FIDDLE: Efficient Assembly of Networks That Satisfy Desired Behavior

Adam A. Butchy
University of Pittsburgh
Cheryl A. Telmer
Carnegie Mellon University
Natasa Miskov-Zivanov (nmzivanov@pitt.edu)
University of Pittsburgh

Research Article

Keywords: Automatic Model Creation , Biological Networks, Extending Biological Networks, Model Construction, Network Reconstruction

Posted Date: June 2nd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-562692/v1

FIDDLE: Efficient assembly of networks that satisfy desired behavior

Adam A. Butchy[1], Cheryl A. Telmer[2], and Natasa Miskov-Zivanov[1,3*]

[1]Department of Bioengineering, University of Pittsburgh, Pittsburgh, PA, USA
(adam.butchy@pitt.edu)
[2]Molecular Biosensor and Imaging Center, Carnegie Mellon University, Pittsburgh, PA, USA
(ctelmer@cmu.edu)
[3]Department of Electrical and Computer Engineering, Department of Computational Biology, University of Pittsburgh, Pittsburgh, PA, USA
(nmzivanov@pitt.edu)
* Corresponding Author

Abstract:

Background: Due to the complexity and redundancy of biological systems, computational models are difficult and laborious to create and update. Therefore, machine reading and automated model assembly are of great interest to computational and systems biologists. Here, we describe FIDDLE (Finding Interactions using Diagram Driven modeL Extension), the tool that we built with the goal to automatically assemble or extend models with the knowledge extracted from published literature. The two main methods developed as part of FIDDLE are called Breadth First Addition (BFA) and Depth First Addition (DFA), and they are based on network search algorithms. To assess the advantages and limitations of BFA and DFA, we applied them on Erdös-Rényi random networks (ER) and Barabási-Albert scale-free networks (BA) as models of biological networks. Starting with several different baseline models and different sets of candidate edges, we conducted a comprehensive evaluation of the BFA and DFA algorithms and their ability to either reconstruct a specified network structure or assemble models that reproduce the same behavior as defined golden models.

Results: When BFA and DFA are applied to random networks, we are able to show that model extension with BFA is more specific and DFA is a more sensitive approach. When applied to scale-free networks, both BFA and DFA demonstrated very limited success.

Conclusion: Results suggest that BFA is a better method for assembling abundant but mixed true and false information, while DFA will better assemble fewer and more truthful interactions. The limited success with scale-free networks suggests a fundamental difficulty with automatically creating biological models due to the inherent redundancy and complexity of biological networks. These methods represent an early attempt and a novel approach to truly

autonomous model creation or extension given different types of knowledge. The source code is available at: https://bitbucket.org/biodesignlab/fiddle/.

Keywords: Automatic Model Creation /Biological Networks/Extending Biological Networks/Model Construction/Network Reconstruction

Background

Computational approaches to modeling large complex systems standardize the representation of knowledge, while simulation of computational models illuminates the dynamics of systems, allowing for discoveries and theoretical advances (1). Modeling of physical and engineered systems has led to insights ranging from explanations of fundamental principles to applications such as robust flight simulators.

However, biological systems have proven difficult to model, in part due to their complexity and robustness to failure and noise. There are two main approaches to modeling these systems, bottom-up and top-down (2). In a bottom-up approach, known molecular interactions are assembled into a model to help explain the system's behavior and predict how the system will respond to new stimuli or inputs. This method has been used extensively by biologists, biochemists, and molecular biologists to manually create models based on the interactions that are supported in scientific literature. In a top-down approach, experimental data—usually collected with high-throughput methods of data collection—is used to infer relationships between elements and determine causal relationships. Top-down approaches employ many different methods such as Bayesian Inference (3), ANOVA calculations (4), and Fuzzy Logic (5) among others. In both the mechanistic bottom-up approach and the data-driven top-down

approach, the model is used to predict the behavior of individual elements in the network (6, 7). Recently, there has been a push to integrate the two methods, using experimental data to contextualize the bottom-up approach, and incorporating prior knowledge into the top-down approach to reduce the number of potential models (8-12). Despite these efforts, modeling biological systems remains a tedious, iterative, and ongoing pursuit. Thus, useful models are rarely built for large systems and even more rarely updated to incorporate new information. This laborious process results in slow model development and few model extensions. It is because of these factors, that system and computational biologists have endeavored to automate the process of model creation and extension.

In the advent of Big Data, more information is available than ever before about molecular interactions. To automatically create models, information can be extracted from literature, queried from databases, or taken from existing baseline models. Public databases such as Reactome (13), MetaCyc (14), OmniPath (15), and STRING (16) offer easy access to millions of interactions. Additionally, there exist a number of model databases with published models that are publicly available such as The Nature Pathway Interaction Database (17), WikiPathways (18), BioModels (19), the Cell Collective (20), and KEGG pathways (21). These databases contain highly targeted, curated published and unpublished models which are created for specific biological context and may not be generalizable to explain other phenomena. When new interactions are discovered, and described in a scientific publication, state-of-the-art machine reading engines such as REACH (22), TRIPS (23), EVEX (24), and Rubicon (25) can extract them, together with other relevant information. These automated readers are able to extract tens of thousands of biological entity interactions from hundreds of papers in a few hours, and

produce a machine-readable, structured output (22). Many of the interactions which reading engines extract are known or incorrect, with only a few exceptions being true, novel interaction which are not yet in curated databases. However, despite this abundance of available interactions, there is still no efficient way to assemble them into reliable models.

Existing methods for collecting biological interactions, assembling a model, and performing simulations include Path2Models (26) and INDRA (27, 28), which automatically generate both quantitative and qualitative models from available pathway information. Both methods can assemble models based on the quality and granularity of the information they are given; however, truly representative models are dependent upon the modeling approach and context, which in turn depends on human oversight, intervention and correctness of starting information. These techniques rely on accurate information, and their performance suffers when the interaction information is inaccurate, incomplete, or from a different biological context. Recently, several other methods have been proposed that combine simulations with formal model analysis (29) to automatically expand, test and select the best model, with respect to a given performance metric. These new tools include the Markov clustering approach (30) and genetic algorithm-based model extension (31). These approaches have different strengths and weaknesses. The Markov clustering approach to model extension is well suited for the combinatorial explosion in numbers of possible extensions while the genetic algorithm approach is overwhelmed by large numbers of extensions. Markov clustering prioritizes strongly connected components at the expense of interactions involving nodes of low degree. The genetic algorithm explores the effect of single extensions distributed throughout the network.

Here, we propose FIDDLE (Finding Interactions using Diagram Driven modeL Extension), a tool that conducts model assembly in a bottom-up approach. The goal of FIDDLE is to assemble a model that satisfies a given performance metric (defined formally in Methods), starting from a list of known biological interactions, and key indicators of the studied biological phenomena. We use the key indicators as ground truths when computing the error in our model performance metric. FIDDLE is able to refine models by systematically adding known biological interactions into intermediate models, measuring changes in model performance, and then adding or discarding interactions based on whether they improve the model performance metric. FIDDLE incorporates two novel algorithms, Breadth First Addition (BFA) and Depth First Addition (DFA), which utilize the same principles as the breadth-first search and depth-first search algorithms in network studies (32). These methods not only represent a new approach to bottom-up model assembly but are also used to demonstrate the existence of key biological properties which hinder automated modeling of biological systems. The model structure can be illustrated using networks, where model elements (such as proteins, genes, biological processes, etc.) are represented by nodes, and the molecular interactions (such as activation, inhibition, etc.) are represented by directed edges. Including the direction in element connections allows for the representation of causal relationships between nodes, and consequently, enables studies of the complex system dynamics that arises from these causal relationships. We show a high-level overview of FIDDLE in Figure 1.

Figure 1: "Overview of the FIDDLE process" FIDDLE is a robust model assembly tool which requires candidate edges, an expectation of model behavior, and an optional baseline

model. FIDDLE iteratively assembles candidate edges into candidate models, simulates them, and calculates their performance using the TME metric.

To evaluate FIDDLE and the proposed BFA and DFA algorithms, we created a number of artificial networks, systematically removed network edges, and tasked FIDDLE to re-build the original model. We used artificial networks so that the structure and dynamics of the original network were completely known, and we could vary key parameters in their creation to observe the impact it would have on reassembly. We created two different types of networks: Erdös-Rényi random networks (ER) (33) and Barabási-Albert scale-free networks (BA) (34). Biological models, power grids and the internet follow scale-free power distributions. We will evaluate the two network types to examine the role of network structure and determine the algorithmic performance by comparing the behavior of the original and the reconstructed model.

We also demonstrate the advantages to using our BFA and DFA algorithms for automated model assembly. Throughout our analysis, we contrast the differences between them, illustrating their relative strengths, weaknesses and commonalities. By using different network structures and different types of candidate knowledge, we are able to gain insight into the complex nature of biological models, as well as point to the future goals and difficulties with completely automated model assembly.

Results

Dataset of True and Candidate Knowledge

We selected a network size of approximately 50 nodes to represent a typical biological model requiring reasonable computational time. We assume that complete models, before any edges are removed represent *golden models,* ideal representations of the actual system. When creating the golden models, the components, structure, and performance are assumed to be known perfectly (which is most often not the case when using existing knowledge about the biological systems). Once removed from the golden model, we save the edges as *candidate knowledge* for model assembly. Finally, to account for variable candidate knowledge, we add a list of false edges to the list of true edges from the golden model and use this new set as an input to FIDDLE. We created two different types of random networks, and the descriptive statistics of these networks are shown in Table 1.

Table 1: Summary statistics of the *golden models* created as an ER random network or a BA scale-free network structure.

Information	ER	BA
Number of Nodes	48.4 ± 1.4	50.0 ± 0.0
Number of Edges	85.5 ± 8.9	96.0 ± 0.0
Model Density	0.037 ± 0.003	0.039 ± 0.000
Model Average Degree	3.53 ± 0.31	3.84 ± 0.00
Undirected Model Clustering	0.06 ± 0.03	0.20 ± 0.06
Undirected Model Diameter	6.9 ± 0.7	5.0 ± 0.4

A summary of the ER random networks and BA networks generated as golden models. Fifty networks were made of each type, and the averages and standard deviations for all network characteristics were determined.

We removed edges from the random networks assuming different probabilities of removal, 0.1, 0.25, 0.5, 0.75, or 1, and stored the removed edges so they could be used by FIDDLE in the model assembly process. Descriptive statistics describing the removal of edges from the 50 ER networks and 50 BA networks are shown in Table 2, where Model Density is defined as the ratio of directed edges to nodes, and Model Degree is the sum of all individual node degrees, divided by the total number of nodes. An example of edge removal is shown in Figure 1.

Table 2: Summary Statistics of Edge Removal

Random Edge Removal for ER Networks				
Removal Probability	Model Density	Model Average Degree	Number of Removed Model Edges	Number of Model Edges
0	0.037 ± 0.002	3.52 ± 0.30	0.0 ± 0.0	85.5 ± 8.8
10	0.032 ± 0.003	3.10 ± 0.33	10.1 ± 5.1	75.3 ± 9.0
25	0.027 ± 0.003	2.59 ± 0.29	22.7 ± 6.9	62.8 ± 7.7
50	0.018 ± 0.003	1.70 ± 0.36	44.0 ± 7.6	41.4 ± 9.2
75	0.009 ± 0.002	0.90 ± 0.23	63.4 ± 6.9	22.0 ± 5.9
100	0.000 ± 0.000	0.00 ± 0.00	85.5 ± 8.8	0.0 ± 0.0
Random Edge Removal of BA Networks				
Removal Probability	Model Density	Model Average Degree	Number of Removed Model Edges	Number of Model Edges
0	0.039 ± 0.000	3.84 ± 0.00	0.0 ± 0.0	96.0 ± 0.0
10	0.035 ± 0.002	3.46 ± 0.18	9.4 ± 4.5	86.6 ± 4.5
25	0.029 ± 0.003	2.87 ± 0.28	24.2 ± 6.9	71.8 ± 6.9
50	0.020 ± 0.003	1.97 ± 0.28	46.6 ± 6.9	49.4 ± 6.9
75	0.010 ± 0.002	0.97 ± 0.21	71.6 ± 5.2	24.4 ± 5.2
100	0.000 ± 0.000	0.00 ± 0.00	96.0 ± 0.0	0.0 ± 0.0

A summary of the features of the ER random networks and BA networks that were generated as golden models, as well as a summary of the random removal of edges. For each network, edges were removed with different probabilities to determine the smallest network required for reconstruction of the original model.

Figure 2: "Illustration of edge removal with varying probabilities" Example of the process of randomly removing model edges with different probabilities. If an edge is removed with 0.1 probability, that edge will also be removed from the network for all higher probabilities of edge removal.

Applying a particular edge removal probability on a set of networks results in a varying percentage of missing edges among these networks. With each edge having a 0.1 probability of being removed, there are some networks missing 5% of their edges, while some are missing up to 15%. As shown in Figure 2, the percentages of removed edges are distributed over most of the range between 0 and 100% (no missing edges to all missing edges). We show in Figure 3 the edge removal probability plotted against Model Density and the Total Model Error. In Methods, we define Total Model Error (TME) as a measure of the difference between the golden and the candidate model performance.

Figure 3: "Illustration of the effect of varying edge removal probability on network properties" Analysis of the effect of different edge Removal Probabilities on Model Density (left) and TME (right).

Analysis of FIDDLE Model Extension

Next, we attempted to reconstruct the models after different percentages of edges had been removed from them. We will refer to edges that were originally in the golden model, before edge removal, as "true" edges, and to edges that were not in the golden model as "false" edges. To determine the impact of candidate knowledge on the model extension procedure, the extension

method takes as input two lists of candidate edges: an *Exclusive list* (EXC) which contains only the "true" edges that need to be in the golden model and are necessary for exact model reconstruction, and an *Inclusive list* (INC) which contains all the edges that are in the EXC, as well as an equal number of "false" edges (randomly generated edges between model nodes that were not in the golden model). The creation of the two lists is meant to reflect the two types of information available to modelers: interactions which are plausible and based on strong literature evidence, usually well known to the scientific community and supported by curated databases, and the information with less evidence, such as the candidate edges that we get from the automated natural language processors. We demonstrate these concepts in Figure 4.

Figure 4: "Example of model assembly using the INC List" In this example, we extend a Baseline Model with the INC list of candidate edges and generate the Assembled Model. Based on whether candidate edges are part of the Golden Model, and included in the Assembled Model, we can determine whether the edges represent TP, FP, TN, or FN. We calculate Recall, TNR, FDR, and FOR from the number of TP, FP, TN, and FN in the Assembled Model.

To measure each method's recall — the fraction of "true" edges that are selected to extend the model with, given a mix of "true" and "false" edges — we first evaluated both methods using the EXC list to extend baseline models, and then we conducted tests with the INC list. The results of each method are shown in Figure 5A.

Figure 5: "Comparison of performance statistics of model extension" (A) Box plots comparing the quartile ranges of the recall of both algorithms under different conditions. The top row is for models based on the ER network while the bottom row is based on BA networks. The left column is comparing the two algorithms when using only the EXC list, while the right column is comparing the algorithms when using the INC list, with equal number of true and false edges. Statistical significance was determined by t-test (* $p < 0.05$, *** $p < 0.001$). (B) Box plots comparing the quartile ranges of the TNR , FDR, and FOR of both algorithms under different conditions. The top row is for models based on the ER while the bottom row is based on BA. In all cases, both algorithms were given the INC list, with equal number of true and false edges. Statistical significance was determined by t-test (* $p < 0.05$, *** $p < 0.001$).

To compare the ability of BFA and DFA to handle true edges and false edges, and to differentiate between them, both methods were given the INC list, and tasked to reconstruct both types of networks, ER and BA networks. We decided to use True Negative Rate (TNR), False Discovery Rate (FDR), and the False Omission Rate (FOR) to quantify the ability of BFA and DFA to classify edges. TNR is also known as Specificity or Selectivity, and it measures the ability to avoid incorporating false edges (the number of false edges not added to the model divided by the total number of false edges). FDR is also known as (1 – Precision) or the Positive Predictive Value (PPV), and is the rate of selecting false edges from all edges (the number of false edges added to the model divided by the total number of edges added to the model). FOR is also known as (1 – Negative Predictive Value (NPV)), and is the rate of true edges not being added to the model (the number of true edges not added to the model divided by the total number

of edges not added to the model). Using these classification statistics, we would want a TNR close to one, and an FDR and FOR close to zero. The calculated rates of TNR, FDR, and FOR, are shown in Figure 5B.

In addition to model reconstruction, we also evaluated our methods on model creation. BFA and DFA algorithms were tasked with minimizing the TME using only the EXC and INC list of interactions and no baseline model. The recall of both BFA and DFA are shown in Figure 6A. To compare the ability of BFA and DFA to detect false edges, we compared networks created by the two algorithms, with respect to TNR, FDR, and FOR, as shown in Figure 6B.

Figure 6: "Comparison of performance statistics of model creation" (A) Box plots comparing the quartile ranges of the recall of both algorithms under different conditions. The top row is for models based on the ER random network while the bottom row is based on BA networks. The left column shows the results obtained when using the EXC list in both algorithms, while the right column shows the results when the algorithms were given the INC list, with equal number of true and false edges. Statistical significance was determined by t-test (* $p < 0.05$, *** $p < 0.001$). (B) Box plots comparing the quartile ranges of the TNR, FDR, and FOR of both algorithms under different conditions. The top row is for models based on the ER random network while the bottom row is based on BA networks. In all cases, both algorithms were given the INC list, with equal number of true and false edges. Statistical significance was determined by t-test (* $p < 0.05$, *** $p < 0.001$).

Throughout the process of model creation and extension, we were interested in comparing the methods with respect to the gradient of change in the TME and the number of extensions used to rebuild the model. To examine the difference in performance of the BFA and DFA algorithms on the ER and BA networks, we plotted the TME against the number of edges added back to the model. This plot is shown in top row of Figure 7. To account for networks with larger number of edges added back to the model and ones that started out with higher levels of TME, we normalized the plots to the highest level of TME. This normalized version is shown in the bottom row of Figure 7.

Figure 7: "Model performance as edges are added back into the model" In the top row of this figure, we see plots of the model TME (y-axis) with number of edges added (x-axis). In the bottom row, we see the plots normalized to TME.

Next, we were also interested in visualizing the assembled network that began as an ER or BA network, and that was extended with the BFA or DFA method with a goal of minimizing the TME. These reconstructed networks are shown in Figure 8.

Figure 8: "Examples of model assembly in random and scale-free networks" Examples of BFA- and DFA-based model assembly for: (A) ER network models, and (B) BA network models.

Comparison of TME by Random Network Type

Lastly, we were interested in explaining the difference in performance of the two network types with respect to the TME. In Figure 9, we began with either an ER or BA network, and removed every two-edge combination from the golden model. In this way, we created a set of models similar to the golden model except for a different and unique pair of edges that we removed. We then created a heatmap of the TME for each reduced model, based on which two edges were removed. In cases of the diagonal, only one edge was removed and therefore serves as the golden model with only one edge missing. These heatmaps comparing the TME of ER and BA networks are shown in Figure 9.

Figure 9: "Heatmap of network response to combinatorial edge removal"
These heatmaps compare the differing responses of ER and BA networks under random edge removals. Columns represent the TME for a model with the same edge missing, while rows also represent the TME for a model with a different edge missing. The diagonal represents models with only one edge missing.

Discussion

Automated model assembly and execution is the goal for systems and computational biologists. In this paper, two novel algorithms for bottom-up model creation were evaluated, and emphasized how complicated the models of biological systems are to create. We used artificial networks with different network structure in order to analyze the BFA and DFA algorithm strengths and weaknesses. The following sections discuss the assembly algorithms, their performance on the artificial models, and future work.

Assembling Models

To investigate the ability of our proposed algorithms to assemble a model with the desired behavior, we selected easily tunable random ER, and scale-free BA networks. Many biological systems have been observed to follow a scale-free power distribution (34). We systematically and randomly removed edges from these networks, which allowed for a varying number of edges to be removed from each network. The density of the network was plotted against edge removal probability, in order to understand how dense these networks were, and how edge removal affected density.

In Figure 3, we can see that both ER and BA models responded linearly to the removal probability. The linear line observed in BA networks is a result of the method of network creation. Since BA networks were created with the same number of edges (albeit in different configurations, direction, and regulation), Model Density was then a linear function of removal probability. In ER networks, since the number of edges to begin with were different between different networks, the obtained network population is more diverse, although the networks still respond linearly to the edge removal probability. Interestingly, this linear relationship is not conserved when observing model performance (TME) as a function of edge removal probability. Rather than a strict linear relationship (in which each edge is equally responsible for the same dip in model performance), the effect of different removal probabilities seems to have no linear relationship with the TME metric. It appears that a small number of missing edges can have a strong impact on model performance in some cases, while a large number of missing edges may register a small impact. As we will see later, it is likely that there exist multiple local minima in the model TME state space but which fail to encapsulate the total model information. In this

way, we are able to minimize the model TME and yet the assembled model will have a different structure than the golden model.

In a paper discussing networks under random or focused attack, Albert et al. (35) used network diameter to illustrate network failure. They were able to show that with the removal of a small number of edges, the diameter of ER networks reacted differently than that of BA networks. Similarly, our study shows that ER networks respond differently to edge removal than BA. Rather than network diameter, the shortest path between any two nodes, we use the TME metric since our models are small by comparison and our networks are directed. In models that aim to capture biological phenomena, nodes with a high degree of connection dwarf the importance, significance, and effect of other nodes (36, 37).

Comparison of the BFA and DFA algorithms

Given that we are using random networks as golden models, the random edge removal makes direct comparison difficult. In order to more reliably compare the two algorithms, we applied them to the exact same baseline model, and we used the same set of candidate edges. We then examined the recall, TNR, FDR, and FOR of the two methods.

In Figures 4 through 7, we see that the BFA algorithm performs much better in TNR, while the DFA algorithm performs much better in recall. The higher TNR of the BFA is due to the scanning of all possible edge additions to select an edge that leads to largest TME reduction, and therefore, more selective inclusion of edges and an increased probability of including true edges (if not redundant or initially detrimental). DFA exhibits higher recall and includes any edge that

improves model performance. As true edges are more likely to improve model performance, DFA is able to add more true edges regardless of the magnitude of improvement to model performance.

The BFS algorithm appears to be a better choice for model assembly when the candidate knowledge is a mix of true and false edges, as it will only incorporate edges vital to the model's performance. On the other hand, the DFA method would be a better choice when correct information is provided, as it performs better at reconstructing the golden model with more true edges. Employing the algorithms in a parallel manner would recommend the best edges that are likely to be in the model and suggest additional edges to explore experimentally. Additionally, the BFA method is easily parallelized for rapid execution on large models.

To assess the algorithms, we plotted the TME metric as a function of edges as they were added to the model (Figure 8). The steep decrease early on of the BFA method shows the quick incorporation of highly important edges that drastically reduce TME. DFA does not exhibit the same performance as any improvement is added immediately. Changes in TME are similar for ER and BA networks, although BA networks appear to begin with less TME and require a fewer number of edges to improve model performance when compared to ER networks. This holds across different percentages of missing edges except when all edges are removed. Once all edges are removed, BA networks have a worse change in TME with the number of edges added; this is likely due to BA networks possessing more edges on average (an artifact of the network creation method). In each case, there appear to be a small number of edges which can drastically reduce TME, regardless of the overall percent of edges being removed from the model. This is best seen

in instances of 0.5 and 0.75 removal probability. This trait disappears when there is just too much information (edges) missing from the model for complex regulations and causal relationships to form. In comparing the performance of BFA and DFA when given either EXC or INC list, it is interesting to note that the INC list performs almost identically (or only slightly better) to the EXC list in BFA. This demonstrates that only True edges are likely to improve the model performance and trigger a steep decrease in TME.

Automatically Created and Extended Models

Next, we were interested in exploring the effect of network structure on each algorithm's performance. As we attempted to reconstruct the golden models with FIDDLE, the BFA and DFA algorithms were successfully able to minimize TME; however, the assembled models were not the same as the golden models (Figure 8). The key reason is that these algorithms function to maximize the performance metric, i.e., minimize TME, while adding the least number of edges. Thus, this approach often results in finding local minima and simplifying the model to its most basic causal interactions that reproduce the expected behavior. We demonstrate some of the limitations of applying a simple error function in Figure 9, where two missing true edges can result in both much better and much worse model performance than either edge individually. There have been similar analyses on the complicating factors of inferring biological networks (38). In a comprehensive analysis of the DREAM3 challenge (38, 39), four main errors were found that would drastically reduce the reconstruction method's confidence and directly impact the algorithm's performance: fan-out error, fan-in error, cascade error, and feed-forward loop error. These errors are inherent symptoms of characteristics of the underlying model structure: highly redundant network connectivity and complex network motifs.

In a review by Kitano [2], he argues that robustness is a fundamental feature of evolvable complex systems, necessary for them to be robust to environmental and genetic mutations. He describes four key mechanisms that help ensure robustness: system control, alternative mechanisms, modularity, and decoupling. Both the existence of alternative mechanism and complex system controls make it difficult to understand and model these systems. The presence of alternative mechanisms allows for high levels of redundancy in biological networks while system control requires a complex interplay of multiple interactions where missing a small subset will fundamentally change the dynamics of the model. Luckily, biological modularity may hold the key into how we can best understand and model these systems. In this paper, we focused on singular molecular interactions and adding one at a time. Perhaps through analyzing potential interactions in the context of the system and understanding the modules as a whole, we may make model assembly and extension easier as these modules can be added together.

Implications for Creation and Extension Methods

This paper describes an approach for the automatic assembly of biological models. The efficient incorporation of novel information, especially from machine reading systems, is critical to maintain updated models of biological systems. The analysis conducted in this work also enables comparisons with other algorithms such as genetic (31) and clustering algorithms (30) and future efforts to combine these algorithms into a hybrid approach for model extension. There has been considerable work (30) showing different clustering methods of model elements and these will be explored for model assembly. We plan on continuing this study by adding groups of interactions together, the best number to add together and interrogate the traits that make edges "good" to be added together. This is a complex problem as it not only depends on what type the

edges are and where they are added but how they are integrated into the network. Also, while the FIDDLE algorithms do not depend on the type of update functions used to simulate the model (arithmetic, Boolean, etc.), the type of update function is expected to impact model extension accuracy.

Further improvements can be realized using a rigorous method for testing models, such as statistical model checking, which allows for efficient verification of system properties (40). Therefore, model checking can be used to select models that best recapitulate real biological systems by estimating how well models satisfy observed or desired system properties.

Conclusions

We have been able to illustrate the advantages and limitations of utilizing breadth or depth first addition algorithms for assembling biological networks from new candidate knowledge extracted from literature. In general, automated assembly of biological models is challenging due to the inherent redundancy and complexity in biological networks. In this paper, we have shown that BFA has higher TNR and DFA is more sensitive when it comes to assembling biological models. These relative strengths suggest different applications: BFA could be used to extend a model in a more focused way, more likely to include only true biological connections while DFA could be used to suggest many different interactions that could be explored for applicability. In general, it appears that BFA is a better method to use if there is an abundance of inaccurate information, while DFA is a better method for accurate information and assembly of a truly representative model. In the future, we plan on improving our algorithms to further advance the effectiveness of automated model assembly from interactions extracted from literature and public databases.

Methods

The FIDDLE tool can be used as a stand-alone tool, or as a part of a larger framework of tools. In this work, we use the DySE framework (41), to simulate, and analyze the models assembled by FIDDLE. Both FIDDLE and DySE are implemented in Python. In addition, the Python package, NetworkX (42) was used to create all networks, while Graphviz (43) was used to visualize them.

Discrete Models and Simulation

The models that we use in this work are composed of elements representing proteins, genes, chemicals, or biological processes, and directed element interactions. For each model element, we define its *influence set*, consisting of its positive (activating) and negative (inhibiting) regulators. The model structure is a network G(V,E), where V is a set of nodes representing model elements, and E is a set of directed edges representing regulatory influences between elements. Here, we assume that each model element can be in one of the three states, OFF (value 0), LOW activity (value 1), and HIGH activity (value 2). In general, FIDDLE is not limited to elements with only three states and is applicable to any discrete model. Each model element can have an *update rule*, which is used to determine element's state given the states of its regulators in the influence set. Update rules that we used in this work are summation functions, where the sum of the state values of negative regulators is subtracted from the sum of the state values of positive regulators, although the FIDDLE algorithms are not dependent on the type of update functions. Together, model elements, element influence sets, the number of values for each element, and element update rules, F, and together comprise an *executable model*, M(V,E,F).

In this work, we use Discrete, Stochastic, Heterogeneous simulator (DiSH) (44), which allows for simulations of discrete models with various types of update functions, and has several different simulation schemes, including deterministic and stochastic. For the analysis we conducted here, we used a uniform, step-based, random-ordered, sequential update scheme (USB-RSQ), described in detail in (44). It has been shown previously (44, 45) that the USB-RSQ simulation scheme is able to recapitulate the intra-cellular network dynamics. DiSH simulates the models starting from an initial state (assigned before simulations), and for a pre-defined number of time steps, T (e.g., when the steady state is reached). Each such simulation run, r, yields for every model element $v_i \in V$, a trajectory of values, $S^r_{v_i} = \left(s^r_{v_i,1}, s^r_{v_i,2}, \ldots s^r_{v_i,T}\right)$, where $s^r_{v_i,t}$ is the state value of element v_i at time point t (t=1,..,T), within run r. Due to the randomness of the update scheme, element trajectories may vary across multiple runs that start with the same initial state. Therefore, for the same time step t, following the approach from (44), we average the values $s^r_{v_i,t}$ across different runs, to obtain average trajectories for all elements. More formally, we compute an average trajectory of model element v_i as:

$$\bar{S}_{v_i} = \frac{1}{R} \cdot \sum_{r=1}^{R} S^r_{v_i} = \frac{1}{R} \cdot \sum_{r=1}^{R} \left(s^r_{v_i,1}, s^r_{v_i,2}, \ldots s^r_{v_i,T}\right) = \left(\bar{s}_{v_i,1}, \bar{s}_{v_i,2}, \ldots \bar{s}_{v_i,T}\right) \quad \text{Eq 1}$$

where R is the overall number of conducted simulation runs. Additionally, we denote the average model state at the final time step T as $Q = \left(\bar{s}_{v_1,T}, \bar{s}_{v_2,T}, \ldots, \bar{s}_{v_N,T}\right)$, where N is the number of elements in the model.

In the experiments that we describe here, for all model elements $v_i \in V$, we assume that their initial state value is $s^r_{v_i,1} = 1$, and for each created model, we conducted R=100 simulation runs, each with T = 2,500. Given that the networks we use here are artificial (they do not represent any real biological system), there is no information about initial values that should be used for

elements, and therefore, we used the middle value among the three possible state values, to minimize the bias that the lowest and highest state values can have on model dynamics.

Model Performance Metric

Model accuracy is usually measured by evaluating the model against a set of ground truths, or expected behaviors, that specify how the model is supposed to respond given a certain initial state and input values. Since our goal is to automatically assemble models, we incorporated within our assembly algorithms a metric that assesses how closely to the specified behavior newly created models perform. Our model performance metric examines the states of all model elements in the last time step T of the average trajectories (usually, a steady state). We denote the expected final state vector of the desired behavior, either observed in a real system or assumed to hold for a *golden model*, as Q_{GM}. Any model that we assemble is considered a *candidate model*, and we denote its final state as Q_{CM}. We define Total Model Error (TME) as a measure of the difference between the golden and the candidate model behavior, that is, we compute the absolute distance between vectors Q_{GM} and Q_{CM}:

$$\mathrm{TME}(Q_{\mathrm{GM}}, Q_{\mathrm{CM}}) = \sum_{i=1}^{N} \left| \bar{s}_{\mathrm{v}_i,T}^{\mathrm{GM}} - \bar{s}_{\mathrm{v}_i,T}^{\mathrm{CM}} \right| \qquad \text{Eq 2}$$

where N represents the number of elements in the model.

Model Assembly Methodology

The inputs to FIDDLE include: a *baseline model*, $\mathrm{M_{BL}}(\mathrm{V_{BL}},\mathrm{E_{BL}},\mathrm{F_{BL}})$, to be extended by FIDDLE into candidate models, a set of *candidate edges*, $\mathrm{E_{NEW}}$, that are used to extend the baseline model, the expected behavior in the form of the final state values of the golden model, Q_{GM}, and the precomputed TME of the baseline model, $\mathrm{TME_{BL}}$.

The **Breadth First Addition** (BFA) procedure starts by evaluating the contribution of each new edge from E_{NEW} to decreasing TME_{BL}, that is, we simulate the model that consists of our original baseline model M_{BL} and a selected new edge, and we compute TME of that extended model according to Eq 2. Next, we permanently move, from E_{NEW} to the model, the new edge that leads to the largest decrease in the original TME_{BL}, and then we repeat the procedure with this new extended model, i.e., similar to what we did with the original model, we evaluate addition of the remaining edges from E_{NEW} to this new model by computing their TME values. This process is repeated until we either find the extended model that matches the expected end values of the golden model (Q_{GM}), there are no more edges in E_{NEW}, or no edge can be added to the baseline without increasing the TME. Our BFA algorithm is outlined more formally in Figure 10.

Figure 10: "Pseudocode of the BFA Algorithm"

The basic structure of the BFA algorithm in a pseudocode format.

The **Depth First Addition** (DFA) procedure, similar to the BFA algorithm, starts with evaluation of edges from E_{NEW} by computing their contribution to decreasing TME of the baseline model. Different from BFA, as soon as we find an edge which leads to a TME lower than the TME_{BL}, we add that edge to the baseline model, and remove it from E_{NEW}. We repeat this procedure using the new extended model and the remaining edges from E_{NEW}, that is, as soon as we encounter an edge that leads to a lower TME than the current model, we add that edge to the model. The DFA algorithm stops when we either find the extended model that matches the expected end values of the golden model (Q_{GM}), there are no more edges in E_{NEW}, or

no edge can be added to the baseline without increasing the TME. Our DFA algorithm is outlined more formally in Figure 11.

Figure 11: "Pseudocode of the DFA Algorithm"

The basic structure of the DFA algorithm in a pseudocode format.

Evaluation Procedure

To evaluate how well FIDDLE can assemble models and to compare and contrast the BFA and the DFA approach, we apply a method commonly used for model reconstruction (46-50). The first step of this method is a generation of golden models. We used two different types of random networks to create our golden models: one type based on Erdös-Rényi random directed networks (33), and another using the Barabási-Albert scale-free network (34). To generate an Erdös-Rényi Network (ER), a random network was seeded with 50 nodes and a probability of connection of 0.035. To create a Barabási-Albert scale-free network (BA), each network was seeded with fifty nodes and two initial edges. For both the ER and BA networks, these parameters were chosen to create artificial networks of roughly the same number of nodes, edges, and density (as shown in Table 1). Models were kept around 50 nodes to best replicate the synthetic models used in the DREAM3 challenge (39). We used the Python library NetworkX (42) to generate these networks and to calculate the network statistics reported in Table 1. Next, we assigned each directed edge in these networks, randomly with equal likelihood, either a positive or negative sign (activating or inhibiting, respectively). Finally, we simulated golden models to record the Q_{GM} values, which are used by the TME function to guide the BFA and DFA algorithms.

In the second step of the evaluation procedure, we created baseline models by randomly removing edges from the golden models. Specifically, we select the removal probability to be 0.1, 0.25, 0.5, 0.75, or 1, and for each edge in the model we randomly choose whether to remove it or not with that selected probability. For example, in a model with 0.5 edge removal probability, an edge was selected, given a 50-50 chance of being removed, either removed or not, and then the next edge was selected. If an edge was removed in a model of low removal probability, it was also missing from a model of higher removal probability. The removed edges were stored in a list called the *Exclusive list* (EXC) since it only contains edges that were removed from the golden model. We also create an *Inclusive list* (INC), which contains every edge in EXC, and an equal number of false edges that are not present in the golden model. The INC and EXC lists are used in the evaluation procedure, where they serve as the candidate knowledge (edges) input to FIDDLE.

We conducted exhaustive analysis of all possible combinations of baseline model (with 0.1, 0.25, 0.5, 0.75, or 1 probability of removing edges from the golden model), extension algorithm (BFA or DFA), and candidate edges (EXC or INC list). In each experiment, we used the final state of the golden model, Q_{GM}, when computing the model performance metric TME of assembled candidate models, in an effort to reconstruct the golden model, or assemble a model that reproduces the same final state as the golden model.

List of Abbreviations

BFA: Breadth First Addition

DFA: Depth First Addition

ER: Erdös-Rényi Random Network

EXC: Exclusive List

FOR: False Omission Rate

FDR: False Discovery Rate

INC: Inclusive List

BA: Barabási-Albert Scale-Free Network

Declarations

Acknowledgments

The authors would like to thank Kai-wen Liang for his instrumental work in the implementation of the BFA algorithm.

Authors' contributions

AB conceived the project. AB created the BFA and DFA algorithms. AB generated all of the networks, the simulation of models, and the analysis of the results. AB, NM, and CT interpreted the results and wrote the manuscript. All authors read and approved the final manuscript.

Availability of data and materials

- Project name: FIDDLE
- Project home page: https://bitbucket.org/biodesignlab/fiddle/
- Archived version: 2c6ad05
- Operating system(s): Linux/Mac
- Programming language: Python3
- Other requirements: openly available pypi packages, detailed in requirements.txt
- License: BSD-2-Clause License
- Any restrictions to use by non-academics: None

Consent for publication

Not applicable.

Competing Interests

The authors declare that they have no competing interests.

Ethics approval and consent to participate

Not applicable.

Funding

This work was funded in part by DARPA award W911NF-17-1-0135.

References

1. Epstein JM. Why Model? 2008.
2. Sobie EA, Lee Y-S, Jenkins SL, Iyengar R. Systems biology—biomedical modeling. Sci Signal. 2011;4(190):tr2-tr.
3. Schäfer J, Strimmer K. An empirical Bayes approach to inferring large-scale gene association networks. Bioinformatics. 2004;21(6):754-64.
4. Küffner R, Petri T, Tavakkolkhah P, Windhager L, Zimmer R. Inferring gene regulatory networks by ANOVA. Bioinformatics. 2012;28(10):1376-82.
5. Raza K. Fuzzy logic based approaches for gene regulatory network inference. Artificial intelligence in medicine. 2018.
6. Madhamshettiwar PB, Maetschke SR, Davis MJ, Reverter A, Ragan MA. Gene regulatory network inference: evaluation and application to ovarian cancer allows the prioritization of drug targets. Genome medicine. 2012;4(5):41.
7. D'haeseleer P, Liang S, Somogyi R. Genetic network inference: from co-expression clustering to reverse engineering. Bioinformatics. 2000;16(8):707-26.
8. Linde J, Schulze S, Henkel SG, Guthke R. Data-and knowledge-based modeling of gene regulatory networks: an update. EXCLI journal. 2015;14:346.
9. Wani N, Raza K. Integrative Approaches to Reconstruct Regulatory Networks From Multi-Omics Data: A Review of State-of-the-Art Methods. 2018.
10. Hecker M, Lambeck S, Toepfer S, Van Someren E, Guthke R. Gene regulatory network inference: data integration in dynamic models—a review. Biosystems. 2009;96(1):86-103.
11. Banf M, Rhee SY. Enhancing gene regulatory network inference through data integration with markov random fields. Scientific reports. 2017;7:41174.
12. Recamonde-Mendoza M, Werhli AV, Biolo A. Systems biology approach identifies key regulators and the interplay between miRNAs and transcription factors for pathological cardiac hypertrophy. Gene. 2019.
13. Fabregat A, Jupe S, Matthews L, Sidiropoulos K, Gillespie M, Garapati P, et al. The Reactome Pathway Knowledgebase. Nucleic Acids Research. 2018;46(D1):D649-D55.
14. Karp PD, Riley M, Paley SM, Pellegrini-Toole A. The MetaCyc Database. Nucleic acids research. 2002;30(1):59-61.
15. Türei D, Korcsmáros T, Saez-Rodriguez J. OmniPath: guidelines and gateway for literature-curated signaling pathway resources. Nature Methods. 2016;13(12):966-7.
16. Szklarczyk D, Morris JH, Cook H, Kuhn M, Wyder S, Simonovic M, et al. The STRING database in 2017: quality-controlled protein-protein association networks, made broadly accessible. Nucleic acids research. 2017;45(D1):D362-D8.

17. Schaefer CF, Anthony K, Krupa S, Buchoff J, Day M, Hannay T, et al. PID: the pathway interaction database. Nucleic Acid Res. 2009;37.
18. Slenter DN, Kutmon M, Hanspers K, Riutta A, Windsor J, Nunes N, et al. WikiPathways: a multifaceted pathway database bridging metabolomics to other omics research. Nucleic acids research. 2018;46(D1):D661-D7.
19. Chelliah V, Juty N, Ajmera I, Ali R, Dumousseau M, Glont M, et al. BioModels: ten-year anniversary. Nucleic Acids Research. 2015;43(D1):D542-D8.
20. Helikar T, Kowal B, McClenathan S, Bruckner M, Rowley T, Madrahimov A, et al. The Cell Collective: toward an open and collaborative approach to systems biology. BMC Syst Biol. 2012;6.
21. Kanehisa M, Furumichi M, Tanabe M, Sato Y, Morishima K. KEGG: new perspectives on genomes, pathways, diseases and drugs. Nucleic Acids Research. 2017;45(D1):D353-D61.
22. Valenzuela-Escárcega MA, Hahn-Powell G, Surdeanu M. Description of the Odin Event Extraction Framework and Rule Language. 2015.
23. Ferguson G, Allen JF, editors. TRIPS: An integrated intelligent problem-solving assistant1998: AAAI Press.
24. Hakala K, Van Landeghem S, Salakoski T, Van de Peer Y, Ginter F. Application of the EVEX resource to event extraction and network construction: Shared Task entry and result analysis. BMC Bioinformatics. 2015;16(Suppl 16):S3-S.
25. Burns GAPC, Dasigi P, de Waard A, Hovy EH. Automated detection of discourse segment and experimental types from the text of cancer pathway results sections. Database : the journal of biological databases and curation. 2016;2016.
26. Buchel F, Rodriguez N, Swainston N, Wrzodek C, Czauderna T, Keller R, et al. Path2Models: large-scale generation of computational models from biochemical pathway maps. BMC Syst Biol. 2013;7(1):116.
27. Gyori BM, Bachman JA, Subramanian K, Muhlich JL, Galescu L, Sorger PK. From word models to executable models of signaling networks using automated assembly. Molecular systems biology. 2017;13(11):954-.
28. Sharp R, Pyarelal A, Gyori B, Alcock K, Laparra E, Valenzuela-Escárcega MA, et al., editors. Eidos, INDRA, & Delphi: From free text to executable causal models. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations); 2019.
29. Liang K-W, Wang Q, Telmer C, Ravichandran D, Spirtes P, Miskov-Zivanov N. Methods to Expand Cell Signaling Models Using Automated Reading and Model Checking. Springer, Cham; 2017. p. 145-59.
30. Ahmed Y, Telmer C, Miskov-Zivanov N. ACCORDION: Clustering and Selecting Relevant Data for Guided Network Extension and Query Answering. arXiv preprint arXiv:200205748. 2020.
31. Sayed K, Bocan KN, Miskov-Zivanov N, editors. Automated Extension of Cell Signaling Models with Genetic Algorithm2018/07//: IEEE.
32. Kozen DC. Depth-First and Breadth-First Search. The Design and Analysis of Algorithms. New York, NY: Springer New York; 1992. p. 19-24.
33. Erdős P, Rényi A. ON THE EVOLUTION OF RANDOM GRAPHS.
34. Barabasi A-L, Albert R. Emergence of scaling in random networks. Science (New York, NY). 1999;286(5439):509-12.

35. Albert R, Jeong H, Barabási A-L. Error and attack tolerance of complex networks. Nature. 2000;406(6794):378-82.
36. Crucitti P, Latora V, Marchiori M, Rapisarda A. Efficiency of scale-free networks: error and attack tolerance. Physica A: Statistical Mechanics and its Applications. 2003;320:622-42.
37. Jeong H, Mason SP, Barabási A-L, Oltvai ZN. Lethality and centrality in protein networks. Nature. 2001;411(6833):41.
38. Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, Stolovitzky G. Revealing strengths and weaknesses of methods for gene network inference. Proceedings of the national academy of sciences. 2010;107(14):6286-91.
39. Prill RJ, Marbach D, Saez-Rodriguez J, Sorger PK, Alexopoulos LG, Xue X, et al. Towards a rigorous assessment of systems biology models: the DREAM3 challenges. PloS one. 2010;5(2):e9202.
40. Legay A, Delahaye B, Bensalem S. Statistical Model Checking: An Overview. Springer, Berlin, Heidelberg; 2010. p. 122-35.
41. Telmer C, Sayed K, Butchy AA, Bocan KN, Holtzapple E, Hansen CE, et al. Dynamic System Explanation, DySE, a framework that evolves to reason about complex systems. Artificial Intelligence for Data Discovery and Reuse. 2019.
42. Hagberg A, Schult D, Swart P. Proceedings of the Python in Science Conference (SciPy): Exploring Network Structure, Dynamics, and Function using NetworkX. 2008. p. 11-5.
43. Ellson J, Gansner ER, Koutsofios E, North SC, Woodhull G. Graphviz and dynagraph—static and dynamic graph drawing tools. Graph drawing software: Springer; 2004. p. 127-48.
44. Sayed K, Kuo Y-H, Kulkarni A, Miskov-Zivanov N. Dish simulator: capturing dynamics of cellular signaling with heterogeneous knowledge. Proceedings of the 2017 Winter Simulation Conference; Las Vegas, Nevada. 3242250: IEEE Press; 2017. p. 1-12.
45. Assmann SM, Albert R. Discrete Dynamic Modeling with Asynchronous Update, or How to Model Complex Systems in the Absence of Quantitative Information. In: Belostotsky DA, editor. Plant Systems Biology. Totowa, NJ: Humana Press; 2009. p. 207-25.
46. Marbach D, Schaffter T, Mattiussi C, Floreano D. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. Journal of computational biology. 2009;16(2):229-39.
47. Ghanbari M, Lasserre J, Vingron M. Reconstruction of gene networks using prior knowledge. BMC systems biology. 2015;9(1):84.
48. Mendes P, Sha W, Ye K. Artificial gene networks for objective comparison of analysis algorithms. Bioinformatics. 2003;19(suppl_2):ii122-ii9.
49. Muldoon JJ, Yu JS, Fassia M-K, Bagheri N. Network inference performance complexity: a consequence of topological, experimental and algorithmic determinants. Bioinformatics. 2019.
50. Van den Bulcke T, Van Leemput K, Naudts B, van Remortel P, Ma H, Verschoren A, et al. SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. BMC bioinformatics. 2006;7(1):43.

Figures

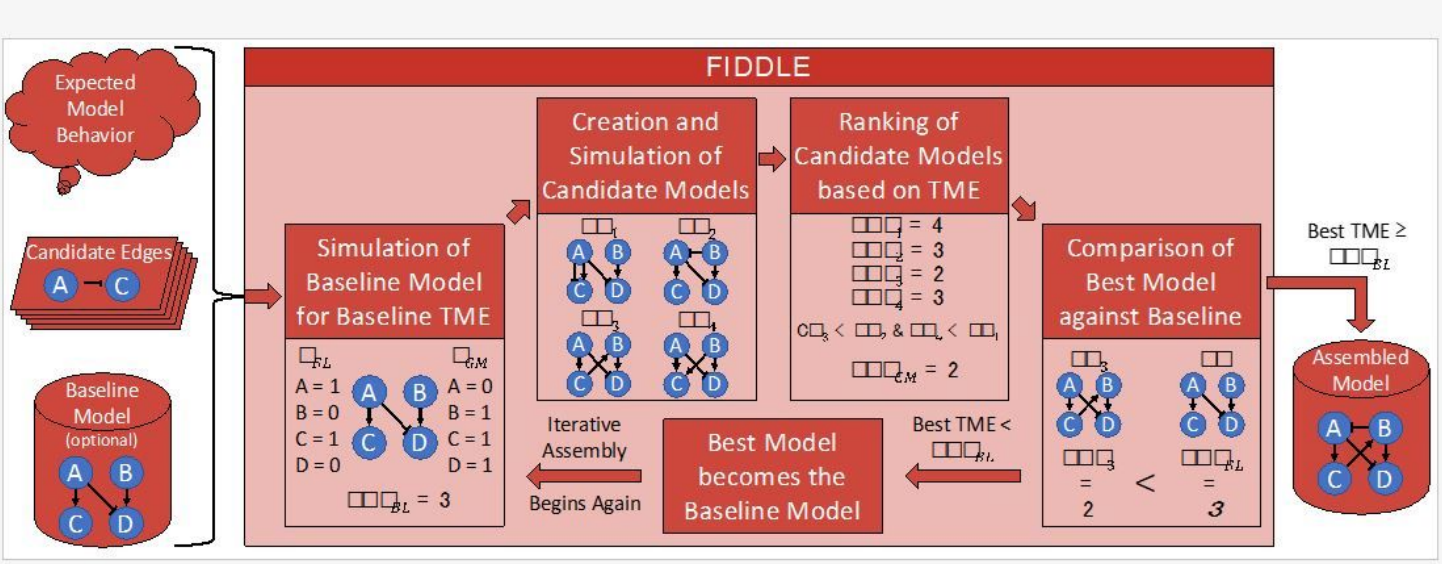

Figure 1

"Overview of the FIDDLE process" FIDDLE is a robust model assembly tool which requires candidate edges, an expectation of model behavior, and an optional baseline model. FIDDLE iteratively assembles candidate edges into candidate models, simulates them, and calculates their performance using the TME metric.

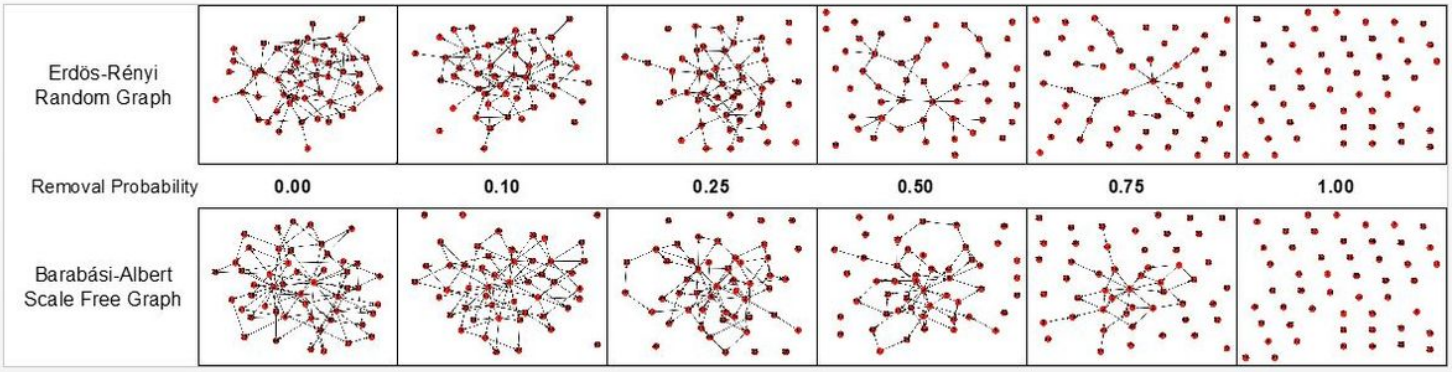

Figure 2

"Illustration of edge removal with varying probabilities" Example of the process of randomly removing model edges with different probabilities. If an edge is removed with 0.1 probability, that edge will also be removed from the network for all higher probabilities of edge removal.

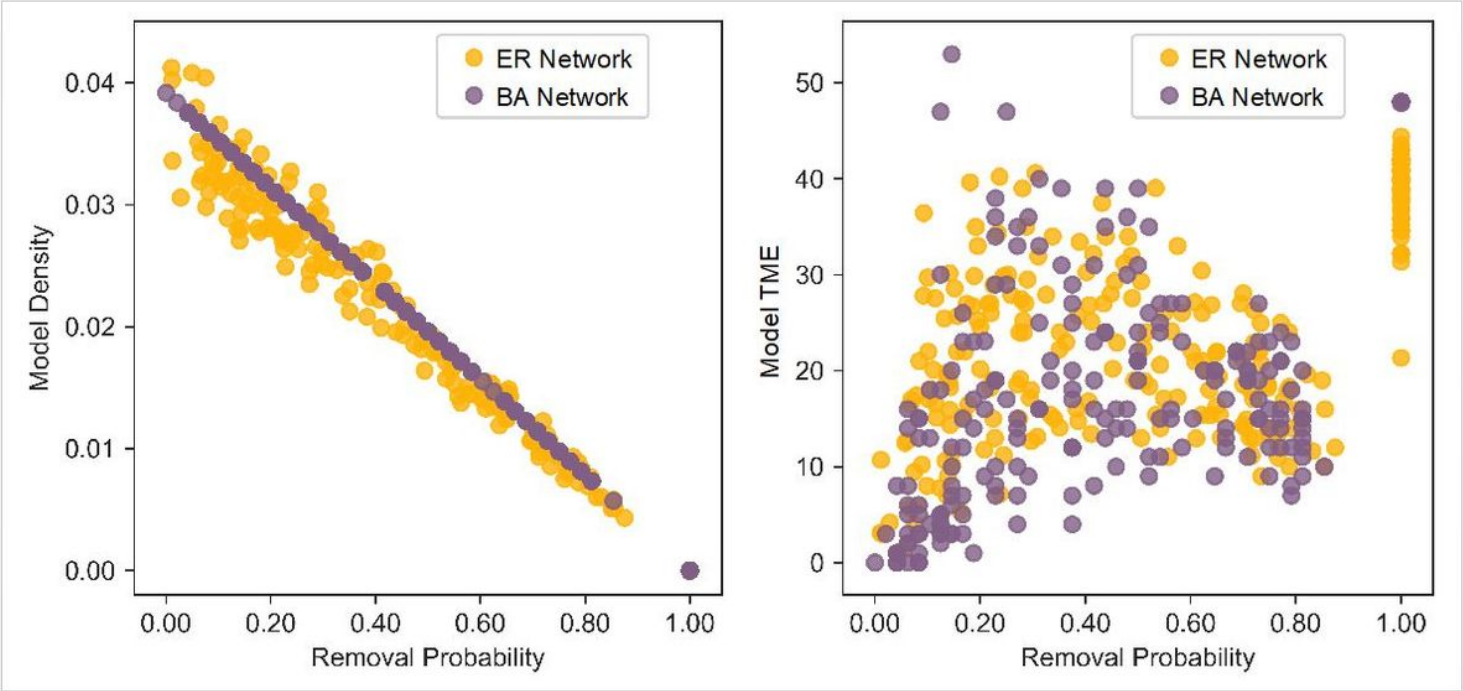

Figure 3

"Illustration of the effect of varying edge removal probability on network properties" Analysis of the effect of different edge Removal Probabilities on Model Density (left) and TME (right).

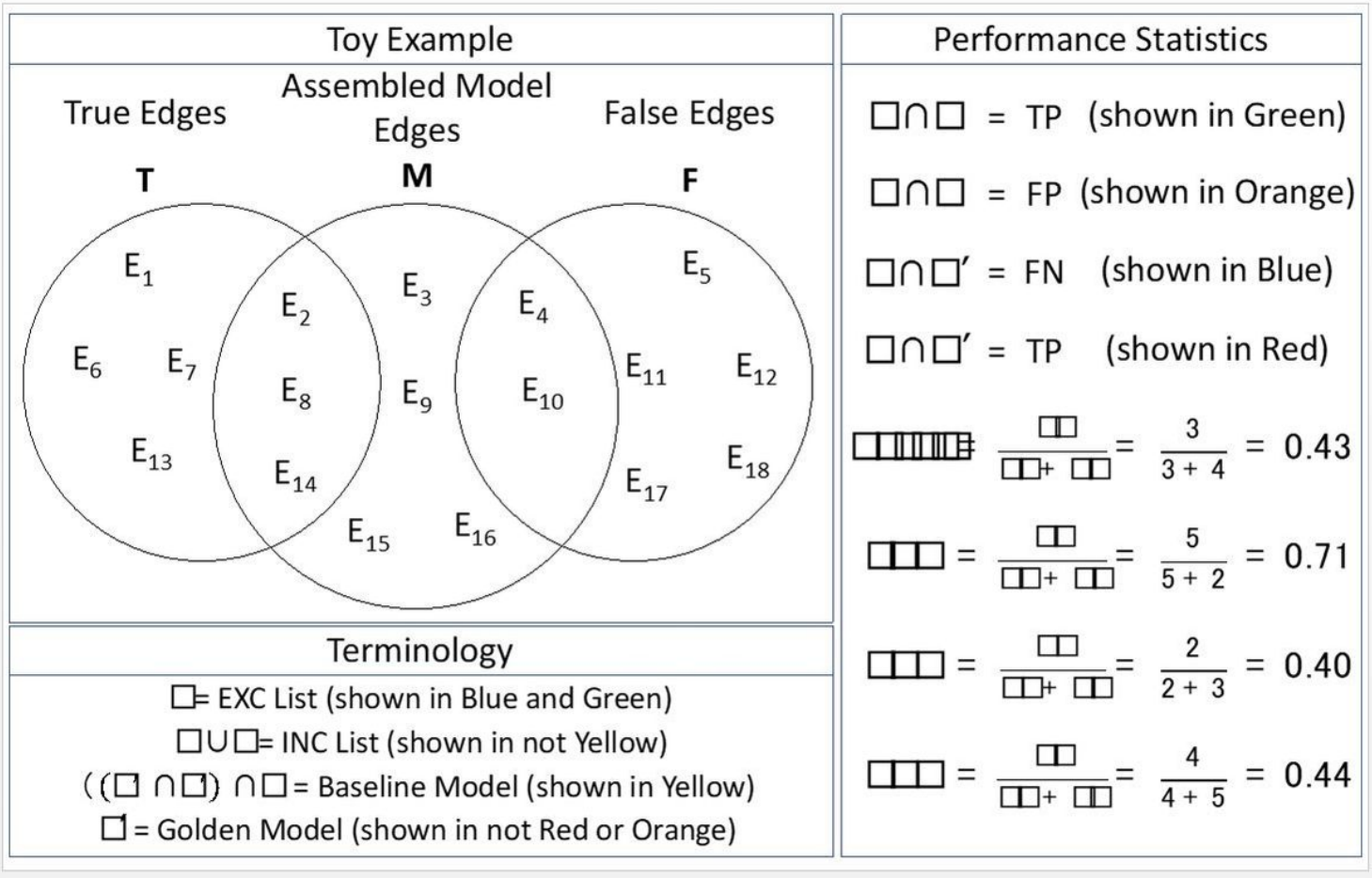

Figure 4

“Example of model assembly using the INC List” In this example, we extend a Baseline Model with the INC list of candidate edges and generate the Assembled Model. Based on whether candidate edges are part of the Golden Model, and included in the Assembled Model, we can determine whether the edges represent TP, FP, TN, or FN. We calculate Recall, TNR, FDR, and FOR from the number of TP, FP, TN, and FN in the Assembled Model.

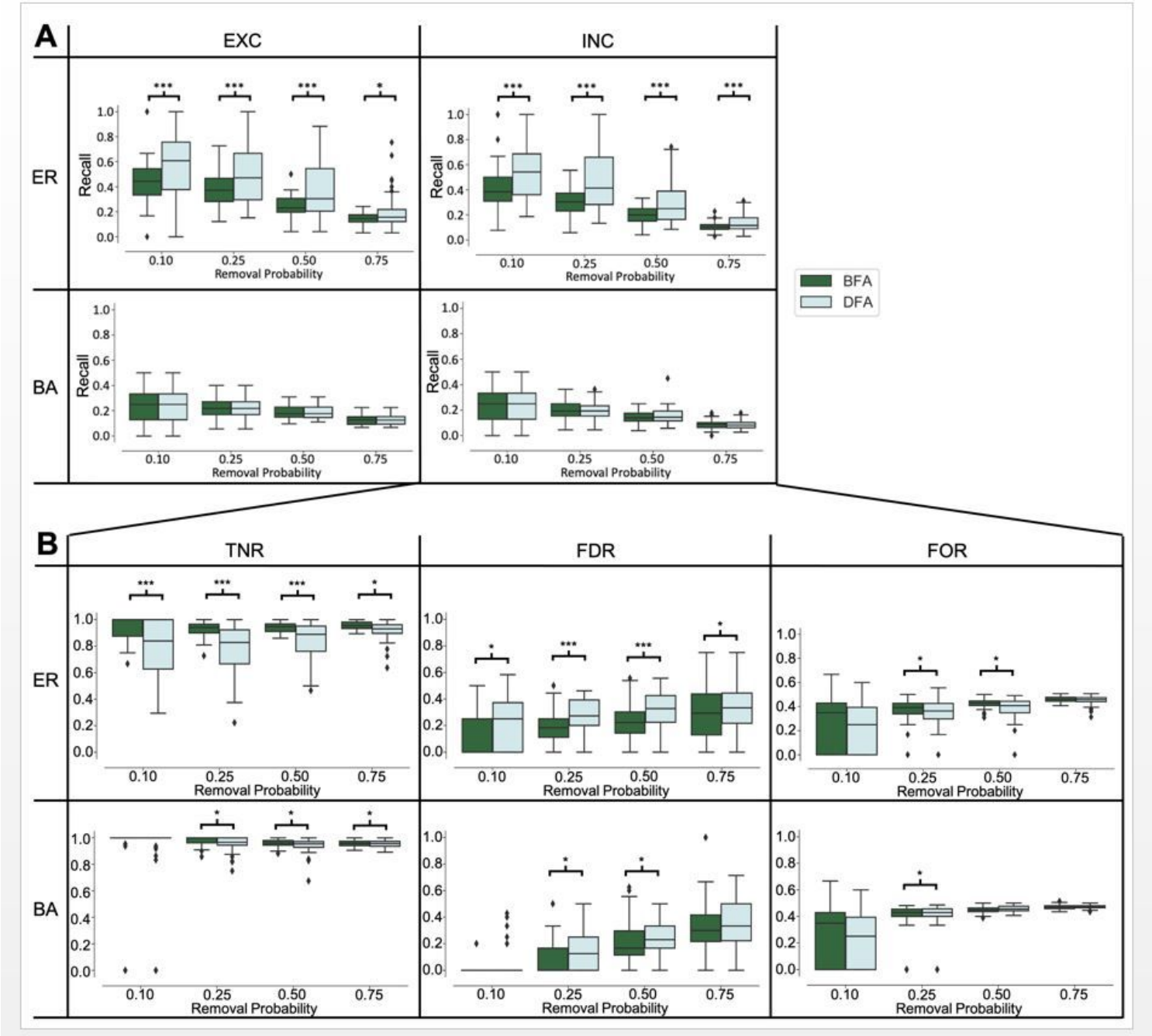

Figure 5

“Comparison of performance statistics of model extension” (A) Box plots comparing the quartile ranges of the recall of both algorithms under different conditions. The top row is for models based on the ER

network while the bottom row is based on BA networks. The left column is comparing the two algorithms when using only the EXC list, while the right column is comparing the algorithms when using the INC list, with equal number of true and false edges. Statistical significance was determined by t-test (* $p < 0.05$, *** $p < 0.001$). (B) Box plots comparing the quartile ranges of the TNR , FDR, and FOR of both algorithms under different conditions. The top row is for models based on the ER while the bottom row is based on BA. In all cases, both algorithms were given the INC list, with equal number of true and false edges. Statistical significance was determined by t-test (* $p < 0.05$, *** $p < 0.001$).

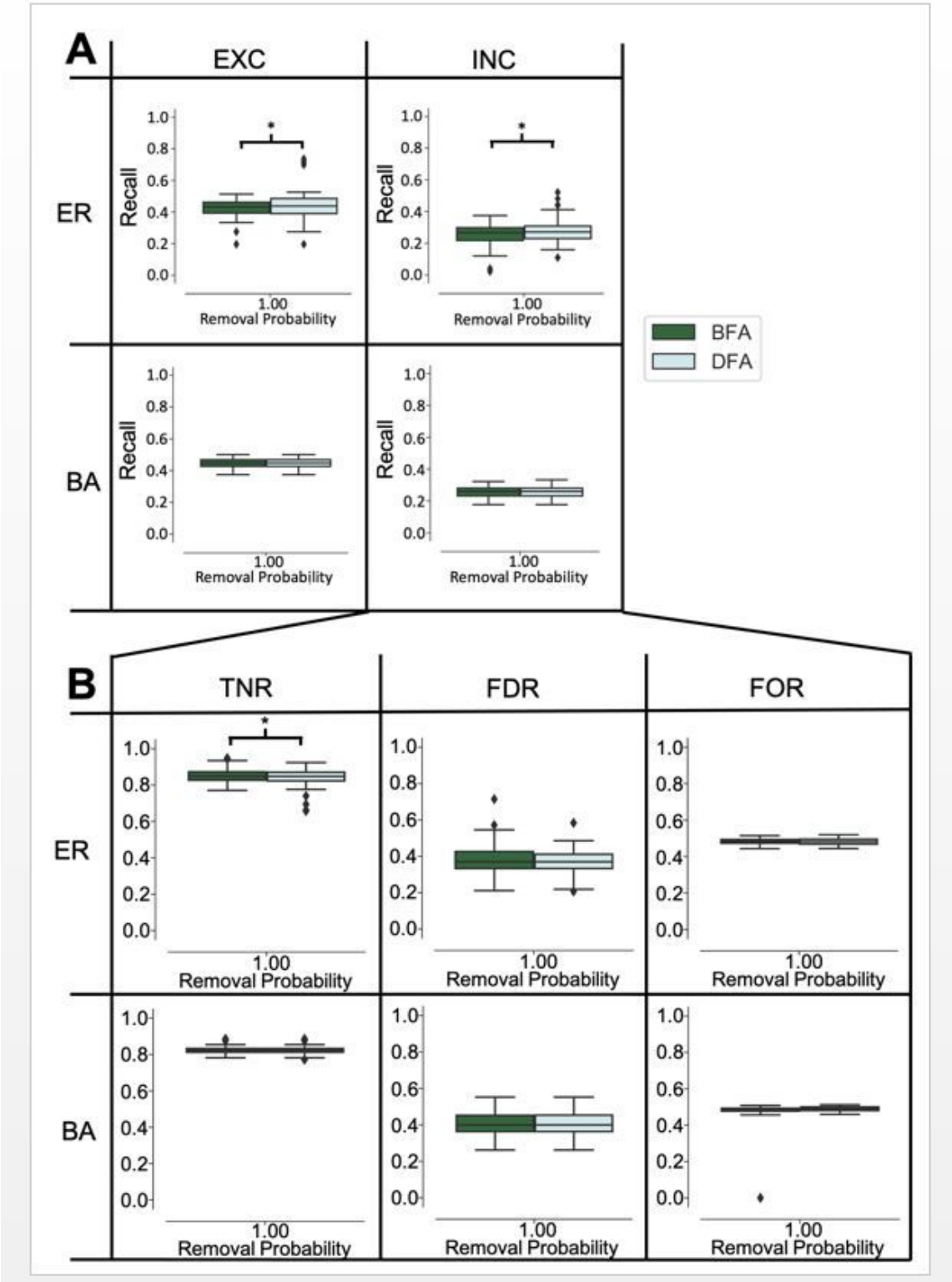

Figure 6

“Comparison of performance statistics of model creation” (A) Box plots comparing the quartile ranges of the recall of both algorithms under different conditions. The top row is for models based on the ER random network while the bottom row is based on BA networks. The left column shows the results obtained when using the EXC list in both algorithms, while the right column shows the results when the algorithms were given the INC list, with equal number of true and false edges. Statistical significance was determined by t-test (* $p < 0.05$, *** $p < 0.001$). (B) Box plots comparing the quartile ranges of the TNR, FDR, and FOR of both algorithms under different conditions. The top row is for models based on the ER random network while the bottom row is based on BA networks. In all cases, both algorithms were given the INC list, with equal number of true and false edges. Statistical significance was determined by t-test (* $p < 0.05$, *** $p < 0.001$).

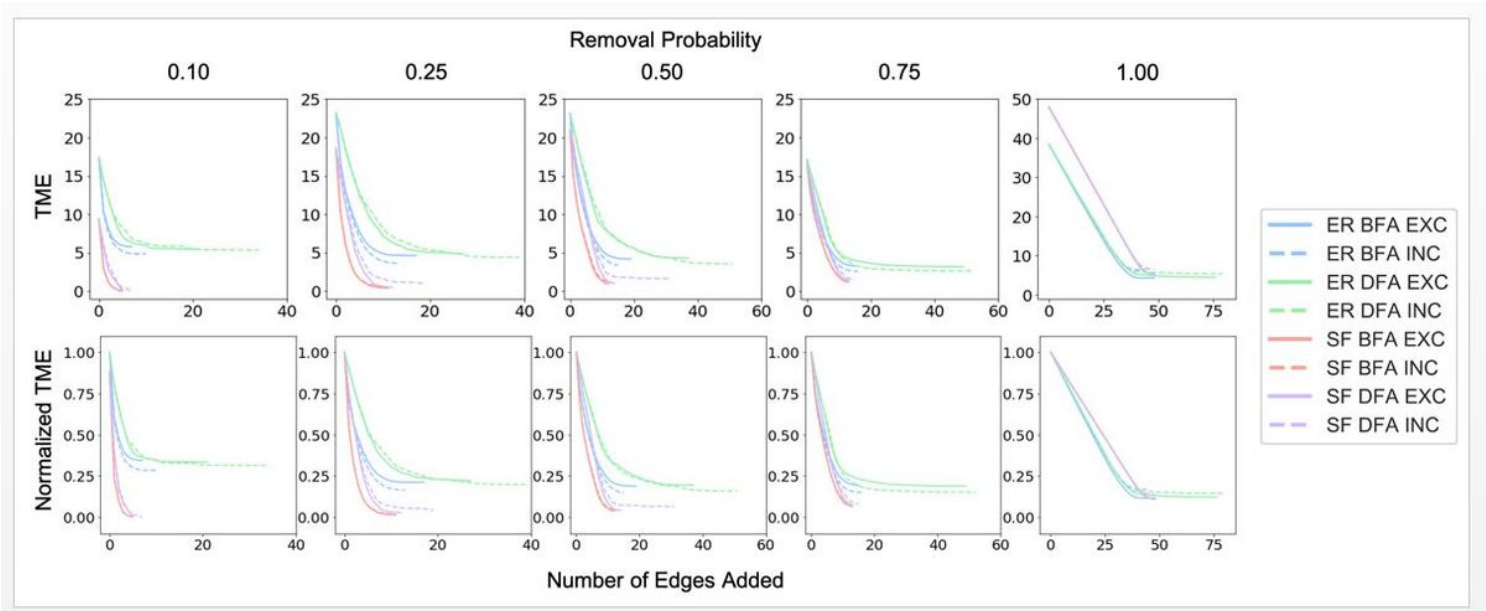

Figure 7

“Model performance as edges are added back into the model” In the top row of this figure, we see plots of the model TME (y-axis) with number of edges added (x-axis). In the bottom row, we see the plots normalized to TME.

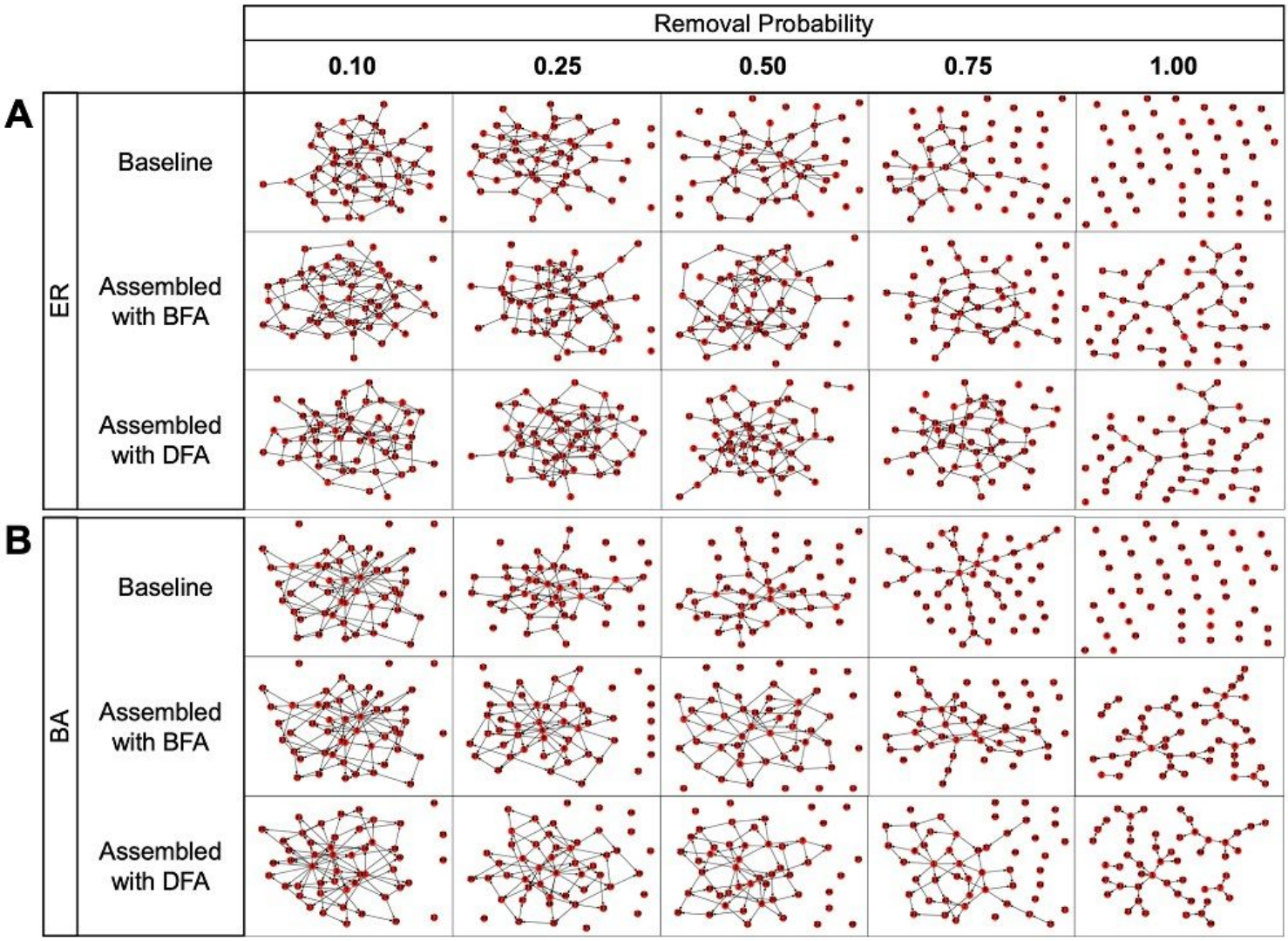

Figure 8

“Examples of model assembly in random and scale-free networks” Examples of BFA- and DFA-based model assembly for: (A) ER network models, and (B) BA network models.

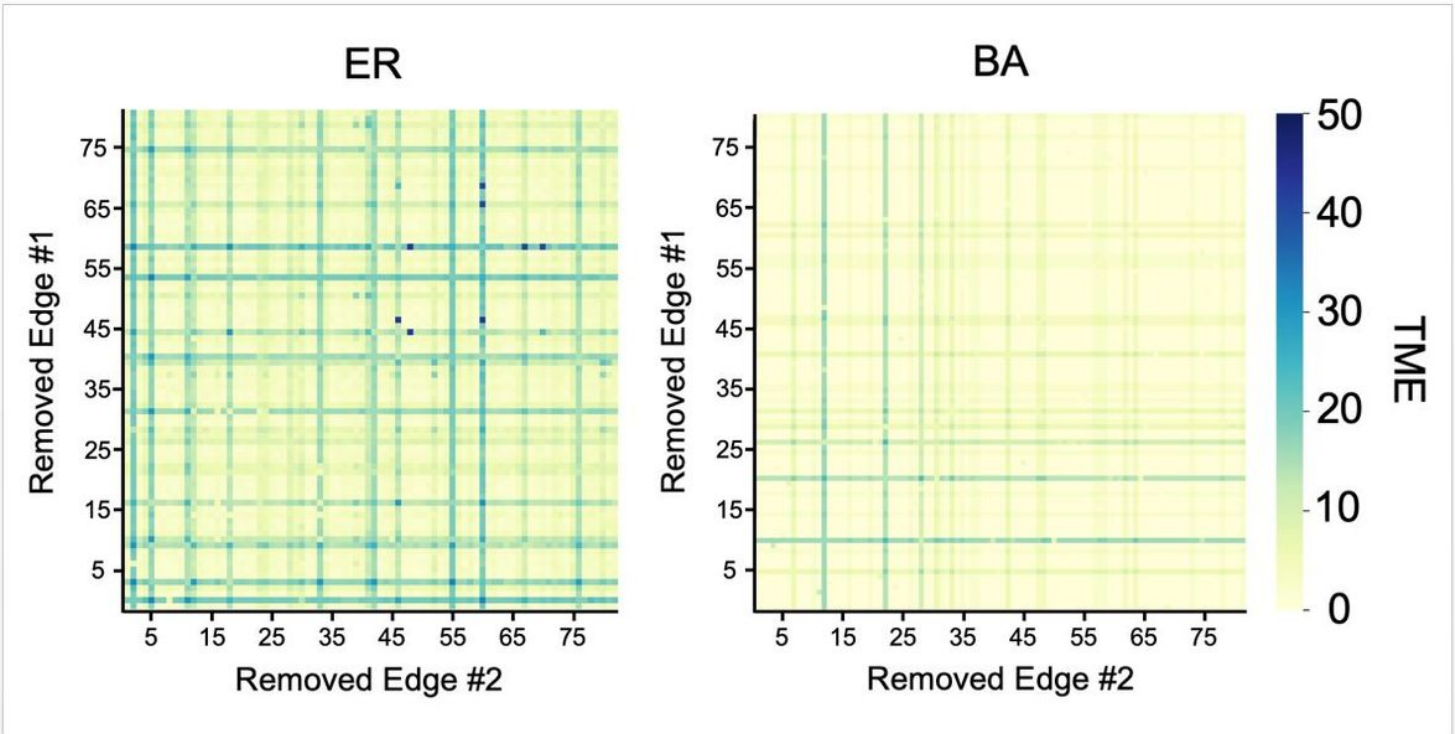

Figure 9

“Heatmap of network response to combinatorial edge removal” These heatmaps compare the differing responses of ER and BA networks under random edge removals. Columns represent the TME for a model with the same edge missing, while rows also represent the TME for a model with a different edge missing. The diagonal represents models with only one edge missing.

Algorithm: Breadth First Addition (BFA)
Input: baseline model (M_{BL}), list of edges (E_{NEW}), TME of the baseline model (TME_{BL}), expected performance of the golden model (Q_{GM}),
Output: extended baseline model that minimizes the TME

while (TME_{BL} != 0) and (E_{NEW} != [])
    Initialize scores = []
    for edge in E_{NEW}:
        M_{CM} = a candidate model is created by adding the edge to the M_{BL}
        simulate M_{CM}
        $TME(Q_{GM}, Q_{CM})$ use the TME function to compare the candidate model to the expected performance of the golden model
        Append TME_{CM} to the scores list
    end for
    find index = min(scores)
    TME_{CM} = scores(index)
    if TME_{CM} < TME_{BL}:
        M_{CM} = a candidate model is created by adding the edge to the M_{BL}
        M_{BL} = M_{CM}
        E_{NEW}.delete(index)
        TME_{BL} = TME_{CM}
    else
        return M_{BL}
    end if
end while
return M_{BL}

Figure 10

“Pseudocode of the BFA Algorithm” The basic structure of the BFA algorithm in a pseudocode format.

Algorithm: Depth First Addition (DFA)
Input: baseline model, list of edges, expected performance of the golden model, current TME
Output: extended baseline model that minimizes the TME

```
while (TME_BL != 0) and (E_NEW != [])
        E_added = FALSE
        for edge in E_NEW:
                M_CM =  a candidate model is created by adding the edge to the M_BL
                simulate M_CM
                TME(Q_GM, Q_CM) use the TME function to compare the candidate model to the
                        expected performance of the golden model
                if TME_CM < TME_BL:
                        M_CM =  a candidate model is created by adding the edge to the M_BL
                        M_BL = M_CM
                        E_NEW.delete(edge)
                        TME_BL = TME_CM
                        E_added = TRUE
                        exit for loop
                end if
        end for
        if (E_added == FALSE)
                return M_BL
        end if
end while
return M_BL
```

Figure 11

“Pseudocode of the DFA Algorithm” The basic structure of the DFA algorithm in a pseudocode format.