1. . . . .	228

- A.4 Overall ranking of an NA method versus its running time for the **PE framework** over all evaluation tests (where a test is a combination of an NA method, a network pair, and an alignment quality measure). By NA method, here, we mean the combination of a PNA or MNA method and the alignment category (Chapter 2.2.4 of the main document). Namely, there are 12 NA methods in the PE framework (four PNA methods associated with the PE-P-P categories and four MNA methods associated with each of the PE-M-M and PE-M-P categories). The running time results are when aligning all network pairs in the Y2H<sub>1</sub> network set, where each method is restricted to use a **single core**. The size of each point visualizes the overall ranking of the corresponding method over all evaluation tests over all network pairs/sets, corresponding to the “Overall rank” column in View I of Fig. 2.5 in the main document; the larger the point size, the better the method. In order to allow for easier comparison between the different alignment categories, “Average” shows the average running times and average rankings of the methods in each alignment category. . . . 229
- A.5 Overall ranking of an NA method versus its running time for the **ME framework** over all evaluation tests (where a test is a combination of an NA method, a network pair, and an alignment quality measure). By NA method, here, we mean the combination of a PNA or MNA method and the alignment category (Chapter 2.2.4 of the main document). Namely, there are 12 NA methods in the ME framework (four PNA methods associated with the ME-P-P categories and four MNA methods associated with each of the ME-M-M and ME-M-P categories). The running time results are when aligning the Y2H<sub>1</sub> network set, where each method is restricted to use a **single core**. The size of each point visualizes the overall ranking of the corresponding method over all evaluation tests over all network pairs/sets, corresponding to the “Overall rank” column in View I of Fig. 2.5 in the main document; the larger the point size, the better the method. In order to allow for easier comparison between the different alignment categories, “Average” shows the average running times and average rankings of the methods in each alignment category. . . . . 230

A.6 Method comparison results for each of the **PE** and **ME** frameworks over all evaluation tests (where a test is a combination of an NA method, a network pair/set, and an alignment quality measure), for T alignments. By NA method, here, we mean the combination of a PNA or MNA method and the alignment category (Chapter 2.2.4 of the main document). Namely, there are 12 NA methods in the PE framework (four PNA methods associated with the PE-P-P categories and four MNA methods associated with each of the PE-M-M and PE-M-P categories) and 12 NA methods in the ME framework (four PNA methods associated with the ME-P-P categories and four MNA methods associated with each of the ME-M-M and ME-M-P categories). The alignment categories are color coded. **View I.** Overall ranking of the NA methods. The “Overall rank” column shows the rank of each method averaged over all evaluation tests, along with the corresponding standard deviation (in brackets). Since there are 12 methods in a given framework, the possible ranks range from 1 to 12. The lower the rank, the better the given method. The “ $p_1$ -value” column shows the statistical significance of the difference between the ranking of each method and the 1<sup>st</sup> best ranked method. The “ $p_2$ -value” column shows the statistical significance of the difference between the ranking of each method and the 2<sup>nd</sup> best ranked method. The “Non. sig. (fail)” column shows the fraction of evaluation tests in which the alignment quality score is not statistically significant, and, in brackets, the fraction of evaluation tests in which the given NA method failed to produce an alignment. Equivalent results over all evaluation tests broken down into functional and topological alignment quality measures, as well as over all evaluation tests broken down into network pairs/sets with known and unknown node mapping, are shown in Supplementary Tables A.4–A.11. **View II.** Alternative view of ranking of the NA methods. Each pie chart shows the fraction of evaluation test ranks that fall into the 1–4, 5–8, and 9–12 rank bins out of all evaluation test ranks in the given alignment category. For example, for the PE framework, in the PE-P-P alignment category, 56%, 26%, and 18% of the evaluation test ranks fall into ranks 1–4, 5–8, and 9–12, respectively, totaling to 100% of the evaluation test ranks in the PE-P-P alignment category. The pie charts allow us to compare the three alignment categories rather than individual NA methods in each category. The larger the pie chart for the better (lower) ranks, and the smaller the pie chart for the worse (higher) ranks, the better the alignment category. For example, in the PE framework, PE-P-P has the most evaluation tests ranked 1–4 and the fewest evaluation tests ranked 9–12, followed by PE-M-P, followed by PE-M-M. This implies that PE-P-P is superior to PE-M-P and PE-M-M. The pie charts are color coded with respect to alignments of network pairs/sets with known and unknown node mapping, and FQ and TQ measures. . . . . 231

A.7	Comparison of protein function prediction accuracy between the <b>new</b> (approach 3) versus the <b>existing</b> prediction approach for multiple alignments (approach 2), for all alignments from the ME framework (i.e., ME-P-P, ME-M-P, and ME-M-M categories). We calculate the prediction accuracy as described in Fig. 2.6 in the main document. Each column shows the precision and recall achieved by the new or existing prediction approach for each NA method, as well as the number of predictions made by the approach. The alignments are separated into networks sets with known and unknown mapping. . . . .	234
A.8	Comparison of protein function prediction accuracy under the <b>PE framework</b> (i.e., PE-P-P, PE-M-P, and PE-M-M categories) and <b>ME framework</b> (i.e., ME-P-P, ME-M-P, and ME-M-M categories). We calculate the prediction accuracy as described in Fig. 2.6 in the main document. Each column shows the precision and recall achieved by the new or existing prediction approach for each NA method, as well as the number of predictions made by the approach. The alignments are separated into networks sets with known and unknown mapping. .	235
A.9	Illustration of the effect of the choice of scaffold network on alignment quality when combining pairwise alignments into a multiple alignment. These are representative results for one of the analyzed TQ measures (NCV-CIQ; panel (a)), one of the analyzed FQ measures (GO correctness – GC; panel (b)), one of the analyzed network sets (Y2H1), and one of the analyzed NA methods (WAVE). Clearly, different choices of scaffold network ( $x$ -axis) yield different alignment quality scores ( $y$ -axis). The same holds for other combinations of alignment quality measures, network sets, and NA methods. In our evaluation, of all scaffold network choices, the one that yields the best multiple alignment is chosen. In this particular representative scenario, it is the human network that was chosen as the scaffold, since this scaffold choice clearly yields significantly better alignment quality than any other scaffold choice. . . . .	236
A.10	Comparison of protein function prediction accuracy under the the PE and ME frameworks, where we use approach 2 for the ME framework (rather than using approach 3 for the ME framework like we do in Fig. 2.7 of the main document). The figure can be interpreted the same way as Fig. 2.6 in the main document. . . . .	237

B.1	Detailed alignment quality results regarding the effect of the <b>number of node colors</b> on alignment quality as a function of noise level for <b>synthetic, specifically geometric</b> , networks using (a) WAVE, (b) MAGNA++, and (c) SANA. Gray squares, light blue circles, dark blue triangles, and black stars indicate the aligned networks containing one, two, three, and four node colors, respectively. For two or more node colors, solid lines represent using HetNC-HomEC, and dashed lines represent using HetNC-HetEC. . . . .	239
B.2	Detailed alignment quality results regarding the effect of the <b>number of node colors</b> on alignment quality as a function of noise level for <b>synthetic, specifically scale-free</b> , networks using (a) WAVE, (b) MAGNA++, and (c) SANA. The figure can be interpreted in the same way as Supplementary Figure B.1. . . . .	240
B.3	Detailed alignment quality results regarding the effect of the <b>number of node colors</b> on alignment quality as a function of noise level for <b>PPI, specifically APMS-Expr</b> , networks using (a) WAVE and (b) SANA. The figure can be interpreted in the same way as Supplementary Figure B.1. Recall that for these larger networks, we have not run MAGNA++ due to its high computational complexity. . . . .	240
B.4	Detailed alignment quality results regarding the effect of the <b>number of node colors</b> on alignment quality as a function of noise level for <b>PPI, specifically APMS-Seq</b> , networks using (a) WAVE and (b) SANA. The figure can be interpreted in the same way as Supplementary Figure B.1. Recall that for these larger networks, we have not run MAGNA++ due to its high computational complexity. . . . .	241
B.5	Detailed alignment quality results regarding the effect of the <b>number of node colors</b> on alignment quality as a function of noise level for <b>PPI, specifically Y2H-Expr</b> , networks using (a) WAVE and (b) SANA. The figure can be interpreted in the same way as Supplementary Figure B.1. Recall that for these larger networks, we have not run MAGNA++ due to its high computational complexity. . . . .	241
B.6	Detailed alignment quality results regarding the effect of the <b>number of node colors</b> on alignment quality as a function of noise level for <b>PPI, specifically Y2H-Seq</b> , networks using (a) WAVE and (b) SANA. The figure can be interpreted in the same way as Supplementary Figure B.1. Recall that for these larger networks, we have not run MAGNA++ due to its high computational complexity. . . . .	242

B.7	Detailed alignment quality results regarding the effect of the <b>number of node colors</b> on alignment quality as a function of noise level for <b>protein-GO, specifically protein-GO-APMS</b> , networks using (a) WAVE and (b) SANA. The figure can be interpreted in the same way as Supplementary Figure B.1. Recall that for these larger networks, we have not run MAGNA++ due to its high computational complexity.	242
B.8	Detailed alignment quality results regarding the effect of the <b>number of node colors</b> on alignment quality as a function of noise level for <b>protein-GO, specifically protein-GO-Y2H</b> , networks using (a) WAVE and (b) SANA. The figure can be interpreted in the same way as Supplementary Figure B.1. Recall that for these larger networks, we have not run MAGNA++ due to its high computational complexity.	243
B.9	Detailed results comparing the <b>running time</b> and effect of the <b>number of node colors</b> for different methods for all tested noise levels on <b>synthetic, specifically geometric</b> , networks. The $x$ -axis the the running time of the method, and the $y$ -axis is the alignment quality. Here we use different shapes to represent the different methods and different colored lines to represent how many node colors are used. Lines are drawn between methods using the same number of colors.	244
B.10	Detailed results comparing the <b>running time</b> and effect of the <b>number of node colors</b> for different methods for all tested noise levels on <b>synthetic, specifically scale-free</b> , networks. The figure can be interpreted in the same way as Supplementary Figure B.9.	245
B.11	Detailed results comparing the <b>running time</b> and effect of the <b>number of node colors</b> for different methods for all tested noise levels on <b>PPI, specifically APMS-Expr</b> , networks. The figure can be interpreted in the same way as Supplementary Figure B.9.	246
B.12	Detailed results comparing the <b>running time</b> and effect of the <b>number of node colors</b> for different methods for all tested noise levels on <b>PPI, specifically APMS-Seq</b> , networks. The figure can be interpreted in the same way as Supplementary Figure B.9.	247
B.13	Detailed results comparing the <b>running time</b> and effect of the <b>number of node colors</b> for different methods for all tested noise levels on <b>PPI, specifically Y2H-Expr</b> , networks. The figure can be interpreted in the same way as Supplementary Figure B.9.	248
B.14	Detailed results comparing the <b>running time</b> and effect of the <b>number of node colors</b> for different methods for all tested noise levels on <b>PPI, specifically Y2H-Seq</b> , networks. The figure can be interpreted in the same way as Supplementary Figure B.9.	249

B.15	Detailed results comparing the <b>running time</b> and effect of the <b>number of node colors</b> for different methods for all tested noise levels on <b>protein-GO, specifically protein-GO-APMS</b> , networks. The figure can be interpreted in the same way as Supplementary Figure B.9.	250
B.16	Detailed results comparing the <b>running time</b> and effect of the <b>number of node colors</b> for different methods for all tested noise levels on <b>protein-GO, specifically protein-GO-Y2H</b> , networks. The figure can be interpreted in the same way as Supplementary Figure B.9.	251
C.1	Distribution of topological similarity (GDV-similarity) between node pairs of a <b>(a,b,c)</b> geometric and <b>(d,e,f)</b> scale-free network and their <b>(a,d)</b> 0%, <b>(b,e)</b> 25% noisy, and <b>(c,f)</b> 50% noisy counterparts. We show three lines representing the distribution of topological similarity for matching, i.e., functionally related, node pairs (blue), for non-matching, i.e., functionally unrelated, node pairs (red), and for 10 random samples of the same size as the set of matching pairs, averaged (purple).	252
C.2	Distribution of topological similarity (GHOST) between node pairs of a <b>(a,b,c)</b> geometric and <b>(d,e,f)</b> scale-free network and their <b>(a,d)</b> 0%, <b>(b,e)</b> 25% noisy, and <b>(c,f)</b> 50% noisy counterparts. We show three lines representing the distribution of topological similarity for matching, i.e., functionally related, node pairs (blue), for non-matching, i.e., functionally unrelated, node pairs (red), and for 10 random samples of the same size as the set of matching pairs, averaged (purple).	253
C.3	Distribution of topological similarity (IsoRank) between node pairs of a <b>(a,b,c)</b> geometric and <b>(d,e,f)</b> scale-free network and their <b>(a,d)</b> 0%, <b>(b,e)</b> 25% noisy, and <b>(c,f)</b> 50% noisy counterparts. We show three lines representing the distribution of topological similarity for matching, i.e., functionally related, node pairs (blue), for non-matching, i.e., functionally unrelated, node pairs (red), and for 10 random samples of the same size as the set of matching pairs, averaged (purple).	254
C.4	Average <b>(a,b)</b> prediction accuracy and <b>(c,d)</b> AUROC of 10-fold cross validation for <b>(a,c)</b> geometric and <b>(b,d)</b> scale-free networks.	255
C.5	Average <b>(a,b)</b> prediction accuracy and <b>(c,d)</b> AUROC of percent training tests for <b>(a,c)</b> geometric and <b>(b,d)</b> scale-free networks.	256
C.6	Average <b>(a)</b> prediction accuracy and <b>(b)</b> AUROC of 10-fold cross validation for real-world networks.	256
C.7	Average <b>(a)</b> prediction accuracy and <b>(b)</b> AUROC of percent training tests for real-world networks.	257

- C.8 Comparison of different TARA evaluation tests in the task of protein function prediction, for GO term rarity thresholds **(a, d, g)** ALL, **(b, e)** 50, and **(c, f)** 25 using ground truth datasets **(a, b, c)** atleast1-EXP, **(d, e, f)** atleast2-EXP, and **(g)** atleast3-EXP. Different percent training tests, specifically 10, 50, and 90, are compared within each panel. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method, averaged over the 10 instances we perform for each test, are shown on the top. For example, the alignment for TARA-90 in **(a)** contains 27,155 aligned yeast-human protein pairs, and predicts 91,618 protein-GO term associations. Raw precision, recall, and F-score values are color-coded inside each panel. 258
- C.9 Comparison of the six considered NA methods for rarity thresholds **(a, d, g)** ALL, **(b, e)** 50, and **(c, f)** 25 using ground truth datasets **(a, b, c)** atleast1-EXP, **(d, e, f)** atleast2-EXP, and **(g)** atleast3-EXP in the task of protein function prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method. For example, the alignment for TARA in **(a)** contains 27,155 aligned yeast-human protein pairs, and predicts 91,618 protein-GO term associations. Raw precision, recall, and F-score values are color-coded inside each panel. . . . . 260
- C.10 Overlap of the functional predictions made by TARA and PrimAlign for GO term rarity thresholds **(a, d, g)** ALL, **(b, e)** 50, and **(c, f)** 25 using ground truth datasets **(a, b, c)** atleast1-EXP, **(d, e, f)** atleast2-EXP, and **(g)** atleast3-EXP. Percentages are out of the total number of unique predictions made by both methods combined. . . . . 262
- C.11 Comparison of TARA on the 2017 versus 2020 networks for rarity thresholds **(a, d, g)** ALL, **(b, e)** 50, and **(c, f)** 25 using ground truth datasets **(a, b, c)** atleast1-EXP, **(d, e, f)** atleast2-EXP, and **(g)** atleast3-EXP in the task of protein function prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method. For example, the alignment for TARA-2017 in panel **(a)** contains 27,155 aligned yeast-human protein pairs, and predicts 91,618 protein-GO term associations. Raw precision, recall, and F-score values are color-coded inside each panel. . . . . 263



C.12	Comparison of different metapath choices for rarity thresholds <b>(a, d, g)</b> ALL, <b>(b, e)</b> 50, and <b>(c, f)</b> 25 using ground truth datasets <b>(a, b, c)</b> atleast1-EXP, <b>(d, e, f)</b> atleast2-EXP, and <b>(g)</b> atleast3-EXP in the task of protein functional prediction. “mp2v- <i>n</i> ” refers to the paths “human $\times$ n $\rightarrow$ yeast $\times$ n” and “yeast $\times$ n $\rightarrow$ human $\times$ n” (Chapter “4.2.2 – TARA-TS’s feature extraction methodology”). The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method are shown above. For example, the alignment for mp2v-3 in <b>(a)</b> contains 27,799 aligned yeast-human protein pairs, and predicts 88,130 protein-GO term associations. Raw precision and recall values are color-coded inside each panel. . . . .	272
C.13	Average prediction accuracy of percent training tests for rarity thresholds <b>(a, d, g)</b> ALL, <b>(b, e)</b> 50, and <b>(c, f)</b> 25 using ground truth datasets <b>(a, b, c)</b> atleast1-EXP, <b>(d, e, f)</b> atleast2-EXP, and <b>(g)</b> atleast3-EXP. A dotted black line indicates the accuracy expected if the classifier makes random predictions. Qualitatively similar results for AUROC are shown in Supplementary Figs. S2. . . . .	274
C.14	Average AUROC of percent training tests for rarity thresholds <b>(a, d, g)</b> ALL, <b>(b, e)</b> 50, and <b>(c, f)</b> 25 using ground truth datasets <b>(a, b, c)</b> atleast1-EXP, <b>(d, e, f)</b> atleast2-EXP, and <b>(g)</b> atleast3-EXP. A dotted black line indicates the AUROC expected if the classifier makes random predictions. . . . .	275
C.15	Comparison of TARA and TARA-TS using 10% of the data as training for rarity thresholds <b>(a, d, g)</b> ALL, <b>(b, e)</b> 50, and <b>(c, f)</b> 25 using ground truth datasets <b>(a, b, c)</b> atleast1-EXP, <b>(d, e, f)</b> atleast2-EXP, and <b>(g)</b> atleast3-EXP in the task of protein functional prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method are shown above. For example, the alignment for TARA-10 in <b>(a)</b> contains 244,433 aligned yeast-human protein pairs, and predicts 538,397 protein-GO term associations. Raw precision and recall values are color-coded inside each panel. . . . .	276

C.16	Comparison of TARA and TARA-TS using 50% of the data as training for rarity thresholds <b>(a, d, g)</b> ALL, <b>(b, e)</b> 50, and <b>(c, f)</b> 25 using ground truth datasets <b>(a, b, c)</b> atleast1-EXP, <b>(d, e, f)</b> atleast2-EXP, and <b>(g)</b> atleast3-EXP in the task of protein functional prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method are shown above. For example, the alignment for TARA-10 in <b>(a)</b> contains 244,433 aligned yeast-human protein pairs, and predicts 538,397 protein-GO term associations. Raw precision and recall values are color-coded inside each panel. . . . .	278
C.17	Comparison of TARA and TARA-TS using 90% of the data as training for rarity thresholds <b>(a, d, g)</b> ALL, <b>(b, e)</b> 50, and <b>(c, f)</b> 25 using ground truth datasets <b>(a, b, c)</b> atleast1-EXP, <b>(d, e, f)</b> atleast2-EXP, and <b>(g)</b> atleast3-EXP in the task of protein functional prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method are shown above. For example, the alignment for TARA-10 in <b>(a)</b> contains 244,433 aligned yeast-human protein pairs, and predicts 538,397 protein-GO term associations. Raw precision and recall values are color-coded inside each panel. . . . .	280
C.18	Pairwise overlap, measured by Jaccard index, of the alignments made by TARA and TARA-TS for rarity thresholds <b>(a, d, g)</b> ALL, <b>(b, e)</b> 50, and <b>(c, f)</b> 25 using ground truth datasets <b>(a, b, c)</b> atleast1-EXP, <b>(d, e, f)</b> atleast2-EXP, and <b>(g)</b> atleast3-EXP, using percent training amounts described in Chapter “4.2.3 – TARA-TS versus TARA in the task of protein functional prediction: toward TARA++”. . . . .	282
C.19	Pairwise overlap, measure by Jaccard index, of the predictions made by TARA and TARA-TS for rarity thresholds <b>(a, d, g)</b> ALL, <b>(b, e)</b> 50, and <b>(c, f)</b> 25 using ground truth datasets <b>(a, b, c)</b> atleast1-EXP, <b>(d, e, f)</b> atleast2-EXP, and <b>(g)</b> atleast3-EXP, using percent training amounts described in Chapter “4.2.3 – TARA-TS versus TARA in the task of protein functional prediction: toward TARA++”. . . . .	283
C.20	Overlap of the alignments made by TARA and TARA-TS for rarity thresholds <b>(a, d, g)</b> ALL, <b>(b, e)</b> 50, and <b>(c, f)</b> 25 using ground truth datasets <b>(a, b, c)</b> atleast1-EXP, <b>(d, e, f)</b> atleast2-EXP, and <b>(g)</b> atleast3-EXP. Percentages are out of the total number of unique aligned node pairs made by both methods combined. The overlaps are for one of the 10 balanced datasets; so, the alignment size of a method may differ from those in Supplementary Figs. S3-S5, where the statistics are averaged over all balanced datasets. . . . .	284

- C.21 Overlap of the predictions made by TARA and TARA-TS for rarity thresholds **(a, d, g)** ALL, **(b, e)** 50, and **(c, f)** 25 using ground truth datasets **(a, b, c)** atleast1-EXP, **(d, e, f)** atleast2-EXP, and **(g)** atleast3-EXP. Percentages are out of the total number of unique predictions made by both methods combined. Precision and recall are shown for each of the three prediction sets captured by the Venn diagram; TARA++’s predictions are those in the overlap. The overlaps are for one of the 10 balanced datasets; so, the prediction number of a method may differ from those in Supplementary Figs. S3-S5, where the statistics are averaged over all balanced datasets. . . . . 285
- C.22 Comparison of four NA methods for rarity thresholds **(a, d, g)** ALL, **(b, e)** 50, and **(c, f)** 25 using ground truth datasets **(a, b, c)** atleast1-EXP, **(d, e, f)** atleast2-EXP, and **(g)** atleast3-EXP in the task of protein functional prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method are shown above, except that TARA++ does not have an alignment *per se*. i.e., TARA++ comes from the overlap of *predictions* made by TARA and TARA-TS; hence the “N/A”s. For example, the alignment for TARA in **(a)** contains 27,155 aligned yeast-human protein pairs, and predicts 91,618 protein-GO term associations. Raw precision and recall values are color-coded inside each panel. For TARA++ and TARA, results are averages over all balanced datasets; the standard deviations are small and thus invisible. . . . . 288
- C.23 Comparison of four NA methods for rarity thresholds **(a, d, g)** ALL, **(b, e)** 50, and **(c, f)** 25 using ground truth datasets **(a, b, c)** atleast1-EXP, **(d, e, f)** atleast2-EXP, and **(g)** atleast3-EXP in the task of protein functional prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method are shown above, except that TARA++ does not have an alignment *per se*. i.e., TARA++ comes from the overlap of *predictions* made by TARA and TARA-TS; hence the “N/A”s. For example, the alignment for TARA in **(a)** contains 27,155 aligned yeast-human protein pairs, and predicts 91,618 protein-GO term associations. Raw precision and recall values are color-coded inside each panel. For TARA++ and TARA, results are averages over all balanced datasets; the standard deviations are small and thus invisible. . . . . 289

C.24	Overlap of the predictions made by TARA++ and PrimAlign for rarity thresholds <b>(a, d, g)</b> ALL, <b>(b, e)</b> 50, and <b>(c, f)</b> 25 using ground truth datasets <b>(a, b, c)</b> atleast1-EXP, <b>(d, e, f)</b> atleast2-EXP, and <b>(g)</b> atleast3-EXP. Percentages are out of the total number of unique predictions made by both methods combined. Precision and recall are shown for each of the three prediction sets captured by the Venn diagram. The overlaps are for one of the 10 balanced datasets. . . . .	290
D.1	Illustration of the two part NoN-GCN layer. In the level 1 NoN-GCN layer, the level 1 node circled in red receives features from its neighbors in its level 1 network as well as features of the level 2 node its level 1 network corresponds to. This is done for every level 1 node in every level 1 network. In the level 2 NoN-GCN layer, the level 2 node circled in red received features from its neighbors in the level 2 network as well as features of each of the level 1 nodes in its level 1 network. This is done for every level 2 node in the level 2 network. . . . .	298
D.2	Comparison of the nine relevant approaches in the task of entity label prediction for synthetic NoNs with 5% across-edge amount and the following rewire-noise amounts: <b>(a)</b> 0%, <b>(b)</b> 10%, <b>(c)</b> 25%, <b>(d)</b> 50%, <b>(e)</b> 75%, and <b>(f)</b> 100%. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw prediction accuracies are shown above. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars. . . . .	305
D.3	Comparison of the nine relevant approaches in the task of entity label prediction for synthetic NoNs with 25% across-edge amount and the following rewire-noise amounts: <b>(a)</b> 0%, <b>(b)</b> 10%, <b>(c)</b> 25%, <b>(d)</b> 50%, <b>(e)</b> 75%, and <b>(f)</b> 100%. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw prediction accuracies are shown above. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars. . . . .	306
D.4	Comparison of the nine relevant approaches in the task of entity label prediction for synthetic NoNs with 50% across-edge amount and the following rewire-noise amounts: <b>(a)</b> 0%, <b>(b)</b> 10%, <b>(c)</b> 25%, <b>(d)</b> 50%, <b>(e)</b> 75%, and <b>(f)</b> 100%. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw prediction accuracies are shown above. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars. . . . .	307

D.5	Comparison of the nine relevant approaches in the task of entity label prediction for synthetic NoNs with 75% across-edge amount and the following rewire-noise amounts: <b>(a)</b> 0%, <b>(b)</b> 10%, <b>(c)</b> 25%, <b>(d)</b> 50%, <b>(e)</b> 75%, and <b>(f)</b> 100%. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw prediction accuracies are shown above. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars. . . . .	308
D.6	Comparison of the nine relevant approaches in the task of entity label prediction for synthetic NoNs with 95% across-edge amount and the following rewire-noise amounts: <b>(a)</b> 0%, <b>(b)</b> 10%, <b>(c)</b> 25%, <b>(d)</b> 50%, <b>(e)</b> 75%, and <b>(f)</b> 100%. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw prediction accuracies are shown above. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars. . . . .	309
D.7	Summarized results of the eight relevant approaches in the task of protein functional prediction for evaluation measures <b>(a)</b> AUPR, <b>(b)</b> precision, <b>(a)</b> recall, and <b>(a)</b> F-score. For each GO term (out of the 131 total), we rank the eight approaches’ classification performances from best (rank 1) to worst (rank 8). If an approach’s performance is not significantly better than expected by random we deem it “non-significant” instead. Then for each approach, we calculate the proportion of times it achieves each rank. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. . . . .	310
D.8	Overlap of GO terms for which L2 SIGN is the best with those for which <b>(a, e, i, m)</b> L1 GCM + L2 GDV, <b>(b, f, j, n)</b> L1 DiffPool + L2 SIGN, <b>(c, g, k, o)</b> Combined all (aka L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN), and <b>(d, h, l, p)</b> GCN-2 are the best in terms of <b>(a, b, c, d)</b> AUPR, <b>(e, f, g, h)</b> precision, <b>(i, j, k, l)</b> recall, and <b>(m, n, o, p)</b> F-score. . . . .	311
D.9	Overlaps of the four combined level approaches for groups <b>(a, c, e, g)</b> “S < C” and <b>(b, d, f, h)</b> “C only” in terms of <b>(a, b)</b> AUPR, <b>(c, d)</b> precision, <b>(e, f)</b> recall, <b>(g, h)</b> F-score. . . . .	312
D.10	Classification performance of the eight relevant approaches for each GO term in terms of AUPR. GO term IDs and the number of positive instances for that GO term are shown above. Random performance is indicated by the dotted black line. Approaches with performance not significantly greater than random are shown in a lighter shade. GO terms are split into the six groups based on how single versus combined level approaches perform. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw scores for each approach for each GO term can be found in Supplementary File D.1. . . . .	313

D.11	Classification performance of the eight relevant approaches for each GO term in terms of precision. GO term IDs and the number of positive instances for that GO term are shown above. Random performance is indicated by the dotted black line. Approaches with performance not significantly greater than random are shown in a lighter shade. GO terms are split into the six groups based on how single versus combined level approaches perform. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw scores for each approach for each GO term can be found in Supplementary File D.2.	314
D.12	Classification performance of the eight relevant approaches for each GO term in terms of recall. GO term IDs and the number of positive instances for that GO term are shown above. Random performance is indicated by the dotted black line. Approaches with performance not significantly greater than random are shown in a lighter shade. GO terms are split into the six groups based on how single versus combined level approaches perform. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw scores for each approach for each GO term can be found in Supplementary File D.3. . . . .	315
D.13	Classification performance of the eight relevant approaches for each GO term in terms of F-score. GO term IDs and the number of positive instances for that GO term are shown above. Random performance is indicated by the dotted black line. Approaches with performance not significantly greater than random are shown in a lighter shade. GO terms are split into the six groups based on how single versus combined level approaches perform. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw scores for each approach for each GO term can be found in Supplementary File D.4. . . . .	316

## TABLES

3.1	Number of nodes and edges in the two considered PPI networks . . .	64
3.2	Number of nodes in the two considered heterogeneous protein-GO networks . . . . .	66
3.3	Number of edges in the two considered heterogeneous protein-GO networks . . . . .	67
4.1	Table of notations and their meanings . . . . .	97
4.2	Representative choices of TARA’s percent training tests for each of the 9 ground truth datasets . . . . .	111
4.3	Comparability of the existing methods considered in this study to TARA in terms of type of information used (T versus S versus TS) and alignment type (one-to-one versus many-to-many) . . . . .	114
4.4	Running time (rounded to the nearest second) comparison of TARA, WAVE, SANA, PrimAlign-T, and PrimAlign-TS for ALL GO terms .	118
4.5	Three NA method groups based on how input data are processed . .	126
4.6	Categories that relevant NA methods belong to . . . . .	128
5.1	Existing approaches that we consider and their generalized NoN counterparts . . . . .	170
5.2	Description of the six GO term groups based on how single-level (S) and combined-level (C), i.e., NoN, approaches perform . . . . .	179
5.3	Number of GO terms in each of the six groups for AUPR, precision, recall, and F-score . . . . .	180
A.1	Details on the PINs that we use in our study . . . . .	207
A.2	Method parameters for PNA that we use in our study . . . . .	208
A.3	Theoretic time complexity . . . . .	210
A.4	Overall ranking of the NA methods for the PE framework . . . . .	211
A.5	Overall ranking of the NA methods for the PE framework . . . . .	213
A.6	Overall ranking of the NA methods for the PE framework . . . . .	214
A.7	Overall ranking of the NA methods for the PE framework . . . . .	215

A.8	Overall ranking of the NA methods for the ME framework . . . . .	216
A.9	Overall ranking of the NA methods for the ME framework . . . . .	217
A.10	Overall ranking of the NA methods for the ME framework . . . . .	218
A.11	Overall ranking of the NA methods for the ME framework . . . . .	219
A.12	Overall ranking of the NA methods for the ME framework . . . . .	220
A.13	Overall ranking of the NA methods for the ME framework . . . . .	221
A.14	Overall ranking of the NA methods for the ME framework . . . . .	222
A.15	Overall ranking of the NA methods for the ME framework . . . . .	223
A.16	Overall ranking of the NA methods for the ME framework . . . . .	224
A.17	Overall ranking of the NA methods for the ME framework . . . . .	225
A.18	Overall ranking of the NA methods for the ME framework . . . . .	226
C.1	Running times (in seconds) of TARA-TS, TARA, PrimAlign, and Sequence, when considering ALL GO terms . . . . .	287
D.1	Running times of each approach in seconds. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN . . . . .	318



## CHAPTER 1

### INTRODUCTION

#### 1.1 Overview

Many real-world systems, from microscopic proteins to worldwide human civilization, are comprised of different kinds of entities and relationships between them. Representing such systems as networks (graphs) [11], in which the entities are modeled as nodes and the relationships as edges, allows for one to study them mathematically. This means that domain scientists can take advantage of the vast and rapidly growing amount of computational approaches for analyzing these networks, deepening the understanding of the systems they model. This dissertation focuses on the development of novel computational approaches for analyzing networks in the biology domain, since understanding the systems modeled by biological networks can give insights into human health, and because most of biological network data is publicly available, unlike in many other domains. However, the ideas in this dissertation are applicable to any network type in any domain.

Protein-protein interaction (PPI) networks are a widely studied type of biological networks since they can be used to model cellular functioning. In such networks, nodes are proteins and edges are PPIs. While biotechnological advancements have made PPI network data available for many species [21, 9, 38, 82], functions of many proteins in many of these species remain unknown [46]. But understanding these functions is important because, for example, understanding proteins' roles in a disease such as cancer can lead to the development of better preventative measures or

treatments. While researches would ideally perform wet lab experiments to uncover proteins' functions, doing so can be expensive, time consuming, and potentially ethically restricted if e.g., dealing with the human species. Therefore, computational analyses for determining proteins' functions, including PPI network analyses, are used in a complementary fashion to deepen the protein functional knowledge that wet lab experiments alone cannot uncover.

Many important proteins and their roles in cellular functioning are (partially) conserved across different species due to evolution. So, one way to uncover proteins' functions is by transferring biological knowledge from a well-studied species, such as *Saccharomyces cerevisiae* (yeast), *Caenorhabditis elegans* (worm), *Drosophila melanogaster* (fly), or *Mus musculus* (mouse), to a less-well-studied one, such as *Homo sapiens* (human). Genomic sequence alignment has traditionally been used for this task, by transferring functional knowledge between conserved (aligned) sequence regions of proteins in different species. However, genomic sequence alignment does not consider the interactions between proteins, which are ultimately what carry out cellular function. So, network alignment (NA) can be used in a complementary fashion to predict what sequence alignment alone cannot [146, 53, 110, 48, 47, 72]. NA aims to find a node mapping (i.e., an alignment) between the compared networks that uncovers regions of high topological (and often sequence) similarity. Then, NA methods assume that aligned proteins across these regions of high topological similarity perform similar functions, i.e., are functionally related. So, analogous to genomic sequence alignment, NA can be used to transfer knowledge across species between their conserved network (rather than just sequence) regions, predicting functions of proteins in the less-well-studied species based on functions of their aligned counterparts in the well-studied species. This task of across-species protein functional prediction is one of the ultimate goals of biological NA. So, one of the key goals of

this dissertation is to introduce novel computational directions for the problem of biological NA.

However, across-species knowledge transfer is not the only way to uncover functions of proteins. Instead, one can infer proteins’ functions based on the proteins’ relationships to other proteins with known functions *within* the same species. Node label prediction [20], a general network science task under the “graph (machine) learning” umbrella, is one such way to do this. Given a single network, the goal is to predict labels of its *nodes*; for example, given proteins in a PPI network, one can train a classifier to uncover patterns between proteins’ PPI network-based features and their functions.

Both NA and node label prediction, in the context of protein functional prediction, deal with networks at the same scale: PPI networks. However, at a finer-grained scale, proteins themselves consist of a sequence of amino acids that folds in 3D space. This structure of a protein has important implications for its function, so computational analysis of protein structure is another lens under which one can study proteins. While traditional structural analysis typically relies on 3D geometric transformations [182], methods based on protein structure networks (PSNs), in which nodes are amino acids and edges join amino acids that are close to each other with respect to the folded protein, have been shown to outperform non-network-based models in tasks such as protein structural comparison/classification [55, 125] and protein functional prediction [63]. This introduces another way to uncover proteins’ functions, by analyzing proteins’ structure networks. In particular, graph label prediction [128], a complementary task to node label prediction, can be used. Given multiple networks, the goal is to predict labels of those *networks*; for example, given multiple PSNs, one can train a classifier to uncover patterns between proteins’ PSN-based structural features and their functions.

An interesting multiscale (i.e., multilevel) relationship is evident here. Namely, proteins interact with each other, modeled as PPI networks, and proteins themselves consist of interacting amino acids, modeled as PSNs. That is, nodes in a network at a higher level are themselves networks at a lower level – some entities, in this case proteins, both participate in and are themselves composed of interactions. So, as another key contribution of this dissertation, we argue that such a system of systems should be integrated as a “network of networks” (NoN), and that another type of method, namely entity label prediction using information (features) from both levels of an NoN, is worth exploring for protein functional prediction.

In summary, in this dissertation, we develop novel computational approaches for two kinds of multi-network analyses under the task of protein function prediction: across-species PPI network alignment (Section 1.2) and multilevel NoN-based entity (protein) label prediction (Section 1.3). Again, we note that while this dissertation focuses on the computational biology application of NoNs, systems in other domains can also exhibit this multilevel relationship (Section 1.3.2), and our idea are also applicable to them.

## 1.2 Network alignment

### 1.2.1 Background

NA is closely related to the NP-complete subgraph isomorphism, or subgraph matching, problem, in which the goal is to find a node mapping such that one network is an exact subgraph of another network [35]. However, NA is more general in that it aims to find the best “fit” of one network into another network, even if the first is not an exact subgraph of the second. A widely used measure to quantify this “fit” is the amount of conserved (aligned) edges, i.e., the size of the common conserved

subgraph between the aligned networks. But because maximizing edge conservation is NP-hard [95], all NA methods are heuristics.

NA can be categorized into several broad types, whose high-level input / output / goal differences are as follows.

First, NA can be local or global (Fig. 1.1), like sequence alignment. Local NA aims to find highly conserved network regions but usually results in such regions being small [16, 17, 58, 89, 94, 98, 147, 29, 117]. Global NA aims to maximize overall network similarity; while it usually results in large aligned network regions, these regions are suboptimally conserved [54, 59, 91, 95, 96, 99, 114, 122, 126, 130, 153, 154, 178]. Both have their own (dis)advantages [110, 72]. As global NA has received more attention recently than local NA, we focus on global NA in this dissertation.

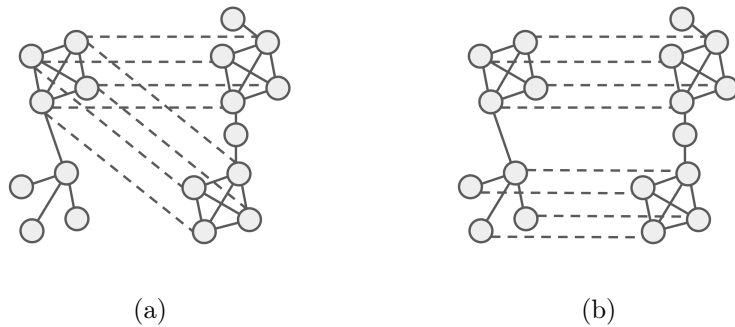


Figure 1.1. Illustration of alignments produced by **(a)** local and **(b)** global NA. Dashed lines are between nodes that are aligned to each other. This figure is adapted from [110].

Second, NA can be one-to-one (each node can be aligned to exactly one distinct node in another network) or many-to-many (a node may be aligned to more than one node in another network). Since both one-to-one and many-to-many alignments

can be used in our considered task of across-species protein functional prediction, we consider both in this dissertation.

Given these categorizations of local, global, one-to-one, and many-to-many NA, it is important to clear up some terminology regarding them (even though we do not consider local NA in this dissertation, we include it below for completeness). Traditionally, given networks  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$ , where, without loss of generality,  $|V_1| \leq |V_2|$ , local NA has meant the same as many-to-many NA: a relation  $R \subseteq V_1 \times V_2$  (Fig. 1.1(a)). Also, global NA has meant the same as one-to-one NA: an injective function  $f : V_1 \rightarrow V_2$  (Fig. 1.1(b)). However, over time, methods claiming to be local and one-to-one, or global and many-to-many, have been proposed.

Thus, there are actually four combinations from the aforementioned categorizations, defined as follows. Given  $S_1 \subseteq V_1$ , local one-to-one NA should be thought of as an injection from  $S_1$  to  $V_2$  where  $S_1$  is much smaller than  $V_1$  (few nodes in  $G_1$  are mapped to nodes in  $G_2$ ). Global one-to-one NA should be thought of as an injection from  $S_1$  to  $V_2$  where  $S_1$  and  $V_1$  have similar, or are the same, size (most nodes in  $G_1$  are mapped to nodes in  $G_2$ ). Local many-to-many NA should be thought of as a relation  $R \subseteq S_1 \times V_2$  where  $S_1$  is much smaller than  $V_1$ . And global many-to-many NA should be thought of as a relation  $R \subseteq S_1 \times V_2$  where  $S_1$  and  $V_1$  have similar, or are the same, size.

Third, NA can be pairwise (aligns two networks) or multiple (aligns three or more networks) [53, 72]. Both pairwise (Fig. 1.2(a, c)) and multiple (Fig. 1.2(b, d)) NA can produce one-to-one (Fig. 1.2(a, b)) or many-to-many (Fig. 1.2(c, d)) alignments. We consider all four possible combinations in this dissertation.

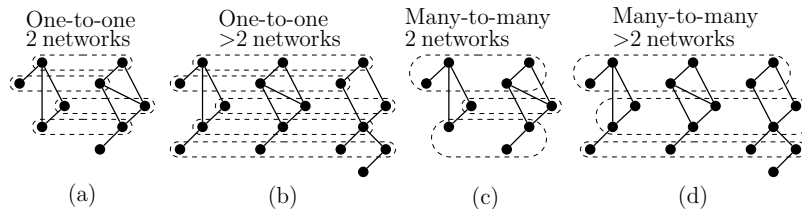


Figure 1.2. Illustration of a **(a)** pairwise one-to-one alignment; **(b)** multiple one-to-one alignment; **(c)** pairwise many-to-many alignment; and **(d)** multiple many-to-many alignment. In this toy example we only show three networks for multiple NA, but multiple NA can be used on more than three networks as well.

### 1.2.2 Related work

To find a good “fit” between networks, NA methods generally consist of two algorithmic components. First, a mathematical heuristic is used to quantify the topological similarity (how close to isomorphic two nodes’ extended neighborhoods are) between nodes across networks [153, 130, 96, 114, 95, 107, 102, 158, 164, 103, 53, 72]. Nodes’ topological similarities are also sometimes combined with non-network-based similarities; for biological NA, incorporating the corresponding proteins’ sequence similarities is popular [153, 95, 130].

Second, these (possibly combined) node similarities are given to an alignment strategy that aims optimize some objective function. Early NA methods’ alignment strategies aimed to maximize the overall similarity between nodes in the alignment, in hopes that doing so would conserve edges and achieve an isomorphic-like matching. For example, methods used matching algorithms, such as a simple greedy approach [153] or the Hungarian algorithm [114], on the nodes’ similarities. As another example, seed-and-extend approaches were also developed [96, 130, 158, 95]; in such approaches, first two highly similar nodes are aligned, i.e., seeded. Then, the most similar of the seed’s neighboring nodes, the neighbors of the seed’s neighbors, etc. are aligned. This step of extending around the seed is intended to improve the over-

all similarity between nodes as well as *implicitly* conserve edges. Later NA methods' alignment strategies shifted to search algorithms. By efficiently searching through the space of possible alignments via genetic algorithms [143, 164] or simulated annealing [103], such algorithms could explicitly optimize for (i.e., find) alignments with high overall node similarity and high edge conservation.

However, as we discuss in Sections 1.2.3.2 and 1.2.3.3 (correspondingly, Chapters 3 and 4), these existing NA methods have drawbacks that we aim to address in this dissertation.

### 1.2.3 Research questions and our contributions

#### 1.2.3.1 Pairwise versus multiple network alignment

It is hypothesized that multiple NA leads to higher quality alignments than pairwise NA [99, 163, 74, 4, 28], and thus to deeper biological insights, since the former considers information from more networks at once compared to pairwise NA. However, because fairly comparing pairwise and multiple NA is difficult due to their different output types (Fig. 1.2), the hypothesis that multiple NA is superior to pairwise NA had not been tested previously.

**Research question.** So, the work in this dissertation aims to answer whether multiple NA indeed leads to deeper biological insights, i.e., to higher across-species protein functional prediction accuracy, than pairwise NA.

**Our contributions.** To answer this, we develop an evaluation framework to allow for a fair comparison of pairwise and multiple NA [166]. In particular, we evaluate prominent pairwise and multiple NA methods on synthetic and real-world biological networks, using topological (are aligned regions isomorphic-like) and functional (are aligned nodes functionally related) alignment quality measures. We compare pairwise and multiple NA in both a pairwise (native to pairwise NA) and multiple (native to multiple NA) manner. Pairwise NA is expected to perform better under the pairwise



evaluation framework. Indeed this is what we find. Multiple NA is expected to perform better under the multiple evaluation framework. Shockingly, we find this not always to hold; pairwise NA is often better than multiple NA in this framework, depending on the choice of evaluation test. Furthermore, pairwise NA is faster than multiple NA on average. For these reasons, we focus on pairwise NA for the remainder of the dissertation.

### 1.2.3.2 Heterogeneous network alignment

Regardless of the NA category, one issue of existing biological NA methods is that the alignments they produce are of low functional alignment quality. In particular, when comparing PPI networks of different species, aligned nodes (proteins) often do not correspond to proteins that perform the same biological function [130, 115, 30, 110, 72]. In other words, current NA methods are not achieving their goal, and thus NA cannot effectively transfer functional knowledge across networks.

This could be because existing NA methods deal with homogeneous networks, where nodes are of a single type and edges are of a single type. However, networks can have nodes or edges of more than one type (Fig. 1.3). For example, different biological entities, such as proteins, phenotypes, or drugs, can be modeled as nodes, and different types of interactions, such as protein-protein, phenotype-phenotype, drug-drug, protein-phenotype, protein-drug, or phenotype-drug associations can be modeled as edges. Even within PPI networks, proteins themselves can be categorized in different ways, for example based on if they are associated with a certain disease or not. Capturing such heterogeneous information via a new node similarity heuristic, or developing an alignment strategy that favors aligning nodes of the same type, could lead to higher quality alignments.

**Research question.** So, the work in this dissertation aims to answer whether heterogeneous NA is superior to homogeneous NA.

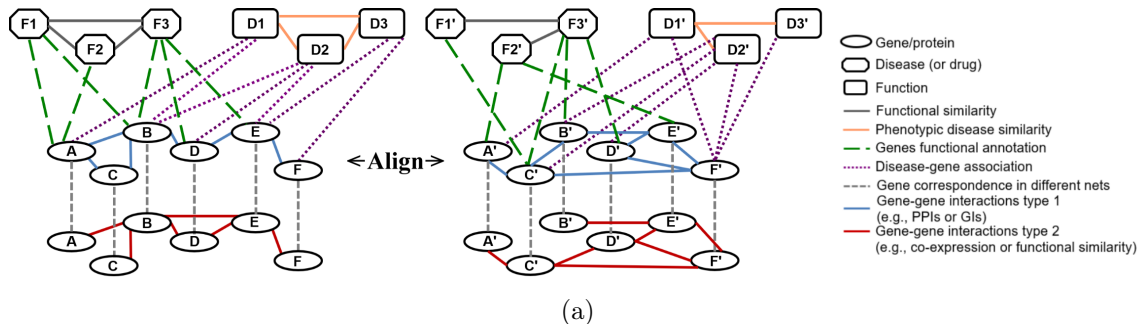


Figure 1.3. Illustration of two heterogeneous networks, each containing different node as well as edge types (or colors). In a given network, different node shapes represent different node types, and different line styles represent different edge types. If we do not consider the ovals with red edges (the bottom portion of the network), then we have a heterogeneous network with different node types, and thus implicitly different edge types. If we only consider the ovals with blue or red edges, then we have a heterogeneous network with different edge types but a single node type (also called multimodal networks with two edge modes). The goal of HetNA as we define it is to find a node mapping between heterogeneous networks that contain different node types, different edge types, or both.

**Our contributions.** To answer this, we generalize three prominent homogeneous NA methods, WAVE [158], MAGNA++ [164], and SANA [103], to their heterogeneous counterparts to allow for heterogeneous NA for the first time [70]. We introduce several algorithmic novelties for this. Namely, these existing methods compute homogeneous graphlet-based node similarities and then find high-scoring alignments with respect to these similarities, while simultaneously maximizing the amount of conserved edges. Instead, we extend homogeneous graphlets to their heterogeneous counterparts, which we then use to develop a new measure of heterogeneous node similarity. Also, we extend  $S^3$ , a state-of-the-art measure of edge conservation for homogeneous NA, to its heterogeneous counterpart. Then, we find high-scoring alignments with respect to our heterogeneous node similarity and edge conservation measures. In evaluations on synthetic and real-world biological networks, our proposed heterogeneous NA methods lead to higher-quality alignments, are more robust to

noise in the data, and are comparable in terms of running time compared to their homogeneous counterparts.

### 1.2.3.3 Data-driven network alignment

Another possible reason that existing NA methods produce alignments of low functional quality is due to the assumption that they make. Namely, they assume that topologically similar nodes, i.e., those with isomorphic-like neighborhoods to each other, are functionally related. Hence, many existing NA methods use node similarity heuristics that quantify this isomorphic-like matching. However, this assumption may be flawed. In the context of PPI networks, much data is still incomplete or noisy [93]. This alone can cause mismatches between proteins' topological similarity and functional relatedness. For example, if a set of three proteins that are all linked to each other via PPIs (i.e., a triangle) is in reality fully evolutionary conserved (i.e., perform similar functions) between two species, then the two triangles in the two species are topologically similar. But say that one of the three PPIs that actually exists in reality is missing in exactly one of the two species' current PPI networks due to data noise. Then, it is a 3-node path in that species that should be aligned to a triangle in another species in order to identify functional relatedness. That is, proteins that perform similar functions are now topologically dissimilar due to the data noise.

Even when PPI network data become complete, the traditional assumption of topological similarity is unlikely to hold due to biological variation between species. Namely, molecular evolutionary events such as gene duplication, deletion, or mutation may cause PPI network topology to differ across species' evolutionary conserved (i.e., functionally related) network regions. Even for protein sequence alignments, pairwise sequence identity as low as 30% is sufficient to indicate evolutionary conservation (i.e.,

homology) for 90% of all protein pairs [140]. So, one can perhaps expect evolutionary conserved PPI networks of different species to be as topologically dissimilar.

**Research question.** The work in this dissertation aims to answer whether the pre-defined topological similarity assumption of existing NA methods holds, and if not, to shift the paradigm of NA so that it does not rely on this assumption, instead learning from the data what nodes should be aligned.

**Our contributions.** We find that the topological similarity assumption does not hold well. Namely, given node pairs known to be functionally related and node pairs known to be functionally unrelated, we analyze the distributions of topological similarity for the two groups. If the topological similarity distribution of functionally related pairs is very different from that of functionally unrelated pairs, then topological similarity can distinguish between functionally related and unrelated pairs. However, we find that the topological similarity distributions of the two groups are actually close to each other – if one selects a topologically similar pair, it is almost an equal chance for that pair to be functionally related as functionally unrelated.

Consequently, we shift the paradigm of how the NA problem is approached. Specifically, we redefine NA as a data-driven framework, called TARA (data-driven NA), which attempts to learn the relationship between nodes’ “topological relatedness” and their functional relatedness without assuming that topological relatedness means topological similarity [68]. Because the distinction between topological relatedness and similarity is crucial to understanding our framework, we illustrate and describe the difference in Fig. 1.4.

TARA makes no assumptions about what nodes should be aligned, distinguishing it from existing NA methods. Specifically, TARA trains a classifier to predict whether two nodes from different networks are functionally related based on their network topological patterns (features). We find that TARA *is* able to make accu-

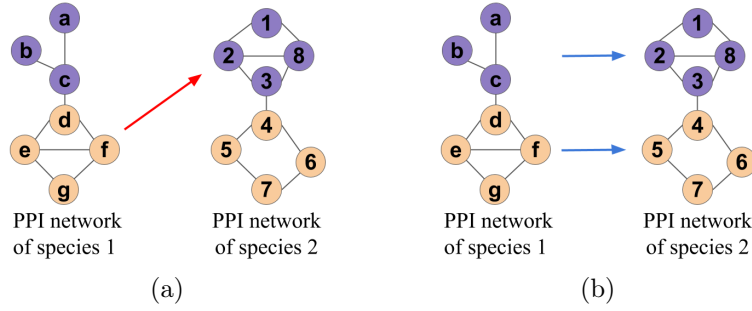


Figure 1.4. Illustration of the existing notion of topological similarity versus our new notion of topological relatedness. Suppose that we are aligning PPI networks of two different species, where for simplicity, only parts of the whole networks are shown. Also, suppose that a color corresponds to the function that a node performs, in this case the “purple” function or the “orange” function. **(a)** An NA method based on topological similarity will produce an alignment with low functional quality on our example networks. Such a method will align nodes  $d$ ,  $e$ ,  $f$ , and  $g$  in species 1 to nodes 1, 2, 3, and 8 in species 2 because each set of nodes forms the same subgraph: a square with a diagonal (square-with-diagonal). However, the species 1 nodes perform the “orange” function, while the species 2 nodes perform the “purple” function – the nodes are not functionally related. **(b)** On the other hand, an NA method based on topological relatedness will produce an alignment with high functional quality on our example networks. This is because such a method will learn that 3-node paths in species 1 should be aligned to square-with-diagonals in species 2, since the 3-node path consisting of nodes  $a$ ,  $b$ , and  $c$  in species 1 performs the same function (“purple”) as the square-with-diagonal consisting of nodes 1, 2, 3, and 8 in species 2; and that square-with-diagonals in species 1 should be aligned to squares in species 2, since the square-with-diagonal consisting of nodes  $d$ ,  $e$ ,  $f$ , and  $g$  in species 1 performs the same function (“orange”) as the square consisting of nodes 4, 5, 6, and 7 in species 2. Using these learned patterns, the method will try to align the rest of the nodes between the networks (not shown in the figure), transferring the functions of 3-node paths to square-with-diagonals, and of squares-with-diagonals to squares. In essence, noisy data or evolutionary events can be captured by topological relatedness but not topological similarity.

rate predictions. TARA then takes each pair of nodes that are predicted as related to be part of an alignment, thus achieving high functional quality.

Recall that NA methods can either rely solely on topological information to align nodes, or they can use both topological and protein sequence information. TARA as initially implemented is of the former, using only topological information. In this context, we find that TARA outperforms existing state-of-the-art NA methods that also use topological information, WAVE and SANA, and even outperforms or complements a state-of-the-art NA method that uses both topological and sequence information, PrimAlign [87]. Thus, we go further and extend TARA into TARA++ [69], which uses both topological and sequence information. We do so by creating an integrated network. In particular, we convert two homogeneous networks (PPI networks of the species being compared) into one combined network consisting of proteins of species 1, proteins of species 2, PPIs of species 1, PPIs of species 2, and across-species links based on sequence information between the proteins. Then, we adapt social network embedding to extract features from this integrated network, which we then use in TARA’s data-driven framework. We find that in doing so, TARA++ improves upon both TARA and PrimAlign.

### 1.3 Modeling multi-scale data via a network of networks

#### 1.3.1 Background

To reiterate, sometimes the entities (e.g., proteins) that are represented by nodes in a network (e.g., a PPI network) can themselves be modeled as networks (e.g., PSNs). We argue that the systems involving such entities should be integrated into a “network of networks” (NoN), where nodes in a network at a higher level are themselves networks at a lower level (Fig. 1.5). More specifically, we refer to the higher level of the NoN as the level 2 network (Fig. 1.5(a)), which contains level 2 nodes and level 2 edges. Each level 2 node has a corresponding level 1 network at the lower level of the NoN (Fig. 1.5(b)), which contains level 1 nodes and level 1 edges. We number

levels in this way with the idea that lower-level networks are the building blocks of higher-level networks. However, we tend to discuss level 2 networks first, as doing so is often more convenient for developing intuition. Even though we analyze two-level NoNs in this dissertation, NoNs can encompass more: proteins interact with each other to carry out cellular functioning, cells interact with each other to form tissues, and so on, up the levels of biological organization.

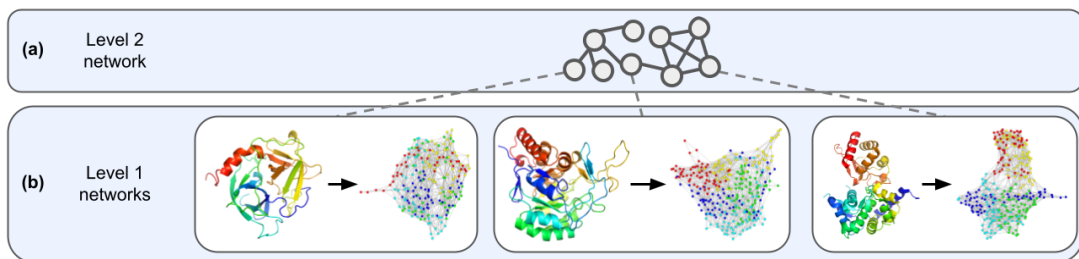


Figure 1.5. Illustration of a two-level biological NoN. Level 2 nodes (proteins) in **(a)** the level 2 network (PPI network) are joined to their corresponding **(b)** level 1 networks (PSNs) by dotted lines. Only three level 1 networks are shown for simplicity, but generally every level 2 node can have a corresponding level 1 network. Nodes in the PSNs are colored based on their corresponding amino acids in the ribbon diagram and are not indicative of node labels.

### 1.3.2 Related work

Some other network models of higher-order data do exist. These include: multiplex, multimodal, multilevel, and interdependent networks [141, 119, 25, 97, 40, 134], which are sometimes used interchangeably and sometimes also referred to as “networks of networks”; hierarchical networks [31]; higher-order networks [172]; hypergraphs [18]; and simplicial complexes [121]. However, these all model different types

of data compared to NoNs as we define them, so we cannot consider these other network types in our dissertation.

There are also studies that do model data as NoNs. However, they differ from our work in terms of data analyzed, application domain, and/or network science task. With respect to data, besides synthetic NoNs, we analyze a PPI network-PSN biological NoN. However, these other studies analyze NoNs where the level 2 network is a disease-disease similarity network and the level 1 networks are disease specific PPI networks [127], where the level 2 network is a social network and the level 1 networks are individuals’ brain networks [56, 129, 12], or where the level 2 network is a chemical-chemical interaction network and level 1 networks are molecule networks [168]. With respect to application domain, while we aim to predict protein function, these other studies aim to identify disease causing genes [127], answer sociologically motivated questions like whether similarities between friends mean they have similar ways of thinking [129], or predict new chemical-chemical interactions [168]. With respect to network science task, while we aim to predict entities’ labels, these other studies aim to identify important entities (level 1 nodes) [127], predict links between entities (level 2 nodes) [168], or embed multiple networks at the same level into a common low dimensional space, using an NoN as an intermediate step [42]. While it *might* be possible to extend some of these existing studies to ours or vice versa, doing so could require considerable effort, as it would mean developing new methods, and code is not publicly available for all of the existing methods. All of this makes any potential extensions hard. As such, we cannot compare against these existing NoN-like methods.

### 1.3.3 Research questions and our contributions

Given our definition of an NoN, we can characterize the task of entity label prediction in the context of this dissertation. Specifically, since the entities of interest



are represented by level 2 nodes and, correspondingly, modeled as level 1 networks, entity label prediction can refer to (i) using only the level 2 network (Fig. 1.5(a)) to predict level 2 nodes’ labels, corresponding to the task of node label prediction in the level 2 network, (ii) using only level 1 networks (Fig. 1.5(b)) to predict level 1 networks’ labels, corresponding to the task of graph label prediction using the level 1 networks, and (iii) using the entire NoN to predict entities’ labels.

**Research question.** Thus, the primary question we aim to answer is whether (iii) is more accurate than (i) and (ii), i.e., if NoN-based entity label prediction is more accurate than each of single-level node label prediction and graph label prediction alone.

**Our contributions.** In tackling this question, we make the following novel contributions: we construct and provide two new sources of NoN data, we develop novel approaches for NoN label prediction, and, most importantly, we are the first to test whether using NoN data in label prediction is more accurate than using only single level data.

Since to our knowledge, labeled NoNs are limited, we provide two new sources of such data. First, we develop an NoN generator that can create a variety of synthetic NoNs (Chapter 5.2.3.1). Intuitively, given any set of single-level random graph generators, such as geometric [133] or scale-free [10], our NoN generator combines random graphs created from these single-level generators at each level. In this way, we can label each entity (level 2 node and its level 1 network) based on which combination of single-level random graph generators it is involved in at the two levels. Our generator can control a variety of network structural parameters, thus allowing for the mimicking of a variety of real-world systems. Second, we construct a biological NoN, consisting of a PPI network from BioGRID [156] at the second level and PSNs for proteins from Protein Data Bank (PDB) [19] at the first level. Proteins are labeled based on their functions via Gene Ontology (GO) annotation data [8] (Chapter

5.2.3.2). For each of the GO terms considered, the goal is to predict whether or not each protein is annotated by that GO term. While computational protein functional prediction is relatively well-studied, the problem is still very relevant, as the accuracy of existing methods for this purpose is typically low. The continued importance of computational annotation of protein function [60] is a major motivator of our dissertation. We expect the NoN data resulting from our work to become a useful resource for future research in both network science and computational biology, including for the problem of protein function prediction.

We also develop novel approaches for NoN label prediction. In general, label prediction approaches extract features of the entities and then perform supervised classification, i.e., prediction of the entities’ labels based on their features. So, for our dissertation, there are three types of approaches to consider: (i) those that extract node-level features (i.e., level 2 only), (ii) those that extract network-level features (i.e., level 1 only), or (iii) those that extract NoN features (i.e., integrated level 1 and level 2). To our knowledge, approaches of type (iii) do not exist yet, so we create NoN features in two ways: by combining existing node- and network-level features and by applying the novel graph neural network (GNN) approach that we propose for analyzing NoNs.

Then, we aim to evaluate whether approaches of type (iii) outperform those of types (i) and (ii). If so, this would provide evidence that NoN-based data integration is useful for label prediction. To determine which approach types are the best, we evaluate them in terms of accuracy for synthetic NoNs, as class sizes are balanced, and in terms of the area under the precision recall curve (AUPR), precision, recall, and F-score for the biological NoN, as class sizes are unbalanced.

For synthetic NoNs, we find that our NoN approaches outperform single-level node and network ones for those NoNs where the majority of nodes are not densely interconnected (i.e., where nodes do not tend to group into densely connected modules).

For NoNs where there are groups of densely interconnected nodes (i.e., where there is clustering structure), an existing single-level approach performs as well as NoN approaches. For the biological NoN, we find that our NoN approaches outperform the single-level ones in a little under half of the GO terms considered. Furthermore, for 30% of the GO terms considered, only our NoN approaches make meaningful predictions, while node- and network-level ones achieve random accuracy. Also, while our GNN approach does not perform the best *overall*, it seems to be useful for otherwise difficult-to-predict protein functions. As such, NoN-based data integration is an important and exciting direction for future research.

#### 1.4 Organization of the dissertation

In general, this dissertation focuses on novel computational approaches using multiple networks across different systems and scales in order to improve protein functional prediction.

Chapter 2 describes an evaluation framework for a fair comparison of pairwise and multiple NA and shows that pairwise NA is often better.

Chapter 3 describes how homogeneous NA can be generalized to heterogeneous NA, leading to higher quality alignments.

Chapter 4 describes a completely new paradigm for NA, data-driven NA, and shows how learning the relationship between nodes' topological relatedness and their functional relatedness leads to higher quality alignments and thus more accurate protein functional prediction than traditional topological similarity-based NA.

Chapter 5 describes a novel, integrative model for multiscale network data analysis, thoroughly evaluating the effects of considering each scale alone vs. together and showing that integration leads to the discovery of knowledge that cannot be found from a single scale on its own.

Chapter 6 concludes the dissertation.

## CHAPTER 2

### PAIRWISE VERSUS MULTIPLE NETWORK ALIGNMENT

The work in this chapter is discussed in the following paper:

- Vipin Vijayan, **Shawn Gu**, Eric Krebs, Lei Meng, and Tijana Milenković (2020), Pairwise versus multiple network alignment, IEEE Access, 8: 41961-41974. [166]

#### 2.1 Introduction

##### 2.1.1 Background and motivation

Recall that pairwise NA (PNA) aligns exactly two networks, while multiple NA (MNA) aligns more than two networks. Since MNA can capture conserved network regions between multiple networks at once, it is hypothesized that MNA may lead to deeper biological insights (i.e., higher-quality alignments) compared to PNA. However, this hypothesis has not been tested yet (for reasons described in the following paragraphs). Because of this, and because both PNA and MNA have the same ultimate goal, which is to transfer knowledge from well- to poorly-studied species, we argue that they need to be compared in order to determine which category of methods produce higher-quality alignments. Furthermore, MNA is computationally harder than PNA, because the complexity of the NA problem can increase exponentially with the number of considered networks [62]. So, a comparison of PNA and MNA in terms of their alignment quality can also answer whether the additional computational complexity of MNA is worth it.

Since typical PNA and MNA methods produce alignments of different types (Fig. 1.2), it has been difficult to compare them. Namely, when aligning two networks, PNA typically produces a *one-to-one* node mapping between the two networks, which results in aligned node *pairs* (Fig. 1.2(a)). When aligning more than two networks, MNA produces a node mapping across the multiple networks, which results in aligned node *clusters*. If an aligned cluster contains more than one node from a single network, then it is a *many-to-many* alignment (Fig. 1.2(d)). If each of the aligned clusters contains at most one node per network, then it is a *one-to-one* alignment (Fig. 1.2(b)). Typical MNA methods produce many-to-many alignments (Fig. 1.2(d)), and they are called many-to-many MNA methods. MNA methods that produce one-to-one alignments (Fig. 1.2(b)) are called one-to-one MNA methods. MNA methods can also be trivially used to align pairs of networks, which results in aligned node clusters for many-to-many MNA methods (Fig. 1.2(c)) and in aligned node pairs for one-to-one MNA methods (Fig. 1.2(a)).

There is sometimes confusion in the literature that one-to-one alignments are automatically global (i.e., outputted by global NA methods), and that many-to-many alignments are automatically local (outputted by local NA methods). However, this is not necessarily the case. First, one-to-one alignments can result in only small regions aligned to each other (clearly without any nodes overlapping), meaning that they are local one-to-one alignments. Second, many-to-many alignments can result in aligned node clusters covering nodes from all analyzed networks, meaning that they are global, many-to-many alignments. In other words, in our opinion, “local” and “global” describe how much of the networks’ nodes are covered by (i.e., are a part of) the given alignment, and not on whether the nodes are aligned in one-to-one or many-to-many fashion. It is important to note that most of the recent one-to-one methods will not actually produce local alignments, because they require all nodes of the smaller networks to be mapped to nodes of the larger networks, automatically leading

to global (one-to-one, or even more formally, injective) alignments. However, this is an algorithmic design choice of many existing methods rather than a requirement of any and every one-to-one method. As discussed in Chapter 1.2, we focus on global NA, considering both one-to-one and many-to-many methods.

Again, because PNA and MNA generally produce alignments of different types (aligned node pairs versus aligned node clusters, respectively), alignment quality measures designed for alignments of one type do not necessarily work for alignments of the other type. Also, alignment quality measures designed for alignments of two networks do not necessarily work for alignments of more than two networks. Due to this difficulty, when a new PNA or MNA method is proposed, it is only compared against other NA methods from the same category. However, since both PNA and MNA have the same goal of across-species knowledge transfer, we argue that there is a need to compare them. This is especially true because early evidence suggests that aligning each pair of considered networks via PNA and then combining the pairwise alignments into a multiple alignment spanning all of the networks can be superior to directly aligning all networks via MNA [39].

### 2.1.2 Our contributions

Thus, we propose an evaluation framework for a fair comparison of PNA and MNA (Fig. 2.1).

We evaluate PNA and MNA on synthetic networks with known true node mapping (we know the underlying alignment that a perfect method should output) and real-world PPI networks of different species with unknown node mapping (we do not know which protein in one species corresponds to which protein in the other species). The network data are discussed in Section 2.2.1.

We evaluate prominent PNA and MNA methods that were published by the beginning of this study, were publicly available, and had user-friendly implementations.

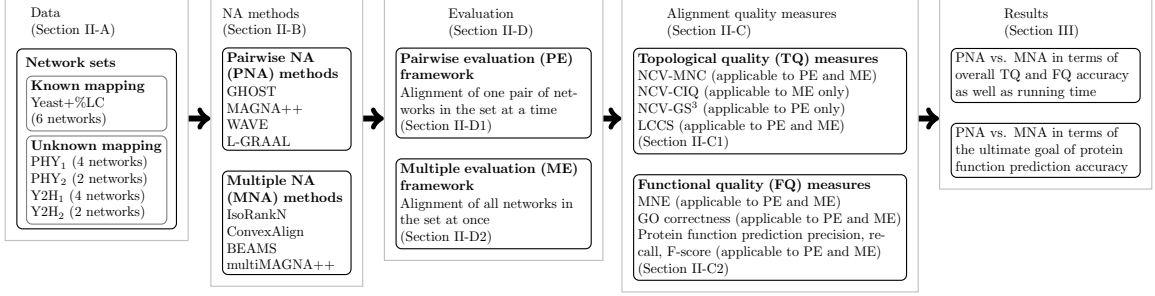


Figure 2.1: Overview of our PNA versus MNA evaluation framework.

This includes four PNA methods (GHOST [130], MAGNA++ [164], WAVE [158], and L-GRAAL [102]), and four MNA methods (IsoRankN [99], BEAMS [4], multi-MAGNA++ [163], and ConvexAlign [74]), which are discussed in Section 2.2.2. Most of these methods are recent and were thus already shown to be superior to many past methods, e.g., IsoRank [153], MI-GRAAL [95], GEDEVO [83], and NETAL [126] PNA methods, plus GEDEVO-M [84], FUSE [62], and SMETANA [142] MNA methods. Note that newer NA methods have appeared since, such as SANA [103], ModuleAlign [75], SUMONA [160], and PrimAlign [87], which is why they were not included here. Importantly, we believe that their inclusion is not required. This is because our goal is **not** to determine the best existing (PNA or MNA) method. Instead, it is to properly evaluate the whole category of prominent recent PNA methods against the whole category of equally prominent recent and thus fairly comparable MNA methods. While the best existing NA method would likely change with introduction of each new method (or possibly even a new measure for evaluating alignment quality), the best category of NA approaches is less likely to change, unless there is a drastic shift in how the NA problem is approached and solved (or possibly even just how alignment quality is evaluated). And one of the purposes of this study is to determine if such a shift is needed.

We evaluate the PNA and MNA methods in terms of their alignment quality (i.e., accuracy) as well as running time. We evaluate alignment quality using topological and functional alignment quality measures. An alignment is of good topological quality if it reconstructs well the underlying true node mapping (when known) and if it has many conserved edges (i.e., if it conserves a large common subgraph between the networks). An alignment is of good functional quality if its aligned node pairs/clusters contain nodes with similar biological functions. The alignment quality measures are described in Section 2.2.3.

We evaluate the PNA and MNA methods in both a pairwise (native to PNA) and multiple (native to MNA) manner, as described in Section 2.2.4.

Section 2.2 describes the data, alignment quality measures, and evaluation framework. Section 2.3 describes our findings.

Since typical PNA and MNA methods produce alignments of different types (Fig. 2.2), it has been difficult to compare them. Namely, when aligning two networks, PNA typically produces a one-to-one alignment between the two networks, which results in aligned node pairs (Fig. 2.2(a)). Recently, more PNA methods have appeared that produce many-to-many alignments, resulting in aligned node clusters (Fig. 2.2(c)). When aligning more than two networks, MNA produces a node mapping which results in aligned node clusters. If each of the aligned node clusters contains at most one node per network, then it is a one-to-one alignment (Fig. 2.2(b)). If an aligned cluster contains more than one node from a single alignment, then it is a many-to-many alignment (Fig. 2.2(d)). Typical MNA methods produce many-to-many alignments (Fig. 2.2(d)). MNA methods that produce one-to-one alignments (Fig. 2.2(b)) are called one-to-one MNA methods. MNA methods can also be trivially used to align pairs of networks, which results in aligned node clusters for many-to-many MNA methods (Fig. 2.2(c)) and in aligned node pairs for one-to-one MNA methods (Fig. 2.2(a)).



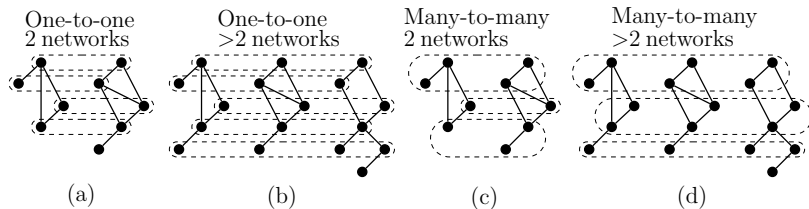


Figure 2.2. Illustration of different alignment types.

Because PNA and MNA generally produce alignments of different types (aligned node pairs versus aligned node clusters, respectively), alignment quality measures designed for alignments of one type do not necessarily work for alignments of the other type. Also, alignment quality measures designed for alignments of two networks do not necessarily work for alignments of more than two networks. Due to this difficulty, when a new PNA or MNA method is proposed, it is only compared against other NA methods from the same category. However, since both PNA and MNA have the same goal of across-species knowledge transfer, we argue that there is a need to compare them. This is especially true because early evidence suggests that aligning each pair of considered networks via PNA and then combining the pairwise alignments into a multiple alignment spanning all of the networks can be superior to directly aligning all networks via MNA [39].

## 2.2 Methods

### 2.2.1 Data

We use five network sets: one synthetic network set with known true node mapping, and four real-world network sets with unknown true node mapping. For each network, we use only its largest connected component.

**Network set with known true node mapping.** This synthetic network set, named Yeast+LC, contains a high-confidence *S. cerevisiae* (yeast) PPI network

with 1,004 proteins and 8,323 interactions [32], along with five lower-confidence yeast PPI networks constructed by adding 5%, 10%, 15%, 20%, or 25% of lower-confidence interactions to the high-confidence PPI network (Supplementary Table A.1). This network set has been used in many existing studies [96, 114, 95, 130, 143, 110, 163]. Since all networks have the same node set, we know the true node mapping. Hence, for this set, we can evaluate node correctness, i.e., how well the given NA method reconstructs the true node mapping (Section 2.2.3.1).

**Network sets with unknown true node mapping.** The four real-world network sets with unknown node mapping are named PHY<sub>1</sub>, PHY<sub>2</sub>, Y2H<sub>1</sub>, and Y2H<sub>2</sub>. Each contains PPI networks of four species, *S. cerevisiae* (yeast), *D. melanogaster* (fly), *C. elegans* (worm), and *H. sapiens* (human). The PPI network data, obtained from BioGRID [21], have been used in recent studies [110, 163]. For each species, four PPI networks are created that contain the following protein interaction types and confidence levels: all physical interactions supported by at least one publication (PHY<sub>1</sub>) or at least two publications (PHY<sub>2</sub>), as well as only yeast two-hybrid physical interactions supported by at least one publication (Y2H<sub>1</sub>) or at least two publications (Y2H<sub>2</sub>) (Supplementary Table A.1). Just as was done in the existing studies, we also remove the fly and worm networks from the PHY<sub>2</sub> and Y2H<sub>2</sub> network sets, because these networks are too small and sparse (53-331 nodes and 33-260 edges), resulting in the PHY<sub>2</sub> and Y2H<sub>2</sub> network sets containing only two networks each. The four network sets have unknown true node mapping, and thus we cannot evaluate node correctness. However, we use alternative measures of alignment quality that are based on Gene Ontology annotations (Section 2.2.3.2).

**Gene Ontology (GO) annotations.** For alignment quality measures (Section 2.2.3) that rely on GO annotations of proteins [159], we use experimentally obtained GO annotations from the GO database from January 2016.

**Protein sequences.** When NA methods use protein sequence information to produce an alignment (Section 2.2.2), we use BLAST protein sequence similarities as captured by E-values [175]. The sequence data were acquired from the NCBI website (<https://www.ncbi.nlm.nih.gov/>).

### 2.2.2 Network alignment methods that we evaluate

We study GHOST, MAGNA++, WAVE, and L-GRAAL PNA methods, and IsoRankN, BEAMS, multiMAGNA++, and ConvexAlign MNA methods.

**PNA methods.** Most NA methods are *two-stage* aligners: first, they calculate the similarities (based on network topology and, optionally, protein sequences) between nodes of the compared networks, and second, they use an alignment strategy to find high scoring alignments with respect to the total similarity over all aligned nodes. GHOST is a two-stage PNA method (Supplementary Section A.1.1). An issue with two-stage methods is that while they find high scoring alignments with respect to total node similarity (a.k.a. node conservation), they do not account for the amount of conserved edges during the alignment construction process. But the quality of an alignment is often measured in terms of edge conservation. To address this, MAGNA++ directly optimizes both edge and node conservation *while* the alignment is constructed (Supplementary Section A.1.1). MAGNA++ is a *search-based* (rather than a two-stage) PNA method. Search-based aligners can directly optimize edge conservation or any other alignment quality measure. WAVE and L-GRAAL were proposed as two-stage (rather than search-based) PNA methods that, just as MAGNA++, optimize both node and (weighted) edge conservation (Supplementary Section A.1.1).

**MNA methods.** IsoRankN, BEAMS, and ConvexAlign are two-stage MNA methods. IsoRankN optimizes node conservation. BEAMS and ConvexAlign optimize both node and edge conservation (Supplementary Section A.1.1). On the other hand,

like MAGNA++, multiMAGNA++ is a search-based method that optimizes both edge and node conservation. IsoRankN and BEAMS produce many-to-many alignments. ConvexAlign and multiMAGNA++ produce one-to-one alignments.

**Aligning using network topology only versus using both topology and protein sequences.** In our analysis, for each method, we study the effect on output quality when (i) using only network topology while constructing alignments (T alignments) versus (ii) using both network topology and protein sequence information while constructing alignments (T+S alignments). For T alignments, we set method parameters to ignore any sequence information. All methods except BEAMS can produce T alignments and all methods can produce T+S alignments. For T+S alignments, we set method parameters to include sequence information. Supplementary Table A.2 shows the specific parameters that we use, and Supplementary Section A.1.1 justifies our parameter choices.

### 2.2.3 Alignment quality measures

Typical PNA methods produce alignments comprising node pairs and typical MNA methods produce alignments comprising node clusters. We introduce the term *aligned node group* to describe either an aligned node pair or an aligned node cluster. With this, we can represent a pairwise or multiple alignment as a set of aligned node groups. For formal definitions, see Supplementary Section A.1.2.

#### 2.2.3.1 Topological quality measures

A good NA method should produce aligned node groups that have internal consistency with respect to protein labels. If we know the true node mapping between the networks, we can let the labels be node names. We consider measures that rely on node names to be capturing topological quality (TQ) of an alignment. If we do not know the true node mapping, we let the labels be nodes' (i.e., proteins') GO

terms. We consider measures that rely on GO terms to be capturing functional quality (FQ) of an alignment; we discuss such measures in Section 2.2.3.2. We measure internal consistency of aligned protein groups in a *pairwise* alignment via precision, recall, and F-score of node correctness (P-NC, R-NC, and F-NC, respectively); these measures, introduced by [110], work for both one-to-one and many-to-many pairwise alignments (Supplementary Section A.1.2.1). We do this in a *multiple* alignment via adjusted multiple node correctness (NCV-MNC); this measure, introduced by [163], works for both one-to-one and many-to-many multiple alignments (Supplementary Section A.1.2.1).

Also, a good NA method should find a large amount of common network structure, i.e., produce high edge conservation. We measure edge conservation in a *pairwise* alignment via adjusted generalized  $S^3$  (NCV-GS<sup>3</sup>); this measure, introduced by [110], works for both one-to-one and many-to-many pairwise alignments (Supplementary Section A.1.2.1). We do this in a *multiple* alignment via adjusted cluster interaction quality (NCV-CIQ); this measure, introduced by [163], works for both one-to-one and many-to-many multiple alignments (Supplementary Section A.1.2.1).

Finally, for a good NA method, conserved edges should form large and dense (as opposed to small or isolated) conserved regions. We capture the notion of large and connected conserved network regions (for *both* pairwise and multiple alignments) via largest common connected subgraph (LCCS). This measure, recently extended from PNA [143] to MNA [163], works for both one-to-one and many-to-many alignments, and for both pairwise and multiple alignments (Supplementary Section A.1.2.1).

### 2.2.3.2 Functional quality measures

Per Section 2.2.3.1, a good alignment should have internally consistent aligned node groups. Instead of protein names as in Section 2.2.3.1, in this section we use GO terms as protein labels to measure internal consistency. Having aligned node groups

that are internally consistent with respect to GO terms is important for protein function prediction.

We measure internal node group consistency with respect to GO terms in two ways. First, we do so via mean normalized entropy (MNE); this measure, introduced by [99] (also, see [163] for formal definition), works for both one-to-one and many-to-many alignments, and for both pairwise and multiple alignments (Supplementary Section A.1.2.2). Second, we do so via an alternative popular measure, GO correctness (GC); this measure, recently extended from PNA [96] to MNA [163], works for both one-to-one and many-to-many alignments, and for both pairwise and multiple alignments (Supplementary Section A.1.2.2).

In addition to measuring internal node group consistency, we directly measure the accuracy of protein function prediction. That is, we first use a protein function prediction approach (Section 2.2.3.3) to predict protein-GO term associations, and then we compare the predicted associations to known protein-GO term associations to see how accurate the predicted associations are. We do so via precision, recall, and F-score measures (P-PF, R-PF, and F-PF, respectively); these measures work for both one-to-one and many-to-many alignments, and for both pairwise and multiple alignments (Supplementary Section A.1.2.2).

### 2.2.3.3 Protein function prediction approaches

Here, we discuss how we predict protein-GO term associations from the given alignment. We use a different protein function prediction approach for each alignment type. Therefore, below, first, we discuss an existing approach that we use to predict protein GO-term associations from pairwise alignments (approach 1). Second, we discuss an existing approach that we use to predict these associations from multiple alignments (approach 2). Third, since the existing approach for multiple alignments (approach 2) is very different from the existing approach for pairwise alignments

(approach 1), to make comparison between pairwise and multiple alignments (i.e., between PNA and MNA) more fair, we extend approach 1 for pairwise alignments into a new approach for multiple alignments (approach 3). As we show in Section 2.3.5.1, our new approach 3 in general improves upon the existing approach 2. So, we propose approach 3 as a new superior strategy for predicting protein-GO term associations from multiple alignments, which is another contribution of this study.

**Approach 1. Existing protein function prediction for pairwise alignments.**

Here, we predict protein GO-terms associations using a multi-step process proposed by [110]. For each protein  $v$  in the alignment that has at least one annotated GO term, and for each GO term  $g$ , first, we hide  $v$ 's true GO term(s). Second, we determine if the alignment is statistically significant with respect to  $g$ , i.e., if the number of aligned node pairs in which the aligned proteins share GO term  $g$  is significantly high ( $p$ -value below 0.05 according to the hypergeometric test; see [110] for details). Repeating this process for all nodes and GO terms results in set  $X$  of predicted protein-GO term associations.

**Approach 2. Existing protein function prediction for multiple alignments.**

Here, we predict protein GO-term associations using the approach of [54], as follows. For each protein  $v$  in the alignment that has at least one annotated GO term, and for each GO term  $g$ , first, we hide the protein's true GO term(s). Second, given that  $v$  belongs to aligned node group  $C$ , we measure the enrichment of  $C$  in  $g$  using the hypergeometric test. If  $C$  is significantly enriched in  $g$  ( $p$ -value below 0.05; see [163] for details), then we predict  $v$  to be associated with  $g$ . Repeating this process for all nodes and GO terms results in set  $X$  of predicted protein-GO term associations.

**Approach 3. New protein function prediction for multiple alignments.**

Here, we introduce a new approach to predict protein GO-term associations from a multiple alignment. First, for each node group  $C_i$  in the alignment,  $C_i$  is converted into a set of all possible  $\binom{|C_i|}{2}$  node pairs in the group. The union of all resulting node

pairs over all groups  $C_i$  forms the set  $F$  of all aligned node pairs. Second, for each protein  $v$  in the alignment that has at least one annotated GO term, and for each GO term  $g$ , we hide  $v$ 's true GO term(s). Third, we determine if the alignment is statistically significant with respect to  $g$ , i.e., if the number of aligned node pairs  $F$  in which the aligned proteins share GO term  $g$  is significantly high ( $p$ -value below 0.05 according to the hypergeometric test; see Supplementary Section A.1.2.3 for details). Repeating this process for all nodes and GO terms results in a set of predicted protein-GO term associations. Our proposed approach 3 is identical to approach 1 except for its first step of converting a multiple alignment into a set of aligned node pairs.

#### 2.2.3.4 Statistical significance of alignment quality scores

Since PNA and MNA methods result in different output types (as they produce alignments that differ in the number and sizes of aligned node groups for the same networks), to allow for as fair as possible comparison of the different NA methods, we do the following. For each NA method, each pair/set of aligned networks, and each alignment quality measure, we compute the statistical significance (i.e.,  $p$ -value) of the given alignment quality score. Then, we take the significance of each alignment quality score into consideration when comparing the NA methods (as explained in Section 2.2.4.3). We compute the  $p$ -value of a quality score of an alignment as described in Supplementary Section A.1.2.4.

#### 2.2.4 Evaluation framework

Given a network set, to fairly compare PNA and MNA, we compare the NA methods when aligning all possible pairs of networks in the set (pairwise evaluation, or PE, framework, Section 2.2.4.1), as well as when aligning all networks in the set at once (multiple evaluation, or ME, framework, Section 2.2.4.2). PNA is expected to perform better under the pairwise evaluation framework (which is native to PNA),



and MNA is expected to perform better under the multiple evaluation framework (which it is native to MNA).

#### 2.2.4.1 Pairwise evaluation framework

In the *PE framework*, given a network set, we compare NA methods using pairwise alignments of all possible pairs of networks in the set. Due to the various ways that a pairwise alignment of two networks can be created using PNA or MNA methods, we categorize the pairwise alignments into the following three categories. Specifically:

- We apply PNA to all possible network pairs, denoting the resulting alignments as the PE-P-P alignment category. Here, since all PNA methods are one-to-one, their pairwise alignments will be one-to-one.
- We apply MNA to all possible network pairs, denoting the resulting alignments as the PE-M-P alignment category. Here, if an MNA method is many-to-many, then its pairwise alignments will also be many-to-many. Otherwise, they will be one-to-one.
- We apply MNA to the whole network set and break the resulting multiple alignment into all possible pairwise alignments, as illustrated in Fig. 2.3(a). Specifically, given a multiple alignment spanning all of the networks (in our Fig. 2.3(a) illustration, three), we create a pairwise alignment for every pair of networks (i.e., three pairs) as follows: for the two networks in a given pair, we remove every node from the multiple alignment that is not a part of the two networks, which results in a pairwise alignment of the two networks. We denote the resulting pairwise alignments as the PE-M-M alignment category. Again, for a one-to-one or many-to-many MNA method, its pairwise alignments will also be one-to-one or many-to-many, respectively.

In the PE framework, we align all pairs of networks within each of the five analyzed network sets (Yeast+%LC, PHY<sub>1</sub>, PHY<sub>2</sub>, Y2H<sub>1</sub>, and Y2H<sub>2</sub>; Section 2.2.1). We evaluate using all alignment quality measures for pairwise alignments, namely F-NC, NCV-GS<sup>3</sup>, and LCCS TQ measures as well as MNE, GC, and F-PF FQ measures (Section 2.2.3).

#### 2.2.4.2 Multiple evaluation framework

In the *ME framework*, given a network set, we compare NA methods using the resulting multiple alignments of the set. Due to the various ways that a multiple alignment of a network set can be created, we categorize the multiple alignments in the following three categories. Specifically:

- We apply PNA to all possible network pairs and combine the resulting pairwise alignments into a multiple alignment that spans all networks in the set using a variation of a method introduced by [39], as illustrated in Figs. 2.3(b)-(c) and Supplementary Section A.1.3. In more detail, given pairwise alignments of all networks pairs in the set (in our Fig. 2.3(b)-(c) illustrations, three pairs of networks,  $(G_1, G_2)$ ,  $(G_2, G_3)$ , and  $(G_1, G_3)$ ), produced by PNA, we combine the pairwise alignments into a multiple alignment as follows. First, we select a “scaffold” network (in our illustration,  $G_2$ ). Second, we create a set of node groups consisting of the pairwise alignments between the scaffold network and the other networks (in our illustration,  $(G_1, G_2)$  and  $(G_2, G_3)$ ). Third, we merge node groups that have at least one node in common. This procedure yields a multiple alignment of all networks in the set. We denote the resulting alignment as the ME-P-P alignment category. Here, even though all PNA methods are one-to-one, their pairwise-combined-to-multiple alignments will be many-to-many.
- We apply MNA to all possible network pairs and combine the resulting pairwise alignments into a multiple alignment that spans all networks in the set using the same variation of the method introduced by [39] as above (Fig. 2.3(b)-(c) and Supplementary Section A.1.3), denoting the resulting alignment as the ME-M-P alignment category. Here, independent of whether an MNA method is one-to-one or many-to-many, its pairwise-combined-to-multiple alignments will be many-to-many.
- We apply MNA to the whole network set to align all networks at once, denoting the resulting alignment as the ME-M-M category. Here, if an MNA method is one-to-one, its direct multiple alignments will also be one-to-one. Otherwise, they will be many-to-many.

In the ME framework, we align each of the analyzed network sets that has more than two networks (Yeast%+LC, PHY<sub>1</sub>, and Y2H<sub>1</sub>; Section 2.2.1). We evaluate using all alignment quality measures for multiple alignments, namely NCV-MNC, NCV-CIQ, and LCCS TQ measures as well as MNE, GC, and F-PF FQ measures (Section 2.2.3).

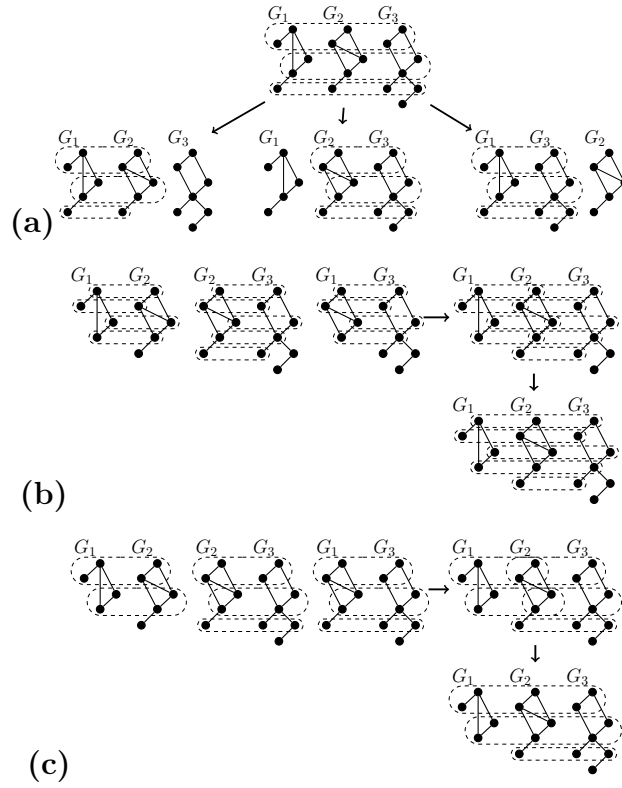


Figure 2.3. Illustration on a set of three networks ( $G_1$ ,  $G_2$ , and  $G_3$ ) of how we convert: **(a)** a multiple alignment to pairwise alignments, **(b)** one-to-one pairwise alignments to a multiple alignment, and **(c)** many-to-many pairwise alignments to a multiple alignment.

#### 2.2.4.3 Comparing the performance of network alignment methods

We compare two NA methods in terms of their alignment quality (i.e., accuracy) and running time.

In terms of alignment quality, given a network pair/set and an alignment quality measure (i.e., in a given *evaluation test*), we compare two NA methods as follows. Let  $x$  and  $y$  be the methods' respective alignment quality scores. If both  $x$  and  $y$  are significant ( $p$ -values below 0.001; Section 2.2.3.4) and are within 1% of each other ( $\frac{|x-y|}{(x+y)/2} < 0.01$ ), then the two methods are *tied*. They are also *tied* if both  $x$  and  $y$  are non-significant. If both  $x$  and  $y$  are significant and not tied, then the method with the best score is superior. If  $x$  is significant and  $y$  is not, then the method with score  $x$  is superior, and vice versa.

Given  $k$  network pairs/sets and  $l$  alignment quality measures, i.e., given  $k \times l$  evaluation tests, for each evaluation test, we rank all methods from the best one to the worst one, as follows. Given the methods' alignment quality scores, for methods with non-significant scores, we rank the methods last. For methods with significant scores, we perform the following procedure. If a given method has the best alignment quality score, then we give it rank 1 (as the 1<sup>st</sup> best method). We give the next best performing method rank 2, and so on. If a given method is tied with the next best performing method, then we rank both methods with the superior (i.e., lower) rank. The subsequent methods are ranked as if the previous methods were not tied. For example, if methods  $a$  and  $b$  are tied, they are both given rank 1, and if method  $c$  is not tied with method  $a$  or method  $b$ , then method  $c$  is given rank 3). We call this resulting rank for a given evaluation test an *evaluation test rank*. We calculate the *overall ranking* of an NA method by taking the mean of its ranks over all  $k \times l$  evaluation tests. To evaluate whether the overall rankings of two methods are significantly different from each other, we apply the one-tailed Wilcoxon signed-rank test on the  $k \times l$  evaluation test ranks of the two methods.

We also compare the NA methods with respect to their running times. Specifically, for each network pair/set, for each alignment category in the PE and ME frameworks, we give the fastest method rank 1, the second fastest method rank 2, and so on. Each method is restricted to use a maximum of 64 cores.

## 2.3 Results and discussion

In Section 2.3.1, we compare the quality of T alignments and T+S alignments. In Sections 2.3.3 and 2.3.4, we compare PNA against MNA in the PE and ME framework, respectively, in terms of TQ and FQ accuracy as well as running time. In Section 2.3.5, we compare PNA against MNA exclusively in terms protein function prediction accuracy, as the main goal of biological NA is to predict protein functions in one species from protein functions in another species, based on the species' network alignment.

### 2.3.1 Topology versus topology+sequence alignments

Network topology alone can be used to find good alignments of PPI networks [96]. But protein sequence information can be used to complement network topology in order to produce superior alignments [109]. Due to the complementarity of network topology and protein sequence information, we expect T+S alignments to have higher alignment quality than T alignments. In fact, we verify this. Namely, for each NA method, we compare the given method's T alignments to their corresponding T+S alignments, in terms of TQ and FQ measures, under the PE and ME frameworks (Fig. 2.4). We find the following.

For networks with known true node mapping, T+S alignments are superior to the corresponding T alignments in almost all cases. Note that as already recognized by [163], for these networks, i.e., for the Yeast+%LC network set, the superiority of T+S alignments over T alignments is not a surprising result. This is because

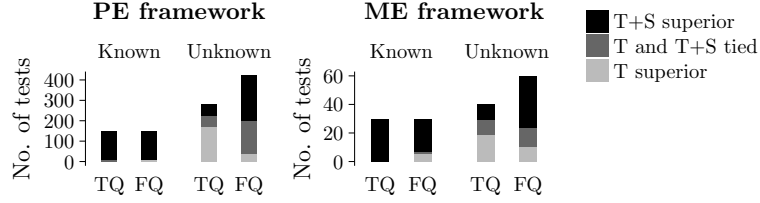


Figure 2.4. Comparison of the quality of T alignments versus the corresponding T+S alignments, under each of the PE and ME frameworks. Each bar shows the number of cases (here, a case refers to a combination of NA method, a network pair/set, and an alignment quality measure) in which the T alignment is superior, the T+S alignment is superior, or the two alignments are tied (i.e., within 1% of each other’s accuracy). The cases are separated into network pairs/sets with known true node mapping and network pairs/sets with unknown true node mapping.

this dataset contains networks that all have the same set of nodes. Consequently, it contains many inter-network pairs of nodes that are the same proteins. Sequence similarities of such matching node pairs are higher than those of other non-matching node pairs. These matching inter-network node pairs can likely form aligned node groups that have very high intra-group sequence similarity due to the node pairs containing identical proteins. This could explain the superiority of T+S alignments over T alignments for the set of networks with known node mapping.

Even for the sets of networks with unknown node mapping (PHY1, PHY2, Y2H1, Y2H2), whose networks contain different node sets, we still see that T+S alignments are overall superior to T alignments. Namely, only in terms of TQ, T alignments are somewhat superior to T+S alignments, but T+S alignments are still superior to or tied with the corresponding T alignments in just under a half of all cases. In terms of FQ, T+S alignments are superior to or tied with the T alignments in almost all evaluation tests.

So, we conclude that T+S alignments are overall superior to T alignments. Because of this, because T+S alignments are more relevant in the computational biology

domain, and because of space constraints, henceforth, we mainly analyze T+S alignments. Importantly, our findings for T+S alignments also hold for T alignments (Supplementary Fig. A.6).

Due to space constraints, for additional results on the similarity (overlap) of the alignments produced the different NA methods, which demonstrate that using protein sequence information overall yields alignment consistency between the different NA methods, see Supplementary Section A.1.4 and Supplementary Figs. A.1–A.3.

### 2.3.2 Method comparison: evaluation details

In Fig. 2.5, we compare PNA and MNA over all evaluation tests (where a test is a combination of a network pair/set and an alignment quality measure) for T+S alignments; analogous comparison for T alignments is shown in Supplementary Fig. A.6. In this section, we discuss how we evaluate and compare PNA and MNA. We show the results of the comparison in Section 2.3.3 for the PE evaluation framework and in Section 2.3.4 for the ME evaluation framework.

In all of Sections 2.3.2, 2.3.3, and 2.3.4, when we refer to an “NA method”, we mean the combination of a PNA or MNA method and an alignment category (Section 2.2.4). Namely, there are 12 NA methods in the PE framework (four PNA methods associated with the PE-P-P category and four MNA methods associated with each of the PE-M-M and PE-M-P categories) and 12 NA methods in the ME framework (four PNA methods associated with the ME-P-P category and four MNA methods associated with each of the ME-M-M and ME-M-P categories). We analyze the NA methods via three views, described below and visualized in Fig. 2.5:

Figure 2.5: Alignment category comparison results for each of the **PE** and **ME** frameworks over all evaluation tests for T+S alignments. The alignment categories (i.e., PE-P-P, etc.) are color-coded. **View I.** Overall ranking of the NA methods. The “Overall rank” column shows the rank of each method averaged over all evaluation tests, along with the corresponding standard deviation (in brackets). **View II.** Alternative view of ranking of the NA methods. Each pie chart shows the fraction of evaluation test ranks that fall into the 1–4, 5–8, and 9–12 rank bins out of all evaluation test ranks in the given alignment category. The pie charts are color-coded with respect to alignments of network pairs/sets with known and unknown node mapping, and TQ and FQ measures. **View III.** Overall ranking of an NA method versus its running time for the Y2H<sub>1</sub> network set. The size of each point visualizes the overall ranking of the corresponding method over all evaluation tests, corresponding to the “Overall rank” column in View I; the larger the point size, the better the method.





- **View I:** Overall ranking of the NA methods, as described in Section 2.2.4.3. Since there are 12 methods in a given (PE or ME) framework, the possible ranks range from 1 to 12. The lower the rank, the better the given method. The “ $p_1$ -value” column shows the statistical significance of the difference between the ranking of the 1<sup>st</sup> best ranked method and each other method. The “ $p_2$ -value” column shows the statistical significance of the difference between the ranking of the 2<sup>nd</sup> best ranked method and each other method. The “Non. sig. (fail)” column shows the fraction of all evaluation tests in which the alignment quality score is not statistically significant, and, in brackets, the fraction of evaluation tests in which the given NA method failed to produce an alignment.
- **View II:** Pie charts showing the fraction of evaluation test ranks that fall into the 1–4, 5–8, and 9–12 rank bins out of all evaluation test ranks in the given alignment category. For example, for the PE framework, in the PE-P-P alignment category, 56%, 26%, and 18% of the evaluation test ranks fall into ranks 1–4, 5–8, and 9–12, respectively, totaling to 100% of the evaluation test ranks in the PE-P-P alignment category. The pie charts allow us to compare the three alignment categories rather than individual NA methods in each category. The larger the pie chart for the better (lower) ranks, and the smaller the pie chart for the worse (higher) ranks, the better the alignment category. For example, in the PE framework, PE-P-P has the most evaluation tests ranked 1–4 and the fewest evaluation tests ranked 9–12, followed by PE-M-P, followed by PE-M-M. This implies that PE-P-P is superior to PE-M-P and PE-M-M.
- **View III:** Overall ranking of an NA method versus its running time, as described in Section 2.2.4.3. In order to allow for easier comparison between the different alignment categories, “Average” shows the average running times and average rankings of the methods in each alignment category.

### 2.3.3 Method comparison: results in the pairwise evaluation framework

We expect that under the PE framework, PNA will perform better than MNA. This is exactly what we observe. So, the most interesting and shocking results of this study do not originate from this section. Instead, they originate from Section 2.3.4 below, when comparing PNA and MNA in the ME framework.

Namely, in the PE framework, the overall ranking of the PNA methods (T+S alignments from the PE-P-P category) is generally better (lower) than the overall ranking of the MNA methods (T+S alignments from the PE-M-P and PE-M-M categories) (View I of Fig. 2.5). An exception is multiMAGNA++’s alignments from

the PE-M-P category (multiMAGNA++ directly applied to network pairs), whose overall ranking is also very good (low). This could be due to multiMAGNA++ being a one-to-one MNA method, which might have caused it to behave similarly as PNA methods (all of which are also one-to-one) when it is used to align only two networks. This is further supported by the fact that the only other considered one-to-one MNA method, ConvexAlign, and specifically its PE-M-P version, is also ranked better (lower) than the remaining two many-to-many MNA methods, IsoRankN and BEAMS. Nonetheless, ConvexAlign still has worse (higher) ranking than any PNA method (View I of Fig. 2.5).

Next, we break down the results into those for networks with known versus unknown node mapping, and also, into those for TQ versus FQ measures (View II of Fig. 2.5); additional, even more detailed results for the PE framework are shown in Supplementary File A.1. For networks with known mapping, we find that PNA performs better than MNA in terms of both TQ and FQ. For networks with unknown mapping, PNA performs better than MNA in terms of TQ, while in terms of FQ, the situation is not as clear.

Namely, for networks with unknown mapping and FQ, as can be seen in View II of Fig. 2.5, MNA falls into the best (lowest) ranks 1-4 in more of the evaluation tests than PNA. This implies that MNA is better than PNA. However, at the same time, MNA also falls into the worst (highest) ranks 9-12 in more of the evaluation tests than PNA. This implies that MNA is worse than PNA. Because we are interested in comparing the whole category of the considered PNA approaches against the whole category of the considered MNA approaches (per our discussion in Section 2.1.2), the above two results combined could be interpreted as MNA and PNA being comparable for networks with unknown mapping and FQ. On the other hand, for the same networks (with unknown mapping) and TQ, as well as for networks with known mapping and both TQ and FQ, PNA falls into the best ranks 1-4 in more of

the evaluation tests than MNA, and at the same time, PNA falls into the worst ranks 9-12 in *fewer* of the evaluation tests than MNA, which means that PNA is superior to MNA.

Another observation is as follows (Supplementary Tables A.4–A.7). For evaluation tests in which PNA is clearly superior in terms of method rankings to MNA (again, with the exception of multiMAGNA++’s PE-M-P version), which are tests excluding networks with unknown mapping and FQ, the best-ranked PNA method (MAGNA++ or WAVE) is significantly superior to the best-ranked MNA method (multiMAGNA++’s PE-M-M version, followed by all other MNA methods that are all similarly ranked), with  $p$ -values below  $1.8 \times 10^{-6}$ . On the other hand, for tests where it is unclear which of PNA and MNA is better, which are tests involving networks with unknown mapping or FQ, the best-ranked MNA method (ConvexAlign’s PE-M-P version) is only marginally better than the best-ranked PNA method (MAGNA++), with  $p$ -values between 0.048 and 0.332. This justifies referring to PNA and MNA as comparable for networks with unknown mapping and FQ, and to PNA as being superior in all other cases.

Next, we want to comment on the two MNA methods that perform well in at least some evaluation tests in the PE (*pairwise*) framework: multiMAGNA++ and ConvexAlign. Both of these methods produce one-to-one mappings, unlike the other two MNA methods, BEAMS and IsoRankN, which produce many-to-many mappings. Given that all PNA (*pairwise*) methods are also one-to-one, it might not be surprising that the two one-to-one MNA methods also perform well in the PE framework. This could be because the existing measures for *pairwise* alignment accuracy favor one-to-one mappings. However, we believe that it is not just the one-to-one aspect of multiMAGNA++ and ConvexAlign that is relevant. First, while multiMAGNA++ performs reasonably well in all tests (networks with both known and unknown node mappings, and both TQ and FQ), ConvexAlign performs poorly for

networks with known mapping or TQ but exceptionally well (marginally better than multiMAGNA++) for networks with unknown mapping and FQ. So, even though both methods are one-to-one, each has its unique (dis)advantages. Second, in Section 2.3.4, which evaluates the methods in the ME (*multiple*) framework, of the four MNA methods, it is again multiMAGNA++ and ConvexAlign that perform the best. This is despite the fact that the existing measures for *multiple* alignment accuracy do *not* necessarily favor one-to-one mappings, and some (especially FQ) actually *favor* many-to-many mappings.

A likely reason why ConvexAlign performs well only for networks with unknown node mapping and FQ is because its parameter values that were recommended and pre-set by its authors and that we use (Supplementary Section A.1.1) were determined via cross-validation, by optimizing FQ (GO term similarity of mapped nodes) in alignments of networks with unknown node mapping (PPI networks of mouse and human) [74]. Hence, ConvexAlign is semi-supervised, i.e., pre-trained to achieve high FQ scores, which makes it biased compared to the other considered NA methods, all of which are unsupervised.

**Accuracy versus running time.** The PNA methods are not only more accurate in general (as demonstrated above), but on average they are also at least somewhat if not much faster (View III of Fig. 2.5). In fact, no MNA method has both better running time and better ranking than any PNA method, while many PNA methods have both better running time and better ranking than every MNA method. Additional results where each method is restricted to use a single core are shown in Supplementary Fig. A.4.

### 2.3.4 Method comparison: results in the multiple evaluation framework

We expect that under the ME framework, MNA will perform better than PNA. Shockingly, we do not find this. Instead, our results reveal the opposite trends, which

match those observed under the PE framework. So, the most interesting results of this study originate from this section.

Namely, in the ME framework, the overall ranking of the PNA methods (T+S alignments from the ME-P-P category) is generally better (lower) than the overall ranking of the MNA methods' T+S alignments from the ME-M-M category, which in turn is generally better than the overall ranking of the MNA methods' T+S alignments from the ME-M-P category (View I of Fig. 2.5). Again, multiMAGNA++ is an exception: its alignments from the ME-M-P category (multiMAGNA++ first being applied to network pairs and then its pairwise alignments being combined into a multiple alignment) are ranked very good (low).

When we inspect the ranking of the methods in more detail (View II of Fig. 2.5), again, we find similar trends as in the PE framework. Namely, for networks with known mapping, we find that PNA performs better than MNA in terms of both TQ and FQ. For networks with unknown mapping, PNA performs better than MNA in terms of TQ. In terms of FQ, just as under the PE framework, MNA falls into the best (lowest) ranks in more of the evaluation tests than PNA, but at the same time, MNA also falls into the worst (highest) ranks in more of the evaluation tests than PNA. Additional, even more detailed results for the ME framework are shown in Supplementary File A.2.

Another result also applies to the ME framework: of the MNA methods, multiMAGNA++ and ConvexAlign perform better than BEAMS and IsoRankN, where multiMAGNA++ performs consistently well across all tests, and ConvexAlign performs extremely well only for networks with unknown node mapping and FQ (Supplementary Tables A.8–A.11).

Notice that under the ME framework, the best (PNA or MNA) methods are all one-to-one. Because all considered PNA methods are one-to-one, one might suspect that PNA may be overall better than MNA in the ME framework not because of

the “pairwise” part but simply because of the “one-to-one” part, possibly because one might suspect our evaluation measures in the ME framework to favor one-to-one methods. However, we argue that this is not the case, as follows.

First, if we could show that any existing one-to-one method performed worse than any existing many-to-many method in our ME framework, this would suffice to show that our ME framework does not favor one-to-one methods. While for our considered methods it is the case that one-to-one (PNA or MNA) methods are superior to many-to-many (MNA) methods, this could be simply because the considered one-to-one methods are more recent and thus more powerful than the considered many-to-many methods. Indeed, when we add to our ME evaluation an older (and thus inferior) one-to-one MNA method, GEDEVO-M [84], we find that this one-to-one method is outperformed by the considered many-to-many MNA methods (Supplementary Tables A.14–A.18). If one-to-one methods had some advantage over many-to-many methods in our ME framework, this would not have happened. So, a method’s performance in our ME framework does not seem to be directly related to it being one-to-one or many-to-many.

Second, by design, our evaluation measures do not favor one-to-one methods. Namely, recall that many of our evaluation measures were proposed by studies that introduced or analyzed many-to-many NA methods (Section 2.2.3). An example is one of our considered FQ measures, mean normalized entropy (MNE), which originates from the IsoRankN study [99], where IsoRankN is one of the considered many-to-many MNA methods. So, MNE is unlikely to favor one-to-one methods, as it was proposed in the many-to-many context. Actually, when we mirror the exact same MNE evaluation as in the IsoRankN study (see [99] for details) on the methods we consider here (rather than combine MNE with our other FQ measures as done so far in the paper), the considered one-to-one methods still perform well (i.e., the best of all considered one-to-one methods is still better than the best of all con-

sidered many-to-many methods) (Supplementary Tables A.12–A.13). That is, even a measure designed explicitly for many-to-many alignments still ranks one-to-one alignments better than many-to-many alignments. This additionally confirms that the overall superiority of the considered one-to-one (PNA or MNA) methods over the considered many-to-many (MNA) methods in the ME framework is likely because the one-to-one methods actually yield higher-quality alignments.

In summary, with these two findings in mind, it is more likely that the considered one-to-one methods perform better than the considered many-to-many methods in the ME framework because recent studies have focused on one-to-one alignments. Consequently, increased research in this area has likely led to better methodological advancements of one-to-one methods compared to many-to-many methods, explaining the one-to-one methods’ superior performance.

**Accuracy versus running time.** When we compare the overall rankings of the NA methods to their running times (View III of Fig. 2.5), again, we find similar trends as in the PE framework: the PNA methods are not only more accurate (as demonstrated above), but on average they are also faster.

Since the PNA methods must align every pair of networks in order to produce a multiple alignment, and since this results in a quadratically increasing running time with respect to the number of networks  $k$ , we ask whether there is some value of  $k$  at which PNA might become less efficient (i.e., slower) than MNA. Due to space constraints, we present this discussion in Supplementary Section A.1.5 and Supplementary Table A.3. Additional results where each method is restricted to use a single core are shown in Supplementary Fig. A.5.



### 2.3.5 Method comparison: focusing on accuracy of protein function prediction

#### 2.3.5.1 New function prediction approach under the multiple evaluation framework

Here, we focus on addressing a potential issue with the existing approach for protein function prediction for multiple alignments, which we have used up to this point. As discussed in Section 2.2.3.3, since the existing approach for multiple alignments (approach 2) is very different than the existing approach for pairwise alignments (approach 1), to make comparison between pairwise and multiple alignments (i.e., between PNA and MNA) more fair, we extend approach 1 for pairwise alignments into a new approach for multiple alignments (approach 3).

Then, we compare the new approach 3 against the existing approach 2, in hope that approach 3 will outperform approach 2. If so, in our subsequent analyses, we will use approach 3 for protein function prediction for multiple alignments. This way, comparing results of approaches 1 and 3 will be much more fair than comparing results of approaches 1 and 2. Consequently, we will be able to more fairly compare PNA against MNA.

Indeed, we find that our new approach 3 overall outperforms the existing approach 2 (Fig. 2.6 and Supplementary Fig. A.7). Specifically, approach 3 is overall comparable to approach 2 for networks with known node mapping (marginally inferior in terms of precision, marginally superior in terms of recall) and it is superior to approach 2 for networks with unknown node mapping (in terms of both precision and recall).

For networks with known node mapping, the number of predictions made by approach 3 is just 0.5%-5.8% larger than that made by approach 2, depending on the NA method, as shown in Supplementary Fig. A.7 (with the exception of ConvexAlign, which produces up to 54% more predictions under approach 3 than under approach 2). The slightly more predictions by approach 3 could explain its slightly lower

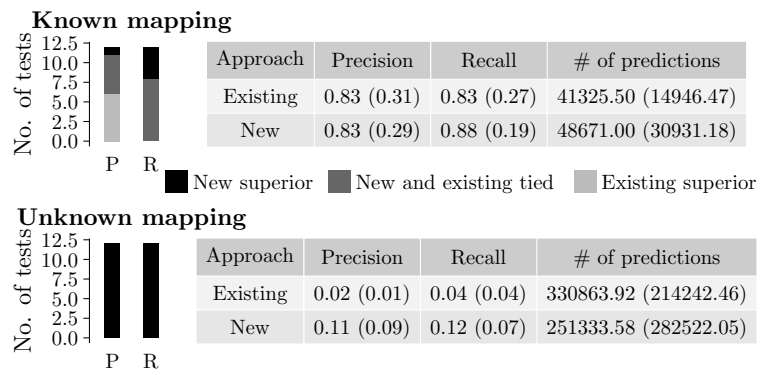


Figure 2.6. Comparison of protein function prediction accuracy between the new (approach 3) versus existing (approach 2) prediction approach for multiple alignments. Each bar on the left of the figure shows the number of cases (i.e., alignments) in which the new approach is superior, the existing approach is superior, or the two approaches are tied. Each table shows the precision, recall, and number of predictions averaged over all tests. In parentheses, we show standard deviations. The results are separated into network sets with known and unknown node mapping.

precision and slightly higher recall. But the differences in the number of predictions as well as accuracy of these two approaches on networks with known mapping are so minor (within 2%-5%) that we consider them as comparable.

For networks with unknown node mapping, the number of predictions made by approach 3 is 2%-72% smaller than the number of predictions made by approach 2, depending on the NA method (with exception of ConvexAlign and BEAMS, which in one instance produce 6% and 158% more predictions, respectively, under approach 3). While the fewer predictions under approach 3 could explain higher precision of approach 3 compared to approach 2, interestingly, approach 3 also results in higher recall than approach 2, despite the latter making more predictions (Fig. 6).

### 2.3.5.2 Protein function prediction under pairwise versus multiple evaluation frameworks

Next, we compare protein function prediction accuracy between the PE and ME frameworks, relying on approach 1 for pairwise alignments and on the fairly comparable approach 3 for multiple alignments. For analogous results where we use the existing approach 2 for the ME framework, see Supplementary Fig. A.10.

For both the network sets with known and unknown node mapping, the predictions under the PE framework have higher precision while the predictions under the ME framework have higher recall (Fig. 2.7 and Supplementary Fig. A.8. Note that here, higher precision and lower recall for the PE framework compared to the ME framework could be due to somewhat fewer predictions under the PE framework than under the ME framework. Also, note that for networks with known node mapping, both sets of predictions have impressively high precision and recall scores, so any difference in their scores (1%-6%) can be considered marginal. This is not the case for networks with unknown node mapping, where the scores are lower. In this case, the superiority of the PE framework’s precision over the ME framework’s precision (17%) is more pronounced than the superiority of the ME framework’s recall over the PE framework’s recall (8%). Additionally, achieving higher precision might be more preferred than achieving higher recall in the task of protein function prediction by experimental scientists who would potentially validate the predictions. Thus, we can argue that overall the PE framework (i.e., pairwise alignments) results in more accurate predictions than the ME framework (i.e., multiple alignments).

## 2.4 Conclusion

We introduce an evaluation framework for a fair comparison of PNA against MNA, in order to test the hypothesis that MNA can capture deeper biological insights, i.e.,

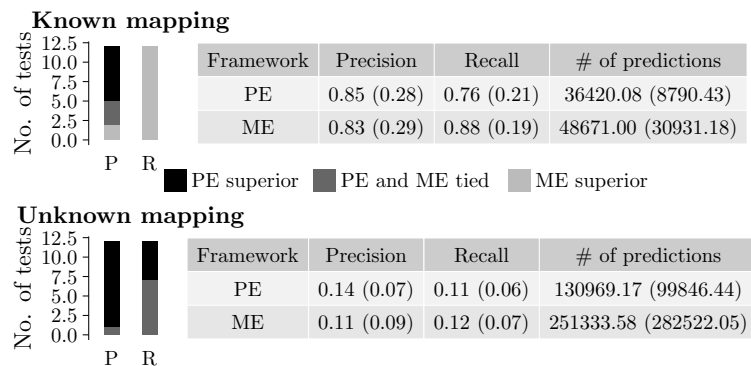


Figure 2.7. Comparison of protein function prediction accuracy under the the PE and ME frameworks. The figure can be interpreted the same way as Fig. 2.6. Here, we use new approach 3 for the ME framework.

produce higher-quality alignments, compared to PNA. We find that (i) the considered PNA methods produce pairwise alignments that are of higher quality than the corresponding pairwise alignments produced by the considered MNA methods, and (ii) the PNA methods produce multiple alignments that are of higher quality than the corresponding multiple alignments produced by the MNA methods. Also, using the pairwise alignments leads to higher protein function prediction accuracy than using the multiple alignments. Importantly, in addition to PNA being overall more accurate, it is also overall faster than MNA. This holds both both of T+S alignments and T alignments.

In our evaluation, i.e., thus far in the paper, we have aimed to compare the two categories of approaches, PNA and MNA, rather than to identify which specific NA method (whether of the PNA or MNA type) is the best, for reasons discussed in Section 2.1.2. Only here, we briefly comment on the performance of the best approach(es) in each category.

In the PNA category, most of the considered approaches, and especially MAGNA++, perform well consistently across the different scenarios (in both PE and ME framework, for both networks with known and unknown node mapping, and for both

TQ and FQ), with some exceptions (Supplementary Tables A.4–A.11). In the MNA category, only multiMAGNA++ works well consistently across all scenarios. Additionally, ConvexAlign works well for FQ and networks with unknown node mapping.

However, no method is always the best (i.e., has an overall rank of 1 over all evaluation tests). Namely, while in both PE and ME frameworks several PNA methods and the multiMAGNA++ MNA method achieve very good (low) overall ranks in the 1-2 range for networks with known node mapping or TQ, for networks with unknown node mapping and FQ, overall ranks start at about 4 (Supplementary Tables A.4–A.11). That is, for networks with unknown mapping and FQ, even the best methods (ConvexAlign and multiMAGNA++) work well for some but not all networks or alignment quality measures. So, there seems to be a lot more room for improvement on how to better perform PNA or MNA to improve FQ (the quality of functional predictions) from networks with unknown mapping (PPI networks of different species). Fig. 2.7 further signals this, given low prediction accuracy under both the PE and ME frameworks.

Importantly, the best approaches in this study in terms of FQ are of the one-to-one type, which we hypothesize is because of heavier recent focus on and thus methodological advancements of such methods compared to those of the many-to-many type, per our discussion in Section 2.3.4. But one-to-one alignments cannot capture gene duplication events that exist in biological networks [33], which require existence of paralogs, i.e., a gene in one network being mapped to multiple genes in the same or another network. While many-to-many alignments can in theory capture these events, the considered many-to-many methods do not perform well in terms of FQ. So, developing better many-to-many methods might be a crucial future step in NA research.

Since we demonstrate in the ME framework that PNA can (by integrating pairwise alignments) produce multiple alignments that are superior to multiple alignments

produced by MNA, we believe that any new MNA methods should be compared not just to existing MNA methods but also to existing PNA methods using our evaluation framework, to properly judge the quality of alignments that they produce. Our suggestion is similar to that of [110], who evaluated local versus global NA (rather than PNA versus MNA) and concluded that any new NA method should be compared against existing local as well as global NA methods.

Moreover, in the ME framework, PNA can produce multiple alignments that are superior to multiple alignments produced by MNA even with the simple variation of the pairwise alignment integration strategy (i.e., scaffolding procedure) introduced by [39]. Any more sophisticated scaffolding procedure that might be developed in the future will yield even more superior PNA-based multiple alignments and consequently even further emphasize the superiority of PNA over MNA. In other words, for MNA to gain advantage over PNA, a drastic redesign of the current MNA algorithmic principles might be needed.

In summary, our current results suggest that perhaps it might be sufficient to focus on the faster PNA and integration of pairwise alignments into multiple ones rather than on the slower MNA. Of course, with development of newer approaches, the conclusions from this study might change. It is crucial that we (the NA community) gain in-depth understanding of practical implications of one-to-one versus many-to-many, pairwise versus multiple, local versus global, and other types of NA. This understanding is even more crucial given recent shift from traditional NA of static and homogeneous (single node type and single edge type) networks towards dynamic [165, 162, 7] or heterogeneous [123, 70] NA, as well as from data-uninformed (i.e., unsupervised) to data-driven (i.e., supervised) NA [68].

## CHAPTER 3

### HETEROGENEOUS NETWORK ALIGNMENT

The work in this chapter is discussed in the following paper:

- **Shawn Gu**, John Johnson, Fazle E. Faisal, and Tijana Milenković (2018), From homogeneous to heterogeneous network alignment via colored graphlets, Scientific Reports, 8, Article number: 12524. [70]

#### 3.1 Introduction

##### 3.1.1 Background and motivation

Recall that regardless of NA category, i.e., pairwise or multiple, existing methods fail to align functionally related proteins. One reason this may happen is that they treat all networks as homogeneous, i.e., containing nodes of one type and edges of one type; we refer to the alignment of homogeneous networks as HomNA. However, a network can have nodes or edges of more than one type (or color). For example, different biological entities, such as proteins, phenotypes, or drugs, can be modeled as nodes, and different types of interactions, such as protein-protein, phenotype-phenotype, drug-drug, protein-phenotype, protein-drug, or phenotype-drug associations can be modeled as edges. Analyzing such heterogeneous multi-node- or multi-edge-type network data can lead to deeper insights into cellular functioning compared to homogeneous network analyses [61]. So, there is a need for being able to perform heterogeneous NA (HetNA).

While an existing method called AlignPI [171] was claimed to align heterogeneous networks, it actually did not perform HetNA as we define it in this study. Namely,

AlignPI was simply used to align two networks of different types to one other (specifically, the human PPI network to the disease-disease association network). However, each of the two considered networks is homogeneous, and thus the networks were aligned in the homogeneous fashion. Another relevant existing method is Fuse [62], which works via data integration. As such, it might appear that Fuse deals with data of different types, i.e., heterogeneous networks. However, it does not. Namely, Fuse aligns homogeneous PPI networks of different species, where the data integration step refers to using information from all of the homogeneous networks to calculate similarities between their nodes. Then, an alignment is still produced in the homogeneous fashion. The remaining relevant existing method is multimodal network alignment [123], which does deal with a special case of the HetNA problem. Namely, it aligns multimodal networks, which are a special case of heterogeneous networks as we define them. A multimodal (also called multiplex) network contains edges of different types (or modes) between the same set of nodes. That is, it contains only a single node type (Fig. 1.3). However, in this study, we define a heterogeneous network as a network that can contain different node types or different edge types (or both), and thus, our definition of HetNA is more broad than that of multimodal network alignment. Importantly, since the multimodal network alignment approach was not published as of completion of our evaluation (i.e., it was available only on arXiv), the code implementing it was not available at the time. So, we were unable to consider this approach in this study.

### 3.1.2 Our contributions

As already noted, current HomNA methods aim to find alignments with high homogeneous node conservation (HomNC) and homogeneous edge conservation (HomEC). So, to generalize HomNA to HetNA, we generalize HomNC to heterogeneous node



conservation (HetNC) and HomEC to heterogeneous edge conservation (HetEC). We describe these modifications intuitively below and formally in Section 3.3.

**From homogeneous to heterogeneous NC.** First, we introduce relevant concepts in the homogeneous context. Intuitively, two nodes from different homogeneous networks are topologically similar if their extended neighborhoods are similar. This idea can be quantified with homogeneous graphlets (small – typically up to 5-node – connected subgraphs), which have been extensively studied in homogeneous network analysis [112, 174, 158, 82, 155, 51, 169, 152]. For each node, for each graphlet, one counts how many times the given node touches each node symmetry group, or node orbit, in the given graphlet (e.g., in a 3-node path, the nodes at the end of the path are symmetric to each other and are thus in the same orbit, but they are distinct from the node in the middle, which is thus in a separate orbit). These counts over all graphlets summarize the extended network neighborhood of the node into its *graphlet degree vector (GDV)*. Then, to compute topological similarity between two nodes, their GDVs are compared.

Second, when we have a heterogeneous (node- or edge-colored) network, we modify the above notion of topological similarity between nodes; now, two nodes from different networks are topologically similar if they are of the same color and if their extended neighborhoods are of similar color and network structure. To quantify this, we extend homogeneous graphlets into heterogeneous (or colored) graphlets, as follows. Given a heterogeneous network containing  $n$  nodes and  $c$  different node (or edge) colors, an exhaustive extension would track both which combinations of node (or edge) colors exist in a given graphlet as well as at which node (or edge) positions in the graphlet the colors occur. With such an approach, the computational complexity of the problem, namely both the enumeration of all possible heterogeneous graphlet types on up to  $n$  nodes (the space complexity) and counting of the heterogeneous graphlets in a network (the time complexity), would increase exponen-

tially with the number of colors [161]. Instead, we propose a more computationally efficient node-colored (or edge-colored) graphlet approach: we only track which combinations of node (or edge) colors exist in a given graphlet but not at which node (or edge) positions in the graphlet the colors occur (Fig. 3.1). Consequently, with our approach: 1) the number of possible colored graphlets and thus the computational space complexity is lower compared to the exhaustive approach, and 2) most importantly, the computational time complexity of counting colored graphlets in a heterogeneous network is the same as that of counting original graphlets in a homogeneous network, unlike with the exhaustive approach (Fig. 3.1). Given node- or edge-colored graphlets, analogous to the GDV of a node in a homogeneous network, we summarize the extended neighborhood of a node in a heterogeneous network with its *node-colored GDV (NCGDV)* or *edge-colored GDV (ECGDV)*. Then, we compute topological similarity between two nodes from heterogeneous networks by comparing the nodes' NCGDVs, ECGDVs, or both. Formal definitions of node-colored and edge-colored graphlets, as well as NCGDVs and ECGDVs, can be found in Section 3.3.

Note that in our evaluation, we consider networks that contain only different node types. As such, our considered data contain different edge types only implicitly, because edges between nodes of different types will by definition be of different types themselves. So, in our evaluation, we need to consider only node-colored graphlets and NCGDVs, but not edge-colored graphlets or ECGDVs. Yet, we propose, define, and provide software implementation for edge-colored graphlets and ECGDVs as well, because these can be used alone for alignment of multimodal networks or combined with node-colored graphlets and NCGDVs for alignment of heterogeneous networks such as those in Fig. 1.3.

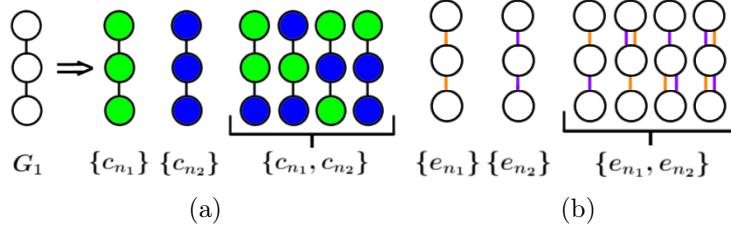


Figure 3.1. Illustration of (a) node-colored and (b) edge-colored graphlets. (a) With the exhaustive approach for enumerating all possible heterogeneous graphlets corresponding to homogeneous graphlet  $G_1$ , i.e., a 3-node path, given two colors, there would be six heterogeneous graphlets, each accounting for both which colors are present in the graphlet and which node position has which color. On the other hand, with our approach, there are three possible colored graphlets, denoted by  $\{c_{n_1}\}$ ,  $\{c_{n_2}\}$ , and  $\{c_{n_1}, c_{n_2}\}$ , each accounting only for which colors are present in the graphlet, ignoring the node-specific color information. Consequently, with our approach, the last four graphlets on the right of the arrow, which all have the same two colors present in them, are treated as the same heterogeneous graphlet. We design our approach in this way primarily to reduce the time complexity of counting heterogeneous graphlets in a network (but consequently, we also reduce the space complexity compared to the exhaustive approach). Namely, with our approach, the computational time complexity of searching for a given colored graphlet in a heterogeneous network remains the same as that of searching for its homogeneous equivalent. This is because the former involves: 1) counting in the heterogeneous network all graphlets, independent of their colors (which is the same as counting homogeneous graphlets in the network), and 2) for each of the homogeneous graphlets found in the network, simply determining which node colors appear in it and thus which node-colored graphlet the non-colored graphlet corresponds to. Step 1 is the time consuming part of the node-colored graphlet counting process, unlike step 2, which is trivial (can be done in constant time). (b) We develop a similar approach for edge-colored graphlets.

The software implementing node-colored and edge-colored graphlet counting is available upon request. We also provide an intuitive graphical user interface (GUI) for easy use by domain scientists.

**From homogeneous to heterogeneous EC.** In HomNA,  $S^3$  (Fig. 3.2) is a state-of-the-art EC measure [164, 103]. To explain  $S^3$ , first, we need to define a conserved edge. Intuitively, given two nodes in one network, and given their aligned counterparts in another network, the alignment is said to conserve an edge (i.e., form a conserved edge) if the two nodes are connected in the first network and the aligned counterparts are connected in the other network. Otherwise, if only the two nodes in the first network are connected or only their aligned counterparts in the other network are connected, but not both, the alignment is said to not conserve an edge (i.e., form a non-conserved edge). Formal definitions of conserved and non-conserved edges can be found in Section 3.3. Then,  $S^3$  is defined the ratio of the number of conserved edges to the number of both conserved and non-conserved edges. Intuitively,  $S^3$  rewards an alignment whenever it aligns an edge in one network to an edge in the other network and penalizes it whenever it aligns an edge in one network to a non-edge in the other network (or vice versa).

We extend  $S^3$  into a new measure of heterogeneous EC. In particular, we redefine what a conserved edge means, by accounting for colors of its aligned end nodes. Specifically, given a conserved edge consisting of nodes  $u$  and  $v$  in one network, and the corresponding aligned nodes  $u'$  and  $v'$ , respectively, in the other network, if both  $u$  and  $u'$  have the same color and  $v$  and  $v'$  have the same color, then the edge is fully conserved. Instead, if either  $u$  and  $u'$  have the same color or  $v$  and  $v'$  have the same color, but not both, then the edge is partially conserved, i.e., its contribution to the heterogeneous  $S^3$  score is penalized. If neither  $u$  and  $u'$  have the same color nor  $v$  and  $v'$  have the same color, then the edge is even less conserved than in the previous case, i.e., its contribution to the heterogeneous  $S^3$  score is penalized even more. Finally, if

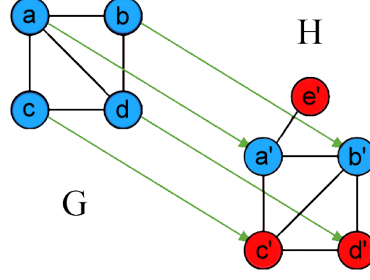


Figure 3.2. Illustration of HomEC and HetEC for an alignment between networks  $G$  and  $H$ . Arrows represent one possible alignment (mapping) between the networks, i.e., their nodes. Note that this node mapping is not the best alignment possible with respect to HomEC, but we use it to illustrate the concepts involved. In the homogeneous case (i.e., if all nodes were of the same color), there exist four conserved edges: the one formed by  $(a, a')$  and  $(a', a')$  – because  $a$  is aligned to  $a'$ ,  $b$  is aligned to  $b'$ , and an edge exists both between  $a$  and  $b$  as well as between  $a'$  and  $b'$ ; the one formed by  $(a, c)$  and  $(a', c')$ ; the one formed by  $(c, d)$  and  $(c', d')$ ; and the one formed by  $(b, d)$  and  $(b', d')$ . On the other hand,  $(a, d)$  and  $(a', d')$  form a non-conserved edge, because while  $a$  is aligned to  $a'$  and  $d$  is aligned to  $d'$ , there is an edge between  $a$  and  $d$  but not between  $a'$  and  $d'$ . For a similar reason,  $(b, c)$  and  $(b', c')$  form another non-conserved edge. So, given the existence of four conserved edges and two non-conserved edges, homogeneous  $S^3$  is  $\frac{\# \text{ conserved edges}}{(\# \text{ conserved edges} + \# \text{ non-conserved edges})} = 4/(4 + 2) = 0.67$ . In the heterogeneous case, for an edge to be conserved, the homogeneous condition is still required. However, we also account for colors of the aligned end nodes of a conserved edge and penalize for color mismatches. Specifically,  $(a, b)$  and  $(a', b')$  are counted as a fully conserved edge (with conservation weight of 1), because in addition to the fact that this edge is conserved in the homogeneous case,  $a$  has the same color as  $a'$ , and  $b$  has the same color as  $b'$ .  $(a, c)$  and  $(a', c')$  are counted as a less conserved edge (with conservation weight of  $\frac{2}{3}$ ), because while  $a$  and  $a'$  have the same color,  $c$  and  $c'$  do not. Similarly,  $(b, d)$  and  $(b', d')$  form a partly conserved edge with conservation weight of  $\frac{2}{3}$ .  $(c, d)$  and  $(c', d')$  are counted as an even less conserved edge (with conservation weight of  $\frac{1}{3}$ ) because neither  $c$  and  $c'$  nor  $d$  and  $d'$  have the same color. Finally,  $(a, d)$  and  $(a', d')$  form a non-conserved edge, just as in the homogeneous case. Given the total edge conservation of  $1 + \frac{2}{3} + \frac{2}{3} + \frac{1}{3} = \frac{8}{3}$  and two non-conserved edges (the same ones as in the homogeneous case), heterogeneous  $S^3$  uses the same formula as  $S^3$  and is  $\frac{\frac{8}{3}}{(\frac{8}{3} + 2)} = 0.57$ .

the edge is non-conserved, we treat it the same as in the homogeneous case. In this way, our new heterogeneous  $S^3$  measure favors both conserving edges and conserving edges whose aligned end nodes match in color.

**From homogeneous to heterogeneous NA.** We modify existing HomNA methods WAVE, MAGNA++, and SANA to perform HetNA by optimizing our new HetNC and HetEC measures (instead of their original HomNC and HomEC measures) with these methods’ ASs. We choose WAVE and MAGNA++ because they rose to the top in the study by Meng *et al.*, 2016 [110], which is a recent comprehensive evaluation of 10 HomNA methods. Since then, SANA appeared and was promising. So, we include SANA into this study as well. We modify all three methods and evaluate their new heterogeneous versions as described below. Detailed descriptions of these methods and their heterogeneous modifications can be found in Section 3.3.

### 3.2 Results and discussion

First, we describe our evaluation framework, specifically data that we use, networks that we align, and parameters of the three considered NA methods. Second, we compare HomNA and HetNA. That is, we compare each of homogeneous WAVE, MAGNA++, and SANA to its heterogeneous counterpart. Recall that there currently exist no HetNA methods, and thus, we cannot compare heterogeneous WAVE, MAGNA++, or SANA to any other hetNA method except to each other. In more detail, we evaluate: 1) the effect of HetNC, i.e., whether using more node colors increases alignment quality (and especially whether using two or more colors, i.e., HetNA, is superior to using a single color, i.e., HomNA), 2) the effect of HetEC, i.e., whether using heterogeneous  $S^3$  over homogeneous  $S^3$  increases alignment quality, and 3) the effect of the alignment method, i.e., which of our three new HetNA methods performs the best with respect to accuracy and running time.

### 3.2.1 Evaluation

We perform three evaluation tests corresponding to three sets of networks: 1) synthetic networks with up to four artificially imposed node colors, 2) homogeneous human PPI networks that have up to four node colors imposed according to proteins' involvement in a combination of aging, cancer, and Alzheimer disease (AD), and 3) heterogeneous human protein-GO networks, where the two node colors correspond to proteins and their Gene Ontology (GO) terms, and edges exist between proteins, between proteins and GO terms, and between GO terms. Note that while we evaluate WAVE and SANA in all three tests, due to MAGNA++'s computational complexity, we evaluate MAGNA++ only in the first test on the smaller synthetic networks but not in the remaining two tests on the larger PPI or protein-GO networks. We align each of the above networks to its noisy versions. Details are as follows.

**Synthetic networks.** We form synthetic networks using two random graph generators, namely: 1) geometric random graphs [133] (GEO) and 2) scale-free networks [10] (SF). The two models have distinct network topologies [113], which enables us to test the robustness of our results to the choice of random graph model. We form five random network instances per model and average results over them to account for the stochastic nature of the models. We set all model network instances to the same size of 1,000 nodes and 6,000 edges. Since the existing random graph generators are not designed to produce heterogeneous networks, we simply randomly assign each node a color out of  $k$  possible colors, where there are approximately  $1000/k$  nodes of each color. We vary  $k$  from one to four. That is, for each synthetic network, we form heterogeneous versions with one, two, three, and four colors.

**Human PPI networks.** We obtain the human PPI network data from BioGRID [21]. We consider two types of PPIs: only affinity capture coupled to mass spectrometry (APMS) and only two-hybrid (Y2H). Sizes of the resulting networks are shown in Table 3.1.

TABLE 3.1

NUMBER OF NODES AND EDGES IN THE TWO CONSIDERED PPI  
NETWORKS

Network	# of nodes	# of edges
APMS	11,450	92,257
Y2H	10,317	41,925

We impose node colors onto each PPI network based on the proteins' involvement in a combination of aging, cancer, and Alzheimer's disease (AD). We obtain a list of sequence-based (Seq) human aging-related genes from GenAge [38] and a list of gene expression-based (Expr) human aging-related genes from the study by Berchtold et al., 2008 [14]. We obtain a list of genes related in cancer from COSMIC [9]. We obtain a list of human genes related to AD from Simpson et al., 2011 [151].

We use these data to impose colors onto nodes in each of the two PPI networks (as well as their noisy counterparts; see below). For a given network, we use sequence-based aging- and cancer-related data to form four different colored versions of the network, as follows:

- In the 1-colored network, we treat all the nodes the same, meaning they have the same color.
- In the 2-colored network, we use the aging-related data to color nodes as "aging-related". Otherwise, they are "non-aging-related". This gives us 270 "aging-related" and 10,047 "non-aging-related" nodes.
- In the 3-colored network, we use aging- and cancer-related data. If a node is present in the aging-related data, we color it "aging-related". If a node is absent there but present in the cancer-related data, we color it as "cancer only". If a node is absent from both, we color it as "non-aging-related and non-cancer". In this way, we have 270 "aging-related", 405 "cancer only", and 9,642 "non-aging-related and non-cancer" nodes.
- In the 4-colored network, we use the same scheme as the 3-colored network, except if a node is present in both data sets, we color it as "both aging-related



and cancer". This gives us 203 "aging-related", 405 "cancer only", 67 "both aging-related and cancer", and 9,642 "non-aging-related and non-cancer" nodes.

To test the robustness of the choice of node color data above, we vary the underlying data. Now, for each of the two PPI network types, we use expression-based aging- and AD-related data to form four colored versions of the given network, as follows:

- In the 1-colored network, we treat all the nodes the same, meaning they have the same color.
- In the 2-colored network, we use the aging-related data to color nodes as "aging-related". Otherwise, they are "non-aging-related". This gives us 2,889 "aging-related" and 7,428 "non-aging-related" nodes.
- In the 3-colored network, we use aging- and AD-related data. If a node is present in the aging-related data, we color it "aging-related". If a node is absent there but present in the AD-related data, we color it as "AD only". If a node is absent from both, we color it as "non-aging-related and non-AD". In this way, we have 2,889 "aging-related", 356 "AD only", and 7,072 "non-aging-related and non-AD" nodes.
- In the 4-colored network, we use the same scheme as the 3-colored network, except if a node is present in both data sets, we color it as "both aging-related and AD". This gives us 2,232 "aging-related", 356 "AD only", 657 "both aging-related and AD", and 7,072 "non-aging-related and non-AD" nodes.

**Human protein-GO networks.** A heterogeneous protein-GO network has two types of nodes: protein and GO term [8], and three types of edges: 1) PPI, 2) protein-GO association, and 3) GO-GO semantic similarity. The PPI data are the same two types of PPI networks as before (APMS and Y2H), protein-GO associations are obtained from the Gene Ontology Consortium [8] based on experimental evidence codes, and GO-GO semantic similarities are computed as follows. We compute semantic similarity between all GOs that annotate at least one protein in the given considered PPI network. We use Lin method [106] to compute the semantic similarity. We form edges between GOs using semantic similarity threshold of 0.7, because the density of the resulting GO-GO network approximately matches the density of the corresponding PPI network. Considering APMS PPIs only and Y2H PPIs only,

we form two heterogeneous protein-GO networks for human, whose sizes are shown in Tables 3.2 and 3.3.

TABLE 3.2  
NUMBER OF NODES IN THE TWO CONSIDERED  
HETEROGENEOUS PROTEIN-GO NETWORKS

Network	Node type		
	# of proteins	# of GO terms	# of all nodes combined
APMS	11,450	5,558	17,008
Y2H	10,317	5,554	15,871

**Creating noisy counterparts of a synthetic, PPI, or protein-GO network.**

Given an original network  $G$ , we construct its noisy counterparts as follows. Considering a noise level of  $x\%$ , we randomly choose  $x\%$  of the edges and remove them from the original network, and then we randomly choose the same number of node pairs that are disconnected in the original network and add edges between them. That is, we randomly rewired  $x\%$  of the edges in the original network. Each noisy network has the same number of nodes and edges as the original network. For each considered original network, we use the following noise levels: 0%, 10%, 25%, 50%, 75%, and 100%. We construct multiple instances of noisy networks at each level to account for the randomness in edge rewiring; then, we average results (i.e., alignment quality) over the multiple runs. For WAVE and SANA, we use at least three instances.

TABLE 3.3

NUMBER OF EDGES IN THE TWO CONSIDERED  
HETEROGENEOUS PROTEIN-GO NETWORKS

Network	Edge type			
	# of PPIs	# of protein-GO associations	# of GO-GO semantic similarities	# of all edges combined
APMS	92,257	24,854	48,731	165,842
Y2H	41,925	24,473	48,873	115,271

For MAGNA++, we only use one instance due to MAGNA++’s high computation complexity.

**Measuring alignment quality.** Since we align an original network to its noisy counterpart, we know the true node mapping between the aligned networks (of course, this mapping is hidden from each NA method when it is asked to produce an alignment). Therefore, we evaluate the quality of the given network by measuring its node correctness, which quantifies how well the alignment matches the true node mapping. Formally, node correctness is the percentage of node pairs from the given alignment that are present in the true node mapping.

### 3.2.2 Comparison of homogeneous and heterogeneous network alignment

We need to define our considered evaluation scenarios. HomNA uses HomNC and HomEC, and we call this scenario HomNC-HomEC. For HetNA, if HetNC is used with HomEC, we call this scenario HetNC-HomEC; if HomNC is used with HetEC, we call this scenario HomNC-HetEC; and if HetNC is used with HetEC, we call this scenario HetNC-HetEC. Note that while MAGNA++ and SANA can optimize both NC and EC because they are search algorithms, WAVE only optimizes NC and it

cannot directly optimize EC, because it is a seed-and-extend algorithm. Hence, while we can evaluate MAGNA++ and SANA in all four of the above scenarios, i.e., while for these two methods we can study the effect on alignment quality of both HomNC versus HetNC and HomEC versus HetEC, for WAVE, we can only study the effect of HomNC versus HetNC.

First, we compare HomNC-HomEC to HetNC-HomEC, to study the effect of HetNC alone on alignment quality, while still considering HomEC in both cases. Then, we compare HetNC-HomEC to HetNC-HetEC to study the effect of HetEC on alignment quality after we have already accounted for HetNC. We perform all of these comparisons comprehensively, using all considered methods on all considered data sets, as described in Section 3.3. We also compare HomNC-HomEC to HomNC-HetEC to additionally study the effect of HetEC on alignment quality without first accounting for HetNC. Here, we perform only several case study comparisons out of all possible comparisons, due to the already comprehensive comparison experiments mentioned above.

**The effect of HetNC.** In terms of accuracy, we expect that for a given noise level, HetNA (i.e., HetNC-HomEC or HetNC-HetEC – two or more node colors) should improve alignment quality over HomNA (i.e., HomNC-HomEC – one node color). Also, we expect that the more colors are used, the better the alignment quality should be, since more information is used in the process of producing the alignment. In addition, we predict that using more colors will make the given method more robust to noise, meaning that we should see a slower decrease in alignment quality as noise increases, compared to using fewer colors. However, alignment quality should be low at the highest noise levels regardless of how many colors we use, since we are essentially aligning two networks with almost random topologies compared to each other. Indeed, we observe these exact trends (Figs. 3.3, 3.4, 3.5, 3.6). Note that the few observed ties occur typically at the lower (0% and 10%) noise levels, which makes

sense because in such cases network similarity can be captured reliably, meaning that all methods perform well.

In terms of time complexity, due to the way we count homogeneous as well as heterogeneous graphlets, time does not increase with more colors. Because of this, and because using more colors results in higher accuracy, we recommend using as many colors as needed. Note, however, that space complexity increases with the increase in the number of considered colors, because there are more possible graphlets; yet, the space complexity is practically feasible for a reasonable number of colors, such as four considered colors in this study (Section 3.3.2 – From homogeneous to heterogeneous NC).

**The effect of HetEC.** In terms of accuracy, we expect improvement of HetNC-HetEC over HetNC-HomEC, because while both HomEC and HetEC favor aligning nodes that conserve edges, unlike HomEC, HetEC also favors aligning nodes whose colors match. Indeed, this is generally what we observe (Fig. 3.7).

However, we see some ties between HetNC-HomEC and HetNC-HetEC. Also, while for MAGNA++ HetNC-HetEC noticeably improves alignment quality over HetNC-HomEC, for SANA, improvements of HetNC-HetEC over HetNC-HomEC are usually small (Figs. 3.4, 3.5, 3.6). (WAVE does not explicitly optimize EC, so we are unable to compare HomEC versus HetEC for WAVE). This could be due to SANA’s algorithm: it explores millions of alignments a second, and thus, it seems to already find high-scoring ones with just HetNC, without the need for HetEC.

For these reasons, we consider the HomNC-HetEC scenario, to properly gauge the true potential of HetEC in the task of HetNA, without any “bias” of also already using HetNC. Here, we analyze only two cases as a proof-of-concept of the effect of HetEC while still considering HomNC. Specifically, the two cases are MAGNA++ on geometric networks and SANA on APMS-Expr networks.

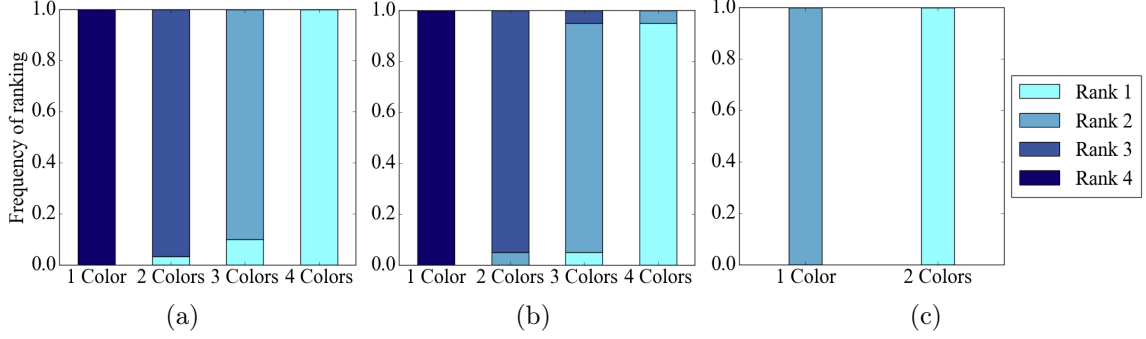


Figure 3.3. Summarized results regarding the effect of the **number of considered node colors** on alignment quality for (a) synthetic networks, (b) PPI networks, and (c) protein-GO networks. In panels (a) and (b), there are up to four considered node colors, while in panel (c), there are up to two considered node colors (see Section 3.2.1 for details). For each case (see below), we compare the different color levels (i.e., numbers of considered colors shown on  $x$ -axes) and rank them from the best (rank 1) to the worst (rank 4 in panels 1 and b, and rank 2 in panel c). Then, we compute the percentage or frequency of all cases (see below) in which the given color level is ranked as the first (rank 1), second (rank 2), third (rank 3), or fourth (rank 4) best among all considered color levels. In panel (a), there are 3 methods (WAVE, MAGNA++, SANA)  $\times$  2 networks (geometric, scale-free)  $\times$  5 noise levels (0%, 10%, 25%, 50%, 75%) = 30 cases. In panel (b), there are 2 methods (WAVE, SANA)  $\times$  4 networks (APMS-Expr, APMS-Seq, Y2H-Expr, Y2H-Seq)  $\times$  5 noise levels (0%, 10%, 25%, 50%, 75%) = 40 cases. In panel (c), there are 2 methods (WAVE, SANA)  $\times$  2 networks (protein-GO-APMS, protein-GO-Y2H)  $\times$  5 noise levels (0%, 10%, 25%, 50%, 75%) = 20 cases. Note that we analyzed an additional noise level (100%), but we leave the corresponding results from this summary figure, because at this level all cases are expected to result in the same (random) alignments (Section 3.2.1 – Creating noisy counterparts of a synthetic, PPI, or protein-GO network). Instead, we show the results for the noise level of 100% in the detailed figures (Figs. 3.4, 3.5, 3.6). Also, note that in this figure, for each case, we choose the best between HetNC-HomEC and HetNC-HetEC.

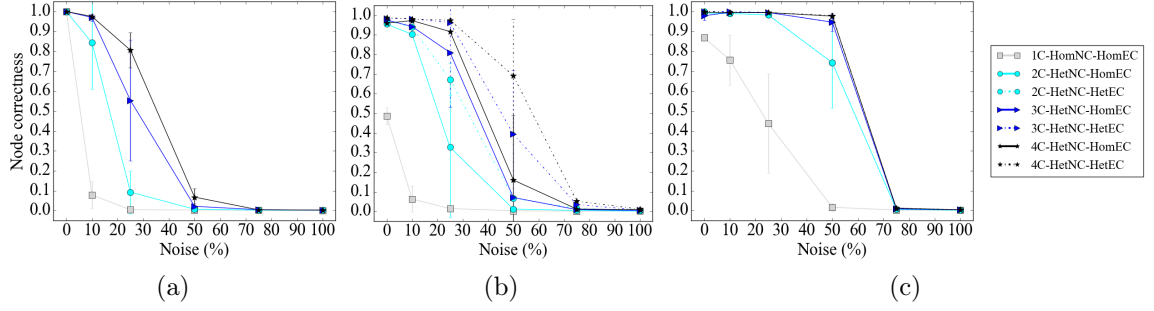


Figure 3.4. Detailed alignment quality results regarding the effect of the **number of node colors** on alignment quality as a function of noise level for **synthetic, specifically geometric**, networks, using (a) WAVE, (b) MAGNA++, and (c) SANA. Gray squares, light blue circles, dark blue triangles, and black stars indicate the aligned networks containing one, two, three, and four node colors, respectively. For two or more node colors, solid lines represent using HetNC-HomEC, and dashed lines represent using HetNC-HetEC. Equivalent results for the remaining synthetic, specifically scale-free, networks are shown in Supplementary Fig. B.2.

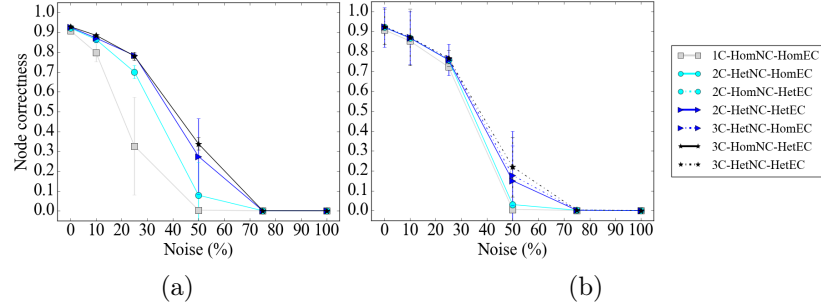


Figure 3.5. Detailed alignment quality results regarding the effect of the **number of node colors** on alignment quality as a function of noise level for **PPI, specifically APMS-Expr**, networks using (a) WAVE and (b) SANA. The figure can be interpreted in the same way as Fig. 3.4. Recall that for these larger networks, we have not run MAGNA++ due to its high computational complexity. Equivalent results for the remaining PPI, specifically APMS-Seq, Y2H-Expr, and Y2H-Seq, networks are shown in Supplementary Figs. B.4, B.5, and B.6.

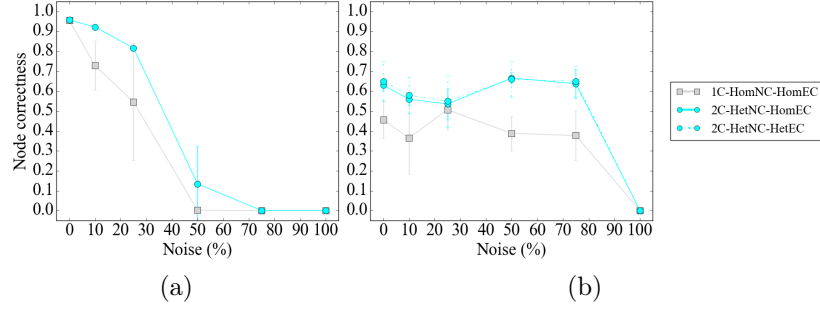


Figure 3.6. Detailed alignment quality results regarding the effect of the **number of node colors** on alignment quality as a function of noise level for **protein-GO**, specifically **protein-GO-APMS**, networks using (a) WAVE and (b) SANA. The figure can be interpreted in the same way as Fig. 3.4. Recall that for these larger networks, we have not run MAGNA++ due to its high computational complexity. Equivalent results for the remaining protein-GO, specifically protein-GO-Y2H, networks are shown in Supplementary Fig. B.8.

For these two cases, we evaluate all of HomNC-HomEC, HetNC-HomEC, HomNC-HetEC, and HetNC-HetEC scenarios (Fig. 3.8). First, for a given scenario, for a given noise level, we ask whether using more colors yields higher alignment quality, as expected. Indeed, this is what we observe. Second, for both MAGNA++ and SANA, HomNC-HetEC improves over HomNC-HomEC (i.e., over HomNA), though for SANA improvements are again small. However, using HetNC alone (HetNC-HomEC) improves alignment quality more than using HetEC alone (HomNC-HetEC). This might not be surprising, because HetNC favors aligning nodes of the same color that also have similar extended neighborhoods, while HetEC does not account for this extended neighborhood. As expected, HetNC-HetEC yields the best alignment quality of all four cases for all colors and all noise levels, except the highest (75% and 100%), as expected. For MAGNA++ on geometric networks, the improvements of HetNC-HetEC over the next best scenario (HetNC-HomEC) are large, while for



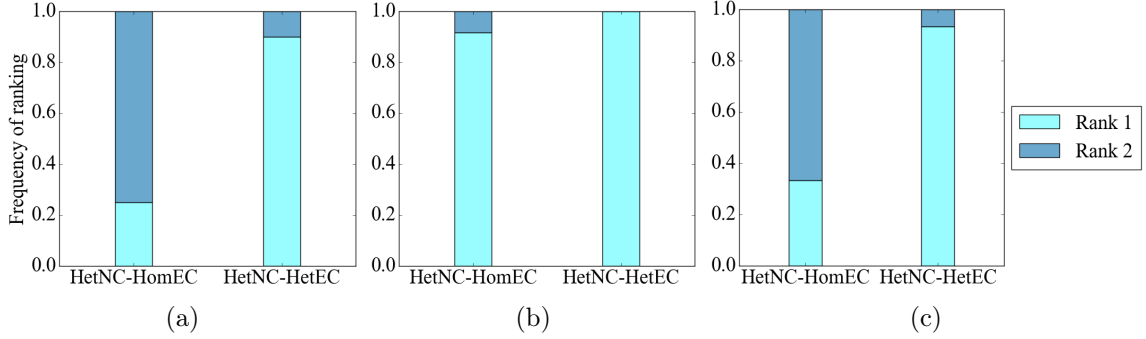


Figure 3.7. Summarized results regarding the effect of using HetEC over HomEC (both with HetNC) on alignment quality for (a) synthetic networks, (b) PPI networks, and (c) protein-GO networks. In all panels, there are two evaluation scenarios (HetNC-HomEC and HetNC-HetEC). For each case (see below), we compare the two considered evaluation scenarios and rank them from the best (rank 1) to the worst (rank 2). Then, we compute the percentage or frequency of all cases (see below) in which the given scenario is ranked as the first (rank 1) and second (rank 2) best among the considered scenarios. In panel (a), there are 2 methods (MAGNA++, SANA)  $\times$  2 networks (geometric, scale-free)  $\times$  5 noise levels (0, 10, 25, 50, 75)  $\times$  3 colors (1 color does not have a HetEC counterpart) = 60 cases. In panel (b), there is 1 method (SANA)  $\times$  4 networks (APMS-Expr, APMS-Seq, Y2H-Expr, Y2H-Seq)  $\times$  5 noise levels (as before)  $\times$  3 colors (as before) = 60 cases. In panel (c), there is 1 method (SANA)  $\times$  2 networks (protein-GO-APMS, protein-GO-Y2H)  $\times$  5 noise levels (as before)  $\times$  1 color (maximum 2 colors, but 1 color does not have a HetEC counterpart) = 10 cases. Note that we analyzed an additional noise level (100%), but we leave the corresponding results from this summary figure, because at this level all cases are expected to result in the same (random) alignments (Section 3.2.1 – Creating noise counterparts of a synthetic, PPI, or protein-GO network). Instead, we show the results for the noise level of 100% in the detailed figures (Figs. 3.4, 3.5, 3.6).

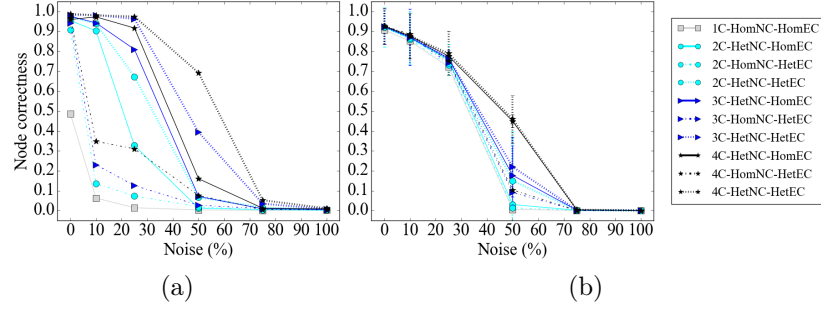


Figure 3.8. Detailed alignment quality results regarding the effect of **HomNC-HetEC** compared to HomNC-HomEC, HetNC-HomEC, and HetNC-HetEC on alignment quality for the two considered case study evaluation tests: (a) geometric networks using MAGNA++ and (b) APMS-Expr networks using SANA. The figure can be interpreted in the same way as Fig. 3.4, except that now solid lines represent HetNC-HomEC, short-long dotted lines represent HomNC-HetEC, and finely dotted lines represent HetNC-HetEC.

SANA on APMS-Expr networks, the improvements over the next best scenario (also HetNC-HomEC) are marginal.

In terms of time complexity, calculating heterogeneous  $S^3$  (i.e., HetEC) has the same complexity as calculating homogeneous  $S^3$  (i.e., HomEC), since counting the number of conserved and non-conserved edges in a heterogeneous network takes the same amount of time as in a homogeneous network. Specifically, checking if node colors match (Section 3.1.2 – From homogeneous to heterogeneous EC) to determine how much conserved an edge is takes constant time. Because of this, and because using both HetNC and HetEC results in the highest accuracy, we recommend using both HetNC and HetEC (i.e., HetNC-HetEC scenario).

**The effect of alignment method.** In terms of accuracy, regardless of noise level, WAVE and SANA generally outperform MAGNA++ (Fig. 3.9). WAVE and SANA have somewhat comparable performance (Fig. 3.9), in the following sense. For synthetic networks, the two are tied in 70% of all evaluation tests, WAVE is superior

to SANA in 10% of the tests, and SANA is superior to WAVE in 20% of the tests. For PPI networks, the two are tied in 50% of all evaluation tests, WAVE is superior to SANA in 15% of the tests, and SANA is superior to WAVE in 35% of the tests. For protein-GO networks, the two are tied in 0% of all evaluation tests, WAVE is superior to SANA in 50% of the tests, and SANA is superior to WAVE in 50% of the tests. Whenever WAVE is superior to SANA, it is typically for lower noise levels (up to 25%) (Fig. 3.10). Whenever SANA is superior to WAVE, it is typically for higher noise levels (above 25%) (Fig. 3.10). These trends for lower versus higher noise levels could be due WAVE's algorithm. At lower noise levels, the networks being aligned are still very similar to each other, so if two nodes are topologically similar, then it is likely that they should be aligned to each other. In this situation, WAVE would start with a good seed and thus be likely to produce a good alignment. At higher noise levels, the networks being aligned are dissimilar. So, two nodes may be topologically similar only because of the random rewiring of edges, but still be (erroneously) mapped to each other. In this situation, WAVE would start with a poor seed and likely lead to a poor alignment. Since SANA is not a seed-and-extend method, it avoids this issue and performs well even at higher noise levels.

In terms of time complexity, MAGNA++ is the slowest of the three methods (Fig. 3.10(a)), which is expected since it uses a genetic algorithm. Of WAVE and SANA, for synthetic networks, which happen to be the smallest of our considered networks, WAVE is faster than SANA (Fig. 3.10(a)). However, keep in mind that the execution time is a parameter in SANA. In that sense, it is possible to run SANA so that it is faster than any other method. However, in this case, SANA might not reach desired alignment quality. It might be possible to run SANA for as long as needed to always beat or at least tie WAVE in terms of alignment quality, but the amount of time would have to be determined empirically for every network pair being

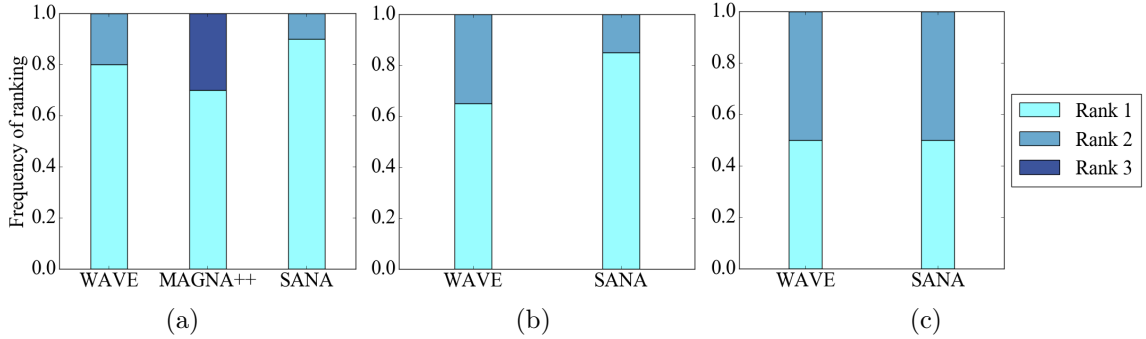


Figure 3.9. Summarized results regarding the effect of the **alignment method** on alignment quality for (a) synthetic networks, (b) PPI networks, and (c) protein-GO networks. In panel (a), there are three considered alignment methods (WAVE, MAGNA++, and SANA). In panels (b) and (c), there are two considered alignment methods (WAVE and SANA; MAGNA++ was not tested because of its high computational complexity). For each case (see below), we compare the alignment methods and rank the different methods from best (rank 1) to worst (rank 3 in panel (a), and rank 2 in panels (b) and (c)). Then, we compute the percentage of all cases in which the given method is ranked as the first (rank 1), second (rank 2), or third (rank 3) best among all considered methods. In panel (a), there are 2 networks (geometric, scale-free)  $\times$  5 noise levels (0, 10, 25, 50, 75) = 10 cases. In panel (b), there are 4 networks (APMS-Expr, APMS-Seq, Y2H-Expr, Y2H-Seq)  $\times$  5 noise levels (as above) = 20 cases. In panel (c), there are 2 networks (protein-GO-APMS, protein-GO-Y2H)  $\times$  5 noise levels (as above) = 10 cases. Note that we analyzed an additional noise level (100%), but we leave the corresponding results from this summary figure, because at this level all cases are expected to result in the same (random) alignments (Section 3.2.1 – Creating noise counterparts of a synthetic, PPI, or protein-GO network). Instead, we show the results for the noise level of 100% in the detailed figures (Figs. 3.4, 3.5, 3.6). Also, note that in this figure, we give each method the best case advantage. That is, we show results for the best of HetNC-HomEC and HetNC-HetEC, and also only for the maximum node color level (four colors in panels (a) and (b), and two colors in panel (c)). We do the latter because of all color levels, it is the maximum color level at which the given method performs the best, for each method. Nonetheless, the results remain qualitatively the same if we account for all considered colored levels.

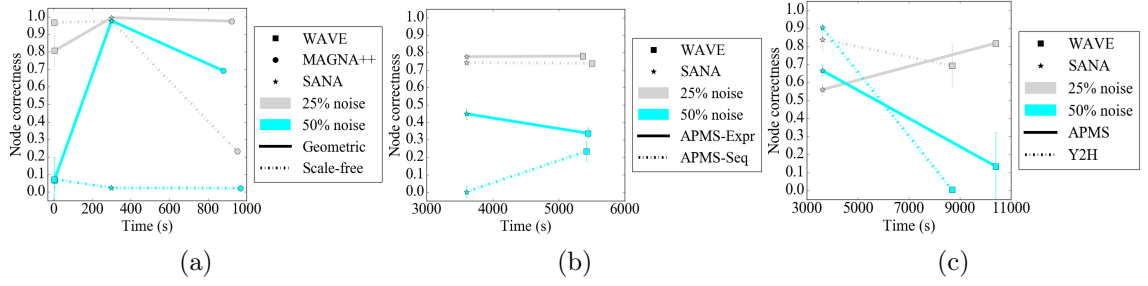


Figure 3.10. Summarized results comparing the **running times versus accuracy** of different methods for 25% and 50% noise on (a) synthetic, specifically geometric and scale-free, (b) PPI, specifically APMS-Expr and APMS-Seq, and (c) protein-GO, specifically APMS and Y2H, networks. The  $x$ -axis is the running time of the given method on the given network data at the given noise level, and the  $y$ -axis is the alignment quality score. Here we use different shapes to represent the different methods, different colors to represent the different noise levels, and solid or broken lines to represent the different network data. Lines are drawn between the different methods for the same noise level and network data, for easier comparison of the different methods. Detailed running time results for all other noise levels and network data are shown in Supplementary Figs. B.9–B.16.

aligned. For PPI and protein-GO networks, which happen to be the largest of our considered networks, SANA is faster than WAVE (Fig. 3.10(b)-(c)).

### 3.3 Methods

#### 3.3.1 Calculating node similarities, i.e., node conservation

Given the GDV for each node in a network, we form a matrix of GDVs over all nodes for each of the two networks being aligned. Then, we combine the two matrices row-wise and perform PCA on the large matrix of the networks' GDVs. We choose the first  $r$  principal components, where  $r$  is at least two and as small as possible such that the  $r$  components account for at least 90% of the variation in the data. Then, for every pair of nodes between the two networks, we calculate their cosine similarity based on the nodes' principal components and scale so the values are between 0 and 1.

**Method parameters.** WAVE does not have any parameters. We set MAGNA++'s parameters as follows: we use initial population size of 15,000 and 2,000 generations, which are the suggested values in the MAGNA++ documentation; we run MAGNA++ on 16 threads on all networks. We give equal weight to MAGNA++'s NC and EC measures, i.e., we set its `a` parameter to 0.5; using this value has been suggested by several studies [164, 110]. We set SANA's parameters as follows: we give equal weight to its NC and EC measures for fair comparability with MAGNA++, i.e., we set the following parameters: `s3` (corresponding to EC) to 1, `esim` (corresponding to NC) to 1, `simFile` to the name of the NC-based node similarity file, and `simFormat` to 1 (this tells SANA to read the similarity file such that each line has 3 columns: node1, node2, and the similarity between them). SANA also has a parameter for how long it should search for alignments. For synthetic networks, we run SANA for the default 5 minutes (`t` 5). For PPI and protein-GO networks,

we increase the  $\mathfrak{t}$  parameter to 60 minutes ( $\mathfrak{t} \ 60$ ), since these networks are larger and thus SANA needs more time to find a good alignment (which we have verified empirically in our evaluation).

### 3.3.2 From homogeneous to heterogeneous node conservation

Here we formalize the notion of heterogeneous (colored) graphlets. For ease of explanation, first, we define node-colored graphlets. Given  $k$  possible node colors from the set  $C_n = \{c_{n_1}, c_{n_2}, \dots, c_{n_k}\}$ ,  $S = 2^{C_n}$  is the set of all possible combinations of colors from  $C_n$ .  $S$  contains  $\binom{k}{0}$  elements with no color (i.e. the empty set),  $\binom{k}{1}$  elements with any one color, and in general  $\binom{k}{i}$  elements with any  $i$  colors. Therefore,  $S$  contains  $2^k$  elements. So  $S \setminus \emptyset$  is the set of all possible color combinations from  $C_n$  that excludes the empty set, which contains  $2^k - 1$  elements. Let  $b_n \in S \setminus \emptyset$ . Given a homogeneous graphlet  $G_i$ , a set of colors  $C_n$ , and some  $b_n$ , define a node-colored graphlet  $NCG_{i,b_n}$  to be the set of all distinct graphs that are isomorphic to  $G_i$ , such that for each graph, each node is colored with one of the colors from  $b_n$ , and also, each color from  $b_n$  has to be present in each such graph. Thus, given  $k$  node colors, there are  $2^k - 1$  possible node-colored graphlets.

As an illustration, let us assume that a heterogeneous network has nodes with two possible colors:  $c_{n_1}$  and  $c_{n_2}$ . These two node colors have 3 possible combinations:  $\{c_{n_1}\}$ ,  $\{c_{n_2}\}$ , and  $\{c_{n_1}, c_{n_2}\}$ . As a result, for each homogeneous graphlet  $G_i$ , there are three possible node colored graphlets (Fig. 3.1).

This definition of node-colored graphlets is more space efficient than the exhaustive approach is: given a heterogeneous network containing  $n$  nodes and  $k$  different colors, with the exhaustive approach, both the number of possible colored graphlets (the space complexity) and the time needed to count such graphlets in the network (the time complexity) increase exponentially with the number of colors. With our approach, however, 1) the number of possible colored graphlets is much smaller

(though still exponential in terms of the number of colors) compared to the exhaustive approach, and 2) the time complexity of counting colored graphlets in a heterogeneous network is the same as that of counting original graphlets in a homogeneous network, unlike with the exhaustive approach.

Regarding the space complexity of our colored graphlet approach, as an illustration, for two colors, with the exhaustive definition, there would be six node-colored graphlets for homogeneous graphlet  $G_1$ , a 3-node path, while with our approach there are only three of them. For three colors, with the exhaustive definition, there would be 18 node-colored graphlets for  $G_1$ , while with our approach there are only seven of them. Although even with our approach, the number of node-colored graphlets increases drastically with the increase of  $k$ , but this is not a major concern because in practice we may expect a relatively small value of  $k$ . For example, one can study a heterogeneous network whose nodes are proteins, functions, diseases, and drugs with  $k$  value of only four.

Just as an orbit (i.e., topological symmetry group) of a homogeneous graphlet [112], we define an orbit of a node-colored graphlet  $NCG_{i,b_n}$  as the set of nodes that are "symmetric" to each other in  $NCG_{i,b_n}$ ; the symmetry ignores node colors (Fig. 3.1). For a homogeneous graphlet with  $x$  orbits, each of its colored graphlets also has  $x$  orbits. That is, given  $k$  node colors, there are  $73 \times (2^k - 1)$  orbits for 2-5-node node-colored graphlets (there are 73 orbits for homogeneous 2-5 node graphlets). Then, we define heterogeneous *node-colored GDV* ( $NCGDV$ ) by counting the number of node-colored graphlets that the given node "touches" at each of the node-colored orbits. Analogous to the homogeneous case, to compare two nodes in heterogeneous networks, we compare their NCGDVs.

Second, analogous to the definitions for node-colored graphlets, without going again through all the formalisms, we define edge-colored graphlets (Fig. 3.1), orbits in edge-colored graphlets, and *edge-colored GDV* ( $ECGDV$ ). In practice, we may



expect a relatively small number of edge colors (e.g., we can study a network whose nodes are genes/proteins and whose edges are PPIs, genetic interactions, gene co-expressions, and signaling interactions with only four edge colors).

Third, the above ideas can be combined to define truly heterogeneous graphlets that have different node and edge colors. For each node-colored graphlet, one can vary its edge colors. Alternatively, it is possible and computationally much simpler to concatenate NCGDVs and ECGDVs, which does not add any additional computational complexity compared to computing only NCGDVs or only ECGDVs.

### 3.3.3 From homogeneous to heterogeneous edge conservation

Let  $u, v$  be two nodes in a network  $G$ , and  $u', v'$  be two nodes in a network  $H$ . Let  $f$  be a mapping (i.e., alignment) from the nodes of  $G$  to the nodes of  $H$  such that  $f(u) = u'$  and  $f(v) = v'$  (another way to say this is that source node  $u$  has image  $u'$ , and source node  $v$  has image  $v'$ ). That is,  $u$  is aligned to  $u'$ , and  $v$  is aligned to  $v'$ . Then, a conserved edge is formed by two edges from different networks such that each end node of one edge is aligned under  $f$  to a unique end node of the other edge. On the other hand, a non-conserved edge is formed by an edge from one network and a pair of nodes from the other network that do not form an edge, such that each end node of the edge is aligned under  $f$  to a unique node of the non-edge. Then, homogeneous  $S^3$  of an alignment is defined as the ratio of conserved edges to the sum of conserved and non-conserved edges (Fig. 3.2) [143]. We define a new measure of heterogeneous EC by modifying  $S^3$  to account for colors of aligned end nodes of a conserved edge, as described and illustrated in Section “3.1.2 – From homogeneous to heterogeneous EC”. Note that our chosen heterogeneous edge conservation weights of 1 for a fully conserved edge in which each of the two pairs of aligned nodes match in color,  $\frac{2}{3}$  for a partly conserved edge in which only one of the two pairs of aligned nodes match in color, and  $\frac{1}{3}$  for even less conserved edge in which none of the two

pairs of aligned nodes match in color, are just one of possible choices, which we use for simplicity, as a proof-of-concept of our new heterogeneous  $S^3$  measure. Other choices of weights are possible.

### 3.3.4 From homogeneous to heterogeneous network alignment

We modify three recent NA methods, WAVE, MAGNA++, and SANA, to account for heterogeneous networks. We describe these algorithms and their modifications below.

**WAVE.** WAVE takes as input two networks and an NC-based matrix that captures pairwise similarities between the nodes across the compared networks, and then uses a seed-and-extend algorithm to align the networks. First, two highly similar nodes are aligned, i.e., seeded. Then, the seed’s neighbors that are similar are aligned, and then the seed’s neighbor’s neighbors that are similar are aligned, and so on, until there is a one-to-one mapping between the networks. By aligning similar nodes, NC is optimized, and by looking at neighbors of already aligned nodes, EC is optimized, though only implicitly.

To account for heterogeneous networks, we simply plug into WAVE’s alignment strategy a new matrix of node similarities that is based on our new hetNC measure generated by our proposed heterogeneous graphlet approach. Based on the fact that the algorithm looks at the neighbors of the seed, WAVE optimizes HetEC implicitly, and there is no ability to incorporate heterogeneous  $S^3$  as an optimization parameter.

**MAGNA++.** MAGNA++ takes as input two networks and an NC-based matrix of node similarities, like WAVE. However, unlike WAVE, MAGNA++ uses a genetic search algorithm as its alignment strategy. MAGNA++ first starts with an initial population of randomly created alignments, the first generation. Then, high-scoring alignments (with respect to some objective function, see below) are given as input to a “crossover” function, which combines two alignments to create a new child alignment.

Many alignments from the initial population are crossed over to form new children alignments, which become the new population for the next generation. This process continues for a user-specified number of generations, and the alignment that scores the highest with respect to the objective function is given as output.

MAGNA++’s objective function can be only NC, only EC, or some combination of both. In the homogeneous case, optimizing a combination of NC (based on homogeneous graphlets) and EC ( $S^3$ ) as objective function was shown to produce the best alignments (where the objective function is  $\alpha \times \text{NC} + (1 - \alpha) \times \text{EC}$ , for some  $0 < \alpha < 1$ ; the best  $\alpha$  value was determined to be 0.5) [164]. Thus, to generalize MAGNA++ to its heterogeneous counterpart, we use MAGNA++’s alignment strategy to optimize the equally weighted combination of colored graphlet-based HetNC and heterogeneous  $S^3$ -based HetEC measures. To account for colored graphlet-based HetNC, we give MAGNA++ as input the colored-graphlet based node similarity matrix. To account for heterogeneous  $S^3$ , we modify the calculation of  $S^3$  to account for node colors; source code for these changes can be found on the project website (see Abstract).

**SANA.** SANA takes as input two networks and an NC-based matrix of node similarities, like WAVE and MAGNA++, and is a search algorithm, like MAGNA++. However, it uses simulated annealing instead of a genetic algorithm as its alignment strategy. SANA starts with a single random alignment rather than a population of random alignments, and in each step it explores “neighboring” alignments (described below). If a neighboring alignment scores higher with respect to the objective function, then it is chosen as the new alignment for the next iteration. Exploring neighboring alignments allows SANA to incrementally calculate the objective function; in particular for  $S^3$ , each move in the exploration process is only a small change in the alignment, and so only the changes in conserved and non-conserved edges resulting directly from the swap or change affect the  $S^3$  value. Note that there is also a small

chance a worse-scoring neighbor is chosen; this chance is described by the “temperature schedule”. Intuitively, the longer SANA has been running, the lower the chance of choosing a worse alignment. This continues for a set amount of time, which is a parameter of SANA. After the algorithm finishes, the alignment of the last iteration is given as output.

SANA’s objective function can be only NC, only EC, or some combination of both, as is the case with MAGNA++. Thus, to generalize SANA to its heterogeneous counterpart, we use SANA’s alignment strategy to optimize the equally weighted combination of colored graphlet-based HetNC and heterogeneous  $S^3$ -based HetEC measures. To account for colored graphlet-based HetNC, we give SANA as input the colored-graphlet based node similarity matrix. To account for heterogeneous  $S^3$ , we modify the incremental calculation of  $S^3$  to account for node colors; pseudocode for these changes can be found on the project website (see Abstract). Note that for our heterogeneous modification of SANA we provide pseudocode rather than modified source code because SANA is not our group’s method (MAGNA++ and WAVE are), and thus, there could be intellectual property restrictions regarding us sharing SANA’s source code. Instead, the user can get the homogeneous SANA’s code from the original authors and then modify it according to our pseudocode to allow for heterogeneous NA.

Here, we explain what a neighboring alignment means according to SANA. Let  $G$  and  $H$  be two networks being aligned, with  $G$  having fewer nodes than  $H$ , and let  $a, b, c, d$  be nodes in  $G$ , and  $a', b', c', d'$  be nodes in  $H$  such that  $a$  is aligned to  $a'$ ,  $b$  to  $b'$ ,  $c$  to  $c'$ , and  $d$  to  $d'$ . There are two kinds of neighboring alignments: swap and change. Swap neighbors differ from the original alignment in exactly two places, i.e., two source nodes in question remain the same but their images are exchanged. For example, given the existing alignment in Fig. 3.2, one of its possible swap neighbors is the alignment where  $a$  is aligned to  $b'$  and  $b$  is aligned to  $a'$ , while all other aspects

of the alignment remain the same. Change neighbors differ in only one place, i.e., a source node in question remains the same but its image is changed. In the example of Fig. 3.2, a possible change neighbor of the given alignment is one where  $a$  is aligned to some  $e'$  that initially was not part of the alignment, while all other aspects of the alignment remain the same. Consequently, if the two networks being aligned are of the same size, only swap neighbors are possible. With just these two types of neighbors, all possible alignments can potentially be reached; however, SANA focuses on those alignments that improve with respect to the objective function.

### 3.4 Conclusion

We modify WAVE, MAGNA++, and SANA to align heterogeneous networks by extending the existing notions of NC and EC to their heterogeneous counterparts. Specifically, we extend homogeneous graphlets to their heterogeneous counterparts, and homogeneous  $S^3$  to heterogeneous  $S^3$ . We evaluate our methods by aligning synthetic, PPI, and protein-GO networks to their noisy counterparts. We show that using more colors leads to better alignments, and that using both heterogeneous NC and heterogeneous EC is the preferred option where available. Also, we find that WAVE and SANA perform equally well at lower noise levels, though SANA does better at higher noise levels.

There are many new directions in which this work could be taken. Faster heterogeneous graphlet counting methods could be developed by using combinatorial relationships between heterogeneous graphlets, akin to existing efficient methods for homogeneous graphlet counting [79, 104, 137, 2]. Or, faster, more scalable methods for capturing the topology of a node in a heterogeneous network could be developed as an alternative to graphlets, such as those based on random walks [66, 41]. Also, our considered networks have up to four colors; aligning networks with more colors, as well as adding explicit (rather than just implicit, as in this study) edge colors,

could show further improvements. Another direction is improving the AS of NA methods. For example, in WAVE, the choice of the first aligned (seed) node pair likely impacts the rest of the alignment. If there are many possibilities for this pair, can an algorithm discover the best one, independent of the noise level in the data? Furthermore, while NA has been extended from dealing with static networks to dealing with dynamic networks [165, 162], the existing dynamic NA work currently only deals with homogeneous dynamic networks. Developing methods to align heterogeneous dynamic networks may yield improvements. In a similar vein, our current heterogeneous work deals with PNA, and so extending it into heterogeneous MNA may be of future interest.

## CHAPTER 4

### DATA-DRIVEN NETWORK ALIGNMENT

The work in this chapter is discussed in the following papers:

- **Shawn Gu** and Tijana Milenković (2018), Graphlets versus node2vec and struc2vec in the task of network alignment, in Proceedings of the 14th International Workshop on Mining and Learning with Graphs (MLG) at the 24th ACM SIGKDD 2018 Conference on Knowledge Discovery & Data Mining (KDD), London, UK, August 19-23, 2018. [67]
- **Shawn Gu** and Tijana Milenković (2020), Data-driven network alignment, PLOS ONE, 15(7): e0234978. [68]
- **Shawn Gu** and Tijana Milenković (2021), Data-driven biological network alignment that uses topological, sequence, and functional information, BMC Bioinformatics, 22: 34. [69]

#### 4.1 TARA: Data-driven network alignment

##### 4.1.1 Introduction

###### 4.1.1.1 Background and motivation

While HetNA did improve alignment quality over HomNA, we believe that HetNA does not address an underlying issue of traditional NA, regardless of homogeneity or heterogeneity. Namely, both the NC and EC measures of traditional NA aim to optimize the topological similarity between networks (in an effort to achieve an isomorphic-like matching), with the assumption that this topological similarity leads to functional relatedness. However, we argue that this assumption does not hold. Recall that there may be several reasons for this.

First, current PPI network data are highly noisy, with many missing and spurious PPIs (and even proteins) [93]. This alone can cause mismatches between proteins’ topological similarity and their functional relatedness. For example, if a set of three proteins that are all linked to each other via PPIs (i.e., a triangle) is in reality fully evolutionary conserved (i.e., functionally related) between two species, then the two triangles in the two species are topologically similar. But say that one of the three PPIs that actually exists in reality is missing in exactly one of the two species’ current PPI networks due to data noise. Then, it is a 3-node path in that species that should be aligned to a triangle in another species in order to identify the functional match. That is, the functionally related regions are now topologically dissimilar due to the data noise.

Second, even when PPI network data become complete, the traditional assumption of topological similarity is unlikely to hold due to biological variation between species. Namely, molecular evolutionary events such as gene duplication, deletion, or mutation may cause PPI network topology to differ across species’ evolutionary conserved (i.e., functionally related) network regions. Even for protein sequence alignments, pairwise sequence identity as low as 30% is sufficient to indicate evolutionary conservation (i.e., homology) for 90% of all protein pairs [140]. So, one can perhaps expect evolutionary conserved PPI networks of different species to be as topologically dissimilar.

Third, there could be additional factors that have yet to be discovered.

Regardless of the causes, this study is the first to provide actual evidence that the traditional topological similarity assumption does not hold. Briefly, we investigate whether functionally related nodes are indeed topologically similar in two tests: on synthetic networks and on real-world PPI networks of different species. In the process, we consider multiple prominent measures of topological similarity. As discussed in Section “4.1.3 – Topological similarity versus functional relatedness”, we



find that functionally related nodes are only marginally more similar (for all considered measures of topological similarity) to each other than at random. This means that aligning topologically similar nodes, as existing NA methods do, has only a marginally higher chance of aligning functionally related nodes than functionally unrelated nodes.

#### 4.1.1.2 Our contributions

Because the assumption of existing NA methods that topologically (and sequence) similar nodes should be aligned (i.e., are functionally related) does not hold, we propose a new paradigm for NA. Namely, we aim to redefine NA as a data-driven framework, which attempts to learn from the data what kind of topological relatedness corresponds to functional relatedness, without assuming that topological relatedness means topological similarity. So, regardless of whether the traditional topological similarity assumption fails due to noisy data, biological variation, or something else, we hypothesize that topological relatedness can better capture functional relatedness than topological similarity can. As topological relatedness and topological similarity are important concepts for understanding our paper, recall their difference from Fig. 1.4.

With our new notion of topological relatedness, we make no assumptions about what nodes should be aligned, distinguishing us from existing NA methods. Specifically, as a proof-of-concept methodological solution to test our new paradigm, we train a supervised classifier that, given a topological feature vector (i.e., low-dimensional embedding) of a node pair, learns from the (training) data when nodes are functionally related and when they are not. Note that because state-of-the-art topological features in the field of NA rely on graphlets [53, 72], we use graphlet-based feature vectors in our new framework. Importantly, we do not use any anchor links between nodes of different networks in order to calculate the feature vector of a node pair,

unlike many existing methods. Then, we use pairs (from the testing data) whose nodes are predicted to be functionally related to build an alignment. In other words, we consider two nodes to be topologically related if they are predicted to be functionally related, and our framework aligns such nodes, unlike existing methods that align topologically similar nodes. Of course, we make predictions only for node pairs that are not in the training data, which avoids any circular argument. So, we convert the NA problem into the problem of across-network supervised protein functional classification. While established supervised versions of many problems do exist, supervised NA has barely been studied before. We refer to the entire framework described above as TARA (data-driven network alignment).

TARA is a global, pairwise, and many-to-many method that does not use sequence similarity-based anchor links. We evaluate TARA against three state-of-the-art NA methods that are as similar as possible to TARA in terms of their algorithmic design or output, namely against WAVE [158], SANA [103], and PrimAlign [87]. Specifically, just like TARA, WAVE and SANA are global and pairwise, do not use anchor links, and furthermore are also graphlet-based. The only difference is that WAVE and SANA are one-to-one, unlike TARA. So, we also analyze PrimAlign, which is many-to-many and also global and pairwise, like TARA. Unlike TARA, PrimAlign does use anchor links in the form sequence similarities between networks. We evaluate each method on both synthetic (geometric and scale-free) and real-world (yeast and human PPI) networks.

Overall, we find that TARA is able to accurately learn what kind of topological relatedness corresponds to functional relatedness, and that TARA is able to predict the functions of proteins more accurately or in a complementary fashion compared to the existing NA methods, even those that use both topological and sequence information, mostly at lower running times. Thus, there is a need for introducing our new data-driven approach.

#### 4.1.1.3 Related work

Traditional biological NA methods typically consist of two algorithmic components. First, the similarity between pairs of nodes is computed with respect to topology, sequence, or both. Second, an alignment strategy identifies alignments that maximize the similarity between aligned node pairs and the amount of conserved edges (intuitively, alignments should preserve interactions). There are two common types of alignment strategies, as follows.

One is seed-and-extend, where first two highly similar nodes are aligned, i.e., seeded. Then, the most similar of the seed’s neighboring nodes (or simply neighbors), the neighbors of the seed’s neighbors, etc. are aligned. This continues until all nodes of the smaller of the two networks are aligned (until a one-to-one node mapping between the two networks is produced). WAVE [158] is a state-of-the-art seed-and-extend alignment strategy that works the best under a graphlet-based topological similarity measure.

The other type of alignment strategy is a search algorithm. Here, instead of aligning node by node like a seed-and-extend method, the solution space of possible alignments is explored, and the one that scores the highest with respect to some objective function is returned. This objective function typically tries to maximize the overall node similarity and the number of conserved edges. SANA [103] is a state-of-the-art search algorithm-based method. Specifically, it uses simulated annealing to search through possible one-to-one alignments, and works the best under an objective function that maximizes the overall graphlet-based topological similarity as well as the number of conserved edges.

On the other hand, PrimAlign [87] is a method with an alignment strategy that does not strictly belong to one of the above two categories. PrimAlign models the network alignment problem as a Markov chain where every node from one network is linked to some or all nodes in the other network with some scores; for PPI net-

works, these scores can be sequence similarities. In other words, PrimAlign makes use of anchor links between networks. The chain is then repeatedly transitioned until convergence, redistributing the across-network link scores using a PageRank-inspired algorithm. Those links that are above a certain threshold are taken as the alignment. As a result, PrimAlign outputs a many-to-many alignment, where a protein from one network may be aligned to many proteins in the other.

A method called MUNK has appeared recently [57]. Like PrimAlign, MUNK also relies on sequence-based anchor links (specifically, homologs) between two networks, but unlike PrimAlign, MUNK uses a matrix factorization approach to embed the nodes into a low dimensional space. Then, it uses these embeddings to calculate similarities between pairs of nodes, and employs the Hungarian algorithm on these similarities to generate an alignment. In preliminary analyses of MUNK on our data, we found that the similarity scores were not able to distinguish between functionally related and functionally unrelated nodes. This, combined with the fact that MUNK appeared after this study has been completed, is why we do not pursue it further.

The above methods do not use any functional (i.e., Gene Ontology (GO) [8]) information in the alignment process, unlike TARA. However, one method, DualAligner [144], does use such information, albeit in a different way than TARA. Given two PPI networks where some of the proteins are annotated with GO terms, DualAligner first forms “function-constrained subgraphs” (connected subgraphs sharing a GO term) in each network. Then, it tries to align subgraphs of the same function across networks. Next, it aligns proteins within these subgraphs that are topologically and sequence similar. Finally, it uses a seed-and-extend strategy around these aligned pairs to match unannotated proteins. However, more recent, state-of-the-art methods have appeared since DualAligner, including WAVE, SANA, and PrimAlign, which is why we do not consider DualAligner in this study.

All of the above methods are unsupervised. That is, they assume that topologically similar nodes are functionally related. Of course, many other such methods exist [72]. However, in the WAVE, SANA, and PrimAlign studies, the three methods were shown to outperform a number of the previous NA methods including AlignMCL [116], AlignNemo [29], CUFID [85], HubAlign [73], IsoRankN [99], L-GRAAL [102], MAGNA [143], MAGNA++ [164], MI-GRAAL [95], NETAL [126], NetCoffee [80], NetworkBLAST [86], PINALOG [135], and SMETANA [142]. In turn, these methods were shown to outperform GHOST [130], IsoRank [153], NATALIE [44], PISwap [27], and SPINAL [3]. So, the fact that WAVE, SANA, and PrimAlign are state-of-the-art, coupled with the fact that they are the most directly comparable to TARA (in terms of algorithmic design or output), is why we focus on them.

In addition, two supervised methods do exist, IMAP and MEgo2Vec, as follows.

IMAP [23] is an NA method that incorporates supervised learning, but in a different way than what we propose. First, IMAP requires an (unsupervised) alignment between two networks as input. Then, it obtains a topological feature vector for each node pair. Node pairs that are aligned form the positive class, and node pairs that are not aligned are sampled to form the negative class. Then, IMAP uses this data to train a linear regression classifier. After training, the data is passed through the classifier again in order to assign a score to every node pair. These scores are used in a matching algorithm (e.g., Hungarian or stable marriage) to form a new alignment, which is then given back as input into the method. This process is repeated for a set number of iterations – in general, it is shown that these iterations improve alignment quality. However, IMAP still makes the assumption that topologically similar nodes should be mapped to each other, meaning it still suffers from the issues of other NA methods. TARA on the other hand learns from the data what kind of topologically related nodes should be mapped to each other. We did attempt to test IMAP in this study, but the code was not available, and when we tried to implement it ourselves,

we could not get the method to work (i.e., we were not able to reproduce results from the IMAP study).

MEgo2Vec [180] proposes a framework to try to match user profiles across different social media platforms. Using graph neural networks and natural language processing techniques to obtain features of pairs of profiles from different platforms, MEgo2Vec then trains a classifier to predict whether two profiles correspond to the same person. However, because MEgo2Vec uses text processing techniques to match users’ names, affiliations, or research interests (in addition to network topological information), it is not directly suitable for matching proteins across PPI networks. Unlike MEgo2Vec, TARA relies solely on topological information (although it can also use external, e.g., sequence, information, this is out of the scope of this study).

There also exists a variety of methods that aim to predict the function of proteins *within a single* PPI network using techniques such as guilt-by-association, clustering, or classification [149, 120]. While this is a valuable research area, we are interested in a different problem – that of *across-network* protein function prediction. As such, we do not consider single-network methods in this study.

Lastly, there exist methods that aim to predict protein function without using any PPI network information. A variety of approaches entered in the Critical Assessment of Functional Annotation (CAFA) challenge [183] fall under this category. For example, the most recent top performing method, GOLabeler [177], uses a combination of protein sequence, amino acid, structural, and biophysical information, in order to predict GO term annotations of proteins. However, these kinds of *non-network* approaches are out of the scope of this *network-focused* study.

## 4.1.2 Methods

### 4.1.2.1 Data

Like many NA methods do, we evaluate TARA on network sets with known node mapping (networks generated from different graph models and their randomly perturbed counterparts) and a network set with unknown node mapping (yeast and human PPI networks).

**Network sets with known node mapping.** We use two network sets with known node mapping, generated from two network (i.e., random graph) models: 1) geometric random graphs [133] and 2) scale-free networks [10]. Because these two models have distinct network topologies [113], we can test the robustness of our results to the choice of model. For a given model, we create a network with 1,000 nodes and 6,000 edges, and then generate five instances of  $x\%$  random perturbation (i.e., we randomly delete  $x\%$  of the edges and then randomly add the same number of edges back), varying  $x$  to be 0, 10, 25, 50, 75, and 100. Because only edges differ between the original network and a randomly perturbed counterpart, we know the correct node mapping, and pairs in this mapping can be considered to be “functionally” related.

**Network set with unknown node mapping.** We use the PPI networks of yeast (5,926 nodes and 88,779 edges) and human (15,848 nodes and 269,120 edges) analyzed by the PrimAlign study [87], obtained from BioGRID [24]. Because we do not know the true node mapping between these networks, we rely on GO annotations to measure the functional relatedness between proteins (discussed below). We accessed the GO data in November 2018.

### 4.1.2.2 TARA: Data-driven network alignment

Recall that TARA trains, on a portion of the data, a supervised classifier using topological relatedness-based feature vectors of node pairs and their labels (whether

the nodes in a given pair are functionally related or not). Then, it aims to predict, on the remainder of the data, if a node pair is functionally related, creating an alignment out of all pairs predicted as such. Finally, this alignment can be used in a protein function prediction framework. Below, we describe how a topological relatedness-based feature vector of a node pair is extracted (subsection “Topological relatedness of a node pair.”), how the classifier is trained and evaluated on each of the network sets (subsections “TARA for a network set with known/unknown node mapping.”), and how the protein function prediction framework works (subsection “TARA as an NA framework for protein function prediction.”).

**Topological relatedness of a node pair.** We quantify topological relatedness using the notion of graphlets. Graphlets are Lego-like building blocks of complex networks, i.e., small subgraphs of a network (a path, triangle, square, etc.). In this study, we consider up to 5-node graphlets. They can be used to summarize the extended neighborhood of a node as follows. For each node in the network, for each topological node symmetry group (formally, automorphism orbit), one can count how many times a given node touches each graphlet at each of its orbits. The resulting counts for all graphlets/orbits are the node’s *graphlet degree vector* (GDV) [112], which has a length of 73 for up to 5-node graphlets. Then, to obtain the feature of a node pair, we simply take the absolute difference of the nodes’ GDVs (GDVdiff). Note that in addition to GDVdiff, we also tested appending the nodes’ GDVs together (GDVappend), and a weighted difference of the nodes’ GDVs based on the GDV similarity [112] calculation (GDVsim). However, the GDVdiff outperformed GDVappend, and while it obtained similar results to GDVsim, GDVdiff is mathematically simpler. As such, we only focus on GDVdiff, as calculated in Algorithm 1.

**TARA for network sets with known node mapping.** First, in order to see whether functional relatedness can even be predicted from topological relatedness,



TABLE 4.1

TABLE OF NOTATIONS AND THEIR MEANINGS

Notation	Meaning
$G_i$	Network $i$
$V_i$	Node set of network $i$
$E_i$	Edge set of network $i$
$ S $	Size of a set $S$
$v_{i_j}$	$j$ th node of network $i$ , i.e., $v_{i_j} \in V_i$
$f(\_, G_i)$	$f : (v_{i_j}, G_i) \rightarrow \mathbb{R}^{73}$ , a function that returns the GDV of node $v_{i_j} \in V_i$
$g(\_, G_i, G_j, d)$	Defined in Algorithm 1
$bal(G_i, G_j, R, R', d)$	Defined in Algorithm 2
$abs(\mathbf{x})$	Element-wise absolute value of a vector $\mathbf{x}$
<code>random.sample(<math>S, n, d</math>)</code>	From set $S$ , randomly sample $n$ elements without replacement, based on random seed state $d$
$U \times V$	Cartesian product of sets $U$ and $V$
$getAln(G_i, G_j, R, R', d, y)$	Defined in Algorithm 3

we evaluate our classifier using 10-fold cross-validation; if not, further study would be pointless. To do so, we start by creating a dataset that is balanced between the positive class (node pairs that are known functionally related) and the negative class (node pairs that are not currently known to be functionally related). But, because there are many more node pairs in the negative class, we undersample them to match the positive class in size, a common technique when dealing with class imbalance [157]. The process for creating one balanced dataset is outlined in Algorithm 2. Then, given this balanced dataset, we split the data into training and testing sets. We sample

---

**Algorithm 1** Extracting the “GDVdiff” feature vector of a node pair. Given two networks  $G_1$  and  $G_2$ , and a node pair  $s_{ij} = (v_{1_i}, v_{2_j})$ , define a function  $g : (s_{ij}, G_1, G_2) \rightarrow \mathbb{R}^{73}$ , computed as follows. For notations and their meanings, see Table 4.1.

---

```

1: let  $\mathbf{a}_{1_i} = f(v_{1_i}, G_1)$ 
2: let  $\mathbf{a}_{2_j} = f(v_{2_j}, G_2)$ 
3: return  $abs(\mathbf{a}_{1_i} - \mathbf{a}_{2_j})$ 

```

---

90% of the data to become the training set (ensuring balanced class sizes), and so the remaining 10% becomes the testing set. For 10-fold cross-validation, we take 10 stratified samples so that each data instance appears in exactly one of the ten testing sets, resulting in 10 “folds”. Given a fold, we then train a logistic regression classifier using the GDVdiff feature for a node pair to predict whether the given two nodes are functionally related, and evaluate this classifier using the accuracy and area under receiver operating characteristic curve (AUROC). For each score, we average over the 10 folds. We also repeat the undersampling 10 times to ensure any outcome is unlikely due to how we sample the negative class. So, we obtain 10 balanced datasets, and thus 10 accuracy and 10 AUROC scores, and for each measure we average the 10 scores. We repeat this process for every random perturbation amount. Note that we also tested Naive Bayes, decision tree, and simple neural network classifiers; trends were qualitatively similar, but logistic regression gave the best results. As such, we focus on logistic regression.

Second, we analyze the amount of data needed to train a good classifier, since only a small amount of data may be available in many real-world applications. For each network model, for each random perturbation amount, we obtain 10 balanced datasets using the same process as above. Then, for a given balanced dataset, we split the data such that  $y\%$  goes into the training set and the remaining  $(100 - y)\%$  goes into the testing set, still keeping the class balance in both the training and testing sets, varying  $y$  from 10 to 90 in increments of 10. For a given value of  $y$ , i.e., for what we call a  $y$  percent training test, we randomly create 10 instances of

---

**Algorithm 2** Creating a balanced dataset. Given two networks  $G_1$  and  $G_2$ , the set of conditions  $R$  that a node pair needs to satisfy to be considered functionally related, the set of conditions  $R'$  that a node pair needs to satisfy to be considered not functionally related, and random seed state  $d$ , define a function  $bal : (G_1, G_2, R, d) \rightarrow (S_1, S'_2)$  such that  $S_1 \subset V_1 \times V_2$  is a set of node pairs satisfying  $R$  (functionally related) and  $S'_2 \subset V_1 \times V_2$  is an equally sized set satisfying  $R'$  (not functionally related). For notations and their meanings, see Table 4.1.

---

- 1: let  $S_1$  be the set of node pairs between  $G_1$  and  $G_2$  satisfying  $R$ , i.e., the set of functionally related node pairs.
  - 2: let  $S_2$  be the set of node pairs between  $G_1$  and  $G_2$  satisfying  $R'$ , i.e., the set of functionally unrelated node pairs.
  - 3:  $S'_2 = \text{random.sample}(S_2, |S_1|, d)$
  - 4: **return**  $(S_1, S'_2)$
- 

this training and testing split, resulting in 10 accuracy and 10 AUROC scores, and for each measure we average results to ensure the outcomes are not due to the how we select the instances. Note that if  $y = 90$  and we were to take stratified samples instead of fully random samples, we would be performing 10-fold cross-validation as above. Finally, we average over all 10 balanced datasets to ensure the outcomes are unlikely due to how we sample the negative class.

**TARA for a network set with unknown node mapping.** Since we do not know the node mapping between yeast and human PPI networks, we must define functional relatedness in a different way compared to for network sets with known node mapping. We use GO annotations to do this. Specifically, if a yeast-human protein pair shares at least  $k$  biological process (BP) GO terms in which the protein-GO term annotations were experimentally inferred (i.e., if a given annotation has one of the following evidence codes: EXP, IDA, IPI, IMP, IGI, IEP), then we say the pair is functionally related. We vary  $k$  from 1 to 3. This gives us three sets of ground truth data, which we refer to as *atleast1-EXP*, *atleast2-EXP*, and *atleast3-EXP*. Also, no matter what  $k$  is, we define a functionally unrelated pair as a pair sharing no GO terms.

Then, for a given  $k$ , functionally related pairs form the positive class, and functionally unrelated pairs form the negative class. Once again because there are many more negative pairs, we take 10 samples that match the size of the positive pairs to create 10 balanced datasets and average over them. Again we use GDVdiff as the feature under logistic regression.

We again perform each of (i) 10-fold cross-validation and (ii)  $y$  percent training tests on the 10 balanced datasets, just like before, except that in the percent training tests, we now only perform the  $y\%$  training/testing split once instead of 10 times for simplicity, since we find the training/testing split does not significantly change the results.

**TARA as an NA framework for protein function prediction.** In addition to 10-fold cross-validation and  $y$  percent training tests (on each of networks with known and unknown node mapping), we evaluate TARA in a third test – that of protein function prediction. This is an important downstream task of NA. We evaluate TARA in this context as follows. For a given set of ground truth data atleast $k$ -EXP, we keep only GO terms that annotate at least two yeast proteins and at least two human proteins; without this constraint, it is impossible for the framework (described below) to make predictions for the GO term. Then, for a given percent training test  $y$ , we train TARA and make predictions on the remaining testing data. Every pair that is predicted to be in the positive class is added to an alignment. We outline the process in Algorithm 3. This alignment, as well as alignments of existing methods that we evaluate against, is then put through the protein function prediction framework proposed by [110], which we summarize as follows. For each protein  $u$  in the alignment (that is annotated by at least  $k$  GO term(s)), we hide  $u$ ’s true GO term(s). Then, for each GO term  $g$ , we determine if the alignment is statistically significant with respect to  $g$ . This is done by calculating if the number of aligned node pairs in which the aligned proteins share  $g$  is significantly high ( $p$ -value less than 0.05 according to

the hypergeometric test [110]). After repeating for all applicable proteins and GO terms, we obtain a list of predicted protein-GO term associations. From this list we can calculate the precision and recall of the predictions.

---

**Algorithm 3** Generating an alignment. Given two networks  $G_1$  and  $G_2$ , the set of conditions  $R$  that a node pair needs to satisfy to be considered functionally related, the set of conditions  $R'$  that a node pair needs to satisfy to be considered not functionally related, random seed state  $d$ , and a  $y$  percent training amount, return an alignment of  $G_1$  and  $G_2$ . For notations and their meanings, see Table 4.1.

---

```

1: Initialize set A to be empty.
2: let  $(S_p, S_n) = \text{bal}(G_1, G_2, R, R', d)$ , as computed by Algorithm 2.
3: let  $Tr_p = \text{random.sample}(S_p, \lfloor y|S_p|/100 \rfloor, d)$ .
4: let  $Te_p = S_p \setminus Tr_p$ 
5: let  $Tr_n = \text{random.sample}(S_n, \lfloor y|S_n|/100 \rfloor, d)$ .
6: let  $Te_n = S_n \setminus Tr_n$ .
7: let  $Tr = Tr_p \cup Tr_n$  be the training data.
8: let  $Te = Te_p \cup Te_n$  be the testing data.
9: Train a predictive function,  $\text{LogReg} : \mathbb{R}^{73} \rightarrow \{0, 1\}$ , with logistic regression, on
    $Tr$ , where the feature vector of node pair  $s_{ij} \in Tr$  is given by  $g(s_{ij}, G_1, G_2)$ , and
   the label of  $s_{ij}$  is  $\{1 \text{ if } s_{ij} \in S_p, 0 \text{ if } s_{ij} \in S_n\}$ .
10: for each  $s_{ij} \in Te$  do
11:   let  $\mathbf{x}_{ij} = g(s_{ij}, G_1, G_2)$ 
12:   if  $\text{LogReg}(\mathbf{x}_{ij}) = 1$  then
13:     A.add( $s_{ij}$ )
14:   end if
15: end for
16: return A

```

For example, if  $G_1$  is the yeast PPI network,  $G_2$  is the human PPI network,  $R$  is the set of conditions for the atleast3-EXP ground truth dataset,  $R'$  is the set of conditions for a protein pair to be considered not functionally related (i.e., shared no GO terms of any kind),  $d$  is 0, and  $y$  is 90%, then  $\text{getAln}(G_1, G_2, R, R', d, y)$  returns the alignment between yeast and human generated by a classifier trained on functionally related protein pairs defined by  $R$ , functionally unrelated pairs defined by  $R'$ , using a random seed state of 0 for sampling, and 90% of the data for training.

---

While in traditional NA evaluations every GO term available is considered, some GO terms may be redundant or too general. Recent work has suggested that taking

the frequency of GO terms (how many proteins a GO term annotates out of all proteins analyzed) into account can deal with these issues [76]; intuitively, less frequent means more informative. However, because there is no hard definition for what makes a GO term rare enough, we consider three thresholds:

- All GO terms (i.e., ALL); this corresponds to traditional NA evaluation.
- GO terms that appear 50 times or fewer (i.e., threshold of 50).
- GO terms that appear 25 times or fewer (i.e., threshold of 25).

For a given GO term rarity threshold, we filter out all GO terms that do not satisfy the threshold. Then, for each *atleast $k$ -EXP* ground truth dataset (see above), we only consider proteins that share at least  $k$  GO terms from the filtered list to be functionally related (keep in mind that for proteins to be considered functionally unrelated, they still must share no GO terms, regardless of rarity). For example, *atleast1-EXP* at the 50 GO term rarity threshold considers proteins that share at least one experimentally inferred biological process GO term, such that each GO term annotates 50 or fewer proteins (out of all proteins from the yeast and human PPI networks we analyze), to be functionally related. In total, we now have nine “ground truth-rarity” datasets, resulting from combinations of the three *atleast $k$ -EXP* ground truth datasets and the three GO term rarity thresholds. Then, for each of these nine datasets, we train and test TARA on protein pairs satisfying the conditions (i.e., being in the *atleast $k$ -EXP* ground truth dataset at the given GO term rarity threshold), and evaluate the resulting alignment using the protein function prediction framework described above. Also, in order to fairly compare TARA to all existing NA methods, we evaluate the existing methods’ alignments with respect to these nine ground truth-rarity datasets. In this way, we can test the effect of both  $k$  in the *atleast $k$ -EXP* ground truth datasets and GO term rarity on prediction accuracy.

### 4.1.3 Results and discussion

#### 4.1.3.1 Topological similarity versus functional relatedness

Here, we provide evidence that the traditional assumption of NA methods, namely that topological similarity corresponds to functional relatedness, does not hold.

First, as a baseline, consider a network aligned to itself, i.e., to its 0% randomly perturbed counterpart. We know the correct node mapping, and pairs in this mapping can be considered functionally related. If we look at the topological similarity between pairs of nodes that should be aligned versus those that should not, we expect the former to be topologically identical, and the latter not to be. Also, we expect the distribution of topological similarities of the matching node pairs to be different than the distribution of topological similarities of non-matching pairs. Indeed, that is what we observe (Fig. 4.1(a) and Supplementary Fig. C.2(a)).

Now, consider a network aligned to its 25% randomly perturbed counterpart. Because only (a portion of) edges change, we still know the correct node mapping, i.e., which nodes are functionally related. At just 25% random perturbation (where networks are still 75% identical), we observe that the topological similarity distribution of node pairs that should be matched is now close to the topological similarity distribution of those that should not (Fig. 4.1(b) and Supplementary Figs. C.2(b) and C.3(b)). In other words, the functionally matching nodes are only marginally more similar to each other than at random. So, even if the two networks being aligned are just a little different (and it is expected that PPI networks of different species are much more different than that), topological similarity is no longer correlated with functional relatedness. This fact holds for multiple prominent topological similarity measures, including GRAAL’s [96] and MAGNA’s [143] GDV similarity measure (Fig. 4.1), GHOST’s [130] spectral signature-based similarity measure (Supplementary Fig. C.2), and IsoRank’s [153] PageRank-based similarity measure (Supplemen-

tary Fig. C.3). Note that while the three measures quantify topologically similarity in mathematically different ways, they all follow the general notion that a high score corresponds to neighborhood regions that are close to isomorphic (as discussed in Section “4.1.1 – Background and motivation”). Because all measures show qualitatively similar trends, and because GDV similarity was shown to outperform both GHOST’s and IsoRank’s similarities [36, 52], we focus on GDV similarity for the following analysis.

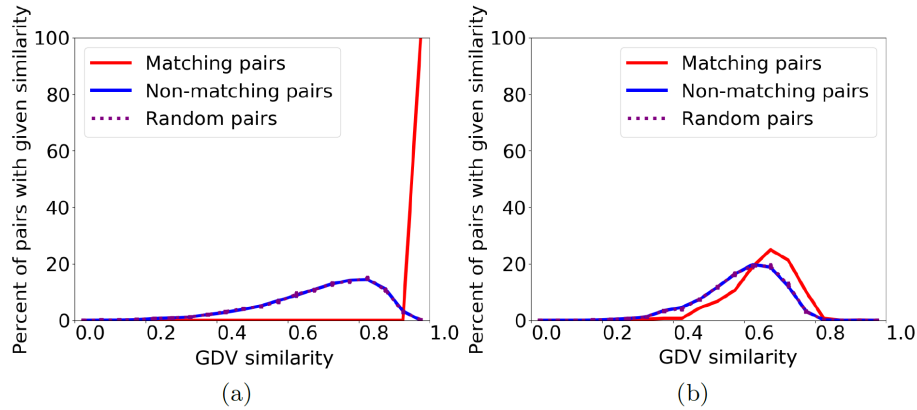


Figure 4.1. Distribution of topological similarity (GDV similarity) between node pairs of a geometric random graph (i.e., a synthetic network) and its **(a)** 0% and **(b)** 25% randomly perturbed counterparts. We show three lines representing the distribution of topological similarity for matching (i.e., functionally related) node pairs (blue), for non-matching, i.e., functionally unrelated, node pairs (red), and for 10 random samples of the same size as the set of matching pairs, averaged (purple). Results are qualitatively similar for 50% random perturbation, scale-free random graphs (a different type of synthetic networks), and GHOST’s and IsoRank’s similarity measures. (Supplementary Figs. C.1-C.3).



Second, we observe this trend, namely that the distributions of topological similarity for functionally related and functionally unrelated protein pairs are close to each other, for real world PPI networks as well (see below). Furthermore, we find that the distributions of sequence similarity are also close to each other for the two sets of protein pairs, and that distributions of the combination of topological and sequence similarities are close to each other as well. Specifically, we analyze the yeast and human PPI networks described in Section “4.1.2 – Data”. Here, we consider proteins that share at least one experimentally inferred biological process GO term to be functionally related, proteins that do not share any GO terms to be functionally unrelated, proteins with GDV similarity of 0.85 or greater to be topologically similar [108], and proteins with E-value of  $10^{-10}$  or lower to be sequence similar [105]. Our findings are as follows:

- Out of all functionally related protein pairs, only  $\sim 28\%$  are topologically similar (Fig. 4.2(a), above the horizontal line), while even out of all functionally unrelated protein pairs,  $\sim 14\%$  are still topologically similar (Fig. 4.2(b), above the horizontal line).
- Out of all functionally related protein pairs,  $\sim 63\%$  are sequence similar (Fig. 4.2(a), to the right of the vertical line), while even out of all functionally unrelated protein pairs,  $\sim 53\%$  are still sequence similar (Fig. 4.2(b), to the right of the vertical line).
- Out of all functionally related protein pairs, only  $\sim 18\%$  are both topologically and sequence similar (Fig. 4.2(a), top right quadrant), while even out of all functionally unrelated protein pairs, only  $\sim 8\%$  are both topologically and sequence similar (Fig. 4.2(b), top right quadrant).

In other words, functionally related nodes are only marginally more similar (for all types of similarity we consider) to each other than at random. Therefore, the existing NA assumption that is based on topological similarity fails, and instead our NA approach that is based on topological relatedness, TARA, is needed.

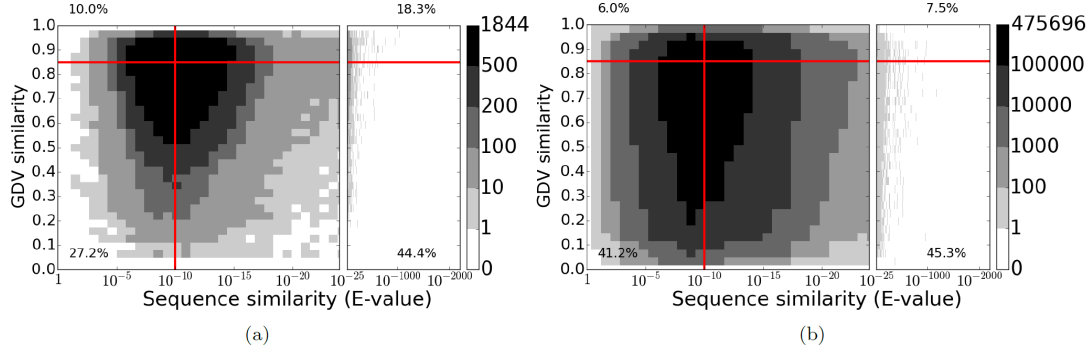


Figure 4.2. Distribution of topological similarity (GDV similarity) versus sequence similarity (E-value) between yeast and human PPI networks of those yeast-human protein pairs that are **(a)** functionally related (i.e., share at least one biological process GO term such that the protein-GO term annotation was experimentally inferred) and **(b)** functionally unrelated (i.e., share zero GO terms). The color of a pixel represents how many node pairs have a given topological similarity and given sequence similarity. The red horizontal and vertical lines indicate the thresholds for topologically similar ( $y \geq 0.85$ ) or sequence similar ( $x \leq 10^{-10}$ ) pairs, and the percentages indicate the fraction of pairs that are in a given quadrant.

#### 4.1.3.2 TARA for network sets with known node mapping

**10-fold cross-validation.** Here we evaluate TARA using 10-fold cross-validation. Specifically, for each network model (geometric and scale free), for each random perturbation level (0, 10, 25, 50, 75, 100), we obtain the average accuracy and average AUROC of the 10 folds. We expect that as the amount of random perturbation increases, prediction accuracy and AUROC decrease since the networks become more and more dissimilar. Indeed, this is what we observe (Fig. 4.3(a) and Supplementary Fig. C.4). Also, we expect a random classifier to give around 50% accuracy since the class sizes are balanced; it will also have 50% AUROC by definition. This is empirically verified by the results at 100% random perturbation, where we are attempting to classify nodes between two completely different networks (Fig. 4.3(a) and Supplementary Fig. C.4).

**Percent training tests.** Again, we expect that as the amount of random perturbation increases, accuracy and AUROC decreases since networks are becoming more dissimilar. Also, we expect that as we increase the amount of training data, the accuracy and AUROC increases as well since more information is being used in the classifier. Overall, these are the trends we observe (Fig. 4.3(b) and Supplementary Fig. C.5). We also see that using 90% of the data as training does not lead to drastic improvements; in fact, it is not always even the best. For some (geometric) networks, as low as 40% still gives comparable results. This is promising, as we do not necessarily have to rely on using a majority of the data for training.

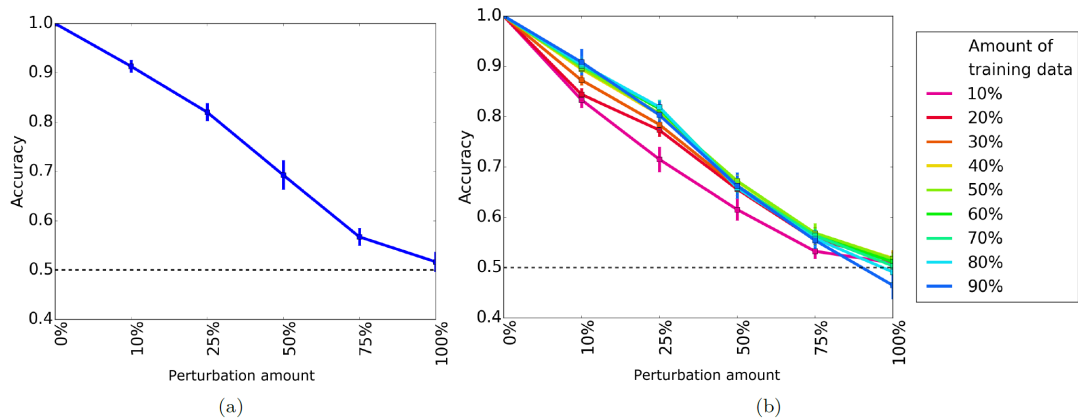


Figure 4.3. Average prediction accuracy of (a) 10-fold cross-validation and (b) percent training tests for a geometric network and its randomly perturbed counterparts. In panel (b), different colored lines represent how much data is used for training; these colors do not apply to panel (a). A dotted black line indicates the accuracy expected if the classifier makes random predictions. Qualitatively similar results for AUROC and for scale-free networks are shown in Supplementary Figs. C.4–C.5.

These tests serve as a proof of concept that there is some learnable pattern between topological and functional relatedness, and so it makes sense to continue this study for real-world networks.

#### 4.1.3.3 TARA for network sets with unknown node mapping

**10-fold cross-validation.** Again, we evaluate TARA using 10-fold cross-validation. We expect that as  $k$  increases, accuracy and AUROC do as well since the conditions for a pair of proteins to be functionally related becomes more stringent. Indeed, this is what we observe (Fig. 4.4(a) and Supplementary Fig. C.6).

**Percent training tests.** We see similar results for percent training as we do for 10-fold cross-validation (Fig. 4.4(b)) and Supplementary Fig. C.7). Note that unlike percent training for synthetic networks, the amount of training data has very little effect on accuracy except for atleast3-EXP. This may be because atleast1-EXP and atleast2-EXP contain a lot more data, meaning even a small percentage is enough to train a good classifier.

Overall, we are able to detect a pattern between topological relatedness and functional relatedness. So, it makes sense to generate an alignment and evaluate TARA in the protein function prediction task.

#### 4.1.3.4 TARA for protein function prediction

Here, we evaluate TARA and existing NA methods in the task of protein function prediction. Specifically, we take the alignments generated from each method and put them through the protein function prediction framework as described above. We first compare TARA’s percent training tests to each other, and then we compare TARA to existing NA methods.

**Comparing TARA’s percent training tests to each other.** For simplicity, we only compare a subset of TARA’s percent training tests. Specifically, because

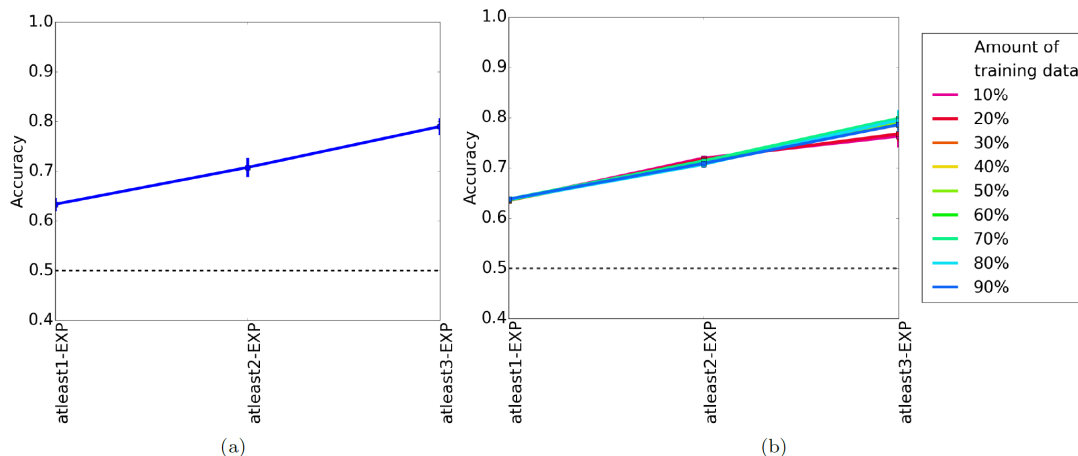


Figure 4.4. Average prediction accuracy of **(a)** 10-fold cross-validation and **(b)** percent training tests for real-world networks. In panel **(b)**, different colored lines represent how much data is used for training; these colors do not apply to panel **(a)**. A dotted black line indicates the accuracy expected if the classifier makes random predictions. Qualitatively similar results for AUROC are shown in Supplementary Figs. C.6–C.7.

classification accuracy does not vary significantly between different percent training tests, we focus on the extremes (10 and 90) and the middle (50). So, we have 27 total evaluation tests for TARA, resulting from combinations of the three percent training tests and the nine ground truth-rarity datasets discussed above.

We expect that as we increase the amount of training data (10 to 50 to 90), precision will increase and recall will decrease. This is because more training data means the classifier will likely be better (increasing precision), but will result in less testing data and thus smaller alignments and fewer predictions (decreasing recall). Similarly, we expect that as we increase  $k$  in our atleast $k$ -EXP ground truth datasets, precision will increase and recall will decrease. This is because at higher  $k$ , we will be training on higher quality data (increasing precision), but there will be less data overall, resulting in smaller alignments and fewer predictions (decreasing recall). Finally, we expect that as we consider rarer GO terms, precision will increase and

recall will decrease. Intuition from existing studies suggests that rarer GO terms are more meaningful [76], so the data will be higher quality (increasing precision), but again there will be less data overall (decreasing recall). Indeed, we observe these trends (Fig. 4.5 and Supplementary Fig. C.8) for all but atleast3-EXP at the 50 and 25 rarity thresholds; there is not enough data for TARA to generate alignments or make predictions for those parameters. Inability to learn on small datasets is one drawback of machine learning methods in general, not just TARA.

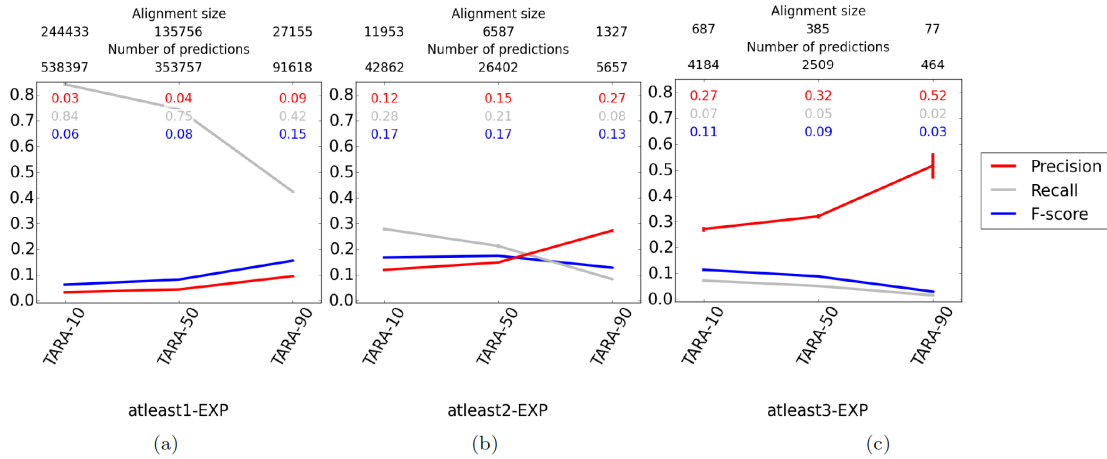


Figure 4.5. Comparison of different TARA evaluation tests in the task of protein function prediction, for the ALL GO term rarity threshold. Different percent training tests, specifically 10, 50, and 90, are compared within each panel, and different ground truth datasets, specifically (a) atleast1-EXP, (b) atleast2-EXP, and (c) atleast3-EXP, are compared across panels. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method, averaged over the 10 instances we perform for each test, are shown on the top. For example, the alignment for TARA-90 for the atleast2-EXP dataset contains 1,327 aligned yeast-human protein pairs, and predicts 5,657 protein-GO term associations. Raw precision, recall, and F-score values are color-coded inside each panel. Complete results for the other rarity thresholds are shown in Supplementary Fig. C.8.

In order to simplify comparisons between TARA and existing NA methods, we choose a representative percent training test (i.e., either TARA-10, TARA-50, or TARA-90) for each of the nine ground truth-rarity datasets discussed previously. In other words, we go from 27 TARA evaluation tests to nine (though we actually have seven since TARA does not make predictions for atleast3-EXP at the 50 and 25 rarity thresholds, per the above discussion). Generally, we try to choose the percent training test that has both high precision (meaning predictions are accurate) and a large number of predictions (meaning we uncover as much biological knowledge as possible), as these represent TARA’s best results. The choices are given in Table 4.2.

TABLE 4.2  
REPRESENTATIVE CHOICES OF TARA’S PERCENT TRAINING  
TESTS FOR EACH OF THE 9 GROUND TRUTH DATASETS

	ALL	50	25
atleast1-EXP	TARA-90	TARA-90	TARA-90
atleast2-EXP	TARA-90	TARA-10	TARA-10
atleast3-EXP	TARA-90	N/A	N/A

**Comparing TARA against existing NA methods.** We compare against three existing methods, WAVE, SANA, and PrimAlign. We compare against these three methods for the following reasons (also, see Section 4.1.1). WAVE and SANA are state-of-the-art methods that use graphlets, just like TARA, allowing us to fairly analyze how much TARA’s supervised process helps. Also, they operate under the

assumption that topologically similar nodes are functionally related, which is what TARA challenges. However, recall that WAVE and SANA are one-to-one methods, while TARA is a many-to-many method. So, we analyze PrimAlign, because it is a state-of-the-art many-to-many method. In addition, PrimAlign operates under the assumption that we challenge, namely that topologically similar nodes are functionally related. Recall from Section “4.1.1 – Related work” that WAVE, SANA, and PrimAlign were already shown to outperform a number of previous NA methods, and hence, we believe that comparing to these three methods is sufficient. Also, keep in mind that a theoretical precision of one is not possible with TARA, unlike WAVE, SANA, and PrimAlign. This is because TARA uses part of the ground truth data for training, meaning it impossible to make predictions for that portion. In other words, TARA is inherently disadvantaged compared to existing methods.

In more detail, WAVE and SANA use graphlet-based topological information like TARA (however, keep in mind that sequence information or any other data could also be used in TARA, which is subject of our future work). Specifically, WAVE and SANA both use GDV similarity to score the similarities of node pairs, and SANA also uses an equal weighing of node conservation and edge conservation (i.e., we set both `s3` and `esim` to 1). Unlike WAVE and SANA, PrimAlign uses both topological (PageRank-based) information and sequence similarity (negative log of E-value) information by default. Specifically, regarding the latter PrimAlign study, which analyzes the same yeast and human PPI networks as we do, considers all sequence similar proteins between the networks with an E-value  $\leq 10^{-7}$ , which results in 55,594 sequence similarity-based anchor links. We run this default version, called PrimAlign-TS. We also analyze a topological version of PrimAlign (PrimAlign-T) for fair comparison with TARA, which in this study uses topological but not sequence information. To create an as fairly comparable as possible topological version of PrimAlign, we instead use the 55,594 most topologically (GDV) similar yeast-human protein pairs as



anchor links. Lastly, we are also interested in using sequence information only (Sequence, or S), in order to better understand the effect of T or S alone. We do so by taking those 55,594 sequence similar pairs from PrimAlign-TS and treating them as the alignment, disregarding any topological information from the PPI networks.

Summarizing the different NA methods, TARA, WAVE, SANA, and PrimAlign-T use topological information, Sequence uses sequence information, and PrimAlign-TS uses both topological and sequence information. Furthermore, recall that the different methods have different levels of comparability to TARA in terms of information used (T versus S versus TS) and alignment type (one-to-one versus many-to-many) (Table 4.3). To show that our assumption holds, namely that topologically related, rather than topologically similar, nodes should be aligned, it would be sufficient to show that TARA, a T method, outperforms the other T methods. If TARA, a T method, also outperforms Sequence or PrimAlign-TS, then this would further underscore the need of a data-driven approach like ours.

We discuss our results below (Fig. 4.6 and Supplementary Fig. C.9). Note that we primarily focus on precision because in terms of potential wet lab validation of some predictions, we believe it is more important to have fewer but mostly correct predictions (e.g., 90 correct out of 100 made) than a greater number of mostly incorrect predictions (e.g., 300 correct out of 1000 made). While in the latter example more predictions are correct (300 versus 90), leading to a higher (almost triple) recall, many more are also incorrect, leading to lower precision (0.3 versus 0.9). However, we do not completely discount recall and F-score, as they may still be valuable measures for other considerations. Also, keep in mind that the expected precision and recall for a random alignment is near 0. A random alignment is not expected to match functionally related proteins, meaning essentially random protein-GO term associations will be predicted.

TABLE 4.3

COMPARABILITY OF THE EXISTING METHODS CONSIDERED IN  
THIS STUDY TO TARA IN TERMS OF TYPE OF INFORMATION  
USED (T VERSUS S VERSUS TS) AND ALIGNMENT TYPE  
(ONE-TO-ONE VERSUS MANY-TO-MANY)

Existing NA method	Fair to TARA in terms of:	
	Information used	Alignment type
WAVE	Yes	No
SANA	Yes	No
Sequence	No	Yes
PrimAlign-T	Yes	Yes
PrimAlign-TS	No	Yes

- Compared to other T methods, TARA is superior to WAVE and SANA in 6/7 tests with respect to precision, and in all seven tests with respect to recall (the seven tests are summarized in Table 4.2). Importantly TARA is always superior to PrimAlign-T, the most fairly comparable method to TARA, in terms of both precision and recall. These trends support our claim that topologically related, not topologically similar, nodes are the ones that are functionally related.
- Compared to PrimAlign-TS, TARA is superior in 3/7 tests (atleast2-EXP for the 50 and 25 rarity thresholds, and atleast3-EXP for ALL GO terms) with respect to precision. Of the remaining four tests, TARA is superior in two and comparable in two with respect to F-score.
- Compared to Sequence, TARA is superior in all seven tests in terms of precision, and superior in 3/7 in terms of recall. Of those remaining four tests, it is still superior in two of them with respect to F-score.

An interesting note is that the precision of PrimAlign-TS is much greater than simply the sum of precision from Sequence and PrimAlign-T, suggesting that combining topological and sequence information in a meaningful way can have compounded

effects. This is promising for incorporating sequence information into TARA, which is our future work.

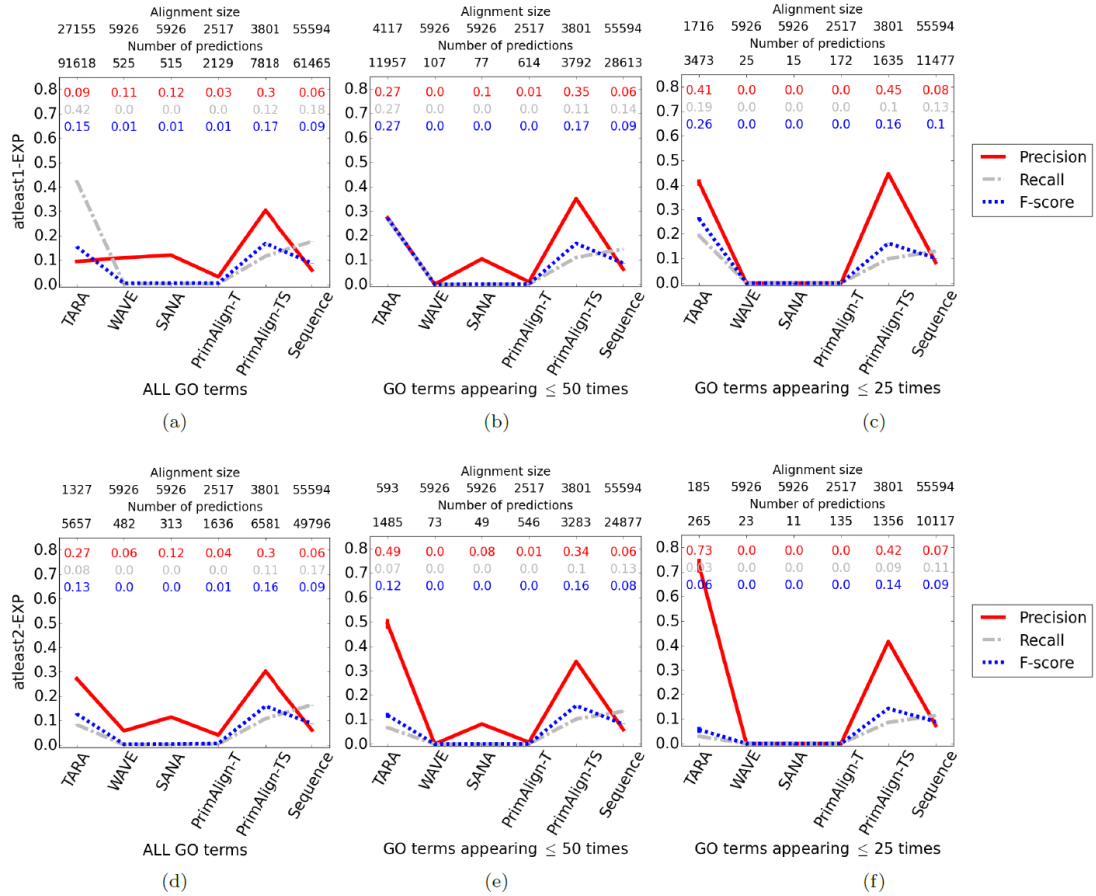


Figure 4.6. Comparison of the six considered NA methods for rarity thresholds (a, d) ALL, (b, e) 50, and (c, f) 25 using ground truth datasets (a, b, c) atleast1-EXP and (d, e, f) atleast2-EXP in the task of protein function prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method. For example, the alignment for TARA in panel (a) contains 27,155 aligned yeast-human protein pairs, and predicts 91,618 protein-GO term associations. Raw precision, recall, and F-score values are color-coded inside each panel. Results for atleast3-EXP are shown in Supplementary Fig. C.9.

While precision, recall, and F-score are important overall measures, it is also necessary to zoom into the actual predictions that the methods make. We focus on TARA and PrimAlign-TS, as these two methods perform the best, with the parameters from Fig. 4.6.

We see that for atleast1-EXP, no matter the rarity threshold, TARA makes many more predictions than PrimAlign-TS, and yet still has comparable precision for the 50 and 25 GO term rarity thresholds (Fig. 4.6). In other words, TARA is potentially uncovering more biological knowledge than PrimAlign-TS but with similar accuracy. For atleast2-EXP, for the ALL GO term rarity threshold, TARA and PrimAlign-TS make a similar number of predictions with similar precision, and for the 50 and 25 rarity thresholds, TARA outperforms PrimAlign-TS, though at the cost of fewer predictions (Fig. 4.6). For atleast3-EXP, for the ALL GO term rarity threshold, TARA outperforms PrimAlign-TS, also at the cost of fewer predictions (Supplementary Fig. C.9).

Importantly, we see that the number of predictions in the overlap of TARA and PrimAlign-TS is generally small (Fig. 4.7 and Supplementary Fig. C.10), suggesting that most of the two methods' predictions are complementary. Therefore, we can say that TARA has some advantage in every case (whether it be precision, recall, or number of predictions), and at worst complements PrimAlign, which even uses sequence information that TARA does not. This, in addition to TARA outperforming WAVE and SANA, justifies the need for introducing our new data-driven approach.

We also look at the time it takes to obtain an alignment for TARA, WAVE, SANA, PrimAlign-T, and PrimAlign-TS for the ALL GO term rarity threshold, as the given threshold has the most data and thus will be the worst case time-wise out of all thresholds. We do not consider Sequence as we did not compute any alignment in this case; instead, the alignment was included from the PrimAlign study. We expect as that  $k$  (in the atleast $k$ -EXP ground truth dataset) increases, the time for TARA

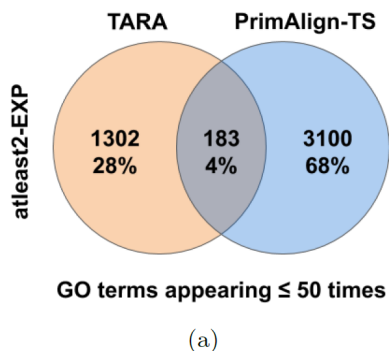


Figure 4.7. Overlap of the functional predictions made by TARA and PrimAlign for atleast2-EXP at the 50 rarity threshold. Percentages are out of the total number of unique predictions made by both methods combined. Complete results for all methods and parameters are shown in Supplementary Fig. C.10 and Supplementary File C.1.

to produce an alignment decreases since there is less (but higher quality) data overall, and thus less data to train on. This is what we observe (Table 4.4). Regarding the existing NA methods, WAVE uses a seed-and-extend alignment strategy, which is expected to take some time. The running time of SANA is a parameter, which we choose to be 60 minutes since SANA requires such time to find a good alignment for networks of the sizes we analyze. We find that WAVE and SANA are both slower than TARA for atleast2-EXP and atleast3-EXP, and SANA is comparable to TARA for atleast1-EXP, meaning that TARA is overall both faster and more accurate at predicting protein function than the two one-to-one NA methods. Lastly, we find that PrimAlign and its variants are fast, which is expected because the method is linear in the number of edges.

#### 4.1.3.5 A closer look at TARA

We also explore why TARA is able to outperform the traditional NA methods. Recall the distributions of topological similarity scores (which traditional NA methods

TABLE 4.4

RUNNING TIME (ROUNDED TO THE NEAREST SECOND)  
COMPARISON OF TARA, WAVE, SANA, PRIMALIGN-T, AND  
PRIMALIGN-TS FOR ALL GO TERMS

Running time (s)	atleast1-EXP	atleast2-EXP	atleast3-EXP
TARA	3,642	210	168
WAVE	1,686	1,686	1,686
SANA	3,600	3,600	3,600
Sequence	N/A	N/A	N/A
PrimAlign-T	3	3	3
PrimAlign-TS	16	16	16

use) from Section “4.1.3 – Topological similarity versus functional relatedness”. When the two networks are just a bit different from each other, nodes that should be matched (i.e., are functionally related) are only marginally more topologically similar to each other than at random, leading to suboptimal alignments. If we analyze TARA’s topological relatedness scores (described below) in the same way, we find that TARA can better distinguish matching node pairs from non-matching node pairs. This could explain why TARA outperforms the traditional NA methods.

To extract topological relatedness scores from TARA’s framework, we do the following. Consider the 90% training test (while this applies to any percent training test, we focus on 90 because TARA-90 generally performs the best), where we first train a classifier on 90% of a balanced dataset. Then, instead of evaluating on the remaining 10% of the data as above, we input the feature vector of each node pair across networks into the trained classifier. Rather than directly outputting whether

a pair is functionally related or not, we obtain the *probability* that the two nodes are functionally related instead. We can interpret this probability as a redefined “relatedness” measure, where now nodes are topologically related if they are likely to be functionally related.

Then, mirroring our initial analyses (Fig. 4.1), we examine the distributions of these topological relatedness scores on the same networks and random perturbation levels. For a geometric network and its 0% randomly perturbed counterpart, we again see a distinct difference between the distributions of matching pairs and all pairs (Fig. 4.8(a)). But, even for 25% random perturbation, the distributions are now different from each other (Fig. 4.8(b)), and this difference is greater than the difference in distributions of the equivalent topological (GDV) similarity scores (Fig. 4.1(b)). In other words, TARA’s topological relatedness scores are better able to distinguish matching node pairs from non-matching node pairs compared to traditional topological similarity scores, which could explain the superior results of TARA over traditional NA methods. Improving these learned topological relatedness scores (e.g., so that the difference in distributions at 25% random perturbation looks like the difference at 0% random perturbation), and using them to produce alignments that are more fairly comparable to some traditional NA methods (e.g., to produce one-to-one alignments) are subjects of our future work.

#### 4.1.3.6 Generalizability of TARA

Just like with any supervised classification approach, the key goal of TARA is to first train the approach on the training portion of the compared networks, and then to test it on the testing portion of the same networks that was hidden during the training process, in order to validate that the approach is accurate on the *known* but hidden knowledge from the testing data. Then, the goal is to retrain TARA on *all*

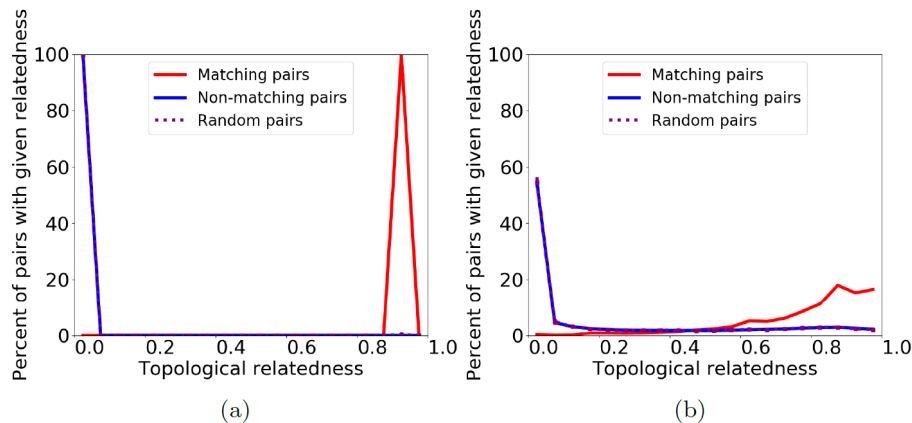


Figure 4.8. Distribution of TARA’s redefined topological relatedness between node pairs of a geometric random graph (i.e., a synthetic network) and its (a) 0% and (b) 25% randomly perturbed counterparts. We show three lines representing the distribution of topological relatedness for matching (i.e., functionally related) node pairs (blue), for non-matching, i.e., functionally unrelated node pairs (red), and for 10 random samples of the same size as the set of matching pairs, averaged (purple).

of the (training plus testing) data in order to predict *novel* knowledge from the same data.

Just like any supervised classification approach in the context of any problem, for highest accuracy in the context of the NA problem, TARA should ideally be trained for each new pair of networks considered. That is, TARA when trained on one pair of networks (using only the training portion of the data), is expected to be at least as accurate when tested on the same pair of networks (using only the testing portion of the data) than when tested on a different pair of networks.

Of course, training TARA on a new pair of networks of interest is possible only if there exists data on whether a node pair is functionally related or not, for those networks. If such data does not exist, i.e., if one cannot determine for the new networks of interest whether a node pair is functionally related or not, then one must



apply a TARA instance pre-trained on a different pair of networks to the new pair of networks.

So, here, we investigate how well TARA performs in this task, i.e., how generalizable it is. Namely, we examine whether we can train TARA on one pair of networks and apply the trained TARA instance to a new pair of networks to still accurately predict functional knowledge. Specifically, we apply TARA, trained on the yeast and human PPI networks as described in Section “4.1.2 – Data” (“2017 networks”), to more recent yeast and human PPI networks from the same database (“2020 networks”). The new networks come from BioGRID version 3.5.181 accessed in February 2020; like for the 2017 networks, we again only include physical interactions.

Details of our experimental setup for this analysis are as follows.

- We repeat the exact same *training* process as before, i.e., on the 2017 networks. Namely, on these networks, we create the same 10 balanced datasets, and, for a given balanced dataset, for each ground truth-rarity dataset and  $y\%$  training amount, we split the data into  $y\%$  training and  $(100 - y)\%$  testing. Then, we train a logistic regression classifier using the GDVdiff feature for a node pair based on the 2017 networks. So, after these steps, we have trained TARA on the same node pairs and features vectors as before.
- But then, we perform *testing* on the 2020 networks. That is, of the node pairs in the 2017-network-based testing set from the previous bullet, we keep only those pairs in which both nodes are present in both the 2017 and 2020 network data. For the resulting node pairs, we compute their new node pair feature vectors based on the 2020 (rather than 2017) networks. We feed the new 2020-network-based feature vectors into the 2017-network-trained classifier, and add any node pair predicted as functionally related to the 2020-network-based alignment. Finally, this alignment is used in the protein function prediction framework. In this way, we can fairly compare results between the 2017-network-based alignments (computed in previous sections) and the corresponding 2020-network-based alignments (computed as just described), since all training is done on the same node pairs with the same 2017-network-based feature vectors, and the testing only differs in which network set (2017 versus 2020) the feature vectors were extracted from. Algorithm 4 outlines this process.
- Repeating for each balanced dataset, we obtain 10 precision, recall, and F-score values, and we average over the 10 values for each measure.

---

**Algorithm 4** Applying a pre-trained TARA instance to a new pair of networks. Given two networks  $G_1$  and  $G_2$ , the set of conditions  $R$  that a node pair needs to satisfy to be in a given ground truth dataset, the set of conditions  $R'$  that a node pair needs to satisfy to be considered not functionally related, random seed state  $d$ , a  $y$  percent training amount, and two networks  $G_3$  and  $G_4$ , return an alignment of  $G_3$  and  $G_4$ . For notations and their meanings, see Table 4.1.

---

- 1: Initialize set  $A$  to be empty.
- 2: let  $S_p, S_n = \text{bal}(G_1, G_2, R, R', d)$ , as computed by Algorithm 2.
- 3: let  $Tr_p = \text{random.sample}(S_p, \lfloor y|S_p|/100 \rfloor, d)$ .
- 4: let  $Te_p = S_p \setminus Tr_p$
- 5: let  $Tr_n = \text{random.sample}(S_n, \lfloor y|S_n|/100 \rfloor, d)$ .
- 6: let  $Te_n = S_n \setminus Tr_n$ .
- 7: let  $Tr = Tr_p \cup Tr_n$ .
- 8: let  $Te = Te_p \cup Te_n$ .
- 9: Train a predictive function,  $\text{LogReg} : \mathbb{R}^{73} \rightarrow \{0, 1\}$ , with logistic regression, on  $Tr$ , where the feature vector of node pair  $s_{ij} \in Tr$  is given by  $g(s_{ij}, G_1, G_2)$ , and the label of  $s_{ij}$  is  $\{1 \text{ if } s_{ij} \in S_p, 0 \text{ if } s_{ij} \in S_n\}$ .
- 10: **Lines 1-9 are identical to those of Algorithm 3, representing the fact that the training processes are identical (assuming that  $G_1, G_2, R, R', d$ , and  $y$  do not change between Algorithms 3 and 4).**
- 11: **for** each  $s_{ij} \in Te$  **do**
- 12:   **if**  $s_{ij} \in V_3 \times V_4$  **then**
- 13:     let  $\mathbf{x}_{ij} = g(s_{ij}, G_3, G_4)$
- 14:     **if**  $\text{LogReg}(\mathbf{x}_{ij}) = 1$  **then**
- 15:        $A.\text{add}(s_{ij})$
- 16:     **end if**
- 17:   **end if**
- 18: **end for**
- 19: **return**  $A$

For example, if  $G_1$  is the 2017 yeast PPI network,  $G_2$  is the 2017 human PPI network,  $R$  is the set of conditions for the atleast3-EXP ground truth dataset,  $R'$  is the set of conditions for a protein pair to be considered not functionally related (i.e., shared no GO terms of any kind),  $d$  is 0,  $y$  is 90%,  $G_3$  is the 2020 yeast PPI network, and  $G_4$  is the 2020 human PPI network,  $\text{getAln}^*(G_1, G_2, R, R', d, y, G_3, G_4)$  returns the alignment between the 2020 yeast and human PPI networks generated by a classifier trained on functionally related protein pairs defined by  $R$ , functionally unrelated pairs defined by  $R'$ , using a random seed state of 0 for sampling, and 90% of the 2017-network-based data for training.

---

Our findings are as follows. By looking at the number of protein function predictions, TARA applied to the 2020 networks generally results in somewhat fewer predictions compared to TARA applied to the 2017 networks. Even if the two TARA versions are equally accurate, the differing number of predictions alone could naturally result in the former having somewhat higher precision and somewhat lower recall. Indeed, this is exactly what we observe (Fig. 4.9). A key result is that the two TARA versions are quite comparable, i.e., that TARA trained on the 2017 networks, when it is tested on the 2020 networks, results in pretty similar (somewhat higher) precision and (somewhat lower) recall values as when it is tested on the 2017 networks. This is extremely encouraging, as it indicates that TARA *is* generalizable in our considered test.

#### 4.1.4 Conclusion

We present TARA as a method that challenges the assumption of current NA methods that topologically similar nodes are functionally related. We have shown that given the topological feature vector of a pair of nodes, TARA can accurately predict whether the nodes are functionally related. In other words, we have designed a method that can detect from training data a pattern between topological relatedness and functional relatedness in both synthetic and real-world networks. Then, taking pairs predicted as functionally related from the testing data as an alignment, we have shown that TARA generally outperforms or complements existing approaches, even those that use sequence similarity-based anchor links across network as input (unlike TARA), in the task of protein function prediction, one of the ultimate goals of NA. As such, TARA provides researchers with a valuable data-driven approach to NA and protein function prediction.

To our knowledge, TARA is the first data-driven NA approach. As such, it is just a proof-of-concept. There are many directions in which this work can be taken. For

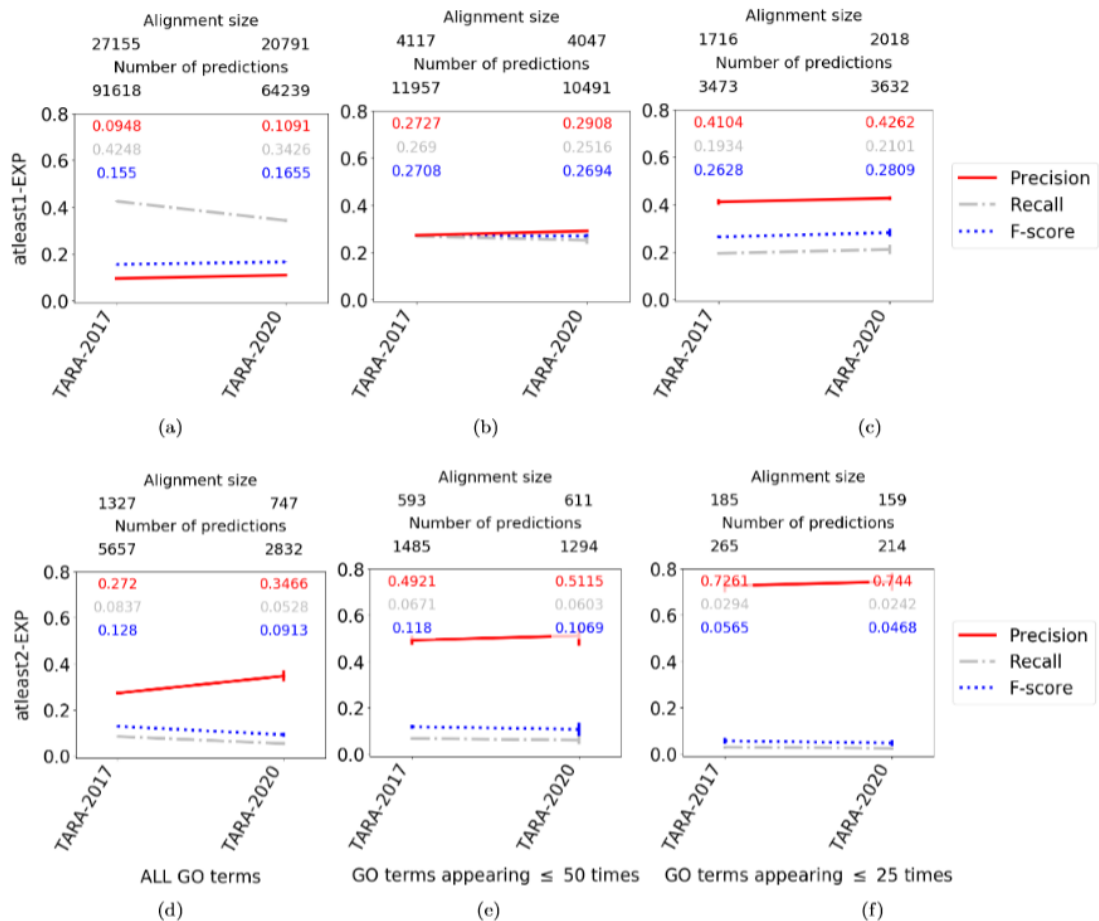


Figure 4.9. Comparison of TARA on the 2017 versus 2020 networks for rarity thresholds (a, d) ALL, (b, e) 50, and (c, f) 25 using ground truth datasets (a, b, c) *atleast1-EXP* and (d, e, f) *atleast2-EXP* in the task of protein function prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method. For example, the alignment for TARA-2017 in panel (a) contains 27,155 aligned yeast-human protein pairs, and predicts 91,618 protein-GO term associations. Raw precision, recall, and F-score values are color-coded inside each panel. Results for *atleast3-EXP* are shown in Supplementary Fig. C.11.

one, we use a relatively simple GDV-based feature of a node pair. However, more sophisticated combinations of GDVs could be explored. Other embedding methods (i.e., ways to extract feature vectors of nodes) such as matrix factorization [77] or

graph convolution networks [180] could show improvement. Also, including sequence similarity-based anchor links like PrimAlign does, is promising, especially given the fact that combining topological and sequence information seems to have compounding effects. Also, we train a simple classifier – logistic regression – but potential improvement could be seen with more sophisticated models. Furthermore, in this study we have focused on pairwise, homogeneous, and static NA. However, there has been work in aligning multiple [74, 166, 163, 81], heterogeneous [70, 111], or dynamic [165, 162, 6] networks. Our general framework could be adapted to each of these types of NA.

## 4.2 TARA++: Data-driven network alignment that integrates topology and sequence to predict function

### 4.2.1 Introduction

While TARA relies only on network topological information to generate alignments, in this section we introduce TARA++, which combines both topological and protein sequence information to generate alignments. An additional categorization of NA methods will aid in understanding our extension of TARA to TARA++. Namely, there are three NA method groups based on how input data are processed, within-network-only, isolated-within-and-across-network, and integrated-within-and-across-network, described in Table 4.5 and Section “4.2.2 – Description of existing NA methods”.

Before describing our extensions of TARA to TARA++, we recount how each of WAVA, SANA, and PrimAlign compares to TARA in terms of algorithmic steps, in order to show where there is room for improvement (Table 4.6). Recall that by learning topological relatedness patterns, TARA outperformed, in the task of across-species protein functional prediction between yeast and human, three state-of-the-

TABLE 4.5

THREE NA METHOD GROUPS BASED ON HOW INPUT DATA ARE  
PROCESSED

NA method group	Description
Within-network-only	Given two PPI networks, each node’s feature is calculated using only the topological information within the given node’s own network, hence the group name. The nodes’ topological features, which summarize the nodes’ extended PPI network neighborhoods, are then used in various alignment processes (Section “4.2.2 – Description of existing NA methods”). For state-of-the-art NA methods from this group, the topological features are based on graphlets [112], which are subgraphs, i.e., small building blocks of networks.
Isolated-within-and-across-network	Given two PPI networks and also sequence information for nodes across networks, each node’s topological feature is calculated in the same way as by within-network-only methods, and <i>only afterwards</i> is the sequence information combined with the topological features. The group name comes from the fact that both within-network topological and across-network sequence information are used, but the two are initially processed in isolation from each other and are combined only after the fact. Then, the combined data are used in various alignment processes (Section “4.2.2 – Description of existing NA methods”). Note that within-network-only methods can easily be used as isolated-within-and-across-network methods when sequence information is available; the latter lead to better alignments than the former [110].
Integrated-within-and-across-network	Given two PPI networks and sequence information for nodes across networks, the two networks are first “integrated” into one by adding across-network “anchor” links (edges) between the highly sequence-similar proteins and <i>only then</i> is any feature extraction or alignment done. So, the third group uses both within-network topological and across-network sequence information. But, they first integrate the two data types and only then process them, hence the group name.

art NA methods, WAVE [158], SANA [103], and PrimAlign [87]. TARA, WAVE, and SANA are all within-network-only methods. They also all use graphlet-based topological node features. Their key difference is that TARA is supervised, ie., it uses topological relatedness, while WAVE and SANA are unsupervised, i.e., they use topological similarity. Thus, WAVE and SANA were the most fairly comparable methods to TARA. So, we could fairly evaluate whether moving from WAVE’s and SANA’s topological similarity to TARA’s supervision-based topological relatedness helped. TARA significantly outperformed WAVE and SANA, so we could conclude that it did help. PrimAlign is one of very few existing integrated-within-and-across-network methods. Because PrimAlign was already shown to outperform many isolated-within-and-across-network methods [87] on the *exact same data* as in TARA’s evaluation [68], there was no need to evaluate TARA against any methods of that type. Importantly, TARA still outperformed PrimAlign, despite the former being a within-network-only method and hence not using any sequence information, unlike the latter. This already showed how powerful the supervised NA paradigm is. In this study, we push the boundary further. TARA “only” showed that going from unsupervised to supervised for within-network-only methods improved alignment accuracy. But, we already know that going from within-network-only to isolated-within-and-across-network in the unsupervised context improves accuracy [110], and that going from isolated-within-and-across-network to integrated-within-and-across-network in the unsupervised context further improves accuracy [87]. So, a method that is both supervised and of the integrated-within-and-across-network type should be the “best of both worlds”. Thus, here, we propose the first ever method of this type.

TABLE 4.6

CATEGORIES THAT RELEVANT NA METHODS BELONG TO

NA method	Method group	Feature type	(Un)supervised?
WAVE	Within-network-only	Topology (graphlets)	Unsupervised
SANA	Within-network-only	Topology (graphlets)	Unsupervised
PrimAlign	Integrated-within-and-across-network	Topology (PageRank-like) and sequence	Unsupervised
TARA	Within-network-only	Topology (graphlets)	Supervised
TARA-TS	Integrated-within-and-across-network	Topology (graphlets) and sequence	Supervised
TARA++	N/A (TARA++ is the overlap of TARA’s and TARA-TS’s predicted protein-GO term annotations)		

#### 4.2.1.1 Our contributions

We introduce TARA-TS (TARA within-network Topology and across-network Sequences information) as a novel method implementing the above idea. Then, for reasons discussed below, we integrate TARA and TARA-TS into our final method, TARA++. Fig. 4.10 summarizes key ideas behind TARA-TS and our evaluation framework.

Like TARA, TARA-TS is supervised. Unlike TARA and like PrimAlign, TARA-TS extracts features from an integrated yeast-human network. As a solution to feature extraction, we leverage the extensive research on graph representation learning [22], which embeds nodes of a network into a low dimensional space such that network structure is preserved; the low-dimensional node representations are then used as node features. Network embedding has primarily been studied on the methodological side in the domains of graph theory and data mining/machine learning, and on the application side in the domain of social networks [37, 64, 22]. So, given recently



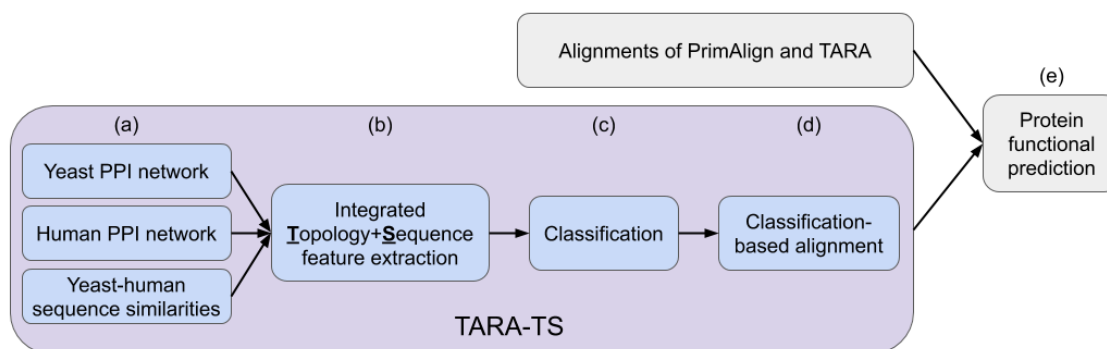


Figure 4.10. Summary of TARA-TS and our evaluation framework. **(a)** TARA-TS aims to align two networks (in this study, yeast and human PPI networks). Besides the networks, TARA-TS also uses sequence similar yeast-human protein pairs as anchor links. See Section “4.2.2 – Data”. **(b)** From the networks and anchor links, TARA-TS builds an integrated yeast-human network and extracts integrated topology- and sequence-based features of node (protein) pairs. See Section “4.2.2 – TARA-TS’s feature extraction methodology”. **(c)** Given the features, TARA-TS trains a classifier on a training set to learn what features distinguish between functionally related and functionally unrelated node pairs, and then the classifier is evaluated on a testing set. To perform this classification, yeast-human node pairs are labeled. If the two nodes in a given pair are functionally related (intuitively, share GO terms), they are labeled with the positive class; if they are functionally unrelated, they are labeled with the negative class. See Section “4.2.2 – Data”. Then, the set of labeled node pairs is split into training and testing sets to perform the classification. Only if classification accuracy is high, i.e., if TARA-TS accurately predicts functionally (un)related nodes to be functionally (un)related, does it make sense to use TARA-TS to create an alignment for protein functional prediction. **(d)** Node pairs from the testing set that are predicted as functionally related are taken as TARA-TS’s alignment. Note that relying on testing data only to create an alignment avoids any circular argument. See Section “4.2.2 – TARA-TS’s classification and alignment generation”. **(e)** Any alignment, of TARA-TS or an existing NA method such as PrimAlign and TARA, can be given to a protein functional prediction framework to predict protein-GO term annotations. Then, the different methods’ alignments are evaluated in terms of their prediction accuracy (we also evaluate their running times). See Section “4.2.2 – Using an alignment for protein functional prediction”.

recognized promise of network embedding in the domain of computational biology [124], we apply it to this domain. Namely, TARA-TS generalizes a prominent network embedding method that was proposed for within-a-single-network machine learning tasks such as node classification, clustering, and link prediction, to the across-network task of biological NA. Given the node features extracted by network embedding, TARA-TS works just as TARA to produce an alignment. Then, we use this alignment for across-species protein functional prediction.

We compare prediction accuracy of TARA-TS (pairwise, global, many-to-many, integrated-within-and-across-network, supervised) with accuracies of TARA and PrimAlign, as they are state-of-the-art NA methods that were already shown to outperform many other existing NA methods on the *exact same data* as what we use here. So, by transitivity, if TARA-TS is shown to be superior to TARA and PrimAlign, this will mean that TARA-TS is superior to the other existing methods as well. Also, of all existing methods, TARA and PrimAlign are the most similar and thus fairly comparable to TARA-TS. Namely, TARA is pairwise, global, many-to-many, and supervised, like TARA-TS. The difference is that TARA is a within-network-only method while TARA-TS is an integrated-within-and-across-network method (Table 4.6). PrimAlign is a pairwise, global, many-to-many, and integrated-within-and-across-network method, like TARA-TS. The difference is that PrimAlign is unsupervised while TARA-TS is supervised (Table 4.6). So, we can fairly test the effect of going from unsupervised to supervised for integrated-within-and-across-network methods.

When we compare TARA-TS against TARA, we actually compare whether using across-network sequence information on top of within-network topological information leads to more accurate predictions, as we expect. Surprisingly, we find that TARA-TS and TARA are almost equally as accurate. Closer examination reveals that their quantitatively similar results are *not* because the two methods are predict-

ing the same information (which would make one of them redundant). Instead, their predicted protein functional annotations are quite complementary. So, we then look at those predictions (protein-GO term associations) that are made by both methods, only those predictions made by TARA-TS but not TARA, and only those predictions made by TARA but not TARA-TS. We find the former (the overlapping predictions) to be more accurate than the predictions made by any one of TARA-TS or TARA alone. Thus, we take this overlapping version of TARA-TS and TARA as our final method, TARA++. In a sense, TARA++ is integrating state-of-the-art research knowledge across computational biology and social network domains, by combining TARA’s graphlet-based topology-only features with TARA-TS’s embedding-based topology-and-sequence features, each of which boosts the other’s performance. Very few studies have explored such a promising direction to date [124]. Importantly, we find that TARA++ not only outperforms TARA but also PrimAlign.

## 4.2.2 Methods

### 4.2.2.1 Data

As typically done in NA studies, we analyze yeast and human PPI networks. We consider the exact same PPI networks of yeast (5,926 nodes and 88,779 edges) and human (15,848 nodes and 269,120 edges) that were analyzed and publicly provided by the authors of the PrimAlign study [87]. These networks were also used in the TARA study [68]. All of this allows us to fairly compare results across all of the methods. The two networks contain only physical PPIs, without multi-edges or self-loops.

Similarly, as anchor links between proteins across the networks, we use the exact same 55,594 yeast-human sequence-similar protein pairs that were analyzed and publicly provided by the authors of the PrimAlign study [87]. These had been produced as follows [80]. All-versus-all sequence comparison using BLASTP [5] was performed on human, mouse, fruit fly, worm, and yeast. Only protein pairs with E-value se-

quence similarities  $\leq 10^{-7}$  had been kept for further consideration, which yielded 55,594 yeast-human protein pairs with such E-values.

Our supervised NA framework requires knowledge about whether two proteins are functionally related. We next outline the procedure for determining functional relatedness, which mirrors the steps from our past TARA study [68]. As typically done, we define functional relatedness using GO annotation data (from August 2019). Considering biological process GO terms and experimentally inferred protein-GO term annotations (evidence codes EXP, IDA, IPI, IMP, IGI, or IEP), if at least  $k$  GO terms are shared between a yeast protein and a human protein, we define that protein pair as functionally related. We vary  $k$  from 1 to 3. These are values of  $k$  that are typically analyzed, because even in unsupervised and especially in supervised NA studies, larger values of  $k$  result in insufficiently many pairs of functionally related nodes [96, 68]. Regardless of the  $k$  value, we define a protein pair as functionally unrelated if the two proteins share no GO terms *of any kind*. This gives the *atleast1-EXP*, *atleast2-EXP*, and *atleast3-EXP* ground truth datasets.

Traditionally, NA studies have considered all GO terms available in a given ground truth dataset. However, it is well known that not all GO terms are “created equally”, meaning that a GO term that is more general and thus higher in the GO tree hierarchy is more likely to annotate a given number of proteins compared to a more specific GO term that is lower in the hierarchy. This is why it might be worth considering only specific-enough GO terms. As a way to deal with this in the context of NA, recent work proposed accounting for the frequency of GO terms (for a given GO term, the number of proteins in the data under consideration that are annotated by that term) [76]. Indeed, in our TARA study, we found that considering rarer (i.e., more specific) GO terms led to higher protein functional prediction accuracy [68]. So, here, we consider the same three GO term rarity thresholds as in the TARA study: (i) all GO terms (i.e., ALL), which corresponds to traditional NA evaluation, (ii) more specific

GO terms that appear 50 times or fewer (i.e., threshold of 50), and (iii) even more specific GO terms that appear 25 times or fewer (i.e., threshold of 25).

For a given GO term rarity threshold, all GO terms not satisfying the threshold are filtered out. Then, for each *atleast $k$ -EXP* ground truth dataset, only proteins that share at least  $k$  GO terms from the remaining list are considered to be functionally related, and still, proteins that share no GO terms, regardless of rarity, are considered to be functionally unrelated. For example, proteins that share at least two (experimentally inferred biological process) GO terms, such that each GO term annotates 25 or fewer proteins, are considered functionally related in the “*atleast2-EXP* at the 25 GO term rarity threshold” dataset. There is a total of nine such “ground truth-rarity” datasets, resulting from combinations of the three *atleast $k$ -EXP* ground truth datasets and the three GO term rarity thresholds.

#### 4.2.2.2 TARA-TS’s feature extraction methodology

TARA-TS needs to extract features that capture both within-network topological and across-network sequence information from the integrated network, which consists of 21,774 nodes (5,926 yeast + 15,848 human proteins) and 413,493 edges (88,779 yeast PPIs + 269,120 human PPIs + 55,594 anchor links). We examine several feature extraction approaches.

First, we use the same graphlet-based feature extraction method as TARA, simply applied to the integrated network rather than the two individual networks; for technical details about the graphlet features that we use, see Supplementary Section C.2.1.1. In this way, we can test whether going from TARA’s within-network-only approach to TARA-TS’s integrated-within-and-across-network approach improves NA accuracy. We refer to this version of TARA-TS as “TARA-TS (graphlets)”.

Second, we apply a prominent network embedding method based on random walks to the integrated network to extract features, namely *node2vec* [66]; for technical

details about node2vec and why we use node2vec over other network embedding methods, see Supplementary Section C.2.1.1. We refer to this version of TARA-TS as “TARA-TS (node2vec)”.

Third, node2vec does not capture heterogeneous information in the integrated network, i.e., does not distinguish between different types of nodes (yeast and human) or edges (yeast PPIs, human PPIs, and yeast-human sequence-based anchor links). So, we also test metapath2vec [41], which essentially is node2vec generalized to heterogeneous networks. Intuitively, this approach uses “metapaths” to capture the heterogeneous information, which define the types of nodes that should be visited by random walks; for technical details about metapath2vec, see Supplementary Section C.2.1.1. We refer to this version as “TARA-TS (metapath2vec)”.

Henceforth, we refer to TARA-TS (graphlets), TARA-TS (node2vec), and TARA-TS (metapath2vec) as different “TARA-TS versions”. If we just say “TARA-TS”, the discussion applies to all three versions.

In theory, the heterogeneous information could be captured not just via metapaths but also via heterogeneous graphlets [70] (versus homogeneous graphlets discussed thus far). However, in practice, heterogeneous graphlet counting is infeasible for as large networks as studied in this paper, due to its exponential computational complexity. This is not an issue for homogeneous graphlet counting because methods such as Orca [79] rely on combinatorics to infer the counts of some (larger) graphlets from the counts of other (smaller) graphlets, significantly reducing the computational complexity. However, no publicly available implementation of combinatorial relationships for counting heterogeneous graphlets exists. Similar holds for a method that *directly* extracts the feature vector of a *node pair* [82], versus extracting graphlet features of *individual nodes* and then combining these, as TARA does: no combinatorial approach for direct node pair graphlets exists. Instead, current heterogeneous and

node pair graphlet counting require exhaustive graphlet enumeration and are thus infeasible.

Lastly, we discuss why we do not use feature vectors from PrimAlign, the next most comparable method to TARA [68] that already integrates within-network topological and across-network sequence information. This is because PrimAlign’s algorithmic design does not allow for feature vector extraction. As discussed in more detail in Section “4.2.2 – Description of existing NA methods”, PrimAlign models the integrated network as a Markov chain, which is then repeatedly transitioned until convergence. This means that the weights between every node pair are updated at the same time, based on the weights of every node pair from the previous state of the chain. So, PrimAlign operates on every node pair *at once* with respect to their weights, rather than on *individual nodes or node pairs* with respect to any kind of feature vector, meaning that we cannot easily extract such information.

#### 4.2.2.3 TARA-TS’s classification and alignment generation

We must first evaluate whether TARA-TS can correctly predict nodes as functionally (un)related. If not, there would be no point to use it to form an alignment. To evaluate this, we train and test a classifier as follows.

For a given ground truth-rarity dataset (Section “4.2.2 – Data”), the positive class consists of functionally related node pairs, and the negative class consists of functionally unrelated node pairs. Because the latter is much larger, we create a balanced dataset by undersampling the negative class to match the size of the positive class, as typically done [157]. Due to randomness in sampling, we create 10 balanced datasets and repeat the classification process for each, averaging results over them.

For a given balanced dataset, we split it into two sets:  $y$  percent of the data is randomly sampled and put into one set, and the remaining  $(100 - y)$  percent is put into the other set. This sampling is done with the constraint that in each of

the two sets, 50% of the data instances have the positive class and 50% have the negative class. Again, due to randomness in sampling, we repeat this 10 times to create 10 data splits of  $y/(100 - y)$  percent and repeat the classification process for each, averaging results over them.

For a given  $y/(100 - y)$  split, we train a logistic regression classifier on the set containing  $y$  percent of the data (the training set). We use this trained classifier to predict on the remaining  $(100 - y)$  percent of the data (the testing set), measuring the accuracy and area under receiver operating characteristic curve (AUROC).

In summary, for a given  $y$ , for each balanced dataset, we have 10 accuracy and 10 AUROC scores, corresponding to the 10 data splits; for each measure, we compute the average over the 10 splits, obtaining a single accuracy and single AUROC. Then, for a given  $y$ , given the single accuracy and single AUROC for each balanced dataset, i.e., given 10 accuracy and 10 AUROC scores for the 10 balanced datasets, for each measure, we compute the average over the 10 balanced datasets to obtain a final accuracy and a final AUROC score for that  $y$ . In our evaluation, we vary  $y$  from 10 to 90 in increments of 10; each variation is called a “ $y$  percent training test”. This allows us to test how the amount of training data affects the results, which is important because in many real-world applications, not much data may be available for training.

Only if the average accuracy and AUROC are high, i.e., if TARA-TS accurately predicts functionally (un)related nodes to be functionally (un)related, does it make sense to use TARA-TS to create an alignment for protein functional prediction. If this is the case, we create an alignment as follows. Given one  $y/(100 - y)$  split and the classifier trained on it, we take every node pair from the testing set that is predicted as functionally related and add it to the alignment. Here, it is important to only use the testing set for the alignment. This way, because there is no overlap between node pairs in the testing set and node pairs in the training set, the alignment will



not contain any node pairs that were trained on. Consequently, this avoids a circular argument when constructing TARA-TS’s alignment. For simplicity, we do not repeat this process for all data splits, as we found that the split choice had no major effect on the classification performance. We only use the “first” one, which in our implementation corresponds to a starting seed of 0 for Python’s random number generator when performing sampling. We have a total of 270 alignments, corresponding to all combinations of the 3 TARA-TS versions, the 9 percent training tests, and the 10 balanced datasets.

#### 4.2.2.4 Using an alignment for protein functional prediction

An ultimate goal of biological NA is across-species protein functional prediction, so each NA method must be evaluated in this context. We use a (TARA-TS’s or an existing method’s) alignment in an established protein functional prediction framework [110], as follows. Suppose that we are evaluating an alignment for the ground truth-rarity dataset *atleastk*-EXP at the  $r$  GO term rarity threshold (e.g., *atleast2*-EXP at the 25 GO term rarity threshold). Let us define “relevant GO terms” as all GO terms in that ground truth-rarity dataset. Then, the framework makes predictions for each protein  $u$  in the alignment that is annotated by at least  $k$  relevant GO terms (i.e., for each protein for which a prediction can actually be made at that ground truth-rarity dataset). To do so, first, the framework hides  $u$ ’s true GO term(s). Then, for each relevant GO term  $g$ , the framework determines if the alignment is significantly “enriched” in  $g$ . The hypergeometric test is used for this, in order to calculate if the number of aligned node pairs in which the aligned proteins share  $g$  is significantly high (see below). If so, then node  $u$  is predicted to be annotated by GO term  $g$ . Repeating for all applicable proteins and GO terms results in the final list of predicted protein-GO term associations. From this prediction list, the framework calculates the precision (percentage of the predictions that are in a given ground

truth-rarity dataset) and recall (percentage of the protein-GO term association from a given ground truth-rarity dataset that are among the predictions).

The hypergeometric test works as follows. If  $Y$  is the set of all yeast proteins and  $H$  is the set of all human proteins, then let  $M = \{(y, h) \in Y \times H \mid \text{each of } y \text{ and } h \text{ is annotated by at least } k \text{ relevant GO terms}\}$ . Let  $N = \{(y, h) \in M \mid \text{each of } y \text{ and } h \text{ is annotated by } g\}$ . Note that  $g$  refers to the same GO term as in the previous paragraph. If  $A$  is the alignment of interest, let  $O = \{(y, h) \in A \mid \text{each of } y \text{ and } h \text{ is annotated by at least } k \text{ relevant GO terms}\}$ . Finally, let  $P = \{(y, h) \in O \mid \text{each of } y \text{ and } h \text{ is annotated by } g\}$ . Then, the  $p$ -value resulting from the hypergeometric test is the probability of seeing  $|P|$  or more successes (i.e., node pairs that share  $g$ ) if we randomly choose  $|O|$  elements from  $M$  given that  $M$  contains  $|N|$  successes (for example, in Python, this would correspond to `1 - scipy.stats.hypergeom.cdf(|P| - 1, |M|, |O|, |N|)`).

We ensure that there is no circular argument when predicting an annotation between a protein  $u$  and a GO term  $g$  from the alignment of interest, even if this particular annotation might have been used to construct the training data. Namely, to predict protein  $u$  as being annotated by GO term  $g$ ,  $u$  must have been aligned to some protein  $v$  that also has GO term  $g$ , in order for the functional knowledge  $g$  to be transferred from  $v$  to  $u$ . For this to happen, node pair  $(u, v)$  must have appeared in the testing data (and been predicted as functionally related, thus being placed into the alignment). This means that  $(u, v)$  could not have appeared in the training data, because the training and testing data do not overlap (Section “4.2.2 – TARA-TS’s classification and alignment generation”). Even if some other node pair  $(u, w)$ , where both  $u$  and  $w$  are annotated by  $g$ , appears in the training data, which could happen only if  $u$  is annotated by  $g$  and  $w$  is annotated by  $g$ , the prediction from the alignment of  $u$  having  $g$  could not have originated from the pair  $(u, w)$  that the alignment was trained on. Instead, this prediction must have originated from node

pair  $(u, v)$  that is not in the training data. This avoids a circular argument when predicting protein-GO term annotations.

#### 4.2.2.5 Description of existing NA methods

Here, we describe existing NA methods to explain why we ultimately compare against TARA and PrimAlign out of all existing methods.

We discuss within-network-only and isolated-within-and-across-network methods first. They have two parts. Initially, similarities are computed for all pairs of nodes across networks. For within-network-only methods, these are topological similarities (computed by comparing the nodes’ topological features). For isolated-within-and-across-network methods, these are a weighted sum of the nodes’ topological and sequence similarities. Then, an alignment strategy aims to maximize the total similarity over all aligned nodes while also conserving many edges. Two types of alignment strategies exist. One type is “seed-and-extend”, which progressively builds an alignment by adding to it one node pair at a time. WAVE [158], when paired with graphlet-based topological similarities, is a state-of-the-art method of this type. The other type is a “search algorithm” that optimizes an objective function over the solution space of possible alignments. We pioneered search algorithm-based NA with MAGNA and MAGNA++ [143, 164]. The more recent SANA [103] is a state-of-the-art approach of this type, whose objective function is generally graphlet-based.

Next, we discuss integrated-within-and-across-network NA methods. PrimAlign [87] is a state-of-the-art method of this type. After linking networks being aligned via anchors, PrimAlign creates a Markov chain out of the integrated network, converting the edge weights to transition probabilities (in an unweighted network, the weights are set to 1 before converting to transition probabilities). The chain is then transitioned repeatedly until it converges, which redistributes the across-network node pair scores

using a PageRank-like algorithm. Node pairs across networks that are above some threshold are outputted as the alignment.

MUNK also links the original networks via anchors, but it uses matrix factorization to obtain an alignment [57]. In our preliminary analyses, MUNK’s similarity scores could not distinguish between functionally related and functionally unrelated proteins. Furthermore, Nelson et al. [124] found IsoRank [154] to outperform MUNK, despite the former being an early method and the latter a recent method. IsoRank was already outperformed by many NA methods that appeared after it, which in turn were outperformed by WAVE and SANA, which were then outperformed by TARA and PrimAlign (see below). Thus, because we compare against TARA and PrimAlign in this study, there is no need to also compare against MUNK.

Unlike TARA++, the previously mentioned methods do not use functional (GO) information to produce alignments but only to evaluate them. DualAligner [144] does use such information, but not to determine classification labels (“functionally related” and “functionally unrelated”) like TARA++ does. Instead, the method aligns groups of nodes that are all annotated with a given GO term, and then seeds-and-extends around these groups to match proteins that do not have any GO annotations, resulting in the final alignment. We do not consider DualAligner in this study, as it is quite old (from 2014). More recent, state-of-the-art methods have appeared since [53, 72].

The above methods are unsupervised. Many other such methods exist [72]. TARA and PrimAlign, which we consider in this study, already outperformed the other methods, including AlignMCL, AlignNemo, CUFID, HubAlign, IsoRankN, L-GRAAL, MAGNA, MAGNA++, MI-GRAAL, NETAL, NetCoffee, NetworkBLAST, PINALOG, SANA, SMETANA, and WAVE [158, 103, 87, 68]. In turn, these outperformed GHOST, IsoRank, NATALIE, PISwap, and SPINAL [103]. This, plus TARA and PrimAlign being the most similar and thus fairly comparable to TARA++, is

why we focus on these two existing methods. Also, some supervised methods (besides TARA, already discussed) exist, as follows.

IMAP [23] uses supervised learning differently than TARA++. As input, IMAP requires a starting (unsupervised, topological similarity-based) alignment between two networks; as such, it still suffers from the topological similarity assumption. Then, it obtains graphlet features for node pairs. Node pairs from the starting alignment form the positive class, while the other node pairs are sampled to form the negative class. Then, IMAP trains a linear regression classifier on these two classes. After, this data is “re-classified”, but instead of assigning a class, IMAP assigns a score corresponding to the probability that the two nodes should be aligned. A matching algorithm (e.g., Hungarian) is applied to these scores to form a new alignment, which is then fed back to IMAP. This process iterates while alignment quality improves. We did try to test IMAP. Its code was not available. Our attempts at implementing IMAP ourselves led to significantly worse results than those reported in the IMAP paper. So, we could not consider IMAP in our evaluation.

MEgo2Vec [180], also supervised, is a social NA method for matching user profiles across different online media platforms. Features of user profiles are obtained using graph neural networks and natural language processing techniques, and these are used to train a classifier to predict whether two profiles from different platforms correspond to the same person. A big part of MEgo2Vec is the various natural language processing techniques to match users’ names, affiliations, or research interests, meaning that it cannot be easily applied to PPI networks.

### 4.2.3 Results and discussion

#### 4.2.3.1 Comparison of TARA-TS versions

**Classification.** Here, we study classification performance of the three TARA-TS versions (graphlets, node2vec, metapath2vec) and TARA, i.e., how correctly they

predict as functionally (un)related the protein pairs from testing data in a given  $y$  percent training test. We would ideally do this on all nine ground truth-rarity datasets. However, two of them, atleast3-EXP at the 50 and 25 thresholds, are too small for TARA-TS and TARA to perform any classification on; data scarcity is a general challenge that machine learning methods face though, and not specific to just TARA-TS and TARA. Thus, we have seven viable ground truth-rarity datasets.

Due to space constraints, we discuss the effect of various parameters ( $k$  in atleast $k$ -EXP, GO term rarity threshold, and  $y$  percent training test) on the classification performance of a given TARA-TS version, for each version, in Supplementary Section C.2.2.1. Instead, here we focus on comparing the three TARA-TS versions and TARA.

We expect all TARA-TS versions to have higher accuracy and AUROC than TARA, as they extract topology plus sequence features from the integrated yeast-human network, unlike TARA, which extracts topology features only within each individual network. However, we find that this is not always the case (Fig. 4.11(a) and Supplementary Figs. C.13–C.14): (i) The *relative* accuracy change of TARA-TS (graphlets) over TARA ranges from -3% (decrease) to 5% (increase), depending on the atleast $k$ -EXP ground truth dataset, GO term rarity threshold, and  $y$  percent training test, with an average change of 0%; and its relative AUROC change ranges from -3% to 5%, with an average change of 1%. (ii) TARA-TS (node2vec) does always improve over TARA though. Its relative accuracy change over TARA ranges from 6% to 27%, with an average change of 14%; and its relative AUROC change ranges from 9% to 32% with an average change of 16%. (iii) As for TARA-TS (metapath2vec), we also see improvement over TARA, though not as large as for TARA-TS (node2vec). In particular, the relative accuracy change of TARA-TS (metapath2vec) over TARA ranges from -1% to 14% with an average change of 6%; and its relative AUROC change ranges from 2% to 15%, with an average change of 7%.

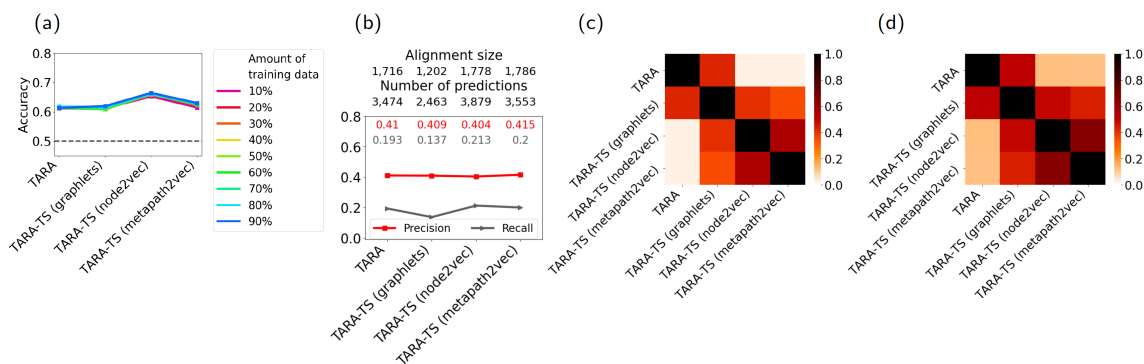


Figure 4.11. Comparison of the three TARA-TS versions and TARA. Comparison of the three TARA-TS versions and TARA for GO term rarity threshold 25 and ground truth dataset atleast1-EXP, in terms of: **(a)** classification accuracy, **(b)** protein functional prediction accuracy, **(c)** overlap between aligned yeast-human protein pairs, and **(d)** overlap between predicted protein-GO term associations. In panel (b), the alignment for e.g., TARA contains 1,716 aligned protein pairs and predicts 3,474 protein-GO term associations. In panels (c)-(d), the pairwise overlaps are measured via the Jaccard index. Panel (a) encompasses all  $y$  percent training tests. Panels (b)-(d) are for the 90% training test. Comparisons of different metapath choices for metapath2vec can be found in Supplementary Fig. C.12. Results for the other ground-truth rarity datasets and percent training tests are shown in Supplementary Figs. C.13–C.19.

Overall, we find that in terms of classification performance, TARA-TS (node2vec) performs the best, followed by TARA-TS (metapath2vec), and followed by TARA-TS (graphlets) and TARA that are tied; all four perform significantly better than at random (Supplementary Figs. C.13–C.14).

**Protein functional prediction.** Here, we evaluate the protein functional prediction accuracy of alignments of the three TARA-TS versions and TARA. Per discussion in Supplementary Section C.2.2.1, for each of TARA-TS and TARA, different  $y$  percent training tests have only marginal differences in classification accuracy. For this reason, henceforth, for simplicity, we only consider the 10, 50, and 90 percent training tests; 10 and 90 allow us to test the extremes and 50 allows us to test the middle. Recall

that classification cannot be performed on two (small) ground truth-rarity datasets, atleast3-EXP at thresholds 50 and 25, so no alignments exist for them, and thus protein functional prediction is not possible. So, for each TARA-TS version and TARA, we have 21 evaluation tests, resulting from combinations of the seven viable ground truth-rarity datasets and the three selected  $y$  values.

First, we analyze each TARA-TS version. The following three trends are expected in terms of each version’s performance. (i) Precision will likely increase and recall will likely decrease as the amount of training data goes from 10 to 50 to 90. We expect precision to increase because a classifier trained on a larger dataset will potentially be more accurate. Consequently, the testing dataset will be smaller. So, the alignment produced by a given method version will contain fewer node pairs. This in turn is expected to yield fewer predictions and thus to decrease recall. (ii) Precision will likely increase and recall will likely decrease as the requirement for functional relatedness becomes more stringent, i.e., as the value of  $k$  in the atleast $k$ -EXP ground truth datasets goes up. Namely, increase in precision is expected because at a larger  $k$  value, training is done on more reliable data. Decrease in recall is expected because at a larger  $k$  value, there will be less data overall, and hence less testing data. So, a similar argument as in point (i) above applies. (iii) Precision will likely increase and recall will likely decrease as the GO term rarity threshold decreases, i.e., as rarer GO terms are considered. This is based on the observation that rarer GO terms may be more meaningful [76, 68], leading to smaller but higher quality data. As such, higher precision and lower recall are expected for similar reasoning as in points (i) and (ii) above. We find that all three expected trends hold for all TARA-TS versions (Supplementary Figs. C.15–C.17).

Second, we compare the performance of the three TARA-TS versions and TARA. Interestingly, even though TARA-TS (node2vec) has superior classification performance (Fig. 4.11(a)), all four methods yield almost equal protein functional pre-



diction accuracy (Fig. 4.11(b) and Supplementary Figs. C.15–C.17). Further unexpected is that TARA-TS has similar accuracy to TARA, despite the former using sequence information that TARA does not. We take a closer look at the alignments and predictions made by each method to see if the different methods are aligning the same nodes, or predicting the same protein-GO term associations. So, we investigate how much their alignments overlap (Fig. 4.11(c)), and how much their predictions overlap (Fig. 4.11(d)). We find that the different methods are all aligning and predicting at least somewhat different information from each other. Yet, their predictions are equally accurate. Furthermore, we find that TARA is more similar to (i.e., overlaps the most with) TARA-TS (graphlets) than to TARA-TS (node2vec) and TARA-TS (metapath2vec), which makes sense since the former uses graphlets to extract feature vectors like TARA, and the latter two do not. Moreover, TARA-TS (node2vec) and (metapath2vec) are more similar to each other than to the other methods, which is also expected since they both use a similar random walk-based feature extraction method.

It is surprising that TARA-TS (graphlets) does not improve upon TARA, i.e., that the additional sequence information does not improve upon only topological information. It is also surprising that TARA-TS (metapath2vec) does not improve upon TARA-TS (node2vec) – both use a similar random walk-based embedding process, but metapath2vec additionally accounts for the heterogeneous information in the integrated network. We discuss potential reasons for these two unexpected findings in Section “4.2.3 – Discussion”.

Because TARA-TS (node2vec) not only yields the best classification performance, predicting functional (un)relatedness the best out of all TARA-TS versions, but also captures the most novel protein functional information compared to TARA (i.e., the predictions it makes overlap the least to those made by TARA out of all TARA-TS

versions), we continue only with TARA-TS (node2vec) as the selected TARA-TS version.

#### 4.2.3.2 TARA-TS versus TARA in the task of protein functional prediction: toward TARA++

Focusing on TARA-TS (node2vec) as the selected TARA-TS version (i.e., simply as TARA-TS), we zoom into the comparison between it and TARA. The two methods have different alignments and make different predictions (Fig. 4.12), so how can they still have similar protein functional prediction accuracy? To answer this, we look at the precision and recall of predictions made by both methods, only those predictions made by TARA-TS but not TARA, and only those predictions made by TARA but not TARA-TS (Fig. 4.12(b) and Supplementary Fig. C.21). From this, we highlight two findings. First, graphlets, which use only topological information, perform as well as network embedding features that use both topological and sequence information. This is supported by the fact that predictions made only by TARA and only by TARA-TS produce similar accuracy in almost all evaluation tests. Second, predictions made by both methods are significantly more accurate than predictions made by any one method alone. We discuss these findings further in Section “4.2.3 – Discussion”.

Because the overlap of predictions of TARA-TS and TARA has such high accuracy, we take it as our new TARA++ method, which we consider further. For simplicity, for each of the seven viable ground truth-rarity datasets, we consider either TARA++10, TARA++50, or TARA++90 as a representative percent training test. That is, we pick seven “TARA++ versus existing methods” evaluation tests from the 21 total. We choose the percent training tests that represent TARA++’s best results. Namely, we look for the percent training test with high precision (predictions are accurate) as well as a large number of predictions (maximize the bio-

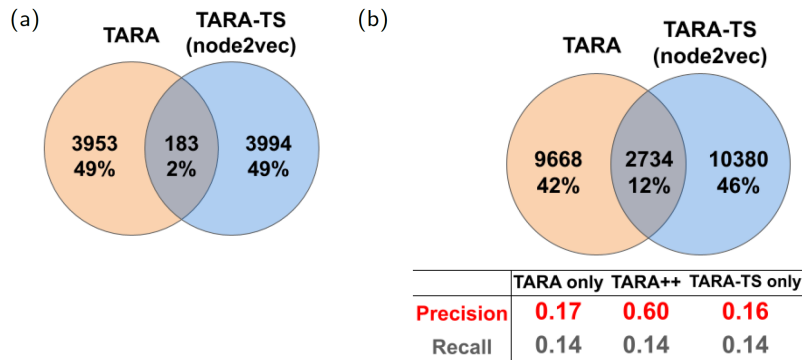


Figure 4.12. Comparison of TARA-TS and TARA in terms of their alignment and prediction overlaps. Comparison of the selected TARA-TS version and TARA for GO term rarity threshold 50, ground truth dataset atleast1-EXP, and the 90% training test, in terms of overlap between their: **(a)** aligned yeast-human protein pairs and **(b)** predicted protein-GO term associations. In panel (b), precision and recall are shown for each of the three prediction sets captured by the Venn diagram; TARA++’s predictions are those in the overlap. The overlaps are for one of the 10 balanced datasets; so, the alignment size and prediction number of a method may differ from those in Fig. 4.11(b), where the statistics are averaged over all balanced datasets. Results for the other ground truth-rarity datasets are shown in Supplementary Figs. C.20–C.21.

logical knowledge uncovered). So, we choose TARA++90 for all ground truth-rarity datasets except atleast2-EXP at the 50 and 25 rarity thresholds, where we choose TARA++10. Henceforth, we refer to all of the selected TARA++ versions simply as TARA++.

#### 4.2.3.3 TARA++ versus existing NA methods in the task of protein functional prediction

**Accuracy.** We compare TARA++’s predictions against those of two most fairly comparable state-of-the-art methods, TARA and PrimAlign. Also, we consider predictions resulting from using only sequence information, Sequence. Here, we treat the 55,594 anchor links by themselves as the alignment; as no topological informa-

tion is used, this is not an NA method. With TARA and Sequence, we can separately analyze each aspect, i.e., within-network topological information only and across-network sequence information only, and evaluate how each compares to our integrative TARA++. (TARA++’s predictions are by definition a subset of TARA’s predictions, and so we expect TARA++ to have higher precision but lower recall than TARA.) With PrimAlign, we can evaluate how this integrative but unsupervised method compares to our also integrative but supervised TARA++. Importantly, TARA and PrimAlign were already shown to outperform many previous NA methods (Section “4.2.2 – Description of existing NA methods”). So, comparing to these two methods is sufficient. Also, keep in mind that like with TARA, a theoretical recall of 1 is not necessarily possible with TARA++. This is because for a given training/testing split, TARA++ uses a part (up to 90%) of the ground truth functional data for training, and so for that split, it is impossible to make predictions for the training data portion, i.e., to transfer functional knowledge from node  $u$  to node  $v$  when the node pair  $(u, v)$  is in the training data.

Both precision and recall are important. However, in the biomedical domain, if one has to choose between the two measures, we believe that precision is favored. As an illustration, let us compare the following two scenarios: (i) making 30 predictions of which 27 are correct, i.e., having a small number of mostly correct predictions, and (ii) making 300 predictions of which 100 are correct, i.e., a large number of mostly incorrect predictions. The former, having higher precision but lower recall than the latter, is more viable for potential wet lab validation.

Our key results are as follows (Fig. 4.13 and Supplementary Fig. C.22). In terms of precision, TARA++ is the best for 6 out of 7 ground truth-rarity datasets. It is only slightly inferior to PrimAlign for 1 out of 7 datasets (atleast1-EXP for ALL GO terms), but TARA++ has much higher recall than PrimAlign on this dataset. Speaking of recall, TARA is expected to always outperform TARA++, and this is

what we observe. Of the remaining existing methods, TARA++ is the best for 2 out of 7 datasets – atleast1-EXP at the ALL and 50 rarity thresholds – even though TARA++ makes much *fewer* predictions than the next best method, Sequence. For the other datasets, TARA++’s recall is lower than that of PrimAlign and Sequence. This is expected, since TARA++ makes fewer predictions than the other methods. Importantly, the difference in recall between TARA++ and every other method is relatively small, for example only 0.06 lower on average compared to TARA, while TARA++ is much better in terms of precision than every other method, for example 0.2 greater on average compared to TARA. As discussed above, such a trade-off between precision and recall is worth it for our task.

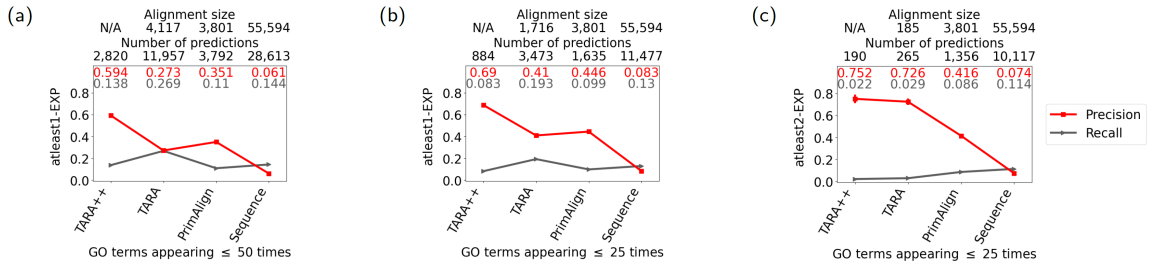


Figure 4.13. Comparison of TARA++ and three existing methods in the task of protein functional prediction. Comparison of TARA++ and three existing methods in the task of protein functional prediction, for rarity thresholds (a) 50 and (b, c) 25, and for ground truth datasets (a, b) atleast1-EXP and (c) atleast2-EXP. The alignment size (the number of aligned yeast-protein pairs) and number of functional predictions (predicted protein-GO term associations) are shown for each method, except that TARA++ does not have an alignment *per se*. i.e., TARA++ comes from the overlap of *predictions* made by TARA and TARA-TS; hence the “N/A”s. For TARA++ and TARA, results are averages over all balanced datasets; the standard deviations are small and thus invisible. Results for the other ground truth-rarity datasets are shown in Supplementary Fig. C.22.

We see that the precision of TARA++ is much greater than just the sum of TARA’s and Sequence’s precisions, suggesting that integrating within-network topological and across-network sequence information has compounded effects. This further highlights the need for such approaches.

In the above analyses, we account for the default number of predictions made by each method for the given ground truth-rarity dataset. These numbers do not necessarily match between the different methods. Consequently, TARA++ may have high precision simply because it makes the fewest number of predictions. Nonetheless, when we enforce that each method produces the same number of predictions, we again find that TARA++ is the best of all considered NA methods in a majority of all evaluation tests, in terms of both precision and recall (Fig. 4.14, Supplementary Section C.2.2.2, and Supplementary Fig. C.23).

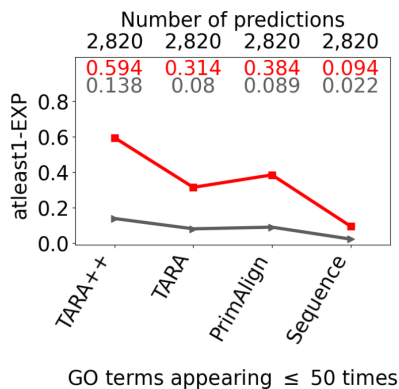


Figure 4.14. Comparison of TARA++ and three existing methods when all make the same number of predictions. Representative results (for one ground truth-rarity dataset) comparing TARA++ and three existing methods in the same way as in Fig. 4.13(a) except that here all methods make the same number of predictions. The remaining results (for the other ground truth-rarity datasets) are shown in Supplementary Fig. C.23.

Combining TARA and TARA-TS into TARA++ results in such high accuracy. So, we also investigate the overlap between TARA++ and PrimAlign (Fig. 4.15 and Supplementary Fig. C.24). The number of overlapping predictions is small, suggesting complementarity between TARA++ and PrimAlign. However, TARA++ still has an advantage when it comes to predicting protein function, as the predictions made only by TARA++ have higher precision for 6 out of 7 ground truth-rarity datasets compared to those made only by PrimAlign. Importantly, the overlap between predictions of TARA++ and PrimAlign has much higher precision than either alone. This is not totally unexpected for reasons discussed in Section “4.2.3 – Discussion”.

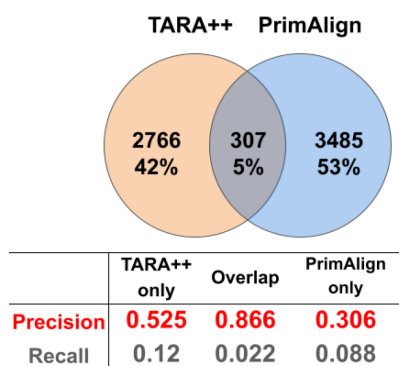


Figure 4.15. Comparison of TARA++ and PrimAlign in terms of their prediction overlaps. Representative results (for GO term rarity threshold 50 and ground truth dataset atleast1-EXP) comparing TARA++ and PrimAlign in the same way as TARA and TARA-TS are compared in Fig. 4.12(b). The remaining results (for the other ground truth-rarity datasets) are shown in Supplementary Fig. C.24.

**Running time.** We analyze the time needed for TARA-TS, TARA, and PrimAlign to compute an alignment when considering the ALL GO term rarity threshold; this

threshold is the worst case (slowest) out of all studied thresholds since it has the most data. As TARA++ comes from the intersection of TARA-TS’s and TARA’s results, its time is either the maximum or sum of TARA-TS’s and TARA’s, if the two are run at the same time or one after the other, respectively. We find the following (also, see Supplementary Table C.2).

As expected, we observe that TARA-TS’s running time decreases as  $k$  (in atleast  $k$ -EXP) increases, since there is less data overall, and thus less data to train on. When comparing TARA-TS and TARA, the former is faster, and this comes from the feature computation time, as both use the same supervised framework. TARA-TS’s node2vec computation is expectedly faster than TARA’s graphlet counting even when using Orca for two reasons. First, the random walks produced by node2vec can be thought of as sampling the network structure, which is much faster than capturing the full network structure like graphlets do.

Second, node2vec is parallelized while Orca is not. Parallelization benefits node2vec a lot: the same number of random walks is performed for each node (parameter `-r:`), so no single node takes much longer than any other. However, for graphlet counting, nodes with e.g., high degrees are the limiting time factor, and so parallelization would not help as much. Also note that TARA-TS’s (and PrimAlign’s) running time is missing the step of computing sequence-based anchor links; these anchors were precomputed and provided by the PrimAlign study. So, TARA-TS (and PrimAlign) has an unfair advantage over TARA. Despite this missing step, regardless of how TARA-TS and TARA are combined to form TARA++, PrimAlign will still be faster. However, it is about half as precise as TARA++. Even though TARA++ is slower, it is still practically feasible. Thus, the extra time is worth the almost doubling of precision.

**TARA++’s robustness to data noise.** Lastly, we investigate TARA++’s robustness to noise (i.e., random perturbation) in the data, since one cannot expect



all real-world data (even the PPI networks we use!) to be perfect. When we incrementally introduce noise in the data, ranging from 0% (original data) to 100% (completely random), we find that TARA++ is fairly robust up to 50% noise (Fig. 4.16 and Supplementary Section C.2.2.3). Beyond 50%, precision and recall drop and eventually reach 0, as expected.

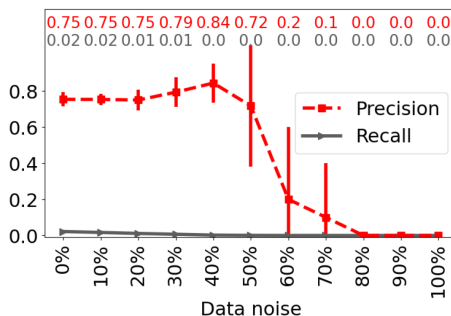


Figure 4.16. Robustness of TARA++ to data noise. Robustness of TARA++ protein functional prediction accuracy as data noise increases from 0% to 100%, for GO term rarity threshold 25 and ground truth dataset atleast2-EXP.

#### 4.2.3.4 Discussion

Recall the two unexpected findings from Section “4.2.3 – Comparison of TARA-TS versions”. Namely, first, it is surprising that TARA-TS (graphlets) does not improve upon TARA, i.e., that the additional sequence information does not improve upon only topological information. A reason may be that the across-network sequence information complements, rather than enhances, the within-network topology information. Some of the predictions made by TARA-TS (graphlets), specifically

those that overlap with TARA’s, may be due to the within-network topology information used by both methods, and the remaining predictions made by TARA-TS (graphlets) may be due to the across-network sequence information, which is not used by TARA. Second, it is surprising that TARA-TS (metapath2vec) does not improve upon TARA-TS (node2vec). Both use a similar random walk-based embedding process, but metapath2vec additionally accounts for the heterogeneous information in the integrated network. The lack of improvement may be because the additional information captured by the considered metapaths is not useful in this task, or because constraining random walks by node type leads to less neighborhood structure being explored. For example, at some point in a random walk, a human node may have many human neighbors, but the walk is forced to move to a yeast node due to the metapath constraints. Then, the neighborhood of that human node will not be well explored. However, because the number of possible metapaths to test in order to find the best one(s) is exponential with respect to the length of the path, it is not feasible to test every possibility, even for short lengths. Thus, an efficient way of selecting appropriate metapaths for a given network would be necessary to continue to pursue metapath-based embedding methods for this task. However, to our knowledge no such selection process exists, which is why we do not pursue this problem beyond the metapaths we have considered.

Also recall the two interesting findings from Section “4.2.3 – TARA-TS versus TARA in the task of protein functional prediction: toward TARA++”. Namely, first, graphlets, which use only topological information, perform as well as network embedding features that use both topological and sequence information. This motivates the need to develop better graphlet-based methods for integrated networks as future work. Second, predictions made by both TARA and TARA-TS are significantly more accurate than predictions made by any method alone. In a sense, their overlap is integrating state-of-the-art research across the computational biology and

social network domains, by combining TARA’s graphlet-based topology-only features with TARA-TS’s embedding-based topology-and-sequence features. So, the overlapping predictions combine the strengths of both domains, showing promise for future domain-crossing endeavors.

Finally, recall from Section “4.2.3 – TARA++ versus existing NA methods in the task of protein functional prediction” that the overlap between predictions of TARA++ and PrimAlign has much higher precision than either alone. This is not totally unexpected, as it suggests that predictions made by multiple methods (as already seen when combining TARA and TARA-TS into TARA++) are the most reliable; adding PrimAlign further strengthens this observation. Also, this echoes the promise of ensemble methods in machine learning. As such, further exploration of integrating different approaches beyond the simple overlapping of their predictions may be fruitful.

#### 4.2.4 Conclusion

TARA and TARA-TS are among the first *biological* NA methods that use supervised learning, despite the introduction of supervised *social* NA methods in recent years. This could be because the study of biological NA began well before the current era of “big data” [88, 89], making unsupervised approaches the traditional option. However, as the amount of biological network data continues to increase, developing data-driven approaches is an important direction. Especially fruitful for the task of NA is integrating research knowledge across biological and social network domains, as we have shown by combining TARA and TARA-TS into TARA++. Namely, TARA++ outperforms state-of-the-art NA methods in the task of protein functional prediction, an ultimate goal of NA. Though, it is still important to note that data-driven approaches are limited when data is scarce. As such, more sophisticated “ensembling” procedures for integrating protein functional prediction approaches to-

gether, beyond the simple overlapping of their predictions as we explored with TARA and TARA-TS (into TARA++), and PrimAlign, could potentially mitigate these limitations and open up new research directions.

As TARA++ is the first data-driven NA method to integrate topological and sequence information, it is just a proof-of-concept. This work can be taken further. We found that graphlet-based features on the isolated networks (on topological information alone) perform as well as embedding-based features on the integrated network (on topological and sequence information combined), even though the latter (using more data) was expected to be better. Thus, developing a graphlet feature that would efficiently deal with an integrated network could yield further improvements. This might include novel algorithms for speeding up counting of heterogeneous graphlets in large data. Heterogeneous graphlets, or heterogeneous network embedding features other than metapath2vec, could better distinguish between different node/edge types in an integrated network and thus only improve over the features considered in this study. Also, we focused on NA of static networks. However, research in NA of dynamic (e.g., aging- or disease progression-related) networks is becoming popular [165, 162]. So, our framework can be adapted to such novel NA categories.

## CHAPTER 5

### MODELING MULTI-SCALE DATA VIA A NETWORK OF NETWORKS

The work in this chapter is discussed in the following paper:

- **Shawn Gu**, Meng Jiang, Pietro H. Guzzi, and Tijana Milenković (2021), Modeling multi-scale data via a network of networks, under review. Also, arXiv:2105.12226 [q-bio.MN]. [71]

#### 5.1 Introduction

Recall that biological NA in the context of protein functional prediction focuses on the PPI network level. However, the finer-grained protein 3D structural level is important as well, as such structures have important implications for proteins' functions. Modeling these structures as PSNs, and taking them together with a PPI network, a multilevel NoN (Figure 1.5) naturally arises. To answer whether this data integration is actually worth it, we analyze whether NoN-based entity label prediction is more accurate than each of single-level node label prediction and graph label prediction alone.

Specifically, since the entities of interest are represented by level 2 nodes and, correspondingly, modeled as level 1 networks, entity label prediction can refer to (i) using only the level 2 network (Fig. 1.5(a)) to predict level 2 nodes' labels, corresponding to the task of node label prediction in the level 2 network, (ii) using only level 1 networks (Fig. 1.5(b)) to predict level 1 networks' labels, corresponding to the task of graph label prediction using the level 1 networks, and (iii) using the

entire NoN to predict entities' labels. Thus, the primary question we aim to answer is whether (iii) is more accurate than (i) and (ii).

#### 5.1.1 Our contributions

In tackling this question, we make the following novel contributions: we construct and provide two new sources of NoN data, we develop novel approaches for NoN label prediction, and, most importantly, we are the first to test whether using NoN data in label prediction is more accurate than using only a single level. Next, we discuss each of these contributions.

Since to our knowledge labeled NoNs are limited, we provide two new sources of such data. First, we develop an NoN generator that can create a variety of synthetic NoNs (Section 5.2.3.1). Intuitively, given any set of single-level random graph generators, such as geometric [133] or scale-free [10], our NoN generator combines random graphs created from these single-level generators at each level. In this way, we can label each entity (level 2 node and its level 1 network) based on which combination of single-level random graph generators it is involved in at the two levels. Our generator can control a variety of network structural parameters (Section 5.2.3.1), thus allowing for the mimicking of a variety of real-world systems. Second, we construct a biological NoN, consisting of a PPI network from BioGRID [156] at the second level and PSNs for proteins from Protein Data Bank (PDB) [19] at the first level. Proteins are labeled based on their functions via Gene Ontology (GO) annotation data [8] (Section 5.2.3.2). For each of the 131 GO terms considered, the goal is to predict whether or not each protein is annotated by that GO term. While computational protein functional prediction is relatively well-studied, the problem is still very relevant, as the accuracy of existing methods for this purpose is typically low. The continued importance of computational annotation of protein function [60] is a major motivator of this study. We expect the NoN data resulting from this study to be-

come a useful resource for future research in both network science and computational biology, including for the problem of protein function prediction.

We also develop novel approaches for NoN label prediction. In general, label prediction approaches extract features of the entities and then perform supervised classification, i.e., prediction of the entities' labels based on their features. So, for this study, there are three types of approaches to consider: (i) those that extract node-level features (i.e., level 2 only), (ii) those that extract network-level features (i.e., level 1 only), or (iii) those that extract NoN features (i.e., integrated level 1 and level 2). To our knowledge, approaches of type (iii) do not exist yet, so we create NoN features in two ways: by combining existing node- and network-level features and by applying the novel graph neural network (GNN) approach that we propose for analyzing NoNs.

Then, we aim to evaluate whether approaches of type (iii) outperform those of types (i) and (ii). If so, this would provide evidence that NoN-based data integration is useful for label prediction. To determine which approach types are the best, we evaluate them in terms of accuracy for synthetic NoNs, as class sizes are balanced, and in terms of the area under the precision recall curve (AUPR), precision, recall, and F-score for the biological NoN, as class sizes are unbalanced.

For synthetic NoNs, we find that our NoN approaches outperform single-level node and network ones for those NoNs where the majority of nodes are not densely interconnected (i.e., where nodes do not tend to group into densely connected modules). For NoNs where there are groups of densely interconnected nodes (i.e., where there is clustering structure), an existing single-level approach performs as well as NoN approaches. For the biological NoN, we find that our NoN approaches outperform the single-level ones in a little under half of the GO terms considered. Furthermore, for 30% of the GO terms considered, only our NoN approaches make meaningful predictions, while node- and network-level ones achieve random accuracy. Also, while deep

learning does not perform the best *overall*, it seems to be useful for otherwise difficult-to-predict protein functions. As such, NoN-based data integration is an important and exciting direction for future research.

### 5.1.2 Related work

Finally, it is important to discuss a few related topics. Given that we study a biological NoN, we must point out that existing studies have combined protein structural data with PPI data [132, 179] for various tasks. However, they generally do so by incorporating more basic *non-network* structural properties, such as proteins’ domains and families, with PPI data. On the other hand, our approaches combine PSN representations of detailed 3D protein structural properties with PPI network data through the NoN representation. Importantly, PSN-based models of protein structures have already been shown to outperform non-network-based (i.e., traditional sequence and “direct” 3D structural) models in tasks such as protein structural comparison/classification [55, 125] and protein functional prediction [63, 15]. Thus, we hypothesize that incorporating state-of-the-art, i.e., PSN-based (rather than traditional sequence or “direct” 3D structural), representations of protein structures with PPI network data into an NoN will be effective. Regardless, the goal of this study is to investigate *network-based* data integration by evaluating whether NoN label prediction is actually more accurate than each of node- and network-level alone. A comparison with other, *non-network-based* data integration schemes is outside the scope of the current study and the subject of future work.

Some other network models of higher-order data exist as well. These include: multiplex, multimodal, multilevel, and interdependent networks [141, 119, 25, 97, 40, 134], which are sometimes used interchangeably and sometimes also referred to as “networks of networks”; hierarchical networks [31]; higher-order networks [172]; hypergraphs [18]; and simplicial complexes [121]. However, these all model different



types of data compared to NoNs as we define them, so we cannot consider these other network types in this study.

There are also studies that do model data as NoNs. However, they differ from our proposed work in terms of data analyzed, application domain, and/or network science task. With respect to data, besides synthetic NoNs, we analyze a PPI network-PSN biological NoN. However, these other studies analyze NoNs where the level 2 network is a disease-disease similarity network and the level 1 networks are disease specific PPI networks [127], where the level 2 network is a social network and the level 1 networks are individuals' brain networks [56, 129, 12], or where the level 2 network is a chemical-chemical interaction network and level 1 networks are molecule networks [168]. With respect to application domain, while we aim to predict protein function, these other studies aim to identify disease causing genes [127], answer sociologically motivated questions like whether similarities between friends mean they have similar ways of thinking [129], or predict new chemical-chemical interactions [168]. With respect to network science task, while we aim to predict entities' labels, these other studies aim to identify important entities (level 1 nodes) [127], predict links between entities (level 2 nodes) [168], or embed multiple networks at the same level into a common low dimensional space, using an NoN as an intermediate step [42]. While it *might* be possible to extend some of these existing studies to ours or vice versa, doing so could require considerable effort, as it would mean developing new methods, and code is not publicly available for all of the existing methods. All of this makes any potential extensions hard. As such, we cannot compare against these existing NoN-like methods.

## 5.2 Methods

### 5.2.1 Network of networks definition

We define an NoN with  $l$  levels as follows. Let  $G^{(l)} = (V^{(l)}, E^{(l)})$  be the level  $l$  network with node set  $V^{(l)}$  and edge set  $E^{(l)} \subseteq V^{(l)} \times V^{(l)}$ . Each “level  $l$  node”  $v_i^{(l)} \in V^{(l)}$  itself corresponds to a “level  $l - 1$  network”  $G_i^{(l-1)} = (V_i^{(l-1)}, E_i^{(l-1)})$ . In other words,  $V^{(l)}$  and  $\{G_1^{(l-1)}, \dots, G_{|V^{(l)}|}^{(l-1)}\}$  are different notations that represent the same underlying concept – the set of entities that are represented by nodes in a level  $l$  network and correspondingly modeled as level  $l - 1$  networks. Note that we allow each level  $l - 1$  network to contain no nodes (and thus no edges). That is,  $G_i^{(l-1)}$  can be an order-zero graph, signifying that  $v_i^{(l)}$  has no corresponding level  $l - 1$  network. We assume that nodes from different level  $l - 1$  networks do not overlap – for example, amino acids (nodes) from different PSNs do not represent the same physical entities, even if the types of the amino acids are the same. That is,  $V_1^{(l-1)} \cap \dots \cap V_{|V^{(l)}|}^{(l-1)} = \emptyset$ . Each level  $l - 1$  node  $v_{ij}^{(l-1)} \in V_i^{(l-1)}$  in each level  $l - 1$  network  $G_i^{(l-1)} \in V^{(l)}$  itself corresponds to a level  $l - 2$  network  $G_{ij}^{(l-2)} = (V_{ij}^{(l-2)}, E_{ij}^{(l-2)})$ . This recursion continues until level 1. We illustrate a two-level NoN in Fig. 1.5.

### 5.2.2 Problem statement

Given an NoN  $\{G^{(2)} = (V^{(2)}, E^{(2)}) \text{ and } \{G_1^{(1)}, \dots, G_{|V^{(2)}|}^{(1)}\}\}$ , its label set  $Y = y_1, \dots, y_c$ , and a function that maps entities (i.e., level 2 nodes and thus their corresponding level 1 networks) to their labels  $f_{true} : V^{(2)} \rightarrow Y$ , the goal is to learn a predictive function  $f_{pred} : V^{(2)} \rightarrow Y$  through supervised classification.

In this study, this predictive function can be learned in three ways: for each level 2 node  $v_i^{(2)}$ , using features based only on  $G^{(2)}$ , i.e., node-level features; for each level 2 node’s corresponding level 1 network  $G_i^{(1)}$ , using features based only on  $G_i^{(1)}$ , i.e., network-level features; and for each entity, using features based on both levels, i.e.,

NoN features. We aim to show that the predictive performance of  $f_{pred}$  when trained on NoN features is higher than those when using node-level and network-level features alone, thereby indicating that NoN-based entity label prediction is more accurate than each of node label prediction and graph label prediction alone. A more formal description of the evaluation, including the training, validation, and testing split; the loss function; and the performance measures, can be found in Section 5.2.5.

### 5.2.3 Data

#### 5.2.3.1 Our synthetic NoN generator

We develop a generator that can create synthetic NoNs with a variety of parameters and multiple levels. In this study, we focus on two levels. While analyzing NoNs of three or more levels would be interesting, doing so would be difficult in the context of this study, especially since available real-world NoN data of so many levels is scarce. Namely, our main goal is to test whether NoN-based integration is worth it. With two levels, there exist very clearly defined and fairly comparable tasks: NoN vs. node-level vs. graph-level label prediction. With more levels, this is no longer the case. So, we leave such investigation of NoNs with more than two levels for future work.

We want our generator to create a two-level NoN with labeled entities (i.e., level 2 nodes, and equivalently, level 1 networks) such that only an approach using information from both levels should be able to attain high entity label prediction accuracy. To accomplish this, it is first useful to understand single-level random graph models and how they generate random graphs with various properties. In particular, we consider the geometric (GEO) [133] and scale-free (SF) [10] models. An instance of GEO (i.e., a random geometric network) is created by placing nodes randomly in Euclidean space and adding an edge between those that are spatially close to each other, resulting in a network with Poisson-like degree distribution and high clustering

coefficient. An instance of SF (i.e., a random scale-free network) is created using the concept of preferential attachment to join nodes – as the network is grown, nodes with high degree are more likely to gain edges (i.e., neighbors) compared to nodes with low degree, resulting in a network with a power-law degree distribution and low clustering coefficient. So, due to the different construction schemes, GEO and SF networks are topologically distinct. As such, node label prediction approaches can easily distinguish between nodes whose network neighborhoods are GEO-like and nodes whose network neighborhoods are SF-like; we can label nodes of the former as “GEO” and nodes of the latter as “SF”. Similarly, graph label prediction approaches can easily distinguish between instances of GEO and instances of SF; we can label networks of the former as “GEO” and networks of the latter as “SF”.

So, to generate an NoN, we combine GEO and SF at the two levels. In particular, let  $(m_1, m_2)$  denote an NoN where the level 2 network is generated using random graph model  $m_2$  and each level 2 node’s level 1 network is generated using random graph model  $m_1$ . Given such an NoN, we label its entities (level 2 nodes and corresponding level 1 networks) based on the  $(m_1, m_2)$  combination, just as we label single-level nodes/networks based on which of GEO or SF they are generated with. Now suppose we generate NoNs for each  $(m_1, m_2) \in \{(GEO, GEO), (GEO, SF), (SF, GEO), (SF, SF)\}$ , i.e., all four possible combinations of GEO and SF at the two levels. An entity label prediction approach would have to incorporate information from both levels in order to accurately predict each of the four labels: if an approach only used level 1 information, it would fail to distinguish between  $(GEO, GEO)$  and  $(GEO, SF)$  since both are of type GEO at level 1, and it would fail to distinguish between  $(SF, GEO)$  and  $(SF, SF)$  since both are of type SF at level 1; level 2 information would be needed. Similarly, if an approach only used level 2 information, it would fail to distinguish between  $(GEO, GEO)$  and  $(SF, GEO)$  since both are of type GEO at level 2, and it would fail to distinguish between  $(GEO, SF)$  and

$(SF, SF)$  since both are of type SF at level 2; level 1 information would be needed. These four combinations of GEO and SF are helpful initial constructions for ultimately generating an NoN that requires information from both levels to accurately make predictions for.

In this previous example, each of  $(GEO, GEO)$ ,  $(GEO, SF)$ ,  $(SF, GEO)$ , and  $(SF, SF)$  is its own “isolated NoN”, disconnected from others. So, to more accurately model a real-world system, our generator joins each isolated NoN at the second level to form a connected NoN with multiple regions; this joining process is described below. We refer to the set of level 2 nodes that originated from an isolated NoN as a “level 2 node group”. We generate these isolated NoNs, each with a fixed number of level 1 and level 2 nodes and edges, such that when combined into a single connected NoN, the resulting NoN’s level 2 network has 15,000 nodes and 300,000 edges to approximate the size of the human PPI network, and so that each level 1 network has 200 nodes and 800 edges to approximate the average size of the PSNs. The entities (level 2 nodes and corresponding level 1 networks) of the resulting NoN have four labels, with an equal number of entities having each label, so balanced multiclass classification is performed. We visualize a toy NoN in Fig. 5.1.

Our generator joins isolated NoNs by randomly removing edges within level 2 node groups and randomly adding the same number of edges across level 2 nodes groups (*across-edge* amount). That is, we repeat the following process  $a\% \times 300,000$  times: (i) randomly select a level 2 node group, (ii) randomly select an edge in that node group, (iii) delete that edge, (iv) randomly select two level 2 nodes from different node groups, and (v) add an edge between the selected nodes. We start with  $a = 5$  to retain most of the level 2 node groups’ originally generated GEO- and SF-like network topologies, and we systematically vary  $a$  to be 25, 50, 75, and 95 to test the effect of breaking the network topologies down. This also means that at  $a = 5$  there

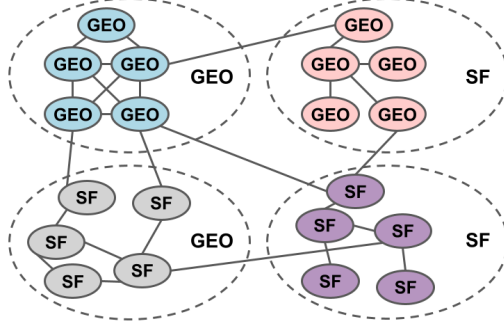


Figure 5.1. A toy synthetic NoN generated from two random graph models. Large dotted circles represent level 2 node groups (originating from isolated NoNs) whose level 2 nodes are connected in a random geometric- (GEO) or scale-free-like (SF) fashion. Small solid circles represent level 2 nodes whose level 1 networks are of the random graph type indicated. Level 1 nodes and edges are not shown. Level 2 nodes are colored based on their label, i.e., their combination of level 1 and level 2 network topology ( $\{(GEO, GEO), (GEO, SF), (SF, GEO), \text{ and } (SF, SF)\}$ ).

is significant clustering (each level 2 node group consists of densely interconnected nodes), while at  $a = 95$  there is very little clustering.

We also introduce random rewiring to test each approach’s robustness to data noise (*rewire-noise* amount). Specifically, for  $r\%$  rewire-noise, for each level 1 network and each level 2 node group, we randomly remove  $r\%$  of the total edges and randomly add the same number of edges back. We vary  $r$  to be 0 (no noise), 10, 25, 50, 75, and 100 (completely random). Combining the  $a$  and  $r$  parameters, we generate a total of  $5 \times 6 = 30$  synthetic NoNs. For a formal description of the NoN generation process and the parameters we vary, see Supplementary Section D.1.1.1.

In this study, we report results for two-model NoNs, i.e., for  $\{GEO, SF\}$ . Note that we also analyzed three-model NoNs, adding the Erdős-Rényi (ER) model [49], i.e., for  $\{GEO, SF, ER\}$ . Because results are qualitatively and quantitatively similar, we do not discuss this analysis in the paper due to space constraints.

### 5.2.3.2 Biological NoN

We also investigate whether integration is useful in the applied task of protein functional prediction. We construct a biological NoN using the human PPI network and the proteins’ associated PSNs (see also Supplementary Section D.1.1.2). We construct a PPI network using human PPI data from BioGrid [156] version 4.1.190; this PPI network has 18,708 nodes and 434,527 edges. Then, we map each protein ID to its corresponding PDB chain, resulting in 4,776 PDB chains. Finally, we construct PSNs from these chains using an established process: nodes represent amino acids and edges join two amino acids if the distance between any of their heavy atoms (carbon, nitrogen, oxygen, or sulfur) is within 6 Å [55]. The obtained biological NoN has 18,708 proteins at level 2, of which 4,776 have PSNs at level 1.

To obtain label information, we rely on protein-GO term annotation data (accessed in October 2020) [8]. Of all protein-GO term annotations, we focus on biological process (BP) GO terms in which the annotations were experimentally inferred (EXP, IDA, IPI, IMP, IGI, IEP). From those, we keep only GO terms annotating the 4,776 proteins that have PSNs, which results in 131 unique GO terms, i.e., classification labels. For each label  $g$ , proteins annotated by  $g$  constitute positive data instances. While we could consider negative data instances to be all proteins not annotated by  $g$ , this could add bias for proteins that are not annotated by  $g$  but are by GO terms related to  $g$  and would also create an extreme positive/negative imbalance. Instead, we define negative data instances to be proteins that are not currently annotated by any BP GO term, reducing the bias and resulting in more balanced classes. Ultimately, each label has between 20 and 277 positive data instances and 61 negative data instances; as there are 131 labels total, we perform binary classification 131 times (Section 5.2.5). Note that not all proteins have labels. Regardless, when extracting information from the level 2 network, we consider all 18,708 nodes and

434,527 edges. However, for each label, we only perform classification on the positive and negative data instances.

#### 5.2.4 Approaches for label prediction

We consider graph theoretic approaches that are based on graphlets [112], and graph learning approaches, namely, SIGN [138] and DiffPool [176].

Graphlets are small subgraphs (a path, triangle, square, etc.) that can be considered the building blocks of networks, and they can be used to extract features of both nodes and networks (Supplementary Section D.1.2). The graphlet-based feature of a node in a general network is called its *graphlet degree vector* (GDV), and GDVs of all nodes in a network can be collected into the network’s *GDV matrix* (GDVM) feature. One drawback of GDVM is that its dimensions depend on the size of the network – if performing graph classification of different sized networks using GDVM features, issues can arise. Thus, we also consider a transformation of GDVM, the graphlet correlation matrix (GCM) [173], which always has the same dimensions regardless of network size.

Given these definitions of graphlet features for nodes in a general network or for the entire general network itself, we now explain which features we use for nodes in a level 2 network and which features we use for level 1 networks. For the former, we extract each level 2 node’s GDV (L2 GDV). For the latter, we extract each level 1 network’s GDVM and GCM (L1 GDVM and L1 GCM). We use L1 GDVM when analyzing synthetic NoNs since we found that it outperformed L1 GCM. For the biological NoN, L1 GCM is the only viable feature since level 1 networks (PSNs) have different numbers of nodes (amino acids).

Then, to obtain NoN graphlet features, we concatenate level 2 nodes’ L2 GDVs with their networks’ L1 GDVMs or L1 GCMs. This results in five graphlet-based features: those for level 1 networks (L1 GDVM and L1 GCM) that are used for



graph label prediction, those for nodes in a level 2 network (L2 GDV) that are used for node label prediction, and those for the entire NoN (L1 GDVM + L2 GDV and L1 GCM + L2 GDV) that are used for entity label prediction. As graphlet-based feature extraction is an unsupervised task, in order to perform classification, for each graphlet-based feature, we train a logistic regression classifier (Supplementary Section D.1.4). So for example, when we say L2 GDV, we mean the L2 GDV feature under logistic regression.

SIGN aims to perform node classification (Supplementary Section D.1.4). It first computes adjacency matrix-based features and then uses them in a neural network classifier. Mathematically, SIGN can be thought of as an ensemble of shallow graph convolutional network (GCN) classifiers, which is why it is a graph learning approach. In this study, when we say L2 SIGN, we mean its adjacency matrix-based features paired with its own classifier for node classification using only a level 2 network.

DiffPool aims to perform graph classification (Supplementary Section D.1.2). For each input network, DiffPool’s GNN summarizes nodes’ initial features into a hidden feature for the entire network. Then, given hidden features corresponding to the input networks, the GNN is trained on these hidden features to perform graph classification. When we say L1 DiffPool, we mean its GNN with the initial features chosen (Supplementary Section D.1.4), for graph classification using only level 1 networks.

As SIGN and DiffPool are single-level graph learning approaches, we also combine them into an NoN graph learning approach. Given each level 2 node’s SIGN feature, we concatenate it with the level 2 node’s corresponding level 1 network’s hidden feature computed by DiffPool’s GNN. The GNN is then trained on these concatenated features to perform classification (any general purpose feature can be incorporated into DiffPool like this). When we say L1 DiffPool + L2 SIGN, we mean entity label prediction using the process described above, incorporating SIGN’s feature into

TABLE 5.1

EXISTING APPROACHES THAT WE CONSIDER AND THEIR  
GENERALIZED NON COUNTERPARTS

Single-level approaches		NoN approaches
Node-level	Network-level	
L2 GDV	L1 GDVM	L1 GDVM + L2 GDV
	L1 GCM	L1 GCM + L2 GDV
L2 SIGN	L1 DiffPool	L1 DiffPool + L2 SIGN
		Combined all (L1 GDVM)
		Combined all (L1 GCM)

“Combined all (L1 GDVM)” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN; “Combined all (L1 GCM)” is similarly named.

DiffPool’s GNN. So, we use three graph learning-based approaches: L1 DiffPool, L2 SIGN, and L1 DiffPool + L2 SIGN.

We also combine L1 GDVM + L2 GDV or L1 GCM + L2 GDV with L1 DiffPool + L2 SIGN to test whether integrating information across the graph theoretic and graph learning domains improves upon either alone. Graphlet-based features can be incorporated into DiffPool using the process described previously.

In summary, thus far, we have described five single-level approaches and five NoN approaches that we use (Table 5.1).

Next, we describe our integrative GCN-based approach. We focus on GCNs for two reasons: (i) recent work has suggested that other GNN architectures do not offer very much benefit over GCNs [148, 170, 138], making such methods more complex for

little gain and (ii) the extension of GCNs to NoNs is intuitive. Note that GCNs (and thus our extension of GCNs to NoNs) are often considered to be performing semi-supervised learning [90], as they make use of the entire network structure, including unlabeled nodes, to infer network features of nodes. But because we make predictions only for labeled nodes (rather than for both labeled and unlabeled nodes), and for simplicity, we continue to refer to our considered task of entity label prediction as supervised in this study.

The basic unit of a GCN is a graph convolutional layer. Graph convolution layers allow each node to see information about its neighbors. So, we generalize graph convolution layers to NoNs so that each node receives information not only from its neighbors (in the same level), but also from its corresponding network at a lower level or from the network it is a part of at a higher level. This would be in line with intuition that, for example, the feature of a protein should contain information about how it interacts with other proteins (i.e., its topology in the level 2 network) and structural properties of the protein itself that allow for such interactions (topology of level 1 nodes in its level 1 network). Then, we can stack multiple NoN graph convolutional layers (with intermediate layers in between) to form an NoN-GCN (Supplementary Section D.1.3). We refer to an NoN-GCN approach using  $\lambda$  layers as “GCN- $\lambda$ ”.

### 5.2.5 Evaluation

For a given NoN  $\{G^{(2)} = (V^{(2)}, E^{(2)})$  and  $\{G_1^{(1)}, \dots, G_{|V^{(2)}|}^{(1)}\}$ , its label set  $Y = y_1, \dots, y_c$ , and a function that maps level 2 nodes (and thus their corresponding level 1 networks) to their labels  $f_{true} : V^{(2)} \rightarrow Y$ , the goal is to learn a predictive function  $f_{pred} : V^{(2)} \rightarrow Y$ . We do this by first splitting the data into three disjoint sets: training ( $V_{tr}^{(2)}$ ), validation ( $V_{val}^{(2)}$ ), and testing ( $V_{te}^{(2)}$ ). Then, we train a classifier on the training set that aims to minimize the cross-entropy loss between  $f_{true}(V_{tr}^{(2)})$  and  $f_{pred}(V_{tr}^{(2)})$ . We use  $V_{val}^{(2)}$  to optimize hyperparameters and finally report the classifier’s

performance on  $V_{te}^{(2)}$ , an independent set never seen in the training process and not used for determining hyperparameters. As typically done, we form these disjoint sets using stratified sampling, repeating multiple times and averaging the results to reduce bias from the randomness of the sampling. For details on hyperparameter optimization and sampling, see Supplementary Section D.1.4.

Regarding how we measure classification performance of an approach, for synthetic NoNs, we report classification accuracy (Supplementary Section D.1.4) since class sizes are balanced. For the real-world NoNs, we report area under precision-recall (AUPR), precision@k, recall@k, and F-score@k (Supplementary Section D.1.4), since class sizes are not balanced. As commonly done, we also perform statistical tests to see whether each approach’s performance is significantly better than random, i.e., is “significant” (Supplementary Section D.1.4).

## 5.3 Results and discussion

### 5.3.1 Accuracy on synthetic networks of networks

We expect NoN approaches to outperform single-level ones. We find that at least one NoN approach (L1 GDVM + L2 GDV, L1 DiffPool + L2 SIGN, L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN, GCN-2, or GCN-3) outperforms or ties (is within 1% of) all single-level approaches (L1 GDVM, L2 GDV, L1 DiffPool, and L2 SIGN) for 30 out of the 30 synthetic NoNs (Fig. 5.2 and Supplementary Figs. D.2-D.6). Specifically, at least one NoN approach outperforms all single-level approaches for 9 out of the 30 NoNs, and at least one NoN approach is tied with L2 SIGN for 21 out of the 30 NoNs. L2 SIGN is the only single-level approach that ties NoN approaches. However, before we discuss why L2 SIGN performs as well as NoN approaches, we need to understand the effects of both across-edge amount and rewiring-noise amount.

Recall that when we increase across-edge amount, level 2 node groups' original GEO- and SF-like network topologies are increasingly broken down and eventually become entirely random. When across-edge amount is high, most edges will exist across level 2 node groups, not within (and there will be very little, if any, clustering structure in the level 2 network). Thus, approaches using only level 2 information (L2 GDV and L2 SIGN) will be making predictions on random data, and approaches that combine level 1 and level 2 information (all NoN approaches) will be making predictions on partially random data (level 1 networks are unaffected by across-edge amount). So, for the former approaches, we expect that as across-edge amount increases, prediction accuracy will drop to 0.25 (since there are four labels and class sizes are balanced, random performance is  $\frac{1}{4}$ ). For the latter approaches, we expect that prediction accuracy will drop to 0.5, for the following reason. The only signal NoN approaches can pick up when across-edge amount is high is from level 1 networks, essentially turning NoN approaches into a single-level approaches (level 1 only). And, the maximum expected accuracy of any single level approach is  $\frac{\# \text{ of models}}{\# \text{ of labels}}$ , or 0.5 (Supplementary Section D.2.1). Indeed, we observe these drops in accuracy for all approaches (Fig. 5.2(c, d) and Supplementary Figs. D.2-D.6).

Recall that we increase rewire-noise amount to investigate approaches' robustness to increasing data noise. When rewire-noise amount is high, both the level 2 node groups' and level 1 networks' original GEO- and SF-like network topologies will now be random (note, however, that clustering structure will not be affected since rewire-noise occurs within node groups, not across). So, all types of approaches will be making predictions on random data. As such, we expect that as rewire-noise amount increases, prediction accuracy will decrease. We observe these drops in accuracy for all approaches except L2 SIGN (Fig. 5.2(b, d) and Supplementary Figs. D.2-D.6), which we discuss below.

To summarize, high across-edge amount leads to a random level 2 network with very little clustering structure, and high rewire-noise amount leads to a random level 2 network with clustering structure (in addition to random level 1 networks). Since L2 SIGN performs poorly for the former (Fig. 5.2(c, d)) but well for the latter (Fig. 5.2(b)), despite the level 2 networks having random network topology in both situations, we hypothesize that L2 SIGN is able to capture the clustering structure in the level 2 network, i.e., it is able to detect the existence of densely interconnected level 2 node groups. So, L2 SIGN is able to perform as well as NoN approaches for 21 out of the 30 NoNs simply because there exists clustering structure in the level 2 networks of those 21 NoNs. L2 SIGN’s ability to capture clustering structure is also likely why at low across-edge amounts, regardless of rewire-noise amount, NoN approaches incorporating L2 SIGN perform as well as they do. This also suggests that when one expects clustering structure in the data, incorporating SIGN could help.

Above, we analyze single-level approaches versus NoN approaches as well as trends regarding across-edge amount and rewire-noise amount. However, recall that the approaches we consider come from either the graph theoretic or graph learning domain. So, we also compare the two domains. For simplicity, we focus on the NoN approaches, i.e., L1 GDVM + L2 GDV from the graph theoretic domain and L1 DiffPool + L2 SIGN from the graph learning domain, as we already know that they outperform or tie single-level approaches. We find that L1 DiffPool + L2 SIGN outperforms L1 GDVM + L2 GDV for 20 out of the 30 NoNs, is tied for 9 out of the 30 NoNs, and is worse for 1 out of the 30 NoNs. However, as discussed above, for NoNs where across-edge amount is low and rewire-noise amount is high, L1 DiffPool + L2 SIGN’s performance likely comes from L2 SIGN. We also investigate whether combining research knowledge from the graph theoretic and graph learning domains improves upon each domain individually. This does not appear to be the case on the

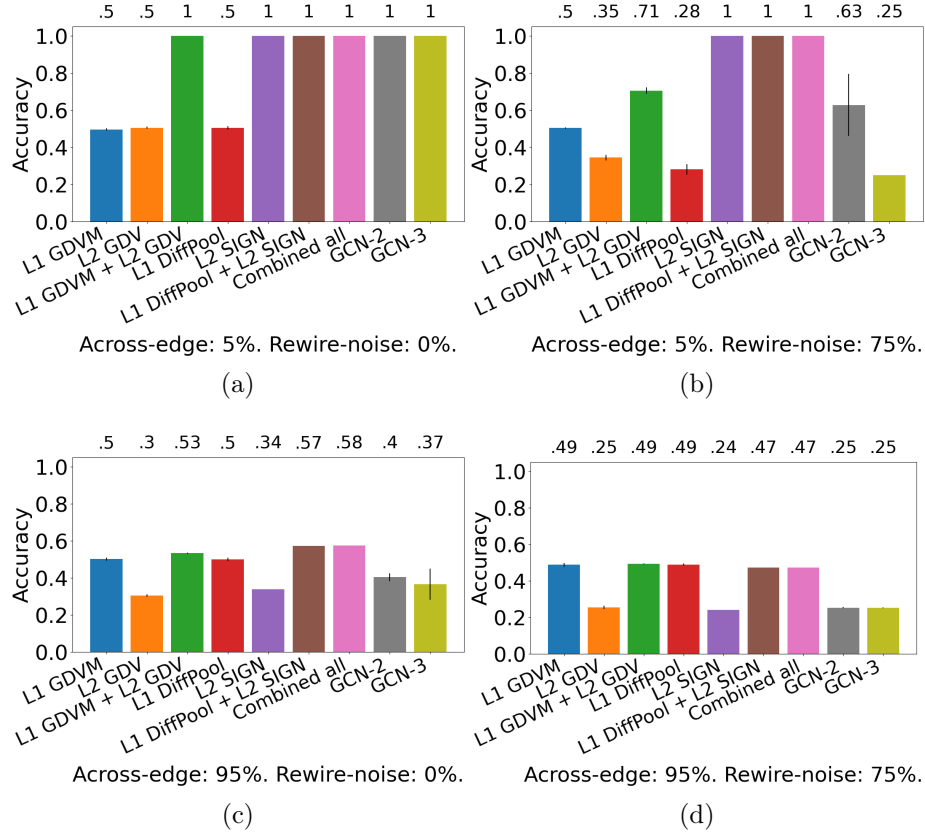


Figure 5.2: Comparison of the nine considered approaches in the task of label prediction for synthetic NoNs with the following parameters: **(a)** 5% across-edge and 0% rewire-noise amount, **(b)** 5% across-edge and 75% rewire-noise amount, **(c)** 95% across-edge and 0% rewire-noise amount, and **(d)** 95% across-edge and 75% rewire-noise amount. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars. Standard deviations are indicated at the top of each bar; some have very small values and are thus not visible. We expect an approach that only uses a single level and does not capture clustering information to have around  $\frac{\# \text{ of models}}{\# \text{ of labels}}$ , or 0.5, accuracy when both across-edge and rewire-noise amount are low (Supplementary Section D.2.1). Results for other parameter combinations are shown in Supplementary Figs. D.2-D.6.

synthetic data, as L1 DiffPool + L2 SIGN is as good as L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN for 29 out of the 30 NoNs and is worse for only one NoN (Fig. 5.2 and Supplementary Figs. D.2-D.6).

Finally, recall that our extensions of existing node/graph label prediction approaches to their NoN counterparts (L1 GDVM + L2 GDV, L1 DiffPool + L2 SIGN, L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN) are concatenation-based, which is why we developed integrative NoN-GCN approaches (GCN-2 and GCN-3) as well. Regarding the NoN-GCN approaches themselves, we expect that GCN-3 will outperform GCN-2, as the former is a deeper model. However, this is not the case, as GCN-3 only outperforms GCN-2 for 2 out of the 30 NoNs, ties for 21 out of the 30, and is worse for 7 out of the 30 (Fig. 5.2 and Supplementary Figs. D.2-D.6). This, combined with the fact that GCN-3 takes more time than GCN-2 (Section 5.3.3), is why we did not consider GCN-3 for the biological NoN. Still, we expect that the integrative NoN-GCN approaches will outperform the concatenation-based ones. We find that while the NoN-GCN approaches do perform well for low across-edge amounts and low rewire-noise amounts, they are not as robust to changes in those parameters compared to the concatenation-based ones. Specifically, NoN-GCN approaches perform as well as concatenation-based ones for 7 out of the 30 NoNs and are worse for 23 out of the 30 NoNs (Fig. 5.2 and Supplementary Figs. D.2-D.6). These findings suggest that deep learning might not offer an advantage on this kind of synthetic data, or that more complex models are needed.

### 5.3.2 Accuracy on the biological network of networks

Again, we expect NoN approaches to improve upon single-level ones. Since we consider 131 GO terms and parsing raw results for every single one would be difficult, we instead present summarized results over the 131. Specifically, given the eight considered approaches (L1 GCM, L2 GDV, L1 GCM + L2 GDV, L1 DiffPool, L2



SIGN, L1 DiffPool + L2 SIGN, L1 GCM + L2 GDV + L1 DiffPool + L2 SIGN, and GCN-2), for each of AUPR, precision, recall, and F-score, for each GO term, we do the following. We rank each of the eight approaches that is significant (Section 5.2.5) from 1st best (rank 1) to 8th best (rank 8), considering any approaches within 1% of each other to be tied. Then, for each approach, we count how many times (i.e., for how many GO terms) it has rank 1, 2, etc. We find that NoN approaches have rank 1 for 49 out of the 131 GO terms with respect to AUPR, 37 out of 131 for precision, 35 out of 131 for recall, 33 out of 131 for F-score, and 69 out of 131 for at least one of the four evaluation measures (Fig. 5.3 and Supplementary Fig. D.7). We examine in more detail why NoN approaches work better than single-level approaches for some but not all GO terms, as follows.

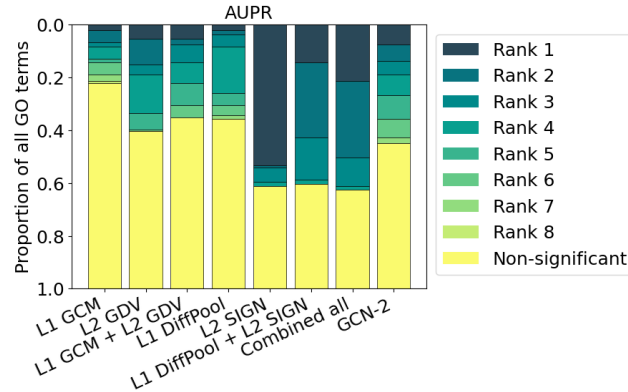


Figure 5.3. Summarized results of the eight considered approaches (as GCN-3 is not used for the biological NoN) in the task of protein functional prediction in terms of AUPR. For each GO term (out of the 131 total), we rank the eight approaches’ from best (rank 1) to worst (rank 8). Then, we calculate the proportion of GO terms each approach achieves each rank. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Results for other evaluation measures are shown in Supplementary Fig. D.7

First, we investigate whether the GO terms for which NoN approaches have rank 1 are different than the GO terms for which L2 SIGN, the best approach overall, has rank 1. If not, then NoN approaches would be redundant to L2 SIGN. To do so, for each NoN approach, we measure the overlap between the set of GO terms for which the given NoN approach has rank 1 and the set of GO terms for which L2 SIGN has rank 1. We find that NoN approaches have rank 1 for mostly different GO terms compared to L2 SIGN, with a maximum overlap of around 6% (Supplementary Fig. D.8). This suggests that NoN approaches are not redundant to L2 SIGN.

So, it makes sense to continue analyzing NoN approaches in comparison to single-level approaches. To better understand for which kinds of GO terms NoN approaches have rank 1 versus for which kinds of GO terms single-level approaches have rank 1, we do the following. For each evaluation measure, we split the 131 GO terms into six groups based on how single-level approaches perform in relation to NoN approaches, with “S” referring to single-level approaches and “C” referring to combined-level (i.e., NoN) approaches, as outlined in Table 5.2. As an example, for AUPR, “S < C” indicates that the performance of single-level approaches (“S”) is worse than (“<”) the performance of NoN approaches (“C”). In other words, the group “S < C” contains all GO terms for which at least one NoN approach has rank 1 (multiple NoN approaches can be tied with each other for rank 1), and all single-level approaches have rank 2 or worse, with respect to AUPR. Note that for a GO term in the above scenario, if no single-level approaches are significant, that GO term would be in the “C only” group instead, corresponding to those GO terms for which only NoN approaches are significant.

Given these groups, we investigate whether there are any GO terms where NoN approaches are necessary if one wants to make accurate predictions. We do so by looking at the number of GO terms for which at least one NoN approach has rank 1 and all single-level approaches are strictly worse, i.e., not tied for rank 1. This

TABLE 5.2

DESCRIPTION OF THE SIX GO TERM GROUPS BASED ON HOW  
SINGLE-LEVEL (S) AND COMBINED-LEVEL (C), I.E., NON,  
APPROACHES PERFORM

S only	At least one “S” approach is significant; no “C” approaches are significant.
S > C	At least one “S” approach is significant and has rank 1; at least one “C” approach is significant but none have rank 1.
S = C	At least one “S” approach is significant and has rank 1; at least one “C” approach is significant and has rank 1.
S < C	At least one “S” approach is significant but none have rank 1; at least one “C” approach is significant and has rank 1.
C only	No “S” approaches are significant; at least one “C” approach is significant.
No sig.	No approaches are significant.

“S < C” and “C only” are where NoN approaches are the best.

corresponds to the number of GO terms in the groups “S < C” and “C only”. We find that NoN approaches have rank 1 and are untied with any single-level approach for around 20%-30% of all GO terms, depending on evaluation measure (Table 5.3). Taking the union over all evaluation measures, we find that there are 33 (25% of) GO terms in “S < C” and 38 (29% of) in “C only”, i.e., a total of 60 (46% of) GO term across the two groups. That is to say, there are 33 GO terms where NoN approaches outperform single-level approaches (but single-level approaches are still significant) for at least one evaluation measure and, importantly, 38 GO terms where

TABLE 5.3

NUMBER OF GO TERMS IN EACH OF THE SIX GROUPS FOR  
AUPR, PRECISION, RECALL, AND F-SCORE

	Number of GO terms in each group for									
	AUPR		Precision		Recall		F-score		Union	
S only	12	9%	20	15%	33	25%	31	24%	46	
S > C	63	48%	45	35%	22	17%	30	23%	75	
S = C	8	6%	8	6%	6	5%	4	3%	18	
S < C	27	21%	8	6%	8	6%	9	7%	33	
C only	14	11%	21	16%	21	16%	20	15%	38	
No sig.	7	5%	29	22%	41	31%	37	28%	43	

For example, for AUPR, there are 14 GO terms in the group “C only”. We also report the union of GO terms in a given group over all measures (Union). For example, there are 38 GO terms in the union of “C only” over all evaluation measures. “S < C” and “C only” are where NoN approaches are the best. The IDs and names of GO terms in each group for each measure can be found in Supplementary Files D.1-D.4.

only NoN approaches are able to perform significantly better than random for at least one evaluation measure. In other words, for those 38 GO terms, only NoN approaches make meaningful protein functional predictions, while single-level ones achieve random accuracy. Taking the groups together, we find that there are 60 GO terms where NoN approaches have rank 1 and single-level approaches are strictly worse for at least one evaluation measure. These results suggest that NoN approaches are necessary, especially if one wants to make predictions for certain GO terms.

Since we now know that NoN approaches are important, we investigate which of them are the best. Here, we comment on results for AUPR (Supplementary Fig. D.9(a)), only noting that results are qualitatively similar for other measures (Supplementary Fig. D.9-D.13). For “S < C”, L1 GCM + L2 GDV + L1 DiffPool +

L2 SIGN, i.e., the combination of graph theoretic and graph learning approaches, is the best overall NoN approach. It has rank 1 for 19 GO terms, while all other NoN approaches have rank 1 for fewer than 19 GO terms (Supplementary Fig. D.9(a)). This suggests that integrating knowledge across domains is somewhat useful. For “C only”, GCN-2 has rank 1 for 9 GO terms, while all other NoN approaches have rank 1 for fewer than 9 GO terms (Supplementary Fig. D.9(b)). In fact, for 7 out of the 9 GO terms, GCN-2 is the only approach that is significant (Supplementary Fig. D.10). This suggests that deep learning could be useful for otherwise difficult-to-predict GO terms.

Finally, note that we did analyze the properties of GO terms in each of the six GO term groups, in order to see whether different GO term groups contain different kinds of GO terms. Specifically, for each group, we computed the distribution of the depths of the GO terms in the GO tree and the distribution of class sizes (number of proteins annotated by each GO term, which ranges from 20 to 277), and compared groups’ distributions to each other. We found that “S < C” and “C only” contain GO terms whose classes sizes are among the smallest, suggesting that NoN approaches may have some potential to make predictions for GO terms with limited training data. And while one might expect that GO terms with small class sizes correspond to those that are deep in the GO tree, we find that there is no significant difference between the six GO term groups with respect to GO term tree depth.

### 5.3.3 Running time analysis

Lastly, we analyze approaches’ running times for the synthetic NoN with 5% across-edge and 0% rewiring-noise amount as a representative; we choose a single NoN for two reasons. The first is that GCN-3 was only run on synthetic NoNs (Section 5.3.1), so they are the only NoNs where we can analyze the trade-off between performance (Fig. 5.2) and running time. The second is simplicity: trends are qualitatively

similar across all synthetic NoNs. For each approach, we record the time to extract all necessary features and the time for one epoch of training the associated classifier. For hardware details, see Supplementary Section D.2.3.

First, GCN-3, which we found does not have a clear advantage over GCN-2 in terms of accuracy (Section 5.3.1), takes 4.25x longer to train. This poor tradeoff between accuracy and running time is why we did not consider GCN-3 for the biological NoN.

Second, recall that L1 DiffPool + L2 SIGN and L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN, the best approaches overall, are as good as each other in terms of accuracy, with the former being worse in only 1 out of the 30 NoNs. Thus, because L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN has longer feature extraction and training time than L1 DiffPool + L2 SIGN (Supplementary Table D.1), L1 DiffPool + L2 SIGN would likely be the better approach to use for a general NoN when considering the trade-off between accuracy and running time. Also recall that L2 SIGN performs as well as L1 DiffPool + L2 SIGN and L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN for 21 out of the 30 NoNs, in those NoNs where there is significant clustering structure in the level 2 network. Thus, if one expects significant clustering structure in the level 2 network of a general NoN, L2 SIGN should be considered, as its feature extraction time is around 77x faster and its training time is around 1.5x faster than those of L1 DiffPool + L2 SIGN (Supplementary Table D.1).

## 5.4 Conclusion

We present a comprehensive framework to test whether integrating network information into an NoN leads to more accurate label predictions than using information from a single level alone. We also develop the first synthetic NoN generator that can create NoNs with a variety of parameters for study, construct a biological NoN from PPI network and PSN data, and propose a novel GCN-based model for label predic-

tion on NoNs. We have shown that on synthetic data, NoN approaches are among the best, and that on a real-world biological NoN, NoN approaches are necessary to make predictions about certain protein functions. As such, research into NoN-based data integration is promising, and likely could be applied to a variety of other tasks, especially as such NoN data becomes readily available.

To our knowledge, this study is the first to investigate data integration for label prediction using NoNs. As such, it is “just” a proof of concept. Many opportunities exist for further advancement of our work. As an example, recall that studies have combined protein sequence and protein structural data with PPI data [181, 132, 179]. So, an important future direction is the comparison between different data integration schemes for various tasks.

As another example, an NoN as we define it might have a limitation when trying to model certain systems. Namely, an interaction between two entities at the higher level may actually be due to a number of interactions occurring at the lower level. For example, an interaction between two proteins occurs due to interactions between subsets of their amino acids. Unfortunately, with current biotechnologies, large-scale data on interactions between proteins is captured at the protein level rather than at amino acid level. So, these fine-grained, amino acid-level interactions cannot be captured by our current NoN model. Advancements to account for them will be necessary once such detailed data become available. This is especially important since even our current, simpler NoN model already leads to improvements compared to current methods. Therefore, a more advanced version should only improve further (for applicable systems). However, our current NoN model does have advantages. Namely, not all systems that can be modeled as complex networks of networks benefit from the more detailed representation. For example, as discussed in Section 5.1, [129] study an NoN where an interaction between two level 2 nodes (individuals in a social network) is based on the individuals’ friendships. This could not be repre-

sented by interactions between subsets of level 1 nodes (neurons of the individuals' brain networks). As our current NoN model would be favorable for such systems, further development of the coarse-grained, and the fine-grained, NoN models are both important directions.

As another example, while our integrative NoN-GCN approach is not significantly better than just combining features from the two levels *overall*, there are some protein functions for which it is the only approach to make non-random predictions. This indicates that the strength of our NoN model is not just from the availability of more features for prediction (i.e., two levels instead of one), but rather also from the actual integration of the two levels that the model provides. Importantly, this also means that research into more sophisticated, scalable, and integrative deep learning models for NoNs, perhaps taking inspiration from SIGN's precomputable neighborhood aggregators, is worth pursuing.

As a final example, we only analyze a two-level NoN in this study, so expanding in scale is an important future direction.



## CHAPTER 6

### CONCLUDING REMARKS

In this dissertation, we investigate how to properly compare NA methods belonging to different categories; why NA methods do not align functionally related nodes, resulting in their poor protein functional prediction performance; and whether more complex multilevel data integration has advantages over simpler single-level representations. We design novel computational methods to improve alignment quality and analyze multilevel data, leading to improvements in protein functional prediction.

First, we introduce an evaluation framework for a fair comparison of PNA against MNA, and we find that PNA often outperforms MNA. In particular, in the ME framework, PNA can (by integrating pairwise alignments) produce multiple alignments that are superior to multiple alignments produced by MNA. Thus, we believe that any new MNA methods should be compared not just to existing MNA methods but also to existing PNA methods using our evaluation framework, to properly judge the quality of alignments that they produce. Moreover, the current process of integrating pairwise alignments (i.e., scaffolding procedure) is relatively simple. Any more sophisticated scaffolding procedure that might be developed in the future will yield even more superior PNA-based multiple alignments and consequently even further emphasize the superiority of PNA over MNA. In other words, for MNA to gain advantage over PNA, a drastic redesign of the current MNA algorithmic principles might be needed. In summary, the results of this work suggest that perhaps it might be sufficient to focus on the faster PNA and integration of pairwise alignments into multiple ones rather than on the slower MNA.

Second, we modify WAVE, MAGNA++, and SANA to align heterogeneous networks by extending homogeneous graphlets to their heterogeneous counterparts. We show that using more colors (i.e., more types of information) leads to better alignments, so using as much heterogeneous information as possible is the preferred option where available. The bottleneck to using more types of heterogeneous data is scalability of heterogeneous graphlets in terms of both time and space complexity. More efficient heterogeneous graphlet counting methods that make use of combinatorial relationships between heterogeneous graphlets, akin to existing efficient methods for homogeneous graphlet counting [79, 104, 137, 2], have been developed [139] and could help with the scalability issues. However, the code does not appear to be publicly available. So, further study is needed on this front. Or, more scalable methods for capturing the topology of a node in a heterogeneous network could be developed as an alternative to graphlets, such as those based on random walks [66, 41].

Third, we present TARA as a method that challenges the assumption of current NA methods that topologically similar nodes are functionally related. TARA can detect, from training data, a relationship between topologies of functionally similar nodes. TARA generally outperforms or complements existing approaches, even those that use sequence similarity-based anchor links across network as input (unlike TARA), in the task of protein function prediction, one of the ultimate goals of NA. Thus, we extend TARA into TARA++ to use both topological and sequence information, and find that it outperforms existing methods. As such, future NA methods should perhaps be developed in the data-driven paradigm rather than in the traditional topological similarity paradigm.

To our knowledge, TARA and TARA++ are the first data-driven approaches for biological NA. As such, they are just proof-of-concepts. There are many directions in which this work can be taken. TARA uses a relative simply combination of two nodes' GDVs to obtain the GDV of a node pair. More sophisticated combinations of

GDVs could be explored. TARA++ explores social network embedding to extract the feature of a node pair using node2vec, but the integrated network could be treated as a heterogeneous network, so heterogeneous feature extraction methods like those discussed above could be explored. Also, in both TARA and TARA++, we train a simple classifier, logistic regression – potential improvement could be seen with more sophisticated models.

There is still another category of NA methods that is not discussed in the dissertation. Namely, most of existing NA methods deal with static networks. That is, they do not account for any temporal (dynamic) information that exists. For example, cellular functioning differs across different ages or stages of cancer, meaning that dynamic or temporal PPI networks could be used to model the aging process or cancer progression. Without considering this temporal data, a lot of information is lost. Only recently, several approaches have been proposed for dynamic NA [165, 162, 7, 45]. These ideas for dynamic NA could be combined with those for each of pairwise, multiple, heterogeneous, and data-driven NA discussed in this dissertation, enabling the analysis of even more complex network data that better model cellular functioning. Importantly though, it is crucial that we (the NA community) gain in-depth understanding of practical implications of local versus global, one-to-one versus many-to-many, pairwise versus multiple, homogeneous versus heterogeneous, non-data-driven versus data-driven, static versus dynamic, and other types of NA, in order to learn from each NA type when developing new methods. Similarly, the subfields of social NA and biological NA are often disjoint – methods from one subfield are often not considered by researchers in the other, despite the problems posed sharing many similar aspects. A better unification of the two could help reduce “redundancies” and accelerate advancements.

Fourth, we present a comprehensive framework to test whether integrating network information into an NoN leads to more accurate label predictions than using

information from a single level alone. We also develop the first synthetic NoN generator that can create NoNs with a variety of parameters for study, construct a biological NoN from PPI network and PSN data, and propose a novel GCN-based model for label prediction on NoNs. We have shown that on synthetic data, NoN approaches are among the best, and that on a real-world biological NoN, NoN approaches are necessary to make predictions about certain protein functions. As such, research into NoN-based data integration is promising, and likely could be applied to a variety of other tasks, especially as such NoN data becomes readily available.

A struggle one might face not just when working with NoNs (especially beyond just two levels), but also with regard to the general trend of increasing network data, is that of scalability. Graph compressibility [100] is an interesting direction to address this issue. Currently, there exist generative graph models, such as vertex/edge replacement grammars [1, 150, 78] that could help. Given an input network, such models can learn to generate networks with similar properties as the input one – in other words, these models contain information about the properties of the input network inside them. In applications where some compression loss would be acceptable, such models could potentially be used to generate portions of relevant networks on demand to save on computational resources.

Network analysis has been rapidly trending toward deep learning, partially owing to the successes of image and natural language processing in that context. However, I do not think this means that traditional graph theoretic algorithms will be left behind. Besides many existing tasks where such algorithms are optimal, I see the potential for a fusion between graph theoretic algorithms and graph deep learning. With the renewed interest in differentiable programming [13], deep learning frameworks have integrated fundamental data structures such as stacks and “taken derivatives” of stack operations [43, 101], and incorporated natural “laws” via ordinary differential equations [26], to great success. So perhaps, traditional graph algorithms can be fitted

into deep learning frameworks as well, providing a stronger backbone for data-driven graph analyses to take off.

Something I greatly appreciate from my PhD journey is the emphasis on “elevator pitch”-type communication of research. After all, most people one interacts with will have expertise in different disciplines, so distilling one’s work into understandable concepts is important for any kind of exchange. This is in contrast to the impression I had in my undergraduate environment, where we always felt “cool” to show off all our technical jargon, and in my opinion, this feeling partially contributes to the disconnect between traditionally technical fields and the general population. This way of thinking, where if someone else is confused by your explanation, it is because *you* haven’t explained it well enough, puts focus on yourself to improve and helps establish common ground for effective communication – something we all could use more of these days.

## APPENDIX A

### PAIRWISE VERSUS MULTIPLE NETWORK ALIGNMENT

#### A.1 Methods

##### A.1.1 NA methods that we evaluate

The PNA methods that we evaluate are GHOST, MAGNA++, WAVE, and L-GRAAL. The MNA methods that we evaluate are IsoRankN, BEAMS, multi-MAGNA++, and ConvexAlign.

**PNA methods.** Most NA methods are *two-stage* aligners: in the first stage, they calculate the similarities (based on network topology and, optionally, protein sequence information) between nodes in the compared networks, and in the second stage, they use an alignment strategy to find high scoring alignments with respect to the total similarity over all aligned nodes. GHOST is an example of two-stage PNA methods. GHOST calculates the similarity of “spectral signatures” of nodes between the compared networks in its first stage. Then, GHOST uses an alignment strategy consisting of a seed-and-extend global alignment step followed by a local search procedure that aims to improve, with respect to node similarity, upon the seed-and-extend step. An issue with two-stage methods is that while they find high scoring alignments with respect to total node similarity (a.k.a. node conservation), they do not take into account the amount of conserved edges during the alignment construction process. But the quality of a network alignment is often measured in terms of the amount of conserved edges. To address this issue, MAGNA++ directly optimizes both edge and node conservation *while* the alignment is constructed; its node conservation mea-

sure typically uses graphlet-based node similarities [112]. MAGNA is a *search-based* (rather than a two-stage) PNA method. Search-based aligners can directly optimize edge conservation or any other alignment quality measure. WAVE was proposed as a two-stage (rather than search-based) PNA method that optimizes both a graphlet-based node conservation measure as well as (weighted) edge conservation by using a seed-and-extend alignment strategy based on the principle of voting. Similarly, L-GRAAL optimizes a graphlet-based node conservation measure and a (weighted) edge conservation measure, but it uses a seed-and-extend strategy based on integer programming.

**MNA methods.** IsoRankN is a two-stage MNA method. It calculates node similarities between all pairs of compared networks using a PageRank-based spectral method. IsoRankN then creates a graph of the node similarities and partitions the graph using spectral clustering in order to produce a many-to-many alignment. BEAMS is a two-stage method that optimizes both a (protein sequence-based) node conservation measure and an edge conservation measure. BEAMS uses a maximally weighted clique finding algorithm on a graph of node similarities to produce a one-to-one alignment, where node similarity is based only on protein sequence information, without considering any topological node similarity information. BEAMS then creates a many-to-many alignment from the one-to-one alignment using an iterative greedy algorithm that maximizes both node and edge conservation. ConvexAlign is also a two-stage method. It optimizes an objective function that combines topological node similarity, optional sequence-based node similarity, and edge conservation. That is, it optimizes both node and edge conservation. ConvexAlign optimizes its objective function with an optimization strategy that is formulated as an integer program, which is relaxed into a convex optimization problem. This problem is then solved using the alternating direction method of multipliers (ADMM). This allows ConvexAlign to align multiple networks simultaneously. Like MAGNA++, multiMAGNA++ is a

search-based method that directly optimizes both edge and node conservation while the alignment is constructed. Of the MNA methods, IsoRankN and BEAMS produce many-to-many alignments, while ConvexAlign and multiMAGNA++ produce one-to-one alignments.

**Aligning using network topology only versus using both topology and protein sequences.** In our analysis, for each method, we study the effect on output quality when (i) using only network topology while constructing alignments (T alignments) versus (ii) using both network topology and protein sequence information while constructing alignments (T+S alignments). For T alignments, we set method parameters to ignore any sequence information. All methods except BEAMS can produce T alignments and all methods can produce T+S alignments. For T+S alignments, we set method parameters to include sequence information. Supplementary Table A.2 shows the specific parameters that we use. Specifically, the methods combine topological information with sequence information in order to optimize  $\theta S_T + (1 - \theta)S_P$ , where  $S_T$  is the (node or edge) cost function based on *topological information*,  $S_P$  is the node cost function based on *protein sequence information*, and  $\theta$  weighs between topological information and sequence information. When  $\theta = 1$ , only network topology is used in the alignment process, and when  $\theta = 0$ , only sequence information is used. We set  $\theta = 0.5$  in our study due to the following reasons (except for ConvexAlign, see below). First, Meng et al. [110], who used the same datasets that we use in our study, showed that as long as some amount of topological information and some amount of protein sequence information are used in the alignment process (i.e., as long as  $\theta$  does not equal 0 or 1), the quality of the resulting alignments is not drastically affected. They showed this for ten PNA methods, including GHOST, MAGNA++, WAVE, and L-GRAAL, which are the PNA methods that we use in this study. Second, it was shown by the original studies which introduced two of the MNA methods used in this study that varying  $\theta$  between 0.3 and 0.7 has no large effect



on the quality of alignments produced by BEAMS and IsoRank [4], and that varying  $\theta$  between 0.2 and 0.8 has no large effect on the quality of alignments produced by FUSE [62]. Third, the original MAGNA++ paper, which multiMAGNA++ is based on, showed that varying  $\theta$  between 0.1 and 0.9 has no large effect on the quality of alignments produced by MAGNA++. So, in the original multiMAGNA++ paper, the  $\theta$  parameter was set to 0.5. We believe that all of this justifies our choice of using  $\theta$  of 0.5 for all methods considered in our study (except for ConvexAlign, see below). Also, using the same  $\theta$  value for all methods (except for ConvexAlign, see below) ensures that any potential differences in results of the different methods are not caused by using different amounts of network topology versus protein sequence information. While in an ideal scenario we would have wanted to use  $\theta = 0.5$  for ConvexAlign’s T+S alignments as well (just like we do for all other considered methods), the authors of ConvexAlign pre-set this value in ConvexAlign’s implementation to a recommended value of 0.343 (see below), thus weighing topological information by 0.343 and sequence information by 0.657. We respect this recommendation and consequently use  $\theta = 0.343$  for ConvexAlign.

Next, we clarify how the given method’s parameter values from Supplementary Table A.2 match the desired  $\theta$  value.

Recall that the methods combine topological information with sequence information in order to optimize  $\theta S_T + (1 - \theta)S_P$ , where  $S_T$  is the (node or edge) cost function based on *topological information*,  $S_P$  is the node cost function based on *protein sequence information*, and  $\theta$  weighs between topological information and sequence information.

For T alignments, we set parameters such that only topological information is used (i.e., such that  $\theta = 1.0$ ). Namely, setting  $\theta = 1.0$  is equivalent to setting the following parameter value(s) for each of the methods, where  $E_T$ ,  $N_T$ , and  $N_S$  are the topological edge conservation function, topological node cost function, and sequence-

based node cost function, respectively. (That is,  $E_T$  and/or  $N_T$  form  $S_T$  from the above  $\theta$ -related formula, and  $N_S$  is  $S_P$  from the above  $\theta$ -related formula.)

- For GHOST, which optimizes  $\alpha N_T + (1 - \alpha)N_S$ , setting  $\theta = 1.0$  corresponds to setting  $\alpha = 1.0$ , i.e., alpha=1.0 in the GHOST implementation.
- For L-GRAAL, which optimizes  $(1 - \alpha)E_T + \alpha N_S$  (where  $E_T$  is edge conservation weighted by topological node similarity), setting  $\theta = 1.0$  corresponds to setting  $\alpha = 0.0$ , i.e., a=0.0 in the L-GRAAL implementation.
- For MAGNA++, which optimizes  $\alpha E_T + (1 - \alpha)(\beta N_T + (1 - \beta)N_S)$ , setting  $\theta = 1.0$  corresponds to setting  $\alpha = 0.5$  and  $\beta = 1.0$ , i.e., setting a=0.5 and inputting only topological node similarity into the MAGNA++ implementation, respectively. Note that we use a=0.5 to give equal weight to edge conservation and node conservation.
- For WAVE, which optimizes  $\alpha N_T + (1 - \alpha)N_S$ , setting  $\theta = 1.0$  corresponds to setting  $\alpha = 1.0$ , i.e., inputting only topological node similarity to the WAVE implementation. Note that WAVE also optimizes edge conservation, but it does so implicitly, as a part of its alignment strategy. That is, edge conservation is not an input parameter of WAVE or its implementation.
- For IsoRankN, which optimizes  $\alpha N_T + (1 - \alpha)N_S$ , setting  $\theta = 1.0$  corresponds to setting  $\alpha = 1.0$ , i.e., alpha=1.0 in the IsoRankN implementation.
- For ConvexAlign, which optimizes  $\lambda_2 E_T + (1 - \lambda_2)(\lambda_1 N_T + (1 - \lambda_1)N_S)$ , setting  $\theta = 1.0$  corresponds to setting  $\lambda_1 = 1.0$ , i.e., inputting no node similarity into the ConvexAlign implementation. Note that we use  $\lambda_2$  of 0.02, as recommended and pre-set by the authors of the ConvexAlign paper. ConvexAlign authors have recommended all of its parameter values after testing them using cross-validation. So, we did not need to set any parameter values ourselves.
- For multiMAGNA++, which optimizes  $\alpha E_T + (1 - \alpha)(\beta N_T + (1 - \beta)N_S)$ , setting  $\theta = 1.0$  corresponds to setting  $\alpha = 0.5$  and  $\beta = 1.0$ , i.e., setting a=0.5 and inputting only topological node similarity into the multiMAGNA++ implementation, respectively. Note that we use a=0.5 to give equal weight to edge conservation and node conservation.

For T+S alignments, we set parameters such that both topological and sequence information is used (i.e., such that  $\theta = 0.5$ , unless recommended otherwise by the authors of the given method). Namely, setting  $\theta = 0.5$  is equivalent to setting the following parameter value(s) for each of the methods.

- For GHOST, which optimizes  $\alpha N_T + (1 - \alpha)N_S$ , setting  $\theta = 0.5$  corresponds to setting  $\alpha = 0.5$ , i.e.,  $\alpha=0.5$  in the GHOST implementation.
- For L-GRAAL, which optimizes  $(1 - \alpha)E_T + \alpha N_S$  (where  $E_T$  is edge conservation weighted by topological node similarity), setting  $\theta = 0.5$  corresponds to setting  $\alpha = 0.5$ , i.e.,  $\alpha=0.5$  in the L-GRAAL implementation.
- For MAGNA++, which optimizes  $\alpha E_T + (1 - \alpha)(\beta N_T + (1 - \beta)N_S)$ , setting  $\theta = 0.5$  corresponds to setting  $\alpha = 0.25$  and  $\beta = 0.33$ , i.e., setting  $\alpha=0.25$  and inputting the combined node similarity information into the MAGNA++ implementation. With these parameter values, topological and sequence-based cost functions are equally weighted. Namely, the optimization formula for MAGNA++ becomes  $0.25E_T + 0.75(0.33N_T + 0.67N_S) = 0.25E_T + 0.25N_T + 0.5N_S = 0.5S_T + 0.5S_P$ , i.e.,  $\theta = 0.5$ , as desired.
- For WAVE, which optimizes  $\alpha N_T + (1 - \alpha)N_S$ , setting  $\theta = 0.5$  corresponds to setting  $\alpha = 0.5$ , i.e., inputting both topological and sequence-based node similarities to the WAVE implementation. Note that WAVE also optimizes edge conservation, but it does so implicitly, as a part of its alignment strategy. That is, edge conservation is not an input parameter of WAVE or its implementation.
- For IsoRankN, which optimizes  $\alpha N_T + (1 - \alpha)N_S$ , setting  $\theta = 0.5$  corresponds to setting  $\alpha = 0.5$ , i.e.,  $\alpha=0.5$  in the IsoRankN implementation.
- For ConvexAlign, which optimizes  $\lambda_2 E_T + (1 - \lambda_2)(\lambda_1 N_T + (1 - \lambda_1)N_S)$ , we use  $\lambda_1 = 0.33$  and  $\lambda_2 = 0.02$ , as recommended and pre-set by the authors of the ConvexAlign paper. ConvexAlign authors have recommended all of its parameter values after testing them using cross-validation. So, we did not need to set any parameter values ourselves. With these two parameter values, the optimization formula for ConvexAlign becomes  $0.02E_T + 0.98(0.33N_T + 0.67N_S) = 0.02E_T + 0.323N_T + 0.657N_S = 0.343S_T + 0.657S_P$ , i.e.,  $\theta = 0.343$ . Clearly, ConvexAlign weighs sequence information higher than the other methods (65.7% of the whole objective function for ConvexAlign, as opposed to 50% of the whole objective function for the other methods). Again, this is because the authors of ConvexAlign suggested doing 65.7% for their method, while our justification for 50% for the other methods is discussed above.
- For multiMAGNA++, which optimizes  $\alpha E_T + (1 - \alpha)(\beta N_T + (1 - \beta)N_S)$ , setting  $\theta = 0.5$  corresponds to setting  $\alpha = 0.25$  and  $\beta = 0.33$  as recommended by the multiMAGNA++ paper, i.e., setting  $\alpha=0.25$  and inputting the combined node similarity information into the multiMAGNA++ implementation. With these parameter values, topological and sequence-based cost functions are equally weighted. Namely, the optimization formula for multiMAGNA++ becomes  $0.25E_T + 0.75(0.33N_T + 0.67N_S) = 0.25E_T + 0.25N_T + 0.5N_S = 0.5S_T + 0.5S_P$ , i.e.,  $\theta = 0.5$ .

### A.1.2 Alignment quality measures

Here, we describe the alignment quality measures that we use to evaluate the NA methods. To do so, we first need to formally define an alignment. Typical PNA methods produce alignments comprising node pairs and typical MNA methods produce alignments comprising node clusters. We introduce the term *aligned node group* to describe either an aligned node pair or an aligned node cluster. With this, we can represent a pairwise or multiple alignment as a set of aligned node groups. Let  $G_1(V_1, E_1), \dots, G_k(V_k, E_k)$  be  $k$  networks with node and edge sets  $V_l$  and  $E_l$ , respectively, for  $l = 1, 2, \dots, k$ . An *alignment* of the  $k$  networks is a set of disjoint node groups, where each group is represented as a tuple  $(a_1, \dots, a_k)$  with the following properties: (i)  $a_l$  is the set of nodes in the node group from network  $G_l$ , i.e.,  $a_l \subseteq V_l$ , for  $l = 1, 2, \dots, k$ , (ii) no two node groups have any common nodes, i.e., given two different groups  $(a_1, a_2, \dots, a_k)$  and  $(b_1, b_2, \dots, b_k)$ ,  $a_l \cap b_l = \emptyset$  for  $l = 1, 2, \dots, k$ , and (iii) there must be at least two nodes in each node group, i.e.,  $|\cup_{l=1, \dots, k} a_l| \geq 2$ . If for each node group in the given alignment there is at most one node from each network, i.e., if for each node group  $|a_l| \leq 1$  for  $l = 1, \dots, k$ , then the alignment is a *one-to-one* alignment. Otherwise, it is a *many-to-many* alignment.

#### A.1.2.1 Topological quality (TQ) measures

A good NA method should produce aligned node groups that have internal consistency with respect to protein labels. If we know the true node mapping between the networks, then we can let the labels be protein names. When the labels are based on the true node mapping, i.e., on protein names, we consider measures that rely on node labels to be capturing topological alignment quality (TQ). If we do not know the true node mapping, we let the labels be GO terms. In this case, since GO terms capture protein functions, we consider measures that rely on GO terms to be captur-

ing functional alignment quality (FQ). We discuss such measures in Supplementary Section A.1.2.2.

Also, a good NA method should find a large amount of common network structure across the compared networks, i.e., produce high edge conservation.

Finally, for a good NA method, conserved edges should form large, dense, connected regions (as opposed to small or isolated conserved regions).

Below, first, we discuss how we measure internal consistency of aligned protein groups in a pairwise alignment. Second, we comment on how we do this in a multiple alignment. Third, we discuss how we measure edge conservation in a pairwise alignment. Fourth, we comment on how we do this in a multiple alignment. Fifth, we discuss how we capture the notion of large, dense, and connected conserved network regions (for both pairwise and multiple alignments).

**Measuring internal node group consistency of a pairwise alignment via precision, recall, and F-score of node correctness (P-NC, R-NC, and F-NC, respectively).** These measures [110] are a generalization of node correctness (NC) from one-to-one to many-to-many pairwise alignments. NC for one-to-one pairwise alignments is the fraction of node pairs from the alignment that are present in the true node mapping. As such, NC evaluates the *precision* of the alignment. NC is extended to many-to-many pairwise alignments as follows. For each aligned node group  $C_i$  in the alignment,  $C_i$  is converted into a set of all possible  $\binom{|C_i|}{2}$  node pairs in the group. The union of all resulting node pairs over all groups  $C_i$  forms the set  $X$  of all aligned node pairs. Then, given the set  $Y$  of all node pairs from the true node mapping, P-NC =  $\frac{|X \cap Y|}{|X|}$ , R-NC =  $\frac{|X \cap Y|}{|Y|}$ , and F-NC is the harmonic mean of P-NC and R-NC. These three measures work for both one-to-one and many-to-many pairwise alignments.

**Measuring internal node group consistency of a multiple alignment via adjusted multiple node correctness (NCV-MNC).** Multiple node correctness

(MNC) [163] is a generalization of the NC measure to multiple alignments. MNC uses the notion of normalized entropy (NE), which measures, for a given aligned node group, how likely it is to observe the same or higher level of internal node group consistency by chance, i.e., if the group of the same size was formed by randomly assigning to it proteins from the compared networks. The lower the NE, the more consistent the node group. Then, MNC is one minus the mean of NEs across all node groups. We refer to Vijayan and Milenković [163] for the formal definition of MNC. Since a good NA method should align (or cover) many of the nodes in the compared networks, as was done by Vijayan and Milenković [163], we adjust the MNC measure to account for node coverage (NCV), which is the fraction of nodes that are in the alignment out of all nodes in the compared networks. Then,  $\text{MNC-NCV} = \sqrt{(\text{NCV})(\text{MNC})}$ . When either NCV or MNC is low, the geometric mean of the two is penalized. The NCV-MNC measure works for both one-to-one and many-to-many multiple alignments.

**Measuring edge conservation of a pairwise alignment via adjusted generalized  $S^3$  (NCV-GS<sup>3</sup>).** Given two compared networks, generalized  $S^3$  (GS<sup>3</sup>) [110] measures the fraction of conserved edges out of both conserved and non-conserved edges, where an edge is conserved if it maps to an edge in the other network and an edge is not conserved if it maps to a non-adjacent node pair (i.e., a non-edge) in the other network. We refer to Meng et al. [110] for formal definition of GS<sup>3</sup>. As was done by Meng et al. [110], we penalize alignments with low node coverage by combining NCV with GS<sup>3</sup> into the adjusted GS<sup>3</sup> measure, NCV-GS<sup>3</sup>, which equals  $\sqrt{(\text{NCV})(\text{GS}^3)}$ . The NCV-GS<sup>3</sup> measure works for both one-to-one and many-to-many pairwise alignments.

**Measuring edge conservation of a multiple alignment via adjusted cluster interaction quality (NCV-CIQ).** CIQ [4] is a weighted sum of edge conservation between all pairs of aligned node groups. We refer to Vijayan and Milenković [163]

for the formal definition of CIQ. As was done by Vijayan and Milenković [163], we penalize alignments with low node coverage by combining NCV with CIQ into the adjusted CIQ, NCV-CIQ, which equals  $\sqrt{(\text{NCV})(\text{CIQ})}$ . The NCV-CIQ measure works for both one-to-one and many-to-many multiple alignments.

**Measuring the size of the largest connected region using largest common connected subgraph (LCCS).** The LCCS measure, which was recently extended from PNA [143] to MNA [163], simultaneously captures the size (i.e., the number of nodes) and the density (i.e., the number of edges) of the largest common connected subgraph formed by the conserved edges, penalizing smaller or sparser subgraphs. We refer to Vijayan and Milenković [163] for the formal definition of LCCS. The LCCS measure works for both one-to-one and many-to-many alignments, and for both pairwise and multiple alignments.

#### A.1.2.2 Functional quality (FQ) measures

Per Supplementary Section A.1.2.1, a good alignment should have internally consistent aligned node groups. Instead of protein names as in Supplementary Section A.1.2.1, in this section we use GO terms as protein labels to measure internal consistency.

Having aligned node groups that are internally consistent with respect to protein labels is important for protein function prediction. In addition to measuring internal node group consistency, we directly measure the accuracy of protein function prediction. That is, we first use a protein function prediction approach (Chapter 2.2.3.3 of the main document) to predict protein-GO term associations, and then we compare the predicted associations to known protein-GO term associations to see how accurate the predicted associations are.

Below, first, we discuss how we measure internal node group consistency with respect to GO terms. Second, we discuss an alternative popular measure for doing the

same. Third, we discuss how we measure the accuracy of protein function prediction, i.e., of predicted protein-GO term associations (note that we describe a strategy that we use to make the predictions in Chapter 2.2.3.3 of the main document).

**Measuring internal node group consistency using mean normalized entropy (MNE).** MNE [99] first uses normalized entropy (NE) to measure GO term-based consistency of an individual aligned node group. The lower the NE, the more consistent the given node group. Then, MNE is the mean of the NEs across all node groups. We refer to Vijayan and Milenković [163] for the formal definition of MNE. The MNE measure works for both one-to-one and many-to-many alignments, and for both pairwise and multiple alignments.

**Measuring internal node group consistency using GO correctness (GC).** GO correctness, which was recently extended from PNA [96] to MNA [163], measures the internal consistency of aligned node groups with respect to GO terms as follows. For each node group  $C_i$  in the alignment,  $C_i$  is converted into a set of all possible  $\binom{|C_i|}{2}$  node pairs in the group. The union of all resulting node pairs over all groups  $C_i$  forms the set  $X$  of all aligned node pairs. A subset of  $X$  that consists of all node pairs in which each of the two nodes is annotated with at least one GO term is denoted as  $Y$ . Then, GO correctness is the fraction of node pairs in  $Y$  in which the two nodes are both annotated with the same GO term. In other words, GO correctness is the fraction of all pairs of aligned nodes in which the aligned nodes share a GO term. The GO correctness measure works for both one-to-one and many-to-many alignments, and for both pairwise and multiple alignments.

**Precision, recall, and F-score of protein function prediction (P-PF, R-PF, and F-PF, respectively).** We describe how we predict protein-GO term associations in Chapter 2.2.3.3 of the main document. Here, we describe how we evaluate accuracy of such predictions. Given predicted protein-GO term associations, we calculate accuracy of the predictions via precision, recall, and F-score measures.



Formally, given the set  $X$  of predicted protein-GO term associations, and the set  $Y$  of known protein-GO term associations,  $P\text{-PF} = \frac{|X \cap Y|}{|X|}$ ,  $R\text{-PF} = \frac{|X \cap Y|}{|Y|}$ , and  $F\text{-PF}$  is the harmonic mean of precision and recall. These three measures work for both one-to-one and many-to-many alignments, and for both pairwise and multiple alignments.

#### A.1.2.3 Protein function prediction approaches

**Approach 3. New protein function prediction for multiple alignments.** We follow our discussion from Chapter 2.2.3.3 of the main document regarding approach 3, our new protein function prediction approach for multiple alignments. Formally, given an alignment of  $k$  networks,  $G_1(V_1, E_1)$ ,  $G_2(V_2, E_2)$ ,  $\dots$ ,  $G_k(V_k, E_k)$ , and given node  $v$  in the alignment that has at least one annotated GO term, and given GO term  $g$ , we hide the protein's true GO term(s) and find the significance of the alignment with respect to GO term  $g$  using the hypergeometric test, as follows. For each node group  $C_i$  in the alignment, we convert  $C_i$  into a set of node pairs  $F_i$  by taking all node pairs in the node group, after which we concatenate the sets of node pairs into a single set  $F$ . Then, let  $V_i^* \subset V_i$  be such that each node in  $V_i^*$  is annotated with at least one GO term. Let  $S_1$  be the set of all possible pairs of proteins in  $F$  such that one protein is in  $V_i^*$  and the other is in  $V_j^*$ , where  $i \neq j$ . Let  $A_i \subset V_i^*$  be such that each node in  $A_i$  is annotated with  $g$ . Let  $S_2$  be the set of all possible pairs of proteins between  $A_i$  and  $A_j$ , where  $i \neq j$ . Let  $K$  be the set of pairs of proteins that are in  $F$  and in  $S_1$ . Let  $X$  be the set of pairs of proteins that are in  $F$  and in  $S_2$ . Then, we use the hypergeometric test to calculate the probability of observing by chance  $|X|$  or more pairs of proteins in  $F$  with each node annotated with  $g$  is  $p = 1 - \sum_{i=0}^{|X|-1} \frac{\binom{|K|}{i} \binom{|S_1|-|K|}{|S_2|-i}}{\binom{|S_1|}{|S_2|}}$ . We consider the alignment to be significant with respect to  $g$  if the  $p$ -value is less than 0.05. We predict  $v$  to be associated with  $g$  if the alignment is significant with respect to  $g$ , resulting in predicted protein-GO term associations. If the alignment

is significant with respect to  $g$ , we predict  $v$  to be associated with  $g$ . Repeating this process for all nodes and GO terms results in predicted protein-GO term associations  $X$ .

#### A.1.2.4 Statistical significance of alignment quality scores

We continue our discussion from Chapter 2.2.3.4 of the main document on how to compute the  $p$ -value of a quality score of an actual alignment. This is done as follows. We construct a set of 1,000 corresponding random alignments (1,000 is what was practically reasonable given our computational resources), under a null model that conserves the following properties of the actual alignment: the number of node groups, the number of nodes in each group, and the network from which each node in each node group originates from. Then, the  $p$ -value of the alignment quality score is the fraction of the 1,000 random alignments with equal or better score than the actual alignment. We consider an alignment quality score to be significant if its  $p$ -value is less than 0.001. Note that if a given method fails to produce an alignment of a network pair/set, we set the  $p$ -values of all quality scores associated with the method and network pair/set to 1 and hence consider all of the associated quality scores to be non-significant.

### A.1.3 Evaluation framework

#### A.1.3.1 Multiple evaluation (ME) framework

We continue our discussion from Chapter 2.2.4.2 of the main document on how we combine the pairwise alignments over every network pair in the given set into a multiple alignment, i.e., how we produce alignments from the ME-P-P and ME-M-P categories. This procedure was inspired by Dohrmann et al. [39]. Given pairwise alignments of  $k$  networks  $G_1(V_1, E_1), \dots, G_k(V_k, E_k)$ , Dohrmann et al. [39] produce a multiple alignment of the  $k$  networks as follows. First, they select a “scaffold”

network  $G_r$  among the  $k$  networks, namely the network whose sum of “topological similarities” to the remaining  $k - 1$  networks is maximized; one of the suggested “topological similarity” measures is Graphlet Degree Distribution (GDD) agreement [136]. Second, they align  $G_r$  to each of the remaining  $k - 1$  networks. Third, they take the union of all aligned node groups from the resulting  $k - 1$  alignments. Let us denote this union as set  $A$ . Since the node groups in set  $A$  are not necessarily disjoint, Dohrmann et al. [39] use set  $A$  to create a new set  $A'$  of aligned node groups that are disjoint. This is done as follows. Let  $A'$  be an empty set. First, randomly pick an aligned node group  $C$  that is currently in  $A$  (initially, all node groups are in  $A$ ) and remove it from  $A$ . Then, remove from  $A$  all node groups that have at least one node in common with  $C$ , and merge the node groups into  $C$ . Repeat this process until there are no more node groups in  $A$  that have at least one node in common with  $C$ . Then, add  $C$  to  $A'$ . Repeat this process until  $A$  is empty. This results in a new multiple alignment  $A'$ . We illustrate this procedure in Fig. 2.3(b,c) of the main document. In our work, instead of choosing one of the  $k$  analyzed networks as a scaffold network using BLAST protein similarity information as Dohrmann et al. [39] does, because the choice of scaffold network affects the quality of the resulting multiple alignment (which we actually validate in Supplementary Fig. A.9), we vary each of the  $k$  networks as the scaffold network  $G_r$ , and we choose the scaffold based on the quality of the resulting multiple alignments. That is, we rank (as explained below) each of the  $k$  multiple alignments, in order to select the best (in terms of the rank) of them. We rank the alignments as follows. For each alignment quality measure, we rank the alignments from the best one to the worst one. (In case of ties, we let the ranks of the tied alignments be the tied alignments’ average rank.) Then, we compute the total rank of each alignment by taking the average of the given alignment’s ranks over all of the alignment quality measures. Finally, we select the best (in terms of the total rank) of all alignments. Note that here, we consider all

measures that can deal with multiple alignments, except NCV-MNC, which we leave out because not all network pairs/sets have the true node mapping (and NCV-MNC requires knowing this mapping), and except MNE, which we leave out so that the number of TQ and FQ measures matches (which is required in order to prevent the ranks to be dominated by topological or functional alignment quality). That is, we consider NCV-CIQ and LCCS TQ measures and GC and F-PF FQ measures.

#### A.1.4 T versus T+S alignments

Here, we continue our discussion from Chapter 2.3.1 of the main document regarding the similarity (overlap) of the alignments produced the different NA methods, each with its T and T+S versions (Supplementary Figs. A.1–A.3). Surprisingly, over all considered network datasets, in each of the PE and ME frameworks, the T+S versions of the different methods are overall more similar than the T+S and T versions of the same method are (with the exception of IsoRankN in the PE framework and also GHOST in the ME framework). That is, the T+S versions of the different methods cluster together in Supplementary Fig. A.1 and are clearly separated from the T versions. In contrast, the T versions do not cluster together. This shows that using protein sequence information overall yields alignment consistency independent of which NA method is used. Similar holds when we break down this analysis for networks with known versus unknown node mapping (Supplementary Figs. A.2–A.3), with the exception of networks with unknown node mapping under the ME framework, where the T and T+S versions of the same approach are often clustered together.

#### A.1.5 Method comparison in the ME framework: accuracy versus running time

The running time discussion in Chapter 2.3.4 of the main document deals with empirical running times of the considered PNA and MNA methods, when the methods

are run on our considered network sets, the largest one of which contains six networks. Since the PNA methods must align every pair of networks in a network set in order to produce a multiple alignment, and since this results in a quadratically increasing running time with respect to the number of networks  $k$ , we next ask whether there is some (larger than six) value of  $k$  at which PNA might become less efficient (i.e., slower) than MNA. To answer this, because of the limited sizes (in terms of  $k$ ) of our considered network sets, we need to analyze the methods' theoretic running time complexities with respect to  $k$  (Supplementary Table A.3). All of the PNA methods' running times grow quadratically with  $k$  due to the required pairwise alignments, per the above discussion. Of the MNA methods, IsoRankN's running time also grows quadratically, ConvexAlign's running time grows cubically, BEAMS' running time grows exponentially, and multiMAGNA++'s time grows linearly with  $k$ . So, when comparing the PNA and MNA methods, only multiMAGNA++ grows slower (i.e., is expected to be faster) with  $k$  than the PNA methods, IsoRankN grows at the same rate as the PNA methods, and ConvexAlign and BEAMS grow faster than the PNA methods. Hence, we do not expect that the MNA methods will have advantage over the PNA methods as  $k$  increases, with the exception of multiMAGNA++. However, note that the analysis of the methods' theoretic running times is different than the analysis of their empirical running times, and also, note that their theoretic as well as empirical running times depend not just on  $k$  but also on the sizes of the considered networks in terms of the numbers of their nodes and edges, and also potentially on some method-specific parameters. For example, while multiMAGNA++ theoretically grows the slowest with  $k$  of all considered PNA and MNA methods, as we can see from its empirical running time analysis (Fig. 2.5, View III, in the main document), multiMAGNA++ is significantly slower than BEAMS on our considered network sets with up to six networks. So, it is hard to estimate the exact value of  $k$  at which

multiMAGNA++ would empirically perform faster than the other methods, as this would also depend on e.g., the size of each network in the considered network set.

## A.2 Results

TABLE A.1

DETAILS ON THE PINS THAT WE USE IN OUR STUDY

Set	Species	No. of proteins	No. of interactions
Yeast+%LC	Yeast+0%LC	1,004	8,323
	Yeast+5%LC	1,004	8,739
	Yeast+10%LC	1,004	9,155
	Yeast+15%LC	1,004	9,571
	Yeast+20%LC	1,004	9,987
	Yeast+25%LC	1,004	10,403
PHY <sub>1</sub>	Fly	7,887	36,285
	Worm	3,006	5,506
	Yeast	6,168	82,368
	Human	16,061	157,650
PHY <sub>2</sub>	Yeast	768	13,654
	Human	8,283	19,697
Y2H <sub>1</sub>	Fly	7,097	23,370
	Worm	2,874	5,199
	Yeast	3,427	11,348
	Human	9,996	39,984
Y2H <sub>2</sub>	Yeast	744	966
	Human	1,191	1,567

The true node mapping is known for the Yeast+%LC network set, unlike for the other network sets. Since the largest connected components of the fly and worm networks in PHY<sub>2</sub> and Y2H<sub>2</sub> are too small, we do not use those networks in our analysis.

TABLE A.2

## METHOD PARAMETERS FOR PNA THAT WE USE IN OUR STUDY

Algorithms	Parameters
PNA methods, T alignments	
GHOST	beta=1e10 alpha=1.0
L-GRAAL	a=0.0 node similarity = graphlet degree vector (GDV) similarity
MAGNA++	m=S3 p=15000 n=10000 a=0.5 node similarity = GDV similarity
WAVE	node similarity = GDV similarity
PNA methods, T+S alignments	
GHOST	beta=1e10 alpha=0.5
L-GRAAL	a=0.5 node similarity = GDV and BLAST protein similarity
MAGNA++	m=S3 p=15000 n=10000 a=0.25 node similarity = GDV and BLAST protein similarity
WAVE	node similarity = GDV and BLAST protein similarity
MNA methods, T alignments	
IsoRankN	K=30 thresh=1e-4 maxveclen=5000000 alpha=1.0
ConvexAlign	lambda_edge=3 numOuterIterations=4 flag_fast=1 mu=150 min_val=0.5 lambda_mul=0.5 node similarity = none
multiMAGNA++	m=CIQ p=15000 n=100000 e=0.5 a=0.5 node similarity = GDV similarity



TABLE A.2 (CONTINUED)

Algorithms	Parameters
MNA methods, T+S alignments	
IsoRankN	K=30 thresh=1e-4 maxvecLen=5000000 alpha=0.5 node similarity = BLAST protein similarity
ConvexAlign	lambda_edge=3 numOuterIterations=4 flag_fast=1 mu=150 min_val=0.5 lambda_mul=0.5 node similarity = BLAST protein similarity
BEAMS	beta=0.4 alpha=0.5 node similarity = BLAST protein similarity
multiMAGNA++	m=CIQ p=15000 n=100000 e=0.5 a=0.25 node similarity = GDV and BLAST protein similarity

We use parameters recommended in the methods’ original publications. The parameter “node similarity” indicates the node similarity information that is inputted to the NA method. Note that graphlet degree vector (GDV) similarity uses only network topological information, while BLAST protein similarity uses only protein sequence information. Note that sometimes different methods use the same name (e.g.,  $\alpha$ ) for different parameters, or they use different names (e.g.,  $\alpha$  versus  $a$ ) for the same parameter. For T alignments, we set parameters such that only topological information is used (i.e., such that  $\theta = 1.0$ ; see Supplementary Section A.1.1). For T+S alignments, we set parameters such that topological and sequence information are equally weighted (i.e., such that  $\theta = 0.5$ ; see Supplementary Section A.1.1), as recommended by Meng et al. [110]. The only exception is ConvexAlign, for which we use a lower  $\theta$  value, as recommended and pre-set in its implementation by its authors. See Supplementary Section A.1.1 for details.

TABLE A.3

## THEORETIC TIME COMPLEXITY

Algorithms	Time complexity
PNA methods	
GHOST	$O(n(\frac{m}{n})^4) = O(\frac{m^4}{n^3})$
L-GRAAL	$O(n^3 + n^2 \frac{m^3}{n}) = O(n^3 + \frac{m^3}{n})$
MAGNA++	$O(n + m)$
WAVE	$O(n^3)$
MNA methods	
IsoRankN	$O(\binom{k}{2} m^2) = O(k^2 m^2)$
ConvexAlign	$O(k^3 n^3)$
BEAMS	$O(k^2 n^2 (\frac{m}{n})^{k+1})$
multiMAGNA++	$O(k(n + m))$

Theoretic time complexity of the considered PNA methods when they align two networks and of the considered MNA methods when they align  $k$  networks, with respect to network size and the number of networks. Regarding network size,  $n$  and  $m$  is the number of nodes and edges, respectively, averaged over all networks under consideration.

TABLE A.4

OVERALL RANKING OF THE NA METHODS FOR THE PE  
FRAMEWORK

NA method	Overall rank	$p_1$ -value	$p_2$ -value	Non-sig (fail)
WAVE (PE-P-P)	1.70 (1.23)	NA	NA	0.00 (0.00)
multiMAGNA++ (PE-M-P)	1.93 (1.32)	2.28e-01	NA	0.00 (0.00)
MAGNA++ (PE-P-P)	3.21 (1.85)	1.39e-04	2.64e-06	0.00 (0.00)
GHOST (PE-P-P)	4.09 (3.66)	1.06e-04	7.27e-04	0.14 (0.14)
LGRAAL (PE-P-P)	4.21 (2.18)	5.13e-08	2.00e-06	0.05 (0.05)
multiMAGNA++ (PE-M-M)	5.09 (1.56)	2.36e-07	6.92e-08	0.00 (0.00)
BEAMS (PE-M-P)	8.74 (1.99)	5.06e-09	6.81e-09	0.02 (0.00)
ConvexAlign (PE-M-M)	9.07 (1.56)	5.23e-09	5.19e-09	0.00 (0.00)
ConvexAlign (PE-M-P)	9.09 (2.27)	5.07e-09	5.02e-09	0.00 (0.00)
BEAMS (PE-M-M)	9.16 (1.90)	5.08e-09	6.59e-09	0.09 (0.00)
IsoRankN (PE-M-P)	9.56 (1.75)	4.80e-09	4.45e-09	0.23 (0.00)
IsoRankN (PE-M-M)	9.63 (2.21)	4.89e-09	4.65e-09	0.33 (0.00)

TABLE A.4 (CONTINUED)

Overall ranking of the NA methods for the **PE framework** over all evaluation tests (where a test is a combination of an NA method, a network pair, and an alignment quality measure) that use **TQ measures**, for T+S alignments, for networks with both known and unknown node mapping. By NA method, here, we mean the combination of a PNA or MNA method and the alignment category (Chapter 2.2.4 of the main document). Namely, there are 12 NA methods in the PE framework (four PNA methods associated with the PE-P-P categories and four MNA methods associated with each of the PE-M-M and PE-M-P categories). The “Overall rank” column shows the rank of each method averaged over all evaluation tests, along with the corresponding standard deviation (in brackets). Since there are 12 methods in a given framework, the possible ranks range from 1 to 12. The lower the rank, the better the given method. The “ $p_1$ -value” column shows the statistical significance of the difference between the ranking of each method and the 1<sup>st</sup> best ranked method. The “ $p_2$ -value” column shows the statistical significance of the difference between the ranking of each method and the 2<sup>nd</sup> best ranked method. The “Frac. non. sig. (failed)” column shows the fraction of evaluation tests in which the alignment quality score is not statistically significant, and, in brackets, the fraction of evaluation tests in which the given NA method failed to produce an alignment.

TABLE A.5

OVERALL RANKING OF THE NA METHODS FOR THE PE  
FRAMEWORK

NA method	Overall rank	$p_1$ -value	$p_2$ -value	Non-sig (fail)
ConvexAlign (PE-M-P)	4.33 (4.25)	NA	NA	0.07 (0.00)
MAGNA++ (PE-P-P)	4.98 (3.43)	3.32e-01	NA	0.11 (0.00)
ConvexAlign (PE-M-M)	5.42 (4.60)	4.29e-02	2.60e-01	0.21 (0.00)
multiMAGNA++ (PE-M-P)	6.14 (4.17)	7.71e-02	4.62e-05	0.19 (0.00)
LGRAAL (PE-P-P)	7.02 (3.82)	2.40e-03	1.10e-03	0.30 (0.05)
WAVE (PE-P-P)	7.21 (4.21)	7.42e-03	6.25e-07	0.25 (0.00)
IsoRankN (PE-M-M)	7.51 (3.48)	2.34e-05	5.87e-04	0.28 (0.00)
multiMAGNA++ (PE-M-M)	7.54 (4.22)	1.55e-03	3.80e-05	0.35 (0.00)
GHOST (PE-P-P)	7.56 (4.33)	2.56e-03	3.61e-06	0.37 (0.16)
IsoRankN (PE-M-P)	7.82 (4.08)	4.04e-05	6.55e-05	0.39 (0.00)
BEAMS (PE-M-P)	8.33 (4.28)	1.50e-05	1.77e-05	0.39 (0.00)
BEAMS (PE-M-M)	8.79 (4.22)	8.96e-06	2.30e-06	0.47 (0.00)

Overall ranking of the NA methods for the **PE framework** over all evaluation tests (where a test is a combination of an NA method, a network pair, and an alignment quality measure) that use **FQ measures**, for T+S alignments, for networks with both known and unknown node mapping. By NA method, here, we mean the combination of a PNA or MNA method and the alignment category (Chapter 2.2.4 of the main document). Namely, there are 12 NA methods in the PE framework (four PNA methods associated with the PE-P-P categories and four MNA methods associated with each of the PE-M-M and PE-M-P categories). The table can be interpreted the same way as Supplementary Table A.4.

TABLE A.6

OVERALL RANKING OF THE NA METHODS FOR THE PE  
FRAMEWORK

NA method	Overall rank	$p_1$ -value	$p_2$ -value	Non-sig (fail)
multiMAGNA++ (PE-M-P)	1.03 (0.18)	NA	NA	0.00 (0.00)
MAGNA++ (PE-P-P)	1.27 (0.91)	1.86e-01	NA	0.00 (0.00)
GHOST (PE-P-P)	1.60 (1.16)	1.31e-02	2.32e-01	0.00 (0.00)
WAVE (PE-P-P)	1.60 (1.45)	2.61e-02	9.42e-02	0.00 (0.00)
LGRAAL (PE-P-P)	3.70 (2.00)	3.77e-05	3.69e-05	0.00 (0.00)
multiMAGNA++ (PE-M-M)	4.97 (1.87)	1.80e-06	2.84e-06	0.00 (0.00)
IsoRankN (PE-M-M)	7.13 (1.85)	9.23e-07	9.36e-07	0.00 (0.00)
IsoRankN (PE-M-P)	7.83 (2.44)	1.47e-06	1.49e-06	0.00 (0.00)
ConvexAlign (PE-M-M)	8.37 (2.97)	1.71e-06	2.03e-06	0.00 (0.00)
BEAMS (PE-M-M)	8.53 (3.50)	5.01e-06	5.24e-06	0.00 (0.00)
BEAMS (PE-M-P)	8.60 (3.56)	5.24e-06	5.36e-06	0.00 (0.00)
ConvexAlign (PE-M-P)	10.60 (1.81)	5.04e-07	5.22e-07	0.00 (0.00)

Overall ranking of the NA methods for the **PE framework** over all evaluation tests (where a test is a combination of an NA method, a network pair, and an alignment quality measure) that use network pairs with **known node mapping**, for T+S alignments, for both TQ and FQ measures. By NA method, here, we mean the combination of a PNA or MNA method and the alignment category (Chapter 2.2.4 of the main document). Namely, there are 12 NA methods in the PE framework (four PNA methods associated with the PE-P-P categories and four MNA methods associated with each of the PE-M-M and PE-M-P categories). The table can be interpreted the same way as Supplementary Table A.4.

TABLE A.7

OVERALL RANKING OF THE NA METHODS FOR THE PE  
FRAMEWORK

NA method	Overall rank	$p_1$ -value	$p_2$ -value	Non-sig (fail)
ConvexAlign (PE-M-P)	4.57 (3.66)	NA	NA	0.06 (0.00)
MAGNA++ (PE-P-P)	5.49 (2.64)	4.81e-02	NA	0.09 (0.00)
multiMAGNA++ (PE-M-P)	5.74 (3.84)	3.70e-02	4.12e-01	0.16 (0.00)
WAVE (PE-P-P)	6.23 (4.32)	1.15e-02	1.52e-01	0.20 (0.00)
ConvexAlign (PE-M-M)	6.40 (4.31)	9.33e-08	8.63e-02	0.17 (0.00)
LGRAAL (PE-P-P)	6.71 (3.62)	2.85e-03	1.66e-01	0.27 (0.07)
multiMAGNA++ (PE-M-M)	7.14 (3.89)	3.71e-03	1.36e-04	0.29 (0.00)
GHOST (PE-P-P)	7.99 (3.83)	1.08e-05	1.83e-05	0.39 (0.21)
BEAMS (PE-M-P)	8.47 (3.47)	3.04e-08	2.33e-06	0.33 (0.00)
IsoRankN (PE-M-P)	8.89 (3.69)	1.43e-10	1.02e-07	0.46 (0.00)
IsoRankN (PE-M-M)	8.97 (3.46)	4.28e-11	3.50e-07	0.43 (0.00)
BEAMS (PE-M-M)	9.13 (3.38)	1.58e-09	7.46e-08	0.44 (0.00)

Overall ranking of the NA methods for the **PE framework** over all evaluation tests (where a test is a combination of an NA method, a network pair, and an alignment quality measure) that use network pairs with **unknown node mapping**, for T+S alignments, for both TQ and FQ measures. By NA method, here, we mean the combination of a PNA or MNA method and the alignment category (Chapter 2.2.4 of the main paper). Namely, there are 12 NA methods in the PE framework (four PNA methods associated with the PE-P-P categories and four MNA methods associated with each of the PE-M-M and PE-M-P categories). The table can be interpreted the same way as Supplementary Table A.4.

TABLE A.8

OVERALL RANKING OF THE NA METHODS FOR THE ME  
FRAMEWORK

NA method	Overall rank	$p_1$ -value	$p_2$ -value	Non-sig (fail)
multiMAGNA++ (ME-M-P)	1.71 (1.25)	NA	NA	0.00 (0.00)
WAVE (ME-P-P)	2.14 (1.46)	3.56e-01	NA	0.00 (0.00)
MAGNA++ (ME-P-P)	3.00 (2.31)	1.01e-01	2.05e-01	0.00 (0.00)
GHOST (ME-P-P)	3.86 (4.02)	9.87e-02	1.40e-01	0.14 (0.00)
multiMAGNA++ (ME-M-M)	4.00 (2.00)	4.46e-02	7.47e-02	0.00 (0.00)
LGRAAL (ME-P-P)	5.00 (3.70)	3.66e-02	2.90e-02	0.14 (0.00)
IsoRankN (ME-M-M)	7.57 (1.13)	1.08e-02	1.11e-02	0.00 (0.00)
ConvexAlign (ME-M-M)	8.71 (2.36)	1.08e-02	1.07e-02	0.00 (0.00)
BEAMS (ME-M-M)	9.14 (2.54)	1.10e-02	1.10e-02	0.29 (0.00)
IsoRankN (ME-M-P)	10.43 (1.99)	1.09e-02	1.10e-02	0.57 (0.00)
BEAMS (ME-M-P)	10.71 (1.89)	1.07e-02	1.09e-02	0.57 (0.00)
ConvexAlign (ME-M-P)	11.43 (0.98)	9.95e-03	1.05e-02	0.43 (0.00)

Overall ranking of the NA methods for the **ME framework** over all evaluation tests (where a test is a combination of an NA method, a network set, and an alignment quality measure) that use **TQ measures**, for T+S alignments, for networks with both known and unknown node mapping. By NA method, here, we mean the combination of a PNA or MNA method and the alignment category (Chapter 2.2.4 of the main document). Namely, there are 12 NA methods in the ME framework (four PNA methods associated with the ME-P-P categories and four MNA methods associated with each of the ME-M-M and ME-M-P categories). The table can be interpreted the same way as Supplementary Table A.4.



TABLE A.9

OVERALL RANKING OF THE NA METHODS FOR THE ME  
FRAMEWORK

NA method	Overall rank	$p_1$ -value	$p_2$ -value	Non-sig (fail)
MAGNA++ (ME-P-P)	4.22 (2.82)	NA	NA	0.00 (0.00)
ConvexAlign (ME-M-M)	5.11 (3.82)	3.83e-01	NA	0.00 (0.00)
ConvexAlign (ME-M-P)	5.44 (5.27)	3.83e-01	6.12e-01	0.11 (0.00)
LGRAAL (ME-P-P)	5.78 (4.18)	3.67e-01	3.63e-01	0.22 (0.00)
GHOST (ME-P-P)	5.89 (4.59)	1.75e-01	4.06e-01	0.11 (0.00)
multiMAGNA++ (ME-M-P)	5.89 (3.98)	7.13e-02	4.06e-01	0.11 (0.00)
IsoRankN (ME-M-M)	6.00 (4.00)	2.20e-01	2.19e-01	0.22 (0.00)
WAVE (ME-P-P)	7.00 (4.47)	1.07e-02	2.36e-01	0.11 (0.00)
multiMAGNA++ (ME-M-M)	7.33 (4.18)	2.36e-02	2.38e-01	0.11 (0.00)
BEAMS (ME-M-M)	7.56 (4.67)	5.32e-02	9.04e-02	0.33 (0.00)
IsoRankN (ME-M-P)	8.78 (3.90)	2.09e-02	5.98e-02	0.44 (0.00)
BEAMS (ME-M-P)	9.00 (4.39)	2.10e-02	4.13e-02	0.56 (0.00)

Overall ranking of the NA methods for the **ME framework** over all evaluation tests (where a test is a combination of an NA method, a network set, and an alignment quality measure) that use **FQ measures**, for T+S alignments, for networks with both known and unknown node mapping. By NA method, here, we mean the combination of a PNA or MNA method and the alignment category (Chapter 2.2.4 of the main document). Namely, there are 12 NA methods in the ME framework (four PNA methods associated with the ME-P-P categories and four MNA methods associated with each of the ME-M-M and ME-M-P categories). The table can be interpreted the same way as Supplementary Table A.4.

TABLE A.10

OVERALL RANKING OF THE NA METHODS FOR THE ME  
FRAMEWORK

NA method	Overall rank	$p_1$ -value	$p_2$ -value	Non-sig (fail)
GHOST (ME-P-P)	1.17 (0.41)	NA	NA	0.00 (0.00)
multiMAGNA++ (ME-M-P)	1.33 (0.82)	5.00e-01	NA	0.00 (0.00)
MAGNA++ (ME-P-P)	1.50 (1.22)	5.00e-01	5.00e-01	0.00 (0.00)
WAVE (ME-P-P)	2.17 (1.83)	1.73e-01	1.86e-01	0.00 (0.00)
LGRAAL (ME-P-P)	3.17 (2.40)	7.45e-02	8.68e-02	0.00 (0.00)
multiMAGNA++ (ME-M-M)	4.17 (2.48)	4.96e-02	4.96e-02	0.00 (0.00)
IsoRankN (ME-M-M)	6.33 (2.66)	2.39e-02	2.72e-02	0.00 (0.00)
IsoRankN (ME-M-P)	7.33 (3.20)	2.67e-02	2.39e-02	0.00 (0.00)
BEAMS (ME-M-M)	8.17 (3.60)	2.67e-02	2.39e-02	0.00 (0.00)
BEAMS (ME-M-P)	8.50 (4.04)	2.84e-02	2.72e-02	0.17 (0.00)
ConvexAlign (ME-M-M)	10.33 (1.03)	1.55e-02	1.55e-02	0.00 (0.00)
ConvexAlign (ME-M-P)	11.17 (2.04)	1.31e-02	1.31e-02	0.17 (0.00)

Overall ranking of the NA methods for the **ME framework** over all evaluation tests (where a test is a combination of an NA method, a network set, and an alignment quality measure) that use network pairs with **known node mapping**, for T+S alignments, for both TQ and FQ measures. By NA method, here, we mean the combination of a PNA or MNA method and the alignment category (Chapter 2.2.4 of the main document). Namely, there are 12 NA methods in the ME framework (four PNA methods associated with the ME-P-P categories and four MNA methods associated with each of the ME-M-M and ME-M-P categories). The table can be interpreted the same way as Supplementary Table A.4.

TABLE A.11

OVERALL RANKING OF THE NA METHODS FOR THE ME  
FRAMEWORK

NA method	Overall rank	$p_1$ -value	$p_2$ -value	Non-sig (fail)
ConvexAlign (ME-M-M)	4.50 (2.76)	NA	NA	0.00 (0.00)
MAGNA++ (ME-P-P)	5.00 (2.31)	3.60e-01	NA	0.00 (0.00)
multiMAGNA++ (ME-M-P)	5.70 (3.80)	2.21e-01	4.39e-01	0.10 (0.00)
ConvexAlign (ME-M-P)	6.20 (5.33)	1.42e-01	3.04e-01	0.30 (0.00)
WAVE (ME-P-P)	6.50 (4.45)	1.92e-01	2.06e-01	0.10 (0.00)
LGRAAL (ME-P-P)	6.80 (4.02)	1.06e-01	3.41e-01	0.30 (0.00)
IsoRankN (ME-M-M)	6.90 (3.48)	3.96e-03	1.92e-01	0.20 (0.00)
multiMAGNA++ (ME-M-M)	6.90 (4.07)	1.10e-01	1.00e-01	0.10 (0.00)
GHOST (ME-P-P)	7.30 (3.95)	6.28e-02	1.20e-01	0.20 (0.00)
BEAMS (ME-M-M)	8.30 (4.19)	2.05e-02	6.30e-02	0.50 (0.00)
BEAMS (ME-M-P)	10.50 (3.17)	2.86e-03	7.06e-03	0.80 (0.00)
IsoRankN (ME-M-P)	10.80 (2.57)	2.82e-03	7.06e-03	0.80 (0.00)

Overall ranking of the NA methods for the **ME framework** over all evaluation tests (where a test is a combination of an NA method, a network set, and an alignment quality measure) that use network pairs with **unknown node mapping**, for T+S alignments, for both TQ and FQ measures. By NA method, here, we mean the combination of a PNA or MNA method and the alignment category (Chapter 2.2.4 of the main document). Namely, there are 12 NA methods in the ME framework (four PNA methods associated with the ME-P-P categories and four MNA methods associated with each of the ME-M-M and ME-M-P categories). The table can be interpreted the same way as Supplementary Table A.4.

TABLE A.12

OVERALL RANKING OF THE NA METHODS FOR THE ME  
FRAMEWORK

NA method	Overall rank
multiMAGNA++ (ME-M-M)	2.25 (1.50)
MAGNA++ (ME-P-P)	3.25 (3.20)
ConvexAlign (ME-M-P)	4.25 (1.71)
GHOST (ME-P-P)	4.25 (2.36)
LGRAAL (ME-P-P)	4.25 (1.50)
WAVE (ME-P-P)	5.00 (2.45)
multiMAGNA++ (ME-M-P)	6.25 (1.50)
IsoRankN (ME-M-M)	7.75 (3.30)
ConvexAlign (ME-M-M)	8.25 (0.96)
IsoRankN (ME-M-P)	9.50 (0.58)

Overall ranking of the NA methods for the **ME framework** over all evaluation tests (where a test is a combination of an NA method and a network set) that use the **mean normalized entropy measure**, for T alignments. By NA method, here, we mean the combination of a PNA or MNA method and the alignment category (Chapter 2.2.4 of the main document). Namely, there are 12 NA methods in the ME framework (four PNA methods associated with the ME-P-P categories and four MNA methods associated with each of the ME-M-M and ME-M-P categories). The alignment categories are color coded. The “Overall rank” column shows the rank of each method averaged over all evaluation tests, along with the corresponding standard deviation (in brackets). Since there are 12 methods in a given framework, the possible ranks range from 1 to 12. The lower the rank, the better the given method.

TABLE A.13

OVERALL RANKING OF THE NA METHODS FOR THE ME  
FRAMEWORK

NA method	Overall rank
LGRAAL (ME-P-P)	3.5 (1.00)
IsoRankN (ME-M-M)	4.25 (4.72)
multiMAGNA++ (ME-M-P)	5.25 (4.35)
MAGNA++ (ME-P-P)	5.50 (4.43)
ConvexAlign (ME-M-M)	6.75 (2.22)
multiMAGNA++ (ME-M-M)	7.00 (4.05)
WAVE (ME-P-P)	7.00 (2.94)
BEAMS (ME-M-P)	7.25 (4.11)
IsoRankN (ME-M-P)	7.5 (4.20)
GHOST (ME-P-P)	7.5 (4.79)
BEAMS (ME-M-M)	8.25 (1.50)
ConvexAlign (ME-M-P)	8.25 (2.06)

Overall ranking of the NA methods for the **ME framework** over all evaluation tests (where a test is a combination of an NA method and a network set) that use the **mean normalized entropy measure**, for T+S alignments. By NA method, here, we mean the combination of a PNA or MNA method and the alignment category (Chapter 2.2.4 of the main document). Namely, there are 12 NA methods in the ME framework (four PNA methods associated with the ME-P-P categories and four MNA methods associated with each of the ME-M-M and ME-M-P categories). The table can be interpreted the same way as Supplementary Table A.12.

TABLE A.14

OVERALL RANKING OF THE NA METHODS FOR THE ME  
FRAMEWORK

NA method	Overall rank	$p_1$ -value	$p_2$ -value	Non-sig (fail)
multiMAGNA++ (ME-M-P)	1.71 (1.25)	NA	NA	0.00 (0.00)
WAVE (ME-P-P)	2.29 (1.60)	2.85e-01	NA	0.00 (0.00)
MAGNA++ (ME-P-P)	3.29 (2.75)	9.87e-02	2.05e-01	0.00 (0.00)
GHOST (ME-P-P)	4.00 (4.08)	1.01e-01	1.40e-01	0.14 (0.00)
multiMAGNA++ (ME-M-M)	4.14 (1.86)	3.67e-02	7.47e-02	0.00 (0.00)
LGRAAL (ME-P-P)	5.14 (3.63)	2.92e-02	2.90e-02	0.14 (0.00)
IsoRankN (ME-M-M)	7.86 (1.35)	1.07e-02	1.12e-02	0.00 (0.00)
GEDEVO-M (ME-M-M)	8.80 (4.66)	2.95e-02	5.28e-02	0.00 (0.00)
ConvexAlign (ME-M-M)	9.14 (2.41)	1.10e-02	1.11e-02	0.00 (0.00)
BEAMS (ME-M-M)	9.43 (2.64)	1.09e-02	1.10e-02	0.29 (0.00)
IsoRankN (ME-M-P)	10.71 (2.29)	1.10e-02	1.11e-02	0.57 (0.00)
BEAMS (ME-M-P)	11.00 (2.16)	1.01e-02	1.10e-02	0.57 (0.00)
ConvexAlign (ME-M-P)	12.00 (1.15)	1.07e-02	1.08e-02	0.43 (0.00)

Overall ranking of the NA methods for the **ME framework** over all evaluation tests (where a test is a combination of an NA method, a network set, and an alignment quality measure) that use **TQ measures**, for T+S alignments, for networks with both known and unknown node mapping. The table mimics the analyses from Supplementary Table A.8 with the inclusion of an additional method, GEDEVO-M associated with the ME-M-M category.

TABLE A.15

OVERALL RANKING OF THE NA METHODS FOR THE ME  
FRAMEWORK

NA method	Overall rank	$p_1$ -value	$p_2$ -value	Non-sig (fail)
MAGNA++ (ME-P-P)	4.22 (2.82)	NA	NA	0.00 (0.00)
ConvexAlign (ME-M-M)	5.11 (3.82)	3.83e-01	NA	0.00 (0.00)
ConvexAlign (ME-M-P)	5.56 (5.43)	3.83e-01	6.12e-01	0.11 (0.00)
LGRAAL (ME-P-P)	5.89 (4.37)	3.67e-01	3.37e-01	0.22 (0.00)
GHOST (ME-P-P)	6.00 (4.77)	1.75e-01	3.83e-01	0.11 (0.00)
multiMAGNA++ (ME-M-P)	6.00 (4.18)	7.13e-02	3.83e-01	0.11 (0.00)
IsoRankN (ME-M-M)	6.11 (4.20)	2.20e-01	2.02e-01	0.22 (0.00)
WAVE (ME-P-P)	7.11 (4.62)	1.07e-02	2.20e-01	0.11 (0.00)
multiMAGNA++ (ME-M-M)	7.44 (4.33)	2.36e-02	2.20e-01	0.11 (0.00)
BEAMS (ME-M-M)	7.67 (4.80)	5.32e-02	8.02e-02	0.33 (0.00)
IsoRankN (ME-M-P)	9.00 (4.12)	2.07e-02	4.80e-02	0.44 (0.00)
BEAMS (ME-M-P)	9.33 (4.66)	2.10e-02	3.28e-02	0.56 (0.00)
GEDEVO-M (ME-M-M)	12.50 (0.84)	1.68e-02	1.78e-02	0.33 (0.00)

Overall ranking of the NA methods for the **ME framework** over all evaluation tests (where a test is a combination of an NA method, a network set, and an alignment quality measure) that use **FQ measures**, for T+S alignments, for networks with both known and unknown node mapping. The table mimics the analyses from Supplementary Table A.9 with the inclusion of an additional method, GEDEVO-M associated with the ME-M-M category.

TABLE A.16

OVERALL RANKING OF THE NA METHODS FOR THE ME  
FRAMEWORK

NA method	Overall rank	$p_1$ -value	$p_2$ -value	Non-sig (fail)
GHOST (ME-P-P)	1.17 (0.41)	NA	NA	0.00 (0.00)
multiMAGNA++ (ME-M-P)	1.33 (0.82)	5.00e-01	NA	0.00 (0.00)
MAGNA++ (ME-P-P)	1.50 (1.22)	5.00e-01	5.00e-01	0.00 (0.00)
WAVE (ME-P-P)	2.17 (1.83)	1.73e-01	1.86e-01	0.00 (0.00)
LGRAAL (ME-P-P)	3.17 (2.40)	7.45e-02	8.68e-02	0.00 (0.00)
multiMAGNA++ (ME-M-M)	4.17 (2.48)	4.96e-02	4.96e-02	0.00 (0.00)
IsoRankN (ME-M-M)	6.33 (2.66)	2.39e-02	2.72e-02	0.00 (0.00)
IsoRankN (ME-M-P)	7.33 (3.20)	2.67e-02	2.39e-02	0.00 (0.00)
BEAMS (ME-M-M)	8.17 (3.60)	2.67e-02	2.39e-02	0.00 (0.00)
BEAMS (ME-M-P)	8.67 (4.23)	2.90e-02	2.84e-02	0.17 (0.00)
ConvexAlign (ME-M-M)	10.50 (1.22)	1.70e-02	1.70e-02	0.00 (0.00)
ConvexAlign (ME-M-P)	11.50 (2.26)	1.68e-02	1.68e-02	0.17 (0.00)
GEDEVO-M (ME-M-M)	12.33 (0.82)	1.68e-02	1.70e-02	0.00 (0.00)

Overall ranking of the NA methods for the **ME framework** over all evaluation tests (where a test is a combination of an NA method, a network set, and an alignment quality measure) that use network pairs with **known node mapping**, for T+S alignments, for both TQ and FQ measures. The table mimics the analyses from Supplementary Table A.10 with the inclusion of an additional method, GEDEVO-M associated with the ME-M-M category.



TABLE A.17

OVERALL RANKING OF THE NA METHODS FOR THE ME  
FRAMEWORK

NA method	Overall rank	$p_1$ -value	$p_2$ -value	Non-sig (fail)
ConvexAlign (ME-M-M)	4.70 (3.02)	NA	NA	0.00 (0.00)
MAGNA++ (ME-P-P)	5.20 (2.44)	3.60e-01	NA	0.00 (0.00)
multiMAGNA++ (ME-M-P)	5.80 (3.99)	2.86e-01	4.80e-01	0.10 (0.00)
ConvexAlign (ME-M-P)	6.50 (5.66)	1.42e-01	3.04e-01	0.30 (0.00)
WAVE (ME-P-P)	6.70 (4.52)	2.07e-01	2.06e-01	0.10 (0.00)
LGRAAL (ME-P-P)	7.00 (4.08)	1.17e-01	3.41e-01	0.30 (0.00)
multiMAGNA++ (ME-M-M)	7.10 (4.09)	1.10e-01	1.10e-01	0.10 (0.00)
IsoRankN (ME-M-M)	7.20 (3.74)	3.96e-03	1.92e-01	0.20 (0.00)
GHOST (ME-P-P)	7.50 (4.03)	7.61e-02	1.30e-01	0.20 (0.00)
BEAMS (ME-M-M)	8.60 (4.38)	2.06e-02	6.30e-02	0.50 (0.00)
GEDEVO-M (ME-M-M)	9.00 (4.85)	1.39e-01	2.05e-01	0.40 (0.00)
BEAMS (ME-M-P)	10.90 (3.41)	2.91e-03	7.12e-03	0.80 (0.00)
IsoRankN (ME-M-P)	11.20 (2.82)	2.86e-03	7.12e-03	0.80 (0.00)

Overall ranking of the NA methods for the **ME framework** over all evaluation tests (where a test is a combination of an NA method, a network set, and an alignment quality measure) that use networks pairs with **unknown node mapping**, for T+S alignments, for both TQ and FQ measures. The table mimics the analyses from Supplementary Table A.11 with the inclusion of an additional method, GEDEVO-M associated with the ME-M-M category.

TABLE A.18

OVERALL RANKING OF THE NA METHODS FOR THE ME  
FRAMEWORK

NA method	Overall rank	$p_1$ -value	$p_2$ -value	Non-sig (fail)
MAGNA++ (ME-P-P)	3.81 (2.74)	NA	NA	0.00 (0.00)
multiMAGNA++ (ME-M-P)	4.12 (3.84)	5.18e-01	NA	0.06 (0.00)
WAVE (ME-P-P)	5.00 (4.31)	1.26e-01	3.91e-02	0.06 (0.00)
GHOST (ME-P-P)	5.12 (4.46)	1.98e-01	1.52e-01	0.12 (0.00)
LGRAAL (ME-P-P)	5.56 (3.95)	1.24e-01	8.38e-02	0.19 (0.00)
multiMAGNA++ (ME-M-M)	6.00 (3.78)	1.87e-02	5.39e-03	0.06 (0.00)
ConvexAlign (ME-M-M)	6.88 (3.79)	3.91e-02	8.88e-02	0.00 (0.00)
IsoRankN (ME-M-M)	6.88 (3.30)	1.32e-02	1.97e-02	0.12 (0.00)
ConvexAlign (ME-M-P)	8.38 (5.21)	1.68e-02	1.42e-02	0.25 (0.00)
BEAMS (ME-M-M)	8.44 (3.98)	3.42e-03	5.35e-03	0.31 (0.00)
IsoRankN (ME-M-P)	9.75 (3.45)	6.25e-04	1.01e-03	0.50 (0.00)
BEAMS (ME-M-P)	10.06 (3.77)	6.50e-04	1.21e-03	0.56 (0.00)
GEDEVO-M (ME-M-M)	10.82 (3.57)	5.36e-03	2.90e-03	0.18 (0.00)

Overall ranking of the NA methods for the **ME framework** over all evaluation tests (where a test is a combination of an NA method, a network set, and an alignment quality measure) for T+S alignments, for both TQ and FQ measures, for networks with both known and unknown node mapping. The table mimics the analyses from View I of Figure 5 from the main document, with the inclusion of an additional method, GEDEVO-M associated with the ME-M-M category.

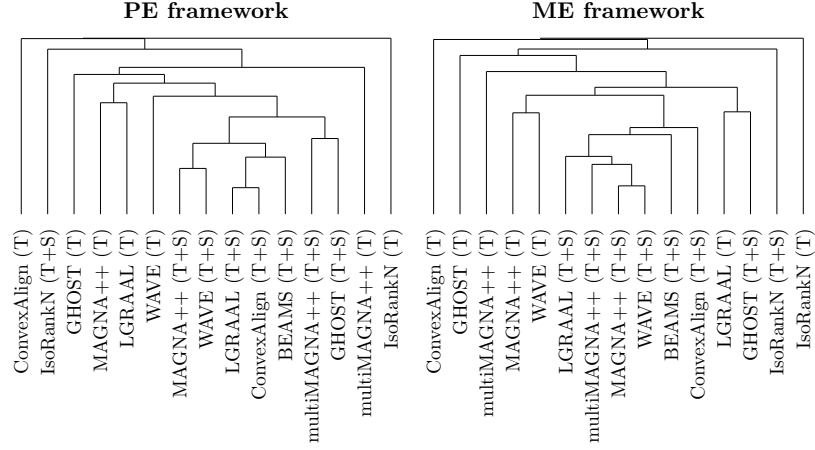


Figure A.1. Clustering of NA methods, each with its T and T+S versions, using each of the **PE** and **ME** frameworks. Clustering is based on pairwise method similarities, which we compute as follows. The similarity between two NA methods is the mean of the Adjusted Rand Index (ARI; explained below) of each pair of corresponding alignments produced by the two NA methods, over all network pairs/sets. Each alignment of a network pair/set is a set of node groups, i.e., a partition of the nodes in all of the networks in the network pair/set, and we measure similarity between two alignments by comparing their partitions using ARI. ARI [167] is a widely used measure to calculate the similarity between two partitions. Given the similarities between all pairs of the NA methods, we cluster using complete linkage hierarchical clustering [50] and visualize the clustering using a dendrogram. The results shown in this figure rely on all alignments over all network sets (Yeast+%LC, PHY<sub>1</sub>, PHY<sub>2</sub>, Y2H<sub>1</sub>, and Y2H<sub>2</sub>). Equivalent results broken down into results for networks with known node mapping and results for networks with unknown node mapping are shown in Supplementary Figs. A.2 and A.3, respectively.

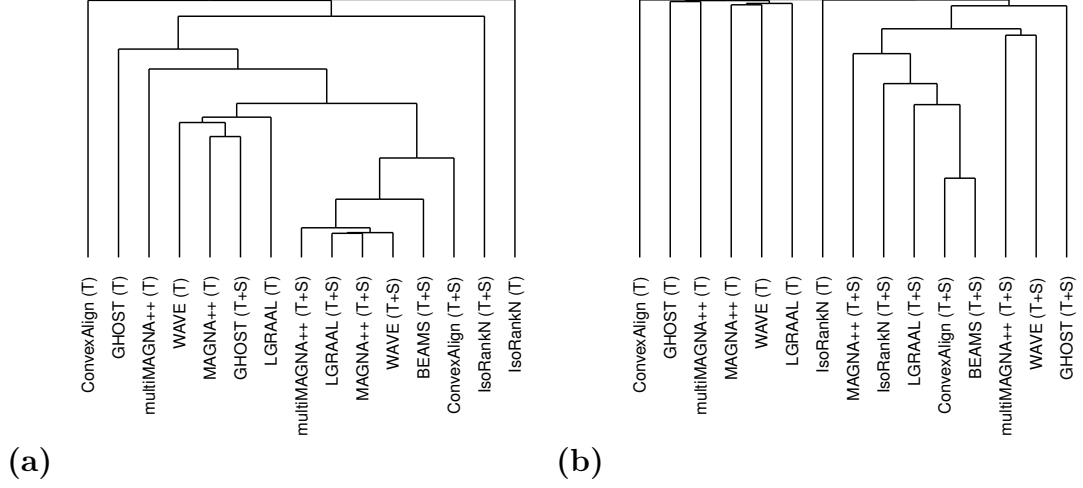


Figure A.2. Clustering of NA methods, each with its T and T+S versions, using all network sets with (a) **known node mapping** and (b) **unknown node mapping** in the **PE framework**. The figure can be interpreted the same way as Supplementary Fig. A.1.

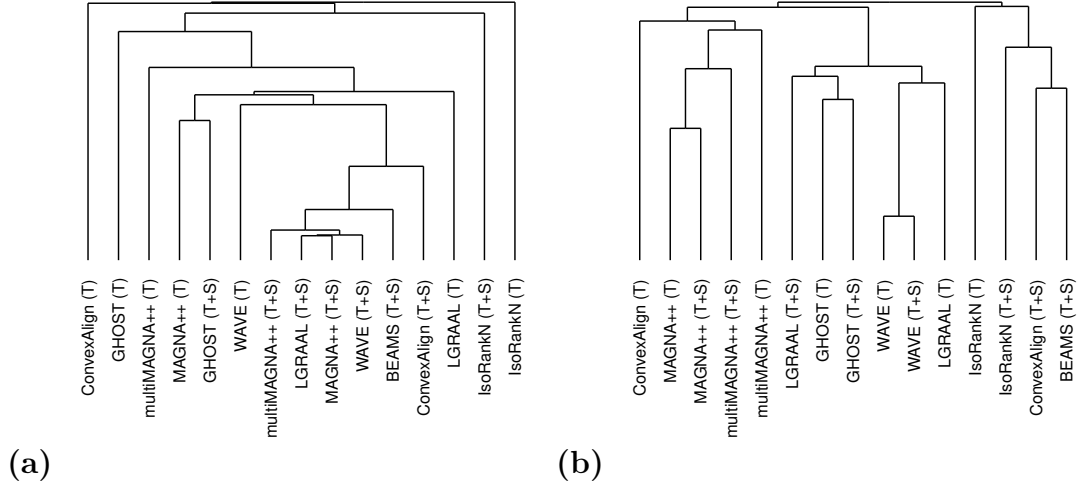


Figure A.3. Clustering of NA methods, each with its T and T+S versions, using all network sets with (a) **known node mapping** and (b) **unknown node mapping** in the **ME framework**. The figure can be interpreted the same way as Supplementary Fig. A.1.

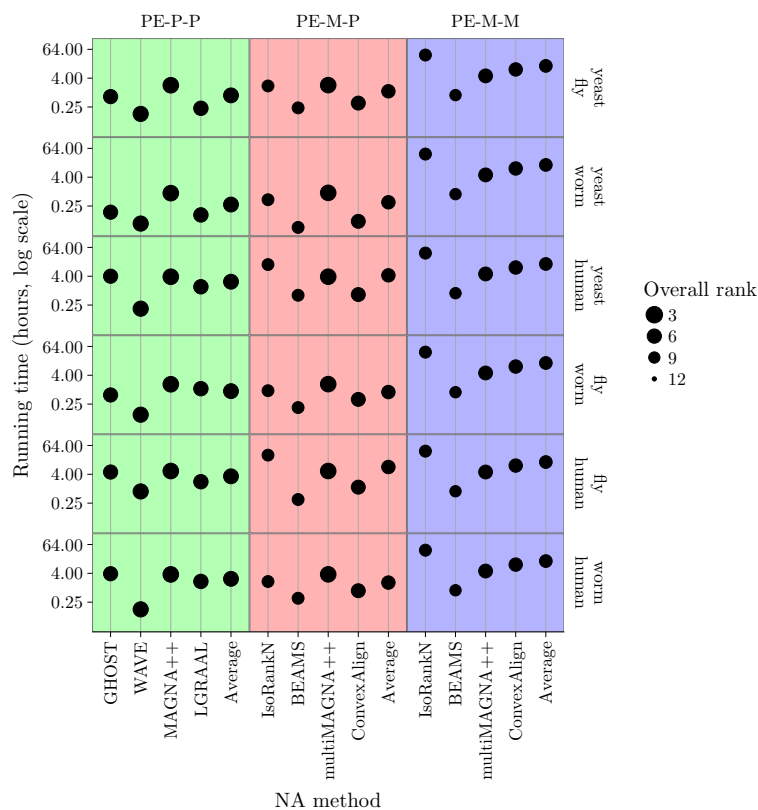


Figure A.4. Overall ranking of an NA method versus its running time for the **PE framework** over all evaluation tests (where a test is a combination of an NA method, a network pair, and an alignment quality measure). By NA method, here, we mean the combination of a PNA or MNA method and the alignment category (Chapter 2.2.4 of the main document). Namely, there are 12 NA methods in the PE framework (four PNA methods associated with the PE-P-P categories and four MNA methods associated with each of the PE-M-M and PE-M-P categories). The running time results are when aligning all network pairs in the Y2H<sub>1</sub> network set, where each method is restricted to use a **single core**. The size of each point visualizes the overall ranking of the corresponding method over all evaluation tests over all network pairs/sets, corresponding to the “Overall rank” column in View I of Fig. 2.5 in the main document; the larger the point size, the better the method. In order to allow for easier comparison between the different alignment categories, “Average” shows the average running times and average rankings of the methods in each alignment category.

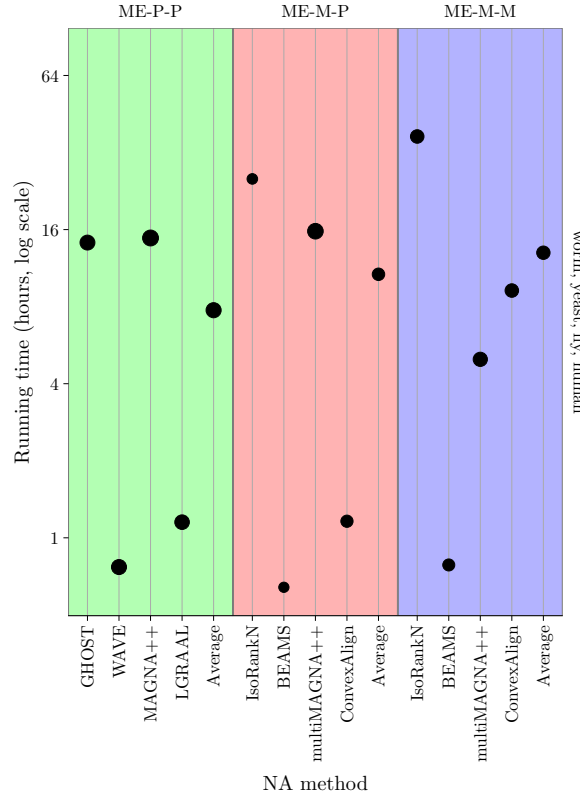
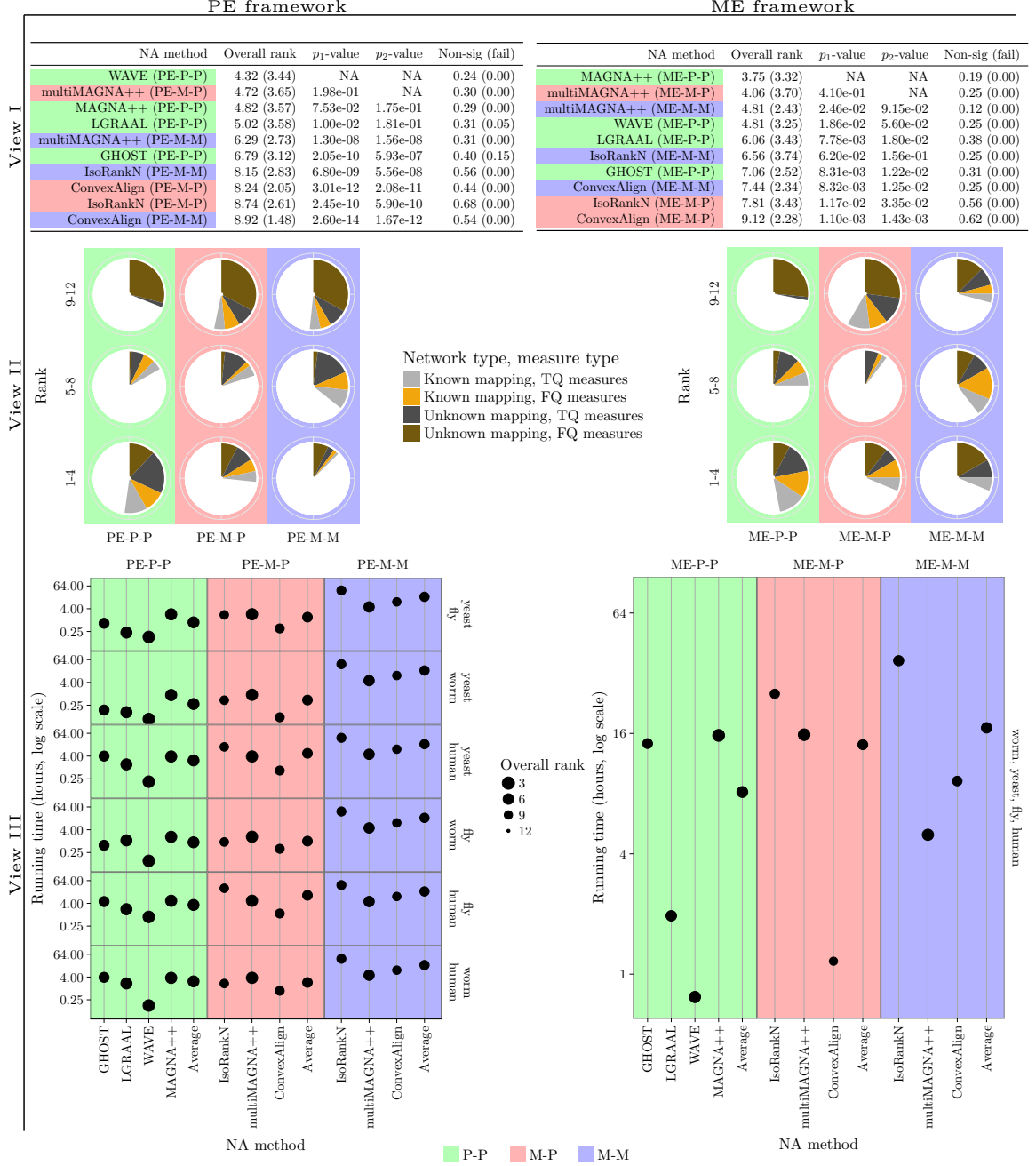


Figure A.5. Overall ranking of an NA method versus its running time for the **ME framework** over all evaluation tests (where a test is a combination of an NA method, a network pair, and an alignment quality measure). By NA method, here, we mean the combination of a PNA or MNA method and the alignment category (Chapter 2.2.4 of the main document). Namely, there are 12 NA methods in the ME framework (four PNA methods associated with the ME-P-P categories and four MNA methods associated with each of the ME-M-M and ME-M-P categories). The running time results are when aligning the Y2H<sub>1</sub> network set, where each method is restricted to use a **single core**. The size of each point visualizes the overall ranking of the corresponding method over all evaluation tests over all network pairs/sets, corresponding to the “Overall rank” column in View I of Fig. 2.5 in the main document; the larger the point size, the better the method. In order to allow for easier comparison between the different alignment categories, “Average” shows the average running times and average rankings of the methods in each alignment category.

Figure A.6. Method comparison results for each of the **PE** and **ME** frameworks over all evaluation tests (where a test is a combination of an NA method, a network pair/set, and an alignment quality measure), for T alignments. By NA method, here, we mean the combination of a PNA or MNA method and the alignment category (Chapter 2.2.4 of the main document). Namely, there are 12 NA methods in the PE framework (four PNA methods associated with the PE-P-P categories and four MNA methods associated with each of the PE-M-M and PE-M-P categories) and 12 NA methods in the ME framework (four PNA methods associated with the ME-P-P categories and four MNA methods associated with each of the ME-M-M and ME-M-P categories). The alignment categories are color coded. **View I.** Overall ranking of the NA methods. The “Overall rank” column shows the rank of each method averaged over all evaluation tests, along with the corresponding standard deviation (in brackets). Since there are 12 methods in a given framework, the possible ranks range from 1 to 12. The lower the rank, the better the given method. The “ $p_1$ -value” column shows the statistical significance of the difference between the ranking of each method and the 1<sup>st</sup> best ranked method. The “ $p_2$ -value” column shows the statistical significance of the difference between the ranking of each method and the 2<sup>nd</sup> best ranked method. The “Non. sig. (fail)” column shows the fraction of evaluation tests in which the alignment quality score is not statistically significant, and, in brackets, the fraction of evaluation tests in which the given NA method failed to produce an alignment. Equivalent results over all evaluation tests broken down into functional and topological alignment quality measures, as well as over all evaluation tests broken down into network pairs/sets with known and unknown node mapping, are shown in Supplementary Tables A.4–A.11. **View II.** Alternative view of ranking of the NA methods. Each pie chart shows the fraction of evaluation test ranks that fall into the 1–4, 5–8, and 9–12 rank bins out of all evaluation test ranks in the given alignment category. For example, for the PE framework, in the PE-P-P alignment category, 56%, 26%, and 18% of the evaluation test ranks fall into ranks 1–4, 5–8, and 9–12, respectively, totaling to 100% of the evaluation test ranks in the PE-P-P alignment category. The pie charts allow us to compare the three alignment categories rather than individual NA methods in each category. The larger the pie chart for the better (lower) ranks, and the smaller the pie chart for the worse (higher) ranks, the better the alignment category. For example, in the PE framework, PE-P-P has the most evaluation tests ranked 1–4 and the fewest evaluation tests ranked 9–12, followed by PE-M-P, followed by PE-M-M. This implies that PE-P-P is superior to PE-M-P and PE-M-M. The pie charts are color coded with respect to alignments of network pairs/sets with known and unknown node mapping, and FQ and TQ measures.

**View III.** Overall ranking of an NA method versus its running time. The latter are running time results when aligning all network pairs in the Y2H<sub>1</sub> network set under the PE framework, and when aligning the Y2H<sub>1</sub> network set under the ME framework, where each method is restricted to use a maximum of 64 cores. The size of each point visualizes the overall ranking of the corresponding method over all evaluation tests over all network pairs/sets, corresponding to the “Overall rank” column in View I; the larger the point size, the better the method. In order to allow for easier comparison between the different alignment categories, “Average” shows the average running times and average rankings of the methods in each alignment category.





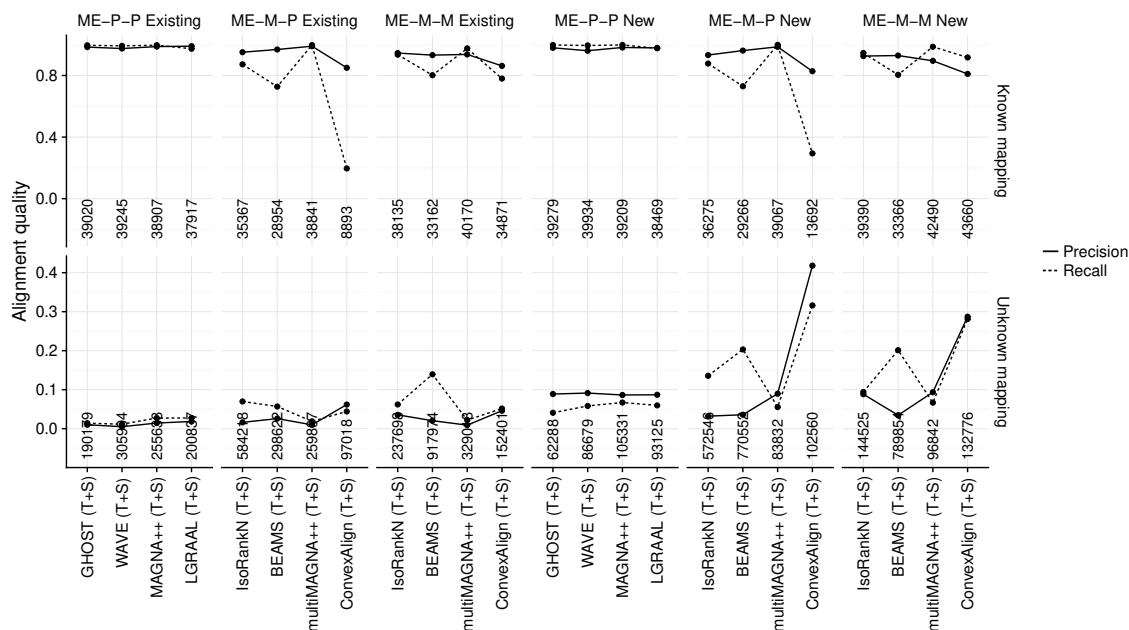


Figure A.7. Comparison of protein function prediction accuracy between the **new** (approach 3) versus the **existing** prediction approach for multiple alignments (approach 2), for all alignments from the ME framework (i.e., ME-P-P, ME-M-P, and ME-M-M categories). We calculate the prediction accuracy as described in Fig. 2.6 in the main document. Each column shows the precision and recall achieved by the new or existing prediction approach for each NA method, as well as the number of predictions made by the approach. The alignments are separated into networks sets with known and unknown mapping.

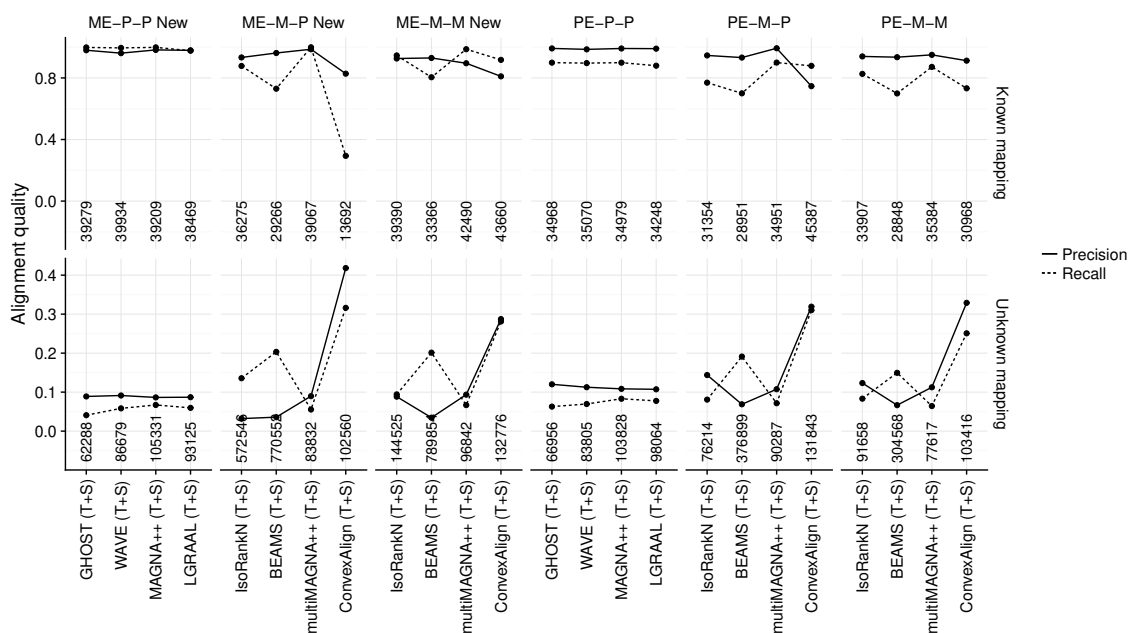


Figure A.8. Comparison of protein function prediction accuracy under the **PE framework** (i.e., PE-P-P, PE-M-P, and PE-M-M categories) and **ME framework** (i.e., ME-P-P, ME-M-P, and ME-M-M categories). We calculate the prediction accuracy as described in Fig. 2.6 in the main document. Each column shows the precision and recall achieved by the new or existing prediction approach for each NA method, as well as the number of predictions made by the approach. The alignments are separated into networks sets with known and unknown mapping.

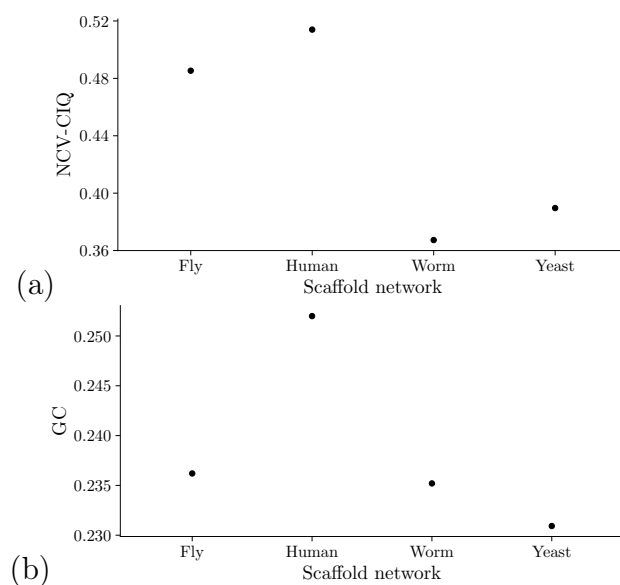


Figure A.9: Illustration of the effect of the choice of scaffold network on alignment quality when combining pairwise alignments into a multiple alignment. These are representative results for one of the analyzed TQ measures (NCV-CIQ; panel (a)), one of the analyzed FQ measures (GO correctness – GC; panel (b)), one of the analyzed network sets (Y2H1), and one of the analyzed NA methods (WAVE). Clearly, different choices of scaffold network ( $x$ -axis) yield different alignment quality scores ( $y$ -axis). The same holds for other combinations of alignment quality measures, network sets, and NA methods. In our evaluation, of all scaffold network choices, the one that yields the best multiple alignment is chosen. In this particular representative scenario, it is the human network that was chosen as the scaffold, since this scaffold choice clearly yields significantly better alignment quality than any other scaffold choice.

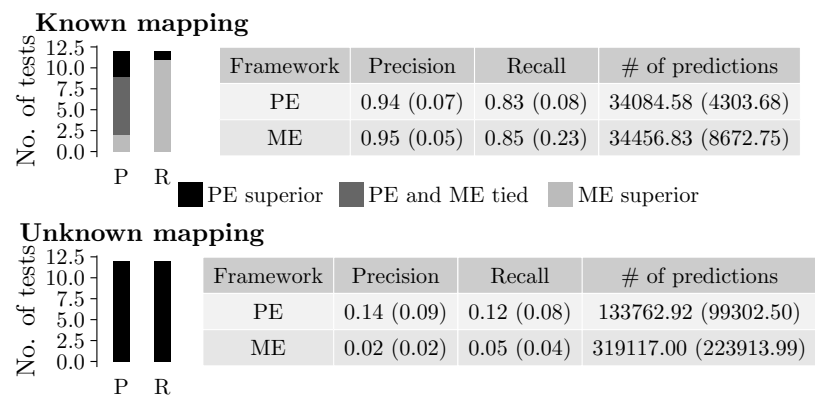


Figure A.10. Comparison of protein function prediction accuracy under the the PE and ME frameworks, where we use approach 2 for the ME framework (rather than using approach 3 for the ME framework like we do in Fig. 2.7 of the main document). The figure can be interpreted the same way as Fig. 2.6 in the main document.

### A.3 Supplementary files

File A.1. GuS022022D\_supplementary1.csv. Detailed alignment quality scores for the PE framework.

File A.2. GuS022022D\_supplementary2.csv. Detailed alignment quality scores for the ME framework.

## APPENDIX B

### HETEROGENEOUS NETWORK ALIGNMENT

#### B.1 Results

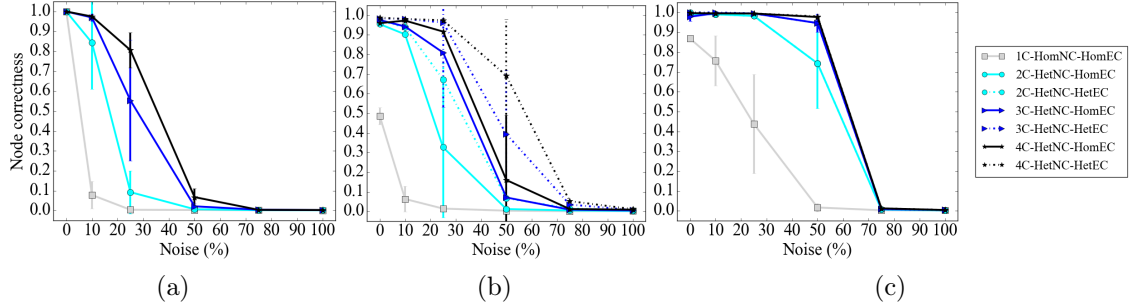


Figure B.1. Detailed alignment quality results regarding the effect of the **number of node colors** on alignment quality as a function of noise level for **synthetic, specifically geometric**, networks using (a) WAVE, (b) MAGNA++, and (c) SANA. Gray squares, light blue circles, dark blue triangles, and black stars indicate the aligned networks containing one, two, three, and four node colors, respectively. For two or more node colors, solid lines represent using HetNC-HomEC, and dashed lines represent using HetNC-HetEC.

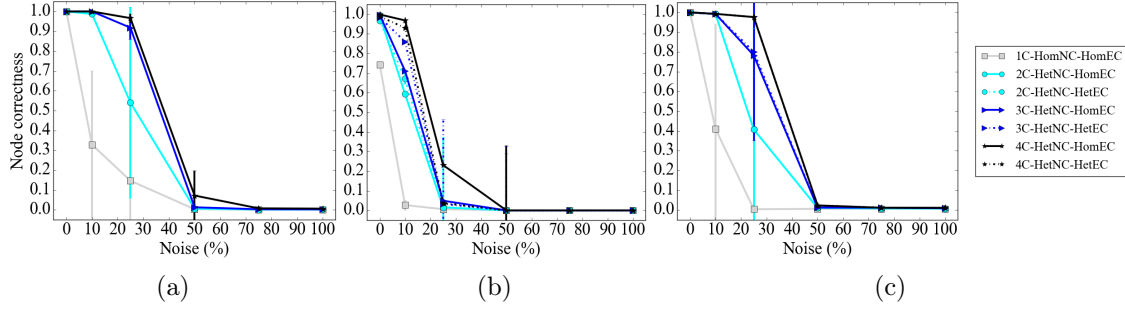


Figure B.2. Detailed alignment quality results regarding the effect of the **number of node colors** on alignment quality as a function of noise level for **synthetic, specifically scale-free**, networks using (a) WAVE, (b) MAGNA++, and (c) SANA. The figure can be interpreted in the same way as Supplementary Figure B.1.

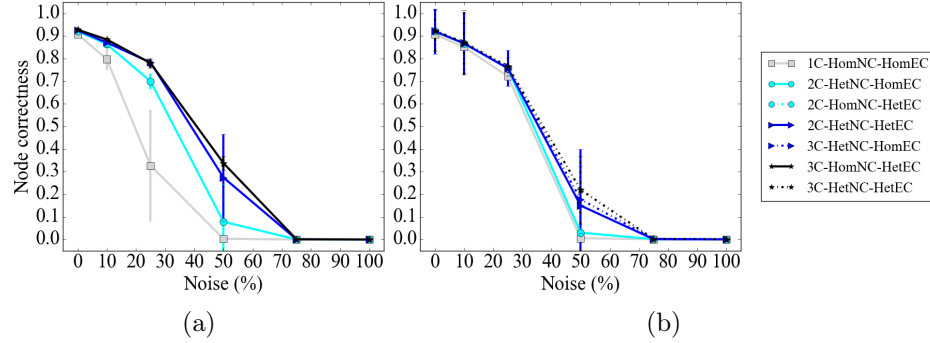


Figure B.3. Detailed alignment quality results regarding the effect of the **number of node colors** on alignment quality as a function of noise level for **PPI, specifically APMS-Expr**, networks using (a) WAVE and (b) SANA. The figure can be interpreted in the same way as Supplementary Figure B.1. Recall that for these larger networks, we have not run MAGNA++ due to its high computational complexity.



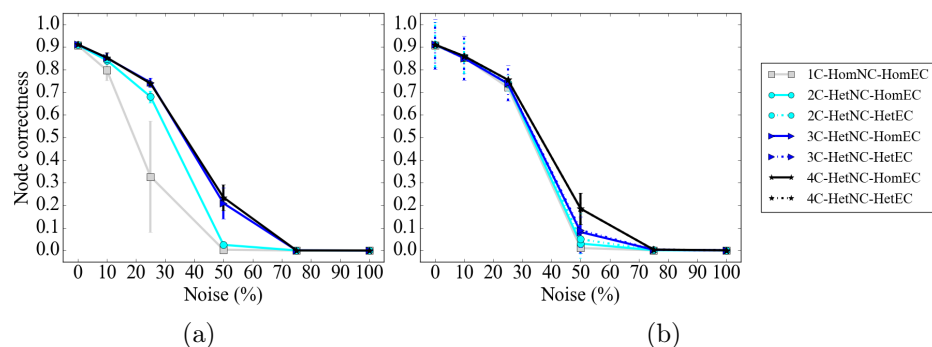


Figure B.4. Detailed alignment quality results regarding the effect of the **number of node colors** on alignment quality as a function of noise level for **PPI, specifically APMS-Seq**, networks using (a) WAVE and (b) SANA. The figure can be interpreted in the same way as Supplementary Figure B.1. Recall that for these larger networks, we have not run MAGNA++ due to its high computational complexity.

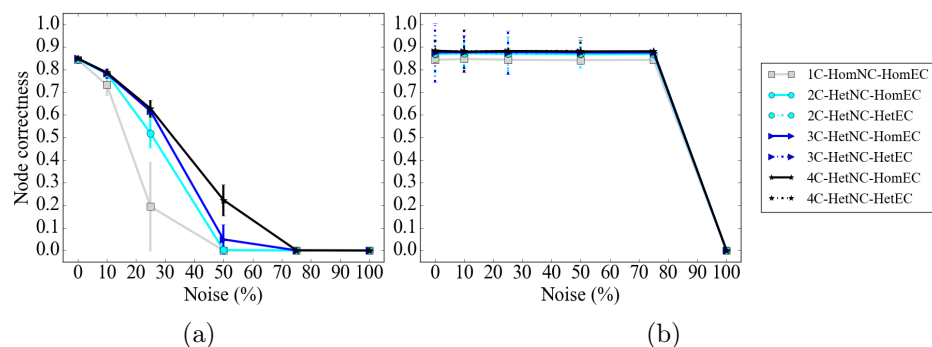


Figure B.5. Detailed alignment quality results regarding the effect of the **number of node colors** on alignment quality as a function of noise level for **PPI, specifically Y2H-Expr**, networks using (a) WAVE and (b) SANA. The figure can be interpreted in the same way as Supplementary Figure B.1. Recall that for these larger networks, we have not run MAGNA++ due to its high computational complexity.

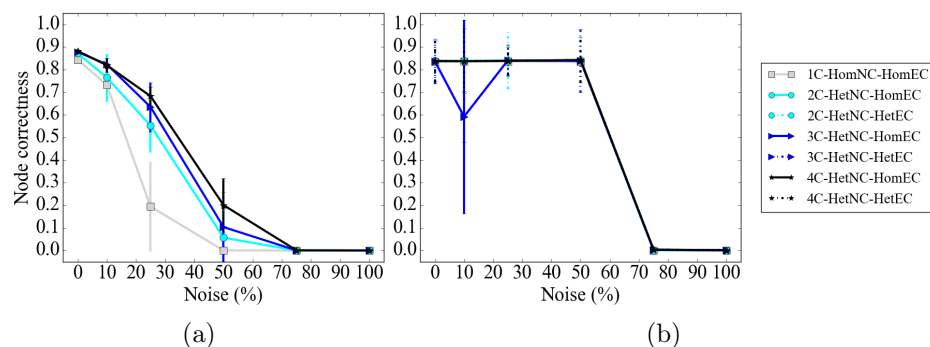


Figure B.6. Detailed alignment quality results regarding the effect of the **number of node colors** on alignment quality as a function of noise level for **PPI, specifically Y2H-Seq**, networks using (a) WAVE and (b) SANA. The figure can be interpreted in the same way as Supplementary Figure B.1. Recall that for these larger networks, we have not run MAGNA++ due to its high computational complexity.

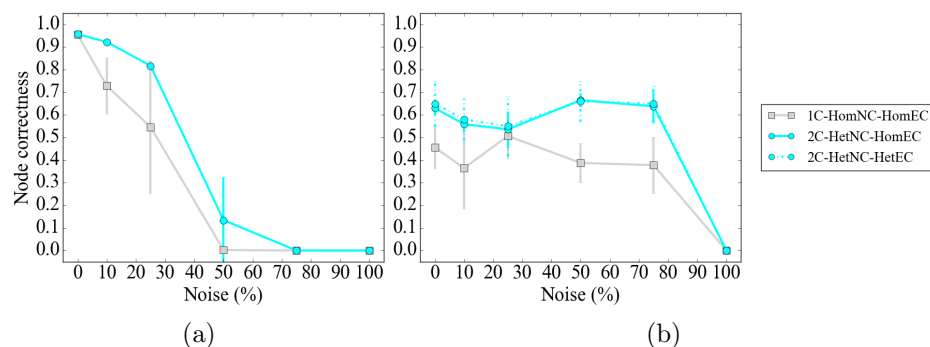


Figure B.7. Detailed alignment quality results regarding the effect of the **number of node colors** on alignment quality as a function of noise level for **protein-GO, specifically protein-GO-APMS**, networks using (a) WAVE and (b) SANA. The figure can be interpreted in the same way as Supplementary Figure B.1. Recall that for these larger networks, we have not run MAGNA++ due to its high computational complexity.

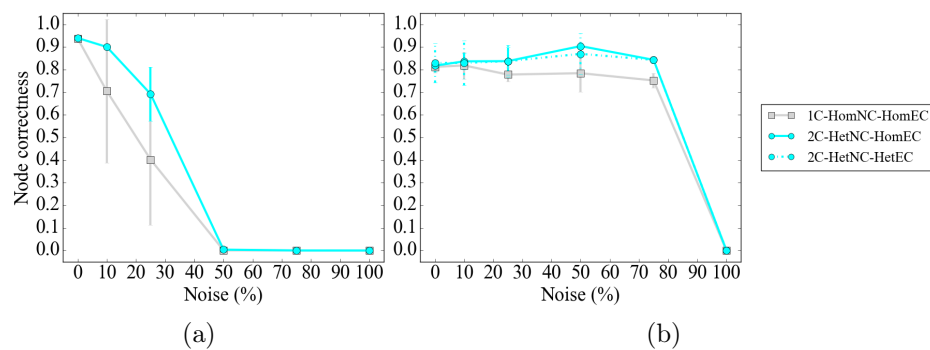


Figure B.8. Detailed alignment quality results regarding the effect of the **number of node colors** on alignment quality as a function of noise level for **protein-GO**, specifically **protein-GO-Y2H**, networks using (a) WAVE and (b) SANA. The figure can be interpreted in the same way as Supplementary Figure B.1. Recall that for these larger networks, we have not run MAGNA++ due to its high computational complexity.

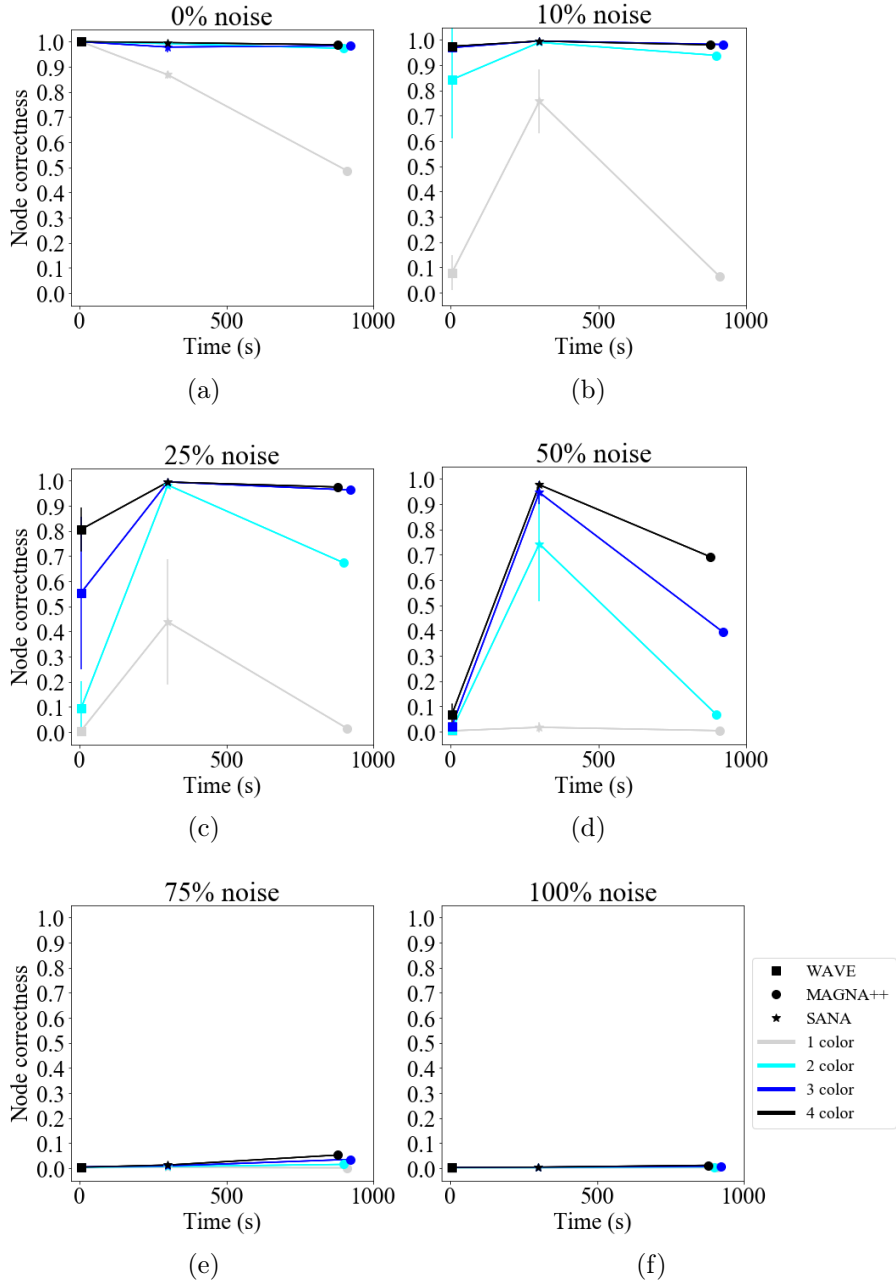


Figure B.9. Detailed results comparing the **running time** and effect of the **number of node colors** for different methods for all tested noise levels on **synthetic, specifically geometric**, networks. The  $x$ -axis the the running time of the method, and the  $y$ -axis is the alignment quality. Here we use different shapes to represent the different methods and different colored lines to represent how many node colors are used. Lines are drawn between methods using the same number of colors.

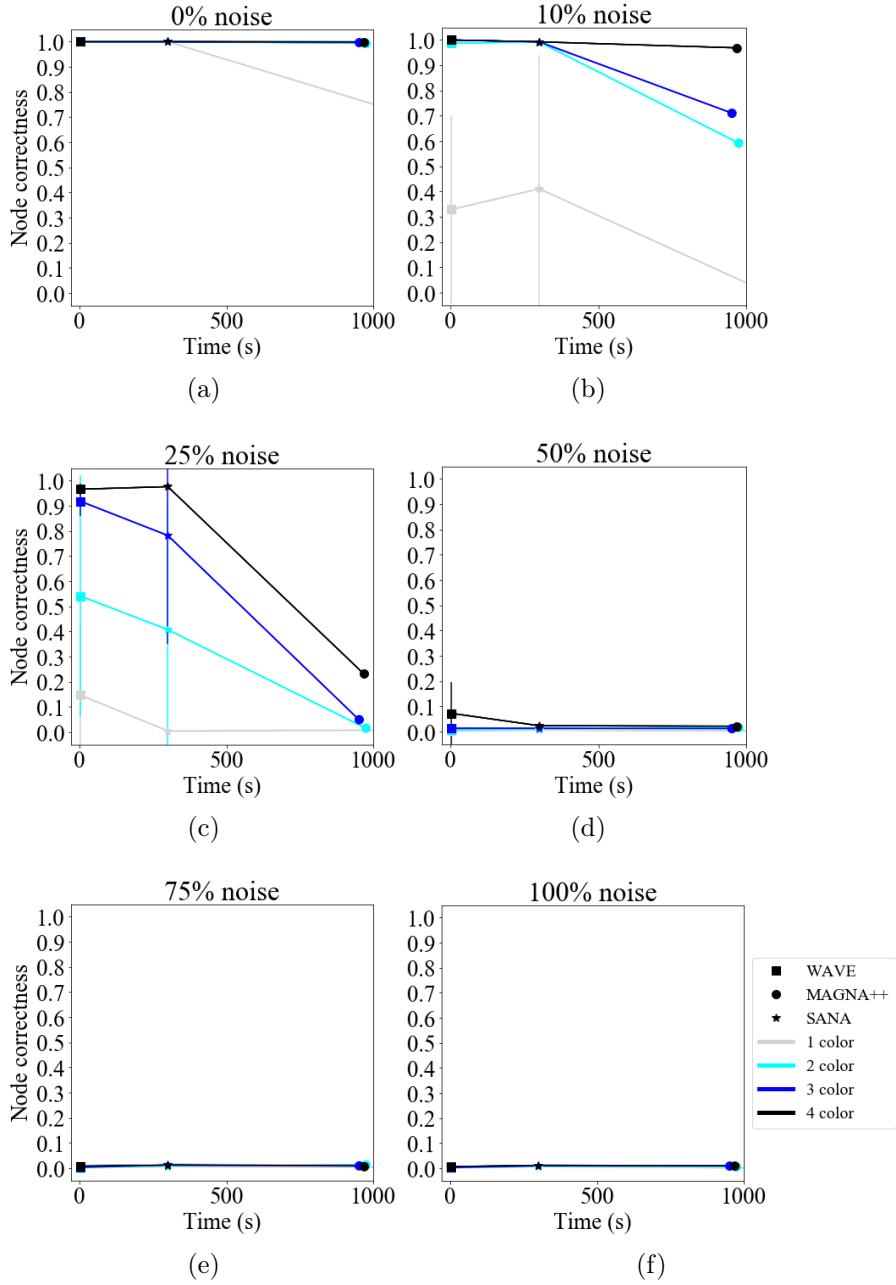


Figure B.10. Detailed results comparing the **running time** and effect of the **number of node colors** for different methods for all tested noise levels on **synthetic, specifically scale-free**, networks. The figure can be interpreted in the same way as Supplementary Figure B.9.

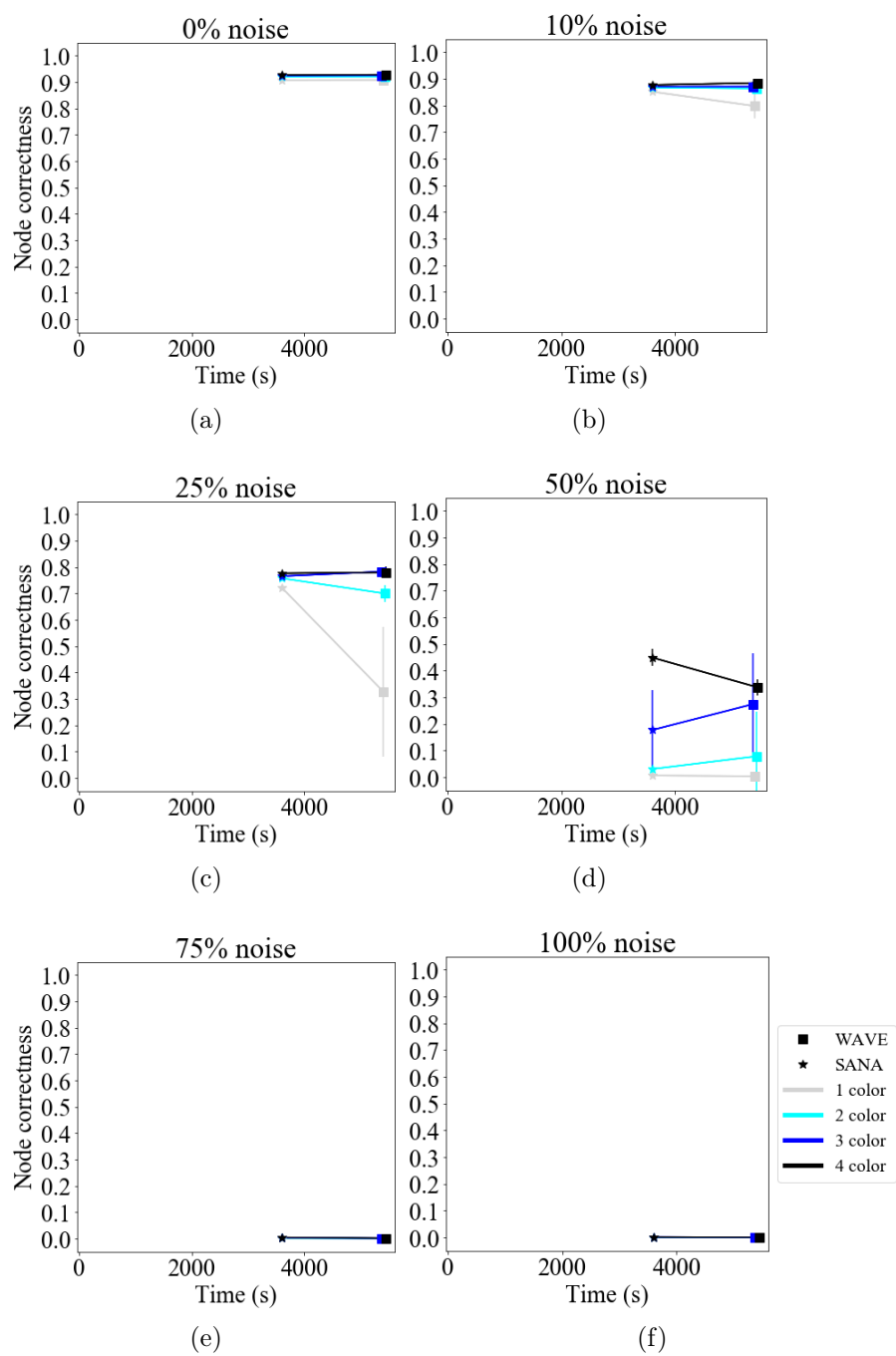


Figure B.11. Detailed results comparing the **running time** and effect of the **number of node colors** for different methods for all tested noise levels on **PPI, specifically APMS-Expr**, networks. The figure can be interpreted in the same way as Supplementary Figure B.9.

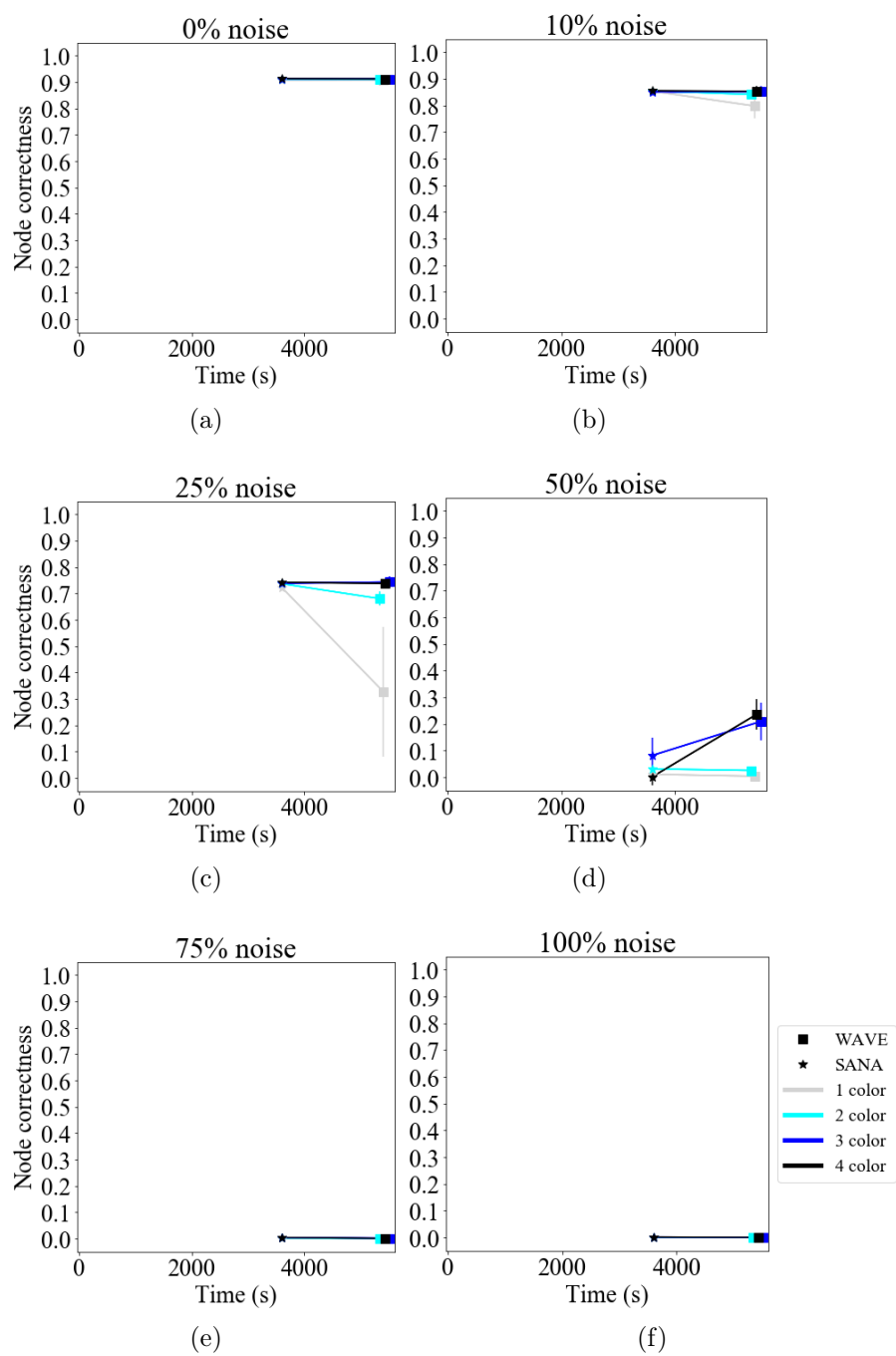


Figure B.12. Detailed results comparing the **running time** and effect of the **number of node colors** for different methods for all tested noise levels on **PPI, specifically APMS-Seq**, networks. The figure can be interpreted in the same way as Supplementary Figure B.9.

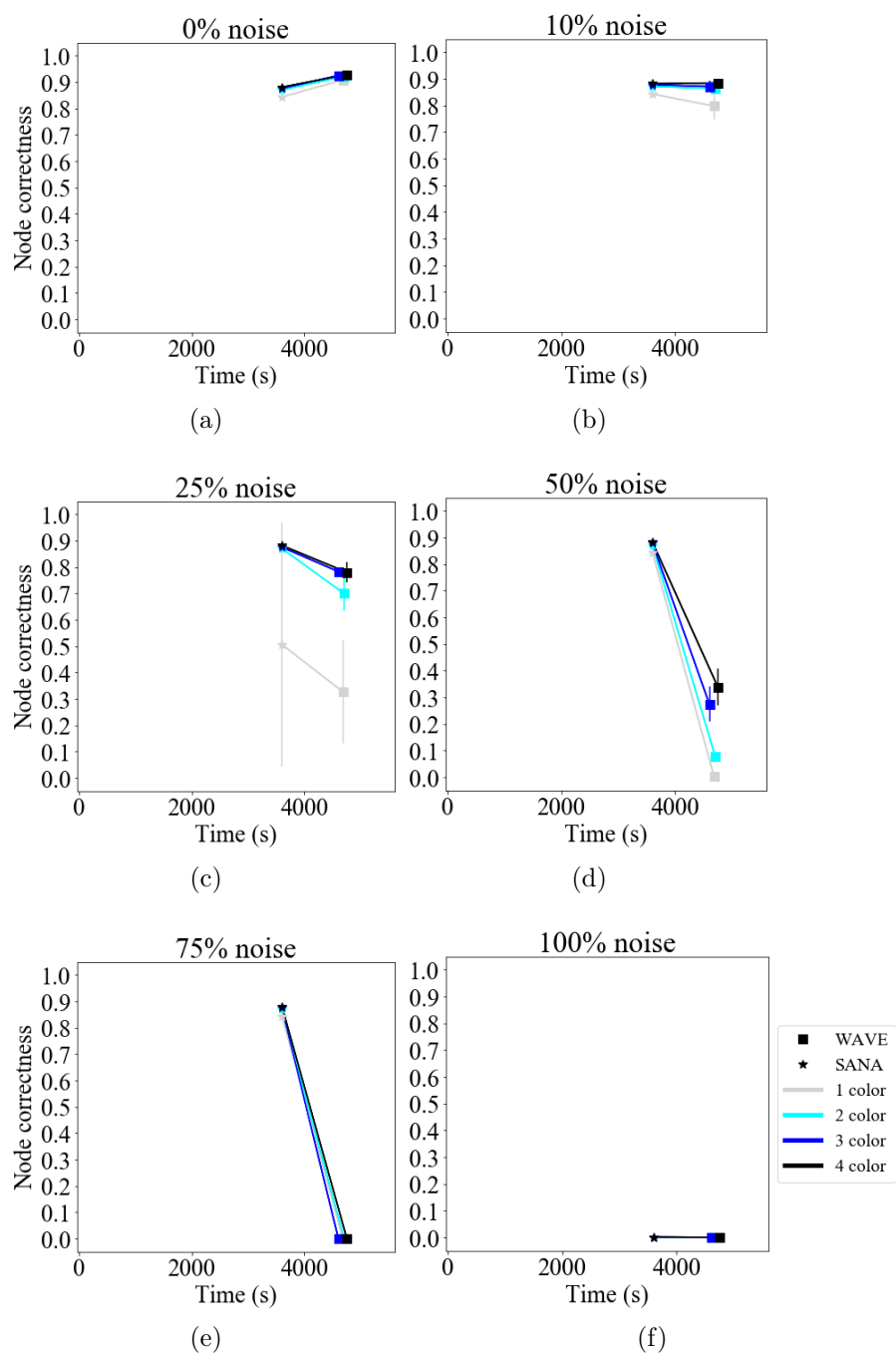


Figure B.13. Detailed results comparing the **running time** and effect of the **number of node colors** for different methods for all tested noise levels on **PPI, specifically Y2H-Expr**, networks. The figure can be interpreted in the same way as Supplementary Figure B.9.



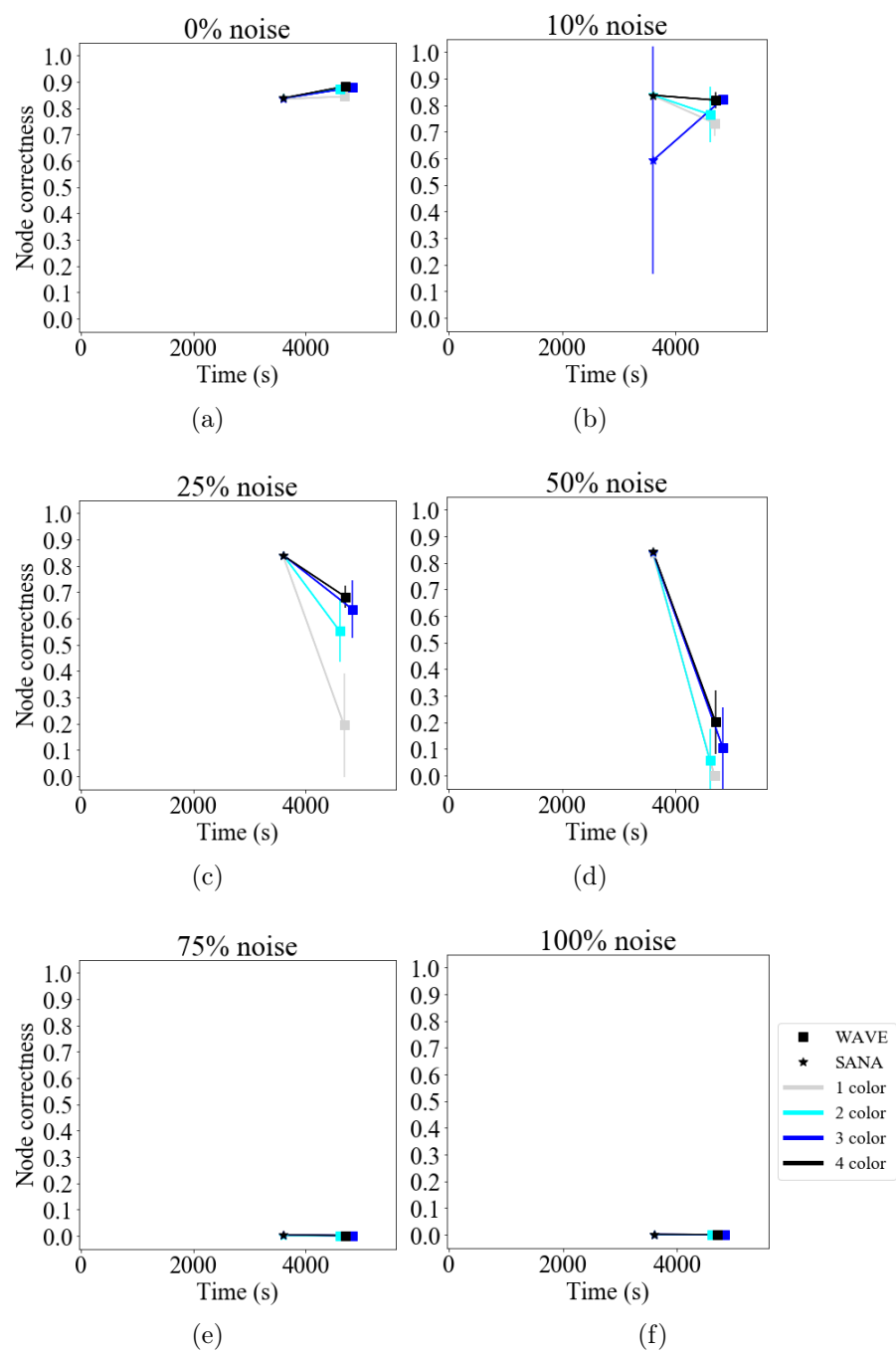


Figure B.14. Detailed results comparing the **running time** and effect of the **number of node colors** for different methods for all tested noise levels on **PPI, specifically Y2H-Seq**, networks. The figure can be interpreted in the same way as Supplementary Figure B.9.

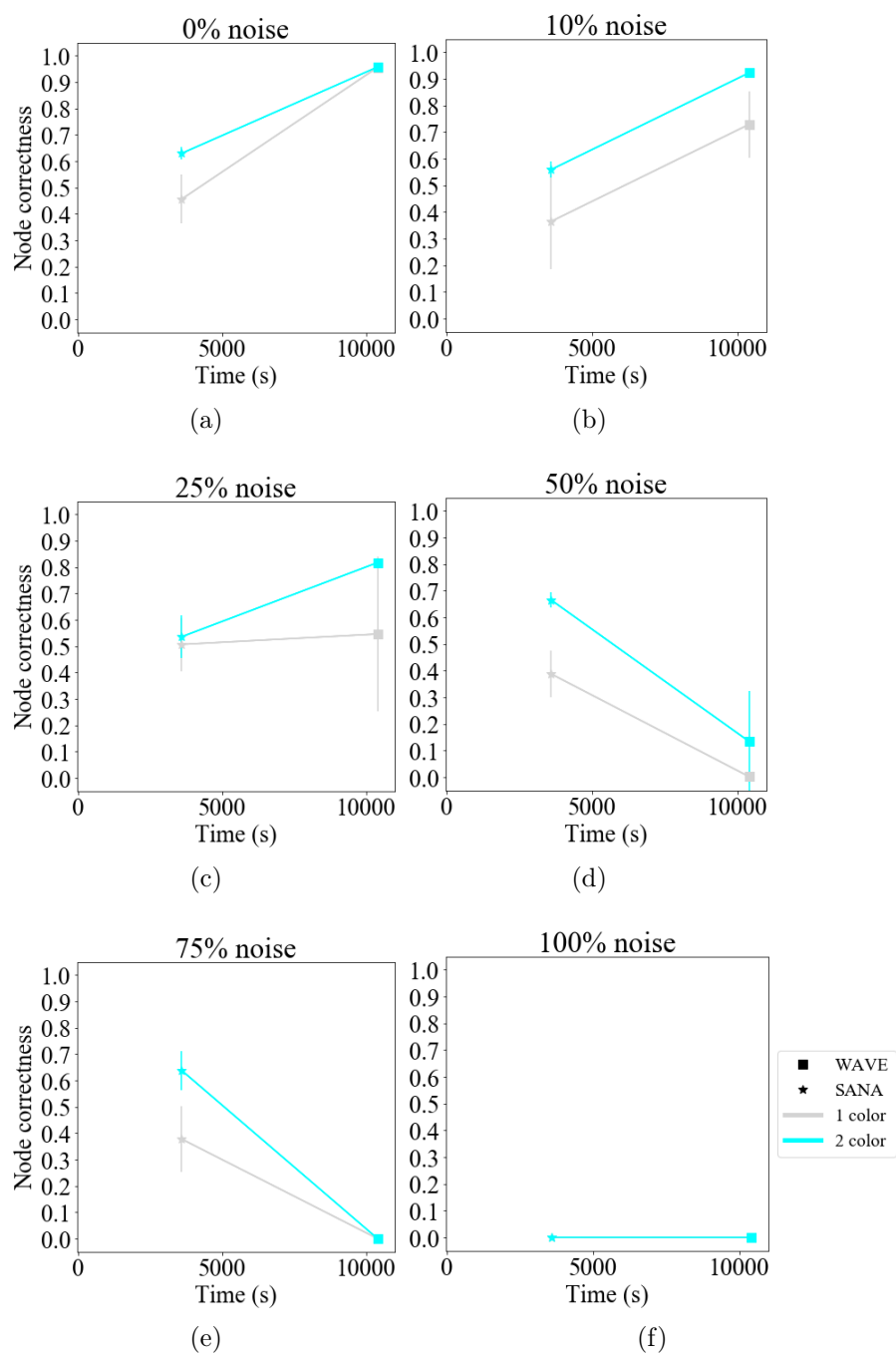


Figure B.15. Detailed results comparing the **running time** and effect of the **number of node colors** for different methods for all tested noise levels on **protein-GO**, specifically **protein-GO-APMS**, networks. The figure can be interpreted in the same way as Supplementary Figure B.9.

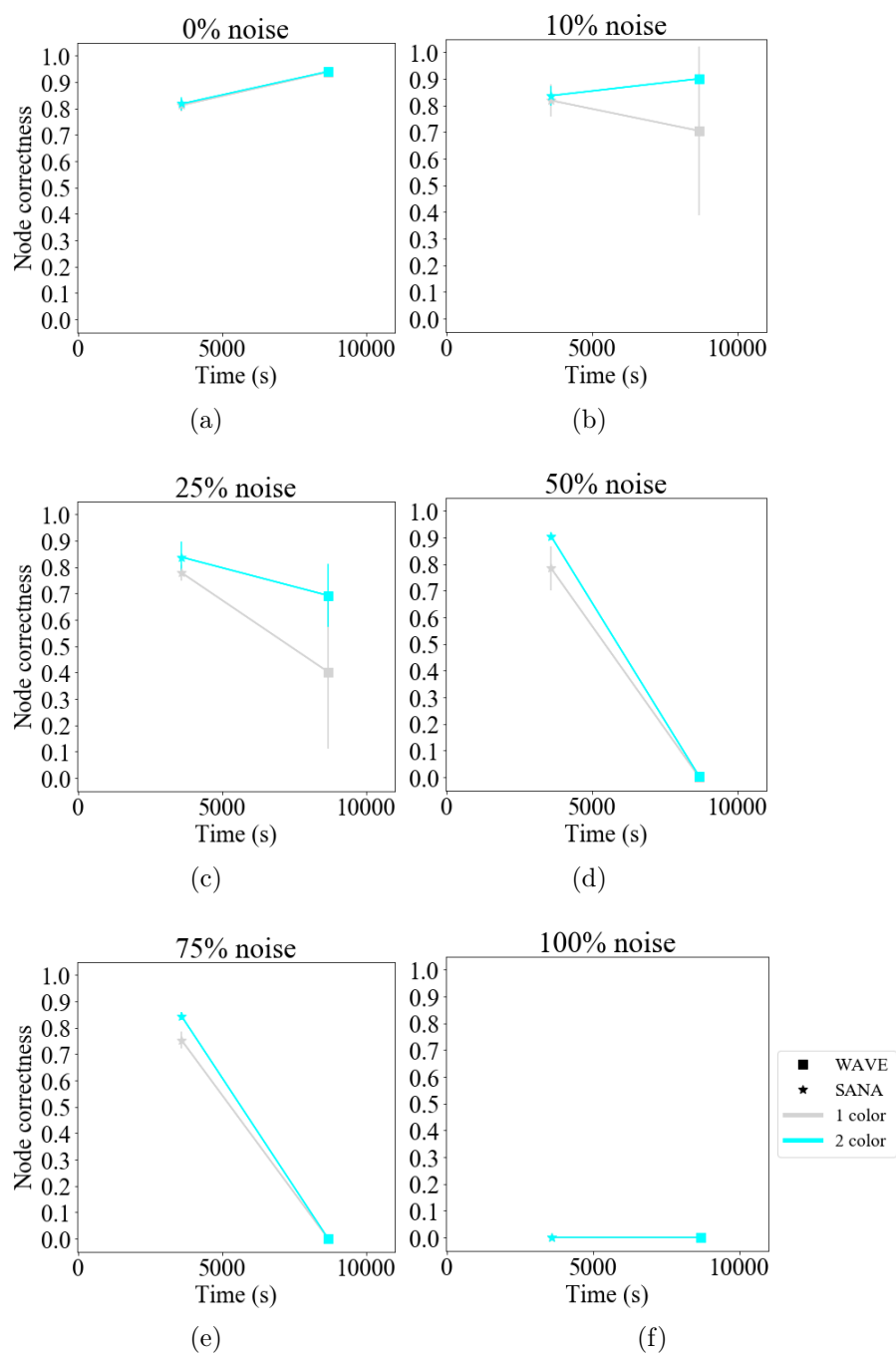


Figure B.16. Detailed results comparing the **running time** and effect of the **number of node colors** for different methods for all tested noise levels on **protein-GO**, specifically **protein-GO-Y2H**, networks. The figure can be interpreted in the same way as Supplementary Figure B.9.

## APPENDIX C

### DATA-DRIVEN NETWORK ALIGNMENT

#### C.1 TARA: Data-driven network alignment

##### C.1.1 Results

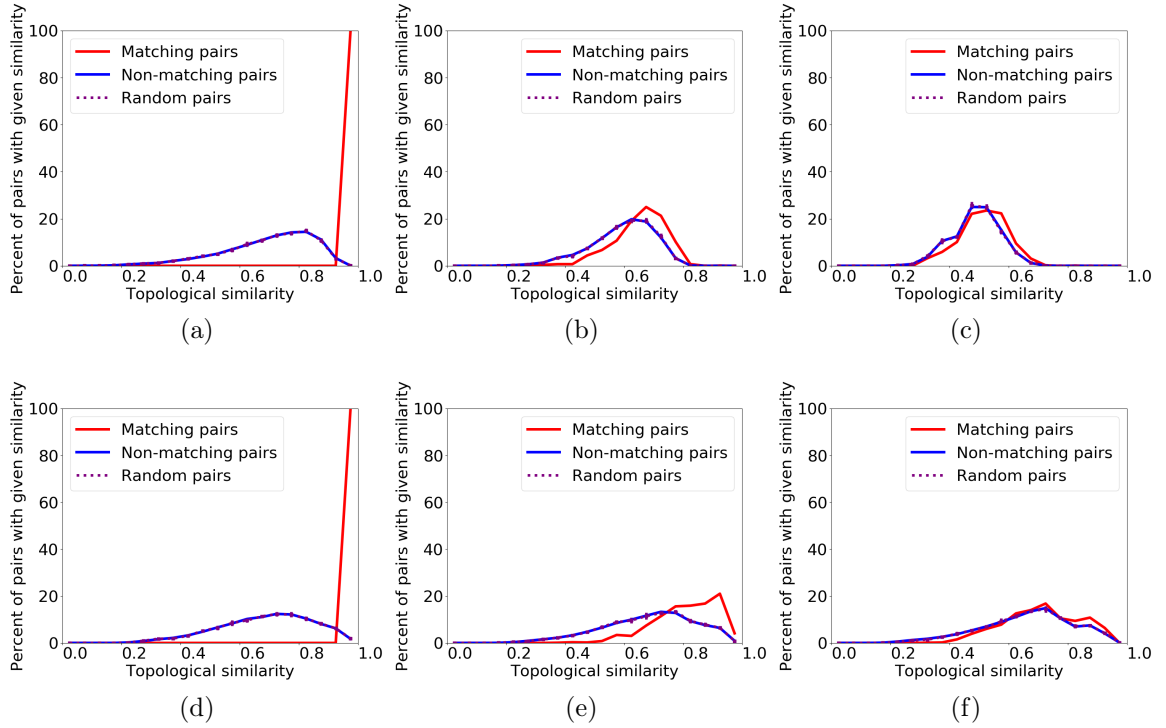


Figure C.1. Distribution of topological similarity (GDV-similarity) between node pairs of a (a,b,c) geometric and (d,e,f) scale-free network and their (a,d) 0%, (b,e) 25% noisy, and (c,f) 50% noisy counterparts. We show three lines representing the distribution of topological similarity for matching, i.e., functionally related, node pairs (blue), for non-matching, i.e., functionally unrelated, node pairs (red), and for 10 random samples of the same size as the set of matching pairs, averaged (purple).

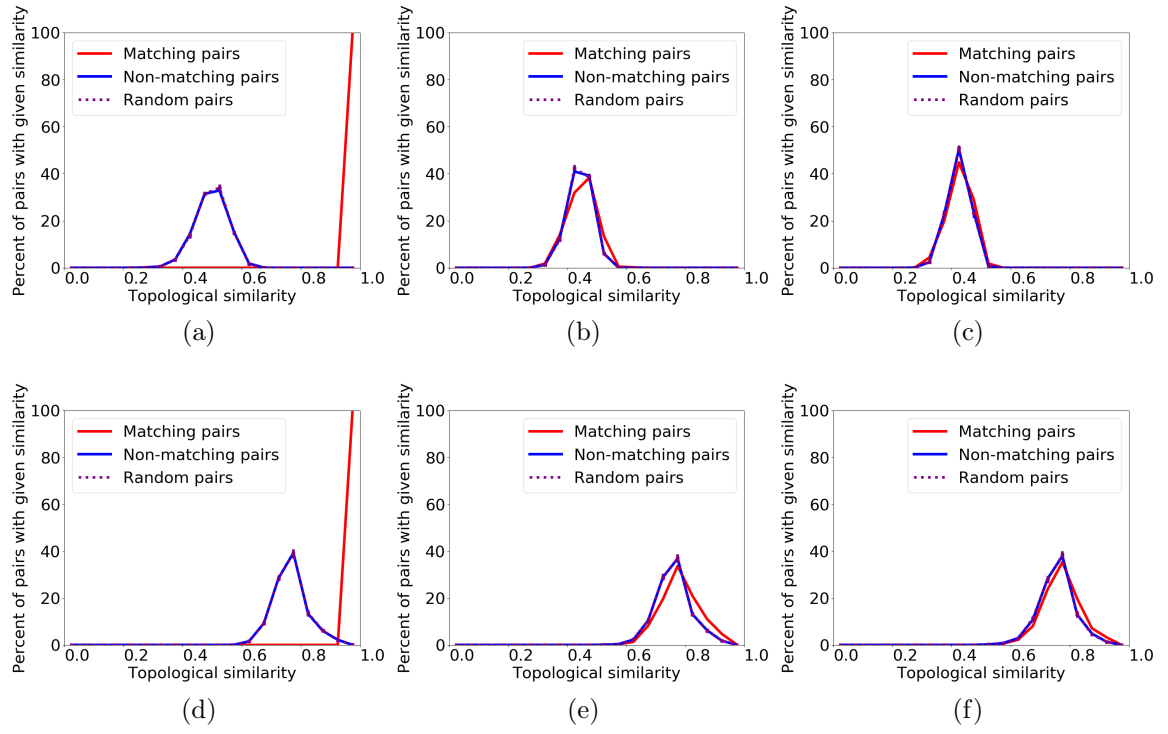


Figure C.2. Distribution of topological similarity (GHOST) between node pairs of a **(a,b,c)** geometric and **(d,e,f)** scale-free network and their **(a,d)** 0%, **(b,e)** 25% noisy, and **(c,f)** 50% noisy counterparts. We show three lines representing the distribution of topological similarity for matching, i.e., functionally related, node pairs (blue), for non-matching, i.e., functionally unrelated, node pairs (red), and for 10 random samples of the same size as the set of matching pairs, averaged (purple).

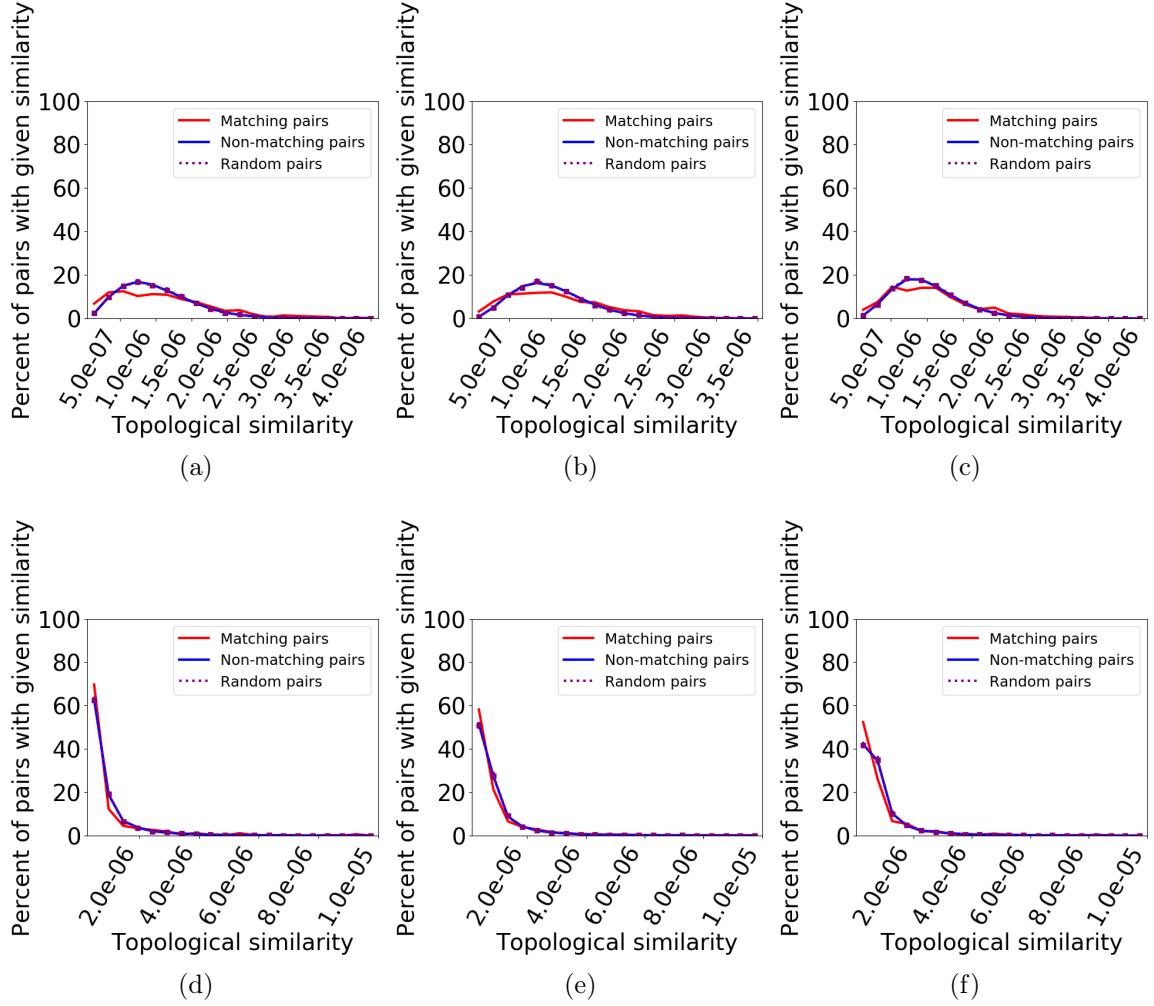


Figure C.3. Distribution of topological similarity (IsoRank) between node pairs of a **(a,b,c)** geometric and **(d,e,f)** scale-free network and their **(a,d)** 0%, **(b,e)** 25% noisy, and **(c,f)** 50% noisy counterparts. We show three lines representing the distribution of topological similarity for matching, i.e., functionally related, node pairs (blue), for non-matching, i.e., functionally unrelated, node pairs (red), and for 10 random samples of the same size as the set of matching pairs, averaged (purple).

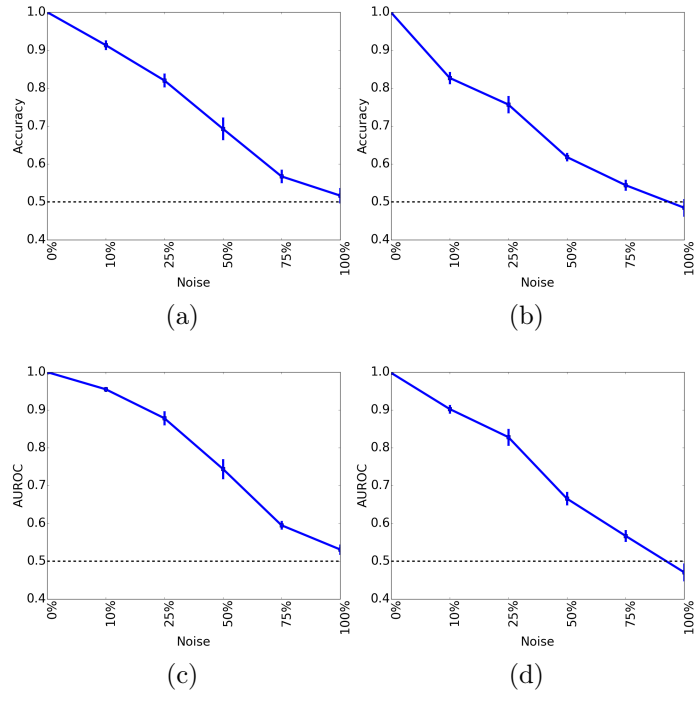


Figure C.4. Average **(a,b)** prediction accuracy and **(c,d)** AUROC of 10-fold cross validation for **(a,c)** geometric and **(b,d)** scale-free networks.

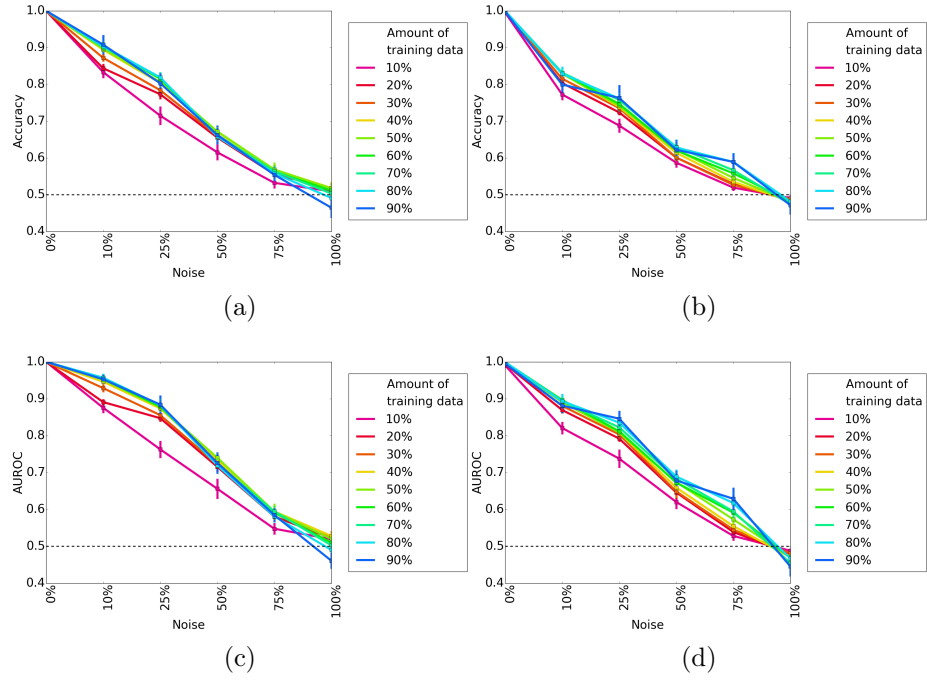


Figure C.5. Average **(a,b)** prediction accuracy and **(c,d)** AUROC of percent training tests for **(a,c)** geometric and **(b,d)** scale-free networks.

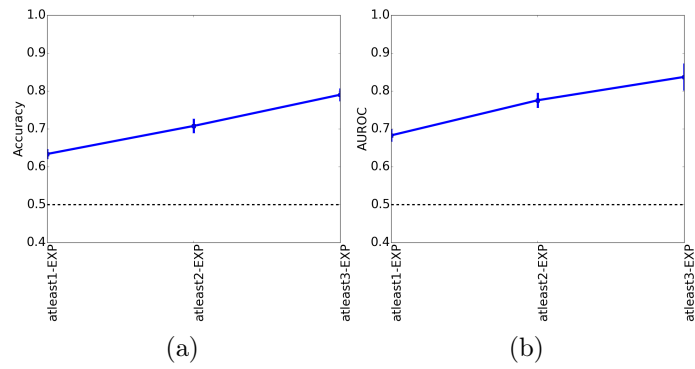


Figure C.6. Average **(a)** prediction accuracy and **(b)** AUROC of 10-fold cross validation for real-world networks.



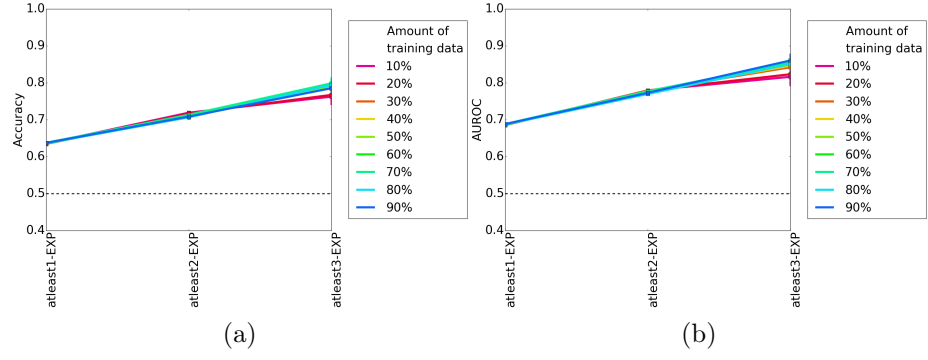


Figure C.7. Average **(a)** prediction accuracy and **(b)** AUROC of percent training tests for real-world networks.

Figure C.8. Comparison of different TARA evaluation tests in the task of protein function prediction, for GO term rarity thresholds **(a, d, g)** ALL, **(b, e)** 50, and **(c, f)** 25 using ground truth datasets **(a, b, c)** atleast1-EXP, **(d, e, f)** atleast2-EXP, and **(g)** atleast3-EXP. Different percent training tests, specifically 10, 50, and 90, are compared within each panel. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method, averaged over the 10 instances we perform for each test, are shown on the top. For example, the alignment for TARA-90 in **(a)** contains 27,155 aligned yeast-human protein pairs, and predicts 91,618 protein-GO term associations. Raw precision, recall, and F-score values are color-coded inside each panel.

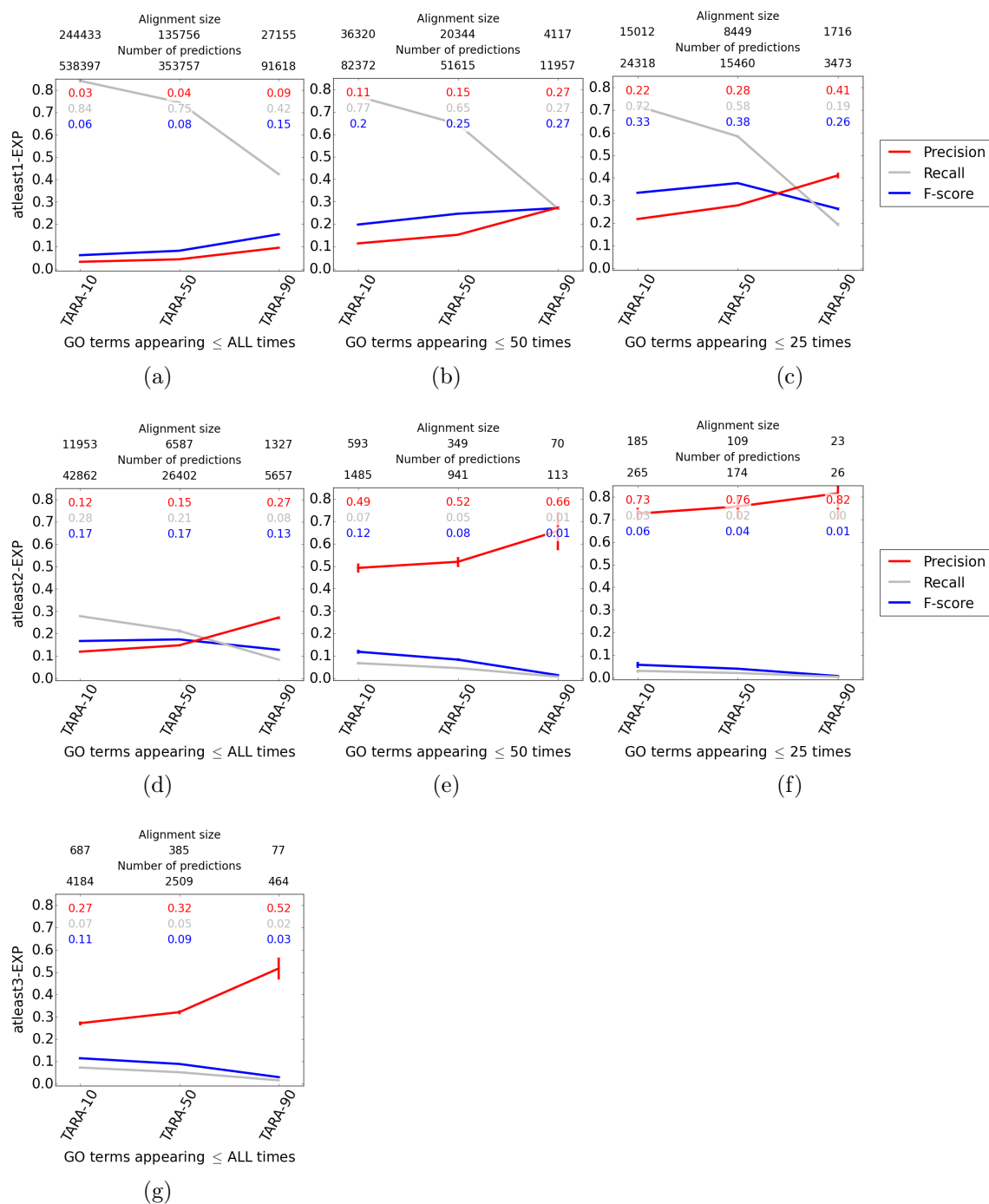
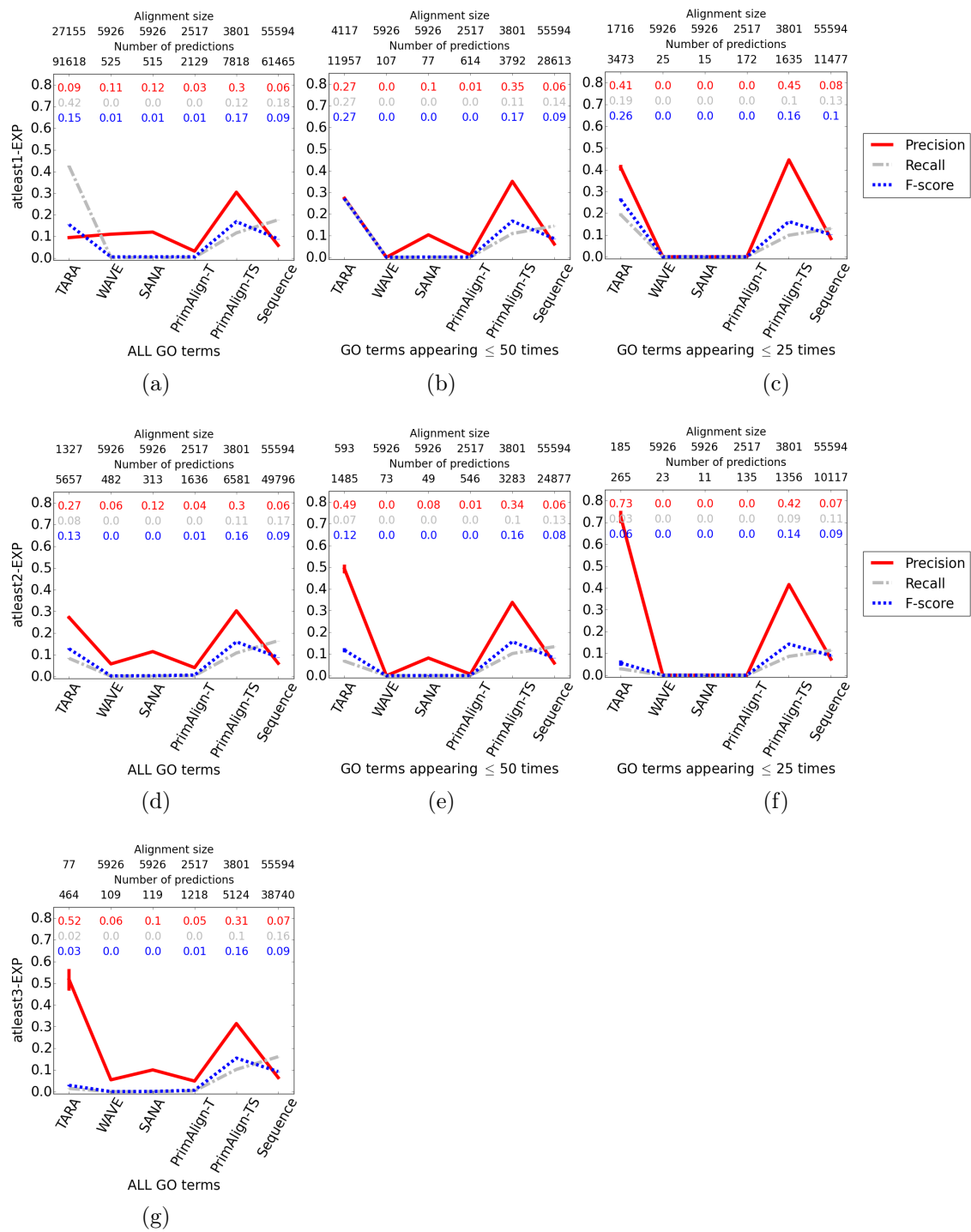


Figure C.9. Comparison of the six considered NA methods for rarity thresholds **(a, d, g)** ALL, **(b, e)** 50, and **(c, f)** 25 using ground truth datasets **(a, b, c)** atleast1-EXP, **(d, e, f)** atleast2-EXP, and **(g)** atleast3-EXP in the task of protein function prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method. For example, the alignment for TARA in **(a)** contains 27,155 aligned yeast-human protein pairs, and predicts 91,618 protein-GO term associations. Raw precision, recall, and F-score values are color-coded inside each panel.



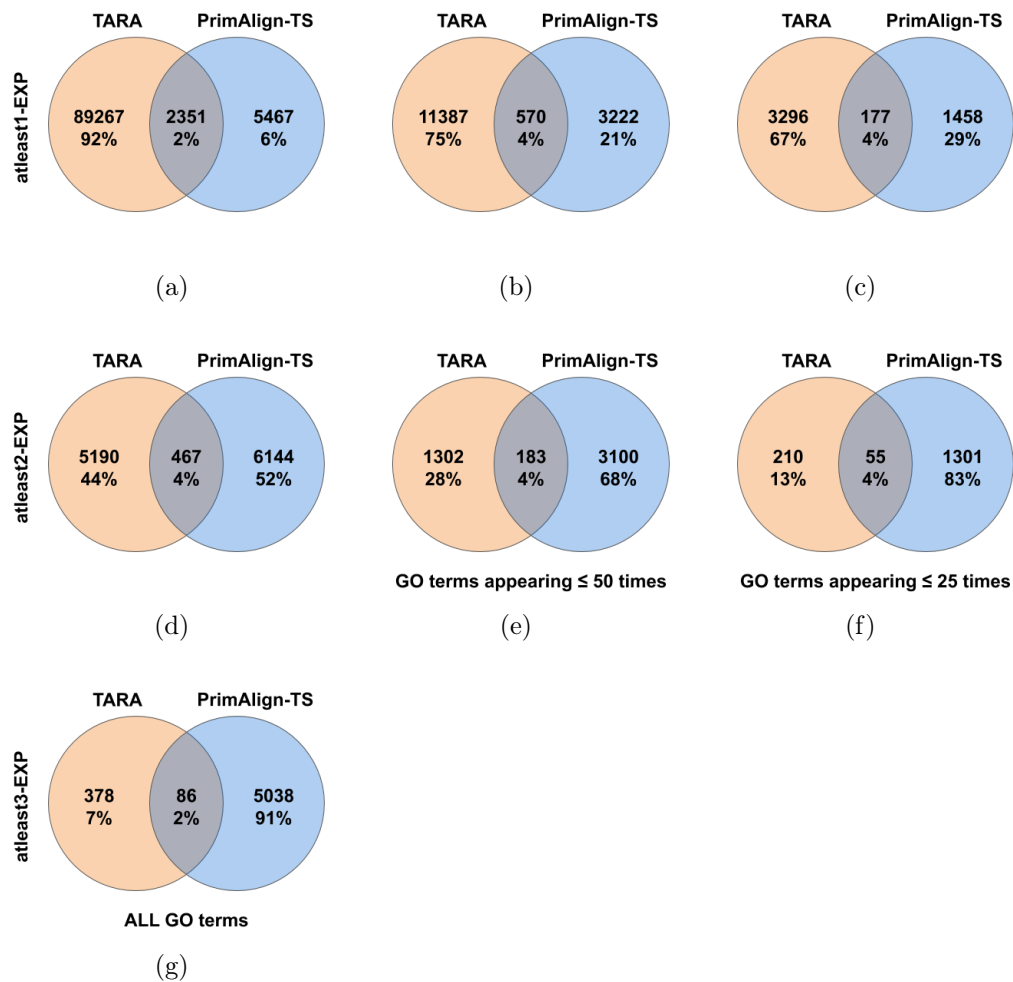
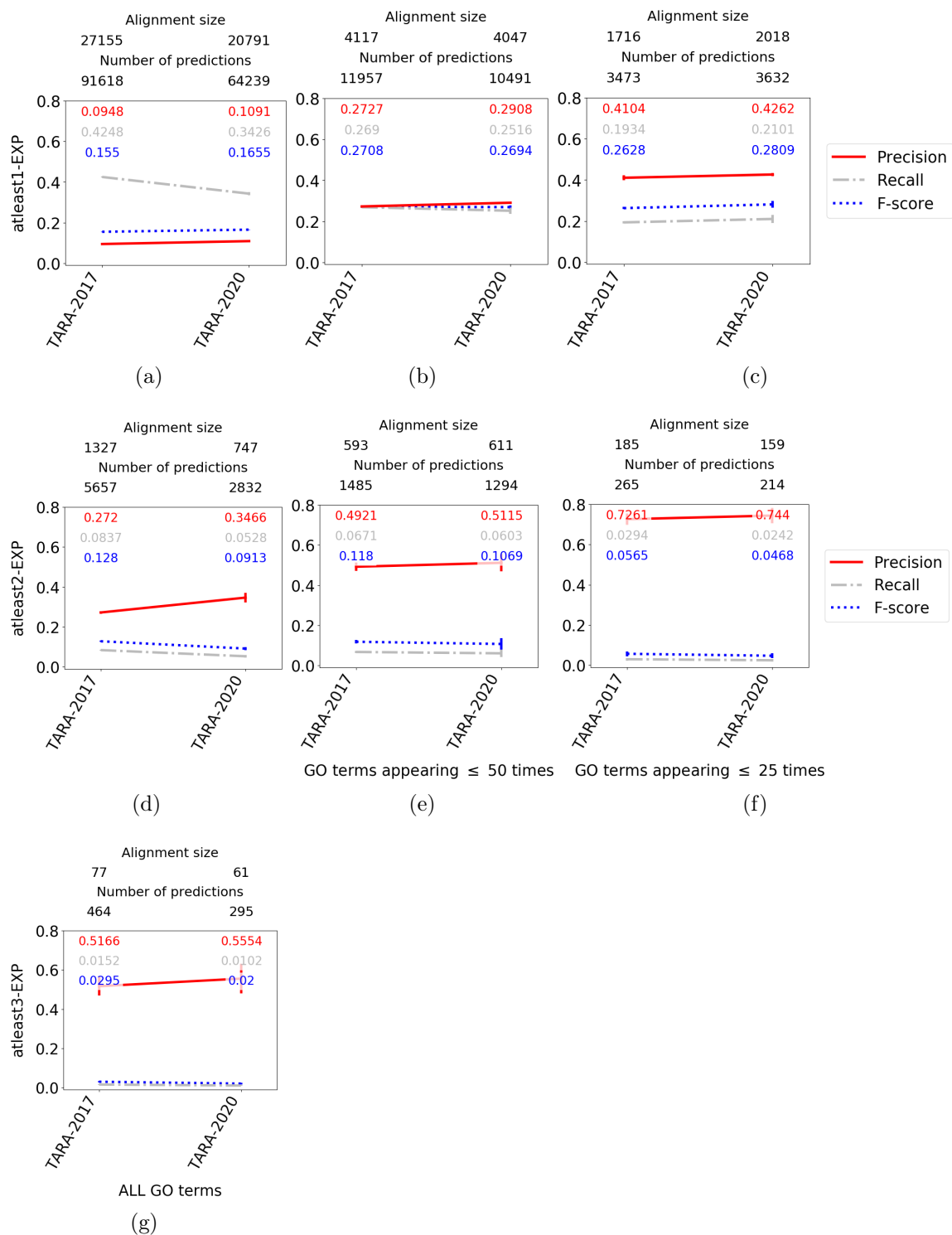


Figure C.10. Overlap of the functional predictions made by TARA and PrimAlign for GO term rarity thresholds (a, d, g) ALL, (b, e) 50, and (c, f) 25 using ground truth datasets (a, b, c) atleast1-EXP, (d, e, f) atleast2-EXP, and (g) atleast3-EXP. Percentages are out of the total number of unique predictions made by both methods combined.

Figure C.11. Comparison of TARA on the 2017 versus 2020 networks for rarity thresholds **(a, d, g)** ALL, **(b, e)** 50, and **(c, f)** 25 using ground truth datasets **(a, b, c)** atleast1-EXP, **(d, e, f)** atleast2-EXP, and **(g)** atleast3-EXP in the task of protein function prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method. For example, the alignment for TARA-2017 in panel **(a)** contains 27,155 aligned yeast-human protein pairs, and predicts 91,618 protein-GO term associations. Raw precision, recall, and F-score values are color-coded inside each panel.





### C.1.2 Supplementary files

File C.1. GuS022022D\_supplementary3.xlsx. Detailed statistics regarding predictions made by TARA and PrimAlign-TS

## C.2 Towards TARA++: Integrating topology and sequence to prediction function

### C.2.1 Methods

#### C.2.1.1 TARA-TS’s feature extraction methodology

**Graphlets.** TARA relies on graphlets, which are small subgraphs (a path, triangle, square, etc., generally up to five nodes). Graphlets are often used to summarize the extended neighborhood of a node into its feature vector, as follows. For a node, for each automorphism orbit (intuitively, node symmetry group) in a graphlet, one can count how many times the node participates in each graphlet at each of its orbits. Then, the graphlet-based feature vector of a node, or the node’s *graphlet degree vector* (GDV), is formed out of the counts of all considered graphlets/orbits that appear in the node’s extended network neighborhood; for details, see [112]. To obtain the feature of a node pair, TARA takes the element-wise absolute difference of the nodes’ GDVs. We found this to be the best (i.e., most accurate) way to simultaneously combine GDVs of both nodes out of all ways that we tested [68]. So TARA-TS can apply the same graphlet counting procedure to the integrated network, obtaining the GDV for each yeast and human node, and taking the absolute difference of two nodes’ GDVs to obtain the feature vector of the yeast-human node *pair*.

**Node2vec.** Node2vec method uses random walks to explore the neighborhood of a node in a network. For a node  $u$ , random walks starting at  $u$  are performed, and the sequence of nodes visited by each random walk is recorded. The number of random walks performed per node is controlled by the parameter “Number of walks per source ( $-\mathbf{r}:$ )”, and the length of a random walk is controlled by the parameter “Length of walk per source ( $-\mathbf{l}:$ )”. This process is repeated for every node in the network. Then, a skip-gram model is applied over all sequences of nodes and the feature vector of each node is obtained; for details, see [66]. The only way to use node2vec, a single-network method, in the multi-network NA task, is to first integrate the two networks

via anchor links, as we do. Otherwise, node2vec fails if applied to the two networks individually [67]. We first apply node2vec to the integrated network with the default parameters to obtain a feature vector for each *node*. Then, as suggested by the node2vec study [66], to get the feature vector of a yeast-human node *pair*, we take the element-wise average of the nodes’ feature vectors.

We use node2vec over other network embedding methods for three reasons. (i) Even more recent methods, when evaluated in their own papers, achieve similar performance as node2vec in many tasks. So, we do not expect them to outperform node2vec in our task. (ii) The node2vec source code is available and well documented, unlike for many other methods. (iii) The goal of this study is not to find the absolute best feature vector for supervised NA, but to test how combining topological and sequence information in supervised NA affects protein functional prediction. If using node2vec already improves upon current NA methods, then using any more sophisticated ways to extract features will only improve further. In our proposed framework, features from any new extraction method can simply be “swapped” in, allowing flexibility for further advancements.

**Metapath2vec.** Metapath2vec requires the user to define “metapaths”, which direct how the random walks move. A metapath example is “human-human-yeast-yeast” (or “human $\times$ 2  $\rightarrow$  yeast $\times$ 2”): start at a human node, move to a randomly chosen neighboring (RCN) human node, move to an RCN yeast node, and move to an RCN yeast node. This metapath is extended such that its length is as close as possible to the `-l:` parameter value. For example, if this value is 12, then this metapath would be repeated  $\lfloor 12/4 \rfloor = 3$  times. Then, given a node  $u$  and the extended metapath, random walks starting at  $u$  are performed such that the nodes visited follow the constraints of the metapath, and the sequence of nodes visited by each random walk is recorded. In the process, node2vec’s `-l:` and `-r:` parameters apply to metapath2vec as well. The procedure is repeated for every node in the network. Then, a skip-gram

model is applied over all sequences of nodes to obtain node features. We use the `metapath2vec++` implementation of `metapath2vec` [41] with the default parameters to obtain each node’s feature vector, and again take the element-wise average of two nodes’ feature vectors to compute the feature vector of a node *pair*. Choosing “optimal” metapaths is non-trivial and often the selection process involves using the same paths as those of previous studies [145, 41]. However, to our knowledge, this study is the first to investigate metapaths on an integrated across-species biological network. Thus, our only option is to do our due diligence and examine reasonable metapaths, to give a fighting chance to `metapath2vec`. We test these metapaths: “human $\times$ n  $\rightarrow$  yeast $\times$ n” and “yeast $\times$ n  $\rightarrow$  human $\times$ n” for  $3 \leq n \leq 10$ , “human $\times$ 25  $\rightarrow$  yeast $\times$ 25” and “yeast $\times$ 25  $\rightarrow$  human $\times$ 25”, “human $\times$ 50  $\rightarrow$  yeast $\times$ 50” and “yeast $\times$ 50  $\rightarrow$  human $\times$ 50”, and the combination of all of the individual metapaths (i.e., we apply the skip-gram model to all node sequences obtained from all considered metapaths). We have verified that the choice of metapath does not affect protein functional prediction accuracy, as illustrated in Supplementary Fig. S1 (see Chapter “4.2.2 – TARA-TS’s classification and alignment generation” and “4.2.2 – Using an alignment for protein functional prediction” for evaluation details). This may be because the metapaths we have chosen are all performing well, or because we have yet to find the best metapaths. Regardless, our goal is to test a variety of reasonable metapaths and choose the best out of them; finding optimal paths is beyond the scope of this study. Because the metapath choice does not impact accuracy in this case, for simplicity, we continue with the combination of all the metapaths.

## C.2.2 Results

### C.2.2.1 TARA-TS versus TARA in the classification context

Here, we comment on the performance of TARA-TS; recall that we use TARA-TS to refer to any of TARA-TS (graphlets, `node2vec`, `metapath2vec`). For a fixed GO

term rarity threshold, as  $k$  in our atleast $k$ -EXP ground truth datasets increases, we expect TARA-TS’s (and TARA’s) accuracy and AUROC to increase, as the condition for proteins to be functionally related becomes more stringent and thus the functional data becomes of higher quality. Also, for a fixed  $k$ , as we decrease the GO term rarity threshold (i.e., consider rarer GO terms), we expect accuracy and AUROC to increase, since rarer GO terms may be meaningful [76], again resulting in higher-quality data. We find the former expectation to hold, for all GO term rarity thresholds (Supplementary Figs. S1-S2). However, for the latter expectation, we find that classification accuracy and AUROC somewhat decrease (Supplementary Figs. S1-S2). This may be because as rarer GO terms are considered, the amount of training data decreases, which is what could be causing performance decreases.

As we increase  $y$ , the amount of training data, we expect accuracy and AUROC to increase, as more data is used during classification. For accuracy, for TARA-TS (graphlets), we observe this for 6/7 ground truth-rarity datasets, although for 4/6 of the datasets, the increase is minimal ( $\sim 1\%$ ). In the remaining case, the accuracy increases until about 60% training data, and then drops. For TARA-TS (node2vec), we observe this for 6/7 ground truth-rarity datasets, although for 4/6, the increase is minimal. In the remaining case, the accuracy increases until about 60% training data, and then drops. For TARA-TS (metapath2vec), we observe this for all 7 ground-truth rarity datasets. For AUROC, for TARA-TS (graphlets), we observe the expected trend for all ground truth-rarity datasets, although for 4/7 of these datasets, the increase is minimal ( $\sim 1\%$ ). For both TARA-TS (node2vec) and TARA-TS (metapath2vec), we observe the expected trend for all ground-truth rarity datasets, although for 3/7 of these datasets, the increase is minimal. These unexpected trends (mostly minor increase of accuracy and AUROC even with large increase of  $y$ ) are promising though, because they mean that TARA-TS does not

have to use a majority of the functional data for training to still obtain good results; even using only 10% of the data seems to suffice.

#### C.2.2.2 Matching the number of predictions made by TARA++

Here, we describe the process for ensuring the different methods make the same number of predictions as TARA++. For a given ground truth-rarity dataset, for a given method, we first rank the predicted protein-GO term associations based on their p-values in the hypergeometric tests (Chapter “4.2.2 – Using an alignment for protein functional prediction”), where a smaller p-value means a better rank. Then, we take the top  $k$  predictions, where  $k$  is equal to the number of predictions made by TARA++. In this way, the best  $k$  predictions of each method are chosen, and every method makes the same number. Finally, we compute the precision and recall of those  $k$  predictions (Fig. 4.14 and Supplementary Fig. C.23). In terms of precision, we find TARA++ is still the best in 4/7 cases. It is inferior to PrimAlign for 1/7 cases, and only slightly worse than TARA in 2/7 cases. In terms of recall, TARA++ is the best in 4/7 cases, tied with TARA in 1/7 cases, and slightly worse than TARA in 2/7 cases. Given these results, we believe that combining TARA and TARA-TS into TARA++ does generally lead to more reliable predictions than other methods, and that TARA++’s high precision is not simply due to it making fewer predictions. In a way, TARA-TS filters out the unreliable predictions from TARA and vice versa.

#### C.2.2.3 Robustness of TARA++ to data noise

Here, we describe how we test TARA++’s robustness to data noise. We introduce three types of noise. (i) We randomly rewire  $x\%$  of the PPI edges in each of the yeast and human networks. This means that for each network, we randomly delete  $x\%$  of the edges and randomly add the same number of edges back such that there are no duplicates. (ii) We randomly rewire  $x\%$  of the across-network anchor links. This

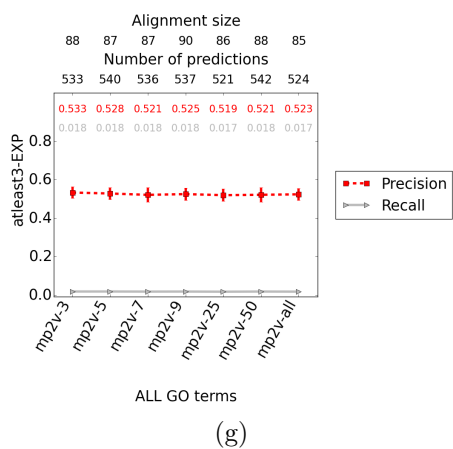
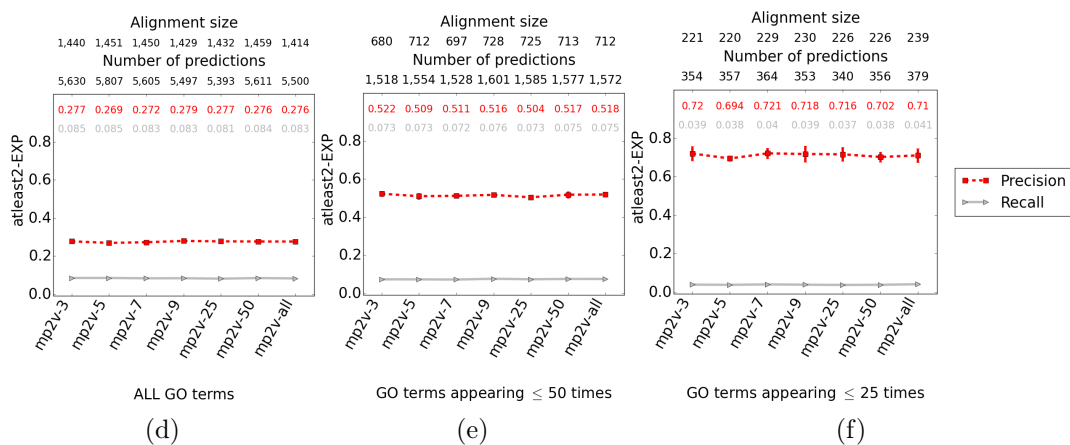
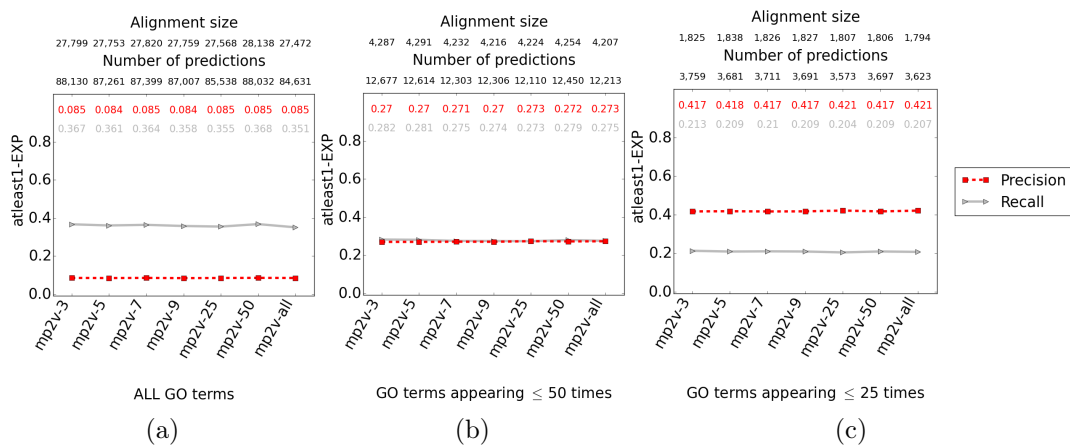
means that we randomly delete  $x\%$  of the yeast-human anchor links and randomly add the same number of yeast-human links back such that there are no duplicates.

(iii) For a given ground truth-rarity dataset, we randomly rewire  $x\%$  of the protein-GO term associations. This means that we randomly delete  $x\%$  of the protein-GO term associations and randomly add the same number of associations back (out of all possible protein-GO term associations for the given ground truth-rarity dataset) such that there are no duplicates. We vary  $x$  from 0 to 100 in increments of 10. We apply each type of noise to its respective data, input the data into TARA and TARA-TS's frameworks, and combine their results into TARA++ as before. We do this for GO term rarity threshold 25 and ground truth dataset atleast2-EXP, as this is where TARA++ has the highest precision.

We find that TARA++'s precision actually increases until 50% noise, after which it then drops and eventually reaches 0, and its recall steadily decreases to 0 (Fig. 4.16). These trends are somewhat expected. As noise increases, the amount of meaningful information in the data decreases, resulting in fewer possible predictions. As such, recall naturally decreases. Because of these fewer predictions though, precision naturally increases. But at a certain point, the data is too noisy, so even the small number of predictions are mostly incorrect, and eventually no predictions can be made. Thus, at that point, precision decreases until it reaches 0.

Figure C.12. Comparison of different metapath choices for rarity thresholds **(a, d, g)** ALL, **(b, e)** 50, and **(c, f)** 25 using ground truth datasets **(a, b, c)** atleast1-EXP, **(d, e, f)** atleast2-EXP, and **(g)** atleast3-EXP in the task of protein functional prediction. “mp2v- $n$ ” refers to the paths “human $\times$ n  $\rightarrow$  yeast $\times$ n” and “yeast $\times$ n  $\rightarrow$  human $\times$ n” (Chapter “4.2.2 – TARA-TS’s feature extraction methodology”). The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method are shown above. For example, the alignment for mp2v-3 in **(a)** contains 27,799 aligned yeast-human protein pairs, and predicts 88,130 protein-GO term associations. Raw precision and recall values are color-coded inside each panel.





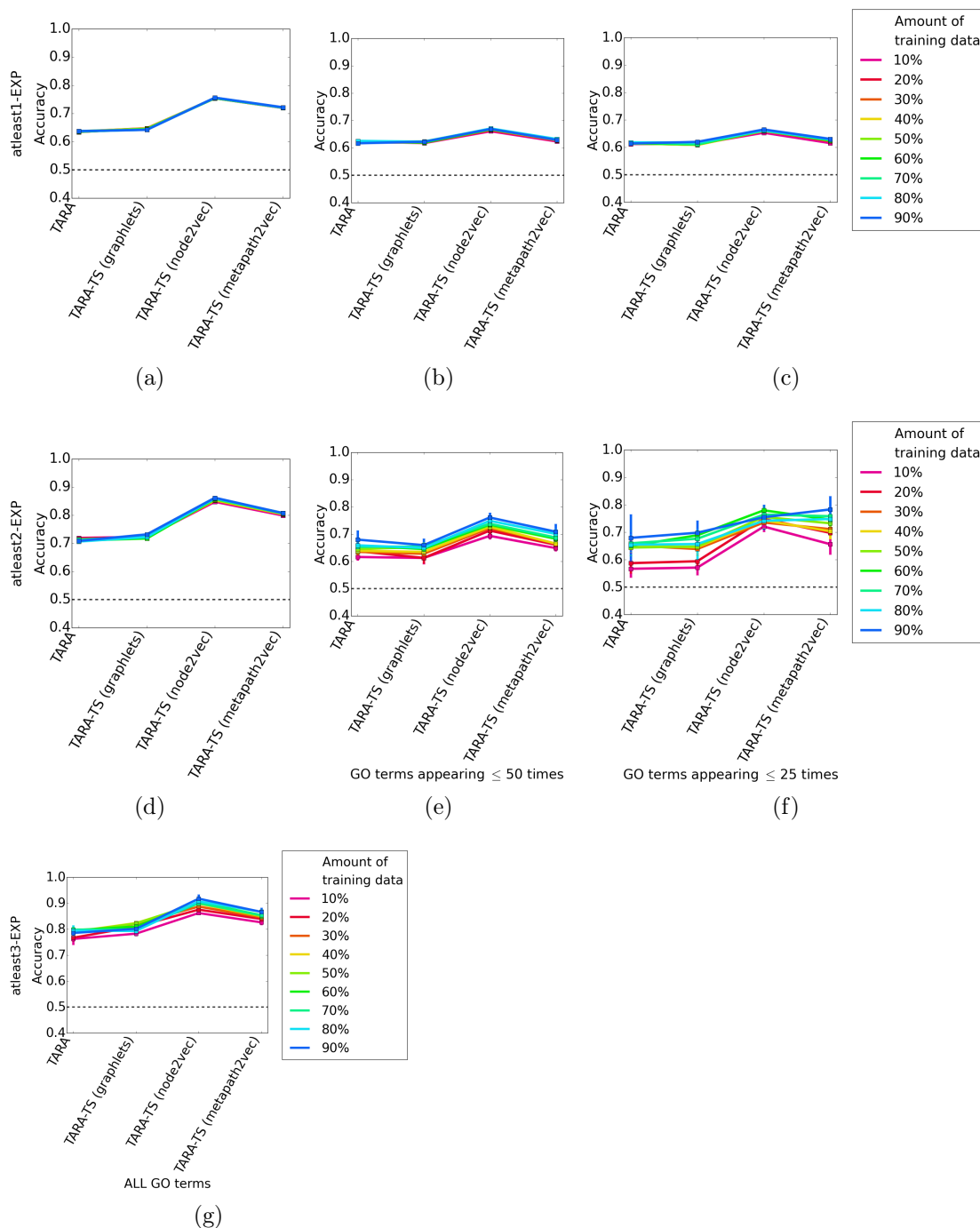


Figure C.13. Average prediction accuracy of percent training tests for rarity thresholds (a, d, g) ALL, (b, e) 50, and (c, f) 25 using ground truth datasets (a, b, c) atleast1-EXP, (d, e, f) atleast2-EXP, and (g) atleast3-EXP. A dotted black line indicates the accuracy expected if the classifier makes random predictions. Qualitatively similar results for AUROC are shown in Supplementary Figs. S2.

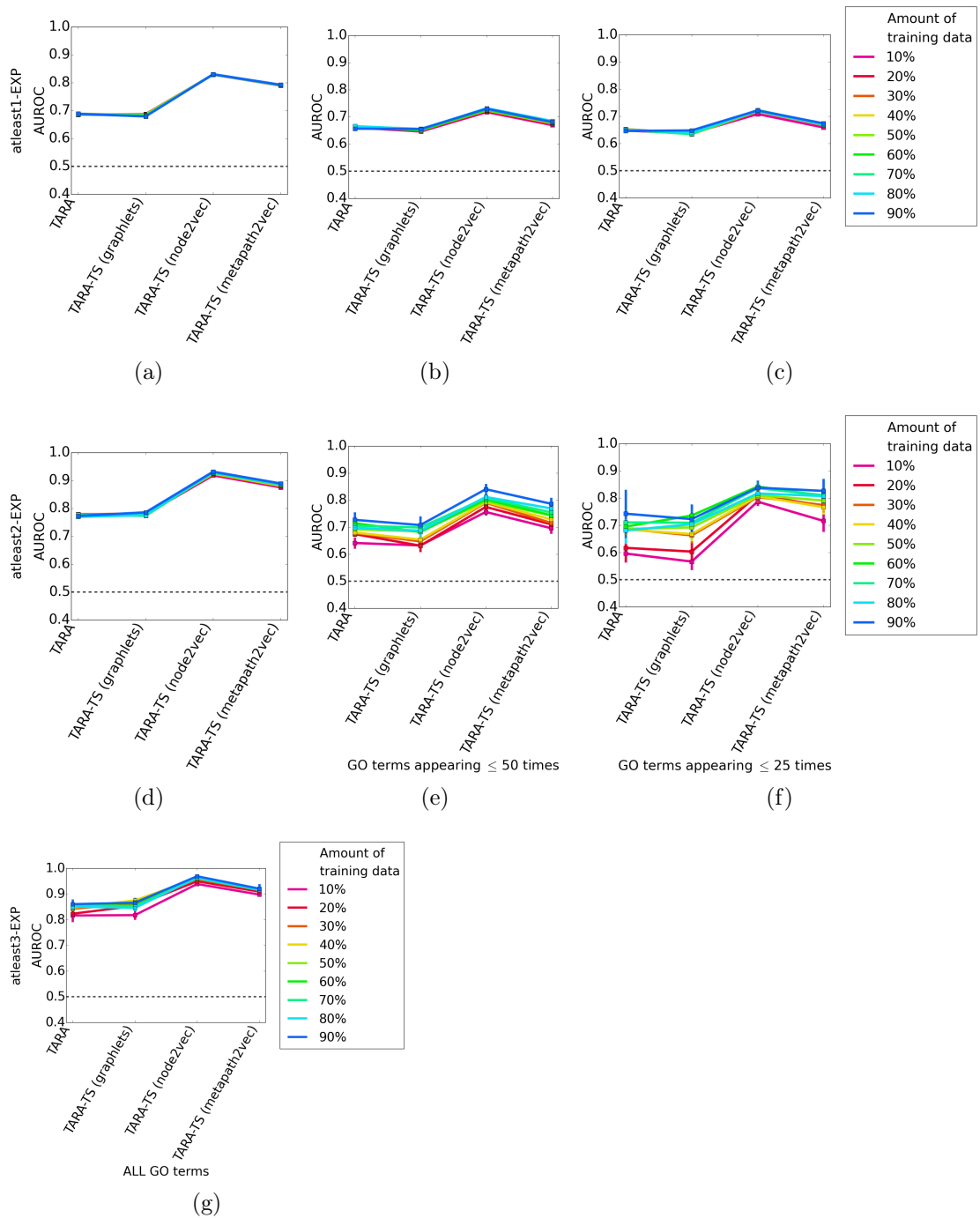


Figure C.14. Average AUROC of percent training tests for rarity thresholds (a, d, g) ALL, (b, e) 50, and (c, f) 25 using ground truth datasets (a, b, c) atleast1-EXP, (d, e, f) atleast2-EXP, and (g) atleast3-EXP. A dotted black line indicates the AUROC expected if the classifier makes random predictions.

Figure C.15. Comparison of TARA and TARA-TS using 10% of the data as training for rarity thresholds **(a, d, g)** ALL, **(b, e)** 50, and **(c, f)** 25 using ground truth datasets **(a, b, c)** atleast1-EXP, **(d, e, f)** atleast2-EXP, and **(g)** atleast3-EXP in the task of protein functional prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method are shown above. For example, the alignment for TARA-10 in **(a)** contains 244,433 aligned yeast-human protein pairs, and predicts 538,397 protein-GO term associations. Raw precision and recall values are color-coded inside each panel.

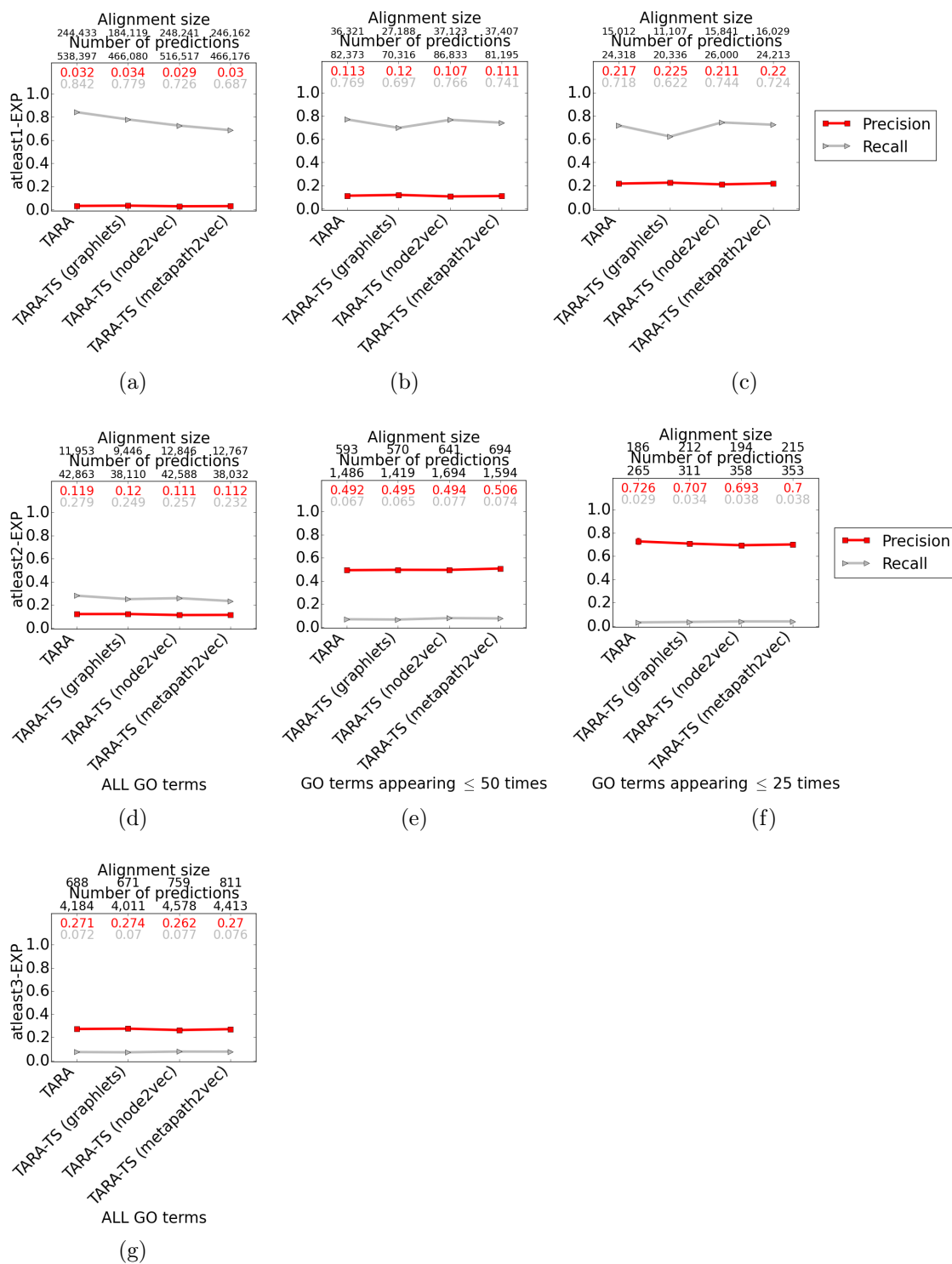


Figure C.16. Comparison of TARA and TARA-TS using 50% of the data as training for rarity thresholds **(a, d, g)** ALL, **(b, e)** 50, and **(c, f)** 25 using ground truth datasets **(a, b, c)** atleast1-EXP, **(d, e, f)** atleast2-EXP, and **(g)** atleast3-EXP in the task of protein functional prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method are shown above. For example, the alignment for TARA-10 in **(a)** contains 244,433 aligned yeast-human protein pairs, and predicts 538,397 protein-GO term associations. Raw precision and recall values are color-coded inside each panel.

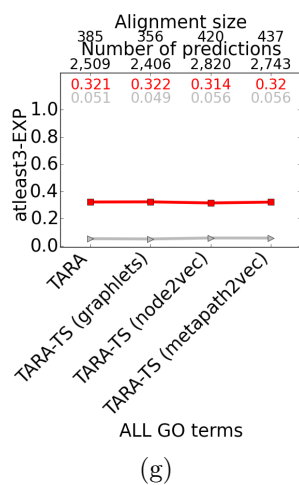
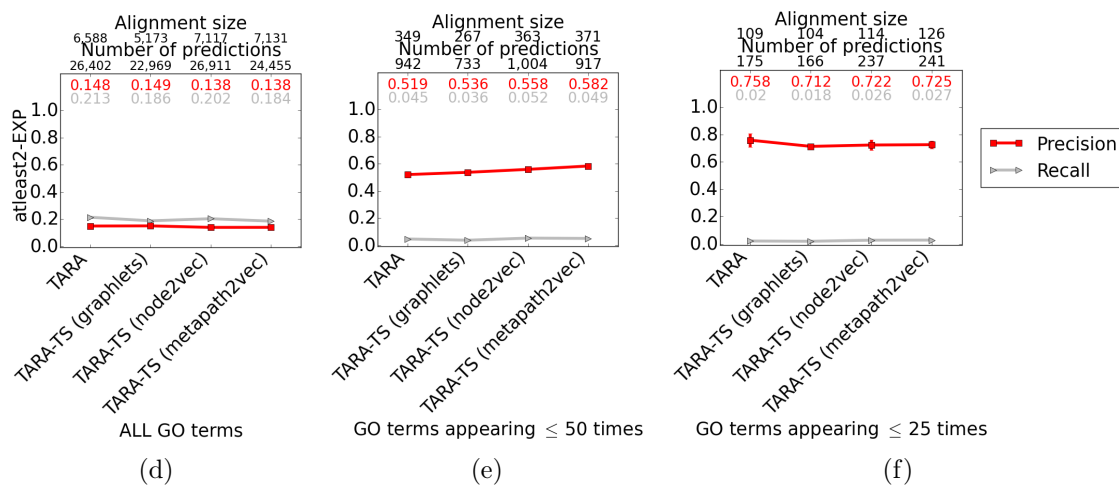
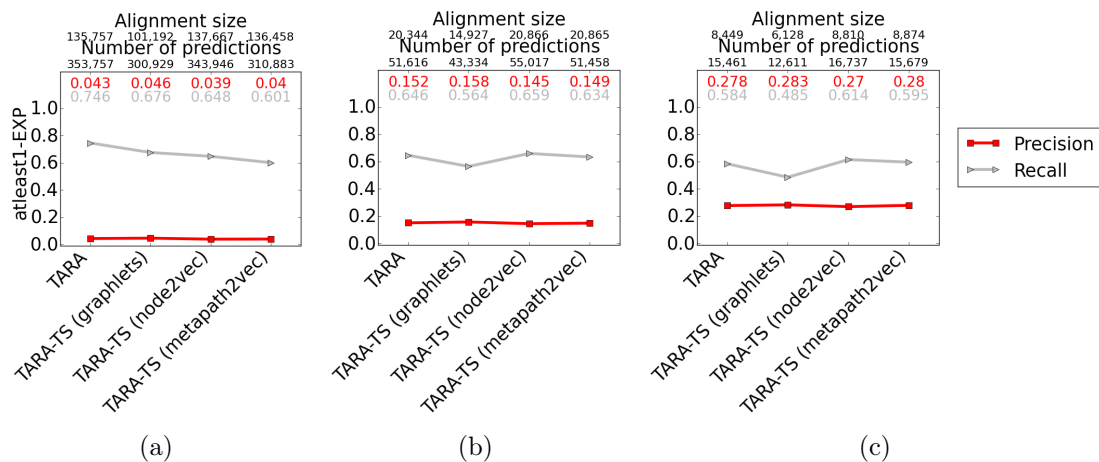
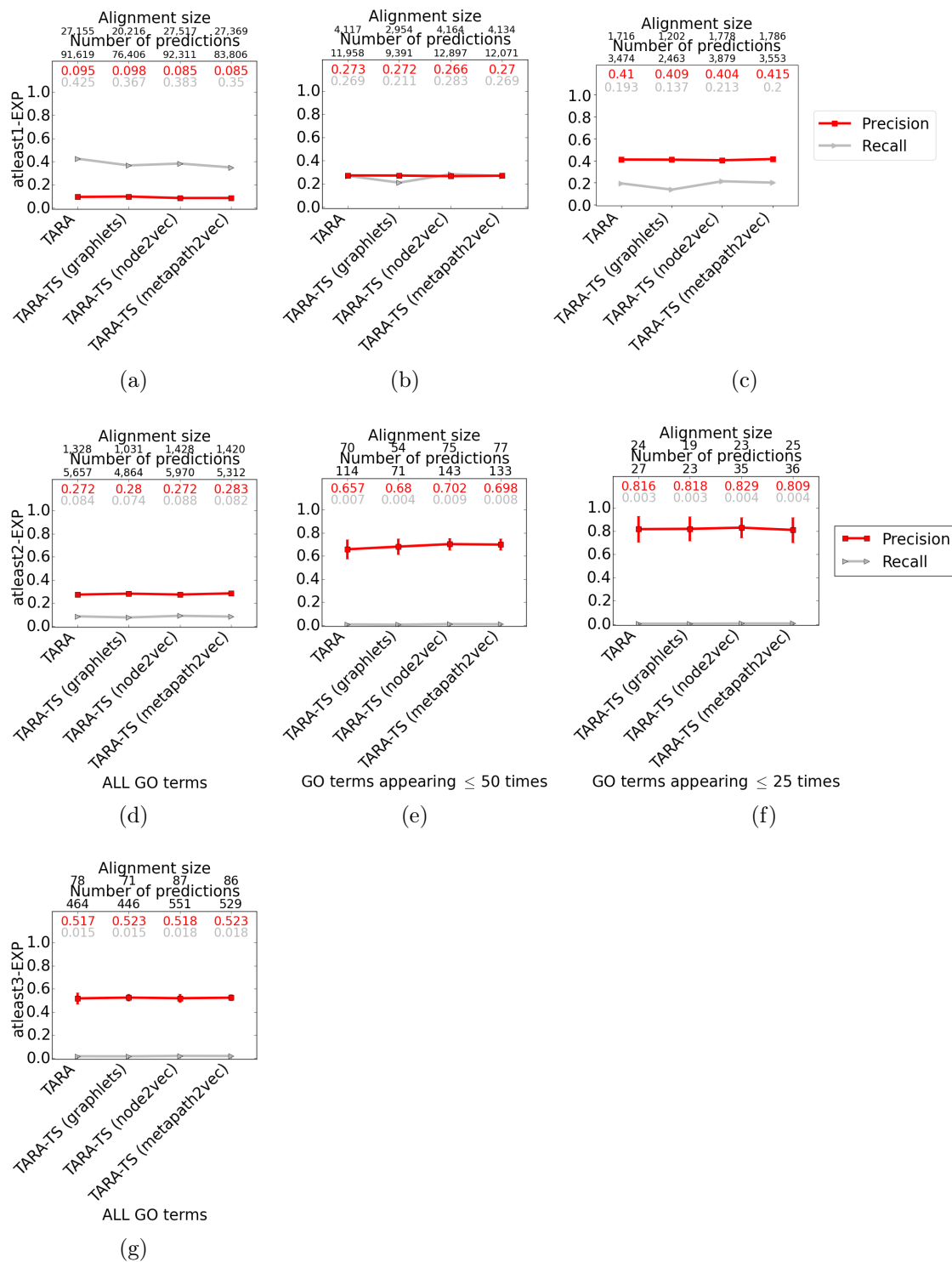


Figure C.17. Comparison of TARA and TARA-TS using 90% of the data as training for rarity thresholds **(a, d, g)** ALL, **(b, e)** 50, and **(c, f)** 25 using ground truth datasets **(a, b, c)** atleast1-EXP, **(d, e, f)** atleast2-EXP, and **(g)** atleast3-EXP in the task of protein functional prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method are shown above. For example, the alignment for TARA-10 in **(a)** contains 244,433 aligned yeast-human protein pairs, and predicts 538,397 protein-GO term associations. Raw precision and recall values are color-coded inside each panel.





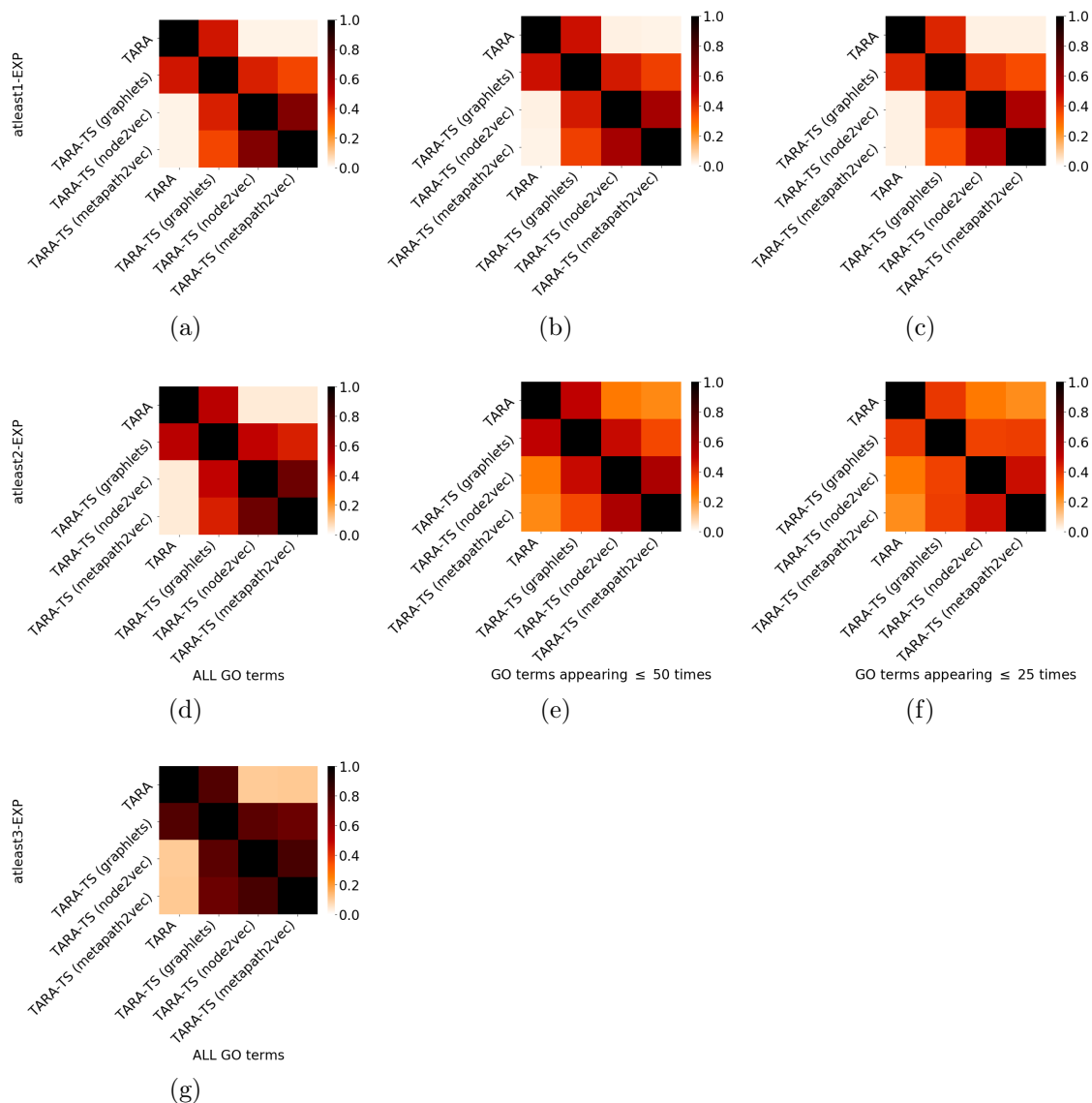


Figure C.18. Pairwise overlap, measured by Jaccard index, of the alignments made by TARA and TARA-TS for rarity thresholds (a, d, g) ALL, (b, e) 50, and (c, f) 25 using ground truth datasets (a, b, c) atleast1-EXP, (d, e, f) atleast2-EXP, and (g) atleast3-EXP, using percent training amounts described in Chapter “4.2.3 – TARA-TS versus TARA in the task of protein functional prediction: toward TARA++”.

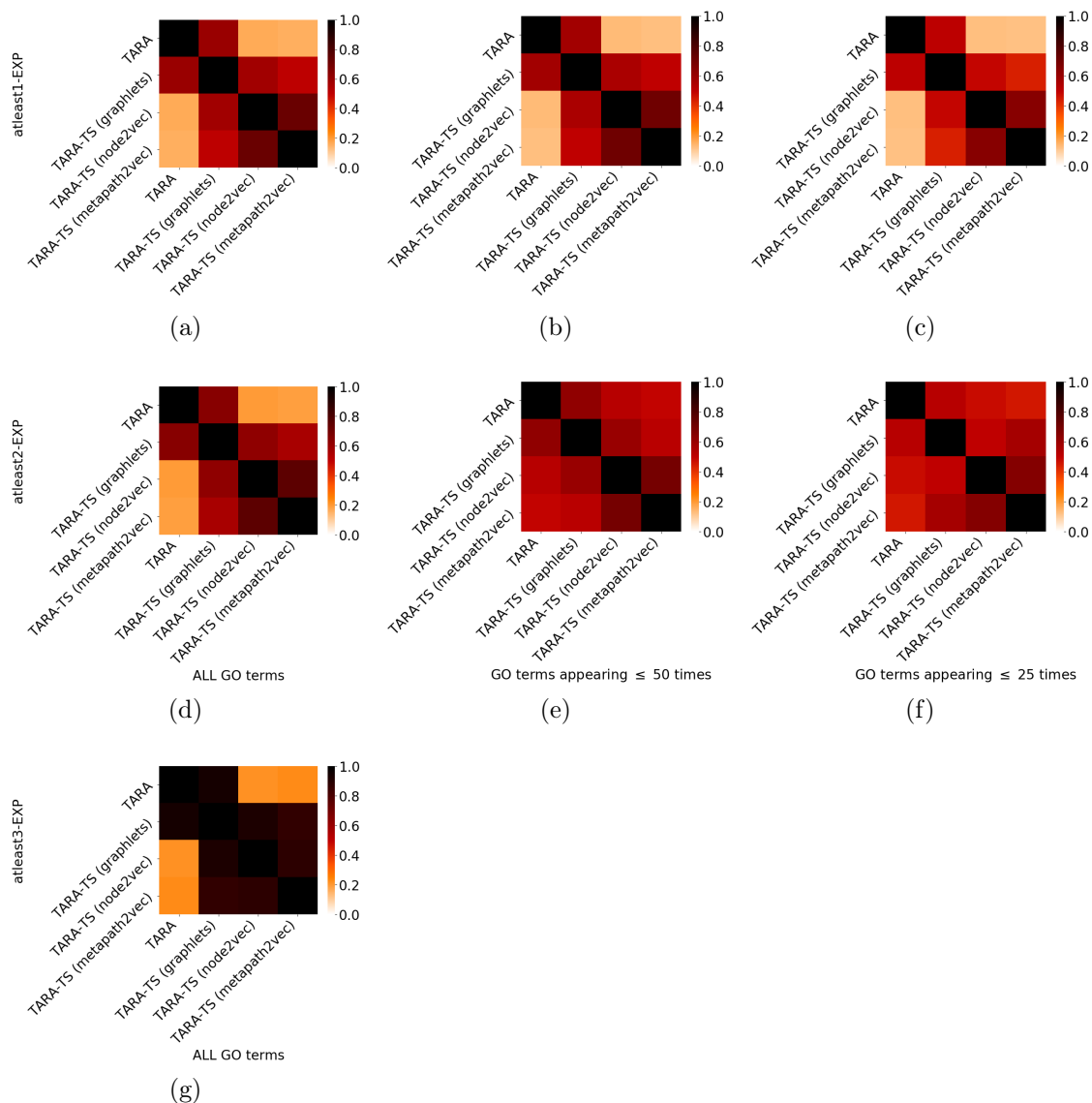


Figure C.19. Pairwise overlap, measure by Jaccard index, of the predictions made by TARA and TARA-TS for rarity thresholds (a, d, g) ALL, (b, e) 50, and (c, f) 25 using ground truth datasets (a, b, c) atleast1-EXP, (d, e, f) atleast2-EXP, and (g) atleast3-EXP, using percent training amounts described in Chapter “4.2.3 – TARA-TS versus TARA in the task of protein functional prediction: toward TARA++”.

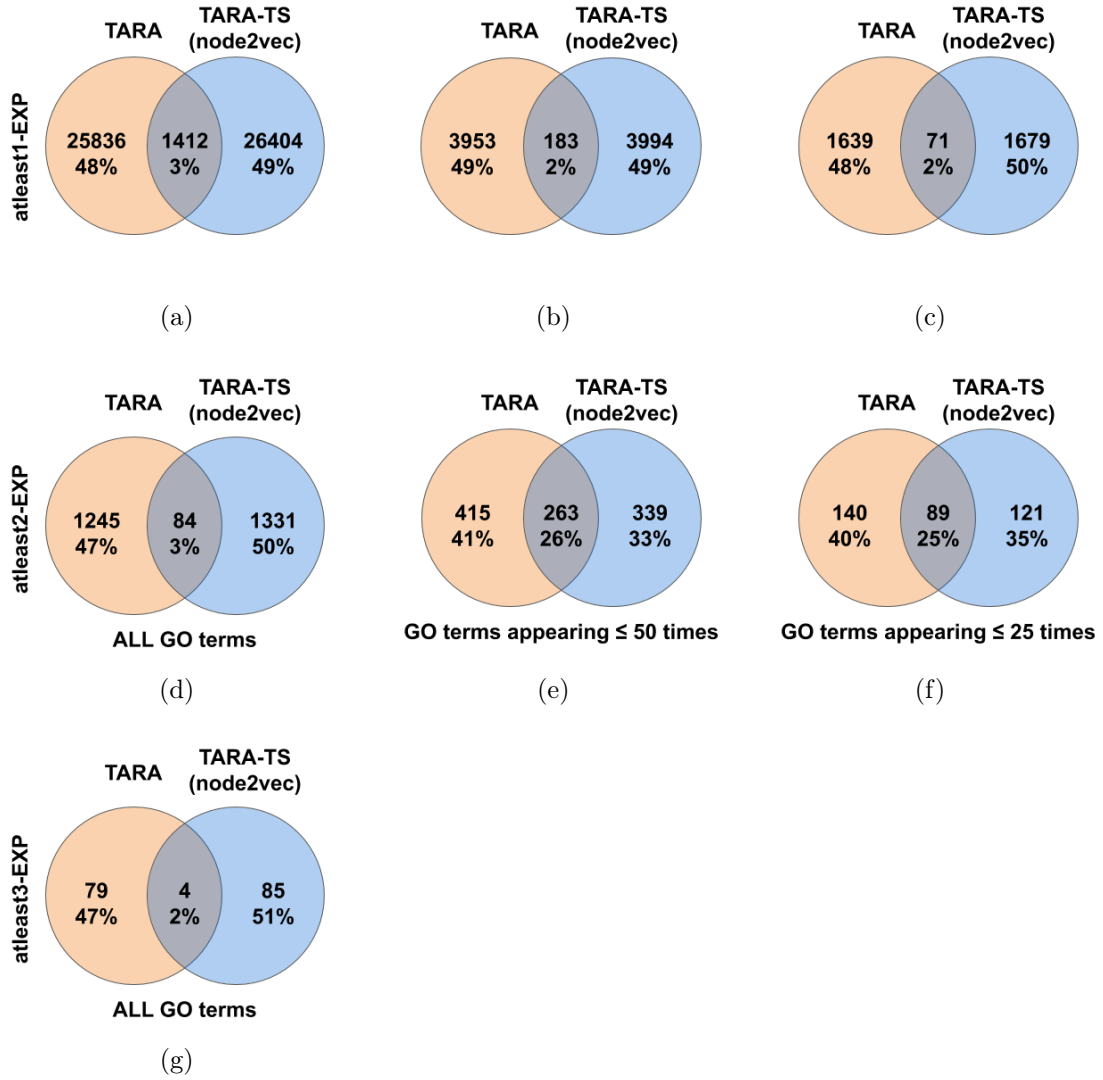


Figure C.20. Overlap of the alignments made by TARA and TARA-TS for rarity thresholds (a, d, g) ALL, (b, e) 50, and (c, f) 25 using ground truth datasets (a, b, c) atleast1-EXP, (d, e, f) atleast2-EXP, and (g) atleast3-EXP. Percentages are out of the total number of unique aligned node pairs made by both methods combined. The overlaps are for one of the 10 balanced datasets; so, the alignment size of a method may differ from those in Supplementary Figs. S3-S5, where the statistics are averaged over all balanced datasets.

Figure C.21. Overlap of the predictions made by TARA and TARA-TS for rarity thresholds **(a, d, g)** ALL, **(b, e)** 50, and **(c, f)** 25 using ground truth datasets **(a, b, c)** atleast1-EXP, **(d, e, f)** atleast2-EXP, and **(g)** atleast3-EXP. Percentages are out of the total number of unique predictions made by both methods combined. Precision and recall are shown for each of the three prediction sets captured by the Venn diagram; TARA++’s predictions are those in the overlap. The overlaps are for one of the 10 balanced datasets; so, the prediction number of a method may differ from those in Supplementary Figs. S3-S5, where the statistics are averaged over all balanced datasets.



TABLE C.1

RUNNING TIMES (IN SECONDS) OF TARA-TS, TARA, PRIMALIGN,  
AND SEQUENCE, WHEN CONSIDERING ALL GO TERMS

	atleast1-EXP	atleast2-EXP	atleast3-EXP
TARA-TS	3811	480	444
TARA	8090	4676	4634
PrimAlign	16	16	16
Sequence	N/A	N/A	N/A

TARA++’s running time is a function of TARA-TS’s and TARA’s (see Chapter “4.2.3 – TARA++ versus existing NA methods in the task of protein functional prediction” in the main paper). We use a precomputed alignment for Sequence (see Chapter “4.2.3 – TARA++ versus existing NA methods in the task of protein functional prediction” in the main paper), hence the “N/A”s.

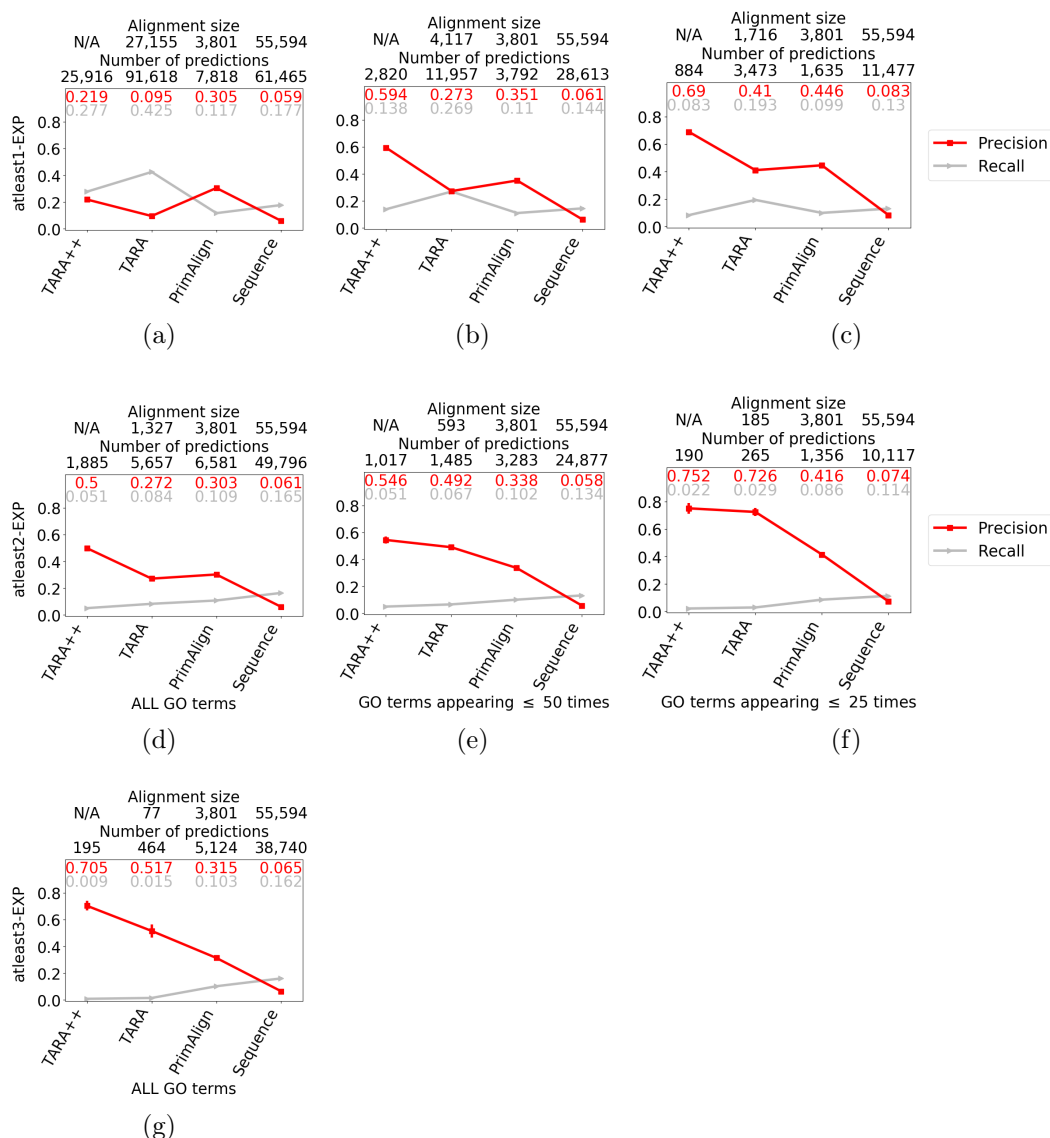


Figure C.22. Comparison of four NA methods for rarity thresholds (a, d, g) ALL, (b, e) 50, and (c, f) 25 using ground truth datasets (a, b, c) atleast1-EXP, (d, e, f) atleast2-EXP, and (g) atleast3-EXP in the task of protein functional prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method are shown above, except that TARA++ does not have an alignment *per se*. i.e., TARA++ comes from the overlap of *predictions* made by TARA and TARA-TS; hence the “N/A”s. For example, the alignment for TARA in (a) contains 27,155 aligned yeast-human protein pairs, and predicts 91,618 protein-GO term associations. Raw precision and recall values are color-coded inside each panel. For TARA++ and TARA, results are averages over all balanced datasets; the standard deviations are small and thus invisible.



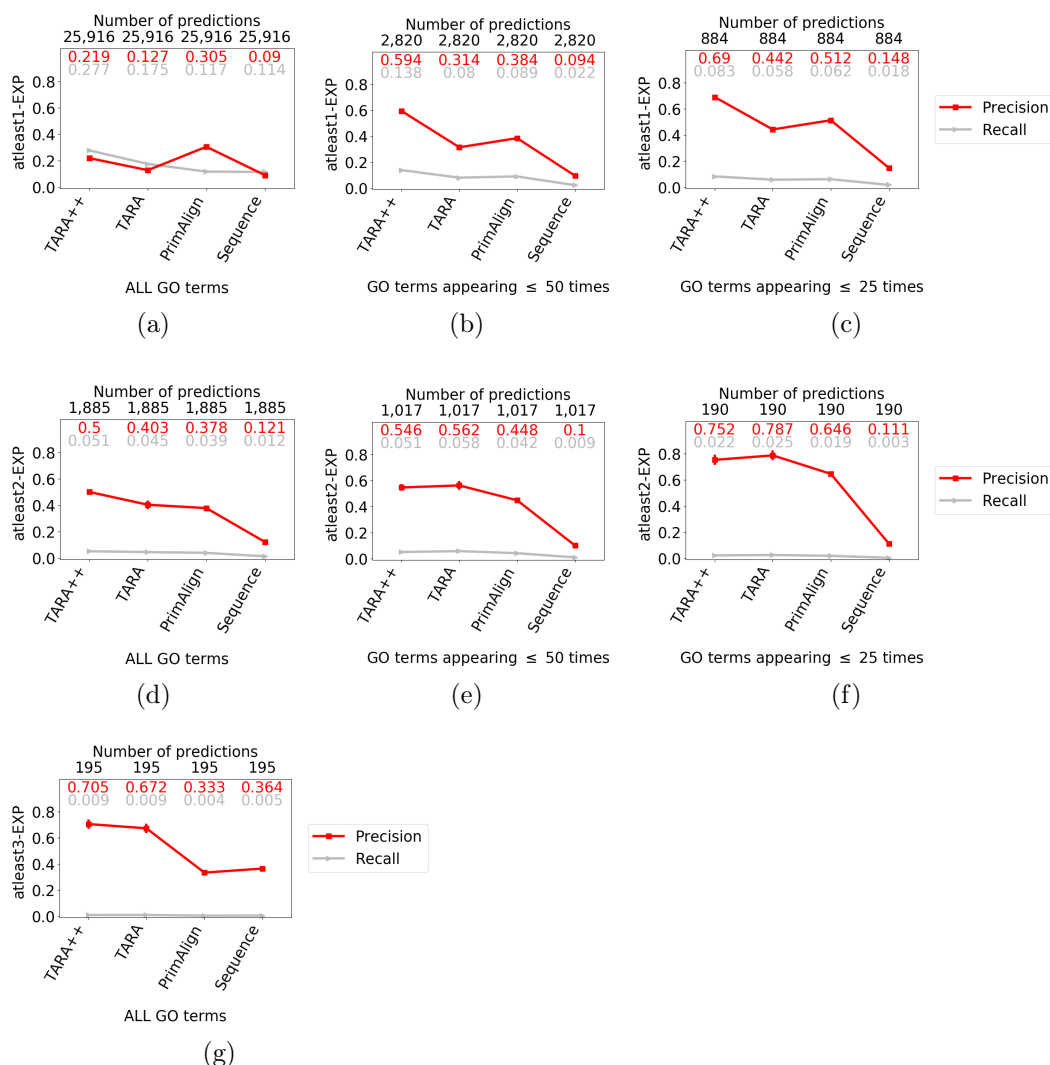


Figure C.23. Comparison of four NA methods for rarity thresholds (a, d, g) ALL, (b, e) 50, and (c, f) 25 using ground truth datasets (a, b, c) at least1-EXP, (d, e, f) at least2-EXP, and (g) at least3-EXP in the task of protein functional prediction. The alignment size (i.e., the number of aligned yeast-protein pairs) and number of functional predictions (i.e., predicted protein-GO term associations) made by each method are shown above, except that TARA++ does not have an alignment *per se*. i.e., TARA++ comes from the overlap of *predictions* made by TARA and TARA-TS; hence the “N/A”s. For example, the alignment for TARA in (a) contains 27,155 aligned yeast-human protein pairs, and predicts 91,618 protein-GO term associations. Raw precision and recall values are color-coded inside each panel. For TARA++ and TARA, results are averages over all balanced datasets; the standard deviations are small and thus invisible.

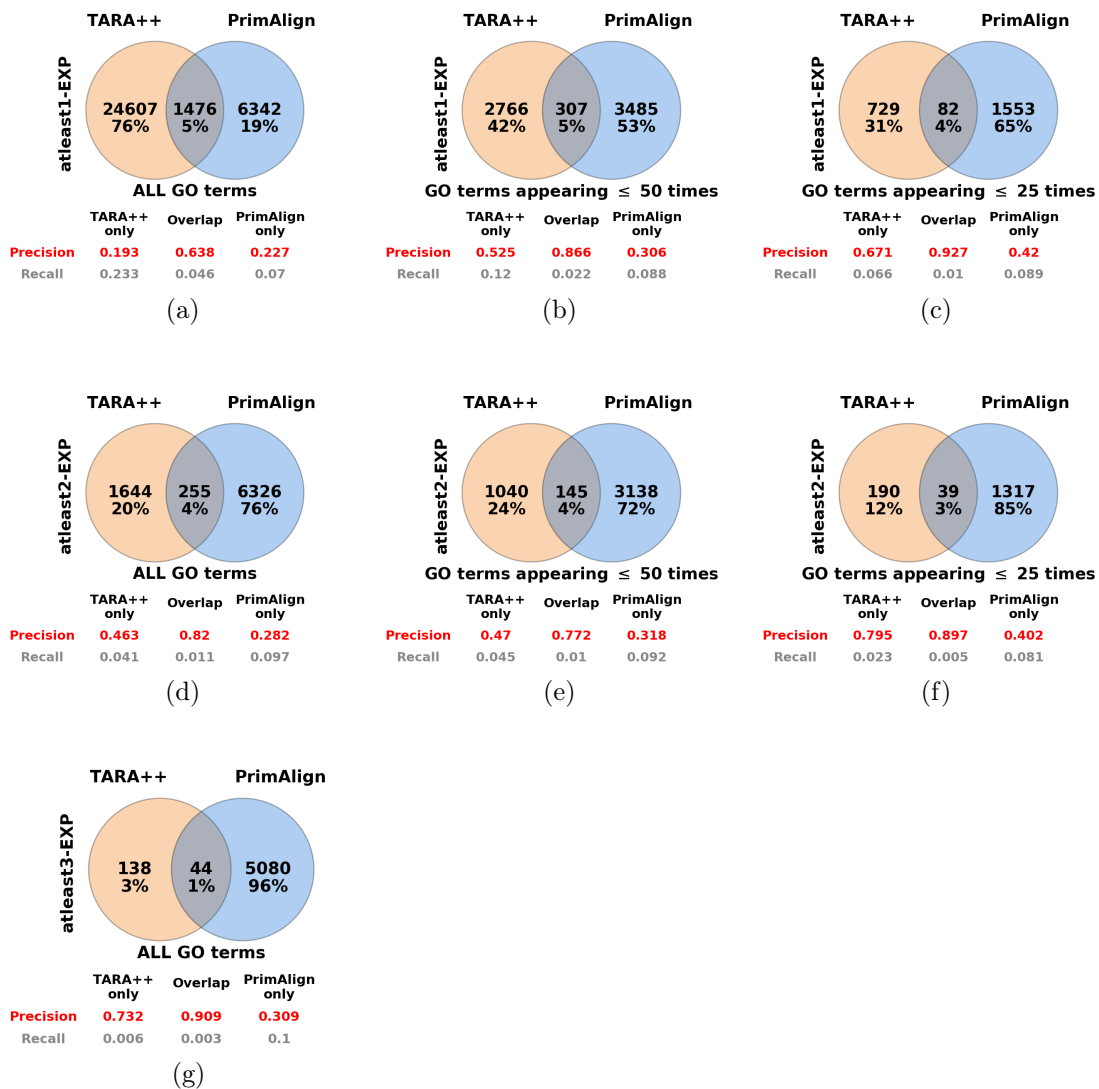


Figure C.24. Overlap of the predictions made by TARA++ and PrimAlign for rarity thresholds (a, d, g) ALL, (b, e) 50, and (c, f) 25 using ground truth datasets (a, b, c) atleast1-EXP, (d, e, f) atleast2-EXP, and (g) atleast3-EXP. Percentages are out of the total number of unique predictions made by both methods combined. Precision and recall are shown for each of the three prediction sets captured by the Venn diagram. The overlaps are for one of the 10 balanced datasets.

## APPENDIX D

### MODELING MULTI-SCALE DATA VIA A NETWORK OF NETWORKS

#### D.1 Methods

##### D.1.1 Data

##### D.1.1.1 Our synthetic NoN generator

Let  $M$  be the set of random graph models we consider for generating the synthetic NoN. For random graph model  $m \in M$ , let  $m(x, y)$  be a random graph of type  $m$  with  $x$  nodes and  $y$  edges. Let  $M^{(2)} = M \times M$  be the set of all possible combinations of the elements of  $M$  with themselves. Let  $|V^{(2)}|$  be the target number of nodes at level 2,  $|E^{(2)}|$  be the target number of edges at level 2,  $|V^{(1)}|$  be the target number of nodes for each network at level 1, and  $|E^{(1)}|$  be the target number of edges for each network at level 1; these parameters allow us to generate synthetic NoNs that approximate the size of real-world NoNs. Note that in our synthetic NoN generation, we fix the size of the level 1 networks to eliminate any effect of level 1 network size; however, our model can easily generate level 1 networks of varying size.

For each  $(m_1, m_2) \in M^{(2)}$ , we generate  $k$  isolated NoN regions where, for each region, the level 2 network is of type  $m_2$  and every level 1 network is of type  $m_1$ . This results in  $k|M^{(2)}|$  total isolated NoN regions. After combining all of them, the resulting NoN should have  $|V^{(2)}|$  nodes and  $|E^{(2)}|$  edges. As such, for each

$(m_1, m_2) \in M^{(2)}$ , we generate  $k$  isolated NoN regions

$$\begin{aligned} \{G_{(m_1, m_2)}^{(2)} = m_2(\lfloor \frac{|V^{(2)}|}{k|M^{(2)}|} \rfloor, \lfloor \frac{|E^{(2)}|}{k|M^{(2)}|} \rfloor) \text{ and} \\ \{G_{(m_1, m_2)_i}^{(1)} = m_1(|V^{(1)}|, |E^{(1)}|) \text{ for } i \in \{1, \dots, \lfloor \frac{|V^{(2)}|}{k|M^{(2)}|} \rfloor\}. \end{aligned} \quad (\text{D.1})$$

Because real-world systems are likely to have many groups of nodes, we set  $k = 5$  for our synthetic NoNs, corresponding to five instances of each of the four random graph model combinations. Then, we connect these isolated NoN regions by randomly removing edges within level 2 node groups and randomly adding the same number of edges across level 2 nodes groups (*across-edge* amount). Specifically, we repeat the following process  $a\% \times |E^{(2)}|$  times: (i) randomly select a level 2 node group, (ii) randomly select an edge in that node group, (iii) delete that edge, (iv) randomly select two level 2 nodes from different node groups, and (v) add an edge between the selected nodes. If the resulting NoN is still disconnected, we redo the process with a different random seed. While we could impose a condition to guarantee connectedness, doing so would bias the generation. If a connected NoN can not be found after 10 tries, we just continue with the last one. We start with  $a = 5$  to retain most of the level 2 node groups' original GEO- and SF-like network topologies, and we vary  $a$  to be 25, 50, 75, and 95 to test the effect of breaking the network topologies down. This also means that at  $a = 5$  there is significant clustering (each level 2 node group consists of densely interconnected nodes), while at  $a = 95$  there is very little clustering.

We also introduce random rewiring to test each method's robustness to data noise (*rewire-noise* amount). Specifically, for  $r\%$  rewire-noise, for each level 1 network, we randomly delete  $\frac{r}{100} \times |E^{(1)}|$  edges and randomly add the same number back. For the level 2 network, for each node group, we randomly delete  $r\%$  of  $\lfloor \frac{|E^{(2)}|}{5|M^{(2)}|} \rfloor$  edges and randomly add the same number back. We vary  $r$  to be 0 (no noise), 10, 25, 50, 75, 100 (completely random).

#### D.1.1.2 PIN-PSN NoN

We construct a biological NoN using the human PIN and the proteins' associated PSNs. We obtain human PPI data from BioGrid [156] version 4.1.190. We keep only physical interactions, remove selfloops and multiedges, and take the largest connected component. This results in a final size of 18,708 nodes and 434,527 edges.

We map proteins in our PIN to their corresponding PDB IDs as follows. Considering the proteins' BioGrid IDs, we use UniProt's [34] mapping service (version 2020\_06) to obtain BioGrid-to-UniProt mappings. Any mappings that are not reviewed (i.e., not Swiss-Prot) are discarded. Next, we remove any mapped data when more than one BioGrid ID is mapped to a UniProt ID and vice versa, leaving only one-to-one mappings between BioGrid IDs and UniProt IDs. Then, we repeat the process starting with the proteins' official symbol IDs. As such, for each protein, we have two UniProt IDs: one originating from its BioGrid ID and the other from its official symbol ID. To remove any ambiguity moving forward, we only keep proteins whose two UniProt IDs are equal. In total, we have 16,079 such UniProt IDs.

Given these UniProt IDs, we again use UniProt's mapping service, but this time to map UniProt IDs to PDB IDs. Then, we remove any PDB ID whose PDB structure has a resolution greater than or equal to  $3.0\text{\AA}$ , as PDB considers these to be "low resolution" [131]. Next, to obtain a one-to-one mapping between UniProt IDs and PDB IDs, we form one set out of every protein sequence associated with the UniProt IDs and another set out of every protein sequence associated with every PDB chain (each PDB ID can have multiple corresponding chains). We perform all-vs-all protein sequence comparison using BLASTP [5] between these two sets and take only reciprocal best hits as our final one-to-one UniProt-to-PDB mappings. After this step, we have 4,776 PDB chains.

Regarding GO term labels, we only consider those GO terms with 20 or more positive instances to ensure there is enough data to perform classification on.

### D.1.2 Existing approaches for label prediction

Recall that we consider graph theoretic approaches based on graphlets and graph learning approaches, namely, SIGN and DiffPool.

Graphlets are small subgraphs (a path, triangle, square, etc.) that can be considered the building blocks of networks, and they can be used to extract features of both nodes and networks. For each node in a general network, for each automorphism orbit (intuitively, node symmetry group) in a graphlet, one can count the number of times the node is a part of a given graphlet orbit. These counts are summarized into the node’s feature, also called its *graphlet degree vector* (GDV); when considering up to 4-node graphlets, GDVs will have length 15. Then, to extract features of the entire network, GDVs of all nodes can be collected into the network’s *GDV matrix* (GDVM) feature. One drawback of the GDVM feature is that its dimensions depend on the number of nodes in the network – if performing graph classification of different sized networks using GDVM features, issues can arise. Thus, we also consider a transformation of the GDVM, the graphlet correlation matrix (GCM) [173], which always has the same dimensions regardless of network size.

Given these definitions of graphlet features for nodes in a general network or for the entire general network itself, we now explain which features we use for nodes in a level 2 network and which features we use for level 1 networks. For the former, we extract each level 2 node’s GDV (L2 GDV). For the latter, we extract each level 1 network’s GDVM and GCM (L1 GDVM and L1 GCM). We use L1 GDVM when analyzing synthetic NoNs since we found that it outperformed L1 GCM. For the biological NoN, L1 GCM is the only viable feature since level 1 networks (PSNs) have different numbers of nodes (amino acids).

Then, to obtain NoN graphlet features, we concatenate level 2 nodes’ L2 GDVs with their networks’ L1 GDVMs or L1 GCMs. This results in five graphlet-based features: those for level 1 networks (L1 GDVM and L1 GCM) that are used for

graph label prediction, those for nodes in a level 2 network (L2 GDV) that are used for node label prediction, and those for the entire NoN (L1 GDVM + L2 GDV and L1 GCM + L2 GDV) that are used for entity label prediction. In order to perform classification, for each graphlet-based feature, we train a logistic regression classifier (Supplementary Section D.1.4). So for example, when we say L2 GDV, we mean the L2 GDV feature under logistic regression.

SIGN consists of two parts. First, it extracts different types of adjacency matrices from a network. SIGN specifically considers the traditional adjacency matrix, the Personalized PageRank-based adjacency matrix [92], the triangle-induced adjacency matrix [118], and their powers (see Supplementary Section D.1.4 for which powers are used); these matrices are concatenated row-wise. Second, they are given as input into a neural network classifier. Mathematically, SIGN overall is equivalent to an ensemble of multiple one-layer-deep (i.e., shallow) GCN classifiers, which is why it is considered a graph learning approach.

DiffPool aims to perform graph classification. However, unlike graphlet-based approaches and SIGN, which extract “general purpose” features of nodes/networks that can be used in any downstream machine learning task (label prediction in our study), DiffPool does not extract general purpose features. Instead, for each input network, given initial features for each node, DiffPool uses a GNN to aggregate the nodes’ initial features into a summary hidden feature for the entire network. Then, given hidden features corresponding to the input networks, the GNN is trained to perform graph classification. Since the GNN is trained over many iterations, the hidden feature is dependent on the training data and can only be used as a part of DiffPool’s GNN. When we say L1 DiffPool, we mean its GNN with the initial features chosen (Supplementary Section D.1.4), for graph classification using only level 1 networks.

As SIGN and DiffPool are single-level graph learning approaches, we also combine them into an NoN graph learning approach. Given each level 2 node’s feature extracted by SIGN, we concatenate it with the level 2 node’s corresponding level 1 network’s hidden feature computed by DiffPool’s GNN. The GNN is then trained on these concatenated features to perform classification (note that any general purpose feature can be incorporated into DiffPool like this). When we say L1 DiffPool + L2 SIGN, we mean entity label prediction using the process described above, incorporating SIGN’s extracted feature into DiffPool’s GNN. So, we use three graph learning-based approaches: L1 DiffPool, L2 SIGN, and L1 DiffPool + L2 SIGN.

We also combine L1 GDVM + L2 GDV or L1 GCM + L2 GDV with L1 DiffPool + L2 SIGN to test whether integrating information across the graph theoretic and graph learning domains improves upon either alone. Graphlet-based features can be incorporated into DiffPool using the process described previously.

In total, we have five single-level approaches: L1 GDVM, L1 GCM, L2 GDV, L1 DiffPool, and L2 SIGN; and five NoN approaches: L1 GDVM + L2 GDV, L1 GCM + L2 GDV, L1 DiffPool + L2 SIGN, L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN, and L1 GCM + L2 GDV + L1 DiffPool + L2 SIGN.

Finally, note that we did test node2vec [66], a prominent random walk-based embedding method, as a graph learning approach. node2vec extracts general purpose features like graphlets and SIGN, so we used it with logistic regression. However, DiffPool outperformed node2vec in level 1 graph classification, SIGN outperformed node2vec in level 2 node classification, and L1 DiffPool + L2 SIGN outperformed any combination involving node2vec in level 2 node classification for the entire NoN.

### D.1.3 Our integrative GCN approach

Here, we describe how we generalize GCNs to apply to NoNs. First, we summarize basic GCNs. Second, we discuss our extensions.



The important unit of a GCN is the graph convolutional layer, which works as follows. For each node in some network  $G = (V, E)$ , the node’s features are aggregated with its neighbors’ features and then these aggregated features are propagated to the next layer of the neural network. More formally, summarized from [90], let  $A$  be the adjacency matrix of  $G$  and  $H$  be a  $|V| \times d$  matrix of  $G$ ’s nodes’ features at the current layer (the  $i^{th}$  row corresponds to the feature of the  $i^{th}$  node). Then, forward propagation is carried out through

$$f(H, A) = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H W), \quad (\text{D.2})$$

where  $W$  is the trainable weight matrix for the current layer,  $\sigma$  is an activation function,  $\tilde{A} = A + I$  is the adjacency matrix with self-loops added (so that mathematically, the aggregation actually includes each node’s features along with its neighbors features) and  $\tilde{D}$  is the diagonal node degree matrix used for normalizing the adjacency matrix.

Essentially, graph convolutions allow each node to see information about its neighbors. So, what if we generalized graph convolutions to NoNs so that each node sees information not only about its neighbors (in the same level), but also about its corresponding network at a lower level or about the network it is a part of at a higher level? This would be in line with our intuition that the feature of a protein should contain information about how it interacts with other proteins (i.e., its topology in the level 2 network) and properties of the protein itself that allow for such interactions (topology of level 1 nodes in its level 1 network). So, we define a two part NoN-GCN layer consisting of one part that propagates level 2 nodes and another part that propagates level 1 nodes that attempts to do this (below and Fig. D.1).

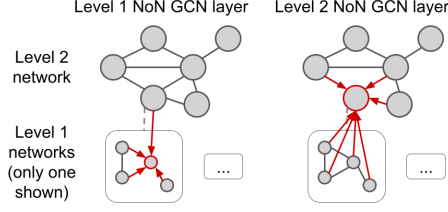


Figure D.1. Illustration of the two part NoN-GCN layer. In the level 1 NoN-GCN layer, the level 1 node circled in red receives features from its neighbors in its level 1 network as well as features of the level 2 node its level 1 network corresponds to. This is done for every level 1 node in every level 1 network. In the level 2 NoN-GCN layer, the level 2 node circled in red received features from its neighbors in the level 2 network as well as features of each of the level 1 nodes in its level 1 network. This is done for every level 2 node in the level 2 network.

Let  $\{G^{(2)} = (V^{(2)}, E^{(2)})$  and  $\{G_1^{(1)}, \dots, G_{|V^{(2)}|}^{(1)}\}$  be an NoN. Let  $A^{(2)}$  be the adjacency matrix of  $G^{(2)}$ . Let  ${}^kH^{(2)}$  be the  $|V^{(2)}| \times d$  matrix of features for  $G^{(2)}$  after the  $k^{th}$  neural network layer; for  $k = 0$ , this would correspond to the input feature matrix (for example,  $G^{(2)}$ 's GDVM or GCM). Let  ${}^kh_i^{(2)}$  be the feature vector of the  $i^{th}$  node  $v_i^{(2)} \in V^{(2)}$  (for example,  $v_i^{(2)}$ 's GDV). Let  $A_i^{(1)}$  be the adjacency matrix of  $v_i^{(2)}$ 's level 1 network  $G_i^{(1)}$ . Let  ${}^kH_i^{(1)}$  be the  $|V_i^{(1)}| \times d$  matrix of features for  $G_i^{(1)}$  after the  $k^{th}$  neural network layer. Let  ${}^kh_{ij}^{(1)}$  be the feature vector of the  $j^{th}$  node  $v_{ij}^{(1)}$  of  $G_i^{(1)}$ .

Propagation at level 2 works as follows. For each node  $v_i^{(2)}$ , for the feature matrix  ${}^kH_i^{(1)}$  of its corresponding level 1 network  $G_i^{(1)}$ , we take the average over all of  ${}^kH_i^{(1)}$ 's rows to obtain a  $1 \times d$  vector as a “summary” feature vector of  $G_i^{(1)}$ . Then, we combine these resulting vectors over all level 1 networks into a  $|V^{(2)}| \times d$  matrix  ${}^k\bar{H}^{(2)}$ , where each row corresponds to a level 1 network. In other words,  ${}^k\bar{H}^{(2)}$  can be thought of as the feature matrix of the level 2 network based on each node's level 1 network (whereas  ${}^kH^{(2)}$  is the feature matrix of the level 2 network based on each level 2 node). Then, our level 2 NoN-GCN layer forward propagation is carried out through

$$\begin{aligned}
{}^{k+1}H^{(2)} &= {}^{k+1}f_{l2}({}^kH^{(2)}, {}^k\bar{H}^{(2)}, A^{(2)}) = \\
&\sigma(D^{(2)\tilde{}}^{-\frac{1}{2}} \tilde{A}^{(2)} D^{(2)\tilde{}}^{-\frac{1}{2}} \\
&({}^kH^{(2)} + {}^k\bar{H}^{(2)}) {}^{k+1}W^{(2)}),
\end{aligned} \tag{D.3}$$

where  ${}^{k+1}W^{(2)}$  is the trainable weight matrix for the level 2 NoN-GCN layer,  $\sigma$  is an activation function,  $\tilde{A} = A + I$  is the adjacency matrix with self-loops added and  $\tilde{D}$  is the diagonal node degree matrix used for normalizing the adjacency matrix.

Propagation at level 1 works as follows. For each node  $v_{i_j}^{(1)}$  in each level 1 network  $G_i^{(1)} = (V_i^{(1)}, E_i^{(1)})$ , we sum its feature with all of its neighbors' features as well as the feature of  $G_i^{(1)}$ 's corresponding level 2 node. Mathematically, this corresponds to the following for each level 1 network. Let  ${}^k\bar{H}_i^{(1)}$  be a  $|V_i^{(1)}| \times d$  matrix consisting of  ${}^kh_i^{(2)}$  repeated  $|V_i^{(1)}|$  times. This can be thought of as the (naive) feature matrix of the level 1 network based on its corresponding level 2 node. Importantly  ${}^k\bar{H}_i^{(1)}$  has the same dimensions as  ${}^kH_i^{(1)}$ . Then, level 1 NoN-GCN layer forward propagation is carried out for one level 1 network through

$$\begin{aligned}
{}^{k+1}H_i^{(1)} &= {}^{k+1}f_{l1_i}({}^k\bar{H}_i^{(1)}, {}^kH_i^{(1)}, A_i^{(1)}) \\
&= \sigma(D_i^{(1)\tilde{}}^{-\frac{1}{2}} \tilde{A}_i^{(1)} D_i^{(1)\tilde{}}^{-\frac{1}{2}} \\
&({}^kH_i^{(1)} + {}^k\bar{H}_i^{(1)}) {}^{k+1}W_i^{(1)}),
\end{aligned} \tag{D.4}$$

where  ${}^{k+1}W_i^{(1)}$  is the trainable weight matrix for the  $i^{th}$  level 1 network for the current level 1 NoN-GCN layer,  $\sigma$  is an activation function,  $\tilde{A} = A + I$  is the adjacency matrix with self-loops added and  $\tilde{D}$  is the diagonal node degree matrix used for normalizing the adjacency matrix.

So, one full NoN-GCN layer takes as input

$$\begin{aligned}
& {}^k H^{(2)}, {}^k \bar{H}^{(2)}, A^{(2)}, \\
& \{{}^k \bar{H}_1^{(1)}, \dots, {}^k \bar{H}_{|V^{(2)}|}^{(1)}\}, \\
& \{{}^k H_1^{(1)}, \dots, {}^k H_{|V^{(2)}|}^{(1)}\}, \\
& \text{and } \{A_1^{(1)}, \dots, A_{|V^{(2)}|}^{(1)}\},
\end{aligned} \tag{D.5}$$

and returns  ${}^{k+1} H^{(2)}$  and  $\{{}^{k+1} H_1^{(1)}, \dots, {}^{k+1} H_{|V^{(2)}|}^{(1)}\}$ . These outputs can then be fed as inputs (along with  ${}^{k+1} \bar{H}^{(2)}$ , which can be constructed from  $\{{}^{k+1} H_1^{(1)}, \dots, {}^{k+1} H_{|V^{(2)}|}^{(1)}\}$ , and each  ${}^{k+1} \bar{H}_i^{(1)}$ , which can be constructed from its corresponding  ${}^{k+1} h_i^{(2)}$ ) into another NoN-GCN layer, thus allowing these layers to be chained.

We refer to a GCN approach using  $\lambda$  layers as “GCN- $\lambda$ ”.

Note that our implementation of the above is based on the **spektral** GNN library [65].

#### D.1.4 Evaluation

For a given NoN  $\{G^{(2)} = (V^{(2)}, E^{(2)})$  and  $\{G_1^{(1)}, \dots, G_{|V^{(2)}|}^{(1)}\}$ , its label set  $Y = y_1, \dots, y_c$ , and a function that maps level 2 nodes to their true labels  $f_{true} : V^{(2)} \rightarrow Y$ , the goal is to learn a predictive function  $f_{pred} : V^{(2)} \rightarrow Y$ . We do this by first splitting the data into three disjoint sets: training ( $V_{tr}^{(2)}$ ), validation ( $V_{val}^{(2)}$ ), and testing ( $V_{te}^{(2)}$ ). Then, we train a classifier on the training set that aims to minimize the cross-entropy loss between  $f_{true}(V_{tr}^{(2)})$  and  $f_{pred}(V_{tr}^{(2)})$ . We use  $V_{val}^{(2)}$  to optimize hyperparameters and finally report the classifier’s performance on  $V_{te}^{(2)}$ . Details are as follows.

Denote  $Y = y_1, \dots, y_c$  to be the set of possible level 2 node labels (recall for synthetic NoNs, given  $m$  random graph models, multiclass classification is done on  $m \times m$  labels; for the real-world NoN, for each of the 131 ground truth datasets, binary classification is done on whether proteins have the corresponding label or not)

and  $f_{true} : V^{(2)} \rightarrow Y$  to be a function that maps level 2 nodes to their true labels. We split the set of level 2 nodes  $V^{(2)}$  into three disjoint subsets as follows.  $p\%$  of the data is randomly removed from  $V^{(2)}$  and put into the training set  $V_{tr}^{(2)}$ . Half of the data remaining from  $V^{(2)}$  is randomly removed and put into the validation set  $V_{val}^{(2)}$ . The remaining data is put into the testing set  $V_{te}^{(2)}$ . This results in three sets with size ratio  $p:\frac{1-p}{2}:\frac{1-p}{2}$ . Importantly, this splitting is done with the constraint that the distribution of node labels in each of the three sets matches the original label distribution of  $V^{(2)}$  as closely as possible (i.e., stratified sampling). We train the classifier on  $V_{tr}^{(2)}$ , optimize hyperparameters using  $V_{val}^{(2)}$ , and report results on  $V_{te}^{(2)}$ . We repeat the random data splitting 3 different times and perform classification for each, reporting the average results over them. We do this 3 times so that 1) the effect of randomness from sampling reduced and 2) running the the approaches is still computationally feasible. For synthetic NoNs, we choose  $p = 0.8$  (corresponding to a 8:1:1 data ratio), as this is a common split amount when data is not scarce. For real-world NoNs, we choose  $p = 1/3$  (corresponding to a 1:1:1 data ratio). Because some of the ground truth sets have as few as 20 positive instances, larger values of  $p$  would result in the validation/testing sets having too few of them.

Below, we describe classifier details. For graphlet-based approaches, we use each of L1 GDVM, L1 GCM, L2 GDV, L1 GDVM + L2 GDV, and L1 GCM + L2 GDV in logistic regression. For L2 SIGN, we use its features in own classifier. We refer to these as “regular classification”. For approaches involving DiffPool, we run them as described in Supplementary Section D.1.2. We refer to these as “DiffPool-based classification”. Finally, we refer to classification using NoN-GCNs as “NoN-GCN-based” classification.

For a given data split, for each feature we consider in regular classification, we train the corresponding classifier using the ADAM optimizer on  $V_{tr}^{(2)}$ . We test the following learning rates  $\{0.1, 0.01, 0.001\}$  and choose the best one with respect to

performance when predicting on  $V_{val}^{(2)}$ . Then, we use this best classifier to predict on  $V_{te}^{(2)}$ .

For a given data split, for DiffPool-based classification, we perform a grid search over the following hyperparameters: **hidden dimension:**  $\{32, 64, 128\}$  and **output dimension:**  $\{32, 64, 128\}$ . We choose the best combination with respect to performance when prediction on  $V_{val}^{(2)}$  and use this best classifier to predict on  $V_{te}^{(2)}$ .

For a given data split, for NoN-GCN-based classification, we train a neural network that consists of two NoN-GCN layers, each followed by dropout layers, followed by a logistic regression classifier (i.e., one fully connected hidden layer). We specifically add this logistic regression classifier on the end of the neural network, rather than directly performing classification from the final NoN-GCN layer, to make the NoN-GCN-based classification as fairly comparable as possible to the regular classification. Note that for synthetic NoNs with two random graph models, we tested a version of the neural network with three NoN-GCN layers. However, because two NoN-GCN layers was as good as three for the majority of the evaluation tests, and because three layers took much more time to compute, we continued with two layers. We also use the ADAM optimizer. We perform a grid search over the following hyperparameters: **learning rate:**  $\{0.1, 0.01, 0.001\}$ , **dropout:**  $\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ , **hidden dimension:**  $\{128, 256, 512\}$  and choose the best combination with respect to performance when predicting on  $V_{val}^{(2)}$ . Then, we use this best classifier to predict on  $V_{te}^{(2)}$ .

Both DiffPool and our NoN-GCN require initial features. Ideally, they should use the same type of initial features so that they are as fairly comparable as possible. Our NoN-GCN has stricter limitations for what initial features can be used because it requires level 2 nodes' initial features to be in the same low dimensional space as level 1 nodes' initial features, otherwise it does not make sense to aggregate them. So, we determined initial features for our NoN-GCN first. We tested random features

of lengths 128, 256, and 512, and nodes' GDVs (each index in the GDV corresponds to the number of times the node participates in that specific graphlet orbit; hence, GDVs are in the same low dimensional space) and found that GDVs were the best. So, we use nodes' GDV as initial features for our NoN-GCN. Thus, we also use nodes' GDVs as initial features into DiffPool.

For synthetic NoNs, we report classification accuracy ( $\#$  of correct predictions / total  $\#$  of entities) since class sizes are balanced. For the real-world NoNs, we report area under precision-recall (AUPR), precision@k, recall@k, and F-score@k, since class sizes are not balanced. Here @k refers to the corresponding measure when only considering the top k predictions. That is, for each approach, for each GO term, we rank each protein for which a prediction is made by the probability that it annotated by the given GO term, as determined by the approach's classifier. Then, we compute the corresponding measure on the top k items of the ranked list. To determine k, for each approach, for each GO term, we do the following. We choose the  $k$  that maximizes the F-score@k where precision@k is greater than recall@k. We impose precision@k to be greater than recall@k because we believe that in the biomedical domain, precision is more important – fewer but mostly correct predictions (e.g., 9 correct out of 10 made), which corresponds to high precision, is better than a greater number of mostly incorrect predictions (e.g., 300 correct out of 1,000 made), which corresponds to high recall, in terms of potential wet lab validation. By choosing  $k$  in this way, we give each classifier the best case advantage. We report precision, recall, and F-score at this  $k$ .

We also test if each approach's performance is significantly better than random. That is, given an approach, for each measure, for each GO term, we use a one sample one-tailed t-test (recall that each approach is run 3 times, corresponding to 3 different training/validation/testing splits) to see if the approach's performance is significantly greater than the value expected by random. Then, for each measure, for

each approach, we perform FDR correction over the 131 GO terms. For each measure, for each GO term, any approach with a corrected p-value  $< 0.05$  is considered significantly better than random for that GO term.

## D.2 Results

### D.2.1 Synthetic NoNs

We expect an approach only using one level to reach an accuracy of  $\frac{\# \text{ of models}}{\# \text{ of labels}}$ , i.e., 0.5. To see why, consider the following example using the L1 GDVM approach. Here, there are four possible labels corresponding to the four possible combinations of random graph models at each level: GEO-GEO, GEO-SF, SF-GEO, SF-SF. Since L1 GDVM uses level 1 information, it will be able to distinguish between GEO and SF level 1 networks but not between level 2 nodes with GEO- and SF- topology. So, L1 GDVM will only have enough information to predict  $\frac{2}{4} = 0.5$  of the labels correctly.

### D.2.2 Biological NoN



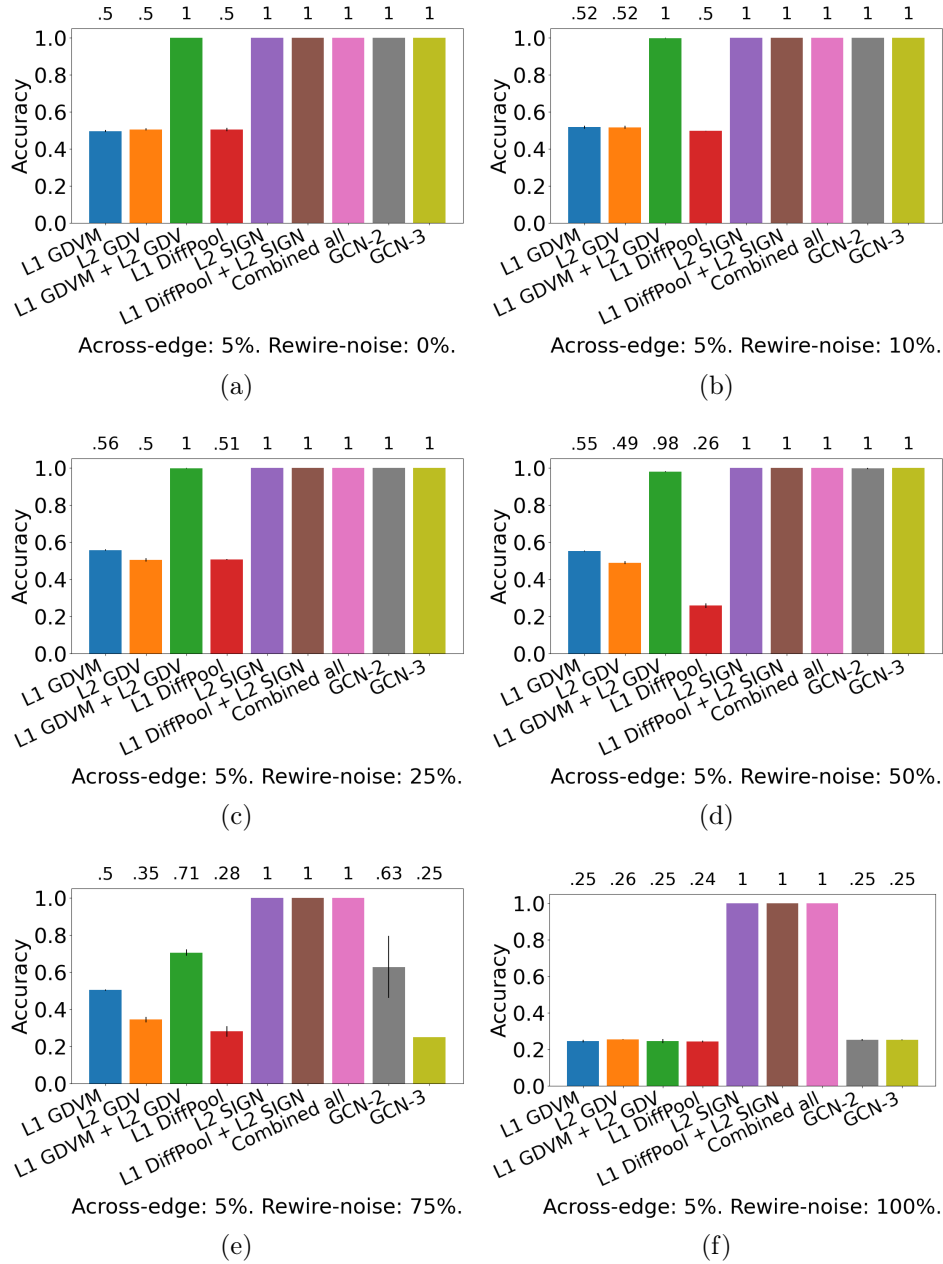


Figure D.2. Comparison of the nine relevant approaches in the task of entity label prediction for synthetic NoNs with 5% across-edge amount and the following rewiring noise amounts: (a) 0%, (b) 10%, (c) 25%, (d) 50%, (e) 75%, and (f) 100%. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw prediction accuracies are shown above. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars.

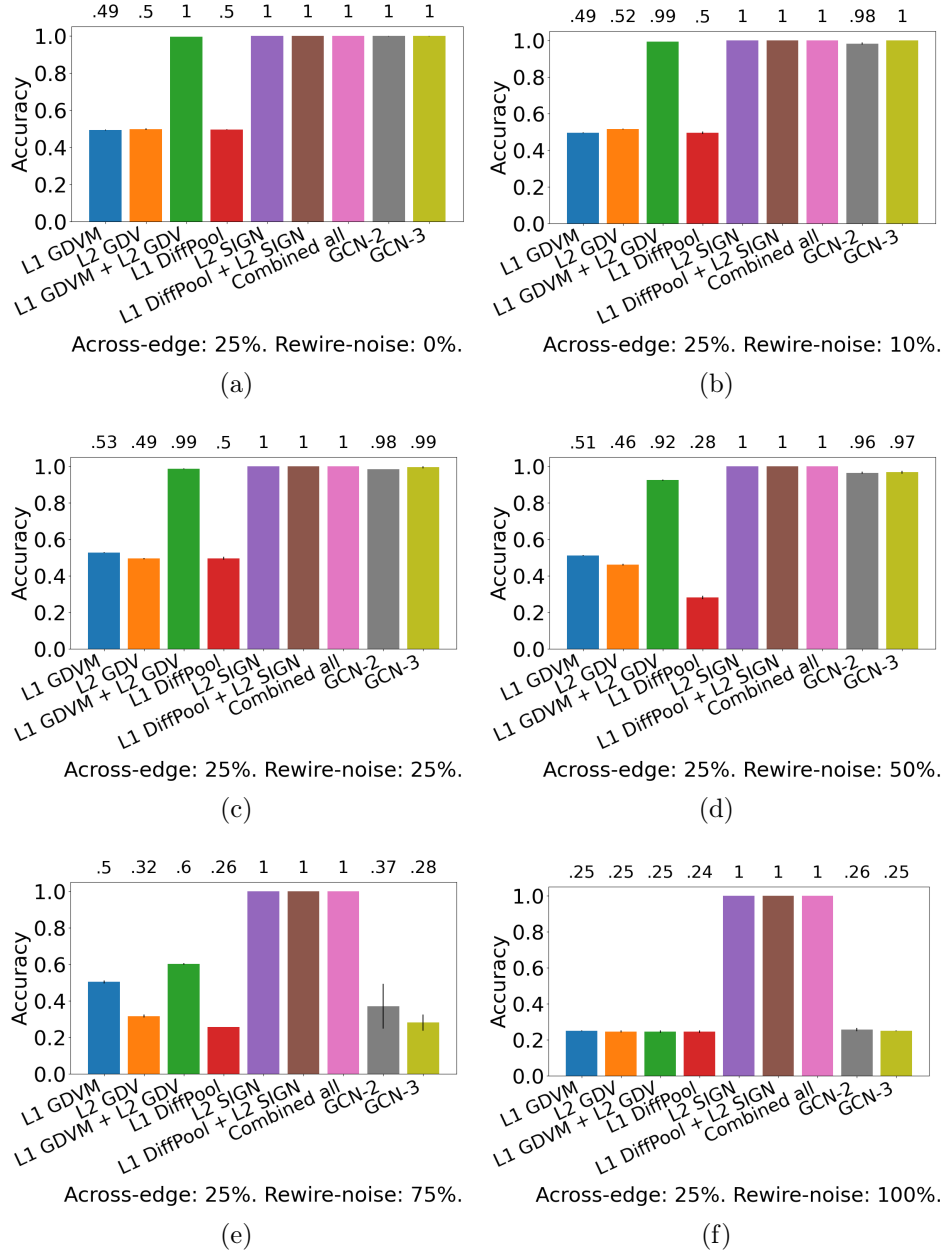


Figure D.3. Comparison of the nine relevant approaches in the task of entity label prediction for synthetic NoNs with 25% across-edge amount and the following rewiring noise amounts: (a) 0%, (b) 10%, (c) 25%, (d) 50%, (e) 75%, and (f) 100%. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw prediction accuracies are shown above. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars.

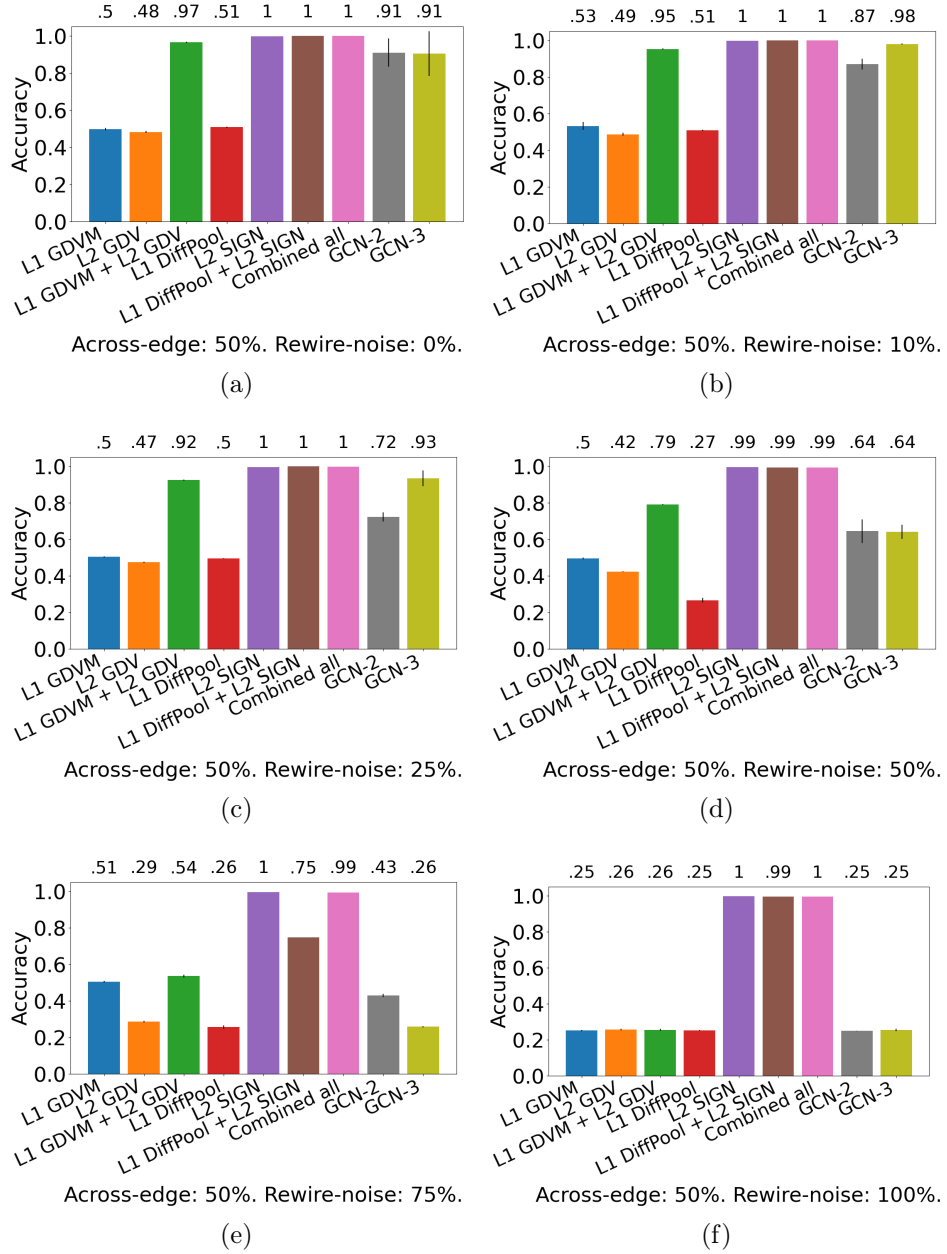


Figure D.4. Comparison of the nine relevant approaches in the task of entity label prediction for synthetic NoNs with 50% across-edge amount and the following rewiring noise amounts: (a) 0%, (b) 10%, (c) 25%, (d) 50%, (e) 75%, and (f) 100%. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw prediction accuracies are shown above. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars.

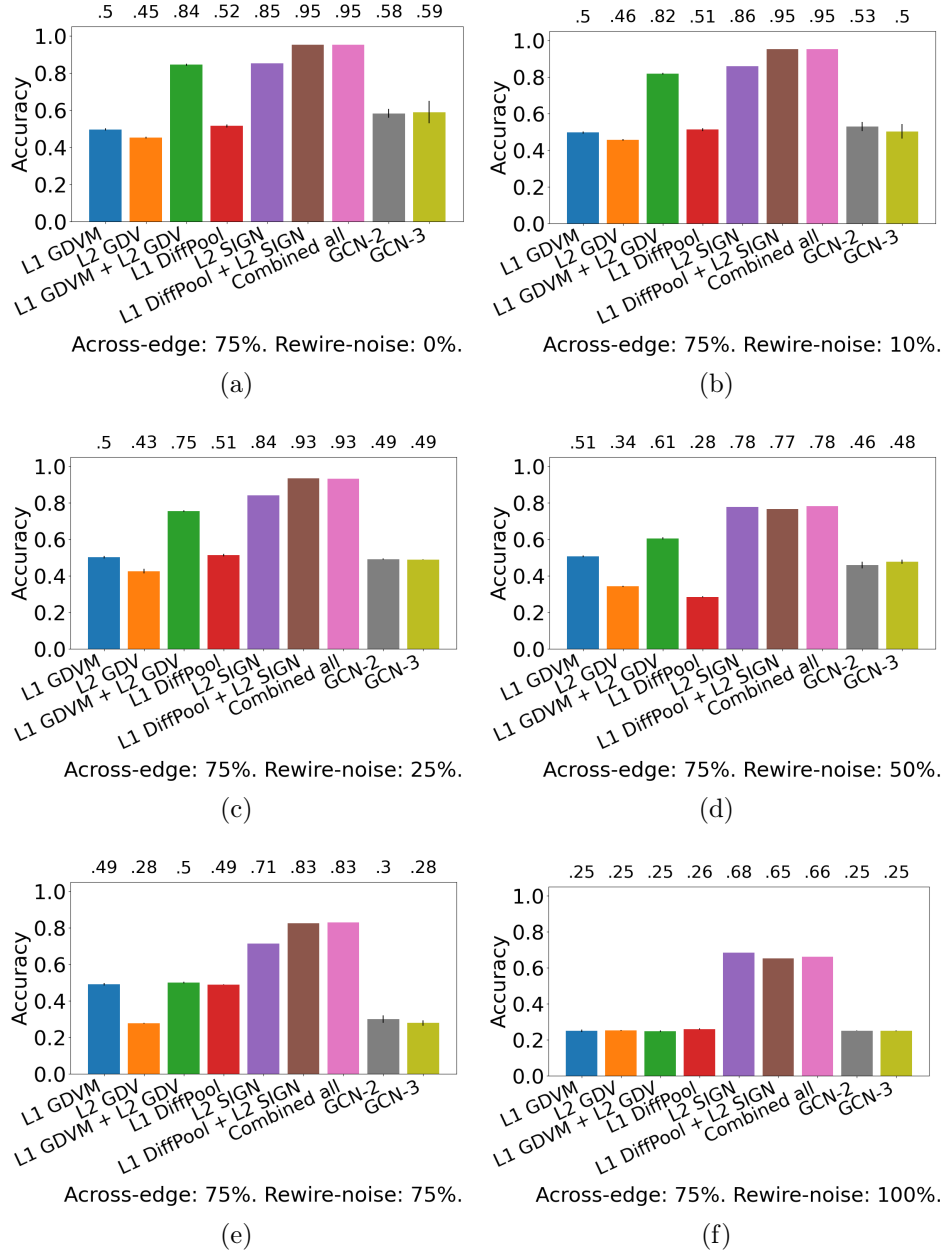


Figure D.5. Comparison of the nine relevant approaches in the task of entity label prediction for synthetic NoNs with 75% across-edge amount and the following rewiring noise amounts: (a) 0%, (b) 10%, (c) 25%, (d) 50%, (e) 75%, and (f) 100%. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw prediction accuracies are shown above. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars.

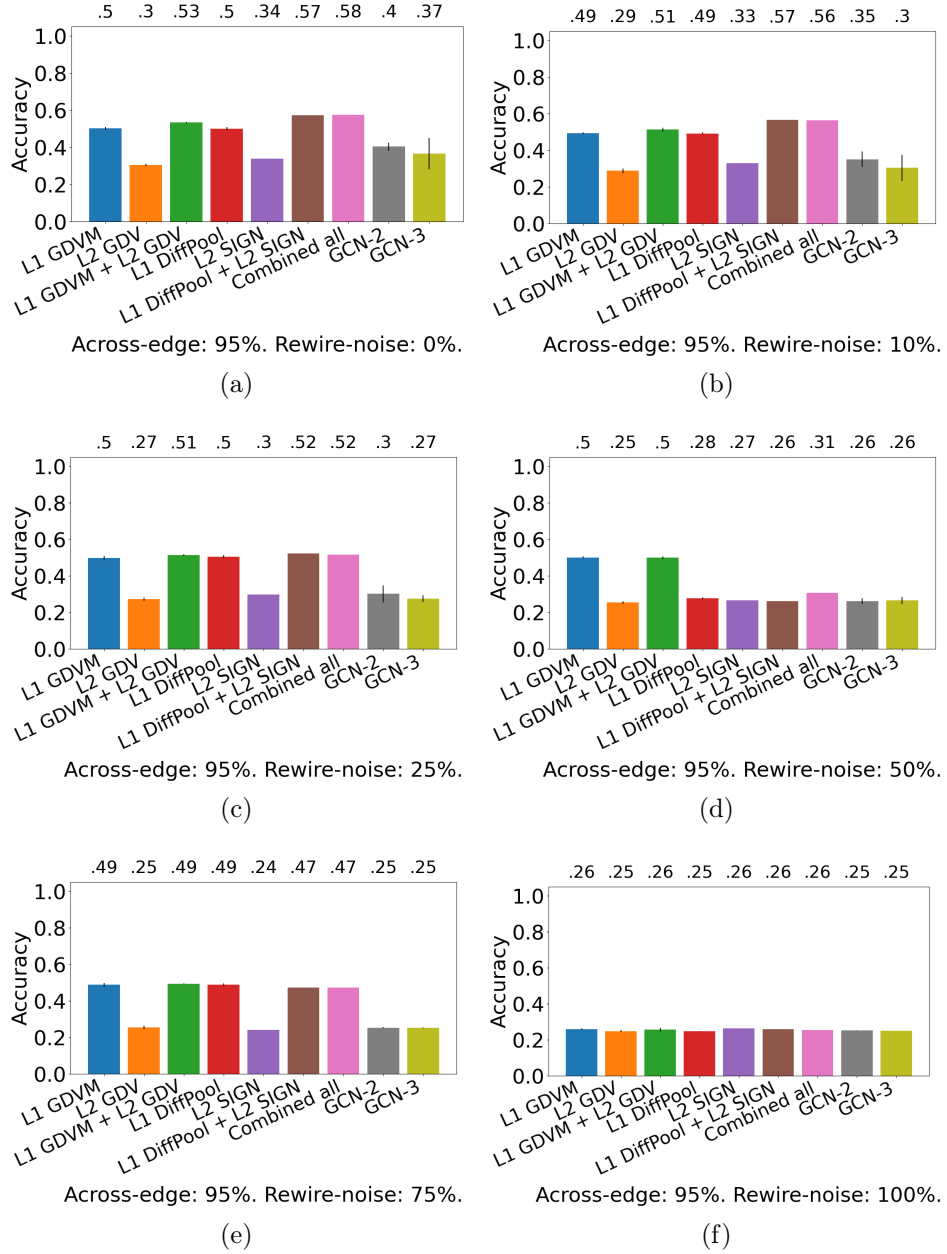


Figure D.6. Comparison of the nine relevant approaches in the task of entity label prediction for synthetic NoNs with 95% across-edge amount and the following rewiring noise amounts: (a) 0%, (b) 10%, (c) 25%, (d) 50%, (e) 75%, and (f) 100%. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw prediction accuracies are shown above. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Accuracy is shown above the bars.

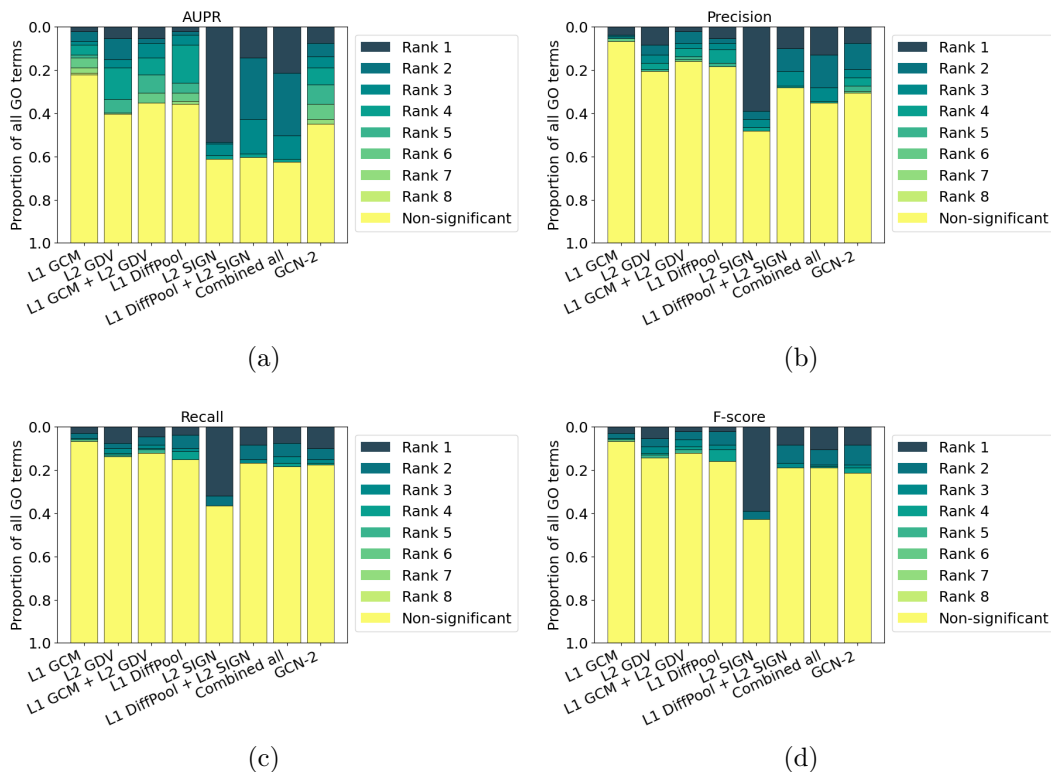


Figure D.7. Summarized results of the eight relevant approaches in the task of protein functional prediction for evaluation measures (a) AUPR, (b) precision, (c) recall, and (d) F-score. For each GO term (out of the 131 total), we rank the eight approaches' classification performances from best (rank 1) to worst (rank 8). If an approach's performance is not significantly better than expected by random we deem it "non-significant" instead. Then for each approach, we calculate the proportion of times it achieves each rank. "Combined all" refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN.

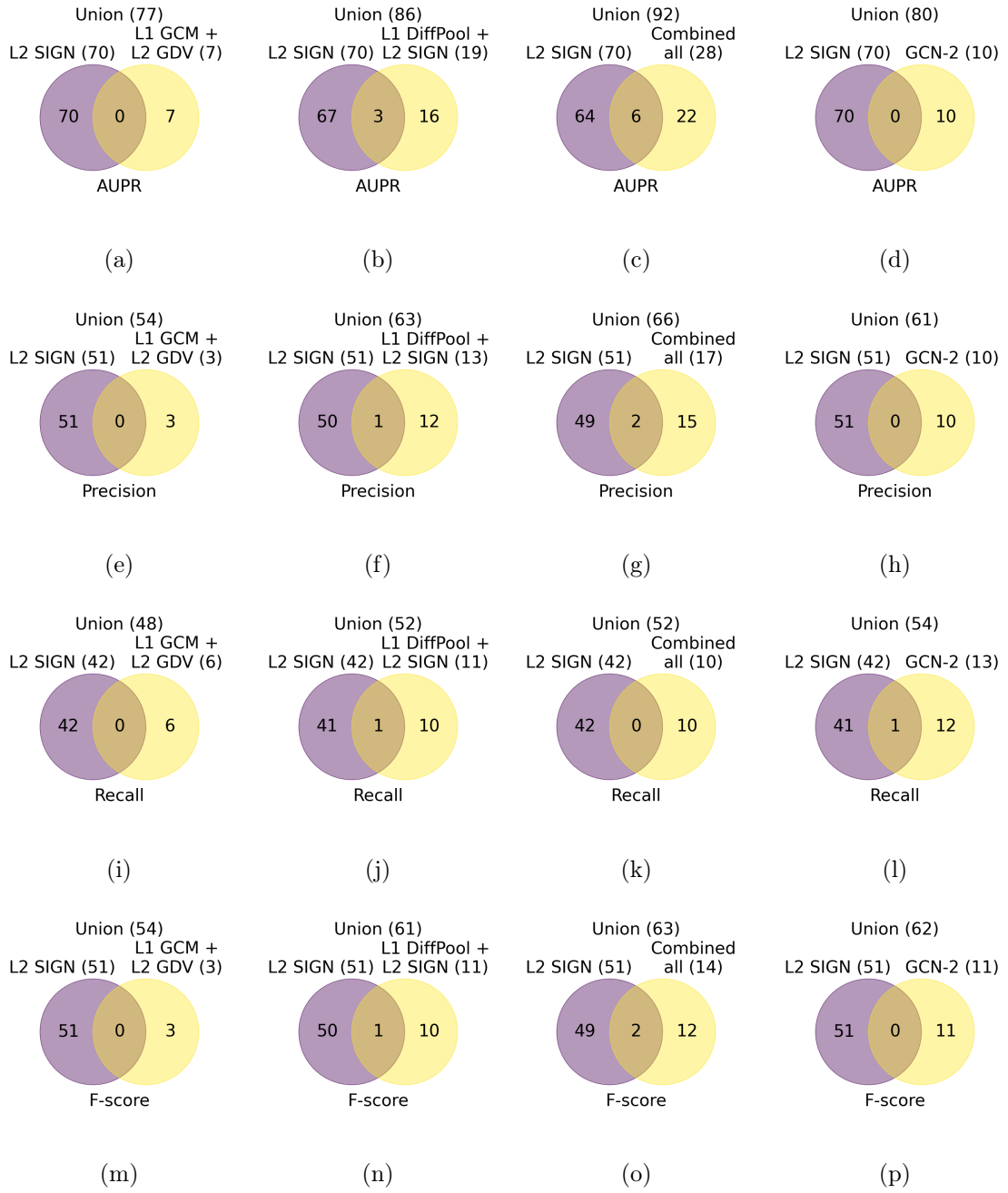


Figure D.8: Overlap of GO terms for which L2 SIGN is the best with those for which (a, e, i, m) L1 GCM + L2 GDV, (b, f, j, n) L1 DiffPool + L2 SIGN, (c, g, k, o) Combined all (aka L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN), and (d, h, l, p) GCN-2 are the best in terms of (a, b, c, d) AUPR, (e, f, g, h) precision, (i, j, k, l) recall, and (m, n, o, p) F-score.

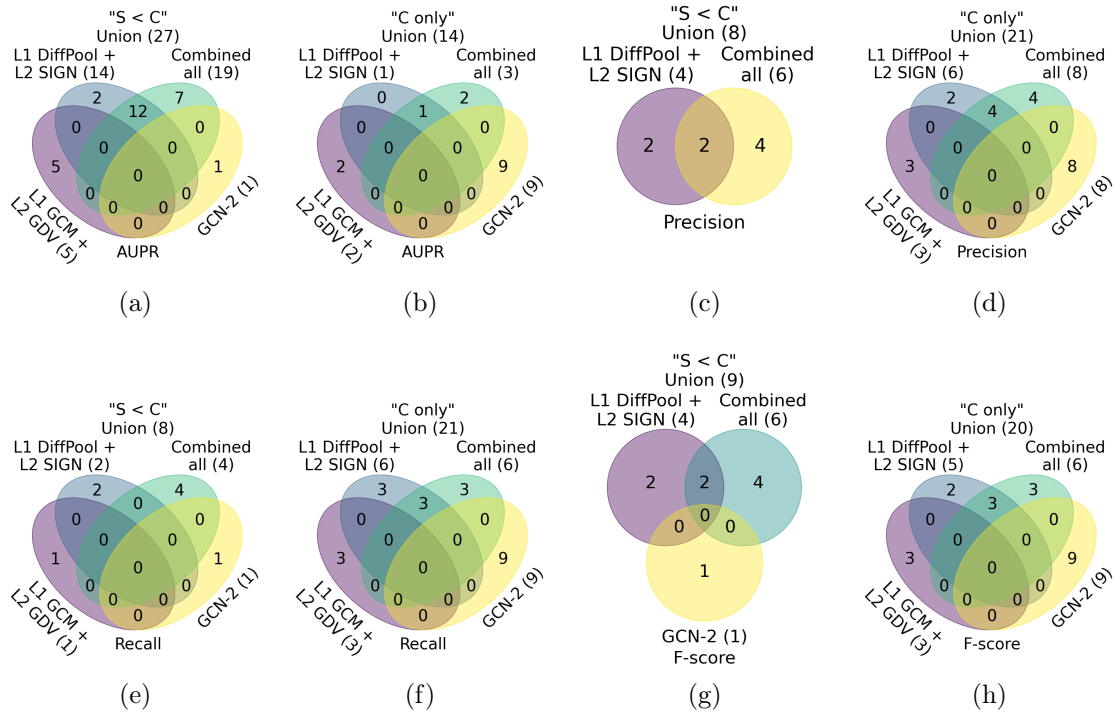


Figure D.9: Overlaps of the four combined level approaches for groups (a, c, e, g) "S < C" and (b, d, f, h) "C only" in terms of (a, b) AUPR, (c, d) precision, (e, f) recall, (g, h) F-score.



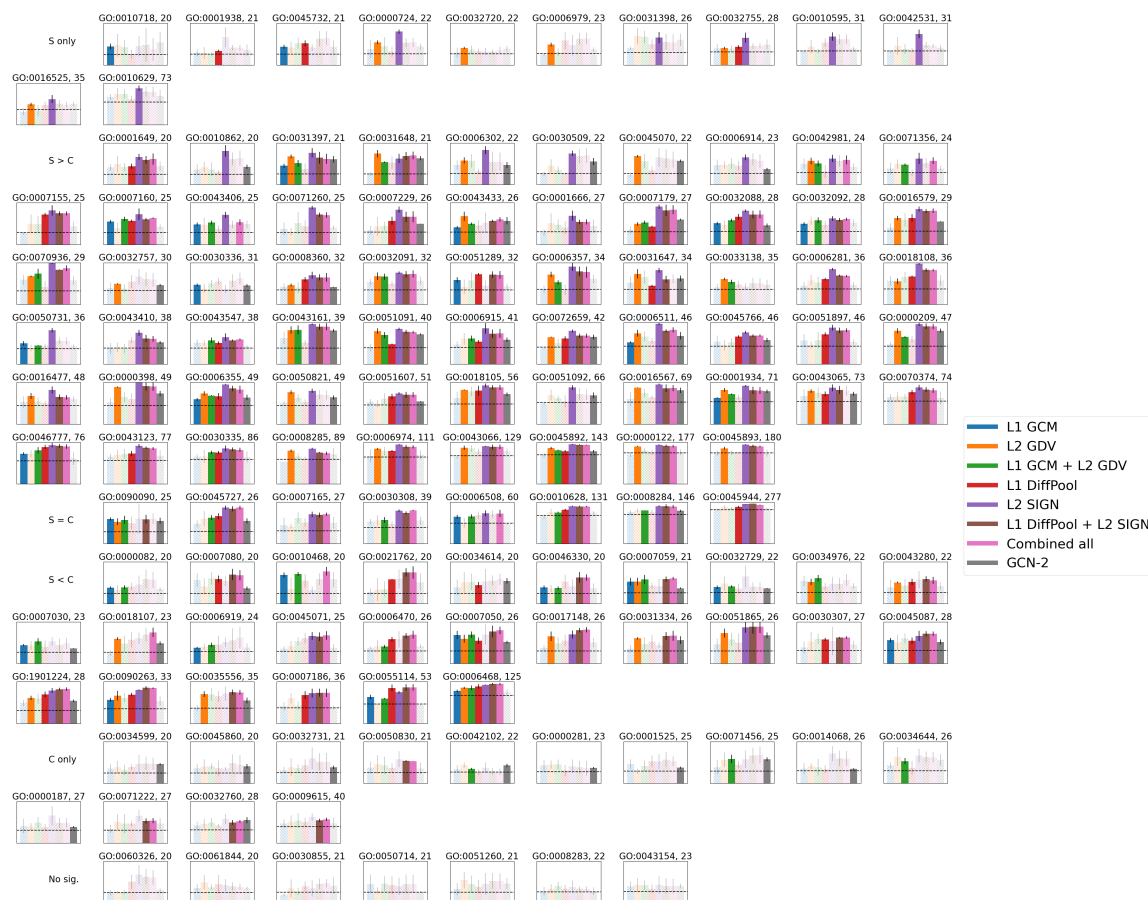


Figure D.10: Classification performance of the eight relevant approaches for each GO term in terms of AUPR. GO term IDs and the number of positive instances for that GO term are shown above. Random performance is indicated by the dotted black line. Approaches with performance not significantly greater than random are shown in a lighter shade. GO terms are split into the six groups based on how single versus combined level approaches perform. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw scores for each approach for each GO term can be found in Supplementary File D.1.



Figure D.11: Classification performance of the eight relevant approaches for each GO term in terms of precision. GO term IDs and the number of positive instances for that GO term are shown above. Random performance is indicated by the dotted black line. Approaches with performance not significantly greater than random are shown in a lighter shade. GO terms are split into the six groups based on how single versus combined level approaches perform. “Combined all” refers to L1 GDV + L2 GDV + L1 DiffPool + L2 SIGN. Raw scores for each approach for each GO term can be found in Supplementary File D.2.



Figure D.12: Classification performance of the eight relevant approaches for each GO term in terms of recall. GO term IDs and the number of positive instances for that GO term are shown above. Random performance is indicated by the dotted black line. Approaches with performance not significantly greater than random are shown in a lighter shade. GO terms are split into the six groups based on how single versus combined level approaches perform. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw scores for each approach for each GO term can be found in Supplementary File D.3.



Figure D.13: Classification performance of the eight relevant approaches for each GO term in terms of F-score. GO term IDs and the number of positive instances for that GO term are shown above. Random performance is indicated by the dotted black line. Approaches with performance not significantly greater than random are shown in a lighter shade. GO terms are split into the six groups based on how single versus combined level approaches perform. “Combined all” refers to L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN. Raw scores for each approach for each GO term can be found in Supplementary File D.4.

### D.2.3 Running times

Reported times for all approaches except those involving DiffPool (L1 DiffPool, L1 DiffPool + L2 SIGN, and L1 GDVM + L2 GDV + L1 DiffPool + L2 SIGN) are obtained by running on the same machine, fully using one core, for fairness; of course, for practical purposes, some approaches can easily be parallelized given available resources. Training for DiffPool-based approaches must be done on GPU. We report their training times on a cluster machine, which means that their times for training are affected by resource availability/scheduling. While DiffPool-based approaches are not run under the same conditions as other approaches, we still commented on their running times, as in a realistic scenario, approaches may be run using different resources as we have done here.

We run all approaches except those involving DiffPool using one core on a 64-core AMD Opteron 6376 machine. We run approaches involving DiffPool on a cluster machine with Dual Twelve-core 2.2GHz Intel Xeon processors and 4 NVIDIA GeForce GTX 1080 Ti GPUs, accessed through Notre Dame’s Center for Research Computing.

TABLE D.1

RUNNING TIMES OF EACH APPROACH IN SECONDS. “COMBINED ALL” REFERS TO L1 GDVM + L2 GDV + L1 DIFFPOOL + L2 SIGN

	Feature extraction	Training (1 epoch)	Total
L1 GDVM	485.0	2.2	487.2
L2 GDV	2.1	1.5	3.6
L1 GDVM + L2 GDV	487.1	2.1	489.2
L1 DiffPool	485.0	140.1	625.3
L2 SIGN	6.4	17.6	23.4
L1 DiffPool + L2 SIGN	491.4	25.6	516.4
Combined all	493.5	29.4	522.5
GCN-2	487.1	30.3	517.1
GCN-3	487.1	128.8	615.1

### D.3 Supplementary files

File D.1. GuS022022D\_supplementary4.csv. Raw AUPR scores of the eight relevant approaches for each GO term in each of the six groups.

File D.2. GuS022022D\_supplementary5.csv. Raw precision scores of the eight relevant approaches for each GO term in each of the six groups.

File D.3. GuS022022D\_supplementary6.csv. Raw recall scores of the eight relevant approaches for each GO term in each of the six groups.

File D.4. GuS022022D\_supplementary7.csv. Raw F-scores of the eight relevant approaches for each GO term in each of the six groups.

## BIBLIOGRAPHY

1. S. Aguinaga, D. Chiang, and T. Weninger. Learning hyperedge replacement grammars for graph generation. *IEEE transactions on pattern analysis and machine intelligence*, 41(3):625–638, 2018.
2. N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield. Efficient graphlet counting for large networks. In *2015 IEEE International Conference on Data Mining (ICDM)*, pages 1–10. IEEE, 2015.
3. A. E. Aladağ and C. Erten. SPINAL: scalable protein interaction network alignment. *Bioinformatics*, 29(7):917–924, 2013.
4. F. Alkan and C. Erten. BEAMS: backbone extraction and merge strategy for the global many-to-many alignment of multiple PPI networks. *Bioinformatics*, 30(4):531–539, 2014.
5. S. F. Altschul, T. L. Madden, A. A. Schäffer, et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.
6. D. Aparício, P. Ribeiro, T. Milenković, and F. Silva. GoT-WAVE: Temporal network alignment using graphlet-orbit transitions. *arXiv preprint arXiv:1808.08195*, 2018.
7. D. Aparício, P. Ribeiro, T. Milenković, and F. Silva. Temporal network alignment via GoT-WAVE. *Bioinformatics*, 35(18):3527–3529, 2019.
8. M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, et al. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25, 2000.
9. S. Bamford, E. Dawson, S. Forbes, J. Clements, R. Pettett, A. Dogan, A. Flanagan, J. Teague, P. A. Futreal, M. R. Stratton, et al. The COSMIC (Catalogue of Somatic Mutations in Cancer) database and website. *British Journal of Cancer*, 91(2):355, 2004.
10. A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.
11. A.-L. Barabási et al. *Network science*. Cambridge University Press, 2016.



12. D. S. Bassett and M. G. Mattar. A network neuroscience of human learning: Potential to inform quantitative theories of brain and behavior. *Trends in Cognitive Sciences*, 21(4):250–264, 2017.
13. A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18, 2018.
14. N. C. Berchtold, D. H. Cribbs, P. D. Coleman, J. Rogers, E. Head, R. Kim, T. Beach, C. Miller, J. Troncoso, J. Q. Trojanowski, et al. Gene expression changes in the course of normal brain aging are sexually dimorphic. *Proceedings of the National Academy of Sciences*, 105(40):15605–15610, 2008.
15. D. Berenberg, V. Gligorijevic, and R. Bonneau. Graph embeddings for protein structural comparison. *3DSIG: Structural Bioinformatics and Computational Biophysics at The 29th Conference on Intelligent Systems for Molecular Biology and the 20th European Conference on Computational Biology (ISMB/ECCB 2021)*., Jul 2021.
16. J. Berg and M. Lässig. Local graph alignment and motif search in biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(41):14689–14694, 2004.
17. J. Berg and M. Lässig. Cross-species analysis of biological networks by bayesian alignment. *Proceedings of the National Academy of Sciences*, 103(29):10967–10972, 2006.
18. C. Berge. *Graphs and hypergraphs*. North-Holland Pub. Co., 1973.
19. H. M. Berman, J. Westbrook, Z. Feng, et al. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.
20. S. Bhagat, G. Cormode, and S. Muthukrishnan. Node classification in social networks. In *Social Network Data Analytics*, pages 115–148. Springer, 2011.
21. B.-J. Breitkreutz, C. Stark, T. Regul, L. Boucher, A. Breitkreutz, M. Livstone, R. Oughtred, D. H. Lackner, J. Bähler, V. Wood, K. Dolinski, and M. Tyers. The BioGRID Interaction Database: 2008 update. *Nucleic Acids Research*, 36: D637–D640, 2008. doi: 10.1093/nar/gkm1001.
22. H. Cai, V. W. Zheng, and K. C.-C. Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.
23. X. Cao, Z. Chen, X. Zhang, and Y. Yu. IMAP: An iterative method for aligning protein-protein interaction networks. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 317–324. IEEE, 2017.

24. A. Chatr-Aryamontri, R. Oughtred, L. Boucher, J. Rust, C. Chang, N. K. Kolas, L. O'Donnell, S. Oster, C. Theesfeld, A. Sellam, et al. The BioGRID interaction database: 2017 update. *Nucleic Acids Research*, 45(D1):D369–D379, 2017.
25. P.-Y. Chen, C.-C. Tu, P. Ting, et al. Identifying influential links for event propagation on twitter: A network of networks approach. *IEEE Transactions on Signal and Information Processing over Networks*, 5(1):139–151, 2018.
26. R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.
27. L. Chindelevitch, C.-S. Liao, and B. Berger. Local optimization for global alignment of protein interaction networks. In *Biocomputing 2010*, pages 123–132. World Scientific, 2010.
28. X. Chu, X. Fan, D. Yao, Z. Zhu, J. Huang, and J. Bi. Cross-network embedding for multi-network alignment. In *The World Wide Web Conference*, pages 273–284, 2019.
29. G. Ciriello, M. Mina, P. H. Guzzi, M. Cannataro, and C. Guerra. AlignNemo: a local network alignment method to integrate homology and topology. *PLOS ONE*, 7(6):e38107, 2012.
30. C. Clark and J. Kalita. A comparison of algorithms for the pairwise alignment of biological networks. *Bioinformatics*, 30(16):2351–2359, 2014.
31. A. Clauset, C. Moore, and M. E. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.
32. S. Collins, P. Kemmeren, X. Zhao, J. Greenblatt, F. Spencer, F. Holstege, J. Weissman, and N. Krogan. Toward a comprehensive atlas of the physical interactome of *saccharomyces cerevisiae*. *Molecular Cell Proteomics*, 6(3):439–450, Mar. 2007.
33. A. I. M. Consortium, M. Dreze, A.-R. Carvunis, B. Charlotiaux, M. Galli, S. J. Pevzner, M. Tasan, Y.-Y. Ahn, P. Balumuri, A.-L. Barabási, V. Bautista, P. Braun, D. Byrdsong, H. Chen, J. D. Chesnut, M. E. Cusick, J. L. Dangl, C. de los Reyes, A. Dricot, M. Duarte, J. R. Ecker, C. Fan, L. Gai, F. Gebreab, G. Ghoshal, P. Gilles, B. J. Gutierrez, T. Hao, D. E. Hill, C. J. Kim, R. C. Kim, C. Lurin, A. MacWilliams, U. Matrubutham, T. Milenkovic, J. Mirchandani, D. Monachello, J. Moore, M. S. Mukhtar, E. Olivares, S. Patnaik, M. M. Poulin, N. Przulj, R. Quan, S. Rabello, G. Ramaswamy, P. Reichert, E. A. Rietman, T. Rolland, V. Romero, F. P. Roth, B. Santhanam, R. J. Schmitz, P. Shinn, W. Spooner, J. Stein, G. M. Swamilingiah, S. Tam, J. Vandenhoute, M. Vidal, S. Waaijers, D. Ware, E. M. Weiner, S. Wu, and J. Yazaki. Evidence for Network Evolution in an Arabidopsis Interactome Map. *Science*, 333(6042):601–607, 2011.

34. U. Consortium. UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Research*, 47(D1):D506–D515, 2019.
35. S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158. ACM, 1971.
36. J. Crawford, Y. Sun, and T. Milenković. Fair evaluation of global network aligners. *Algorithms for Molecular Biology*, 10(1):19, 2015.
37. P. Cui, X. Wang, J. Pei, and W. Zhu. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 31(5):833–852, 2018.
38. J. P. de Magalhães. Aging research in the post-genome era: New technologies for an old problem. *Redox Metabolism and Longevity Relationships in Animals and Plants*. Taylor and Francis, New York and Abingdon, pages 99–115, 2009.
39. J. Dohrmann, J. Puchin, and R. Singh. Global multiple protein-protein interaction network alignment by combining pairwise network alignments. In *BMC Bioinformatics*, volume 16, page S11. Springer, 2015.
40. S. Dong, H. Wang, A. Mostafizi, and X. Song. A network-of-networks percolation analysis of cascading failures in spatially co-located road-sewer infrastructure networks. *Physica A: Statistical Mechanics and Its Applications*, 538: 122971, 2020.
41. Y. Dong, N. V. Chawla, and A. Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 135–144, 2017.
42. B. Du and H. Tong. Mrmine: Multi-resolution multi-network embedding. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 479–488, 2019.
43. B. DuSell and D. Chiang. Learning hierarchical structures with differentiable nondeterministic stacks. *arXiv preprint arXiv:2109.01982*, 2021.
44. M. El-Kebir, J. Heringa, and G. Klau. Natalie 2.0: sparse global network alignment as a special case of quadratic assignment. *Algorithms*, 8(4):1035–1051, 2015.
45. R. Elheshia, A. Sarkar, C. Boucher, and T. Kahveci. Identification of co-evolving temporal networks. *BMC Genomics*, 20(6):434, 2019.
46. K. W. Ellens, N. Christian, C. Singh, V. P. Satagopam, P. May, and C. L. Linster. Confronting the catalytic dark matter encoded by sequenced genomes. *Nucleic Acids Research*, 45(20):11495–11514, 2017.

47. A. Elmsallati, C. Clark, and J. Kalita. Global alignment of protein-protein interaction networks: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 13(4):689–705, 2016.
48. F. Emmert-Streib, M. Dehmer, and Y. Shi. Fifty years of graph matching, network alignment and network comparison. *Information Sciences*, 346:180–197, 2016.
49. P. Erdős and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, 5(1):17–60, 1960.
50. B. S. Everitt, S. Landau, and M. Leese. *Cluster Analysis*. Wiley, 2001.
51. F. E. Faisal and T. Milenković. Dynamic networks reveal key players in aging. *Bioinformatics*, 30(12):1721–1729, 2014.
52. F. E. Faisal, H. Zhao, and T. Milenković. Global network alignment in the context of aging. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 12(1):40–52, 2014.
53. F. E. Faisal, L. Meng, J. Crawford, and T. Milenković. The post-genomic era of biological network alignment. *EURASIP Journal on Bioinformatics and Systems Biology*, 2015(1):3, 2015.
54. F. E. Faisal, H. Zhao, and T. Milenković. Global network alignment in the context of aging. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 12(1):40–52, 2015.
55. F. E. Faisal, K. Newaz, J. L. Chaney, et al. GRAFENE: Graphlet-based alignment-free network approach integrates 3d structural and sequence (residue order) data to improve protein structural comparison. *Scientific Reports*, 7(1): 1–15, 2017.
56. E. B. Falk and D. S. Bassett. Brain and social networks: fundamental building blocks of human experience. *Trends in Cognitive Sciences*, 21(9):674–690, 2017.
57. J. Fan, A. Cannistra, I. Fried, T. Lim, T. Schaffner, M. Crovella, B. Hescott, and M. D. Leiserson. Functional protein representations from biological networks enable diverse cross-species inference. *Nucleic Acids Research*, 47(9):e51–e51, 2019.
58. J. Flannick, A. Novak, B. S. Srinivasan, H. H. McAdams, and S. Batzoglou. Graemlin: general and robust alignment of multiple large interaction networks. *Genome Research*, 16(9):1169–1181, 2006.
59. J. Flannick, A. Novak, C. Do, B. Srinivasan, and S. Batzoglou. Automatic parameter learning for multiple network alignment. In *Research in Computational Molecular Biology*, pages 214–231. Springer, 2008.

60. P. Gaudet, N. Škunca, J. C. Hu, and C. Dessimoz. Primer on the gene ontology. In *The Gene Ontology Handbook*, pages 97–109. Humana Press, New York, NY, 2017.
61. V. Gligorijević and N. Pržulj. Methods for biological data integration: perspectives and challenges. *Journal of the Royal Society Interface*, 12(112):20150571, 2015.
62. V. Gligorijević, N. Malod-Dognin, and N. Pržulj. Fuse: multiple network alignment via data fusion. *Bioinformatics*, 32(8):1195–1203, 2016.
63. V. Gligorijević, P. D. Renfrew, T. Kosciolk, J. K. Leman, D. Berenberg, T. Vatanen, C. Chandler, B. C. Taylor, I. M. Fisk, H. Vlamakis, et al. Structure-based protein function prediction using graph convolutional networks. *Nature Communications*, 12(1):1–14, 2021.
64. P. Goyal and E. Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.
65. D. Grattarola and C. Alippi. Graph neural networks in tensorflow and keras with spektral. *arXiv preprint arXiv:2006.12138*, 2020.
66. A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864. ACM, 2016.
67. S. Gu and T. Milenković. Graphlets versus node2vec and struc2vec in the task of network alignment. *arXiv preprint arXiv:1805.04222*, 2018.
68. S. Gu and T. Milenković. Data-driven network alignment. *PloS one*, 15(7):e0234978, 2020.
69. S. Gu and T. Milenković. Data-driven biological network alignment that uses topological, sequence, and functional information. *BMC bioinformatics*, 22(1):1–24, 2021.
70. S. Gu, J. Johnson, F. E. Faisal, and T. Milenković. From homogeneous to heterogeneous network alignment via colored graphlets. *Scientific Reports*, 8(1):12524, 2018.
71. S. Gu, M. Jiang, P. H. Guzzi, and T. Milenkovic. Modeling multi-scale data via a network of networks. *arXiv preprint arXiv:2105.12226*, 2021.
72. P. H. Guzzi and T. Milenković. Survey of local and global biological network alignment: the need to reconcile the two sides of the same coin. *Briefings in Bioinformatics*, 19(3):472–481, 2017.
73. S. Hashemifar and J. Xu. HubAlign: an accurate and efficient method for global alignment of protein–protein interaction networks. *Bioinformatics*, 30(17):i438–i444, 2014.

74. S. Hashemifar, Q. Huang, and J. Xu. Joint alignment of multiple protein–protein interaction networks via convex optimization. *Journal of Computational Biology*, 23(11):903–911, 2016.
75. S. Hashemifar, J. Ma, H. Naveed, S. Canzar, and J. Xu. ModuleAlign: module-based global alignment of protein-protein interaction networks. *Bioinformatics*, 32(17):i658, 2016.
76. W. B. Hayes and N. Mamano. SANA NetGO: a combinatorial approach to using Gene Ontology (GO) terms to score network alignments. *Bioinformatics*, 34(8):1345–1352, 2017.
77. M. Heimann, H. Shen, T. Safavi, and D. Koutra. Regal: Representation learning-based graph alignment. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 117–126. ACM, 2018.
78. J. Hibshman, S. Sikdar, and T. Weninger. Towards interpretable graph modeling with vertex replacement grammars. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 770–779. IEEE, 2019.
79. T. Hočevár and J. Demšar. A combinatorial approach to graphlet counting. *Bioinformatics*, 30(4):559–565, 2014.
80. J. Hu, B. Kehr, and K. Reinert. NetCoffee: a fast and accurate global alignment approach to identify functionally conserved proteins in multiple networks. *Bioinformatics*, 30(4):540–548, 2013.
81. J. Hu, J. He, Y. Gao, Y. Zheng, and X. Shang. NetCoffee2: A Novel Global Alignment Algorithm for Multiple PPI Networks Based on Graph Feature Vectors. In *International Conference on Intelligent Computing*, pages 241–246. Springer, 2018.
82. Y. Hulovatyy, R. W. Solava, and T. Milenković. Revealing missing parts of the interactome via link prediction. *PLOS ONE*, 9(3):e90073, 2014.
83. R. Ibragimov, M. Malek, and J. Baumbach. GEDEVO: An evolutionary graph edit distance algorithm for biological network alignment. In *GCB*, pages 68–79, 2013.
84. R. Ibragimov, M. Malek, J. Guo, and J. Baumbach. Multiple graph edit distance - simultaneous topological alignment of multiple protein-protein interaction networks with an evolutionary algorithm. In *Proc. of Annual Conf. on Genetic and Evolutionary Computation*, pages 277–284, 2014.
85. H. Jeong, X. Qian, and B.-J. Yoon. Effective comparative analysis of protein-protein interaction networks by measuring the steady-state network flow using a Markov model. In *BMC Bioinformatics*, volume 17, page 395. BioMed Central, 2016.

86. M. Kalaev, M. Smoot, T. Ideker, and R. Sharan. NetworkBLAST: comparative analysis of protein networks. *Bioinformatics*, 24(4):594–596, 2008.
87. K. Kalecky and Y.-R. Cho. PrimAlign: PageRank-inspired Markovian alignment for large biological networks. *Bioinformatics*, 34(13):i537–i546, 2018.
88. B. P. Kelley, R. Sharan, R. M. Karp, T. Sittler, D. E. Root, B. R. Stockwell, and T. Ideker. Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *Proceedings of the National Academy of Sciences*, 100(20):11394–11399, 2003.
89. B. P. Kelley, B. Yuan, F. Lewitter, R. Sharan, B. R. Stockwell, and T. Ideker. PathBLAST: a tool for alignment of protein interaction networks. *Nucleic Acids Research*, 32(suppl\_2):W83–W88, 2004.
90. T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
91. G. W. Klau. A new graph-based method for pairwise global network alignment. *BMC Bioinformatics*, 10(1):S59, 2009.
92. J. Klicpera, S. Weißenberger, and S. Günnemann. Diffusion improves graph learning. In *33rd Conference on Neural Information Processing Systems*, 2019.
93. M. Kotlyar, C. Pastrello, Z. Malik, and I. Jurisica. IID 2018 update: context-specific physical protein–protein interactions in human, model organisms and domesticated species. *Nucleic Acids Research*, 47(D1):D581–D589, 2019.
94. M. Koyutürk, Y. Kim, U. Topkara, S. Subramaniam, W. Szpankowski, and A. Grama. Pairwise alignment of protein interaction networks. *Journal of Computational Biology*, 13(2):182–199, 2006.
95. O. Kuchaiev and N. Pržulj. Integrative network alignment reveals large regions of global network similarity in yeast and human. *Bioinformatics*, 27(10):1390–1396, 2011.
96. O. Kuchaiev, T. Milenković, V. Memišević, W. Hayes, and N. Pržulj. Topological network alignment uncovers biological function and phylogeny. *Journal of the Royal Society Interface*, 7(50):1341–1354, 2010.
97. M. Li, Q. Zhang, and Y. Deng. Evidential identification of influential nodes in network of networks. *Chaos, Solitons & Fractals*, 117:283–296, 2018.
98. Z. Liang, M. Xu, M. Teng, and L. Niu. NetAlign: a web-based tool for comparison of protein interaction networks. *Bioinformatics*, 22(17):2175–2177, 2006.
99. C.-S. Liao, K. Lu, M. Baym, R. Singh, and B. Berger. IsoRankN: spectral methods for global alignment of multiple protein networks. *Bioinformatics*, 25(12):i253–i258, 2009.

100. C. W. Lynn and D. S. Bassett. Compressibility of complex networks. *arXiv preprint arXiv:2011.08994*, 2020.
101. A. Mali, A. Ororbia, D. Kifer, and L. Giles. Recognizing long grammatical sequences using recurrent networks augmented with an external differentiable stack. In *International Conference on Grammatical Inference*, pages 130–153. PMLR, 2021.
102. N. Malod-Dognin and N. Pržulj. L-GRAAL: Lagrangian graphlet-based network aligner. *Bioinformatics*, 31(13):2182–2189, 2015.
103. N. Mamano and W. B. Hayes. SANA: simulated annealing far outperforms many other search algorithms for biological network alignment. *Bioinformatics*, 33(14):2156–2164, 02 2017. ISSN 1367-4803. doi: 10.1093/bioinformatics/btx090. URL <https://dx.doi.org/10.1093/bioinformatics/btx090>.
104. D. Marcus and Y. Shavitt. Rage—a rapid graphlet enumerator for large networks. *Computer Networks*, 56(2):810–819, 2012.
105. L. R. Matthews, P. Vaglio, J. Reboul, H. Ge, B. P. Davis, J. Garrels, S. Vincent, and M. Vidal. Identification of potential interaction networks using sequence-based searches for conserved protein-protein interactions or “interologs”. *Genome Research*, 11(12):2120–2126, 2001.
106. G. K. Mazandu and N. J. Mulder. DaGO-fun: tool for gene ontology-based functional analysis using term information content measures. *BMC Bioinformatics*, 14(1):284, 2013.
107. V. Memišević and N. Pržulj. C-GRAAL: Common-neighbors-based global GRAPh ALignment of biological networks. *Integrative Biology*, 4(7):734–743, 2012.
108. V. Memišević, T. Milenković, and N. Pržulj. Complementarity of network and sequence information in homologous proteins. *Journal of integrative bioinformatics*, 7(3):275–289, 2010.
109. V. Memišević, T. Milenković, N., and N. Pržulj. Complementarity of network and sequence structure in homologous proteins. *Journal of Integrative Bioinformatics*, 9:121–137, 2010.
110. L. Meng, A. Striegel, and T. Milenković. Local versus global biological network alignment. *Bioinformatics*, 32(20):3155–3164, 2016.
111. M. Milano, P. H. Guzzi, and M. Cannataro. HetNetAligner: a novel algorithm for local alignment of heterogeneous biological networks. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 598–599. ACM, 2018.



112. T. Milenković and N. Pržulj. Uncovering biological network function via graphlet degree signatures. *Cancer Informatics*, 6:CIN–S680, 2008.
113. T. Milenković, J. Lai, and N. Pržulj. GraphCrunch: a tool for large network analyses. *BMC Bioinformatics*, 9(1):70, 2008.
114. T. Milenković, W. L. Ng, W. Hayes, and N. Pržulj. Optimal network alignment with graphlet degree vectors. *Cancer Informatics*, 9:121, 2010.
115. T. Milenković, H. Zhao, and F. E. Faisal. Global network alignment in the context of aging. In *Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics*, page 23. ACM, 2013.
116. M. Mina and P. H. Guzzi. AlignMCL: Comparative analysis of protein interaction networks through Markov clustering. In *2012 IEEE International Conference on Bioinformatics and Biomedicine Workshops*, pages 174–181. IEEE, 2012.
117. M. Mina and P. H. Guzzi. Improving the robustness of local network alignment: design and extensive assessment of a markov clustering-based approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 11(3):561–572, 2014.
118. F. Monti, K. Otness, and M. M. Bronstein. Motifnet: a motif-based graph convolutional network for directed graphs. In *2018 IEEE Data Science Workshop (DSW)*, pages 225–228. IEEE, 2018.
119. F. Morone, K. Roth, B. Min, et al. Model of brain activation predicts the neural collective influence map of the brain. *Proceedings of the National Academy of Sciences*, 114(15):3849–3854, 2017.
120. R. Mugur, P. Smitha, and M. Pallavi. Predicting the Functions of Unknown Protein by Analyzing Known Protein Interaction: A survey. *Biomedical and Pharmacology Journal*, 11(3):1707–1715, 2018.
121. J. R. Munkres. *Elements of Algebraic Topology*. CRC press, 2018.
122. A. Narayanan, E. Shi, and B. I. Rubinstein. Link prediction by de-anonymization: How we won the kaggle social network challenge. In *The 2011 International Joint Conference on Neural Networks (IJCNN)*, pages 1825–1834. IEEE, 2011.
123. H. Nassar and D. F. Gleich. Multimodal network alignment. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 615–623. SIAM, 2017.
124. W. Nelson, M. Zitnik, B. Wang, J. Leskovec, A. Goldenberg, and R. Sharan. To embed or not: network embedding as a paradigm in computational biology. *Frontiers in Genetics*, 10, 2019.

125. K. Newaz, M. Ghalehnovi, A. Rahnama, P. J. Antsaklis, and T. Milenković. Network-based protein structural classification. *Royal Society Open Science*, 7(6):191461, 2018.
126. B. Neyshabur, A. Khadem, S. Hashemifar, and S. S. Arab. NETAL: a new graph-based method for global alignment of protein–protein interaction networks. *Bioinformatics*, 29(13):1654–1662, 2013.
127. J. Ni, M. Koyuturk, H. Tong, J. Haines, R. Xu, and X. Zhang. Disease gene prioritization by integrating tissue-specific molecular networks using a robust multi-network model. *BMC Bioinformatics*, 17(1):453, 2016.
128. G. Nikolentzos, P. Meladianos, and M. Vazirgiannis. Matching node embeddings for graph similarity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.
129. C. Parkinson, A. M. Kleinbaum, and T. Wheatley. Similar neural responses predict friendship. *Nature Communications*, 9(1):1–14, 2018.
130. R. Patro and C. Kingsford. Global network alignment using multiscale spectral signatures. *Bioinformatics*, 28(23):3105–3114, 2012.
131. PDB. Pdb101: Learn: Guide to understanding pdb data: Resolution, 2020. URL <http://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/resolution>.
132. W. Peng, J. Wang, J. Cai, et al. Improving protein function prediction using domain and protein complexes in PPI networks. *BMC Systems Biology*, 8(1):35, 2014.
133. M. Penrose. *Random geometric graphs*, volume 5. Oxford University Press, 2003.
134. M. G. Perich and K. Rajan. Rethinking brain-wide interactions through multi-region ‘network of networks’ models. *Current Opinion in Neurobiology*, 65:146–151, 2020.
135. H. T. Phan and M. J. Sternberg. PINALOG: a novel approach to align protein interaction networks—implications for complex detection and function prediction. *Bioinformatics*, 28(9):1239–1245, 2012.
136. N. Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.
137. M. Rahman, M. A. Bhuiyan, and M. Al Hasan. Graft: An efficient graphlet counting method for large graph analysis. *IEEE Transactions on Knowledge and Data Engineering*, 26(10):2466–2478, 2014.

138. E. Rossi, F. Frasca, B. Chamberlain, et al. SIGN: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*, 2020.
139. R. A. Rossi, N. K. Ahmed, A. Carranza, D. Arbour, A. Rao, S. Kim, and E. Koh. Heterogeneous network motifs. *arXiv preprint arXiv:1901.10026*, 2019.
140. B. Rost. Twilight zone of protein sequence alignments. *Protein Engineering*, 12(2):85–94, 1999.
141. K. Roth, F. Morone, B. Min, and H. A. Makse. Emergence of robustness in networks of networks. *Physical Review E*, 95(6):062308, 2017.
142. S. M. E. Sahraeian and B.-J. Yoon. SMETANA: accurate and scalable algorithm for probabilistic alignment of large-scale biological networks. *PLOS ONE*, 8(7):e67995, 2013.
143. V. Saraph and T. Milenković. MAGNA: maximizing accuracy in global network alignment. *Bioinformatics*, 30(20):2931–2940, 2014.
144. B.-S. Seah, S. S. Bhowmick, and C. F. Dewey Jr. DualAligner: a dual alignment-based strategy to align protein interaction networks. *Bioinformatics*, 30(18):2619–2626, 2014.
145. J. Shang, M. Qu, J. Liu, L. M. Kaplan, J. Han, and J. Peng. Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. *arXiv preprint arXiv:1610.09769*, 2016.
146. R. Sharan and T. Ideker. Modeling cellular machinery through biological network comparison. *Nature Biotechnology*, 24(4), 2006.
147. R. Sharan, S. Suthram, R. M. Kelley, T. Kuhn, S. McCuine, P. Uetz, T. Sittler, R. M. Karp, and T. Ideker. Conserved patterns of protein interaction in multiple species. *Proceedings of the National Academy of Sciences of the United States of America*, 102(6):1974–1979, 2005.
148. O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.
149. A. Shehu, D. Barbará, and K. Molloy. A survey of computational methods for protein function prediction. In *Big Data Analytics in Genomics*, pages 225–298. Springer, 2016.
150. S. Sikdar, J. Hibshman, and T. Weninger. Modeling graphs with vertex replacement grammars. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 558–567. IEEE, 2019.
151. J. E. Simpson, P. G. Ince, P. J. Shaw, P. R. Heath, R. Raman, C. J. Garwood, C. Gelsthorpe, L. Baxter, G. Forster, F. E. Matthews, et al. Microarray analysis of the astrocyte transcriptome in the aging brain: relationship to Alzheimer’s pathology and APOE genotype. *Neurobiology of Aging*, 32(10):1795–1807, 2011.

152. O. Singh, K. Sawariya, and P. Aparoy. Graphlet signature-based scoring method to estimate protein–ligand binding affinity. *Royal Society Open Science*, 1(4):140306, 2014.
153. R. Singh, J. Xu, and B. Berger. Pairwise global alignment of protein interaction networks by matching neighborhood topology. In *Annual International Conference on Research in Computational Molecular Biology*, pages 16–31. Springer, 2007.
154. R. Singh, J. Xu, and B. Berger. 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.
155. R. W. Solava, R. P. Michaels, and T. Milenković. Graphlet-based edge clustering reveals pathogen-interacting proteins. *Bioinformatics*, 28(18):i480–i486, 2012.
156. C. Stark, B.-J. Breitkreutz, T. Reguly, et al. BioGRID: a general repository for interaction datasets. *Nucleic Acids Research*, 34(suppl\_1):D535–D539, 2006.
157. Y. Sun, A. K. Wong, and M. S. Kamel. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04):687–719, 2009.
158. Y. Sun, J. Crawford, J. Tang, and T. Milenković. Simultaneous optimization of both node and edge conservation in network alignment via WAVE. In *International Workshop on Algorithms in Bioinformatics*, pages 16–39. Springer, 2015.
159. The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.
160. E. G. Tuncay and T. Can. SUMONA: A supervised method for optimizing network alignment. *Comput. Biol. Chem.*, 63:41–51, 2016.
161. V. Vacic, L. M. Iakoucheva, S. Lonardi, and P. Radivojac. Graphlet kernels for prediction of functional residues in protein structures. *Journal of Computational Biology*, 17(1):55–72, 2010.
162. V. Vijayan and T. Milenković. Aligning dynamic networks with DynaWAVE. *Bioinformatics*, page btx841, 2017.
163. V. Vijayan and T. Milenković. Multiple network alignment via multi-MAGNA++. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, PP(99), Aug 2017. doi: 10.1109/TCBB.2017.2740381.
164. V. Vijayan, V. Saraph, and T. Milenković. MAGNA++: Maximizing accuracy in global network alignment via both node and edge conservation. *Bioinformatics*, 31(14):2409–2411, 2015.

165. V. Vijayan, D. Critchlow, and T. Milenković. Alignment of dynamic networks. *Bioinformatics*, 33(14):i180–i189, 2017.
166. V. Vijayan, S. Gu, E. Krebs, L. Meng, and T. Milenković. Pairwise versus multiple global network alignment. *IEEE Access*, 2020.
167. N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11:410–420, 2007.
168. H. Wang, D. Lian, Y. Zhang, et al. GoGNN: Graph of graphs neural network for predicting structured entity interactions. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 2020.
169. X.-D. Wang, J.-L. Huang, L. Yang, D.-Q. Wei, Y.-X. Qi, and Z.-L. Jiang. Identification of human disease genes from interactome network using graphlet interaction. *PLOS ONE*, 9(1):e86142, 2014.
170. F. Wu, T. Zhang, H. de Souza, et al. Simplifying graph convolutional networks. *Proceedings of Machine Learning Research*, 2019.
171. X. Wu, Q. Liu, and R. Jiang. Align human interactome with phenome to identify causative genes and networks underlying disease families. *Bioinformatics*, 25(1): 98–104, 2009. doi: 10.1093/bioinformatics/btn593.
172. J. Xu, T. L. Wickramaratne, and N. V. Chawla. Representing higher-order dependencies in networks. *Science Advances*, 2(5):e1600028, 2016.
173. Ö. N. Yaveroglu, N. Malod-Dognin, D. Davis, et al. Revealing the hidden language of complex networks. *Scientific Reports*, 4:4547, 2014.
174. Ö. N. Yaveroglu, T. Milenković, and N. Pržulj. Proper evaluation of alignment-free network comparison methods. *Bioinformatics*, 31(16):2697–2704, 2015.
175. J. Ye, S. McGinnis, and T. L. Madden. BLAST: improvements for better sequence analysis. *Nucleic Acids Research*, 34(Web Server issue):W6–W9, July 2006.
176. R. Ying, J. You, C. Morris, et al. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 4805–4815, 2018.
177. R. You, Z. Zhang, Y. Xiong, F. Sun, H. Mamitsuka, and S. Zhu. Golabeler: improving sequence-based large-scale protein function prediction by learning to rank. *Bioinformatics*, 34(14):2465–2473, 2018.
178. M. Zaslavskiy, F. Bach, and J.-P. Vert. Global alignment of protein–protein interaction networks by graph matching methods. *Bioinformatics*, 25(12):i259–1267, 2009.

179. F. Zhang, H. Song, M. Zeng, et al. DeepFunc: a deep learning framework for accurate prediction of protein functions from protein sequences and interactions. *Proteomics*, 19(12):1900019, 2019.
180. J. Zhang, B. Chen, X. Wang, H. Chen, C. Li, F. Jin, G. Song, and Y. Zhang. MEgo2Vec: Embedding Matched Ego Networks for User Alignment Across Social Networks. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 327–336. ACM, 2018.
181. S. Zhang, H. Chen, K. Liu, and Z. Sun. Inferring protein function by domain context similarities in protein-protein interaction networks. *BMC Bioinformatics*, 10(1):1–6, 2009.
182. Y. Zhang and J. Skolnick. TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic Acids Research*, 33(7):2302–2309, 2005.
183. N. Zhou, Y. Jiang, T. R. Bergquist, A. J. Lee, B. Z. Kacsoh, A. W. Crocker, K. A. Lewis, G. Georghiou, H. N. Nguyen, M. N. Hamid, et al. The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. *Genome Biology*, 20(1):1–23, 2019.