Automatic Generation of Graph Models for Complex Networks by Genetic Programming

Alexander Bailey
Brock University
St. Catharines, Canada
ab04bf@brocku.ca

Mario Ventresca
University of Toronto
Toronto, Canada
mario.ventresca@utoronto.ca

Beatrice Ombuki-Berman
Brock University
St. Catharines, Canada
bombuki@brocku.ca

ABSTRACT

Complex networks have attracted a large amount of research attention, especially over the past decade, due to their prevalence and importance in our daily lives. Numerous human-designed models have been proposed that aim to capture and model different network structures, for the purpose of improving our understanding the real-life phenomena and its dynamics in different situations. Groundbreaking work in genetics, medicine, epidemiology, neuroscience, telecommunications, social science and drug discovery, to name some examples, have directly resulted. Because the graph models are human made (a very time consuming process) using a small subset of example graphs, they often exhibit inaccuracies when used to model similar structures. This paper represents the first exploration into the use of genetic programming for automating the discovery and algorithm design of graph models, representing a totally new approach with great interdisciplinary application potential. We present exciting initial results that show the potential of GP to replicate existing complex network algorithms.

1. INTRODUCTION

A network can generally be considered as any set of items which are interconnected by physical or conceptual physical links. A *graph* is the natural mathematical abstraction of this idea. A *complex network*[17] is a network where the structure or pattern of the links impart some meaning, it is important to note that the term *complex* as it is used here does not necessarily refer to the size of the graph or the number of links. Complex networks arise naturally, for instance in the form of patterns of contact between people, creating social networks[22][21]. The network formed between academic papers which cite each other[10] and the linkage structure of the world wide web are both examples of information networks[15]. Power distribution systems[3] and airline routes[3], as well as the physical Internet are all examples of technological networks[3][18][2]. While complex networks are all around us, they are still not well-understood and the field of complex networks is still rapidly growing.

Building graphs to visualize and understand information has been a long-standing practice in many fields including the social sciences, biology, chemistry, physics, computer science, and mathematics. Understanding the patterns of connections that arise between components in these networks furthers our understanding of the interactions and behaviours of the components, as well as the behaviour of any processes acting on the network itself[18]. A *graph model* is an algorithm that describes the manner in which vertices are connected (Figure 1 illustrates this process).

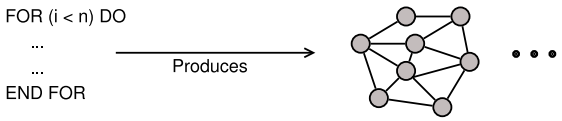


Figure 1: A *graph model* produces a set of graphs

By building graph models that describe the pattern of connections within networks, many of their properties can be precisely measured and accurate estimations of the behaviours of processes acting upon them can be made[18]. For example, if we could understand the patterns of connections that arise in epidemiological networks then the spread of disease within those networks can be better predicted; the number of infected individuals at a given time step could perhaps be estimated, or effective vaccination practices could be devised. Accurate graph models could contribute significant advancements in many fields given that there is a method for producing them, a task that traditionally has been done by hand – a difficult and time consuming process[17][18].

An automated approach to the construction of graph models would significantly reduce the amount of human effort required, and could potentially out-perform human attempts at constructing graph models, especially when the network at hand has a large number of edges or vertices. Humans are only capable of manually processing a relatively small amount of data, so human-generated models are created using a small subset of real-world examples. These models are not necessarily accurate for other graphs, although evidence can be given as to their general structure. Automated models have the potential for being robust to any given real-world data.

The automatic construction of graph models is not without its caveats. The most obvious being the choice of metrics used to determine how well the constructed graph model fits the target network. The process that created the target network is not known *a priori*, so the problem is one

of graph comparison. However, this is not a graph isomorphism problem because the goal is not to produce the *same* graph but rather an algorithm that produces a set of graphs with similar properties as the given graph. The structure of two graphs can be compared without checking for isomorphism, but if the selection of comparison metric is not a careful one then a seemingly good match could be a poor one in reality. Another very important consideration is that the discovered model must not only be able to replicate the structural properties of the given example graphs, but also be able to replicate the manner in which it grows over time. Overfitting to the given graph is a strong possibility.

This paper proposes the first attempt at the automatic generation of graph models for complex networks. The following sections examine the feasibility of Koza-style Genetic Programming (GP) for discovering graph models from known undirected, unweighted, graphs of less than 500 vertices. All target graphs have been constructed using known graph model algorithms, allowing for an algorithmic comparison. Section 2 gives a brief overview of the relevant graph models, a description of some real-world networks, and provides some necessary definitions. Section 3 describes the GP system used, the GP language, as well as the fitness function and evaluation process. Finally, section 4 describes the experimental setup and results.

2. BACKGROUND

There are a number of existing graph models that have been proposed to model specific properties observed in real-world networks. Some have gained a great deal of attention in the literature, and perhaps the three most groundbreaking works have been the *random graph model* by Erdös and Rényi[11], the Watts and Strogatz *small-world model*[23], and the Barabási and Albert model[5]. The Erdös and Rényi model being the first foray into the construction of graph models and, as such, is the most well-studied. This model is important because it shows that understanding a model means that properties of the graphs produced by it can be computed exactly, or estimated accurately. Understanding a model and how it generates a given network requires an understanding of some basic properties.

2.1 Measurable Properties of Networks

Estimations about the similarity of graphs constructed with a graph model to real-world networks can be made by measuring various properties within the graphs and comparing them to a measure of the same properties in the real-world network. A large number of property measures have been proposed, but some commonly used for comparing network structure include the *average geodesic path length*, the *diameter* of the network, the *average vertex degree*, the *clustering coefficient*, sometimes called *transitivity*, and the *degree distribution*[4][17].

The **degree**, k, of a vertex is the number of edges that are attached to it. The **average vertex degree**, $\langle k \rangle$, is the average of all vertex degrees in the network.

$$\langle k \rangle = \frac{1}{n} \sum_{i=1}^{n} k_i, \tag{1}$$

where n is the number of vertices in the network, and k_i is the degree of vertex i.

A **geodesic path** is defined as the minimum distance between between two vertices and the **average geodesic path length** is defined as the average of all geodesic paths in the network. Let l be the average geodesic path length, n be the number of vertices, and i and j be vertex pairs[17].

$$l = \frac{2}{n(n+1)} \sum_{i \geq j} d_{ij}, \tag{2}$$

The **diameter** of a network is the largest geodesic path between any vertex pair. A **component** is a subgraph $G' \subseteq G$ where there exists a path between any pair of vertices $v_1, v_2 \in G'$, and there does not exist a $v_3 \in G \setminus G'$ such that there exists a path between v_3 to any $v \in G'$.

The **global clustering coefficient** measures how connected a vertex's neighbours are to each other, *i.e.* it is a measure of the number of triangles in the graph[25][17].

$$C = \frac{number\ of\ triangles \times 3}{number\ of\ connected\ triples} \tag{3}$$

A "connected" triple in this context means a set of three vertices a, b, c where a is connected to b and c is connected to b but a and b may or may not be connected[17]. For local as well as alternate definitions of the global clustering coefficient, see Watts and Strogatz [23], Newman[17] and Zager[25]. The clustering coefficient is frequently a good indicator as to which graph model fits a given network or to check if a network may just contain random connections[2, 18, 17]. The global clustering coefficient can be computed in $\mathcal{O}(|V|\langle k \rangle^2)$ where $|V|$ is the cardinality of the vertex set, and $\langle k \rangle$ is the average vertex degree.

The **degree distribution** refers to the distribution that describes the probability of choosing a random node with a given degree within a network[17]. The degree distribution can reveal quite a bit about a graph's structure, but alone it does not give all the structural information. Consider the two graphs in Figure 2; the graph on the left and the graph on the right have the same degree distribution but are not equivalent.

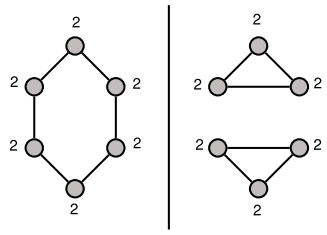


Figure 2: Two non-equivalent graphs with the same degree distribution

Despite this caveat, degree distributions can still give quite a bit of information. In many cases it is enough information to identify graphs produced by one model versus another model, or enough information to discern which model may be most suited to a particular network[17, 18]. The *Kolmogrov-Smirnov test* (KS-test) is a good candidate for comparing degree distributions because the only restriction is that the distributions be continuous. The test statistic, $D_{n,n'}$, for a two-sample KS-test is defined as[9]: $D_{n,n'} = sup_{\chi}|F_{1,n}(\chi) - F_{2,n'}(\chi)|$, where $F_{1,n}$ is the empirical distribution function of the first sample, and $F_{2,n'}$ is the empirical distribution function of the second sample. Minimizing $D_{n,n'}$ minimizes the difference between the distributions.

2.2 Graph Models of Complex Networks

Intuitively, it is easy to surmise that real-world networks have a structure that is not random. Generally speaking, the structure and pattern of links between nodes is likely based on the function of the network itself[1, 24, 12, 23]. Observationally, networks dealing with people have global clustering coefficients which tend to be larger still[19, 6]. The appropriate measure can actually reveal a relationship between a particular network and a particular model[4]. While real-world networks provide an interesting and practical application for complex network research, this paper will concentrate on known models – the goal of this study is to demonstrate the efficacy of a GP approach to automatic model generation and this is most evident when the true graph model is known.

2.2.1 The Erdös and Rényi Model

The Erdös and Rényi model builds a graph by taking some number of vertices n and connecting each vertex pair with some probability p, producing a graph denoted $G_{n,p}$. The model represents the *ensemble* of all $G_{n,p}$ graphs in which a graph having m edges appears with probability $p^m(1-p)^{(M-m)}$, where $M = \frac{1}{2}n(n-1)$ is the maximum number of edges[18]. In the limit of large n, keeping the mean degree $z = p(n-1)$ constant, the probability, p_k, of a vertex having degree k is

$$p_k = \left(\begin{array}{c} n \\ k \end{array} \right) p^k (1-p)^{n-k} \simeq \frac{z^k e^{-z}}{k!}, \tag{4}$$

with the approximation becoming exact in the limit of large n and fixed k. The degree distribution for the random graph is a Poison distribution[18].

If it was known that a network was constructed using the Erdös and Rényi model, and how that model was configured, then very accurate estimations could be made about the nature of diffusive processes acting on the network or transmissions within the network. Estimations about the effect of removing nodes or edges could also be computed. These estimations, made based on the model and not the network itself, would not only apply to the current network configuration but scale with the network if it were to grow or shrink. This illustrates the motivation behind constructing models of complex networks.

2.2.2 The Small-world Model

The *small-world* model, proposed by Watts and Strogatz[23], focuses on modelling two important aspects of many real-world networks – the diameter of the network remains small despite a large number of vertices within it, and if two vertices share the same neighbour then it is very likely that they are themselves neighbours. The Watts and Strogatz *small-world* model is built by creating a ring of L vertices in which each vertex is connected to each of its neighbours up to k spaces away from itself. Then each edge is considered in turn and with probability p one of its ends is "rewired" to a new vertex chosen uniformly at random. This results in a high connectivity among a vertex's neighbouring vertices as well as a low upper bound on geodesic path length. While the Watts-Strogatz model produces graphs with transitivity values and path lengths which are similar to some real-world networks, it fails to produce graphs with realistic degree distributions. It can be shown that the degree distribution of small-world graphs corresponds to[17]:

$$p_k = e^{-cp} \frac{(cp)^{k-c}}{(k-c)!}, \tag{5}$$

where p_k is the probability of a vertex having degree k, c is the initial degree of each vertex before the rewiring process, p is the probability of rewiring. Depending on the application the unrealistic degree distribution may or may not be a problem, the model was never intended to produce graphs with distributions that matched those found in real-world networks, only the transitivity and path length were considered. Models such as the Barabasi-Albert model attempt to address the issue of degree distribution.

2.2.3 The Barabási-Albert Model

The Barabási-Albert[5] was based on the observation that many real-world networks, such as the World Wide Web (WWW) exhibit a degree distribution which follows a power-law[18]. In such networks, the probability $P_k =\sim k^{-\alpha}$, where α is some constant, is the probability that a vertex chosen uniformly at random has degree k. Price had observed a similar degree distribution in citation networks which are directed and acyclic and had constructed his own earlier model to which the Barabási-Albert model bears some resemblance[20][17].

1. *Growth:* Starting with (m_0) vertices, at each time step a vertex with, $m \leq m_0$ edges, is added and attached to m different vertices already present.
2. *Preferential attachment:* The probability that a new vertex will be connected to vertex i depends on the degree k_i of vertex i, such that

$$\Pi(k_i) = k_i \left[\sum_j k_j \right]^{-1}. \tag{6}$$

It can be shown that after t time steps the scale-free network will have $N = t + m_0$ vertices and mt edges.

3. GP SYSTEM

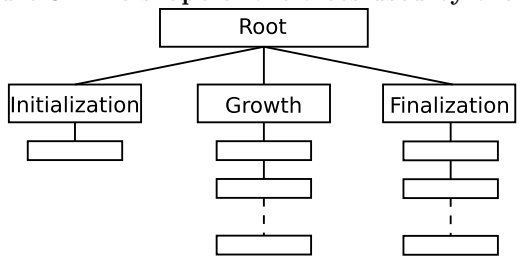
Results were generated using RobGP[13, 14], a Koza-style Genetic Programming (GP) system, subtree crossover and grow-style mutation were used as genetic operators. The initial population was generated using the grow method[16]. All operations were strongly typed. Each GP-evolved model was executed three times each to produce three graphs per individual, features in the produced graphs were then compared to features in a target network in order to assign a fitness value to each evolved model in the population.

3.1 GP Language

In order to produce useful trees, strong typing was used to enforce a particular tree shape, which was evaluated in the following manner. The root node has three branches, each containing a list of actions. One branch for initialization, one branch which defines growth actions, and one branch which describes finalization actions. Operations in the *initialization* branch were responsible for adding vertices or specifying how vertices were to be added during the graph building process. *Growth* operations were responsible

for adding edges to the graph, and was executed n times per evaluation, where n is the desired number of vertices in the generated network. *Finalization* operations are any operations which needed to assume edges and vertices were already present in the graph, edge removal, for example. Figure 3 illustrates the tree structure.

Figure 3: The shape of the trees used by the GP



While there has been no previous research applying GP to the problem of automatically generating graph models for complex networks, there are quite a few proposed models that have been constructed manually[11, 5, 23]; Newman provides a good overview[17]. The GP language was designed with various existing graph models in mind. In addition to a basic set of math operators, $\{+, -, *, \%\}$, Ephemeral random constants (ERC), IF structures, and the relational $<$ operator, the language includes:

- Initialization actions:
  - `ADD_ALL_NODES`: Initialize all nodes.
  - `BUILD_RING`: Add all nodes and build a ring.
  - `SET_GROW_NODES`: Specify nodes are added at each iteration (grow).
- Growth actions:
  - `CREATE_TRIANGLE`: Creates a triangle which includes the current node in the active graph.
  - `CONNECT_W_PROB(p)`: Connect to each node with a specified probability, p.
  - `CONNECT_RAND`: Connects the current node in the active graph to some random node in the active graph.
  - `CONNECT_STUB(p)`: Connects an edge from the current node to a node that has previously executed `CONNECT_STUB(p)`. If no suitable node has made an *edge request*, then an *edge request* is made. Requests are stored in a binary heap structure, heapified by the degree of the vertex which made them. Requests are satisfied probabilistically according to priority. A request to satisfy is selected randomly from the range $p|R|$ starting from the top of the heap, where $|R|$ is the cardinality of the set of edge requests. Once all edge requests made by a node are satisfied its entry is removed from the priority queue of edge requests. A vertex can make more than one request and a vertex cannot satisfy its own request. Each call to `CONNECT_STUB` can only make or satisfy a single request.
- Finalization actions:
  - `REWIRE_EQUAL_PROB(p)`: Rewire all edges with probability p.
  - `REWIRE_RANDOM`: Randomly rewire edges.
  - `REMOVE_PROB(p)`: Remove each edge with probability p.
- Terminals:
  - `CURRENT_NODE_DEGREE`: Returns the degree of the current vertex in the active graph.
  - `AVG_DEGREE`: Returns the average vertex degree in the active graph.
  - `MAX_DEGREE`: Returns the max vertex degree in the active graph.
  - `BETWEENESS_CENTRALITY`: Gives the betweenness centrality of the current vertex in the active graph[17].
  - `CLOSENESS_CENTRALITY`: Gives the closeness centrality of the current vertex in the active graph[17].
  - `TOTAL_VERTEX_COUNT`: Returns the vertex count of the active graph.
  - `TOTAL_EDGE_COUNT`: Returns the edge count of the active graph.

Probabilities are automatically selected from a pre-generated list, P of probabilities where,

$$P = \{0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}.$$

All functions that take probabilities as an argument take a floating point number, a, as a parameter. The value a is mapped to an index, $i = a \bmod |P|$. The probability $p = P_i$ is then passed to the function.

3.2 Fitness Function and Evaluation

The objective of the GP is to evolve graph models that produce graphs having similar characteristics to some target network. Because the model which produced the target network is unknown to the GP system, the target network itself must be compared to the graphs produced by the evolved models, called the *active models*. This gives an indirect measure of the fitness of the model. Comparing graphs for isomorphism is known to lie somewhere between the P and NP complexity classes[8]. However, this is *not* the goal in this work. Reasonable graph models have been benchmarked in the past by comparing how well they reproduce a particular *feature* of a set of complex networks[5, 23, 17]. The goal is not to reproduce a graph identical to the target network, rather to determine a graph generation algorithm able to produce graphs that are similar in some way or set of ways. The GP used in this work used a multi-objective, weighted and normalized, *summed-ranks* strategy[7]. The raw fitness objectives are defined as:

$$\begin{aligned} F_1Raw &= |l(tgt) - \tfrac{1}{3}\textstyle\sum_{i=1}^{3} l(actv)_i|, \\ F_2Raw &= [C(tgt) - \tfrac{1}{3}\textstyle\sum_{i=1}^{3} C(actv)_i]^2, \\ F_3Raw &= \tfrac{1}{3}\textstyle\sum_{i=1}^{3} D_{tgt,actv_i} \end{aligned} \tag{7}$$

The raw fitnesses are used during evolution in order to compute the ranks of each model. For presentation, they are

adjusted to values in the range $[0, 1]$ with a value of 1 being the most desirable. These adjusted fitnesses are computed as follows:

$$\begin{aligned} F_1 &= [1 + (\tfrac{F_1 Raw}{n})]^{-1}, \\ F_2 &= [1 + |C(tgt) - \tfrac{1}{3}\textstyle\sum_{i=1}^{3} C(actv)_i|]^{-1}, \\ F_3 &= 1 - \tfrac{F_3}{n} \end{aligned} \tag{8}$$

where i corresponds to one of each of 3 active graphs produced by each model, $l(target)$ is the average geodesic path length of the target graph, $l(actv)$ is the average geodesic path length of an active graph, $C(tgt)$ and $C(actv)$ are global clustering coefficients of the target and an active graph respectively, and $D_{tgt,actv_i}$ is the KS-test statistic comparing the degree distribution of the target graph to an active graph.

The objective weights were empirically determined, and set at $\frac{1}{4}$, $\frac{1}{4}$, $\frac{1}{2}$, for F_1Raw, F_2Raw, and F_3Raw, respectively. It was found via preliminary experimentation that the KS-test statistic was much harder to minimize than the average path length or clustering coefficient. Objective weighting was the simplest way to keep the GP from trading off small gains in the differences in average path length and clustering coefficient at the expense of the fit of the degree distribution. The decision for using a squared difference versus an absolute difference was established empirically for F_1Raw and F_2Raw.

The degree distribution yields a large amount of information regarding the structure of a graph and the KS-test statistic was used to incorporate some of that information[17]. The average geodesic path length and the clustering coefficient have also been shown to be extremely useful measures for estimating graph structure[4].

Before the fitness values are computed for any graph produced by an evolved model, the generated graph was first simplified to remove any self-loops or multi-edges. When calculating the average geodesic path length, if there was no path between a vertex pair the length of the path was returned as the size of the vertex set, a value larger than any possible path. If the graph was empty, the worst possible fitness values were assigned.

4. EXPERIMENTATION AND RESULTS

The objective of the experiments conducted was to see if the GP system could automatically reproduce reasonable approximations of the Erdös-Rényi model, the Barabási-Albert model, and the Small-world model given a sample graph produced by each algorithm as a target. All experiments used the parameters described in Table 1. During preliminary experimentation a population size of fifty was found to produce results similar to larger populations. Erdös-Rényi target graphs were produced with an edge probability of, $p = 0.05$. Barabási-Albert target graphs were always created using linear preferential attachment, a power of $\alpha = 1$, and by adding one vertex per iteration. Small-world target graphs were always created using 6 neighbours, 3 on each side, and a rewiring probability of 0.2.

All GP trees presented here were collected and converted into a more readable pseudocode. The functions described in Section 3.1 represent the GP language being used.

In the first experiment the GP system was initially used to evolve three populations of algorithms meant to approximate the Erdös-Rényi model. Data was collected using 200, 300, and 400 vertex target graphs, respectively. The second experiment was meant to evolve individuals which approximated the Barabási-Albert model, a 200 vertex graph generated by the Barabási-Albert model was used as a target. The third experiment evolves individuals to approximate the Small-world model, a 200 vertex graph was used as a target. The best individuals from each experiment were collected by comparing their fitness values, and by inspection of the average degree distribution plot vs the degree distribution plot of the target models. The best of these evolved algorithms were called ER_{200}, ER_{300}, ER_{400}, SW_{200}, and BA_{200}. The first two letters of their names indicate the type of target model which was used to evolve the individual, and the numeric part indicates the size of that target model.

The fourth and final experiment, was conducted in order to examine how well the evolved models predicted the growth of their target graphs. Once the individuals are collected, they are each used to generate thirty graphs at sizes of 200, 400, 600, 800, and 1000 vertices. Graphs of the same number of vertices were then generated by the Erdös-Rényi, Barabási-Albert, and Small-world models. The graphs produced by ER_{200}, ER_{300}, ER_{400}, SW_{200}, and BA_{200} were compared to graphs of the same size produced by the model they were to approximate. The fitness functions were applied in order to compare the graphs produced by the evolved models to the graphs produced by the original algorithms. The average fitness values, μ, and the standard deviations, σ, were recorded. Being able to predict future trends with the model is enormously important and informative, and it is a major motivation for this work. That is, producing only graphs of the same size as the target is of very limited use and we show here the ability of the evolved graphs to capture the growth dynamics of the given algorithms.

Table 1: GP parameters.

Parameter	**Value**
Generations	70
Population Size	50
Initialization Method	Grow
Grow Min	3
Grow Max	5
Selection	Tournament, k=3
Graphs per evaluation	3
Crossover	Subtree Crossover, 0.90
Mutation	Grow, 0.1, linearly decreasing min depth = 1, max depth = 4
Runs	30

Algorithm 1 ER_{200} algorithm

$P[] = \{0.01, 0.02, 0.05, 0.1,$
$0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$;
`BUILD_RING()`; {Add all nodes and connect in a ring.}
for each node n **do**
  `CONNECT_W_PROB(`P[`TOTAL_EDGE_COUNT()` MOD 13]);
end for

4.1 Results

The ER_{200} algorithm in Algorithm 1 bears a strong similarity to the Erdös-Rényi model given in Section 2.2.1. It

Algorithm 2 ER_{300} algorithm

```
ADD_ALL_NODES();
for n in 1..N do
  CONNECT_W_PROB(0.01);
  CONNECT_RAND(); {Connect node n to a random node.}
  CONNECT_STUB(TOTAL_VERTEX_COUNT()); {Satisfy an
  edge request in the first P[|V|] * |queue| portion of the
  request queue or make a request if there are none to
  satisfy.}
  CONNECT_W_PROB(0.01);
end for
REWIRE_RANDOM(); {Randomly rewire all edges.}
```

Algorithm 3 ER_{400} algorithm

```
P[] = {0.01, 0.02, 0.05, 0.1,
       0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0};
ADD_ALL_NODES();
for n in 1..N do
  CONNECT_W_PROB(P[TOTAL_EDGE_COUNT() MOD 13]);
  CONNECT_STUB(CLOSENESS_CENTRALITY() MOD 13);
  CONNECT_W_PROB(P[TOTAL_EDGE_COUNT() MOD 13]);
  CONNECT_RAND();
  CONNECT_STUB(BETWEENNESS_CENTRALITY() MOD 13);
end for
REWIRE_RANDOM();
```

makes use of the `CONNECT_W_PROB` function. However, the `BUILD_RING` function adds some additional structure, guaranteeing a connected graph exists, the Erdös-Rényi model makes no such guarantee. However, the target graph does contain a very large connected component which seems to have led to this outcome.

The ER_{300} algorithm is more similar to the Erdös-Rényi model, as shown in Algorithm 2. It adds all vertices to the graph, then systematically adds edges. Note that in the case of a 300 node graph the probabilistic section adds $nodes_p = 0.01{*}300{*}2{*}300 = 1800$ nodes. The `CONNECT_STUB` function adds 150 edges – one edge for each vertex pair. This gives a total of $1800 + 150 = 1950$ edges. The edges are then rewired uniformly over the entire graph. This is similar to the idea of adding a set number of edges randomly to a graph which contains only vertices. In fact, the algorithm automatically discovered a different formulation of the Erdös and Rényi model. The probability of adding an edge in the evolved model becomes[17]:

$$p = m \left(\begin{array}{c} n \\ 2 \end{array} \right)^{-1} = 1950 \left(\begin{array}{c} 300 \\ 2 \end{array} \right)^{-1} = 0.0435, \qquad (9)$$

which is very close to the probability parameter used to construct the target graph.

The ER_{400} model in Algorithm 3 also uses `ADD_ALL_NODES` initialization function. Note the use of the `REWIRE_RANDOM` function, which destroys any initial structure to the graph. The sum result of these operations is to assure the correct number of edges are in the model – similar to the behaviour observed in ER_{300}.

The BA_{200} model in Algorithm 4 utilizes the `GROW_NODES` function, the Barabási-Albert algorithm also adds nodes on each iteration as opposed to all at once. Although the BA_{200} algorithm shows no solid mechanism for preferential attachment. Using `GROW_NODES` in conjunction with the `CONNECT_RAND` function means that nodes added earlier on in the construction of the graph will have higher opportunity to gather edges. The Barabási-Albert model produces many nodes of low degree and only a small number of nodes of high degree, and thus the evolved model is capturing some of this preferential attachment behaviour. However, objective F_3 will not heavily penalize models which do not produce node degrees that occur with low frequency. Future research will propose methods capable of addressing this issue.

Algorithm 4 BA_{200} algorithm

```
SET_GROW_NODES(); {Create an empty graph, add a new
node at each iteration.}
for n in 1..N do
  CONNECT_RAND();
end for
```

Algorithm 5 SW_{200} algorithm

```
BUILD_RING();
for n in 1..N do
  CONNECT_RAND();
  CREATE_TRIANGLE();
  CREATE_TRIANGLE();
end for
REMOVE_PROB(TOTAL_VERTEX_COUNT() MOD 13);
REMOVE_PROB(TOTAL_VERTEX_COUNT() MOD 13);
```

The SW_{200} model, shown in Algorithm 5 is strikingly similar to the actual Small-world algorithm which produced its target graph. The SW_{200} algorithm creates a ring, creates two triangles – the connection pattern which contributes to a high clustering coefficient – and adds one random edge, creating shortcuts across the ring and a short average path length. It finally removes some number of edges probabilistically, breaking up some of the structure introduced by the triangles. A similar effect to rewiring some edges with equal probability as occurs in the target model.

Figures 4, 5 show comparisons of the average histogram produced by thirty 1000 vertex evolved graphs per model to a target graph of the same size of the corresponding target algorithm. The error bars represent one standard deviation. The plots show that the distributions are approximately the same shape and that they overlap considerably. Figure 6 shows a 200 node graph generated by the BA_{200} evolved model and the corresponding target graph produced by the Barabási-Albert model is shown in Figure 7. They display a similar branching structure, but the graph produced by the evolved model does not contain a similar number of high degree vertices. As previously indicated, this will be resolved in subsequent research.

Table 2 shows the average results, μ, and the standard deviations, σ, of the comparisons of graphs produced by the evolved models to target graphs of various sizes which were produced by the algorithms the individuals were evolved to emulate. A fitness value of 1 is the best possible, and 0 is the worst.

Also, Table 2 shows how the evolved models performed when they were used to generate graphs much larger than the initial target graphs. Each evolved model generated thirty graphs of each size and the graphs they produced

Figure 4: The average ER_{400} histogram of thirty 1000 vertex graphs to the histogram of a 1000 node Erdös-Rényi graph.

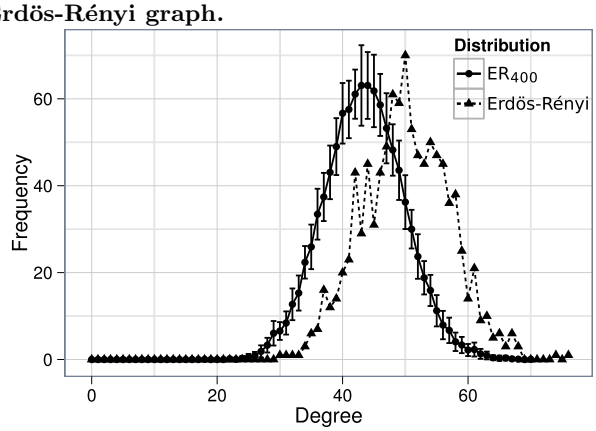


Figure 5: The average SW_{200} histogram of thirty 1000 vertex graphs to the histogram of a 1000 node Small-world graph.

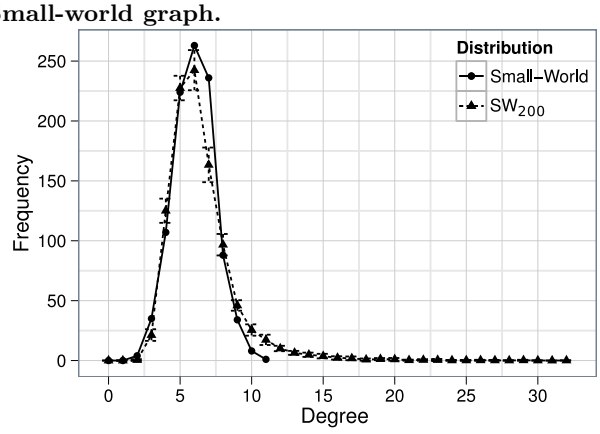


Figure 6: A 200 node graph produced by the BA_{200} model.

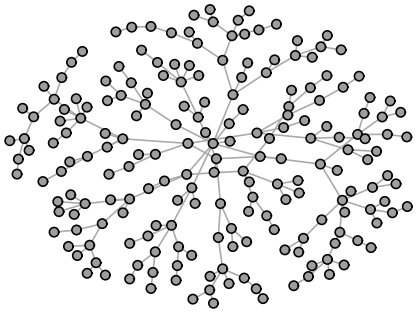
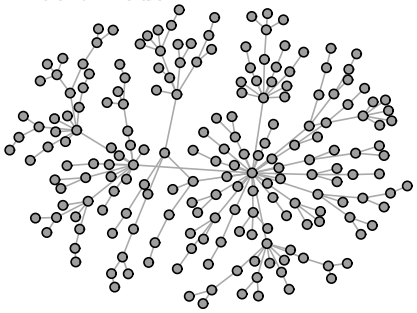


Figure 7: A 200 node graph produced by the Barabási-Albert model.



were compared to target graphs of the same size using fitness functions F_1, F_2, and F_3. The averages and standard deviations from the comparisons are shown in the table. Each model does very well at each target size, and the standard deviations show there is very little variation in the kinds of graphs the evolved models produce.

5. CONCLUSION

This paper presented the first attempt to automating the discovery and design of complex network graph models. Genetic programming was employed to with the goal of reconstructing known and common graph models. This allowed for a more direct evaluation of the efficacy of the approach. The quality of the evolved models was ascertained by not only comparing to the target model, but also respective future manifestations that were unknown during the GP evolution phase. This also allows to determine whether the resulting evolved models are over-fitting to the target graph.

Our results focused on undirected and unweighted graphs of 200 to 500 vertices in size. We find that the proposed GP language is capable of reasonably discovering the underlying model algorithm of each of the Erdos-Renyi, Watts Strogatz and Barabasi-Albert target graphs. An interesting result was obtained for the Erdos-Renyi case, where an alternate formulation of the graph model was discovered. In all cases the evolved models also performed admirably when tested into the future. These results are extremely promising.

This research direction has just commenced, and as such there remains a large number of open questions to be addressed. Further evaluation and expansion of the basic GP language presented here is the most obvious first step. However, considering other graph models and graphs of much

Table 2: Comparison of graphs produced by evolved models to graphs produced by real algorithms

Model	Obj	Size, n									
		$n = 200$		$n = 400$		$n = 600$		$n = 800$		$n = 1000$	
		μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
ER_{200}	F_1	0.997	7.34×10^{-5}	0.999	2.02×10^{-5}	0.999	2.65×10^{-5}	0.999	8.75×10^{-6}	0.999	5.01×10^{-6}
	F_2	0.978	2.93×10^{-3}	0.973	1.22×10^{-3}	0.974	8.32×10^{-4}	0.974	7.98×10^{-4}	0.973	6.41×10^{-4}
	F_3	0.810	0	0.844	1.26×10^{-3}	0.875	6.42×10^{-3}	0.889	1.11×10^{-2}	0.898	5.28×10^{-3}
ER_{300}	F_1	0.997	1.91×10^{-4}	0.999	3.41×10^{-5}	0.999	1.35×10^{-5}	0.999	6.19×10^{-6}	0.999	6.03×10^{-6}
	F_2	0.986	6.56×10^{-3}	0.975	1.23×10^{-3}	0.975	1.24×10^{-3}	0.975	7.33×10^{-4}	0.974	4.76×10^{-4}
	F_3	0.857	9.80×10^{-3}	0.867	1.12×10^{-2}	0.883	6.98×10^{-3}	0.894	6.85×10^{-3}	0.903	6.23×10^{-3}
ER_{400}	F_1	0.999	1.02×10^{-4}	0.999	2.95×10^{-5}	0.999	1.48×10^{-5}	0.999	7.72×10^{-6}	0.999	4.61×10^{-6}
	F_2	0.993	3.38×10^{-3}	0.997	1.34×10^{-3}	0.996	9.08×10^{-4}	0.995	6.71×10^{-4}	0.994	3.89×10^{-4}
	F_3	0.928	1.80×10^{-3}	0.964	7.51×10^{-3}	0.961	7.44×10^{-3}	0.960	4.85×10^{-3}	0.955	4.53×10^{-3}
BA_{200}	F_1	0.988	0	0.991	0	0.995	2.28×10^{-4}	0.995	8.53×10^{-5}	0.998	8.14×10^{-5}
	F_2	1	0	1	0	1	0	1	0	1	0
	F_3	0.95	0	0.81	0	0.860	8.03×10^{-3}	0.826	1.75×10^{-3}	0.832	1.01×10^{-2}
SW_{200}	F_1	0.998	1.63×10^{-4}	0.999	8.34×10^{-5}	0.999	3.78×10^{-5}	0.999	2.28×10^{-5}	0.999	2.04×10^{-5}
	F_2	0.957	5.07×10^{-3}	0.949	2.99×10^{-3}	0.968	2.59×10^{-3}	0.952	1.75×10^{-3}	0.971	1.49×10^{-3}
	F_3	0.932	2.28×10^{-2}	0.920	1.18×10^{-2}	0.949	1.33×10^{-2}	0.956	9.12×10^{-3}	0.927	1.50×10^{-2}

larger size is also an important direction. Similarly, considering directed and weighted graphs, as well as more dynamic graph models is enormously interesting. The application to real-world problems and comparing to best-known human approximations is also research of utmost importance to truly ascertain the degree of human-competitiveness this approach can attain.

6. REFERENCES

[1] L. A. Adamic and B. A. Huberman. Growth dynamics of the world wide web. *Nature*, 401(6749):131, 1999.

[2] R. Albert and A. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74(1):47, 2002.

[3] L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley. Classes of small-world networks. *PNAS*, 97(21):11149–11152, 2000.

[4] G. M. Ames, D. B. George, C. P. Hampson, A. R. Kanarek, C. D. McBee, D. R. Lockwood, J. D. Achter, and C. T. Webb. Using network properties to predict disease dynamics on human contact networks. *Proc. R. Soc. B*, 2011. published online before print.

[5] A. L. Barabási and R. Albert. Emergence of scaling in random networks. *Science (New York, N.Y.)*, 286(5439):509–512, Oct. 1999.

[6] A. L. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *PHYSICA A*, 311:3, 2002.

[7] P. Bentley and J. Wakefield. Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. In *Soft Computing in Engineering Design and Manufacturing*, pages 231–240, London, June 1997. Springer.

[8] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph, 1998.

[9] I. Chakravarti, R. Laha, and J. Roy. *Handbook of methods of applied statistics*. Number v. 1 in Wiley series in probability and mathematical statistics. Wiley, 1967.

[10] L. Egghe and R. Rousseau. *Introduction to Informetrics : quantitative methods in library, documentation and information science*. Elsevier Science Publishers, 1990.

[11] P. Erdös and A. Rényi. On random graphs, i. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[12] D. A. Fell and A. Wagner. The small world of metabolism. *Nature Biotechnology*, 18(11):1121–1122, 2000.

[13] R. Flack. *Evolution of Architectural Floor Plans*. Brock COSC TR CS-1103, January 2011.

[14] R. Flack. The robgp genetic programming system, 2011. http://robgp.sourceforge.net/.

[15] B. A. Huberman. *The Laws of the Web: Patterns in the Ecology of Information*. MIT Press, Cambridge, MA, USA, 2001.

[16] J. R. Koza. *On the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[17] M. Newman. *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010.

[18] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[19] M. E. J. Newman and M. Girvan. Community structure in social and biological networks. *PNAS*, 99(12):7821–7826, 2002.

[20] D. D. S. Price. A general theory of bibliometric and other cumulative advantage processes. *JASIST*, pages 292–306, 1976.

[21] J. P. Scott. *Social Network Analysis: A Handbook*. SAGE, Thousand Oaks, CA, 2000.

[22] S. Wasserman and K. Faust. *Social network analysis*. Cambridge University Press, Cambridge, 1994.

[23] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[24] S. H. Yook, H. Jeong, A. L. Barabási, and Y. Tu. Weighted evolving networks. *Phys. Rev. Lett.*, 86(25):5835–5838, 2001.

[25] L. Zager, M. I. of Technology. Dept. of Electrical Engineering, and C. Science. Graph similarity and matching. 2005.