Research Article

A Genetic Simulated Annealing Algorithm to Optimize the Small-World Network Generating Process

Haifeng Du1, Jiarui Fan1, Xiaochen He1, and Marcus W. Feldman1,2

1Center for Administration and Complexity Science of Xi’an Jiaotong University, Xi’an, Shanxi Province 710049, China
2Morrison Institute for Population and Resource Studies, Stanford University, Stanford, CA 94305, USA

Correspondence should be addressed to Marcus W. Feldman; mfeldman@stanford.edu

Received 7 August 2018; Accepted 15 October 2018; Published 6 November 2018

Academic Editor: Lingzhong Guo

Network structure is an important component of analysis in many parts of the natural and social sciences. Optimization of network structure in order to achieve specific goals has been a major research focus. The small-world network is known to have a high average clustering coefficient and a low average path length. Previous studies have introduced a series of models to generate small-world networks, but few focus on how to improve the efficiency of the generating process. In this paper, we propose a genetic simulated annealing (GSA) algorithm to improve the efficiency of transforming other kinds of networks into small-world networks by adding edges, and we apply this algorithm to some experimental systems. In the process of using the GSA algorithm, the existence of hubs and disassortative structure is revealed.

1. Introduction

The small-world phenomenon is receiving much attention in the study of complex systems. Milgram first pointed out the small-world phenomenon by verifying the six-degree separation hypothesis in 1967 [1]. Subsequently, activities such as the game “Six Degrees of Kevin Bacon” [2] or the baseball players’ social network [3] have shown that social networks can have small-world characteristics. Watts and Strogatz proposed a dynamic network model (WS model) to demonstrate the small-world phenomenon [4], namely, in networks with higher average clustering coefficient (ACC) and lower average path length (APL). Since then, a substantial volume of research related to small-world network models has been produced. Since the rewired edges introduced into the dynamic model by the WS model may lead to isolated nodes [4], a model that added shortcut edges was proposed by Watts and Newman [5]. They replaced random reconnection by adding edges randomly, thus avoiding the possibility of generating isolated nodes. Monasson [6] studied the Laplacian operator on “small-world” lattices, which plays an important role in the diffusion of information. In order to solve the problem of redundancy and low efficiency of the network, a direct small-world network model (DSW) was proposed to optimize the network, which could be applied effectively to study dissemination of community information and epidemiological transmission [7]. In these models, the small-world phenomenon lies between the regular network structure and random network structure, and the dynamics that lead to evolution of small-world networks have been explored.

In many real-world fields, including computer science [8], propagation and control of disease [9], transport networks [10], biology [11], brain neurons [12, 13], and sociology [14], small-world networks emerge. The dynamic system model with the small-world property can accelerate the speed of signal transmission and improve computing capacity and computational synchronization. Rumors and epidemics spread more easily in small-world networks than in regular networks. Many well-known biological networks exhibit interconnectedness of small-world network nodes, and this network model can simulate some important properties of neural networks [15]. Li Nan et al. [16] used small-world theory to change the structure of a game network and found that the small-world network had the fastest cooperative convergence and greatest information transmission efficiency. Cowan and Jonard [17] stated that a network structure with lower average path length and higher clustering coefficient is
beneficial to knowledge diffusion, which can improve the frequency of knowledge transfer and promote cluster development. These networks have been shown to strongly diffuse information, which has prompted research on how to create small-world networks. Kleinberg [18] proposed a decentralized search model for small-world networks and gave an optimized network description. Inspired by Kleinberg’s research [18] on small-world networks, Yang and Wang [19] formulated a decentralized small-world optimization strategy, which has excellent search performance and efficiency. Walsh [20] studied the search problem by a new strategy of randomization and restarts to reduce the cost of solving the problem using graph theory. Schleich et al. [21] focused on vehicular ad hoc networks (VANETs) and proposed a multiobjective algorithm to create the small-world property in unconnected networks, which could find a high-quality solution to a multi-objective optimization problem. Cavalcanti et al. [22] introduced shortcuts in wireless ad hoc networks, analyzed the system from a percolation perspective, and were able to significantly improve connectivity. These studies have improved the efficiency of the generating process to some extent, but the optimization problem in terms of improving the efficiency of transforming other kinds of networks into small-world networks by adding edges has still not been resolved. An efficient optimizing algorithm for improving the small-world property has also not yet been proposed.

In this paper, in order to improve the efficiency of the generating process, we design an optimization model to improve the efficiency of generating networks that have a strong small-world property. This is useful as it correlates with a set of important targets such as improving diffusion efficiency. The rest of the paper is organized as follows. In Section 2, we review previous studies related to constructing or transforming small-world networks and define a synthesized small-world property as our objective function. In Section 3, we propose a genetic algorithm based on simulated annealing to optimize the small-world property. In Section 4, we test the performance of our proposed algorithm in regular networks and random networks. Our conclusions and ideas for further work are presented in Section 5.

2. Related Background

2.1. Evaluation of Optimization Methods. The small-world network structure is intermediate between regular network structure and random network structure. Watts and Ströglatz first proposed a dynamic model of small-world network (WS model) [4], in which they randomly reconnected existing edges in a regular network with probability $p$ ($0 < p < 1$), as shown in Figure 1. This construction generated a graph between regular ($p = 0$) and random ($p = 1$). In the process of reconnecting edges, they found an intermediate status that has high ACC, like regular graphs, and low APL, like random graphs, which became known as the “small-world property.” The WS model not only introduces randomness but also introduces long-range connections into the regular network, which can reduce APL and maintain relatively high ACC compared with the original network. However, this model may lead to an unconnected network.

In fact, optimizing the efficiency of the small-world-network-generated involved maximizing the ACC and minimizing the APL. In order to avoid the problem of isolated nodes in the process of rewiring edges, Newman and Watts focused on the dynamic small-world model by adding edges (NW model) [5]. This method takes less time and is linearly related to the number of added edges, but it cannot guarantee the effectiveness of the resulting small-world network when the number of edges is limited. Moreover, the process of adding edges in this model has a random component, and the optimization result is usually unstable.

The greedy algorithm, which adds edges one by one, can be efficient in optimizing the small-world property. In real life, this method has been applied to the optimization of wind [23] and tidal stream resources [24] in order to improve power generation efficiency. The strategy of the greedy algorithm is to add $k$ edges one by one, with each of the added edges giving the maximum small-world property. Compared with the NW model, the greedy algorithm can obtain stable results and improve the efficiency of the generating process. However, some deficiencies of the greedy algorithm need to be acknowledged. First, it traverses the search space for each edge to be added. If $k$ edges are added to a network of size $N$, the time complexity of the greedy algorithm is $O(kN^2)$, which means that maximizing the small-world property suffers from a combinatorial explosion problem. For a network defined as a graph $G = (V, E)$ with $N = ||V||$ nodes and $K = ||E||$ the initial number of edges, where $|V_i| \in V$ represents the set of nodes, there can be $\left( \frac{N(N-1)}{2}-K \right)$ ways to add $k$ edges to the network. So if we want to traverse all the solutions, the complexity can be frustrating even for a small network. Second, the greedy algorithm fails to find the global extremum and may have a high probability of falling into a local extremum.

In order to solve these problems, we propose a genetic simulated annealing (GSA) algorithm, which has been widely used and has proved to be effective in solving such combinatorial explosion problems. Simulated annealing (SA) is integrated into a genetic algorithm (GA), which can guarantee the diversity of the population and improve the global search. Combining SA with GA, Sirag et al. used a model of genetic activity based on the Boltzmann distribution to control the rate of population convergence [25]. Boseniuk et al. also combined the two algorithms to solve relevant optimization problems [26]. Some other problems, such as SaaS optimization placement [27], geometric constraint problems [28], and the resource-constrained project scheduling problem (RCPSP) [29] have also been solved by this method.

2.2. The Ideology of the New Algorithm. Watts and Ströglatz pointed out that small-world network has the characteristic of relatively low APL, like a random lattice, and relatively high average ACC, like a regular lattice [4]. In this paper, we focus on APL and the average ACC as the main targets to measure the small-world property.

APL is the average shortest distance between all pairs of nodes in the network. For an undirected graph $G = (V, E)$, where $N = ||V||$ represents the number of nodes, $K = ||E||$
represents the initial number of edges, and \(d(v_i, v_j)\) represents the distance between nodes \(v_i, v_j \in V\), \(N(N-1)\) is the number of paths of all pairs of nodes in the network. APL can be written as \(APL = \sum_{ij} d(v_i, v_j)/N(N-1)\). On the other hand, \(ACC_{local}\) indicates the level of node aggregation in a graph; \(ACC_{local}\) of node \(v\) with \(n_v\) neighbors is \(ACC_{local} = 2|E(Γ_v)|/n_v(n_v-1)\), where \(|E(Γ_v)|\) is the number of edges in the relational neighborhood of \(v\) and \(n_v(n_v-1)/2\) is the number of possible edges in the relational neighborhood of node \(v (v \in V)\). \(ACC_{average}\) is the mean value of the \(ACC_{local}\) for all nodes in the network; i.e., \(ACC_{average} = \sum_{i} ACC_{local}/N\). Then \(SW\) is defined as

\[
SW = \frac{ACC_{average}}{APL}. \tag{1}
\]

We use \(SW\) to maximize the small-world property that is defined in (1). It is clear that a greater \(SW\) corresponds to a better small-world property. Therefore, \(SW\) can be regarded as a standard for evaluating the effectiveness of optimizing methods; i.e., the greater the \(SW\) of the optimized network, the better the algorithm performs.

In this paper, we maximize \(SW\) by adjusting edges. Generally, a network can be defined as an undirected graph \(G = (V, E)\) with \(N = |V|\) nodes, \(K = |E|\) the initial number of edges, and \(\{v_i\} \in V\) the set of nodes. We add \(k\) edges to produce a network \(G' = (V', E')\) that gives the maximum \(SW\). The objective function can be formulated as

\[
\text{max} \quad SW \\
\text{s.t.} \quad G(V, E) \rightarrow G'(V', E') \tag{2}
\]

In (2), \(SW\) is the small-world property, and \(|E'|\) represents the number of edges in the optimal network. To maximize \(SW\), we propose the GSA algorithm in the next section.

3. Algorithm

3.1. Framework. Algorithm 1 gives the framework of the GSA algorithm. We first set the necessary parameters and input the initial network to be optimized. We then start a series of iteration loops. Before the iteration number \(i\) reaches \(I_{max}\), we repeat a process for finding the best solution. In this repeated process, we first calculate the score of population by the objective function in (2); we then select the parent population via seed selection and roulette selection; next, we generate the offspring population by Monte Carlo sampling. When the iteration number \(i\) reaches \(I_{max}\), we output the group of added edges and the optimized network.

3.2. Representation and Initialization. The solution can be encoded as a chromosome consisting of sequential numbers and represents the unconnected pairs of nodes. We encrypt all the unconnected edges in the order of the sequential numbers, and the length of a chromosome is equal to the number of added edges in an optimization. Figure 2 shows an illustration of the representation. The second gene of the chromosome changes from 2 to 3, which entails dropping the edge between nodes 1 and 4 and adding a shortcut between nodes 1 and 5. Initially, we generate a population and randomly assign a set of numbers to each gene in the population.

3.3. Natural Selection. Natural selection is divided into seed selection and roulette selection. Seed selection ensures that the best chromosome is selected at each selection event. All population scores are sorted and the best 5% are selected as the seed at each selection. The remaining part is by roulette selection, which is based on the principle of "survival of the fittest". The higher the chromosome’s score, the higher the probability of being selected. This ensures some diversity in the population.

3.4. Genetic Mutation and Simulated Annealing. As shown in Algorithm 2, each gene in the chromosome is mutated at rate \(P_m\). If mutation happens, a new number is selected randomly from encryption to replace the original number. Whether the original number will be replaced or not is decided by the Monte Carlo method: we calculate the score of the original (\(SW_{old}\)) and the mutant (\(SW_{new}\)). If the mutant’s score is higher than the original, then the mutant is accepted instantly; otherwise the mutant is accepted with probability \(P_a\) produced according to (3). The temperature will then fall
1. Input: The initial network: G; The number of added edges: k; The size of population: S_{pop}; The mutation rate: P_m; The number of iteration: I_{max}; The initial annealing temperature: T; The cooling coefficient of temperature: DELTA.
2. Encrypt unconnected edges of G, then get edges encryption;
3. Initialize population with edges encryption, i = 1;
4. Repeat;
5. Calculate score of population using SW by evaluate population;
6. Execute operation of natural selection to population according to score, 5% population were selected by seed selection and 95% population were selected by roulette selection;
7. Execute operation of mutation to population;
8. T = T \ast DELTA, i += 1;
9. If i < I_{max}, turn to 4, else end;
10. Output: the group of added edges, the optimized network.

Algorithm 1: GSA algorithm.

<table>
<thead>
<tr>
<th>Positions</th>
<th>Node</th>
<th>Node</th>
<th>Sequence Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

Chromosome 1: 1 2 5 8
Chromosome 2: 1 3 5 8

Figure 2: Illustration of genetic representation. Upper left lists the unconnected pairs of nodes and their sequence numbers. Lower left shows the corresponding chromosomes. Right shows the network topology corresponding to chromosomes. The solid lines represent the original edges of the network, the dashed lines represent added edges, and the numbers next to nodes denote the node number.

by multiplying the cooling coefficient \(DELAT^{i-1}\) (\(DELAT\) is initialized as 0.98), and this completes one iteration. \(T\) is the initial temperature of simulated annealing, which should be 10 times \(\Delta f\), where \(\Delta f = SW_{\text{new}} - SW_{\text{old}}\). Here \(i\) is the number of the iterations.

\[
P_a = e^{\Delta f / T}
\]

\[
\Delta f = SW_{\text{new}} - SW_{\text{old}}
\]

\[
T_i = T \ast DELAT^{i-1}
\]

The advantage of adding simulated annealing to the genetic algorithm is that it may not only prevent the population from falling into a local optimal solution, but also increase the diversity of the population. In addition, without the simulated annealing, the optimization due to the genetic algorithm will be unstable due to the randomness of gene mutation.

3.5. Iteration Loop. The iteration loop in the GSA algorithm is shown in Figure 3. First, we evaluate the population’s fitness with SW. Second, we choose the population by natural selection. Finally, we execute gene mutation, which contains the mechanism of simulated annealing. The annealing temperature is adjusted after each iteration.

4. Experimental Analysis

4.1. Experiment Overview. The experiment was executed on a quad-core laptop (CPU: Intel(R) Core (TM) i7-3537U, 2.50GHz, Memory: 8.00 GB) installed with Windows 10 operating system. We used Python 3.6.3 as our programming language combined with NetworkX. The experiments for each optimization were carried out six times, and we selected the best results and the average results for the different methods. Table 1 shows the parameters used in the experiments.
1. Input: The initial network: $G$; The size of population: $S_{pop}$; The mutation rate: $P_m$; The annealing temperature: $T$; The cooling coefficient of temperature: $DELTA$.
2. Choose a new chromosome from $S_{pop}$.
3. Initialize a random number $n$ in $(0, 1)$, if $n < P_m$, turn to 4; If $n \geq P_m$, turn to 9.
4. Select a random integer from edges encryption to replace one of numbers in the chromosome, thus produce an mutated chromosome;
5. Calculate $SW_{old}$ by add edges with original chromosome and get $SW_{new}$ with mutated chromosome;
6. $\Delta f = SW_{new} - SW_{old}$;
7. If $\Delta f > 0$, replace original chromosome by mutated chromosome and turn to 9; If $\Delta f < 0$, turn to 8;
8. $P_a = e^{(\Delta f/T)}$, initialize a random number $m$ in $(0, 1)$. If $m < P_a$, replace original chromosome by mutated chromosome;
9. If all the number in the original chromosome have been selected then turn to 2, else turn to 3;
10. End.

**Algorithm 2:** The Monte Carlo method.

![Algorithm Diagram]

**Figure 3:** The iteration loop.

### 4.2. Optimization Results on Regular Network and Random Network

#### 4.2.1. Optimization Results for the Small-World Property

In this section, we execute the NW model, the greedy algorithm, and the GSA algorithm in a regular network (the network size is 50 and the average node degree is 4) and a random network (the network size is 50 and the probability of edge creation is 0.1), respectively, to test its performance on different network structures. The $SW$, assortativity coefficient $r$, and topological rules are used to measure the performance of different algorithms in the experiments.

The assortativity coefficient $r$ can be formulated \[30\] as

$$r = \frac{M^{-1} \sum_{i} j_i k_i - \left[ M^{-1} \sum_i (1/2) (j_i + k_i) \right]^2}{M^{-1} \sum_i (1/2) (j_i^2 + k_i^2) - \left[ M^{-1} \sum_i (1/2) (j_i + k_i) \right]^2}$$

where $j_i$, $k_i$ are the degrees of the vertices at the ends of the $i^{th}$ edge. From the analysis, if $r > 0$, the network has assortative structure, which means that large-degree nodes tend to connect with large-degree nodes. If $r < 0$, then the network has disassortative structure, which means large-degree nodes tend to connect with small-degree nodes. $r = 0$ means there is no correlation between node connectivity properties.

Figure 4 shows the optimization results for three algorithms on the regular network and the random network. The results of the six runs of the greedy algorithm are all the same, so the average and maximum values are the same and their curves overlap in Figure 4. Figure 4(a) also shows a curve labeled “corresponding network,” which is a random network corresponding to the results of conventional network optimization on the regular network. We randomly reconnected all the edges (including both original edges and added edges) of the optimal regular network and the value $SW$ of this reconnected network is represented by the curve.
Table 1: Parameters of the experiment.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{\text{max}}$</td>
<td>Maximum number of iterations</td>
<td>500</td>
</tr>
<tr>
<td>$S_{\text{pop}}$</td>
<td>Size of population</td>
<td>100</td>
</tr>
<tr>
<td>$T$</td>
<td>The initial temperature</td>
<td>0.01</td>
</tr>
<tr>
<td>$P_{m}$</td>
<td>Probability of Mutation</td>
<td>0.025</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>Cooling coefficient Delta</td>
<td>0.98</td>
</tr>
</tbody>
</table>

Figure 4: Optimizing results for improving efficiency of transforming other networks to small-world networks. The figures show small-world property for three algorithms. (a) shows the results of small-world property optimization on a regular network. The results of corresponding network compute the average value $SW$ after 100 times by reconnecting all the edges randomly on a regular network. (b) shows the results of small-world property optimization on a random network. Since the results of the six runs of the greedy algorithm are the same, the max and mean values of the greedy algorithm overlap.
4.2.2. The Principle of Hubs. In this section, we explore the topological structures of added edges generated by the three algorithms on the optimized regular network. In Figures 5–7 we see that the NW model, the greedy algorithm, and the GSA algorithm have obvious topological rules, like the Matthew effect [31]. Figure 5 shows the NW model has no rules for adding edges. Figures 6 and 7 show the existence of hubs in the process of adding edges via the greedy algorithm and the GSA algorithm, respectively. Figure 7 shows that all the added edges were connected to the same node via the GSA algorithm when the added edges k is smaller than the network size. Not all the added edges are connected to the same node by the greedy algorithm as shown in Figure 6. Thus, in the process of adding edges the phenomenon of hubs with the GSA algorithm is more apparent than for the greedy algorithm.

From Figures 6 and 7, we see that edges can be added with hubs to obtain the topological structure of the small-world phenomenon in a regular network. This method produces a structure with a higher SW quickly and efficiently. A regular network can be defined as a graph $G = (V, E)$ with $N = |V|$ nodes and $|E|$ initial number of edges, $\{ v_i \} \in V$ represents the set of nodes. $v_i = n$ is the original degree of each node. If we add $k$ edges to produce a network $G' = (V', E')$ with the maximum SW, the function can be formulated as (5) via the GSA algorithm.

$$\begin{align*}
\text{max} & \quad SW \\
\text{s.t.} & \quad G(V, E) \xrightarrow{k} G'(V, E') \\
& \quad v_i = n \rightarrow v_i' = n + k
\end{align*}$$

\[ (5) \]
4.2.3. Assortativity. In order to determine the structural effect of different algorithms for optimizing SW, we use the assortativity coefficient \( r \) [30], which measures the extent of assortative (or disassortative) mixing.

Figure 8 shows the assortativity coefficient \( r \) of the optimized regular network and a random network. Both networks show a similar trend. The NW model shows assortative structures while the greedy algorithm and the GSA algorithm show disassortative structure. That is, large-degree nodes tend to connect with small-degree nodes in optimizing the small-world property by the GSA algorithm and the greedy algorithm. Compared with the greedy algorithm, the GSA algorithm shows disassortative structure more clearly. Figure 8(a) shows that the GSA algorithm is more disassortative than the greedy algorithm when \( k > 2 \) edges are added to the regular network. Figure 8(b) shows that GSA performs much better when \( k > 10 \) on a random network. Especially when \( k > 36 \), the GSA algorithm decreases \( r \) sharply, leading to significant growth in SW. This also means that the GSA algorithm shows the same principle of hubs as for the regular network when the structure of the random network reaches a certain scale.

4.3. Experimental Results on Benchmark Network. In this section, we discuss simulations run on a new computer-generated network, which is an improved network based on the simulation network proposed by Newman [32]. The simulation network is divided into 4 communities and consists of 128 nodes with an average degree of 16. We introduce a parameter \( \mu \) to represent the probability of links between different communities; \( 1 - \mu \) represents the probability of links lying within the same communities.

We use Newman’s modularity \( Q \) [32], which measures community structure, to determine whether there is a small-world phenomenon in the network. \( Q \) can be formulated as

\[
Q = \frac{1}{4\omega} \sum_i \sum_j \left( \omega_{ij} - \frac{\omega_i \omega_j}{2\omega} \right) (s_is_j + 1)
= \frac{1}{4\omega} \sum_i \sum_j \left( \omega_{ij} - \frac{\omega_i \omega_j}{2\omega} \right) s_is_j
\]

for a particular division of the network into two communities. The coefficient \( s_i = 1 \) if node \( i \) belongs to community 1, and the coefficient \( s_j = -1 \) if node \( j \) belongs to community 2. Observe that the quantity \((1/2)(s_is_j + 1)\) is 1 if node \( i \) and node \( j \) are in the same community and 0 otherwise. The coefficient \( \omega_{ij} \) equals 1 if there is a relationship between node \( i \) and node \( j \); otherwise, \( \omega_{ij} = 0 \). The sum \( \omega_i = \sum_j \omega_{ij} \) indicates the degree of node \( i \), and \( 2\omega = \sum_i \omega_i = \sum_i \sum_j \omega_{ij} \) represents the sum
of the degrees in the network. Therefore, the larger the modularity \( Q \), the closer the connections within the community, which means the network has more community structure. From Table 2, the original benchmark network has obvious community structure and shows a small-world phenomenon.

In order to measure the topological rules of added edges, we use the formula \( P_k = \frac{k_{\text{out}}}{k} \), where \( k \) is the number of added edges and \( k_{\text{out}} \) is the sum of added edges between communities. From Table 2, the GSA algorithm performs better than the NW model on small-world property \( SW \) and modularity \( Q \). \( P_k \) shows that GSA tends to connect nodes within the community and has few edges between communities, while the NW model tends to connect randomly, based on the proportion of unconnected edges in the network. This is because the increase in \( ACC \) is greater than the decrease in APL when adding edges inside a community, compared with adding edges between communities. Figure 9(c) shows that when the number of added edges is \( k = 50 \), the sum of added edges between communities equals 1 via the GSA algorithm. It also shows that the GSA algorithm can clearly enhance the small-world phenomenon on benchmark networks.

5. Conclusion

In this paper, we propose a synthesized target \( SW \), which concisely indicates the small-world property. In order to optimize the small-world property, the GSA algorithm is shown to be effective. We explore network topologies to find a principle of adding edges by topological analysis. We exhibit disassortative structure and the clear existence of hubs in the process of adding edges. We also test and verify the efficiency of our algorithm on benchmark networks to explore community structure. Our algorithm performs well in enhancing the small-world status of the original benchmark network. We optimize only one indicator, \( SW \), in this paper, but in many cases, it would be preferred to minimize APL and maximize ACC at the same time. This multiobjective structural optimization should be explored in future work.

Data Availability

The data supporting our results comes from our designed procedure. The procedure code and the final results are included within the Supplementary Material files.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.
Table 2: Comparison between the NW model and the GSA algorithm.

<table>
<thead>
<tr>
<th>Index</th>
<th>NW model</th>
<th>GSA algorithm</th>
<th>NW model</th>
<th>GSA algorithm</th>
<th>NW model</th>
<th>GSA algorithm</th>
<th>NW model</th>
<th>GSA algorithm</th>
<th>NW model</th>
<th>GSA algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SW$</td>
<td>0.1778</td>
<td>0.1782</td>
<td>0.1784</td>
<td>0.1786</td>
<td>0.1776</td>
<td>0.1773</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Q$</td>
<td>0.1149</td>
<td>0.1154</td>
<td>0.1153</td>
<td>0.1160</td>
<td>0.1157</td>
<td>0.1154</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_k$</td>
<td>0</td>
<td>0.8833</td>
<td>0.8500</td>
<td>0.8611</td>
<td>0.8667</td>
<td>0.8733</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Acknowledgments

The research was supported by the Key Project from the National Science Foundation of China (Grant No. 12AZD110), the Humanities and Social Science Talent Plan, and the Morrison Institute for Population and Resource Studies at Stanford.

Supplementary Materials

A concise description of each supplementary material file is listed in Table 1. The folder “code” contains the source code of three algorithms used in the paper. The folder “results of data” contains all the results of running the program. Specifically, it includes experimental results from running on regular
networks, random networks, and benchmark networks, as well as all the topological graphs from running the regular networks via three algorithms. (Supplementary Materials)

References


