Accurate and Efficient Gene Function Prediction using a Multi-Bacterial Network

Jeffrey N. Law[1], Shiv D. Kale[2], and T. M. Murali *[3]

[1]Genetics, Bioinformatics, and Computational Biology Ph.D. program, Virginia Tech, Blacksburg VA
[2]Fralin Life Sciences Institute, Virginia Tech, Blacksburg VA
[3]Department of Computer Science, Virginia Tech, Blacksburg VA

Summary

The rapid rise in newly sequenced genomes requires the development of computational methods to supplement experimental functional annotations. The challenge that arises is to develop methods for gene function prediction that integrate information for multiple species while also operating on a genomewide scale. We develop a label propagation algorithm called FastSinkSource and apply it to a sequence similarity network integrated with species-specific heterogeneous data for 19 pathogenic bacterial species. By using mathematically-provable bounds on the rate of progress of FastSinkSource during power iteration, we decrease the running time by a factor of 100 or more without sacrificing prediction accuracy. To demonstrate scalability, we expand to a 73-million edge network across 200 bacterial species while maintaining accuracy and efficiency improvements. Our results point to the feasibility and promise of multi-species, genomewide gene function prediction, especially as more experimental data and annotations become available for a diverse variety of organisms.

Introduction

The number of fully sequenced prokaryotic genomes is increasing dramatically due to diminishing sequencing costs [1]. The overwhelming majority of the genes in these genomes have not been studied experimentally. For instance, fewer than 0.01% (about 10,000 out of 104 million) of prokaryotic genes in the UniProt Knowledgebase [2] have a Gene Ontology (GO) [3] annotation with an experimental evidence code. To address this gap, several computational methods have been developed to associate molecular functions and biological processes with genes lacking experimental annotations [4–11]. These predicted associations assist in prioritizing follow-up experiments to determine the biological functions performed by gene products [12]. Network-based techniques are among the most accurate and widely-used approaches for gene function prediction [4–6, 9].

Existing approaches for gene function prediction are limited in one of two ways. Methods that apply on a genomewide scale, i.e., predict functions for all genes, usually operate on individual organisms [4, 5, 13]. On the other hand, techniques that integrate information across multiple species usually target one gene or a small group of related genes at a time [8, 11, 14]. Given the large number of sequenced genomes that are available, there is a need to develop algorithms that can predict the functions of genes in multiple species simultaneously on a genomewide basis. This problem is challenging because multi-species gene networks can rapidly become orders of magnitude larger than those for a single species, requiring prediction methods to be very efficient while maintaining their accuracy.

To address this challenge, we develop a novel label propagation algorithm called FastSinkSource that can efficiently make genomewide function predictions across multiple organisms. We mathematically prove that the rate of convergence of the node scores computed by FastSinkSource is geometric in the number of

*Corresponding author: murali@cs.vt.edu

iterations. This property enables us to decrease the running time of FastSinkSource by a factor of 100 or more without sacrificing prediction accuracy.

We demonstrate the applicability of FastSinkSource using a leave-one-species-out (LOSO) validation framework that mimics the challenge of making large-scale functional predictions for the genome of a newly-sequenced species where none of its genes have any annotations. We apply this evaluation framework to a sequence similarity network (SSN) of 19 bacterial species integrated with intra-species networks from the STRING database. FastSinkSource maintains accuracy and efficiency improvements even when applied to a large-scale 200-species network with 73 million edges.

Results

Overview of FastSinkSource

Network propagation methods typically use either power iteration or other linear system solvers such as conjugate gradient to compute node prediction scores [13, 15–17]. The algorithms typically iterate until the maximum difference of node scores from one iteration to the next is $\leq \epsilon$, where ϵ is a pre-defined parameter such as 1×10^{-5} [13, 15–17]. In our use of SinkSource [15], a precursor of FastSinkSource, we found that hundreds of iterations were necessary for most GO terms before reaching $\epsilon = 1 \times 10^{-4}$ (see the next section). This trend motivated us to seek alternative methods for establishing convergence.

Given a GO term, our goal is to compute node scores in order to rank nodes by how likely they are to be associated with that term. Therefore, a definitive criterion would be to terminate when the relative ordering of all nodes by their scores could no longer change, after which point additional iterations would give no added benefit. In this work, we prove that FastSinkSource has the following key property: for every node, the difference between its score at convergence and its score after i iterations decreases geometrically with i (see Appendix A.1). The decrease depends on a parameter $0 < \alpha < 1$, which is akin to the teleportation probability in a random walk with restart. This property offers two significant benefits: (a) It inspires a strategy for checking convergence based on when we have correctly ranked all the nodes. This new approach has the additional benefit that it allows us to specify a subset of nodes whose rankings we seek to fix, e.g., during cross validation. (b) If we stop after i iterations, we have an assessment of how much we have under-estimated every node's score and an upper bound on the number of node pairs that may be ranked incorrectly with respect to each other. We describe these benefits in more detail and how we use them in the section "The FastSinkSource algorithm".

Assessing the trade-off between accuracy and speed

By design, FastSinkSource converges faster, i.e., with a smaller number of iterations as we decrease the parameter α (see proof in Appendix A.1). Additionally, for a fixed α, FastSinkSource terminates faster as we decrease the number of iterations. Therefore, we sought to test the trade-off between accuracy and speed by varying either α or the number of iterations before we stopped FastSinkSource.

For this analysis, we used a sequence similarity network for 19 pathogenic species of bacteria integrated with the interaction networks for each organism from the STRING database (see "Datasets"). The combined network contains a total of 75K nodes and 2.8M edges (Table S2). In these evaluations, we used annotations with experimental evidence codes only, which we denote as EXPC annotations. We focus our presentation on the results for the Biological Process (BP) GO terms. We briefly mention results for Molecular Function (MF) categories.

We ran FastSinkSource for eight different values of α, namely, 1, 0.99, 0.95, 0.9, 0.8, 0.7, 0.6, and 0.5. For each value of α, we performed a LOSO evaluation, i.e., we left out the annotations of all genes for each species in turn, and assessed how well we could predict them using only the annotations of the other 18 organisms. We limited our analyses to GO terms that had at least 50 annotations summed over all 19 organisms, at least 10 annotations in a given "left-out" species, and at least 10 annotations in the other 18 species. In total, 285 BP and 104 MF terms, respectively, satisfied these criteria. For values of $\alpha < 1$, we executed FastSinkSource to convergence using our new test. Specifically, we used as many iterations as needed to fix the relative rankings of the subset of nodes that were either positive or negative examples in the left-out species. Note that we required knowledge only of whether or not a gene was in this subset, i.e.,

we did not know which gene was a positive and which was a negative example. For $\alpha = 1.0$, our new strategy for testing convergence is not applicable since we can only prove a trivial bound of unity between a node's final score and its score after i iterations. Hence, we chose to stop FastSinkSource after 1,000 iterations. We selected this value since we observed that to reach a reasonable value of $\epsilon = 1 \times 10^{-4}$, FastSinkSource with $\alpha = 1.0$ required a median of 960 iterations, with a median absolute deviation (MAD) of 503.5.

We explored the effect of varying α on the accuracy of FastSinkSource by comparing the resulting distributions of $F_{\max}$ values over the BP GO terms we tested. The median $F_{\max}$ value did decrease gradually as we decreased α (Figure 1a). For $\alpha \geq 0.9$, we observed a statistically indistinguishable difference with $\alpha = 1.0$ (uncorrected rank-sum test p-value > 0.25). We chose to use $\alpha = 0.95$ for subsequent analyses.

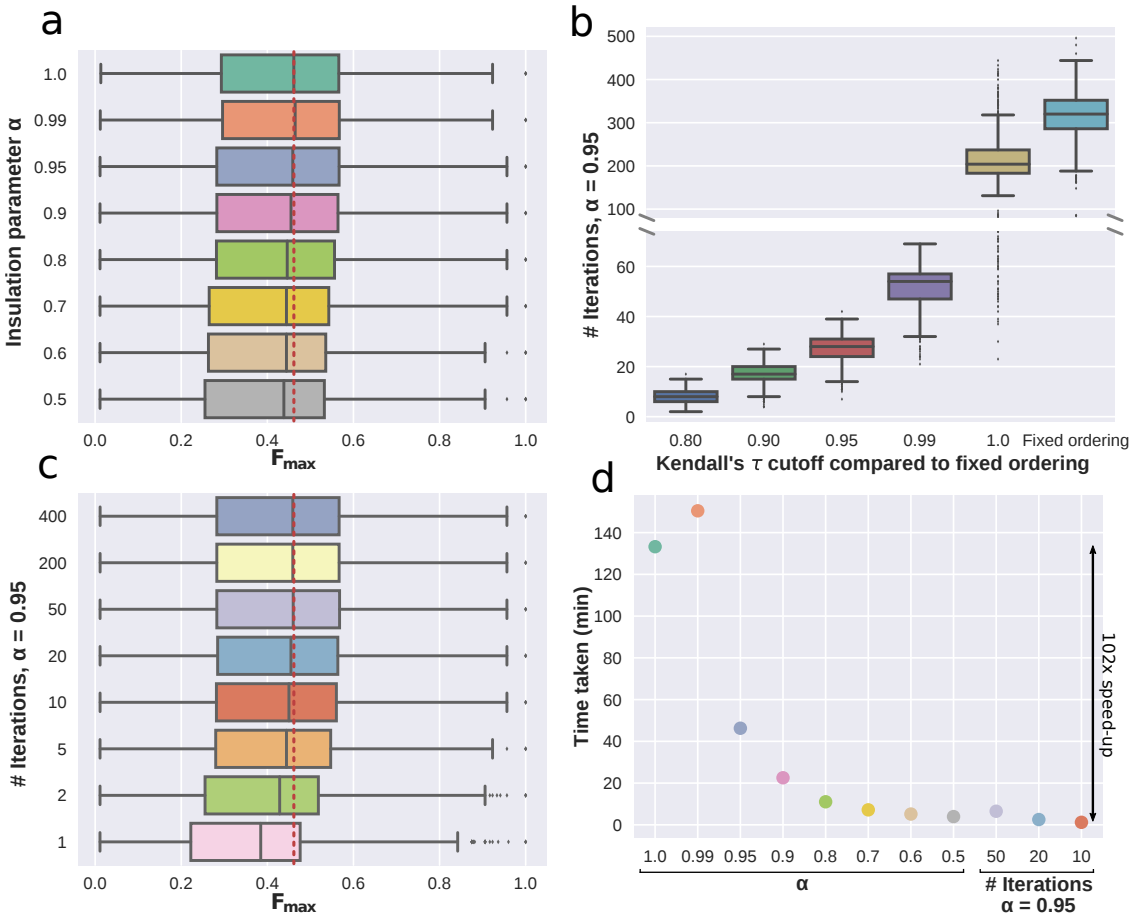

Figure 1: Trade-off between accuracy and speed of FastSinkSource for LOSO validation on the 19 species SSN integrated with STRING networks (a total of 787 species-GO term pairs). (**a**) Variation of $F_{\max}$ distributions with α. The vertical red dotted line represents the median $F_{\max}$ for $\alpha = 1.0$. (**b**) Number of iterations required to fix the rankings of the left-out positive and negative examples, or to reach a specified value of Kendall's τ in comparison to the fixed ranking. (**c**) Variation of $F_{\max}$ distributions with the number of iterations ($\alpha = 0.95$). The vertical red dotted line represents the median $F_{\max}$ for $\alpha = 1.0$ in panel (a). (**d**) Total time taken by FastSinkSource while varying α ($F_{\max}$ values shown in (a)) or the number of iterations with $\alpha = 0.95$ ($F_{\max}$ values shown in (c)). Colors are the same as in (a) and (c).

Next, we considered varying the number of iterations for which we ran FastSinkSource. We first executed FastSinkSource to convergence, i.e., till the node orderings were fixed. We then re-executed FastSinkSource and for each $i > 0$, we compared how similar the node rankings after iteration i were to the fixed ordering. We computed this similarity using Kendall's τ, a measure of rank correlation. Given two different rankings of the nodes, this measure counts the fraction of node pairs that are not inverted, i.e., the fraction of node pairs u and v such that u appears before v in both rankings or vice-versa. Thus, the closer τ is to unity, the

more similar are the two rankings. For each GO term, we recorded the number of iterations needed to reach a Kendall's τ value of 0.80, 0.90, 0.95, 0.99, and 1.0 (Figure 1b). The median number of iterations required to fix the node ordering completely was 320. Interestingly, we observed that FastSinkSource computed this ranking earlier with a median of 204 iterations (Figure 1b, Kendall's $\tau = 1.0$); subsequent iterations did not change the ordering but were necessary to guarantee that it would not be modified by more computations. Even more strikingly, after a median of 8 and 17 iterations, Kendall's τ was already as high as 0.80 and 0.90, respectively. These results suggested that while about 320 iterations may be required to provably fix the rankings of the left-out positive and negative nodes, reducing the number of iterations by a factor of 20 does not have a material impact of the relative orderings of most of the positive and negative node pairs. Therefore, fewer iterations may not be deleterious to the accuracy of FastSinkSource during evaluation.

To test this hypothesis, we fixed $\alpha = 0.95$, ran FastSinkSource for eight different numbers of iterations (400, 200, 50, 20, 10, 5, 2, and 1), and recorded the $F_{\max}$ values after LOSO validation (Figure 1c). The $F_{\max}$ distribution for 400 iterations was statistically indistinguishable from the distribution for each of the next four largest number of iterations (200, 50, 20, and 10, rank-sum test, p-value > 0.25). We concluded that decreasing the number of iterations by as large a factor as 40 (from 400 to 10) did not affect the overall distribution of LOSO performance. The running time improved by a factor of 102 (132.9 minutes for SinkSource with $\alpha = 1.0$ for 1,000 iterations vs. 1.3 minutes for FastSinkSource with $\alpha = 0.95$ for 10 iterations, Figure 1d). We chose to run FastSinkSource with $\alpha = 0.95$ and 10 iterations in all subsequent analyses for BP GO terms.

We repeated the above process for all 104 MF GO terms with at least 50 annotations summed over all 19 organisms (total of 271 species-GO term pairs). We found similar results to those for BP with no statistically distinguishable difference for $\alpha \geq 0.9$ compared to $\alpha = 1.0$ nor for 400 iterations compared to 200, 50, 20 and 10 iterations (uncorrected p-value > 0.25, Figure S2a and c). The Kendall's τ reached values of 0.8 and 0.9 with a median of 5 and 14 iterations, respectively, which was slightly faster than for BP terms. Therefore, we also chose to use $\alpha = 0.95$ and 10 iterations for MF GO terms in subsequent analyses.

Testing the Effect of BLAST E-value Cutoffs

When building the sequence-similarity network, we initially considered using a stringent cutoff on the BLAST E-value and rely on network propagation to make predictions using edges corresponding to high homology. Another option open to us was to include relatively high E-value edges in the network, but limit propagation only to direct neighbors to minimize the potentially deleterious effects of poor homology on prediction quality. Hence, we studied the effect of using different E-value cutoffs to construct the sequence-similarity network. We also compared the accuracy of FastSinkSource against GeneMANIA and BirgRank, two leading network-based gene function prediction methods, and against a baseline method called Local (see "Methods" for brief descriptions of these approaches).

We tested E-value cutoffs of 1×10^{-25}, 1×10^{-15}, 1×10^{-6}, 0.1, 5, 20 and 50. We observed that this parameter had a striking effect on the size of the network (Table S2), with the largest SSN containing over five million edges, over seven times as large as the smallest network. We found that when we raised the E-value cutoff from 1×10^{-25} to 0.1, the median $F_{\max}$ for each method increased by more than 0.1 (comparing Figure 2a to Figure 2b). This difference was statistically significant for each algorithm (e.g., rank-sum test p-value $= 4.9 \times 10^{-11}$ for FastSinkSource). Raising the E-value cutoff past 0.1 did not improve the median $F_{\max}$ further for any of the methods (Figure S3a). In further analyses, we used the SSN with an E-value cutoff of 0.1 and refer to it as the SSN.

Incorporating STRING Networks

Techniques that utilize multiple complementary data sources have been shown to be more accurate than those that use a single data source [6, 7, 18]. Therefore, we integrated the SSN with networks from the STRING database, a widely-used resource for multiple types of high-quality interaction data (such as based on physical binding, co-expression, and co-occurence) that is available for many species [19]. However, as the ranges of edge weights in the SSN and the STRING networks differ (1–180 and 1–1,000, respectively), we needed to carefully balance and combine these disparate networks to achieve optimal prediction performance. To accomplish this integration, we tested two methods: term-by-term weighting described in the original

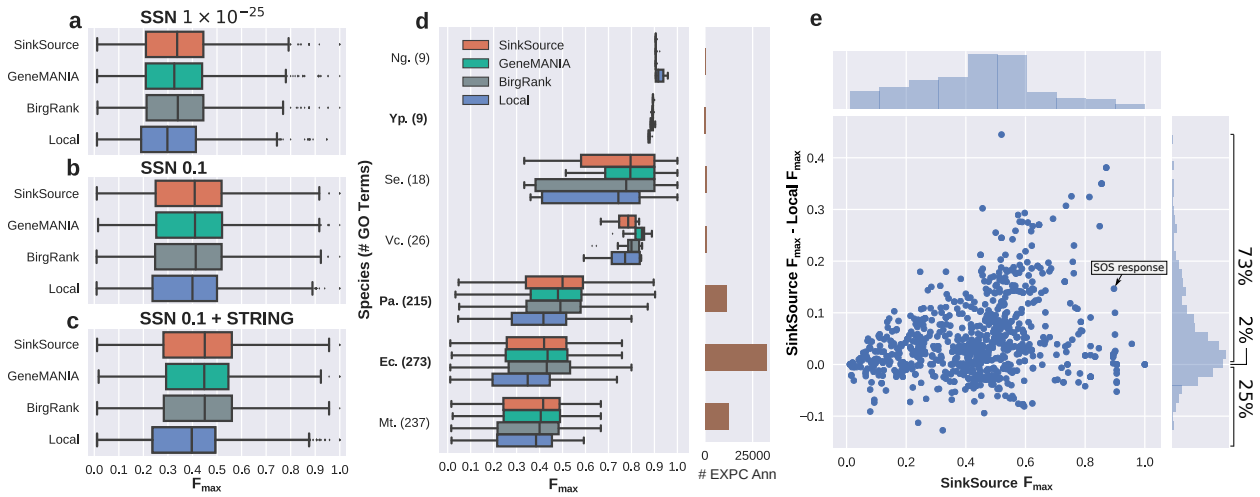

Figure 2: Comparison of $F_{\max}$ results for LOSO evaluation across four algorithms and three networks. (**a**) SSN (E-value $\leq 1 \times 10^{-25}$), (**b**) SSN (E-value ≤ 0.1), (**c**,**d**,**e**) SSN (E-value ≤ 0.1) integrated with STRING, (**d**) LOSO results for individual species, sorted in decreasing order of the median $F_{\max}$ for FastSinkSource. The number of BP GO terms with ≥ 10 annotations appears in parentheses next to each species name. The species name is in bold if the difference between the distributions for FastSinkSource and Local was statistically significant (rank-sum test Bonferroni-corrected p-value < 0.05). The right-hand side shows the number of GO term-annotation pairs with experimental evidence codes for each species. Species names are abbreviated as follows: *Escherichia coli K-12* (Ec), *Mycobacterium tuberculosis* (Mt), *Neisseria gonorrhoeae* (Ng), *Pseudomonas aeruginosa* (Pa), *Salmonella enterica* (Se), *Vibrio cholerae* (Vc), *Yersinia pestis* (Yp). (**e**) Difference in $F_{\max}$ between FastSinkSource and Local by the $F_{\max}$ of FastSinkSource for each GO term. A few of the terms are highlighted.

GeneMANIA publication (GMW; we use this abbreviation to differentiate the weighting method from the network propagation algorithm) [4] and Simultaneous Weights with Specific Negatives (SWSN) [20] which weights networks using multiple related terms simultaneously (see "Datasetes"). Each of these methods computes an optimal weight for each network in order to maximize the total weight of the edges between pairs of positively labeled genes while minimizing the total weight of the edges between genes of opposite labels. For the SWSN method, we used all terms of a given hierarchy as this was shown to perform the best in a previous evaluation [20]. We could use only SWSN for combining weights for BirgRank since this algorithm makes predictions for all GO terms for a given gene simultaneously. In addition to deciding between GMW and SWSN, we also considered how to select a cutoff on the edge weights in the STRING networks. To this end, we used low, medium, high, and very high stringency cutoffs (150, 400, 700, and 900, respectively) on the edge weights of the STRING networks and integrated each of these networks with the SSN. We evaluated these networks using LOSO validation. For each species we left out, we weighted the networks using only annotations of the other 18 organisms we retained.

For BP GO terms, we found that all algorithms achieved the highest median $F_{\max}$ with either the high or very high stringency cutoffs, while the low cutoff decreased the median $F_{\max}$ for each of them (Figure S3c). In contrast, for each algorithm, the performance of GMW did not seem to be affected by the stringency cutoff, with comparable $F_{\max}$ values for SWSN with the two highest cutoffs. However, SWSN was faster than GMW (6.4 minutes vs 14.4 minutes for SWSN and GMW, respectively, for the same evaluation as used in Table 1). An additional benefit of SWSN is that the resulting edge weights do not change from one GO term to another. Therefore, we chose to use SWSN with a stringency cutoff of 700 to integrate the networks. We use the phrase "SSN+STRING" to refer to the integrated network.

We sought to determine whether the SSN integrated with the STRING networks would yield more accurate predictions than the SSN alone. Comparing Figure 2b to Figure 2c, we observed that the network propagation methods were able to utilize the additional information in the STRING networks to improve predictions, increasing the median $F_{\max}$ by about 0.05 for BP terms over using the SSN only (Figure 2b vs

Figure 2c). These improvements were statistically significant for each method (uncorrected rank-sum test p-values 1.1×10^{-4}, 1.2×10^{-3}, 9.1×10^{-4} for FastSinkSource, GeneMANIA, and BirgRank, respectively). In contrast, we observed that incorporating the STRING networks into the SSN slightly hindered the results for Local (median $F_{\max}$ decrease of 0.002). These results demonstrate a clear advantage of the network propagation methods over Local (e.g., FastSinkSource vs Local $F_{\max}$ uncorrected rank-sum test p-value 8.6×10^{-9}). The inclusion of STRING networks slightly decreased the performance for MF terms at all stringency cutoffs suggesting that the data in STRING was not ideal for predicting these annotations (Figure S3d). Therefore we used only the SSN for MF terms subsequently.

Finally, we compared the running times for each of the algorithms for LOSO evaluation of BP terms on the integrated networks (Table 1). FastSinkSource achieved the fastest running time of the propagation methods with a factor of 17 and factor of 32 speed improvement over GeneMANIA and BirgRank, respectively. Note that FastSinkSource and GeneMANIA performed network-wide predictions, i.e., they computed a score for each unknown example in every species. In contrast, for BirgRank, we limited predictions to the set of proteins that were either positive or negative examples in the left-out species.

Method	Time (minutes)	Solver	Parameters
FastSinkSource	2.6	10 power iterations	$\alpha = 0.95$
GeneMANIA	44.0	Conjugate gradient	$tol = 1 \times 10^{-5}$
BirgRank	82.3	Power iteration	$\alpha = 0.9, \lambda = 0.01, \epsilon = 1 \times 10^{-4}$
Local	0.3	-	-

Table 1: Total running time of the algorithms in minutes (LOSO, BP terms, EXPC annotations, SSN+STRING network) and the solver and parameters for each method. We describe parameter selection for BirgRank in Appendix A.2. We measured the time for each algorithm in independent runs using a single core on a computer with a 2.2 GHz processor and 32GB of memory running CentOS version 7.

We next considered the results for individual species with the SSN+STRING network. We focused on the seven (out of 19) organisms that had at least one BP GO term with at least 10 or more EXPC annotations. For three out of seven species, the improvement of FastSinkSource over Local was statistically significant (Bonferroni-corrected (BF) rank-sum test p-value < 0.05, bold species abbreviations in Figure 2d). Overall the three network propagation methods had fairly similar $F_{\max}$ distributions with some slight differences on a per-species basis. We noted considerable variation among species, with the median $F_{\max}$ for FastSinkSource ranging from 0.4 for *Mycobacterium tuberculosis* to 0.9 for *Neisseria gonorrhoeae*. The median $F_{\max}$ for a species was negatively correlated with the number of annotations with experimental evidence codes for that organism (right side of Figure 2d, Spearman's rank correlation $\rho = -0.93$, p-value $= 2.5 \times 10^{-3}$) suggesting that recovering experimentally-derived annotations in an organism is more challenging when fewer annotations of the same type are available in other organisms.

We observed that while the number of annotations for *Pseudomonas aeruginosa* is similar to the number for *M. tuberculosis*, the latter has a significantly smaller median $F_{\max}$ value (rank-sum test p-value $= 1.2 \times 10^{-8}$). We hypothesized that *M. tuberculosis* may not be closely related to the other species, making it more difficult to recover its annotations. Inspection of the phylogenetic tree of these 19 species confirmed this hypothesis: we see that *M. tuberculosis* is the most distant of the organisms, being the only species from the phylum *Actinobacteria* (Figure S4).

The $F_{\max}$ values varied over almost the entire range between 0 and 1 in several of our analyses (Figure 2(a)–(d)). This wide spread occurred because we considered all GO terms that met our annotation criteria. These GO terms had a wide range of specificity values, as measured by the number of annotated genes. We examined if the $F_{\max}$ for a given GO term-species pair was related to the fraction of annotations left-out for that species and term (out of the total number of annotations for that term across all 19 species). The $F_{\max}$ had a modest negative correlation with the fraction of annotations (Pearson's correlation $= -0.21$, p-value $= 1.2 \times 10^{-9}$). We reasoned that the prediction problem may be challenging for a well-annotated GO term in a species if the term is sparsely annotated in other species, and *vice versa*. Despite the wide spread in the observed range of $F_{\max}$ values, the results in this section point to statistically-significant differences among algorithms.

Examining the Improvement of Network Propagation Over the Baseline

We now turn our attention to a more detailed GO-term-by-GO-term comparison of network propagation algorithms, specifically FastSinkSource, and Local. We directly compared the $F_{\max}$ values of FastSinkSource and Local for each BP GO term across all species. We observed that the $F_{\max}$ value for FastSinkSource was larger than that of Local for 73% of terms and smaller for 25% of the terms (Figure 2e). We observed similar advantages for MF terms (Figure S2i) as well as for GeneMANIA and BirgRank over Local (data not shown). We concluded that network propagation offers an advantage in gene function prediction over a method that considered only the immediate neighbors in the network.

To examine the improvement of FastSinkSource over Local further, we compared the genes with the 30 highest scores computed by each method for the term "SOS response" (GO:0009432) and the left-out species *P. aeruginosa* (Figure 3). We chose this GO term for a case study due to i) the relatively large improvement of $F_{\max}$ (0.15) for FastSinkSource over Local (0.9 and 0.75, respectively), ii) the small number of positive experimental annotations (15 for *P. aeruginosa*), which facilitated the visualization of the relevant networks, and iii) the known importance of SOS response in the development of antibiotic resistance and other aggressions by *P. aeruginosa* [21–23].

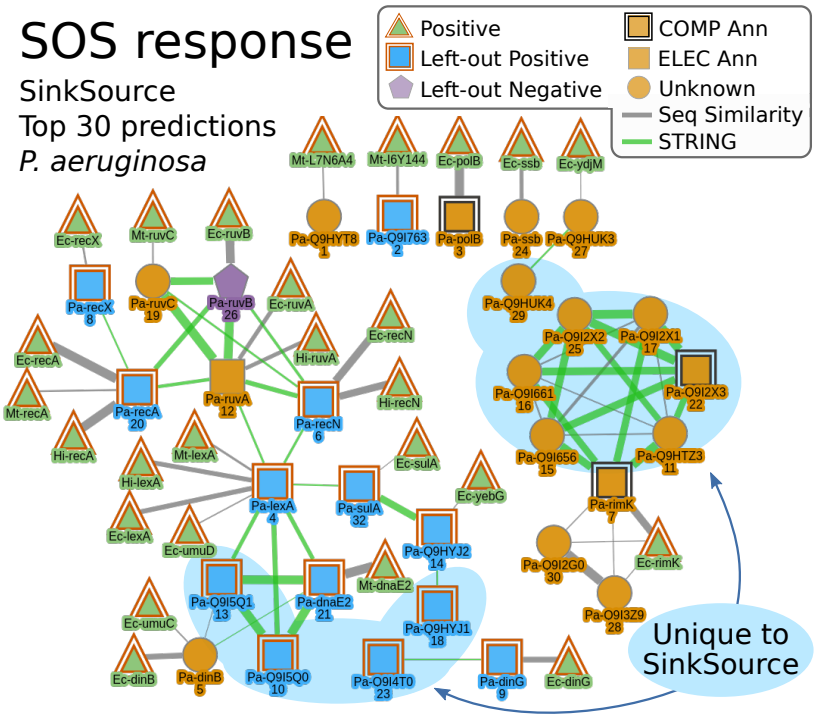

Figure 3: Network connecting the 30 genes with the highest score computed by FastSinkSource for the LOSO evaluation of the GO term "SOS response" (GO:0009432), where the left-out species is *P. aeruginosa*. We show the induced subgraph of these 30 nodes, and the connections to neighboring positive examples for reference. To simplify the graph, we do not show neighboring negative examples, nor positive/negative examples more than one edge away from a prediction, both of which can influence FastSinkSource predictions. We denote genes annotated with this term (with an EXPC evidence code) as either a "Left-out Positive" (i.e., *P. aeruginosa*) or a "Positive" (i.e., the 18 other species). We did not use COMP and ELEC annotations when computing scores, but display them here for reference. Each node's name has two parts: a species name, abbreviated as in Figure 2, and a gene symbol. The number below a node name indicates the the node's ranking ordered by score. The width of an edge is proportional to its weight. This graph, as well as the network connecting the 30 genes with the highest score computed by Local, are available on GraphSpace, the interactive graph sharing website (FastSinkSource: https://graphspace.org/graphs/27042, Local: https://graphspace.org/graphs/27041) [24].

The 30 highest scoring genes computed by FastSinkSource and Local contained 13 and 9 of the 15 left-out positive examples, respectively. Of the remaining 17 genes scored by FastSinkSource (that did not have an EXPC annotation), 10 overlapped with those from Local (orange nodes in Figure 3 that are not inside a blue shape). Examining the annotations of these 10 genes closely, we were able to divide them into three groups. (a) Four genes (*ruvA*, *ruvB*, *rimK*, *polB*) have an ELEC or COMP annotation to SOS response, of which one was a left-out negative example (*ruvB*), suggesting that our current method for defining negative examples can be improved. (b) Four genes (*dinB*, *ruvC*, *ssb*, Q9HYT8) have an ELEC or COMP annotation to functions closely related to SOS response, namely "DNA Repair" (GO:0006281) and/or "cellular response to DNA damage stimulus" (GO:0006974). *dinB* encodes an error-prone polymerase, Pol IV, which is a known component of the SOS response in *E.coli* [25]. *RuvC* has been shown to act specifically on intermediary DNA repair structures catalyzed by the RecA protein [26], and is expressed during SOS response in *M. tuberculosis* [27]. Thus two of the four genes are well characterized components of SOS response in other bacterial systems. (c) Two genes (Q9I2G0, Q9I3Z9) had no annotation directly relevant to SOS response.

Next, we considered the seven FastSinkSource-specific genes that were unknown examples (orange nodes inside blue shape, right side of Figure 3). One gene has a COMP annotation to SOS response (Q9I2X3) and the rest are uncharacterized experimentally with little-to-no functional information available in the literature. Interestingly, five of these genes clustered tightly with two of the COMP annotated predictions, Q9I2X3 and *rimK*, suggesting they could also be involved in SOS response. This case study also highlights the benefits of incorporating interactions from the STRING database. Besides increasing the accuracy of predictions (e.g., enabling the recovery of four additional left-out positive examples, blue nodes inside blue shape in the bottom left of Figure 3) , the edges from STRING provided more context on how these genes are related to one another. For example, the edges connecting the genes *ruvA*, *ruvB* and *ruvC* have the types "co-occurrence" and "neighborhood" with scores above 650, meaning these genes co-occur in many other species and are in close proximity in the genome.

LOSO Evaluation with Computational and Electronic Evidence Codes

Most of the 19 organisms we considered had very few EXPC annotations, i.e., based on experimental evidence. We observed that over 95% of the 25,000 curator-reviewed annotations based on computational analysis (COMP) for these organisms had the evidence codes IBA (Inferred from Biological aspect of Ancestor, 19,300 annotations) or ISS (Inferred from Sequence or structural Similarity, 4,500 annotations). We reasoned that these annotations were most likely based on experimental annotations in other organisms. Therefore, we mimicked the inclusion of more experimentally-based annotations by expanding our analysis to include COMP annotations (see "Datasets" for the relevant evidence codes). Specifically, we repeated our LOSO analysis using EXPC and COMP annotations in 18 species, and evaluated each method's ability to recover the COMP annotations in the left-out species using the SSN+STRING networks. We limited our analysis to GO terms which had at least 50 annotations across all 19 organisms, at least 10 annotations in a given left-out species, and at least 10 annotations in the other 18 species. A total of 473 BP terms across nine organisms (1,656 species-term pairs) satisfied these criteria. Due to the high similarity of the $F_{\max}$ distributions between the three network propagation methods, we show and discuss the results only for FastSinkSource and Local, with results for all four methods in Figure S5.

We observed that recovering COMP annotations (Figure 4a) was much easier than recovering EXPC annotations (Figure 2d). For eight of the nine species with at least 10 COMP annotations, the median $F_{\max}$ values of FastSinkSource and Local were 0.95 or larger. The reason for the improved performance of both algorithms is likely to be our earlier observation that most COMP annotations had IBA or ISS evidence codes. The exception was *V. cholerae* (median $F_{\max}$ 0.85 and 0.78 for FastSinkSource and Local, respectively), which has many more COMP annotations than the other species. The improvement of FastSinkSource over Local was statistically significant only for two organisms (rank-sum test BF-corrected p-value < 0.05, bold species abbreviations in Figure 4a).

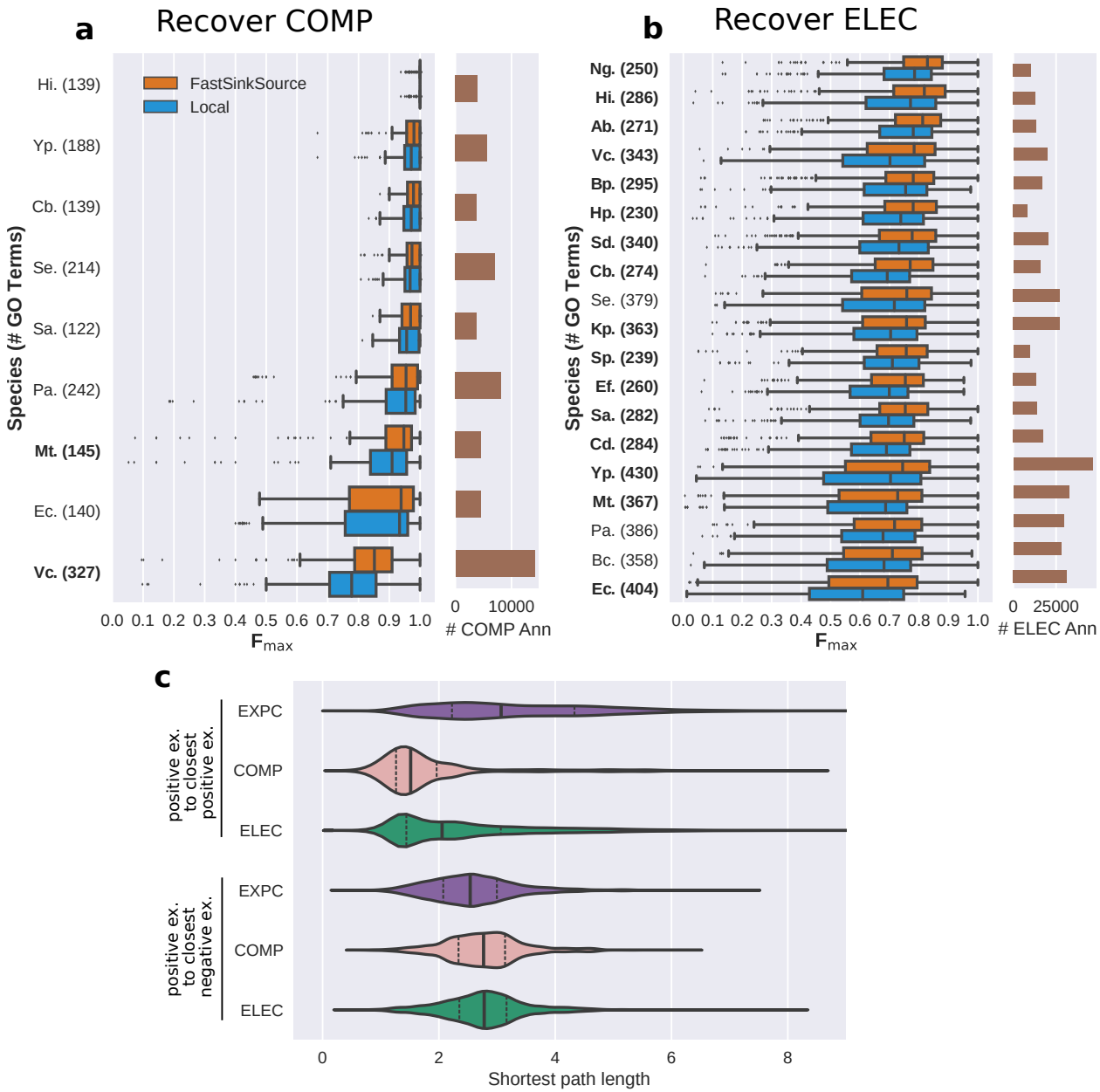

Figure 4: Comparison of $F_{\max}$ results for LOSO evaluation for COMP and ELEC evidence codes using the SSN+STRING network. A bold species name indicates that the difference between the distributions for FastSinkSource and Local was statistically significant (rank-sum test BF-corrected p-value < 0.05). (**a**) Evaluation of recovery of COMP annotations in the left-out species. (**b**) Evaluation of recovery of ELEC annotations in the left-out species. Species name abbreviations not defined in Figure 2 are the following: *Acinetobacter baumannii* (Ab), *Burkholderia cepacia* (Bc), *Bordetella pertussis* (Bp), *Clostridium botulinum* (Cb), *Clostridioides difficile* (Cd), *Enterococcus faecium* (Ef), *Haemophilus influenzae* (Hi), *Helicobacter pylori* (Hp), *Klebsiella pneumoniae* (Kp), *Staphylococcus aureus* (Sa), *Shigella dysenteriae* (Sd), *Streptococcus pyogenes* (Sp). (**c**) Violin plots of the weighted shortest path lengths from the annotations of a given left-out species (*E. coli*, *M. tuberculosis*, or *P. aeruginosa*), to the closest positive (top-half) or negative (bottom-half) example in the other species. The black line shows the median and the dotted lines show the 25th and 75th percentiles.

Encouraged by these results, we reasoned that it would be appropriate to also evaluate the recovery of

ELEC annotations since their reliability has been shown to rival that of non-experimental curated annotations [28]. For these annotations, 788 BP terms across all 19 species (6043 species-term pairs) satisfied our criteria. For ELEC annotations, the median of the $F_{\max}$ values of FastSinkSource across the species ranged from a median of 0.7 to 0.85 with a statistically significant improvement of FastSinkSource over Local for 16 out of 19 species (rank-sum test BF-corrected p-value < 0.05, see Figure 4b). GeneMANIA and BirgRank did not perform as well relative to FastSinkSource with a statistically significant improvement over Local for only 6/19 and 2/19 species, respectively.

Comparing the results in Figure 2(c), Figure 4(a), and Figure 4(b), we observed two main trends. First, both FastSinkSource and Local had better $F_{\max}$ values for COMP annotations than for EXPC and IEA annotations. Second, the differences between the $F_{\max}$ values for FastSinkSource and for Local were much smaller for COMP annotations than for EXPC and ELEC annotations. To probe the causes of these trends, we computed the distances of the left-out positives to the positives in the other 18 species for each analysis, i.e., corresponding to EXPC, COMP, and ELEC annotations. We limited these computations to the three species well-annotated in all analyses, and, for a given species, to the GO terms used in all analyses. In the SSN+STRING network, we computed the absolute value of the base-10 logarithm of the degree-normalized weight of each edge. Next, for each GO term, we used these values to compute the shortest path from each positive example in the left-out species to any positive example in the other 18 species. We repeated this process to compute the distances of the left-out positive examples to the negative examples of the other 18 species.

Comparing the first and second distributions in Figure 4c, we found that the distances between the left-out EXPC annotations and the retained EXPC annotations (median of 3.1, Median Absolute Deviation or MAD of 1) were greater, by and large, than the distances between left-out COMP annotations and the retained EXPC+COMP annotations (median of 1.5, MAD of 0.3). Moreover, the distances from the left-out ELEC annotations to the other EXPC+COMP annotations (third distribution in Figure 4c, median of 2.1, MAD of 0.7) lay in-between the values in the first two distributions. Interestingly, for EXPC annotations, the distance to negative examples (fourth distribution, median of 2.5, MAD of 0.5) was smaller than the distance to positive examples, which is not the case for the other two sets of annotations. A p-value $< 10^{-300}$ for the Kruskal-Wallis test indicated that these six sets of distances did not originate from the same distribution. Next, we conducted all pairwise comparisons between the six distributions using Dunn's test. Each comparison was statistically significant (p-value $< 10^{-135}$), except between the last two distributions (distances to negative examples from left-out COMP and from left-out ELEC annotations). We note that the large sizes of the distributions (14,000 to 92,000 values) may have led to an overestimation of the statistical significance. This pattern of variation in the distances may serve to partially explain the trends we observed in comparing Figure 2(c), Figure 4(a), and Figure 4(b). Since COMP annotations were quite proximal to the closest positive annotations in other species, network propagation (FastSinkSource) was not superior to Local. On the other hand, EXPC annotations were relatively farther to the closest positive annotations in other species. Hence, the network propagation inherent in FastSinkSource yielded better predictive performance than the purely direct-neighbor-based viewpoint of Local. We acknowledge that distances to closest positive examples do not capture the picture completely since propagation in FastSinkSource can involve several paths in the network.

Scaling to 200 Bacterial Species

To test the ability of FastSinkSource and other algorithms to scale to larger networks, we built a SSN with 200 bacterial species. We chose the 200 bacterial species with the largest number of GO annotations (protein-GO term pairs) with EXPC or COMP evidence codes. These species had a wide range of genome sizes, ranging from 438 (*Mycoplasma genitalium*) to 10,020 protein-coding genes (*Streptomyces bingchenggensis*) in UniProt. We used all-vs-all BLASTP and an E-value cutoff of 0.1 to build the SSN, which yielded a network with approximately 815,000 nodes and 73 million edges. We did not include STRING networks in this analysis.

We repeated the previously-described LOSO validations with this network. As 84% of EXPC annotations in the 200 bacterial species were from the 19 organisms we had considered in earlier analyses, we discuss only our ability to recover COMP and ELEC evidence codes using LOSO validation. Recall that we define positive, negative, and unknown examples using annotations with EXPC and COMP evidence codes. We

limited our analyses to BP terms. We found that 42 species had at least one GO term with 10 or more COMP annotations, and all 200 species passed that cutoff for ELEC annotations. A total of 711, and 761 GO terms satisfied our annotation criteria for COMP and ELEC annotations, respectively.

We began by setting $\alpha = 0.95$ and re-evaluating the trade-off between accuracy of FastSinkSource and its speed. To limit the running time of this analysis, we subsampled 5% of the 711 terms with COMP annotations. To evaluate a representative subsample of GO terms, we grouped them by their number of annotations into three bins: (i) 50–200, (ii) 200–500, and (iii) 500 or more, and sampled terms uniformly at random evenly between these three sets. For these selected terms, we needed a median of 126 iterations to fix the ordering of all nodes (Figure S6a). The median number of iterations required to reach a Kendall's τ of 0.8 and 0.9 compared to the fixed ordering were 5 and 11 respectively, suggesting that using the same number of iterations (10) that worked well for the 19 species SSN+STRING may also approximate the fixed ordering well for the 200 species network. This evaluation using 5% of the GO terms took a total of 5.4 hours to run. Next, we expanded LOSO validation to all GO terms and compared SinkSource with $\alpha = 1$ for 1,000 iterations and FastSinkSource with $\alpha = 0.95$ for 10 iterations. The median value of $F_{\max}$ dropped from 0.990 for $\alpha = 1.0$ with 1,000 iterations, to 0.982 for $\alpha = 0.95$ with 10 iterations (Figure S6c). This decrease of 0.08% in the median $F_{\max}$ value was statistically significant (rank-sum test p-value $= 2.5 \times 10^{-11}$) likely because of the large number of species-GO terms pairs (6,591).

When we repeated this approach for ELEC annotations, we needed a median of 564 iterations to fix the node ordering for 5% of terms (sampled using the same strategy as for COMP annotations) with 8 and 20 iterations to reach Kendall's τ values of 0.8 and 0.9, respectively (Figure S6b). This analysis took a total of 6.7 hours. We decided to maintain the parameter choice of 10 iterations. When we compared SinkSource with $\alpha = 1.0$ and 1,000 iterations to FastSinkSource with $\alpha = 0.95$ and 10 iterations, the median $F_{\max}$ value surprisingly increased from 0.727 to 0.736 (Figure S6d). This marginal increase in the median value of $F_{\max}$ (1.2%) was also statistically-significant (rank-sum test p-value $= 3.5 \times 10^{-10}$) for the same reason as before: the number of species-GO terms was even larger (73,708). Therefore, we chose to maintain $\alpha = 0.95$ with 10 iterations for these evaluations, which corresponded to a speed-up of a factor of 109 (28 hours vs. 3,031 hours) and of 117 (36 hours vs. 4,284 hours) for COMP and ELEC annotations, respectively, over $\alpha = 1$ with 1,000 iterations (Figure S6e,f).

Algorithm	Median $F_{\max}$	MAD	# Significant Species
Recover COMP annotations (42 species)			
FastSinkSource	0.982	0.02	4
GeneMANIA	0.983	0.02	6
BirgRank	0.974	0.02	0
Local	0.977	0.02	-
Recover ELEC annotations (200 species)			
FastSinkSource	0.736	0.10	20
GeneMANIA	0.721	0.10	0
BirgRank	0.714	0.10	0
Local	0.709	0.10	-

Table 2: Comparison of $F_{\max}$ results of LOSO evaluation for different evidence codes and algorithms. (**a**) Recovery of COMP annotations in a left-out species from EXPC and COMP annotations in other species. Number of species-GO term pairs: 6,951. (**b**) Recovery of ELEC annotations in a left-out species from EXPC and COMP annotations in other species. Number of species-GO term pairs: 73,708. "MAD" stands for Median Absolute Deviation. The column titled "# Significant Species" shows the number of species for which the improvement in median $F_{\max}$ value of a given algorithm over Local was statistically significant (rank-sum test, corrected by number of species).

Next, we compared the results of LOSO validation for all four algorithms. Rather than show the $F_{\max}$ distributions for all 200 species individually, we have summarized the results in Table 2. We observed that the median $F_{\max}$ values of each algorithm were fairly similar to those in Figure 4; thus, many of the observations made in earlier sections remain true here. When recovering COMP annotations, the number of species for

which FastSinkSource and GeneMANIA had a statistically significant improvement over Local were similar (four and six organisms, respectively, out of 42), whereas BirgRank was not able to improve over Local for any species. For ELEC annotations, FastSinkSource was the only method able to significantly improve over Local (20 species out of 200). We also compared the running times of each of the methods (Table S3). For many GO terms, several species had annotations only with ELEC evidence codes. Since we were leaving out only organisms with EXPC or COMP annotations, we optimized FastSinkSource and GeneMANIA further for each of these GO terms by executing them simultaneously for all species with only ELEC annotations for that term. BirgRank was significantly slower than FastSinkSource (factor of 150) due to the large number of genes that were evaluated and because we could not apply this speed-up to it. FastSinkSource was still faster than GeneMANIA, but only by a factor of 1.5.

Finally, we asked if the inclusion of 181 additional species improved the accuracy of predictions for the original 19 bacteria. For each of the three groups of evidence codes, we compared the $F_{\max}$ distributions for FastSinkSource on the 19-species SSN to the results on the 200-species SSN. To ensure a fair comparison, for each set of evidence codes, we limited our attention to those terms present in the evaluations for both groups of species. For EXPC annotations, the addition of species significantly improved the performance for two (out of seven) organisms: *M. tuberculosis*, which was most distantly related to the other 19 bacteria, and *S. enterica* (rank-sum test BF-corrected p-value < 0.05, Figure S7a). For COMP annotations, four out of nine species showed significant improvements (*M. tuberculosis*, *V. cholerae*, *C. botulinum*, *S. aureus*, Figure S7b). Surprisingly, for ELEC annotations, we actually observed a significant decrease in $F_{\max}$ values for four out of 19 species (*H. influenzae*, *S. aureus*, *Y. pestis*, *C. botulinum*), with no bacteria showing significant improvements (Figure S7c). This reversal could potentially be due to the fact that when defining positive and negative examples, we only used annotations with the evidence codes under consideration, which are EXPC or COMP in this case. Hence a gene may be a negative example to a term even if it has an ELEC annotation to that term. We reasoned that these additional negative examples may cause the significant decreases for ELEC annotations. We tested this hypothesis by relabelling negative examples as unknown examples even if they had only an ELEC annotation to the GO term and then repeated the LOSO validation for ELEC annotations. No species had either a statistically significant decrease or increase in $F_{\max}$ distribution for the 200-species analysis compared to the the 19-species evaluation (Figure S7d). We concluded that supplementing annotations from additional species improved prediction accuracy for EXPC and for COMP annotations.

Discussion

We have developed a highly-scalable network-based algorithm called FastSinkSource for gene function prediction that can accurately assimilate information from 100s of genomes. On a network with 73 million edges connecting 815,000 genes in 200 bacterial species, FastSinkSource took an average of 15 seconds per GO term to compute prediction scores, achieving a speed that was over 100 times faster than SinkSource, while maintaining accuracy improvements over the BLAST-based Local baseline. The benefits of our approach include the ability to make predictions for all genes simultaneously and increasing the number of positive training examples by pooling information in multiple organisms.

Based on these results, we believe that FastSinkSource has the potential to be useful in predicting functions for genes in a newly sequenced genome. To this end, FastSinkSource needs access to a sequence-similarity network connecting genes in other genomes, Ideally, these genomes are phylogenetically related to the new genome. This network may be computed in a pre-processing step. With this network in hand, we envision a strategy that first connects each gene in the new genome to genes in other genomes based on sequence similarity. Subsequently, for each GO term, we run FastSinkSource on the resulting network. A challenge that arises is how to determine the ideal value of α and the number of iterations to execute FastSinkSource. In our experiments, we used LOSO validation to determine the values of these parameters. We performed this validation on all sufficiently-annotated GO terms (at least 10 genes). As the number of species included in the network increases, the LOSO process becomes slower.

We expand upon an approach we used for the 200-species network to propose an alternative method that may be more efficient than the LOSO-based strategy. This idea requires the user to provide a desired value τ of Kendall's correlation coefficient as an input. We select a random subset S of GO terms. For each term

in S, we execute FastSinkSource with different values of $\alpha < 1$ to convergence, i.e., till we have determined the relative orders of all node scores. For each value of α, after each iteration, we compute the Kendall's correlation coefficient between the node order at this stage and the ranking upon convergence with this value of α. We stop as soon as this value is at least as large as τ. This approach may yield a different number of iterations for each GO term. We record the median of these values as the choice to use for the analysis with all GO terms. Note that this approach can result in a different number of iterations for each value of α. The user has the freedom to select the parameter combination of their choice. Alternatively, the user may provide a desired value of α as an input parameter.

A central theme in our work is that it is sufficient to rank the nodes in a network correctly instead of computing their scores to a desired degree of accuracy, as has been noted for random walks [29]. Mathematical arguments on the rate of convergence of FastSinkSource underpinned our approach. More generally, these principles may be useful in other applications of propagation algorithms in network biology, such as computing disease-related genes, gene modules, and drug targets [30].

Currently, the method computes scores for the unknown examples in all organisms in the network and not just a single species of interest. Further improvements in the efficiency of FastSinkSource are possible, e.g., by limiting the propagation to a subgraph of the network more directly relevant for the prediction task. The challenge will be to ensure that we compute the relative ranks of the relevant nodes accurately. Another valuable direction of research may be to integrate these genomewide network-based function prediction methods with the techniques used for assigning gene function in genome annotation pipelines. While we focused on bacteria, our methods apply generally to any set of related species of interest. As more experimentally-based annotations and data become available, we anticipate that algorithms for genomewide, multi-species gene function prediction will become increasingly valuable.

Acknowledgements

The research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the Army Research Office (ARO) under cooperative Agreement Number [W911NF-17-2-0105]. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, ARO, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

Author Contributions

Conceptualization, J.L., T.M.M.; Methodology, J.L., S.K., T.M.M.; Validation, J.L; Formal Analysis, J.L., T.M.M.; Resources, J.L, S.K., T.M.M.; Data Curation, J.L.; Software, J.L.; Writing – Original Draft, J.L., S.K., T.M.M.; Writing – Review & Editing, J.L., S.K., T.M.M.; Visualization, J.L., T.M.M.; Supervision, T.M.M; Funding Acquisition, S.K., T.M.M.;

Declaration of Interests

The authors declare that they have no competing interests.

References

[1] Miriam Land, Loren Hauser, Se-Ran Jun, Intawat Nookaew, Michael R. Leuze, Tae-Hyuk Ahn, Tatiana Karpinets, Ole Lund, Guruprased Kora, Trudy Wassenaar, Suresh Poudel, and David W. Ussery. Insights from 20 years of bacterial genome sequencing. *Functional & Integrative Genomics*, 15(2):141–161, 2015.

[2] The UniProt Consortium. UniProt: the universal protein knowledgebase. *Nucleic Acids Research*, 45(D1):D158–D169, 2017.

[3] The Gene Ontology Consortium. Expansion of the Gene Ontology knowledgebase and resources. *Nucleic Acids Research*, 45(D1):D331–D338, 2017.

[4] Sara Mostafavi, Debajyoti Ray, David Warde-Farley, Chris Grouios, and Quaid Morris. GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function. *Genome Biology*, 9(Suppl 1):S4, 2008.

[5] Hyunghoon Cho, Bonnie Berger, and Jian Peng. Compact integration of multi-network topology for functional analysis of genes. *Cell Systems*, 3(6):540–548.e5, 2016.

[6] Vladimir Gligorijević, Meet Barot, and Richard Bonneau. deepNF: Deep network fusion for protein function prediction. *Bioinformatics*, 34(22):3873–3881, 2018.

[7] Domenico Cozzetto, Daniel WA Buchan, Kevin Bryson, and David T Jones. Protein function prediction by massive integration of evolutionary analyses and multiple data sources. *BMC Bioinformatics*, 14(Suppl 3):S1, 2013.

[8] Damiano Piovesan, Manuel Giollo, Emanuela Leonardi, Carlo Ferrari, and Silvio C.E. Tosatto. INGA: Protein function prediction combining interaction networks, domain assignments and sequence similarity. *Nucleic Acids Research*, 43(W1):W134–W140, 2015.

[9] Biaobin Jiang, Kyle Kloster, David F. Gleich, and Michael Gribskov. AptRank: an adaptive pagerank model for protein function prediction on bi-relational graphs. *Bioinformatics*, 33(12):1829–1836, 2017.

[10] Jeffrey M Yunes and Patricia C Babbitt. Effusion: Prediction of protein function from sequence similarity networks. *Bioinformatics*, 35(3):442–451, 2018.

[11] Chengxin Zhang, Wei Zheng, Peter L. Freddolino, and Yang Zhang. MetaGO: Predicting gene ontology of non-homologous proteins through low-resolution protein structure prediction and protein-protein network mapping. *Journal of Molecular Biology*, 430(15):2256–2265, 2018.

[12] Yi-Chien Chang, Zhenjun Hu, John Rachlin, Brian P. Anton, Simon Kasif, Richard J. Roberts, and Martin Steffen. COMBREX-DB: an experiment centered database of protein function: Knowledge, predictions and knowledge gaps. *Nucleic Acids Research*, 44(D1):D330–D335, 2015.

[13] Sheng Wang, Hyunghoon Cho, ChengXiang Zhai, Bonnie Berger, and Jian Peng. Exploiting ontology graph for predicting sparsely annotated gene function. *Bioinformatics*, 31(12):i357–i364, 2015.

[14] Aashish Jain and Daisuke Kihara. Phylo-PFP: Improved automated protein function prediction using phylogenetic distance of distantly related sequences. *Bioinformatics*, 2018.

[15] T. M. Murali, Matthew D. Dyer, David Badger, Brett M. Tyler, and Michael G. Katze. Network-based prediction and analysis of HIV dependency factors. *PLoS Comput Biol*, 7(9):e1002164+, September 2011.

[16] Mustafa Coskun, Ananth Grama, and Mehmet Koyuturk. Efficient processing of network proximity queries via Chebyshev acceleration. In *KDD*, pages 1515–1524, 2016.

[17] Tolga Can, Orhan Çamoğlu, and Ambuj K. Singh. Analysis of protein-protein interaction networks using random walks. In *Proceedings of the 5th International Workshop on Bioinformatics*, BIOKDD '05, pages 61–68, New York, NY, USA, 2005. ACM.

[18] Yuxiang Jiang, Tal Ronnen Oron, Wyatt T. Clark, Asma R. Bankapur, et al. An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome Biology*, 17(1):184, 2016.

[19] Damian Szklarczyk, John H Morris, Helen Cook, Michael Kuhn, Stefan Wyder, Milan Simonovic, Alberto Santos, Nadezhda T Doncheva, Alexander Roth, Peer Bork, Lars J. Jensen, and Christian von Mering. The STRING database in 2017: Quality-controlled protein-protein association networks, made broadly accessible. *Nucleic Acids Research*, 45(D1):D362–D368, 2016.

[20] Noah Youngs, Duncan Penfold-Brown, Kevin Drew, Dennis Shasha, and Richard Bonneau. Parametric Bayesian priors and better choice of negative examples improve protein function prediction. *Bioinformatics*, 29(9):1190–1198, 2013.

[21] Bénédicte Michel. After 30 years of study, the bacterial SOS response still surprises us. *PLoS Biology*, 3(7):e255, 2005.

[22] Zeynep Baharoglu and Didier Mazel. SOS, the formidable strategy of bacteria against https://doi.org/10.1093/nar/gkv523 aggressions. *FEMS Microbiology Reviews*, 38(6):1126–1145, 2014.

[23] Didier Hocquet and Xavier Bertrand. Metronidazole increases the emergence of ciprofloxacin- and amikacin-resistant *Pseudomonas Aeruginosa* by inducing the SOS response. *Journal of Antimicrobial Chemotherapy*, 69(3):852–854, 2013.

[24] Aditya Bharadwaj, Divit Singh, Anna Ritz, Allison N. Tegge, Christopher L. Poirel, Neil Adames, Pavel Kraikivskiy, Kurt N. Luther, Shiv D. Kale, Jean Peccoud, John J. Tyson, and T. M. Murali. GraphSpace: Stimulating interdisciplinary collaborations in network biology. *Bioinformatics*, 33(19):3134–3136, 2017.

[25] Gregory J McKenzie, Peter L Lee, Mary-Jane Lombardo, PJ Hastings, and Susan M Rosenberg. SOS mutator DNA polymerase IV functions in adaptive mutation and not adaptive amplification. *Molecular cell*, 7(3):571–579, 2001.

[26] Hazel J Dunderdale, Fiona E Benson, Carol A Parsons, Gary J Sharpies, Robert G Lloyd, and Stephen C West. Formation and resolution of recombination intermediates by *E. coli* RecA and RuvC proteins. *Nature*, 354(6354):506, 1991.

[27] Lisa F Dawson, Joanna Dillury, and Elaine O Davis. RecA-independent DNA damage induction of *Mycobacterium tuberculosis ruvC* despite an appropriately located SOS box. *Journal of Bacteriology*, 192(2):599–603, 2010.

[28] Nives Škunca, Adrian Altenhoff, and Christophe Dessimoz. Quality of computationally inferred Gene Ontology annotations. *PLoS Computational Biology*, 8(5):e1002533, 2012.

[29] Amy N Langville and Carl D Meyer. *Google's PageRank and beyond: The science of search engine rankings*. Princeton University Press, 2006.

[30] Lenore Cowen, Trey Ideker, Benjamin J Raphael, and Roded Sharan. Network propagation: a universal amplifier of genetic associations. *Nature Reviews Genetics*, 18(9):551–562, Sep 2017.

[31] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *The Twentieth International Conference on Machine Learning, August 21-24, 2003, Washington, DC USA*, pages 912–919, 2003.

[32] Chao Zhang, Shan Jiang, Yucheng Chen, Yidan Sun, and Jiawei Han. Fast Inbound Top-K Query for Random Walk with Restart. *Mach. Learn. Knowl. Discov. Databases*, 9285:608–624, Sep 2015.

[33] Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 16:321–328, 2004.

[34] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[35] Sara Mostafavi and Quaid Morris. Fast integration of heterogeneous data sources for predicting gene function with limited annotation. *Bioinformatics*, 26(14):1759–1765, 2010.

STAR Methods

The FastSinkSource algorithm

As described in the section "Overview of FastSinkSource", we seek to improve the efficiency of SinkSource by developing alternative methods for establishing convergence. We first describe the SinkSource algorithm in detail before extending it to FastSinkSource.

SinkSource

A standard formulation of network-based GO annotation prediction is as a semi-supervised problem. Here, some nodes have functional labels and the goal is to assign labels to unlabeled nodes. More formally, we are given a weighted undirected FLN $G = (V, E, w)$, where w_{uv} represents the weight of the edge (u, v). In addition, for a GO term τ, we have a partition of the nodes in V into three sets: V^+, V^- and V^0 positive, negative, and unknown examples, respectively. The goal is to compute a score $s(u)$ for each node u that is an unknown example, by propagating the positive and negative labels across G. Strictly speaking, the score depends on τ as well but we omit it for clarity.

SinkSource computes a score $s(u)$ between 0 and 1 for each u that is an unknown example for τ by minimizing the function

$$S(G, s) = \sum_{(u,v)\in E} w_{uv}\big(s(u) - s(v)\big)^2, \tag{1}$$

with the scores of positive and negative examples fixed at 1 and 0, respectively. The function $S(G, s)$ is minimized when, for each unknown example u,

$$s(u) = \frac{\sum\limits_{v\in N(u)} w_{uv} s(v)}{d(u)}, \tag{2}$$

where $N(u)$ is the set of neighbors of node u and $d(u)$ is the weighted degree of node u in G. Since the scores of the positive and negative examples are fixed, we can separate Equation (2) into two parts: one corresponding to contributions from the set $N^0(u)$ of neighbors of u that are unknown examples and the second to a constant contribution from the set $N^+(u)$ of positively-labeled neighbors of u as follows:

$$s(u) = \frac{\sum\limits_{v\in N^0(u)} w_{uv} s(v)}{d(u)} + f(u), \tag{3}$$

where

$$f(u) = \frac{\sum\limits_{v\in N^+(u)} w_{uv}}{d(u)}.$$

Note that negatively-labeled neighbors of u do not contribute to the score since each of their scores is fixed at 0.

We define $\mathbf{s}$ to be a $|V^0| \times 1$ vector formed by the node scores of all unknown examples and $\mathbf{f}$ to be a $|V^0| \times 1$ constant vector in which uth element $f(u)$ is the contribution to $s(u)$ from the positively-labeled neighbors of u. Further, we let the $|V^0| \times |V^0|$ matrix $\mathbf{P}$ contain the degree-normalized edge weights in G among pairs of nodes in V^0, i.e., $P_{uv} = w_{uv}/d(u)$ for every pair of unknown examples (u, v). Then for every unknown example u, we can combine the score equations (3) into a single linear system:

$$\mathbf{s} = \mathbf{P}\mathbf{s} + \mathbf{f}. \tag{4}$$

After initializing $\mathbf{s}$ to be the zero vector, we compute $\mathbf{s}$ by repeatedly applying Equation (4). This process of power iteration is known to converge [31], resulting in the value of $\mathbf{s} = (\mathbf{I} - \mathbf{P})^{-1}\mathbf{f}$.

FastSinkSource

FastSinkSource uses essentially the same equation as SinkSource to compute scores (Equation (4)), with the addition of a parameter $0 < \alpha \leq 1$ that controls the proportion of score each node receives from its neighbors.

$$\mathbf{s} = \alpha(\mathbf{P}\mathbf{s} + \mathbf{f}) \tag{5}$$

Note that if $\alpha = 1$, this equation is the same as for SinkSource (Equation (4)). Below we describe how to compute an upper and lower bound on the score of each node after each iteration and the benefits of these bounds. Then we describe the FastSinkSource algorithm.

Lower and Upper Bounds. If $s(u, i)$ denotes the score of node u after i iterations, then we can prove that the score for every node increases with the number of iterations, i.e., $s(u, i) \leq s(u, i + 1)$, and that

$$s(u, i) \leq s(u) \leq s(u, i) + \frac{\alpha^i \|\mathbf{f}\|}{(1 - \alpha)}, \tag{6}$$

where $\|\mathbf{f}\|$ is the largest absolute value of the entries in $\mathbf{f}$. In other words, the score $s(u, i)$ is a lower bound on the final score $s(u)$ and $s(u, i) + \frac{\alpha^i \|\mathbf{f}\|}{(1-\alpha)}$ bounds $s(u)$ from above. Thus the difference between the sought-for score and its value after i iterations decreases geometrically with i. We based our proof of this result (Appendix A.1) on techniques used by Zhang *et al.* [32], who used similar ideas for finding the k nodes with the highest RWR scores.

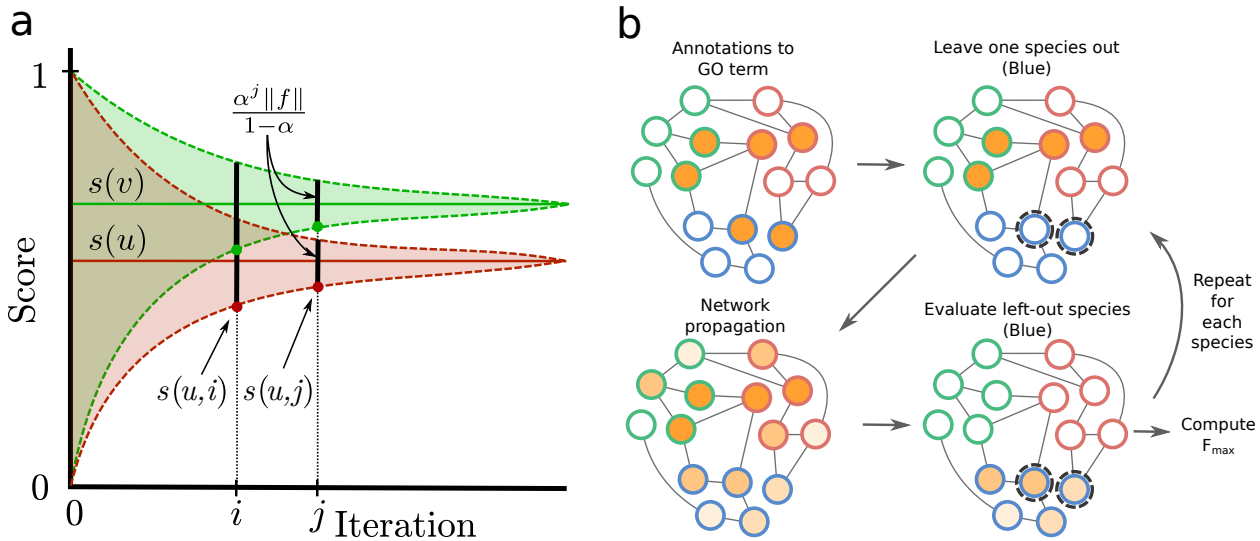

Figure 5: Illustration of convergence of FastSinkSource and LOSO validation. (**a**) Geometric convergence of scores in SinkSource. The solid red line indicates the score $s(u)$ for a node u that we are seeking to compute. The shaded red region shows how different $s(u)$ can be from the score computed for node u after each iteration. The green line and region correspond to node v. At iteration i, we cannot be certain if $s(v)$ is larger or smaller than $s(u)$ because the difference between their computed scores after i iterations is less than the error term. By iteration j, however, the difference is larger than the error term, which implies that $s(v) > s(u)$. (**b**) Overview of LOSO validation. The picture shows a SSN where each border color represents genes from a different species. The orange nodes represent genes annotated with a given GO term (top left). All annotations are left out for a given species (top right) and evaluated how well they can be recovered (bottom right) by propagating the annotations from the known genes to the nearby unknown genes (bottom left). This process is repeated for all GO terms and all species.

Now consider the scores of two nodes u and v after i iterations (Figure 5a). If $s(u, i) \leq s(v, i) \leq s(u, i) + \frac{\alpha^i \|\mathbf{f}\|}{(1-\alpha)}$, then the interval spanned by the lower and upper bounds for $s(u)$ overlaps the corresponding

interval for $s(v)$. In this case, we cannot be sure whether the final score $s(u)$ will be larger or smaller than $s(v)$. However, if we observe in a later iteration j that $s(v,i) > s(u,i) + \frac{\alpha^i \|\mathbf{f}\|}{(1-\alpha)}$, then the intervals are disjoint and we are guaranteed that $s(u) < s(v)$ (Figure 5a). We can conclude that if the interval spanning the lower and upper bounds of u does not overlap with any other node's interval after a certain number of iterations, then we have determined u's final rank, i.e., the rank upon convergence. By extension, if this property is true for every node, then we have ranked all the nodes correctly and no further iterations are needed. This observation has two significant benefits:

(a) It inspires an alternative strategy for checking convergence. At the end of each iteration, we sort all the nodes in increasing order of score. Then for each index k and node u_k in this sorted order, we check if $s(u_k,i) + \frac{\alpha^i \|\mathbf{f}\|}{(1-\alpha)} < s(u_{k+1},i)$. If this inequality is true for every index, then the rankings will not change in subsequent iterations and we say that the process has converged. We sorted the nodes by score at the end of each iteration. Note that we need only compare $O(|V|)$ pairs of node scores to check for convergence. This approach is especially useful when we only need to compute the correct ranking of a subset of nodes by their scores, e.g., during LOSO validation.

(b) If we stop after i iterations, we have an estimate $(\frac{\alpha^i \|\mathbf{f}\|}{(1-\alpha)})$ of how much we have under-estimated every node's score. We also have an upper bound on the number of node pairs that may be ranked incorrectly with respect to each other, which is the number of node pairs that have overlapping lower and upper bounds. We compute this number in $O(|V|)$ time as follows. For each index k, we compute a pointer to the last index l_k such that the score at index l_k lies within the upper bound for node k. We add $l_k - k$ to the number of overlaps. (To compute l_1, we walk along the sorted list from index two onward.) To compute l_{k+1}, we walk along the list from l_k. In this manner, we analyze each index only once.

It is possible for two or more nodes to have the same lower bounds and the same upper bounds in every iteration. Our approach can determine the rank of these nodes as a group with respect to the other nodes. In this case, we simply give these matching nodes the same ranking as we have no way to distinguish how to order them in relation to each other.

Other Algorithms

We compared FastSinkSource to a baseline method we call Local and to two network propagation methods: GeneMANIA [4], and BirgRank [9]. GeneMANIA utilizes Gaussian Random Field (GRF) label propagation [33] to diffuse labels from positive and negative examples to make term-based predictions [4]. SinkSource is similar to GeneMANIA but does not allow the scores of the given positive and negative examples to change [15]. BirgRank (BI-Relational Graph page RANK) constructs a bi-relational graph with a given network and a GO hierarchy, connecting each gene to every GO term for which it has an annotation. It applies Random Walk with Restarts (RWR) [34] to diffuse the annotation information across this network [9].

Local. Each gene's score for a GO term is the weighted average of the scores of its neighbors. Local mimics a basic procedure for assigning GO term annotations to a query gene q: use BLAST to determine if q's sequence is similar to that of a gene q' in another organism and then transfer the annotations of q' to q.

GeneMANIA [4]. Given a weighted, undirected network $G = (V, E, w)$, and a label vector $\mathbf{y}$ where $y(u)$ represents the prior evidence for gene u having a function of interest, this algorithm computes a discriminant score $s(u)$ between -1 and 1 to each gene u in the network, which we can threshold to classify the genes. The value of $y(u)$ is 1 or -1 if u is a positive or negative example, respectively. Let the number of positive examples be n^+ and the number of negative example be n^-. If u is an unknown example, then $y(u) = \frac{n^+ - n^-}{n^+ + n^-}$, the mean of the labels of the labeled nodes. To compute the vector containing the discriminant scores for each node, we solve the following optimization problem:

$$\mathbf{s} = \arg\min_{\mathbf{s}} \left(\sum_{u \in V} (s(u) - y(u))^2 + \sum_{(u,v) \in E} w_{uv} (s(u) - s(v))^2 \right), \tag{7}$$

where the minimization ranges over all vectors in $\mathbb{R}^m$ and m is the number of nodes in the graph. Let $\mathbf{W} \in \mathbb{R}^{m \times m}$ denote the adjacency matrix of G, and $\mathbf{P} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ denote the normalized network, where $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}_{uu} = \sum_v w_{uv}$. To minimize the sum in Equation (7), we solve the linear system of equations $(\mathbf{I} + \mathbf{D} - \mathbf{P})\mathbf{s} = \mathbf{y}$.

BirgRank [9]. While BirgRank is also a network propagation algorithm, it has significant differences from SinkSource and GeneMANIA. As we describe below, the first difference is that it directly incorporates the GO hierarchy into the diffusion process. Second, BirgRank propagates labels on a gene-by-gene basis, meaning it makes all function predictions for a single gene simultaneously. Third, it does not incorporate negative examples.

BirgRank utilizes three datasets: i) a weighted, undirected network G represented as the column normalized adjacency matrix $\mathbf{W} \in \mathbb{R}^{m \times m}$ where m is the number of genes; ii) a set of gene-function annotations represented in a binary matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ where n is the number of functions, and $R_{ij} = 1$ if gene i is annotated by function j, and is 0 otherwise; and iii) a function hierarchy represented by a binary matrix $\mathbf{H} \in \mathbb{R}^{n \times n}$ where $H_{ij} = 1$ if the term i is a child of term j, and is 0 otherwise. BirgRank combines these three matrices into a single matrix in order to incorporate the hierarchy $\mathbf{H}$ into the propagation. BirgRank performs random walks with restarts (RWR) controlled by four parameters: (a) α is the standard parameter for the restart probability, (b) μ controls the proportion of transition along edges in $\mathbf{W}$ versus along edges in $\mathbf{R}$ (that connect genes in G to the terms annotating them in H), (c) θ controls the probability of restart from a given gene versus the probability of restart from the functions annotated to it, and (d) λ controls the direction of transitions within the hierarchy (i.e., up or down) with $\mathbf{H}^* = \lambda\mathbf{H} + (1-\lambda)\mathbf{H}^T$.

To compute the prediction scores, BirgRank solves the following system of linear equations:

$$\left(\begin{bmatrix} \mathbf{I}_m & 0 \\ 0 & \mathbf{I}_n \end{bmatrix} - \alpha \overline{\begin{bmatrix} \mu\mathbf{W} & 0 \\ (1-\mu)\mathbf{R}^T & \mathbf{H}^* \end{bmatrix}}\right) \begin{bmatrix} \mathbf{X}_W \\ \mathbf{X}_H \end{bmatrix} = (1-\alpha) \begin{bmatrix} \theta\mathbf{I}_m \\ (1-\theta)\mathbf{R}^T \end{bmatrix}, \tag{8}$$

where the bar over the block matrix indicates the the whole matrix is column normalized, and I_n and I_m are $n \times n$ and $m \times m$ identity matrices, respectively. Each column vector $\mathbf{v}$ in $\mathbf{X}_W \in \mathbb{R}^{m \times m}$ contains the RWR scores for a single gene, i.e., $\mathbf{X}_{Wuv}$ is the probability that a random walker who restarts to gene v will visit gene u. Each column vector in the lower block of the solution, $\mathbf{X}_H \in \mathbb{R}^{n \times m}$, contains the RWR scores from a single gene to the nodes corresponding to GO terms, i.e., $\mathbf{X}_{Huv}$ is the probability of the random walker who restarts to gene v or to v's annotations in $\mathbf{R}$ will visit the term u. We derive the function predictions scores for each gene from $\mathbf{X}_H$.

In the original BirgRank paper [9], the annotation matrix $\mathbf{R}$ contained only the direct annotations. Since we evaluated on a term-by-term basis, we opted to include all propagated annotations in $\mathbf{R}$, i.e., if a gene was annotated to a GO term, we include annotations of that gene to all ancestors of that term in the GO hierarchy.

The original MATLAB implementation of BirgRank solved for $\mathbf{X}_W$ and $\mathbf{X}_H$ directly, i.e., by computing the inverse of the matrix on the left of Equation (8) and multiplying by the vector on the right, thus making predictions for all nodes. We were able to greatly reduce the running time of BirgRank by computing scores for only the left-out positive and negative examples in LOSO validations. To select the parameters for BirgRank, we systematically varied them and performed LOSO evaluation (BP EXPC annotations, SSN+STRING network). We selected the values that gave the highest median $F_{\max}$: $\alpha = 0.9$, $\mu = 0.5$, $\theta = 0.5$, and $\lambda = 0.01$ (Appendix A.2).

Implementation

We implemented each of these algorithms (SinkSource, FastSinkSource, Local, GeneMANIA, and BirgRank) in Python using the SciPy (v1.1.0) and NumPy (v1.15) libraries. For GeneMANIA and BirgRank, we translated their MATLAB implementations on a line-by-line basis to SciPy sparse matrix operations. We ensured the correctness of our implementations by comparing our output values to those output by the corresponding MATLAB implementation, using the 19-species SSN and BP EXPC annotations. They matched exactly. To solve the GeneMANIA linear system, we utilized the conjugate gradient (CG) solver implemented in the SciPy (v1.1.0) Python package and used SciPy's default tolerance cutoff of 1×10^{-5}. To compute the scores

for BirgRank, we used power iteration and stopped when the maximum difference of node scores between iterations i and $i-1$ (i.e., ϵ) was $\leq 1 \times 10^{-4}$).

Datasets

Gene Ontology Annotations

We obtained GO [3] annotations for the genes in the 200 bacterial species from the UniProt-GOA database ("goa_uniprot_all.gaf.gz" Downloaded on September 18, 2017). We considered three sets of evidence codes: (i) Experimental as well as those used in the CAFA evaluations [18] (EXPC): EXP, IDA, IPI, IMP, IGI, IEP, TAS, IC; (ii) Computational Analysis (COMP): ISS, ISO, ISA, ISM, IGC, IBA, IBD, IKR, IRD, RCA; and (iii) Electronic Analysis (ELEC): IEA.

Positive, Negative, and Unknown Examples

We say a given gene g is *directly* annotated to a GO term t if this annotation appears in the annotations (GAF) file. For a given t, we defined g as

(a) a *positive example* if g was directly annotated to t or to a descendant of t in the GO DAG,

(b) a *negative example* if g was not directly annotated to t or to an ancestor or descendant of t in the GO DAG, but also had at least one other annotation, and

(c) an *unknown example* otherwise.

This approach is commonly used in the literature [4]. We used only the "is_a" edge type when defining these positive and negative examples. In several analyses, we restricted our attention to a specific set of evidence codes, e.g., EXPC. In such a situation, we discarded all direct annotations with other evidence codes before computing the different sets of examples.

We utilized the negative examples in different aspects of this work: (a) for algorithms that use negatives examples (SinkSource and GeneMANIA), (b) for integrating multiple networks (see "Network Integration"), and (c) for defining false positives when calculating the $F_{\max}$ during evaluation (see "Leave-One-Species-Out Validation").

Sequence Similarity Network

We created a network based upon sequence similarity of the putative proteomes of 19 clinically-relevant pathogenic bacteria. We selected these species because they have varying levels of annotations (Table S1) and they are somewhat phylogenetically diverse (Figure S4). We obtained the protein sequences of the reference proteome of each species from UniProt [2] (downloaded on June 14, 2018). In instances where multiple UniProt reference strains were available for a pathogen, we chose the strain with the largest number of GO annotations (see Table S1 for the taxonomy IDs and the number of annotations). We ran all-vs-all BLASTP using the default parameter values and the protein sequences in the 19 species as the database. We processed the results by retaining the weaker score for all reciprocated matches, removing self-comparisons, and using the absolute value of the base-10 logarithm of the E-value as the edge weight. For edges where the E-value was 0, we assigned a weight of 180, which is the absolute value of base-10 logarithm of the smallest non-zero E-value observed (rounded).

We tested various E-value cutoffs from 1×10^{-25} to 50. If the E-value cutoff was larger than one, we added the base-10 logarithm of the cutoff to every edge weight to ensure that it was positive. See Table S2 for the sizes of these networks.

For the analysis with 200 species, we selected the bacteria with the largest number of EXPC or COMP annotations, which ranged from six to 16,685 for *Gluconobacter oxydans* and *Escheria coli*, respectively. We built the 200-species SSN using the same steps as the 19 species SSN, with an E-value cutoff of 0.1.

STRING Networks

We integrated species-specific STRING functional association networks for 14 of the 19 bacterial species which had networks available in the STRING database (v10.5, downloaded on September 18, 2017) [19]. We converted the STRING IDs to UniProt IDs using the mappings available from UniProt. STRING assigns edge weights based on multiple lines of evidence of association including physical binding, gene expression, and orthology mapping. We utilized six STRING networks: neighborhood, fusion, cooccurence, coexpression, experimental, and database. We found including additional STRING networks did not improve the quality of results (results not shown). We also tested various cutoffs of the STRING combined edge scores including 150, 400, 700, and 900 (low, medium, high and very high stringency). See Table S2 for network sizes.

Network Integration

We compared two kernel-based network integration methods.

GM-2008: Mostafavi and Morris integrated multiple networks on a GO term-by-GO term basis using ridge regression [4]. Given m networks represented as adjacency matrices $W_1, ..., W_m$, they combined them using a weighted sum $\mathbf{W}^* = \sum_{i=0}^{m} \mu_i W_i$, where they computed the network weights $\mu^* = [\mu_1, \mu_2, ..., \mu_m]$ independently for each prediction task (i.e., one set of weights for each GO term) by solving the following constrained linear regression problem:

$$\mu^* = \min_{\mu} \left(\mathbf{t} - \mathbf{\Omega}\mu\right)^T \left(\mathbf{t} - \mathbf{\Omega}\mu\right),$$
$$\text{s.t. } \mu_i \geq 0 \quad i = \{1, ..., m\}.$$

We define the terms in this equation next. If n^+ and n^- are the number of positive and negative examples, respectively, and $n = n^+ + n^-$ is the total number of labelled examples, then the target vector $\mathbf{t}$ contains $(n^+)^2 + n^+n^-$ entries, one for each positive-positive example pair and one for each positive-negative example pair. The value in $\mathbf{t}$ is $\frac{n^-}{n}^2$ for each positive-positive example pair and $-\frac{n^+n^-}{n^2}$ for each positive-negative example pairs. The matrix Ω has m columns, one for each of the networks and $(n^+)^2 + n^+n^-$ rows, corresponding to the positive-positive and positive-negative pairs. The resulting vector μ^* contains the relative importance of each network, based on how well its edge weights match the positive pairs versus the positive-negative pairs.

SWSN: Mostafavi and Morris extended their method to compute optimal weights for multiple related GO terms simultaneously (Simultaneous Weights, SW) [35]. A drawback of this approach was that it treated all non-positive examples as negative examples. Youngs *et al.* later modified the SW method to be able to use specific negative examples for each GO term (Simultaneous Weights with Specific Negatives, SWSN) [20]. The SWSN method assigns network weights by solving the following problem:

$$\mu^* = \min_{\mu} \sum_{c=1}^{h} \left(\mathbf{t}_c - \mathbf{\Omega}\mu\right)^T \left(\mathbf{t}_c - \mathbf{\Omega}\mu\right),$$

where the index c runs over a set of related GO terms. They found that computing the weights to all GO terms in a particular hierarchy (e.g., BP or MF) performed better than any other grouping of functions. Therefore, we also grouped GO terms by hierarchy to compute the optimal weights for our networks.

Leave-One-Species-Out Validation

To evaluate and compare the predictive performance of these algorithms, rather than use standard cross-validation techniques, we opted to evaluate predictions on a genome-wide scale using a Leave-One-Species-Out (LOSO) validation method that mimicked the challenge of making large-scale functional predictions for the genome of a newly-sequenced species where none of the genes have any annotations. For this evaluation, we left out the annotations of all genes for each species in turn, and assessed how well we could recover them

by applying each algorithm on the annotations of the other organisms (see Figure 5b). When evaluating predictions for a given left-out species, we calculated true positives and false positives from the positive and negative examples of that species. To compare algorithms, we used the $F_{\max}$ value, which is the maximum of the harmonic mean of the precision and recall over the entire precision-recall curve.

Software and Dataset Availability

A Python implementation of all algorithms, the SSN and STRING networks, and the GO annotations used in this research are available for download at http://bioinformatics.cs.vt.edu/~jeffl/supplements/2019-fastsinksource.

Supplementary Materials

A Supplementary Text

A.1 Properties of the FastSinkSource Algorithm

To keep this section self-contained, we start by reformulating the FastSinkSource algorithm. Recall that the input to FastSinkSource is a weighted undirected graph $G = (V, E, w)$, where $w_{uv} \geq 0$ represents the weight of the edge (u, v). In addition, for a GO term τ, we have a partition of the nodes in V into three sets: V^+, V^- and V^0 positive, negative, and unknown examples, respectively. We also have a parameter $0 < \alpha < 1$ in the input. Note that we do not permit $\alpha = 1$ here since some of our lemmas and proofs do not apply to this value.

We fix the score of every positive example at 1 and for every negative example at 0. The goal is to compute a score $s(u)$ between 0 and 1 for each node u that is an unknown example by propagating the positive and negative labels across G using the following equation:

$$s(u) = \frac{\alpha \sum\limits_{v \in N(u)} w_{uv} s(v)}{d(u)} = \frac{\alpha \sum\limits_{v \in N^0(u)} w_{uv} s(v)}{d(u)} + f(u) \tag{9}$$

where $N(u)$ is the set of neighbors of node u, $N^0(u)$ is the subset of neighbors of node u that are unknown examples, $d(u)$ is the weighted degree of node u in G, and

$$f(u) = \frac{\alpha \sum\limits_{v \in N^+(u)} w_{uv}}{d(u)},$$

where $N^+(u)$ is the set of neighbors of u that are positive examples. Note that $f(u) \geq 0$ is a constant for every node u.

We define $\mathbf{s}$ to be a $|V^0| \times 1$ vector formed by the node scores of all unknown examples and $\mathbf{f}$ to be a $|V^0| \times 1$ constant vector in which the uth element is $f(u)$. Further, we let the $|V^0| \times |V^0|$ matrix $\mathbf{P}$ contain the degree-normalized edge weights in G among pairs of nodes in V^0, i.e., $P_{uv} = w_{uv}/d(u)$ for every pair of unknown examples (u, v). Then for every unknown example u, we can combine the score equations (9) into a single linear system:

$$\mathbf{s} = \alpha \mathbf{P} \mathbf{s} + \mathbf{f}. \tag{10}$$

We use power iterations to solve this system of linear equations. Let $s(u, i)$ denote the score computed for node u after i iterations and let $\mathbf{s}^{(i)}$ denote the vector formed by these scores after i iterations. We initialize by $s(u, 0) = 0$ for every node u in G, i.e., we set $\mathbf{s}^{(i)} = 0$. For every node u, we compute its score in iteration $i + 1$ using the equation

$$s(u, i+1) = \alpha \sum_{v \in N^0(u)} P_{uv} s(v, i) + f(u) \tag{11}$$

In other words, we have that

$$\mathbf{s}^{(i+1)} = \alpha \mathbf{P} \mathbf{s}^{(i)} + \mathbf{f}. \tag{12}$$

We first prove that the score for every node u does not decrease from one iteration to the next, is a lower bound on $s(u)$, and reaches this value in the limit. We say that a vector is *non-negative* if every element in it is non-negative. Given two vectors $\mathbf{x}$ and $\mathbf{y}$, we use $\mathbf{x} \leq \mathbf{y}$ to denote that $\mathbf{y} - \mathbf{x}$ is non-negative.

Lemma A.1. *For every node $u \in V$, $s(u, i) \leq s(u, i+1) \leq s(u)$ for every $i \geq 1$.*

Proof. Combining Equation (11) with the fact that no entry in $\mathbf{P}$ is negative, we see that

$$s(u, i+1) - s(u, i) = \alpha \sum_{v \in N^0(u)} P_{uv} \big(s(v, i) - s(v, i-1) \big) \geq 0,$$

Moreover, by applying induction to eq. (5), we can prove that

$$\mathbf{s}^{(i)} = \sum_{j=0}^{i} (\alpha \mathbf{P})^j \mathbf{f} \ \leq \ \sum_{j=0}^{\infty} (\alpha \mathbf{P})^j \mathbf{f}, \tag{13}$$

where the last inequality follows from the fact that every entry in any power of $\mathbf{P}$ is non-negative. We can prove that for every $j \geq 1$, every row sum in $\mathbf{P}^j$ is bounded from above by one. Therefore, in combination with $0 < \alpha < 1$, we have $\lim_{j\to\infty}(\alpha\mathbf{P})^j = 0$. We can further show that $\mathbf{I} - \alpha\mathbf{P}$ is invertible and that this inverse equals the infinite sum on the right-hand size of Equation (13). Moreover, since $\mathbf{I} - \alpha\mathbf{P}$ is invertible, we can solve eq. (10) for $\mathbf{s}$ to obtain $\mathbf{s} = (\mathbf{I} - \alpha\mathbf{P})^{-1}\mathbf{f}$. We conclude that for every $i \geq 1, \mathbf{s}^{(i)} \leq \mathbf{s}$, which proves the inequality on the right-hand side of the lemma. □

This proof implies the following corollary to the lemma.

Corollary A.1. *For every node $u \in V$, $\lim_{i\to\infty} s(u,i) = s(u)$.*

We now prove an upper bound on $s(u,i)$ for every node u. In the following lemma and in its proof, we use $\|\mathbf{f}\|$ to denote the largest value in $\mathbf{f}$. For a matrix $\mathbf{A}$, we use $\|\mathbf{A}\|$ to denote the largest row sum in A. Note that $\|\mathbf{P}\| = 1$ and that $\|\mathbf{P}\mathbf{s}\| \leq \|\mathbf{P}\|\|\mathbf{s}\|$.

Lemma A.2. *For every node $u \in V$ and for every $i \geq 1$,*

$$s(u,i+1) \leq s(u,i) + \alpha^i \|\mathbf{f}\| \text{ and}$$
$$s(u) \leq s(u,i) + \frac{\alpha^i \|\mathbf{f}\|}{(1-\alpha)}$$

Proof. For every $i \geq 1$, Equation (12) implies that

$$\mathbf{s}^{(i+1)} - \mathbf{s}^{(i)} = \alpha\mathbf{P}(\mathbf{s}^{(i)} - \mathbf{s}^{(i-1)})$$

By computing the maximum value on both sides, we have

$$\begin{aligned}\|\mathbf{s}^{(i+1)} - \mathbf{s}^{(i)}\| &= \|\alpha\mathbf{P}(\mathbf{s}^{(i)} - \mathbf{s}^{(i-1)})\| \\ &\leq \|\alpha\mathbf{P}\| \cdot \|\mathbf{s}^{(i)} - \mathbf{s}^{(i-1)}\| \\ &= \alpha\|\mathbf{s}^{(i)} - \mathbf{s}^{(i-1)}\|\end{aligned}$$

By combining induction with this inequality, we can prove that

$$\|\mathbf{s}^{(i+1)} - \mathbf{s}^{(i)}\| \leq \alpha^i \|\mathbf{s}^{(1)} - \mathbf{s}^{(0)}\| = \alpha^i \|\mathbf{f}\| \tag{14}$$

Thus, we have shown that the for every node, the difference between its scores in iterations $i+1$ and i decreases geometrically with i, thereby proving the first inequality in the lemma. To complete the proof by demonstrating the second inequality, we need to relate the node's score at convergence to its score after i iterations. We do so by proving an upper bound on how much the node's score can increase after i iterations.

For every integer $m > i$, we have

$$\begin{aligned}\|\mathbf{s}^{(m)} - \mathbf{s}^{(i)}\| &= \|\sum_{j=i}^{m-1} (\mathbf{s}^{(j+1)} - \mathbf{s}^{(j)})\| \\ &\leq \sum_{j=i}^{m-1} \|\mathbf{s}^{(j+1)} - \mathbf{s}^{(j)}\| \\ &\leq \sum_{j=i}^{m-1} \alpha^j \|\mathbf{f}\| \qquad \text{by Equation (14)} \\ &= \alpha^i \|\mathbf{f}\| \sum_{j=0}^{m-i-1} \alpha^j = \alpha^i \|\mathbf{f}\| \Big(\frac{1-\alpha^{m-i}}{1-\alpha}\Big)\end{aligned}$$

Since $\lim_{m\to\infty} s(u,m) = s(u)$ by Corollary A.1, we have that

$$\begin{aligned}\|\mathbf{s}-\mathbf{s}^{(i)}\| &= \lim_{m\to\infty} \|\mathbf{s}^{(m)}-\mathbf{s}^{(i)}\| \\ &\le \alpha^i \|\mathbf{f}\| \lim_{m\to\infty} \left(\frac{1-\alpha^{m-i}}{1-\alpha}\right) \le \frac{\alpha^i\|\mathbf{f}\|}{(1-\alpha)},\end{aligned}$$

which completes the proof. □

We now consider how the node scores after each iteration change when we decrease the value of α. We first prove that as α decreases, the score for each node after i iterations also decreases. Then we prove that as α decreases, the difference in node scores from one iteration to the next also decreases, which means that fewer iterations are necessary for convergence. We introduce an additional piece of notation here. Let $s_\beta(u,i)$ be the score of node u after i iterations of SinkSource for a given GO term and with the value of α set to β.

Lemma A.3. *Let $0 < \beta < \gamma \le 1$. Then for every node $u \in V$ and for every integer $i \ge 1$, $s_\beta(u,i) \le s_\gamma(u,i)$.*

Proof. We prove the lemma by induction. To prove the base that $s_\beta(u,1) \le s_\gamma(u,1)$, we observe that

$$\begin{aligned}s_\beta(u,1) &= \beta \sum_{v\in N^0(u)} P_{uv} s_\beta(v,0) + \beta \sum_{v\in N^+(u)} p_{uv}, \text{ by Equation (11)} \\ &= \beta \sum_{v\in N^+(u)} p_{uv}, \text{ since } s_\beta(v,0) = \text{for every node } v \in V\end{aligned}$$

Similarly,

$$s_\gamma(u,1) = \gamma \sum_{v\in N^+(u)} p_{uv}$$

Since $\beta < \gamma$, we have

$$s_\beta(u,1) \le s_\gamma(u,1)$$

The inductive hypothesis is that for some integer $i-1 \ge 1$, $s_\beta(u,i-1) \le s_\gamma(u,i-1)$ for every node $u \in V$. We have that

$$\begin{aligned}s_\beta(u,i) &= \beta \sum_{v\in N^0(u)} P_{uv} s_\beta(v,i-1) + \beta \sum_{v\in N^+(u)} p_{uv} \\ &\le \beta \sum_{v\in N^0(u)} p_{uv} s_\beta(v,i-1) + \gamma \sum_{v\in N^+(u)} p_{uv}, \text{ since } s_\beta(u,1) \le s_\gamma(u,1) \\ &\le \beta \sum_{v\in N^0(u)} p_{uv} s_\gamma(v,i-1) + \gamma \sum_{v\in N^+(u)} p_{uv}, \text{ by the inductive hypothesis} \\ &< \gamma \sum_{v\in N^0(u)} p_{uv} s_\gamma(v,i-1) + \gamma \sum_{v\in N^+(u)} p_{uv}, \text{ since } \beta < \gamma \\ &= s_\gamma(u,i)\end{aligned}$$

thus proving the statement for i and completing the proof. □

We can also show that the difference between a node's score in iterations i and $i-1$ itself decreases as the value of α becomes smaller.

Lemma A.4. *Let $0 < \beta < \gamma \le 1$. Then for every node $u \in V$ and for every integer $i \ge 1$,*

$$s_\beta(u,i) - s_\beta(u,i-1) \le s_\gamma(u,i) - s_\gamma(u,i-1).$$

Proof. Consider any integer $i \geq 1$. By Equation (11), we have

$$
\begin{aligned}
s_\beta(u,i) - s_\beta(u,i-1) &= \beta \sum_{v \in N^0(u)} P_{uv} s_\beta(v,i-1) - \beta \sum_{v \in N^0(u)} P_{uv} s_\beta(v,i-2), \\
&\quad \text{since the contributions from the positively-labeled neighbors cancel each other} \\
&< \gamma \sum_{v \in N^0(u)} P_{uv} s_\beta(v,i-1) - \gamma \sum_{v \in N^0(u)} P_{uv} s_\beta(v,i-2), \text{ since } \beta < \gamma \\
&\leq \gamma \sum_{v \in N^0(u)} P_{uv} s_\gamma(v,i-1) - \gamma \sum_{v \in N^0(u)} P_{uv} s_\gamma(v,i-2), \text{ by Lemma A.3} \\
&= s_\gamma(u,i) - s_\gamma(u,i-1),
\end{aligned}
$$

thus completing the proof. □

A.2 Parameter Selection for BirgRank

To find the best parameters for BirgRank, we performed LOSO evaluation (BP EXPC annotations, SSN+STRING network with E-value < 0.1) and varied the parameters α, μ, and λ, which control the RWR restart probability, proportion of the flow within G vs. to H, percentage of restart to a given node vs. its annotations, and the direction of flow within the GO hierarchy, respectively. We did not need to vary the parameter θ since LOSO evaluation removed all annotations from the nodes on which we ran BirgRank. We started by varying μ with $\alpha = 0.5$ and $\lambda = 0.5$ and observed that μ had a minor effect on the median $F_{\max}$ (green points in Figure S1a). Hence, we decided to set $\mu = 0.5$ for further analysis. Next we observed that varying α, with $\lambda = 0.5$, resulted in a non-monotonic and considerable change in the median $F_{\max}$ (blue points in Figure S1a). While varying λ on the other hand, with $\alpha = 0.5$, we observed the $F_{\max}$ decreased monotonically with increase in λ (red points in Figure S1a). Therefore, we selected $\lambda = 0.01$ for further analysis. With $\lambda = 0.01$ and $\mu = 0.5$, we varied α and observed the highest $F_{\max}$ of 0.45 with $\alpha = 0.9$ (blue points in Figure S1b. We selected this value of α and repeated our experiments with λ and with μ to test our earlier settings for these parameters. We confirmed that varying μ had a marginal effect (green points in Figure S1b) and that $\lambda = 0.01$ maximized the $F_{\max}$ (red points in Figure S1b).

Note that the value of $\lambda = 0.01$ corresponds to propagating from parents to children in the GO DAG almost entirely. In contrast, Jiang *et al.* did not observe an appreciable effect of this parameter on their results. This difference could be due to the fact that we did not use only the direct annotations as Jiang *et al.* did in their experiments; instead, we propagated all annotations up the GO hierarchy to facilitate term-by-term evaluations.

BirgRank did not perform better than the other propagation methods in our LOSO evaluations. One of the evaluations performed by Jiang *et al.* involved withholding all annotations from a given percentage of proteins, which is similar to our LOSO validation. They observed that in this evaluation, their methods (BirgRank and AptRank) were not able to outperform GeneMANIA and other propagation methods, which parallels our results.

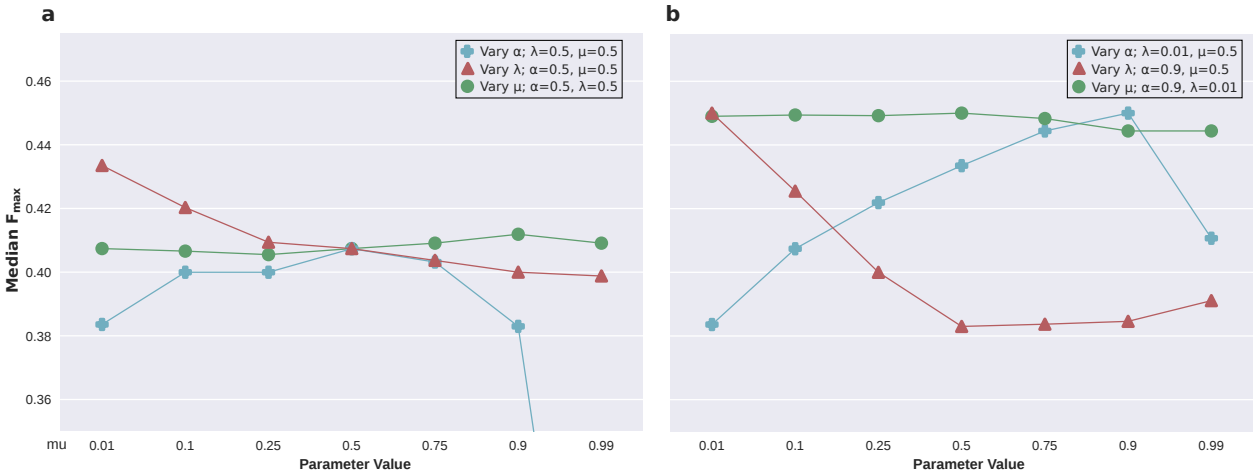

Figure S1: BirgRank parameter comparison of the $F_{\max}$ results of the LOSO evaluation of 19 bacterial species SSN (E-value <= 0.1) + STRING using BP EXPC annotations. Related to Figure 2. (**a**) We first fixed every pair of parameters at 0.5 and then tested a range of values for the third parameter. (**b**) We then set $\lambda = 0.01, \mu = 0.5$ and varied α, then set $\alpha = 0.9, \mu = 0.5$ and varied λ, and finally set $\alpha = 0.9, \lambda = 0.01$ and varied μ.

B Supplementary Tables

NCBI Taxon ID	Species Name	# BP & MF Annotations	# Genes All Ev. Codes	# Genes EXPC+COMP	# Genes EXPC
83333	*E. coli*	25927	3747	3056	2811
83332	*M. tuberculosis*	26529	5248	1749	1356
208964	*P. aeruginosa*	20009	4085	1974	922
85962	*H. pylori*	4955	999	113	106
99287	*S. typhimurium*	17568	3365	1116	91
243277	*V. cholerae*	14103	2523	2038	75
632	*Y. pestis*	29728	6510	874	62
242231	*N. gonorrhoeae*	6396	1294	32	32
93061	*S. aureus*	8998	1772	552	22
71421	*H. influenzae*	8286	1387	502	18
272563	*C. difficile*	11182	2480	10	10
301447	*S. pyogenes*	5563	1173	7	6
441771	*C. botulinum*	10433	2260	597	3
257313	*B. pertussis*	10026	2368	8	2
272620	*K. pneumoniae*	16504	3720	5	1
300267	*S. dysenteriae*	12491	2780	6	0
509170	*A. baumannii*	8222	1693	0	0
269482	*B. cepacia*	17573	4341	3	0
333849	*E. faecium*	9272	1991	0	0

Table S1: Overview of number of annotations per species. The columns with "# Genes" in the title show how many genes have at least one BP or MF annotation with the given evidence code.

SSN E-value cutoff	STRING cutoff	# Nodes	# Edges
1×10^{-25}	-	62,226	701,966
1×10^{-25}	400	71,947	1,877,431
1×10^{-15}	-	65,825	1,121,470
1×10^{-6}	-	69,386	1,699,808
0.1	-	72,578	2,452,339
0.1	700	74,857	2,776,002
0.1	400	75,895	3,607,442
0.1	150	75,901	7,857,418
5	-	78,200	2,913,320
20	-	79,783	3,694,193
50	-	79,826	5,116,441

Table S2: Network sizes for various BLAST E-value and STRING cutoffs.

	Time (hrs)	
Algorithm	COMP	ELEC
FastSinkSource	28.4	36.5
GeneMANIA	41.1	51.2
BirgRank	317.9	5483.9
Local	3.2	4.6

Table S3: Running times of each method during LOSO validation. We used the same parameters as shown in Table 1.

C Supplementary Figures

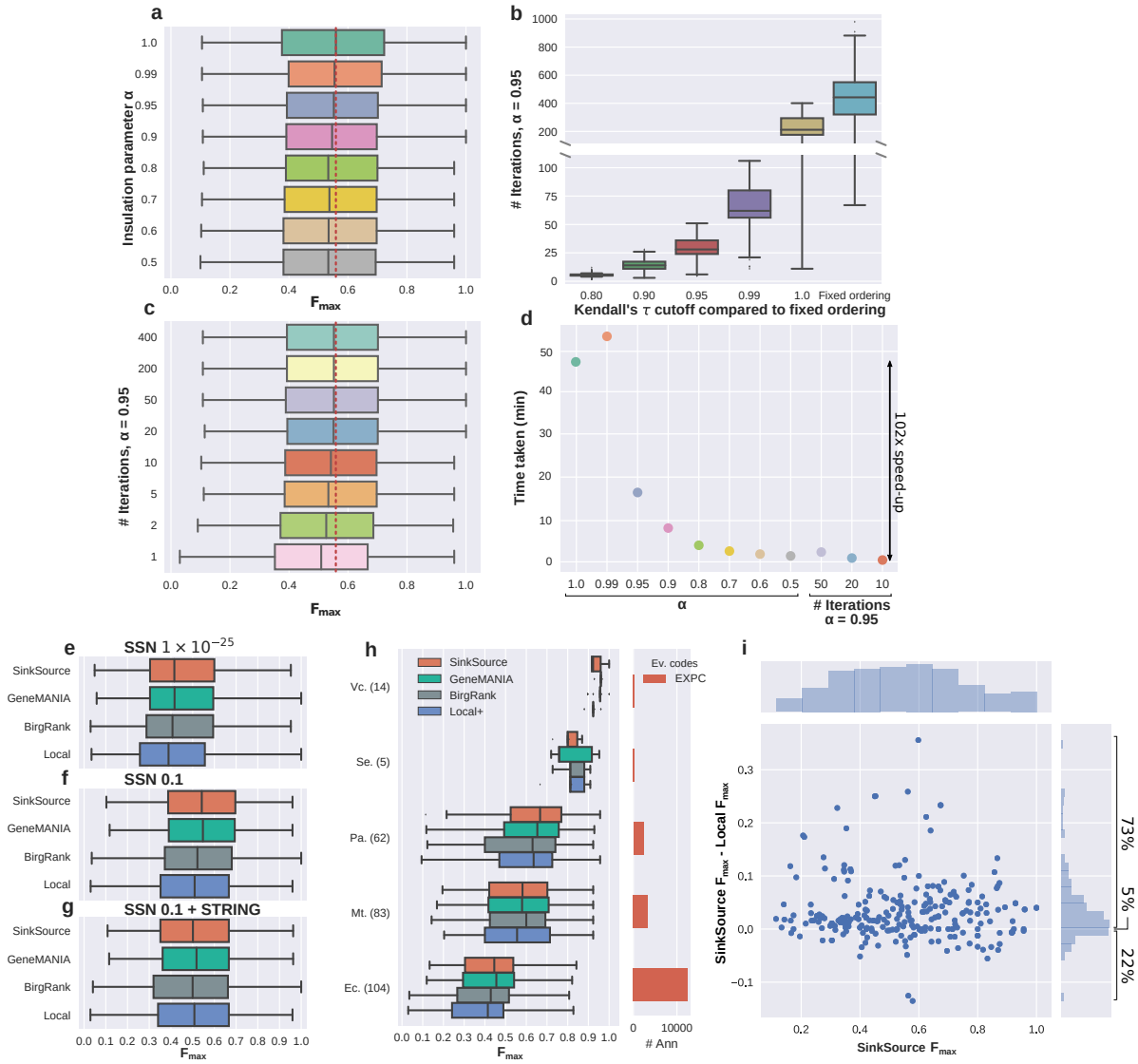

Figure S2: Trade-off between accuracy and speed for SinkSource on the 19 bacterial species SSN (E-value <= 0.1) with 117 MF GO terms (≥ 50 annotations), as well as a comparison of $F_{\max}$ results of four algorithms and three networks for MF terms, for the LOSO evaluation. Related to Figure 1 and Figure 2. (**a**) Variation of $F_{\max}$ distributions with α. The vertical red dotted line represents the median $F_{\max}$ for $\alpha = 1.0$. (**b**) Number of iterations required to fix the rankings of the left-out positives and negatives, or to reach a specified value of Kendall's τ in comparison to the fixed ranking. (**c**) Variation of $F_{\max}$ distributions with the number of iterations ($\alpha = 0.95$). The vertical red dotted line represents the median $F_{\max}$ for $\alpha = 1.0$ in panel (a). (**d**) Total time taken by SinkSource while varying α (shown in a) or the number of iterations with $\alpha = 0.95$ (shown in c). Colors are the same as in a and c. (**e**) SSN (E-value $\leq 1 \times 10^{-25}$), (**f**) SSN (E-value ≤ 0.1), (**g**,**d**,**e**) SSN (E-value ≤ 0.1) integrated with STRING, (**h**) LOSO results for individual species, sorted in decreasing order of median $F_{\max}$. The number of MF GO terms with ≥ 10 annotations appears in parentheses next to each species name. The species name is in bold if the difference between the distributions for SinkSource and Local was statistically significant (rank-sum Bonferroni-corrected p-value < 0.05). The right-hand side shows the number of GO term-annotation pairs with experimental evidence codes for each species. Species names are abbreviated as follows: *Escherichia coli K-12* (Ec), *Mycobacterium tuberculosis* (Mt), *Pseudomonas aeruginosa* (Pa), *Salmonella enterica* (Se), *Vibrio cholerae* (Vc), (**i**) Difference in $F_{\max}$ between SinkSource and Local by the $F_{\max}$ of SinkSource for each GO term.

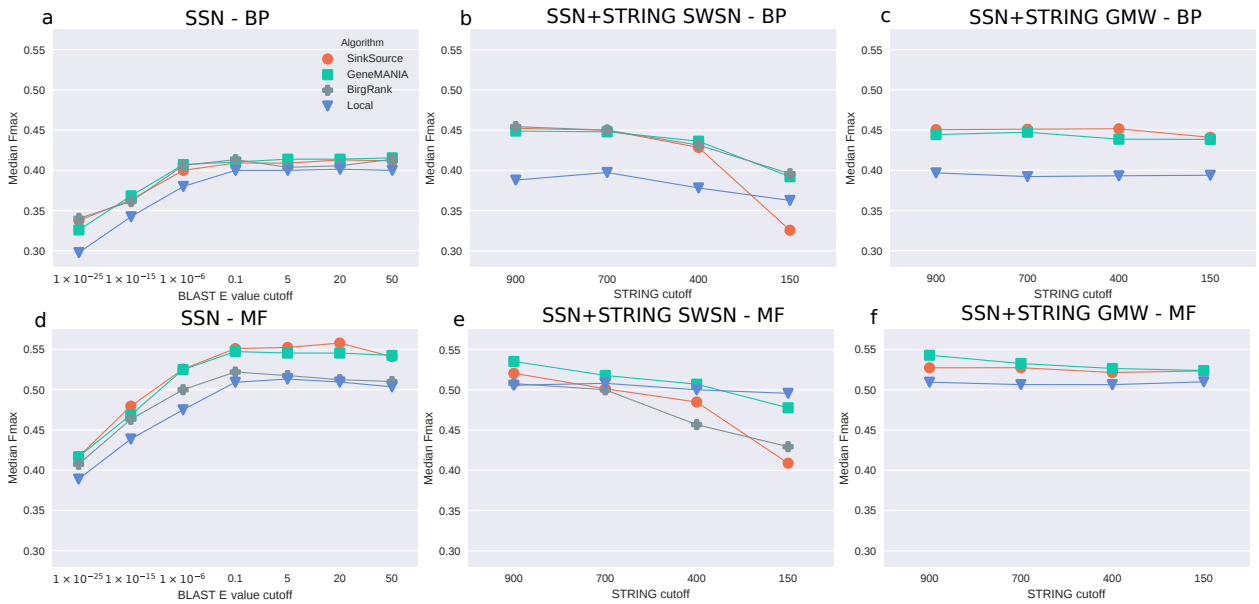

Figure S3: Comparison of the median $F_{\max}$ from the LOSO evaluation of EXPC annotations using various BLAST E-value cutoffs (**a**, **d**) and STRING cutoffs (**c**, **d**, **e**, **f**). Related to Figure 2. (**b**, **e**) SWSN and (**c** and **f**) GMW integration of the STRING networks with the SSN (E-value < 0.1).

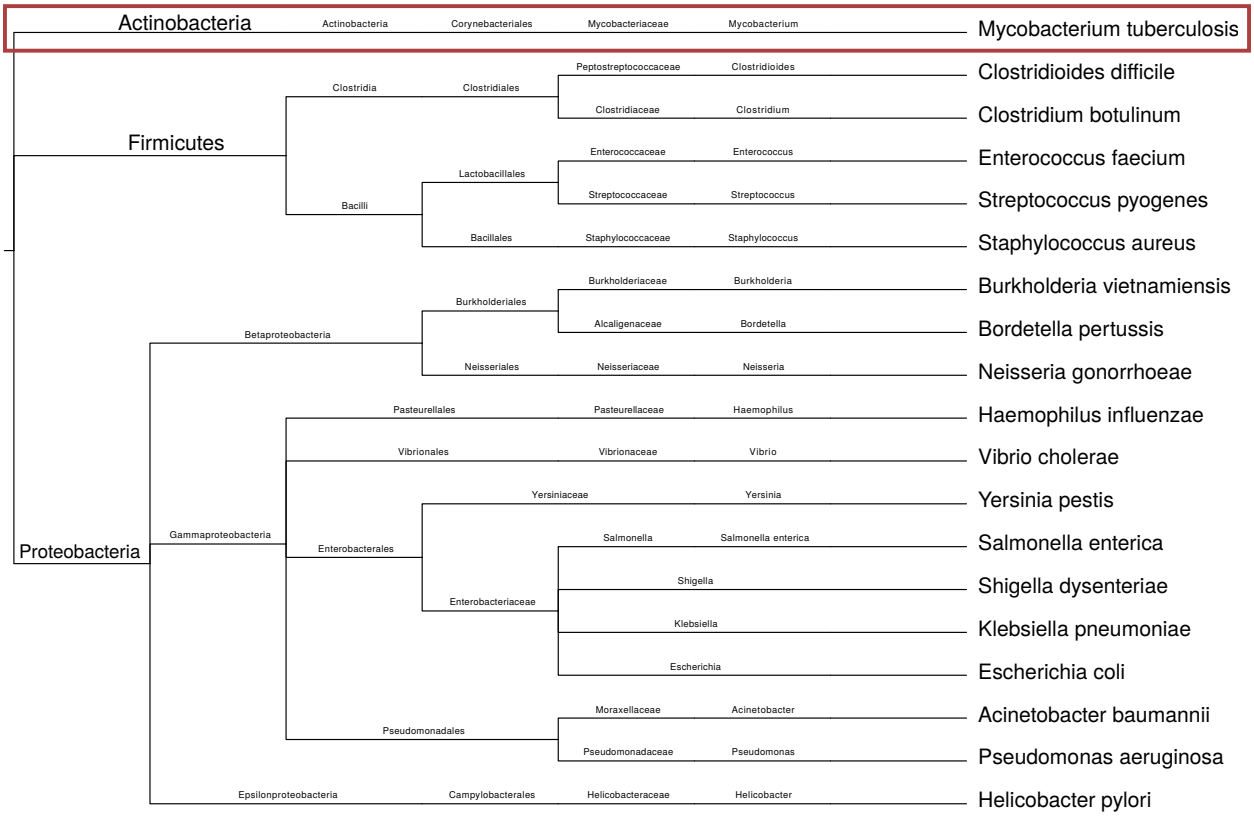

Figure S4: Phylogenetic tree for the 19 organisms used in this study. Related to Figure 2. *M. tuberculosis* is highlighted as it is the most distant of the species. We generated the tree using the FigTree software (http://tree.bio.ed.ac.uk/software/figtree)

.

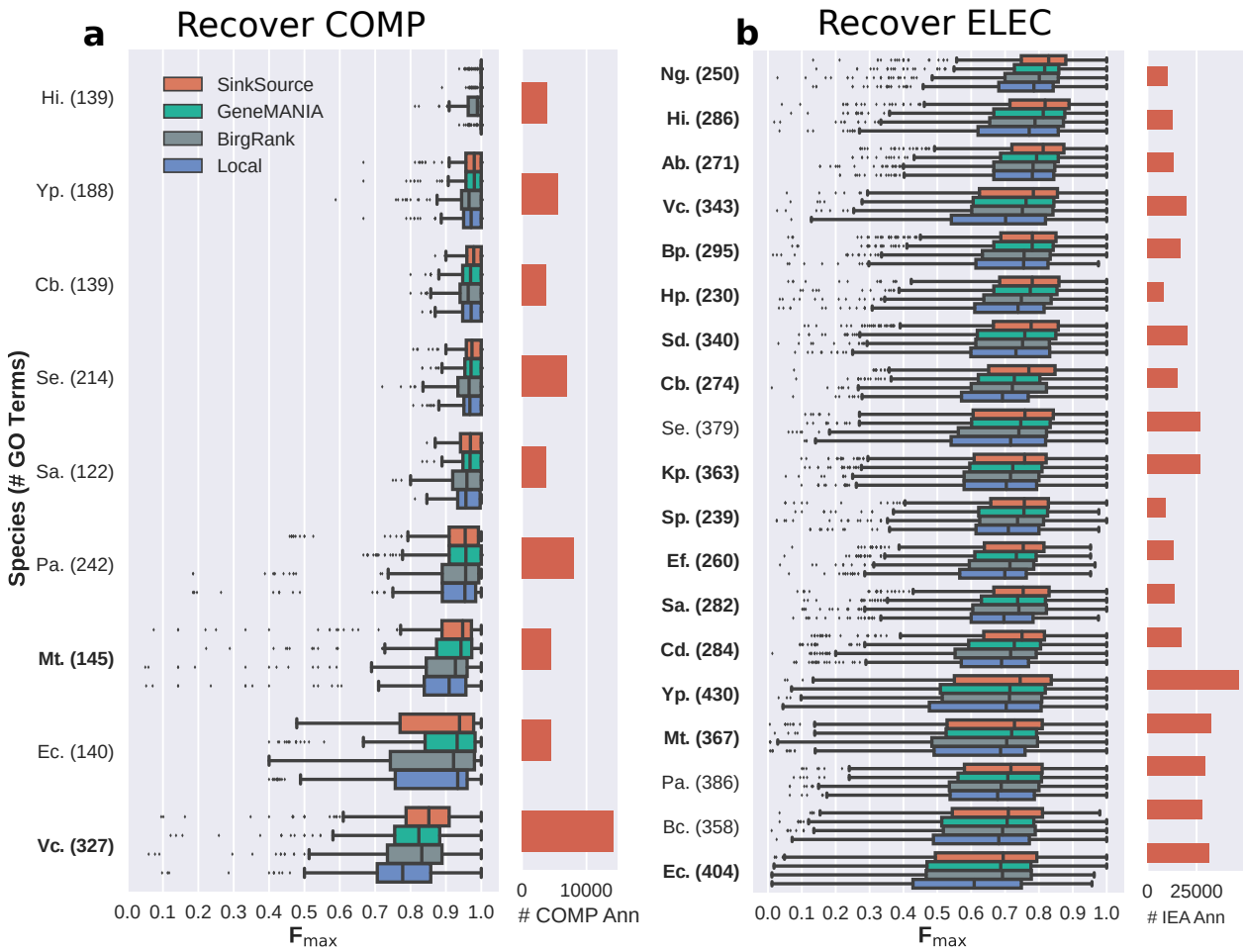

Figure S5: Comparison of $F_{\max}$ results for BP LOSO evaluation for COMP and ELEC evidence codes using the SSN+STRING network. Related to Figure 4. This figure shows the same results as Figure 4a and b, except for all four algorithms. A bold species name indicates that the difference between the distributions for FastSinkSource and Local was statistically significant (rank-sum test Bonferroni-corrected p-value < 0.05). (**a**) Evaluation of recovery of COMP annotations in the left-out species. (**b**) Evaluation of recovery of ELEC annotations in the left-out species. Species name abbreviations defined in Figure 2 and Figure 4.

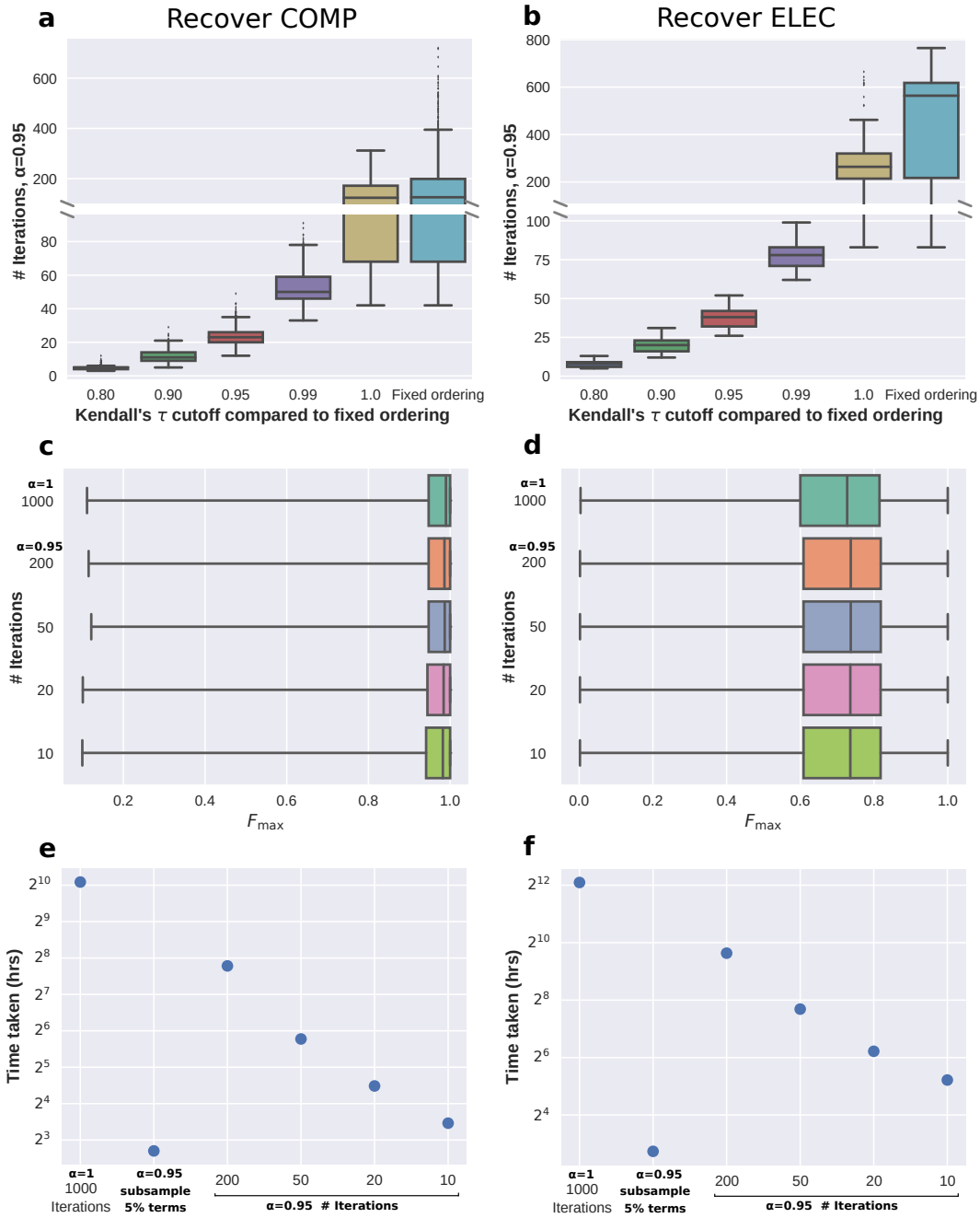

Figure S6: Trade-off between accuracy and speed for SinkSource on the 200 bacterial species SSN (E-value ≤ 0.1) with 511 BP GO terms (≥ 50 annotations) for the LOSO evaluation using EXPC and COMP to make predictions, and either COMP (**a**, **c**, **e**) or ELEC (**b, d, f**) to evaluate in the left-out species. Related to Table 2. (**a** and **b**) Number of iterations required to fix the rankings of the left-out positives and negatives, or to reach a specified value of Kendall's τ in comparison to the fixed ranking. To limit the running time of this analysis, we limited this analysis to 5% of terms sampled uniformly at random, where we sampled evenly between three bins split by the number of annotations per term: (i) 50-200, (ii) 200-500, and (iii) 500+ annotations. (**c** and **d**) Variation of $F_{\max}$ distributions with the number of iterations ($\alpha = 0.95$) as well as $\alpha = 1$ with 1000 iterations. (**e** and **f**) Total time taken by SinkSource (shown in **c** and **d**) for $\alpha = 1$ with 1000 iterations, and while varying the number of iterations with $\alpha = 0.95$. "$\alpha = 0.95$ subsample 5% terms" refers to the time taken for the analyses shown in **a** and **b**. The y-axis is on a $\log_2$ scale.

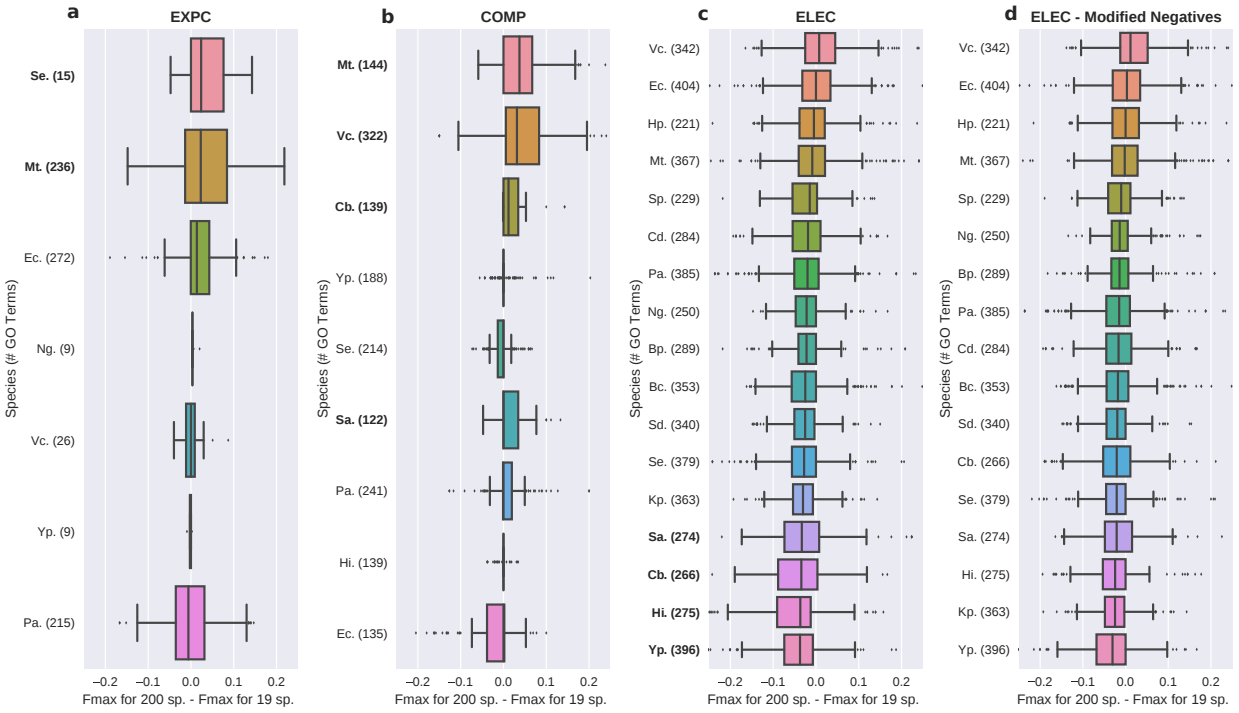

Figure S7: Per-species distributions of the difference between LOSO $F_{\max}$ with the 200 species SSN and with the 19 species SSN when evaluating recovery of (**a**) EXPC, (**b**) COMP, (**c**) ELEC, or (**d**) ELEC with modified negatives. Related to Table 2. We sorted the species by the median difference in $F_{\max}$. A species name is in bold if the median increase or decrease was statistically significant.