BIOCODE: A Data-Driven Procedure to Learn the Growth of Biological Networks

Emre Sefer

Abstract—Probabilistic biological network growth models have been utilized for many tasks including but not limited to capturing mechanism and dynamics of biological growth actitivies, null model representation, capturing anomalies, etc. Well-known examples of these probabilistic models are Kronecker model, preferential attachment model, and duplication-based model. However, we should frequently keep developing new models to better fit and explain the observed network features while new networks are being observed. Additionally, it is difficult to develop a growth model each time we study a new network. In this paper, we propose BIOCODE, a framework to automatically discover novel biological growth models matching user-specified graph attributes in directed and undirected biological graphs. BIOCODE designs basic set of instructions which are common enough to model a number of well-known biological graph growth models. We combine such instruction-wise representation with a genetic algorithm based optimization procedure to encode models for various biological networks. We mainly evaluate the performance of BIOCODE in discovering models for biological collaboration networks, gene regulatory networks, metabolic networks, and protein interaction networks which features such as assortativity, clustering coefficient, degree distribution closely match with the true ones in the corresponding real biological networks. As shown by the tests on the simulated graphs, the variance of the distributions of biological networks generated by BIOCODE is similar to the known models variance for these biological network types.

Index Terms—Biological Networks, Graph Mining, Network Growth Models, Algorithms

---

1 INTRODUCTION

Research of the dynamics by which temporal evolution of biological networks occur is a key component in understanding how such biological networks operate. Especially, understanding the dynamics and appearance of topological features in biological networks such as modularity, assortativity, disassortativity, and shrinking diameter is notably important. Creating idealized graph growth models is a successful method in understanding how such graph features emerge in the first place. Examples of such idealized graph growth models are preferential attachment models [e.g. [1], [2]], duplication/mutation models [3], [4], [5], [6], [7], [8], the Kronecker model [9], [10], forest fire model [11], and other models [12], [13], [14], [15], [16], [17], [18], [19]. Those models describe the biological networks growth mechanistically and probabilistically. In common, those models express such growth by union of various operations such as node duplication, node expansion, node/edge creation, node/edge deletion, and influence propagation.

Besides simulating realistic biological network growth, such graph growth models are used for other purposes in different applications where growth dynamics do not need to be interpretable in some applications. As an example, growth models may help in inferring the historical networks [20], may be helpful in anonymization [9], can be utilized to test the performance of lengthy large-scale graph methods, may be used as null models to detect anomalous graph features.

The first theoretical studies on network models has begun with Erdos-Rényi model [21]. Following research on network models found small-world [22] and a scale-free node degree distribution [1] properties as frequent real world network attributes and designed growth models to generate such properties. Subsequent growth models included additional properties as objectives across different domains. One such property is clustering coefficient for protein interaction networks [6], [7], [23] which resulted in DMC (duplication, mutation, with complementarity) model. Another property is shrinking diameter for an temporally evolving social network [11] which ended up in forest fire model. Following attempts [24], [25] have designed manual models which fit multiple extra properties simultaneously and have generated real-word like graphs. More recent models try to also match richer node features in addition to real world network topologies [26], [27]. It is a challenging task to create a feasible, parsimonious, network growth model that fits well to the data. As we study large-scale and different types of networks, we will identify new properties which require developing novel growth models. Nevertheless, the degree custom-made growth models model the desired network properties will depend on model designer's capabilities and creativity.

In this paper, we come up with a formal characterization of network growth models which encode well-known and frequently-used network growth models in addition to many more undiscovered models. Additionally, we introduce an optimization framework which can automatically discover novel models that better fit the desired network attributes in the aforementioned formal setting. Models learned by the proposed framework can generate many sample networks across different classes matching input properties. These learned models are relatively easy to understand and interpretable with an effort of certain degree. In many scenarios, graph motifs can be frequently mined

- *Emre Sefer is with the Department of Computer Science, Ozyegin University, Istanbul, Turkey.*

  E-mail: emre.sefer@ozyegin.edu.tr

in the set of generated growth models as generated models are in general distinct and better fit the desired properties. These mined motifs are in general successfull in modeling a specific network attribute. Lastly, in numerous situations, computationally derived growth models outperform human-designed models in matching real-world attributes.

Among the existing work, only a few research has focused on automatically designing network growth models. Some of the earlier frameworks can adapt existing models to novel graph data via recalculating model parameters governing graph dynamics. As an example, Kronecker graph model parameters can be estimated better by integrating Markov Chain Monte Carlo (MCMC) method to its parameter estimation, especially for matching several very large network attributes [9]. Another example is estimating the parameters for additional recursive growth models [24]. However, those methods are restricted to estimating network model parameters and they cannot mimick novel network growth dynamics. [28] focus on selecting the best models among a few current models. According to this work, DMC model fits protein-protein interaction graphs the best [7]. Nonetheless, their approach does not fit parameters for current models and does not generate novel models.

Here, we design a framework called BIOCODE to address those inadequacies via encoding fundamental graph operations and other graph model defining structures as instructions operating in a virtual machine with multiple registers. Intuitively, providing an effective set of atomic instructions and network growth structures are our main motivation. A series of such consecutive instructions define a network growth dynamics iteration, and recurrent iterations of these series of instructions temporally grow a graph. One of our main contribution in BIOCODE is that only a number of operations are enough to model a duplication model, a forest-fire-like model, preferential attachment model, and supposedly more growth models. Among these operations, 4 of them have parameters whereas the rest of operations are parameterless. Moreover, the machine operated by BIOCODE operations has only 3 registers. Such smaller machine design restrains the total number of candidate programs which allows for an efficient search of the solution space by a genetic algorithm.

BIOCODE allows us to learn biological network growth models automatically and quickly which assures a number of biological networks key features. BIOCODE models often outperform human-designed models in fitting to the fundamental topological graph features of clustering coefficient, assortativity, and degree distribution. Particularly, model learned by BIOCODE on yeast protein interaction networks [29] generate graphs better than the popular DMC model which simulates these protein interaction networks in terms of agreeing to the observed the degree distribution and clustering coefficient values. Additionally, we can outperform Kronecker model with the best parameters [11] in terms of generating graphs that match the degree distribution and assortativity of a recent biological co-authorship network [30]. Lastly, the models identified by BIOCODE is better than a Kronecker model in terms of simultaneously matching node degree distribution, assortativity, and clustering coefficient of a gene regulatory network. In our

settings, the graphs generated by BIOCODE learned models are more diverse than the ones generated by the competing human-designed models, showing that models generated by BIOCODE are correctly random network models.

Even though the proposed BIOCODE framework generates unattributed graphs, the process suggested by BIOCODE is quite common and widespread. We can extend BIOCODE to different graph classes. The technique pointed by BIOCODE allows for automatic and more systematic graph growth dynamics study.

2 THE BIOCODE FRAMEWORK

We come up with BIOCODE framework where we can express growth models programmatically and briefly. A register machine together with 15 instructions executing on the register machine are defined. Every series of machine instructions is a correct program concisely encoding a biological graph growth model. Basic and particular operations impacting the topological graph features are included in BIOCODE instruction set. Few instructions are included to direct the program flow and manage registers. Mainly, instructions that are natural growth model structures are included in BIOCODE instruction set. The selected instructions may represent a number of unknown and existing biological models.

BIOCODE machine changes an evolving graph's topology while it executes a program. Every single execution of BIOCODE instructions in a program outlines a growth process single step. We execute BIOCODE program from scratch till the end t times in order to evolve a network for t time steps. t is linked to the output graph size, and t is an implicit parameter for each BIOCODE program. BIOCODE machine registers are filled randomly with the graph nodes between the successive program calls, modeling the successive growth steps. When combined with a number of randomized instructions, this randomization between successive program runs aids BIOCODE to encode probabilistic growth models as BIOCODE programs. Consequently, the same program's separate executions almost always generate dissimilar biological networks.

2.1 BIOCODE registers

BIOCODE executes instructions on a 3 register virtual machine. These registers are r0, r1, r2 which can store positive integers. Register values mainly correspond to node IDs, even though their values may also correspond to parameters used by several instructions. Register may take a special value NIL showing the register is idle. BIOCODE keeps a program counter, PC which displays the presently running instruction. Once an instruction is run, program counter is increased in order to maintain a sequential execution of the program as long as one of the control flow instructions updates the program counter. As achieved by REWINDinstruction below, BIOCODE programs can modify themselves to support looping to a certain extenrt. As in a traditional computer program, program is terminated when program counter location passes beyond the program length.

Let V be the evolving graph's nodes, BIOCODE incorporates a limited amount of memory $L : V \rightarrow V$ which

is able to store a single node ID for each node in V. Here, $L(v)$ value on vertex v may not necessarily be v's node ID, instead it may be an another node's ID. This ability of graph vertices to store IDs of any other graph vertex is the key factor on BIOCODE programs spreading a vertex's influence in the evolving graph (See Section 2.2 for details). v does not have a label when $L(v)$ =NIL. It is possible to have more complicated influence operations and memory models. However, our experiments show that good agreement can be achieved across multiple different settings by our proposed minimal design.

2.2 BIOCODE instruction set

It is a difficult and long-established problem to design instructions on virtual and physical processors. We carefully included an operation in BIOCODE instruction set if such operation represents a fundamental graph operation. Each BIOCODE instruction is easily comprehensible and resemble the operations seen in human-created graph growth models. Union of those instructions may end up in growth models which can generate graphs with the required features. BIOCODE instructions may be expanded by including further instructions to incorporate novel graph growth processes. Optimizing a hard objective becomes relatively easier via a carefully-designed good instruction set. However, totally resolving instruction set design problem is not this paper's focus. Instead, via our experiments, we show that a single instruction set in Table 1 performs quite accurately for many biological graph classes.

2.2.1 BIOCODE instructions

We can divide BIOCODE instructions in Table 1 into four groupings: 1- Register instructions, 2- Control flow instructions, 3- Graph instructions, and 4- Influence instructions. Among these groupings, the first two groupings focus on managing BIOCODE state and managing the BIOCODE program's control flow as suggested by their names. The third and fourth groupings focus on transforming the evolving graph's topology. See Table 1 below for a full operations list. Next, we characterize the impact of these operations on the resulting generated graphs and on BIOCODE machine state. Section 3 discusses the expressibility of a number of graph growth models by those operations.

Register instructions: One can directly manage the contents of 3 registers by the register instructions. The CLEAR r2 adjusts the r2's value to NIL. The SWAP instructions swaps r0 and r1's values. The SAVE instruction puts r0's value also into r2. Contrarily, LOAD copies r2's value into r0. The SET(i) operation puts integer i to r2.

Control flow instructions: BIOCODE operations order of execution is modified by these control flow instructions. SKIP_INSTRUCTION(p) moves the program counter by 2 with probability p. In this case, the following instruction is conditionally run with probability $1 - p$ by such probabilistic movement. Another instruction in this category, REWIND(r, i), models for loop-like behaviours. Its first critical argument r specifies how manu times program counter must be decreased when REWIND(r, i) is run. In another words, this parameter also models how far program counter must go in reverse direction. Its second argument

i defines how many times the operations must be run. Whenever BIOCODE executes REWIND(r, i), it is updated via decreasing i by 1. Whenever i becomes 0, program counter value will not be rewound as BIOCODE will stop execution. These REWIND(r, i) parameters are reset within successive program runs.

Graph instructions: The graph instructions primarily modify topology of graph. The GENERATE_EDGE instruction generates an edge in the graph. BIOCODE retrieves the node IDs from r0(u) and r1(v) registers, and generates $\{u, v\}$ edge. GENERATE_EDGE instruction does not alter the register states, so this instruction does not have an impact if $\{u, v\}$ edge is already part of the graph. The NEW_NODE instruction generates a new vertex in the evolving graph. BIOCODE retrieves vertex ID from register r0. The RANDOM_EDGE instruction uniformly and randomly chooses an edge $\{u, v\}$ in the graph, and puts corresponding vertices u into r0 and v into r1. Finally, RANDOM_NODE instruction picks a vertex randomly and uniformly and puts this vertex to r0.

Influence instructions: Graph vertices can have an influence upon other nodes by influence instructions. $L(v) = u$ means vertex u influences v. Such influence mechanism is essential in generating graphs with different types of features. One example is homophily in which common topological neighborhoods are shared by vertices to a certain degree. A vertex can influence a subset of vertices in its neighbourhood by the main influence instruction, INFLUENCE_NEIGHBOURS(p). While running the INFLUENCE_NEIGHBOURS(p) instruction, BIOCODE retrieves the vertex ID u from r0 where u turns into the influential or central vertex. Afterwards, BIOCODE propagates this u mark to u's every neighbour v by assigning $L(v) = u$ probabilistically. Such probabilistic assignment takes place independently for each such vertex with probability p. In turn, every newly marked node v marks its neighbours having content u with probability $p^{d(u,v)}$, where $d(u, v)$ is the distance of shortest path between u and v. In this case, vertices v such that $d(u, v)$ <r2 might be impacted by this influence instruction unless r2 is NIL. When r2 =NIL, the influence instruction keeps executing till the probabilistic process terminates and so process does not mark any more vertices.

There are 3 further instructions that INFLUENCE_NEIGHBOURS(p) instruction operates together with: CLEAR_INFLUENCED instruction removes L values such that the following instructions may operate with a clean memory. The DISCONNECT_FROM_INFLUENCED instruction retrieves a vertex u from r0 register, and deletes all edges $\{u, v\}$ satisfying $L(v) = u$. CONNECT_TO_INFLUENCED generates edges between the vertex in r0, w, and all vertices marked with the content of r1 = u. CONNECT_TO_INFLUENCED generates edges $\{w, v\}$ for all v such that $L(v) = u$. Making the two vertices neighbourhoods more like each other is a common mechanism provided by CONNECT_TO_INFLUENCED. Figure 1 represents all those influence instructions.

Table 1: Complete set of BIOCODE instructions

Operation Type	Operation	Definition
Register	CLEAR r2	set r2 to NIL
	SWAP	Swap r0 and r1 contents
	SAVE	Clone register r0 content to r2
	LOAD	Clone register r2 content to r0
	SET(i)	Clone vertex ID to r2
Control flow	SKIP_INSTRUCTION(p)	Pass over the following operation
	REWIND(r, i)	Go back r lines i times
Influence	CLEAR_INFLUENCED	Clean all tags in L
	DISCONNECT_FROM_INFLUENCED	Delete edges to the neighbours tagged with u
	CONNECT_TO_INFLUENCED	Add edges to neighbours tagged with u
	INFLUENCE_NEIGHBOURS(p)	Tag neighbours with u
Graph	GENERATE_EDGE	Generates an edge
	NEW_NODE	Introduces a new vertex
	RANDOM_EDGE	Randomly selects an edge
	RANDOM_NODE	Randomly selects a vertex

Figure 1: The summary of 3 influence instructions. First of all, vertex u puts an influence mark on its neighbours with probability p. Then, the influenced neighbours v propagates the influence to their neighbours with probability p^2. When vertex u separates from its marked neighbours, 2 gray edges shown by the gray arrows will be deleted from the graph. Lastly, w may connect to other vertices u has put an influence upon.

3 REPRESENTING EXISTING MODELS

We show the general applicability of BIOCODE by showing its expressive power on 3 well-studied biological network growth models: forest fire (FF) [11], duplication and mutation with complementarity (DMC) [7], Barabási-Albert (B-A) [1]. We code BIOCODE programs matching these models key features. Those biological growth models illustrate different topological aspects by matching different styles of realistic biological networks. As an example, graphs generated by the DMC model exhibit a wide spectrum of clustering coefficients matching the ones seen in protein-protein interaction networks. Similarly, graphs generated by the FF model show densification power law attribute and shrinking diameter while they evolve. Although there are major variations in these growth models dynamics and in the graphs features they generate, rather elementary BIOCODE programs can represent those models by using the same set of basic operations. BIOCODE operations are reused over separate models indicating their high-quality in disclosing a variety of network growth dynamics.

3.1 Barabási-Albert

According to B-A growth process, newy added nodes attach to higher-degree nodes with a higher probability. [1]. B-A process produces networks with scale-free distribution which is frequently observed in real-world biological and social networks. According to the scale-free distribution, too many low-degree vertices in the distribution are followed by few very high degree vertices.

Algorithm 1 defines a BIOCODE program that similarly mimicks the B-A model. Even though fine differences between the original B-A model and the program exist, the graphs generated by B-A model resemble the ones produced by BIOCODE program. The B-A model's fundamental part is defined in lines 3–5. Among these lines, the RANDOM_EDGE operation in line 3 selects an edge that has high-degree vertices at its endpoints with high probability. The probability of a randomly selected edge e containing vertex u is relative to u's degree: $\frac{d(u)}{E} = 2\frac{d(u)}{\sum_{v \in v} d(v)}$. While e is chosen, operations in lines 4 and 5 randomly select an endpoint for e, ensuring vertex selection within e is without bias. The model defined by Algorithm 1 picks up vertices relative to their degree as required by B-A model with a small difference; contrary to B-A model, as BIOCODE program runs, vertex degrees $d(u)$ are modified. Then, instruction in line 7 connects the newly introduced vertex to u accomplishing the newly introduced vertex's preferential attachment. The following REWIND instruction loops over this process such that the newly introduced vertex is connected to i current vertices.

3.2 Duplication and Divergence

DMC model (Duplication and mutation with complementarity model) [7] focuses on generating graphs that mimick protein interaction graphs topological attributes. Network evolves by the duplication of current vertices in DMC model. The DMC model has q_{con} and q_{mod} parameters controlling the network growth as follows: Every newly introduced, duplicated vertex u selects an anchor vertex

Algorithm 1 B-A

1: NEW_NODE   ▷ Generate a new vertex u
2: SAVE
3: RANDOM_EDGE   ▷ Randomly pick up an edge e
4: SKIP_INSTRUCTION(0.5)▷ Randomly select vertex v of e
5: SWAP
6: LOAD
7: GENERATE_EDGE   ▷ Generate an edge between vertices u and v
8: REWIND$(5, i)$ ▷ Randomly attach newly introduced vertex to i current vertices

---

Algorithm 2 DMC

1: RANDOM_NODE   ▷ Place a randomly selected vertex v in r0
2: SET(1) ▷ Set r2 (k-hop for influence) to 1
3: INFLUENCE_NEIGHBOURS(1.0) ▷ Influence neighhbours of v
4: SWAP   ▷ Swap r0 and r1
5: NEW_NODE   ▷ Introduce a vertex u to the graph and place it in r0
6: CONNECT_TO_INFLUENCED ▷ Attach/Connect vertex u to influenced vertices
7: CLEAR_INFLUENCED
8: INFLUENCE_NEIGHBOURS$(\frac{q_{mod}}{2})$ ▷ Influence the neighbours of u
9: SWAP
10: INFLUENCE_NEIGHBOURS$(\frac{q_{mod}}{2})$ ▷ Influence the neighbors of v
11: DISCONNECT_FROM_INFLUENCED   ▷ Remove edges from vertex v
12: SWAP
13: DISCONNECT_FROM_INFLUENCED   ▷ Remove edges from vertex u
14: CLEAR_INFLUENCED
15: SKIP_INSTRUCTION$(1.0 - q_{con})$ ▷ Pass over addition of edge $\{u, v\}$
16: GENERATE_EDGE   ▷ Create the edge with q_{con} probability

---

v and connects to all neighbours of v. For every vertex w adjacent to both v and u, an edge is randomly selected attaching w either to v or u, and selected edge is deleted with q_{mod} probability. Lastly, u and v are connected with q_{con} probability by introducing an edge between them. The frequent occurence of gene duplication is the main motivation behind such network growth dynamics especially in protein interaction networks, where genes synthesizing proteins within the genome are frequently duplicated. At the beginning, the duplicated genes are identical copies so the resulting proteins keep all of the interactions seen in the original protein. Nonetheless, once duplication is over, the interactions between the original and duplicated genes begin to differentiate as the evolutionary pressure on genes in keeping the original interactions is decreased. We design Algorithm 2 in BIOCODE which approximates the DMC model quite closely.

Algorithm 2 introduces the DMC model coded in BIOCODE that is somewhat different than the one proposed by Vazquez et al. [7]. In our process, we cannot exactly simulate the process of choosing the common neighbours of v and u with q_{con} probability, and removing the edge to either of them. Though, we can accomplish such dynamics similarly by influencing the common neighbours of each vertex with probability $\frac{q_{mod}}{2}$ as shown in lines 8 and 10 once v's neighborhood are duplicated to u. In this case, influence operation behaves precisely same as the traditional DMC operation only if the influenced neighbours do not intersect with each other. A neighbouring vertex might be influenced by both v and u. Complementarity attribute of DMC model means that the edge to either v or u is deleted, but not both. In our corresponding BIOCODE program, complementarity is kept as the program will overwrite the mark on the common vertex, assuring program can only delete one of the edges $\{v, w\}$ and $\{u, w\}$. That process may end up with q_{mod} values which results in a marginally different impact in BIOCODE procedures. However, BIOCODE produced DMC graphs exhibit clustering coefficents (section 5.3) and Zipf plots similar to the original DMC, that are the key attributes DMC model creators have stressed out in their paper. Besides, the graphs generated by BIOCODE Algorithm 2 exhibit clustering coefficients and Zipf plots similar to the ones found in yeast protein interaction graph. So, even though there are fine differences, the BIOCODE algorithm 2 keeps the fundamental components and features of the true DMC model.

3.3 Forest Fire

[11] introduced the forest fire (FF) model to better model the frequently observed real-world network properties such as temporal densification of the graph under a certain parameter range, shrinking diameter, and in and out-degree scale-free degree distributions. The forest fire model can be easily and intuitively explained from graph growth prospect. Here, we introduce a hardly changed model which applies to undirected graphs. Once a newly introduced vertex u is added to the graph, such vertex selects a current vertex v randomly and uniformly which then acts as an agent and the edge between v and u is joined. Afterwards, forest fire model draws a natural number n from a geometric distribution with success probability b, and v's n neighbours are selected and burned. FF model introduces an edge from vertex u to each of those burned vertices, and the procedure of choosing a number of neighbouring vertices and burning these vertices is recursively rerun.

FF model is encoded by BIOCODE program in Algorithm 3. Graphs generated by the BIOCODE program and the FF model are same in terms of fundamental graph features. Particularly, the graphs generated by Algorithm 3 display densification power law and shrinking diameter for certain range of parameters during graph evolution over time.

4 LEARNING BIOCODE MODELS

By expressing biological network growth models in terms of a number of BIOCODE operations, learning a biological graph growth model over BIOCODE can be expressed formally as an optimization problem over the BIOCODE

Algorithm 3 FF

1: RANDOM_NODE   ▷ Place a random vertex in r0
2: CLEAR r2   ▷ Clean r2 contents for complete graph influence
3: INFLUENCE_NEIGHBOURS(b)   ▷ Propagate influence recursively as breadth-first
4: SWAP   ▷ Put the random vertex into r1
5: NEW_NODE   ▷ Introduce newly created vertex, u
6: GENERATE_EDGE
7: CONNECT_TO_INFLUENCED ▷ Connect/Attach vertex u to influenced vertices

---

instructions search domain. BIOCODE uses genetic programming methods to learn a set of instructions generating biological networks that mimick given set of graph attributes as close as possible. BIOCODE encodes those network attributes within an individual BIOCODE program's fitness function. Recovering the formerly introduced growth models is not the main goal of BIOCODE learning process, instead we focus on learning programs which grow graphs that represent specific graph classes as measured by particular similarity metrics.

4.1 Constructing a fitness function

BIOCODE defines an attribute collection $x = [x_1, x_2, \ldots, x_m]$ as a m-long feature vector where each entry x_i can represent a single scalar value such as assortativity, or it can represent a vector of values such as multiple independent samples of the graph's effective radius during its evolution. Representing the fundamental and necessary graph attributes BIOCODE will match as part of growth model is the main objective of attribute collection step. Let $s_l(.,.)$ be a user-defined similarity metric between the collections l^{th} attributes. To calculate the similarity between any two attribute collections of the same dimension, we define a possibly weighted metric $s(x^i; x^j)$ as in:

$$s(x^i; x^j) = \sum_{t=1}^{m} w_l s_l(x_l^i, x_l^j) \quad (1)$$

where similarity measure $s_l(.,.)$ can simply be inverse of the difference between two attributes for single scalar values. Or, it can also represent a metric of the distribution similarities for nonscalar attributes. There are two conditions on $s(x^i; x^j)$: 1- $s(x^i; x^j)$ must get the maximum value when $x^j = x^i$, 2- $s(x^i; x^j)$ must be a monotonically non-decreasing function of the similarity between the two attributes. One can weight every attribute separately by the weights w_l in Eq. 1 which then causes optimization process to prefer some attributes more than the others. We use $w_l = 1$ for all l in our experiments.

The fitness of a BIOCODE program is defined by using Eq. 1. Let x^P be a random variable defining the attribute collection for the graph produced by non-deterministic program P, and let x^T be a target attribute collection. Then, our problem becomes searching for optimal P^* such that:

$$P^* = \underset{P}{\text{argmax}} \ \mathbb{E}[s(x^P, x^T)], \quad (2)$$

where the expectation is calculated over P's multiple non-deterministic executions. In this case, BIOCODE searches for

the optimal program P^* such that the attributes of the graph produced by P^* should be the most similar to the attributes of the graph given by x^T according to similarity measure $s(.,.)$. This optimization problem cannot be easily tackled since number of candidate programs in the search space is massive. BIOCODE handles this problem efficiently by using a genetic programming algorithm that is proven to be quite useful across difficult optimization problems.

4.2 Optimization with genetic algorithms

.

In BIOCODE, we apply the optimization procedures in genetic algorithm by using the ECJ package [31]. We utilize ECJ's capabilities for following reasons: 1- Parallel evaluation of individuals inside a generation, 2- Customization of the breeding and selection processes for more than one subpopulations, 3- Applying NSGA-II multi-objective optimization [32], and 4- Handling various representations for variable and fixed length genomes. Every candidate individual in the genetic program describes a program. We calculate the fitness of all individual candidates in the constant-size population at each generation. BIOCODE evaluates the fitness of each program by executing the program for k iterations, and then compares the program's attribute vector x^P with the target attribute vector. That evaluation process is rerun for M times, and mean of the calculated results is presented such that the program P's fitness is:

$$F(P) = \text{avg} \ s(x^P, x^T) \quad (3)$$

As an alternative, we may calculate the mean for each $s_l(x_l^i; x_l^j)$ in Eq. 1 as an independent objective, and apply a multi-objective optimization procedure such as NSGA-II [32].

As part of BIOCODE optimization process, we breed individual programs and the programs blend with each other by a two-point crossover operation. This crossover operation vary the programs length and content. As part of each generation's final step, individual programs contest in a tournament where two randomly selected programs are compared consecutively and the higher fitness value determines the winning programs. Tournament winners turn into individuals of the next population. BIOCODE draws individual populations with replacement, and so drawn individuals are copied in the next population. More fit programs have a higher chance to succeed in the tournaments, so such members of the genetic algorithm have a higher chance to survive into the subsequent generation.

5 APPLICATIONS TO REAL AND SYNTHETIC BIOLOGICAL NETWORKS

We evaluate the performance of our proposed framework BIOCODE in learning programs that generate graphs matching both real biological networks and synthetic networks predefined attributes. We consider the following BIOCODE parameters in our experiments unless otherwise noted. BIOCODE programs in the optimization process first generation start with randomly selected 10 operations. Every generation comprises 100 programs which are evaluated by a single-objective or a multi-objective fitness functions

as in sections 5.1 and 5.2 respectively. BIOCODE advances individual programs to next generations by tournament. At the beginning of every generation, two-point crossover is used to breed the individual population from the chosen individuals from earlier generation. This crossover mechanism creates novel individual programs which can be of different length. These individuals are then mutated with rate 0.1. After carrying out this optimization approach for 15 generations, the fittest program from the ultimate generation is chosen to be the resulting representative BIOCODE program. We compare other models against this representative program. We have implemented BIOCODE in Scala. BIOCODE and datasets used in this paper are available at https://github.com/seferlab/biocode.

5.1 Learning scale-free graphs

B-A model is motivated by generating graphs with scale-free node degree distribution which is a fundamental property of many real-word biological networks, and BIOCODE is able to learn growth models that generate scale-free distributions. Given the massive growth model space characterized by the operation set and corresponding parameters, it is not clear whether effective exploration of that search space can be achieved.

BIOCODE uses a shape function to calculate the degree distributions similarity. In this case, one can select the goodness of fit to a scale-free distribution as an attribute. However, this method cannot be generalized to distributions other than scale-free distributions. Generally, our goal is to produce graphs matching an arbitrary degree distribution's shape. We model the shape of arbitrary distribution by defining the shape ψ_{shape} as the cumulative distribution of vertex degrees where degrees of the vertices (support of the distribution) is scaled to range 0 and 1. We can compare the degree distribution of different size graphs after such scaling. Similarity measure for the shape attribute is defined as:

$$s_{shape}(\psi_{shape}^i, \psi_{shape}^j) = \frac{1}{\left\| \psi_{shape}^i - \psi_{shape}^j \right\|_1 + \epsilon} \quad (4)$$

where ϵ, as a tiny positive constant, ensures that fitness is well-defined when the compared shapes exactly match with each other. The single parameter of B-A model is i which is the number of existing nodes to which a newly introduced node attaches. We retrieve the target node degree distribution shape for $i = 3, 4, 5, 6$ by producing graphs for such i values and obtaining scale-free exponents maximum likelihood estimates of $\alpha = 2.6, 2.7, 2.8, 2.9$ for each i. Then, by utilizing s_{shape} in Eq. 3, degree distribution shape difference between the estimated target shape and the program generated graphs characterizes the program fitness.

By using this fitness function, BIOCODE can learn multiple different programs that generate scale-free graphs. One of the most effective BIOCODE programs that generate scale-free graph is shown in Algorithm 4. We test the possibility of a scale-free degree distribution by using statistical tests specifically designed for scale-free distribution as defined in [33]. Although BIOCODE has not explicitly used the α parameter in the fitness function, the mean α values of the

---

Algorithm 4 Instance of Learned Scale-Free Model

1: NEW_NODE
2: RANDOM_NODE
3: CONNECT_TO_INFLUENCED
4: CLEAR r2
5: SET(1)
6: RANDOM_EDGE
7: DISCONNECT_FROM_INFLUENCED
8: RANDOM_NODE
9: GENERATE_EDGE
10: INFLUENCE_NEIGHBOURS(0.692)

---

graphs generated from the learned models passing the scale-free test is 2.69, that is fairly similar to the target graphs α.

Indeed, we claim that discovering a BIOCODE program which generates graphs with a scale-free degree distribution is reasonably easy for the optimization process of BIOCODE. One of the optimization trace while trying to fit a degree-distribution generated from the B-A model with $i = 4$ is shown in Figure 2. According to Figure 2, BIOCODE discovers scale-free models in the first generation even before selection has started to make an impact on the population. Scale-free programs total fitness increases fast, and such total fitness is considerably higher than the total fitness of the remaining individuals without scale-free distribution by generation 6 as seen in Figure 2. Our observations mainly indicate the following: 1- Scale-free model discovery is not so challenging, 2- The possibility of scale-free distribution in the graph appears to correlate quite well with the shape function.

Figure 2: Total fitness for shape function (s_{shape}) at each generation. The total fitness for individuals failing the scale-free test is shown in blue, whereas the total fitness for programs passing the scale-free test is plotted in red. Even after 5th generation, scale-free individuals total fitness is almost two times as big as the non-scale-free programs.

5.2 Performance on a biological collaboration network

We test the performance of BIOCODE over a co-authorship network of genome-wide association studies (GWAS) [30].

Particularly, we focus on such robust biological collaboration network of "repeated co-authorship" where scientist pairs have an edge between them if these scientists have published together more than one time. This biological collaboration network has high assortativity value of 0.19 showing that highly-collaborating scientists have an edge with scientists that also collaborate profoundly. In this case, BIOCODE concurrently optimizes for assortativity and the shape distribution attributes by utilizing the multi objectivecprocedure discussed in section 4.2.

We evaluate the performance by comparing the graphs produced by BIOCODE program to the graphs produced via Kronecker model [9]. The Kronecker model recreates many real-world network attributes by its recursive and fast procedure. We estimate Kronecker model parameters by using the KronFit maximum likelihood method on the GWAS network. We compare real GWAS graph to the attributes of 100 graphs produced by each model. The learned BIOCODE program outperforms the best-fit Kronecker model in terms of better matching the degree distribution shape and the assortativity of the true graph as in Figure 3. The mean shape difference of the BIOCODE model is more similar to the co-authorship network's shape than the shape for Kronecker model. The average assortativity for Kronecker graphs is 0.165 whereas the average for BIOCODE graphs is 0.206. On the other hand, BIOCODE generated graphs have a broader range of assortativity scores (std. dev 0.0208) than the Kronecker graphs scores (std. dev 0.00629).

networks [7]. Our approach is similar the one described in Section 5.2, but DMC model is used for the baseline comparison instead of the Kronecker model. The best parameters for DMC model are identified as $q_{con} = 0.37$ and $q_{mod} = 0.55$ via a grid search over the parameter space. The parameters are chosen such that the graphs generated by BIOCODE model match the diameter, clustering coefficient, and match the number of edges of the input protein interaction graph as close as possible. According to Figure 4, the graphs produced by BIOCODE program is considerably more similar to the real PPI network in terms of target attributes than the ones generated by the DMC model. BIOCODE program generates graphs with mean average clustering 0.091 (standard deviation 0.006) matching the true average clustering coefficient of 0.099 of the interaction network quite accurately. In contrast, the graphs generated by DMC model have mean average clustering coefficient of 0.227 (standard deviation 0.013) that is truly far away from the original interaction network's value. The shape distribution results in Figure 4 show the similar output. The average shape distribution distance of the random graphs produced by the BIOCODE program is 4.58 (standard deviation 1.69), where the average distance between DMC graphs is 15.48 (standard deviation 6.29). In addition to generating graphs better matching the target network, BIOCODE program show less variance and higher stability of parameters in terms of those metrics.

Figure 3: GWAS biological collaboration target network. Each point in the plot shows an individual produced graph from a model. The x axis shows the assortativity difference between the target network and a graph. The y axis shows the shape difference between the target network and a graph. The green dot shows an exact match to the target network.

Figure 4: Protein protein interaction target network. Each point in the plot shows an individual produced graph from a model. The x axis shows the average clustering coefficient difference between the target network and a graph. The y axis shows the shape difference between the target network and a graph. The green dot shows an exact match to the target interaction network.

5.3 Performance on a protein interaction network

BIOCODE is capable of learning a program that produces graphs similar a recently compiled and high-quality yeast protein interaction network [34], [35]. BIOCODE optimizes for both clustering coefficient and shape distribution, which are biologically important in protein interaction

5.4 Performance on a gene regulatory network

In the previous sections, BIOCODE has outperformed the compared growth model while optimizing models simultaneously for two network attributes. Across different growth models, the particular network attributes were selected for optimization since such attributes had already been studied in the corresponding network class. However, BIOCODE

optimization is not limited to 2 target attributes. Here, we discuss the possibility of extending the learning process to more than two attributes. We learn a BIOCODE program over gene regulatory (GR) network discussed in [9], [36] by simultaneously optimizing for all 3 attributes such as average clustering coefficient, assortativity, and shape. We evolve 150 programs for 25 generations over the gene regulatory network. We compare the graphs produced by the optimized BIOCODE program to the ones produced by the Kronecker model as in Section 5.2.

Figure 5 includes 3 plots which show the closeness of BIOCODE produced graphs to the gene regulatory network for all attribute pairs. In reality, graphs generated by BIOCODE outperforms the graphs produced by Kronecker model in terms of better matching the gene regulatory graph. A single optimized BIOCODE program was optimized with respect to all 3 attributes at a single time even though the plots in the figure display 2 dimensions at once. Graphs generated by BIOCODE program have low variance with respect to the target network topological properties, similar to the protein interaction network in Section 5.3.

5.5 BIOCODE generates random models

The graphs generated by BIOCODE program has higher diversity than the ones generated by human-designed models. We use spectral distance between 100 graphs produced by both the B-A model and the BIOCODE program to evaluate the diversity of graphs. The spectral distance is a plausible graph similarity metric correlating highly with the graph edit distance [37]. We calculated the spectral distances between graphs by using discretized histogram of the normalized Laplacian eigenvalue distribution over 100 bins. In this case, the spectral distance becomes Euclidean distance between such histograms.

BIOCODE generates nondeterministic models as seen in Table 2. Ensemble of graphs produced by BIOCODE models have higher diversity than the ensemble of graphs generated by the B-A model while matching the target attributes better. We observe similarly higher diversity when we repeat this experiment comparing the the graphs generated byBIOCODE with the graphs produced by DMC over yeast PPI network [34].

6 CONCLUSIONS AND FUTURE WORK

We come up with BIOCODE framework to represent network growth dynamics as programs consisting of basic and expressive list of operations. Such programs are general enough to closely approximate diverse set of biological graph growth models. Besides, models with the desired attributes can be searched effectively by combining efficient encoding of BIOCODE with an efficient genetic algorithm. Across gene regulatory, protein interaction, and biological collaboration networks, the proposed optimization process is reasonably fast: It takes less than 30 minutes for 2 objectives and less than 4 hours for 3 objectives. In this setting, this optimization process can generate BIOCODE programs which compete strongly with the carefully-designed hand-coded network growth models. Such hand-coded models are mainly introduced to match graph attributes in related domains.

Figure 5: Gene regulatory target network. Each point in the plot shows an individual graph produced from a model. The difference between the coefficients of the gene regulatory network and a graph produced from the model are shown for all 3 network attribute pairs. The green dot defines the gene regulatory graph and shows the origin.

Graphs with scale-free degree distribution can be generated by BIOCODE for a number of attachment parameters i such that these programs pass stringent statistical tests to verify scale-free property. In truth, BIOCODE learning procedure discover scale-free programs fast, and in the end, produces numerous different programs which generate graphs passing the scale-freeness verification test. Moreover, BIOCODE generates graphs that are more varied than the graphs produced by the B-A model. Overall, these results

Table 2: Mean and standard deviation ($\mu \pm \sigma$) of spectral distance between all graphs produced by BIOCODE programs and by the B-A model.

i	3	4	5	6
B-A	0.0096 ± 0.0068	0.0044 ± 0.0017	0.0039 ± 0.0016	0.0036 ± 0.0015
BIOCODE	0.0151 ± 0.0125	0.0262 ± 0.0216	0.0298 ± 0.0238	0.0192 ± 0.0162

show the ubiquitousness of scale-free degree distribution feature.

The introduced BIOCODE framework generates unattributed graphs that can be both directed and undirected, but it can be improved to generate graphs with edge and node attributes as well. A number of enhancements are possible via expanding the instruction set. As an example, one can add instructuons to diffuse node attributes from one part of graph to remaining parts. More complicated influence procedure can be incorporated to BIOCODE as well. BIOCODE machine needs to be enhanced to support these instructions. For instance, BIOCODE needs to add an edge memory similar to the existing label memory to handle edge attributes. These enhacements are not so challenging, despite it is critical to design them carefully.

Lastly, even though individual instructions as part of BIOCODE programs can be interpreted quite easily, the growth dynamics of programs generated by the learning process may not be so clear to a certain extent. As a future work, we plan to focus on analyzing ensembles of optimized programs to identify understandable growth mechanisms via identifying generally appearing instruction motifs. For instance, by analyzing the programs similar to Algorithm 4 in detail, we have identified several repeated instruction patterns which can generate edges to current vertices proportional to vertex degrees and can mimick scale-free graphs. Influence instructions, GENERATE_edge, and NEW_NODE are commony observed in these patterns. Finding instruction sets that are understandable as a unit can be achieved by mining BIOCODE programs for repeating instruction motifs.

REFERENCES

[1] A. L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, oct 1999.

[2] R. Rak and E. Rak, "The fractional preferential attachment scale-free network model," *Entropy*, vol. 22, no. 5, 2020. [Online]. Available: https://www.mdpi.com/1099-4300/22/5/509

[3] A. Bhan, D. J. Galas, and T. G. Dewey, "A duplication growth model of gene expression networks," *Bioinformatics*, vol. 18, no. 11, pp. 1486–1493, nov 2002. [Online]. Available: http://www.kgi.edu/html/noncore/faculty/dewey/bioinf.pdf

[4] I. Ispolatov, P. L. Krapivsky, and A. Yuryev, "Duplication-divergence model of protein interaction network," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 71, no. 6, p. 061911, jun 2005. [Online]. Available: https://journals.aps.org/pre/abstract/10.1103/PhysRevE.71.061911

[5] R. Sole, R. Pastor-Satorras, E. Smith, and T. KEPLER, "A model of large-scale proteome evolution," *Advances in Complex Systems (ACS)*, vol. 05, pp. 43–54, 02 2002.

[6] S. A. Teichmann and M. M. Babu, "Gene regulatory network growth by duplication," *Nature Genetics*, vol. 36, no. 5, pp. 492–496, may 2004. [Online]. Available: http://www.nature.com/naturegenetics

[7] A. Vázquez, A. Flammini, A. Maritan, and A. Vespignani, "Modeling of protein interaction networks," *Complexus*, vol. 1, no. 1, pp. 38–44, 2003. [Online]. Available: https://www.karger.com/DOI/10.1159/000067642

[8] A. Jasra, A. Persing, A. Beskos, K. Heine, and M. De Iorio, "Bayesian inference for duplication–mutation with complementarity network models," *Journal of Computational Biology*, vol. 22, no. 11, pp. 1025–1033, 2015, pMID: 26355682. [Online]. Available: https://doi.org/10.1089/cmb.2015.0072

[9] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *The Journal of Machine Learning Research*, vol. 11, pp. 985–1042, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756039

[10] C. Seshadhri, A. Pinar, and T. G. Kolda, "An in-depth analysis of stochastic kronecker graphs," *J. ACM*, vol. 60, no. 2, May 2013. [Online]. Available: https://doi.org/10.1145/2450142.2450149

[11] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: Densification laws, shrinking diameters and possible explanations," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, New York, USA: ACM Press, 2005, pp. 177–187. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1081870.1081893

[12] G. G. Piva, F. L. Ribeiro, and A. S. Mata, "Networks with growth and preferential attachment: modelling and applications," *Journal of Complex Networks*, vol. 9, no. 1, 04 2021, cnab008. [Online]. Available: https://doi.org/10.1093/comnet/cnab008

[13] M. Falkenberg, J.-H. Lee, S.-i. Amano, K.-i. Ogawa, K. Yano, Y. Miyake, T. S. Evans, and K. Christensen, "Identifying time dependence in network growth," *Phys. Rev. Research*, vol. 2, p. 023352, Jun 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.2.023352

[14] D. S. Callaway, J. E. Hopcroft, J. M. Kleinberg, M. E. J. Newman, and S. H. Strogatz, "Are randomly grown graphs really random?" *Physical Review E*, vol. 64, no. 4, p. 041902, sep 2001. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.64.041902

[15] S. N. Dorogovtsev, J. F. F. Mendes, and A. N. Samukhin, "Structure of Growing Networks with Preferential Linking," *Physical Review Letters*, vol. 85, no. 21, pp. 4633–4636, nov 2000. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.85.4633

[16] W. K. Kim and E. M. Marcotte, "Age-Dependent Evolution of the Yeast Protein Interaction Network Suggests a Limited Role of Gene Duplication and Divergence," *PLoS Computational Biology*, vol. 4, no. 11, p. e1000232, nov 2008. [Online]. Available: https://dx.plos.org/10.1371/journal.pcbi.1000232

[17] R. Kumar, J. Novak, and A. Tomkins, "Structure and evolution of online social networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 2006. New York, New York, USA: Association for Computing Machinery, 2006, pp. 611–617. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1150402.1150476

[18] N. Przulj, O. Kuchaiev, A. Stevanović, and W. Hayes, "Geometric evolutionary dynamics of protein interaction networks," *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pp. 178–89, 01 2010.

[19] L. Huang, L. Liao, and C. H. Wu, "Evolutionary analysis and interaction prediction for protein-protein interaction network in geometric space," *PLOS ONE*, vol. 12, no. 9, pp. 1–19, 09 2017. [Online]. Available: https://doi.org/10.1371/journal.pone.0183495

[20] S. Navlakha and C. Kingsford, "Network archaeology: Uncovering ancient networks from present-day interactions," *PLoS Computational Biology*, vol. 7, no. 4, p. 1001119, apr 2011. [Online]. Available: www.nih.gov

[21] P. Erdos and A. Renyi, "On the evolution of random graphs," *Publ. Math. Inst. Hungary. Acad. Sci.*, vol. 5, pp. 17–61, 1960.

[22] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world9 networks," *Nature*, vol. 393, no. 6684, pp. 440–442, jun 1998. [Online]. Available: https://www.nature.com/articles/30918

[23] K. Voordeckers, K. Pougach, and K. J. Verstrepen, "How do regulatory networks evolve and expand throughout evolution?" *Current Opinion in Biotechnology*, vol. 34, pp. 180–188, 2015, systems biology • Nanobiotechnology. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0958166915000208

[24] L. Akoglu and C. Faloutsos, "RTG: A recursive realistic graph generator using random typing," *Data Mining and Knowledge Discovery*, vol. 19, no. 2, pp. 194–209, oct 2009.

[25] G. Palla, L. Lovász, and T. Vicsek, "Multifractal network generator," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 107, no. 17, pp. 7640–7645, apr 2010. [Online]. Available: www.pnas.org/cgi/doi/10.1073/pnas.0912983107

[26] Y. Sun, Y. Yu, and J. Han, "Ranking-based clustering of heterogeneous information networks with star network schema," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, New York, USA: ACM Press, 2009, pp. 797–805. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1557019.1557107

[27] M. Kim and J. Leskovec, "Multiplicative Attribute Graph Model of Real-World Networks," *Internet Mathematics*, vol. 8, no. 1-2, pp. 113–160, 2012. [Online]. Available: https://projecteuclid.org/euclid.im/1339678185

[28] M. Middendorf, E. Ziv, and C. H. Wiggins, "Inferring network mechanisms: The Drosophila melanogaster protein interaction network," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 9, pp. 3192–3197, mar 2005. [Online]. Available: www.pnas.orgcgidoi10.1073pnas.0409515102

[29] V. Janjić, R. Sharan, and N. Pržulj, "Modelling the yeast interactome," *Scientific Reports*, vol. 4, no. 1, pp. 1–8, mar 2014. [Online]. Available: www.nature.com/scientificreports

[30] B. K. Bulik-Sullivan and P. F. Sullivan, "The authorship network of genome-wide association studies," p. 113, feb 2012. [Online]. Available: http://gephi.org/

[31] E. O. Scott and S. Luke, "ECJ at 20: Toward a general metaheuristics toolkit," in *GECCO 2019 Companion - Proceedings of the 2019 Genetic and Evolutionary Computation Conference Companion*. New York, NY, USA: Association for Computing Machinery, Inc, jul 2019, pp. 1391–1398. [Online]. Available: https://dl.acm.org/doi/10.1145/3319619.3326865

[32] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, apr 2002.

[33] A. Clauset, C. R. Shalizi, and M. E. Newman, "Power-law distributions in empirical data," pp. 661–703, nov 2009.

[34] R. Oughtred, J. Rust, C. Chang, B.-J. Breitkreutz, C. Stark, A. Willems, L. Boucher, G. Leung, N. Kolas, F. Zhang, S. Dolma, J. Coulombe-Huntington, A. Chatr-aryamontri, K. Dolinski, and M. Tyers, "The biogrid database: A comprehensive biomedical resource of curated protein, genetic, and chemical interactions," *Protein Science*, vol. 30, no. 1, pp. 187–200, 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/pro.3978

[35] T. A. Gibson and D. S. Goldberg, "Improving evolutionary models of protein interaction networks," *Bioinformatics*, vol. 27, no. 3, pp. 376–382, feb 2011. [Online]. Available: https://academic.oup.com/bioinformatics/article/27/3/376/319112

[36] T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. T. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E. G. Jennings, H. L. Murray, D. B. Gordon, B. Ren, J. J. Wyrick, J.-B. Tagne, T. L. Volkert, E. Fraenkel, D. K. Gifford, and R. A. Young, "Transcriptional regulatory networks in saccharomyces cerevisiae," *Science*, vol. 298, no. 5594, pp. 799–804, 2002. [Online]. Available: https://science.sciencemag.org/content/298/5594/799

[37] R. C. Wilson and P. Zhu, "A study of graph spectra for comparing graphs and trees," *Pattern Recognition*, vol. 41, no. 9, pp. 2833–2841, sep 2008.

Emre Sefer obtained his B.Eng from Bogazici University, Department of Computer Engineering in 2008, M.S. in Computer Science from University of Maryland College Park in 2011, and Ph.D. in Computational Biology from Carnegie Mellon University in 2015. After completing his Ph.D. Dr. Sefer had a brief a post-doc at CMU Machine Learning Department with Ziv-Bar Joseph. He is currently an assistant professor in Computer Science Department, Ozyegin University. His academic research has focused on Bioinformatics, and Machine Learning applications on social and economic networks. He has published in number of journals and conferences during his PhD, receiving best research paper award at Recomb 2016 conference.