Network structure optimization for social networks by minimizing the average path length

Wei Du · Gang Li · Xiaochen He

Received: 30 September 2021 / Accepted: 31 January 2022 / Published online: 16 February 2022
© The Author(s), under exclusive licence to Springer-Verlag GmbH Austria, part of Springer Nature 2022

Abstract
Network structure plays an important role in the natural and social sciences. Optimization of network structure in achieving specified goals has been a major research focus. In this paper, we focus on structural optimization in terms of minimizing the network’s average path length (APL) by adding edges. We suggest a memetic algorithm to find the minimum-APL solution by adding edges. Experiments show that the proposed algorithm can solve this problem efficiently. Further, we find that APL will ultimately decrease linearly in the process of adding edges, which is affected by the network diameter.

Keywords Network · Structure · Optimization · Algorithm

Mathematics Subject Classification 68W50

These authors contributed equally to this work.

Xiaochen He
xiaochenhe@xjtu.edu.cn

Wei Du
duwei@mail.xjtu.edu.cn

Gang Li
18202910309@163.com

1 School of Public Policy and Administration, Xi’an Jiaotong University, West Xianning Road, Xi’an 710049, Shaanxi, People’s Republic of China

2 Center for Administration and Complexity Science, Xi’an Jiaotong University, West Xianning Road, Xi’an 710049, Shaanxi, People’s Republic of China

3 School of Economics and Finance, Xi’an Jiaotong University, Yanta West Road, Xi’an 710061, Shaanxi, People’s Republic of China
1 Introduction

Social networks have been analyzed over the past several decades. Most social network research can be classified into the following three fields: (1) analysis of network characteristics, which focuses on finding indices to quantify features of network structure (such as modularity) and utilizing these indices to distinguish broad categories of network structure (such as small-world phenomena, scale-free property) [1–4]; (2) exploration of mechanisms of network structure formation using statistical or dynamic models such as the Watts–Strogatz or Barabási–Albert models [5–7]; (3) the dynamic evolution of networks, which focuses on the transformation of network structures based on specific models in many fields, such as transmission of infection [8] and human cooperation [9]. Obviously, network structure plays a fundamental role in the analysis of social networks, and optimizing network structure can help to improve network performance.

Previous research about optimizing network structure has shown that the primary methods to improve network performance include adding edges, rewiring edges and adding nodes with changing network topology. Watts and Strogatz [5] found the small-world network structure by rewiring network edges through optimization of the clustering coefficient and average path length (APL). Other scholars have concentrated on improving network characteristics by adjusting network structure [10–12]. In this paper, we call this process “network structure optimization”. It is the process of adjusting the network structure (adding, deleting or rewiring edges) to improve performance with respect to a specified outcome. Although there are different methods of improving structural performance, a standard definition of structural optimization has yet to be proposed. Generally, a network can be defined as a graph $G = (V, E)$, where $\{v_i\} \in V$ represents the set of nodes; $\{e_{ij}\} \in E \rightarrow \{0, 1\}$ represents the set of edges, $e_{ij} = 1$ denotes a relationship between nodes i and j, and $e_{ij} = 0$ indicates no relationship between nodes i and j. Network structure optimization is as follows: for a specific performance $P_G$ of a network $G = (V, E)$, structural optimization can be formulated as the process of optimizing $P_G$ by changing the network structure $G = (V, E)$ to $G' = (V', E')$, which produces $P_{G'}$ such that $P_{G'} < P_G$.

Optimizing network structure has several applications, such as in transportation networks [13, 14] and communication networks [15]. Take the airline network as a real world example, because of resource constraints, creating some new airlines between different cities should fully consider the maximization of the whole network’s efficiency. So we want to select a few connections that will maximize the overall benefit by making the network a smaller world.

Problems related to structural optimization have been widely discussed [10, 16–19]. Barahona et al. [20] treated balance as an optimization problem, and Wang et al. [21] proposed that the transformation of an imbalanced network to a balanced one at the lowest cost is also an optimization problem. Yang and Wang [22] proposed a decentralized small-world optimization method for improving search efficiency. There are different indicators to reflect the effect of network optimization, among them, the small-world properties attracts most attention. For example, Schleich et al. [23] found a high-quality solution to improve small-world properties on vehicular ad hoc networks. Du et al. [24] presented a more effective method for optimizing small-world network
generating process. Alstott et al. [25] proposed a method that create a highly clustered network by starting with an existing network and rearranging edges, without adding or removing them.

The small-world properties can be characterised by several methods. However, our focus in this article is the APL. APL is a fundamental quantity that contains important information for understanding the behaviors of networks [26]. For example, APL can be used to analyze topological structures of real world systems such as the World Wide Web [7], to measure the transfer efficiency of a metabolic network [27], or to measure robustness with respect to random failures [28]. To minimize APL, we can either decrease communication cost [29], or increase synchronization [30–34]. Here, we take APL as the index to be optimized.

In this paper, we propose a memetic algorithm to optimize APL. This is useful as it correlates with a set of important indexes such as robustness or transmission efficiency, a lower APL is a highly desirable property since it facilitates and amplifies dynamic processes in the network, such as information dissemination, or viral marketing. Several methods focusing on the optimization of APL have been proposed, but the effectiveness of the optimization can still be improved, especially for optimization of random networks. This paper proposes a more effective method to optimize APL. Moreover, some network features of the optimal networks need to be analyzed in order to determine the effect of optimizing APL on different networks, and this analysis can find out the added edges with which feature can decrease APL more efficiently. And we also compute several indices related to optimal networks, such as assortative index or network diameter that relate to the optimal network structure. In the analysis, we also find that APL ultimately decreases linearly in the process of adding edges, and this occurs because the network diameter affects the optimization of APL, which is ignored in the previous studies. In addition, this phenomenon can be used to design more effective methods to optimize APL.

The remainder of the paper is structured as follows. We define structural optimization in terms of minimizing APL in Sect. 2; we propose an optimizing algorithm in Sect. 3; the algorithm is tested on different computer-generated networks in Sect. 4; Sect. 5 gives our conclusions and suggests further work.

2 Related background

Minimizing APL has been the subject of much research. Donetti et al. [35] proposed a new class of networks, named “entangled networks”, which had small APL and large girths. To speed up the convergence in Donetti’s model, Xuan et al. [36] suggested a simulated annealing method to achieve the optimal network by minimizing APL. Keren [37] explored the reduction of APL in binary decision diagrams using a spectral technique based on properties of the Walsh spectrum of a Boolean function and its autocorrelation function. Garijo et al. [38] proposed a novel method to minimize the largest distance between any two points in plane Euclidean network.

In this paper, we focus on how to add edges in a network to get an optimal APL, this process can be applied to real optimization problems such as enterprise mergers or location planning. Specifically, for a connected network $G = (V, E)$ with $n = ||V||$
nodes and $\|E\|$ the initial number of edges, we add $k$ edges ($1 \leq k \leq \frac{n(n-1)}{2} - \|E\|$) to produce a network $G' = (V, E')$ with the minimum APL. Our objective function can be expressed as Eq. (1):

$$\begin{align*}
\min APL \\
s.t. \|E'\| - \|E\| = k
\end{align*}$$  \hspace{1cm} (1)

It should be noted that optimization of APL suffers from combinatorial explosion problem. There can be $\binom{n(n-1)}{2} - \|E\|$ ways of adding $k$ edges to a network with $n$ nodes. Since the time to compute APL is $O(n^3)$ finding the best solution takes time $O(n^{2k+3})$, which is high even for a small size network. This problem can be seen as an amendment to link recommendations, namely to select from the candidates link recommendations that improve the APL, and many related algorithms can be employed to accomplish this [39, 40]. Meyerson and Tagiku [41] first defined this link recommendation as Average Shortest Path Distance Minimization (ASPDM) and proposed an approximation method (ASPDM Algorithm) to solve this problem; this method finds a source vertex $S$, from which there is a lower value of the sum of path lengths to other nodes, and then adds edges from $S$ to $k$ other nodes to give a lower APL. The complexity of this method is $O(n^5)$, which is quite time-saving, and the solution was proved to be at least as good a solution, but there still exists a large distance from the global optimal solution. Upon adding a new edge, the path length of some pairs of nodes will be changed, while the path length of other pairs remains unchanged; Parotsidis et al. [42] proposed the EdgeEffect Algorithm to maximize the reduction of those changed pairs’ lengths. The complexity of the EdgeEffect Algorithm is $O(n^3 \cdot \|E\|)$, which is higher than that of the ASPDM Algorithm, but the solution is more effective. Different variants of these algorithms have been considered, leading to qualitatively similar results [43, 44]. A greedy algorithm adds edges one by one, and has proved to be the most efficient method in optimizing APL [42, 45, 46]. Gozzard et al. [47] suggested two efficient algorithms to minimize the APL length by link addition. He also proposed two algorithms, FirstMinAPL algorithm and SecondMinAPL algorithm, where the FirstMinAPL algorithm perform better in small scale network. The complexity of FirstMinAPL algorithm is $O(n^3) + O(n^2 \cdot \|E\|)$. Papagelis and Manos [48] proposed a novel method named “path screening”, which quickly identifies important shortcuts to guide the augmentation of the graph. And the complexity of the Path Screening Algorithm is $O(n^2 \cdot \log n) + O(n^2 \cdot \|E\|)$, a lower complexity and the solution is also effective. In this paper, we modify this greedy algorithm as follows: a new edge is added at each step; specifically, for the first step, we choose the edge giving the minimum APL from the $\binom{n(n-1)}{2} - \|E\|$ nonexistent edges, and for the second step we choose the best edge from the remaining $\binom{n(n-1)}{2} - \|E\| - 1$ nonexistent edges, and for the $k^{th}$ step we choose the best edge from the remaining $\binom{n(n-1)}{2} - \|E\| - k + 1$ edges. The total time complexity of this greedy algorithm is $O(n^5 \cdot k)$. This algorithm
Network structure optimization for social ... 1465

has difficulty finding the globally optimal solution since it performs a local search. Its solution by adding \(k\) edges depends completely on the solution of adding \(k - 1\) edges. Therefore, a new optimizing algorithm that uses a global search technique is required to find more precise solutions.

3 The algorithm

Since most structural optimizations suffer from the combinatorial explosion problem, evolutionary algorithms (such as a genetic algorithm or simulated annealing) that combine global and local search techniques have been widely used and can be effective in preventing combinatorial explosion [49, 50]. Memetic algorithm is first defined by Moscato and Norman [51]. In their early definition, it were a modification of Genetic Algorithms employing also a local search operator for addressing the Traveling Salesman Problem. Since memetic algorithm were not proposed as specific optimization algorithms, but as a broad class of algorithms inspired by the diffusion of the ideas and composed of multiple existing operators, the community started showing an increasing attention toward these algorithmic structures as a general guideline for addressing specific problems. Memetic algorithm have been successfully applied, in recent years, to solve complex real-world problems and displayed a high performance in a large number of cases [52]. In this section, we propose a memetic algorithm combined with a genetic algorithm and a heuristic local search to minimize the APL.

3.1 Framework

Algorithm 1 gives the framework of our proposed algorithm. We first input the necessary parameters and load the network matrix to be optimized. Then we generate an initial population by the function Initial_Population(). Before the iteration number reaches Imax, or the objective function (APL) remains unchanged for 40 iterations, we repeat a process for finding the best solution. In this repeated process, we first select the parent population for genetic operations by use of the function Tournament_Selection(); then we operate crossover and mutation in the Genetic_Operation() to generate the offspring population; next, we use Local_Search() to find a better solution after the Genetic_Operation(), which can speed up the convergence; Update_Population() sorts the objective function of all chromosomes and selects better performing ones to construct a new population.

The process of memetic algorithm is shown in Fig. 1. First, we evaluate the population’s fitness, i.e. APL. Second, we choose the population by natural selection. Then we execute genetic operation, which contains the mechanism of crossover and mutation. Finally, combined with local search, we find the better result after continuous iteration executing of this process.
Algorithm 1 Framework of our algorithm

Require: maximum number of iterations: $I_{\text{max}}$; size of population: $S_{\text{pop}}$; size of mating pool: $S_{\text{pool}}$; size of tournament: $S_{\text{tour}}$; probability of crossover: $P_c$; probability of mutation: $P_m$.
Ensure: the position of added edges; the updating network’s matrix; APL of the updating network.

1: Load: the initial network adjacency matrix;
2: $P$ ← Initial_Population($S_{\text{pop}}$);
3: repeat
4: $P_{\text{parent}}$ ← Tournament_Selection($P$, $S_{\text{pool}}$, $S_{\text{tour}}$);
5: $P_{\text{offspring}}$ ← Genetic_Operation($P_{\text{parent}}$, $P_c$, $P_m$);
6: $P_{\text{offspring}}'$ ← Local_Search($P_{\text{offspring}}$);
7: $P$ ← Update_Population($P$, $P_{\text{offspring}}'$);
8: until Termination.

Fig. 1 Illustration of the proposed algorithm

3.2 Representation and initialization

Our target is to determine in which positions we should add new edges to reduce the APL. To this end, we first find the positions of those pairs of nodes that are not connected (nonexistent edges). Then we give a set of sequence numbers to these positions; thus our solutions can be encoded as chromosomes consisting of these corresponding sequence numbers. An example is shown in Fig. 2, where the second entry of the chromosome changes from 2 to 3, so the added edge between Node 1 and Node 4 changes to the edge between Node 2 and Node 4.

At initialization, we generate a population and randomly assign a set of numbers to every element of the chromosomes in the population. To speed up convergence and retain population diversity, we add two new sub-populations: (1) each element of the chromosomes (i.e., the added edge) connects the two highest degree nodes; (2) each element of the chromosomes connects the highest degree node and the lowest degree node. In other words, the initial population can be divided into three parts: solutions with random assignment, assortative connecting assignment and disassortative connecting assignment.
Network structure optimization for social ...  

Fig. 2 Illustration of the representation. The upper left table is the position of disconnected edges represented by a set of sequence numbers. Upper right is an illustration of chromosomes. The lower part is the topological graph corresponding to the chromosomes. The solid lines represent the existing edges, the dashed lines represent the added edges, and the number near the node is the sequence number of the node.

### 3.3 Genetic operation

The genetic operation consists of two parts: crossover and mutation.

In the crossover procedure, we randomly select two chromosomes from the parent population with probability $P_c$ and generate two offspring chromosomes. The common elements of the two offspring chromosomes remain unchanged. For the remaining elements that differ between the chromosomes, we give an ordering, and for every unshared element, crossover operates with probability $a$; if $a < 0.5$, the element remains unchanged; and if $a \geq 0.5$, the corresponding elements (with the same ordering number) of the two offspring are swapped. Figure 3 gives an illustration of crossover. The sequence numbers 11 and 7 are elements common to both offspring chromosomes, so they do not need to exchange. For the other four elements, assuming that the probability $a$ equals 0.4, 0.8, 0.7, and 0.2, respectively, the first and fourth unshared element remain unchanged since $a < 0.5$, and the second and third unshared elements are swapped between the two chromosomes since $a \geq 0.5$.

In the mutation procedure, we randomly select a chromosome with probability $P_m$, and for a selected element of the chromosome, we randomly choose a different sequence number to replace that element.
Fig. 3 Illustration of crossover. The first line is the generalized probability $a$. The upper two chromosomes are parental chromosomes while the lower two are offspring chromosomes. The small numbers near the elements are the ordering numbers of the unshared elements, and the elements with the same ordering number are corresponding elements. We check every unshared element of the two parent chromosomes; if the generalized $a < 0.5$, the element remains unchanged in the offspring chromosomes, while if $a \geq 0.5$, the corresponding elements are swapped.

3.4 Local search

By incorporating some priori knowledge, a local search can efficiently reduce useless explorations and speed up the convergence of genetic algorithms [53]. In this section, we employ Path Length Learning and Neighbor Learning, respectively.

Path Length Learning is shown in Algorithm 2. We first update the network matrix by decoding the chromosome. Then, we compute the degrees of nodes connected by the added edges and delete the edge with the minimum sum degree of those pairs of nodes. Finally, we add a new edge to the pair of nodes that has the longest path length. This learning technique helps produce a better result.

Algorithm 2 Path Length Learning

Require: the chromosome $X_{offspring}$; the adjacency matrix of the initial network $M$.
Ensure: $X_{offspring2}$.
1: $X_{offspring2} = X_{offspring}$;
2: Update the matrix $M$ by decoding $X_{offspring}$;
3: Find the position of the pair of nodes with longest path length of the updated matrix (the position is represented by the sequence number $g$);
4: Check every element of $X_{offspring}$ and compute the degrees of nodes connected by the added edge decoded by the element;
5: Find the element $i$ with the minimum sum value of nodes pair degrees;
6: $X_{offspring2}(i) = g$;

In Neighbor Learning, illustrated in Algorithm 3, for each element in the offspring chromosome, we choose the value of its best neighbor (a neighbor of an added edge is one of the nonexistent edges sharing one of the same nodes) to judge whether this choice produces a better result. If the new choice decreases APL, then we accept the new element, and the result improves.
Algorithm 3 Neighbor Learning

Require: the chromosome \(X_{\text{offspring}}\); the number of added edges \(k\); the adjacency matrix of the initial network \(M\).

Ensure: \(X_{\text{offspring}}\).

1: \(X_{\text{offspring}} = X_{\text{offspring}}\)
2: Update the matrix \(M\) by decoding \(X_{\text{offspring}}\);
3: for \(i = 1; i \leq k; i + + \) do
4: for \(m = 1; \) number of \(X_{\text{offspring}}\)'s neighbor; \(m + + \) do
5: \(X_{\text{offspring}}(i) = X_{\text{offspring}}(i)\)’s neighbor;
6: end for
7: find \(X_{\text{offspring}}(i)\)’s neighbor \(q\) with minimum \(\text{APL}(X_{\text{offspring}})\);
8: \(X_{\text{offspring}}(i) = q\);
9: if \(\text{APL}(X_{\text{offspring}}) < \text{APL}(X_{\text{offspring}})\) then
10: \(\text{APL}(X_{\text{offspring}}) = \text{APL}(X_{\text{offspring}})\)
11: end if
12: end for

3.5 Complexity analysis

If we add \(k\) edges to a network of size \(n\), the time complexity of the proposed algorithm can be estimated as follows: At each iteration, we first need to carry out the crossover \(S_{\text{pool}} \div 2\) times and mutation \(S_{\text{pool}}\) times, where \(S_{\text{pool}}\) is the size of the mating pool for the genetic operation. Since the time complexity of computing the APL is \(O(n^3)\), the genetic operation will cost \(O(S_{\text{pool}}(k + n^3))\). Second, on performing the path learning, updating the matrix costs time \(O(k)\), finding the added edge with the minimum node degrees costs \(O(kn)\), finding the nonexistent edge with longest path length costs \(O(n^3)\), so the path learning will cost \(O(n^3 + kn + k)\). Third, to perform neighbor learning, we consider \(n\) neighbors of each added edge, and it will cost \(O(kn^4)\) to compute the APL of all those neighbors. As a result, the complexity of our algorithm is \(O(kn^4)\) at each iteration. If the network size \(n\) is big enough, the complexity of the Memetic Algorithm is less than that of Greedy Algorithm.

4 Experiments

In this section, we test our algorithm on different computer-generated networks and real-world networks.

4.1 Computer-generated networks

The experiments were carried out on a 2.40 GHz CPU, 4.00 GB Memory and Windows 10 operating system using MATLAB to execute the procedure. Since the algorithm we proposed is not parameter-free, we need to set some values for these parameters in advance, such as the size of population or mating pool. Some of these parameters are set to the fixed values that we found by trial and error in order to ensure that the proposed algorithm has excellent performance. Table 1 shows the parameters used in the experiments.
Table 1: Parameters of the experiments

| Parameter | Meaning                  | Value |
|-----------|--------------------------|-------|
| $I_{\text{max}}$ | Maximum number of iterations | 200   |
| $S_{\text{pop}}$ | Size of population       | 200   |
| $S_{\text{pool}}$ | Size of mating pool      | 100   |
| $S_{\text{tour}}$ | Size of tournament      | 2     |
| $P_c$      | Probability of Crossover | 0.1   |
| $P_m$      | Probability of Mutation  | 0.9   |

Table 2: Full names of the abbr

| Abbr          | Full name                                                        |
|---------------|-----------------------------------------------------------------|
| APL           | Average Path Length                                             |
| Big–Big       | Big–Big assortative connecting method                           |
| Big–Small     | Big–Small disassortative connecting method                       |
| ASPDM algorithm | Average shortest path distance minimization algorithm           |
| FirstMinAPL algorithm | First minimize average path length algorithm                   |

We first ran our algorithm on random networks with different network sizes, and compared it with five other methods: (1) add $k$ edges at one time, where each of the added edges connects the two highest degree nodes (Denoted as Big–Big, which is an assortative connecting method); (2) add $k$ edges at one time, where each of the added edges connects the highest degree node and the lowest degree node (denoted as Big–Small, which is a disassortative connecting method); (3) add $k$ edges one by one, with each of the added edges giving the minimum APL as described in Sect. 2 (denoted as the Greedy Algorithm). We set $k$ from 1 to 50; (4) ASPDM Algorithm proposed by Meyerson and Tagiku [41]; (5) EdgeEffect Algorithm proposed by Parotsidis et al. [42]; (6) FirstMinAPL Algorithm proposed by Gozzard et al. [47]; (7) Path Screening Algorithm proposed by Papagelis and Manos [48]. Since evolutionary algorithms have inherent randomness, many of them, including our proposed algorithm, may fall into a local minimum in some cases. We ran the above-mentioned algorithms ten times. For convenience, we present Table 2 gives that the abbr. and their full names that appear in the article.

For the results of the mean APL for these ten runs, the Greedy Algorithm can find the best solutions, while the Memetic Algorithm can find better solutions than other methods. For finding the minimum APL for these ten runs, the Memetic Algorithm performs best. In order to make the experimental results easier to comprehend and for convenience, we list only the results of the minimum APL for the ten runs of these algorithms. We set the number of added edges to 10; Fig. 4 shows the optimizing result with different network sizes. Compared with the other methods, our algorithm and Greedy Algorithm can always find lower-APL solutions with different network sizes.

Then we ran our algorithm on three different networks (random network, regular network and scale-free network of size 30) with different numbers of added edges...
to explore the effect of each added edges. Figure 5 shows the optimizing results for different networks. Since the degrees of regular network nodes are all equal, the results of Big–Big and Big–Small are the same; here we record just the result of Big–Big on the regular network. Since ASPDM Algorithm adds edges from a source vertex S to other nodes, when all nodes are connected to S in the process of adding edges, then no more edges will be added as shown in Fig. 5.

Compared with the other five methods, our algorithm can find lower-APL solutions except for a few solutions of the Greedy Algorithm and EdgeEffect Algorithm which do better than the Memetic Algorithm on the regular network (this is because our priori knowledge is less useful on regular networks). We conclude that in most cases, our proposed algorithm can effectively minimize APL. Further, we find that the solutions of Big–Small are close to the optimizing solutions in Fig. 5a and c. In a sense, disassortative connecting can find good solutions for minimizing APL. Moreover, we find when the number of added edges increases to a threshold value, APL begins to decline linearly no matter which method is used. This phenomenon can be explained by the feature of Critical Diameters [54, 55]. Specifically, when we add enough edges to the network, the longest path length of the network becomes 2 (i.e. the network diameter is 2), in which case, if we randomly add a new edge between a pair of nodes, we can only change the path lengths of these two nodes to 1, and thus the APL can be reduced by just $2 \frac{2}{n(n-1)}$ for each step, which constitutes a linear decline.

In this section, we analyze the structure of the optimal networks to determine the effect of optimizing APL on different networks. The assortative index $r$ and cluster coefficient $c$ are important indices for measuring network characteristics [56]. The assortative index $r$ measures the characteristic of assortative (or disassortative) mixing, and can be formulated as:
Fig. 5  Optimizing results of APL for three different networks
Network structure optimization for social ... 1473

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

where \( j_i, k_i \) are the degrees of the vertices at the ends of the \( i^{th} \) edge [57]. The cluster coefficient \( c \) measures average local density \( (c = \frac{\sum_v CC_v}{n}) \), where \( CC_v = \frac{|E(\Gamma_v)|}{k_v(k_v - 1)}, |E(\Gamma_v)| \) is the number of edges in the relational neighborhood of node \( v \), and \( k_v(k_v - 1) \) is the number of possible edges in the relational neighborhood of node \( v \). As APL decreases monotonically, the Memetic Algorithm results for \( r \) and \( c \) fluctuate because there can be more than one best solution with different connection rules that minimizes APL when more edges are added as shown in Figs. 6, 7 and 8. Further, the variation in \( r \) and \( c \) appears to be quite different in the three different network models.

For the random network experiments, as shown in Fig. 6, \( r \) does not show a regular trend, but the optimal networks turn out to be disassortative since most \( r \) values are negative, while \( c \) shows an increasing trend as more edges are added to the initial network, which means that a random network is gradually evolving into a small-world network. This figure shows two interesting phenomena. First, disassortative connection for adding edges may contribute more to the decrease of APL. Second, decreasing APL in random networks can simultaneously increase the cluster coefficient, which is helpful in constructing small-world networks.

For the regular network experiments, \( r \) and \( c \) both decline with the first few edges added as shown in Fig. 7. Thus, we can conclude that when adding a small number of edges, a disassortative connection is a good choice to decrease APL. Since a regular network initially has a high cluster coefficient, \( c \) will decrease when we add edges in a disassortative way. However, when we add a larger number of edges to a regular network, the disassortative connection becomes less effective. This conclusion is different from that for the random network.

For optimization of the scale-free network as shown in Fig. 8, all signs of \( r \) are negative, which indicates that optimal networks are all disassortative. However, \( c \) increases as new edges are added, because adding more edges to a scale-free network, especially linking high-degree nodes with other nodes, has a high probability of constructing more triangles, which will result in a higher \( c \). The property of connections in scale-free networks is similar to that in random networks, and the disassortative feature of the connections is more obvious in scale-free networks. Here, the assortative index \( r \) shows great difference between greedy algorithm and memetic algorithm, even if both of them perform excellent in optimizing APL, as shown in Fig. 7a. This is because different algorithm has different edge addition strategy, even if all of them give the same low APL, and the edges they add may be completely different. And it is confirmed when compare Fig. 9b and c, where the added edges are not same. Furthermore, in Fig. 5c, when the number of added edges exceeds 10, the network enters the state of critical diameter, and the changes of APL are basically similar. Then the difference of edge addition strategy between these two algorithms will lead to difference in \( r \). Specifically, when the network reaches the critical diameter, the strategy
of greedy algorithm tends to adding edges randomly, but memetic algorithm tends to link between nodes with large degree value in scale-free network due to the mechanism of path length learning. Therefore, when the number of added edges reaches a certain level, the algorithm result is similar to Big–Big, and the difference with greedy algorithm becomes larger.
4.2 Real-world networks

After the experiment on computer-generated networks, we find that the greedy algorithm and our proposed memetic algorithm perform best. Further, we compare the performance of the two algorithms on three real-world networks: (1) Dolphins network, which is a social network of bottlenose dolphins \[58\]; (2) Terrorists network, which contains contacts between suspected terrorists involved in the train bombing of Madrid on March 11, 2004 as reconstructed from newspapers \[59\]; (3) Windsurfers network, which contains interactions between 28 Grévy’s zebras in Kenya \[60\]. In all of these networks, we ignore the weight of the edges for convenience. On each net-
work, we compare the results of adding 5 edges, 10 edges and 15 edges respectively, Table 3 shows the results.

From Table 3, the results show that the memetic algorithm perform equally even better than that of greedy algorithm, which proves the effective of memetic algorithm on real-world networks.

We further observe the difference of adding edges between the two algorithms on the network topology. Figure 9 show the topology of $k = 10$ added edges on Dolphins network. Fig. 9b and c show the topology of $k = 10$ added edges via the greedy algorithm and memetic algorithm respectively. Overall, both two algorithms focus on the edges between big communities and the nodes whose degree is higher, which proves such a choice is indeed an effective strategy. More specifically, at this point,
Table 3  Comparison between the Greedy algorithm and the Memetic algorithm

| Network          | Nodes | Edges | Algorithm       | Number of added edges |
|------------------|-------|-------|-----------------|-----------------------|
|                  |       |       | Greedy algorithm| 0  | 5  | 10 | 15 |
| Dolphins network | 62    | 159   | Memetic algorithm| 3.36 | 2.87 | 2.70 | 2.58 |
| Terrorists network | 64    | 243   | Greedy algorithm| 2.69 | 2.46 | 2.34 | 2.23 |
| Windsurfers network | 27    | 111   | Memetic algorithm| 1.67 | 1.63 | 1.62 | 1.61 |

Fig. 9  The figures show the topology of $k = 10$ added edges on Dolphins network. a shows the topology of the initial network. b–c hide the initial edges and show the added edges. b shows the topology of $k = 10$ added edges via the greedy algorithm. c shows the topology of added edges $k = 10$ via the memetic algorithm.

memetic algorithm’s performance is more obvious because some edges in Fig. 9b existing within the community.

5 Conclusion

In this paper, we have focused on the optimization of a network by minimizing APL by adding edges, and we propose a memetic algorithm to find optimal solutions. The experimental results show that our algorithm can minimize APL efficiently. In the analysis, we also find that APL will ultimately decrease linearly in the process of adding edges, which is directly affected by the network diameter. Further, we compute the assortative index and cluster coefficient for optimal networks with different initial network structures and find that these two properties of optimal solutions may be quite different. We optimize only one indicator, APL, in this paper, but in many cases we need to optimize more than one feature of networks, for example minimizing APL.
and maximizing the cluster coefficient at the same time. This multi-objective structural optimization should be explored in future work.

**Funding** This work is supported by National Natural Science Foundation of China(72104194), China Postdoctoral Science Foundation(2021M700107) and Fundamental Research Funds for the Central Universities(SK2021003).

**Declarations**

**Conflict of interest** The authors declare that they have no conflict of interest.

**References**

1. Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proc Natl Acad Sci 99(12):7821–7826. https://doi.org/10.1073/pnas.122653799
2. Karrer B, Newman MEJ (2011) Stochastic blockmodels and community structure in networks. Phys Rev E Stat Nonlinear Soft Matter Phys 83:016107. https://doi.org/10.1103/PhysRevE.83.016107
3. Gao C, Ma Z, Zhang AY, Zhou HH (2018) Community detection in degree-corrected block models. Ann Stat 46(5):2153–2185. https://doi.org/10.1214/17-AOS1615
4. Wang J, Zhang J, Liu B, Zhu J, Guo J (2021) Fast network community detection with profile-pseudo likelihood methods. J Am Stat Assoc 1–14. https://doi.org/10.1080/01621459.2021.1996378
5. Watts DJ, Strogatz SH (1998) Collective dynamics of ‘small-world’ networks. Nature 393(6684):440–442. https://doi.org/10.1038/30918
6. Barabási A-L, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512. https://doi.org/10.1126/science.286.5439.509
7. Albert R, Barabási A-L (2002) Statistical mechanics of complex networks. Rev Mod Phys 74(1):47–97. https://doi.org/10.1103/revmodphys.74.47
8. Wang W, Liu Q-H, Liang J, Hu Y, Zhou T (2019) Coevolution spreading in complex networks. Phys Rep 820:1–51
9. Li A, Zhou L, Su Q, Cornelius SP, Liu Y-Y, Wang L, Levin SA (2020) Evolution of cooperation on temporal networks. Nat Commun 11(1):2259. https://doi.org/10.1038/s41467-020-16088-w
10. Wang Y, Zeng A, Di Z, Fan Y (2011) Enhancing synchronization in growing networks. EPL 96(5). https://doi.org/10.1209/0295-5075/96/58007
11. Shi D, Chen G, Thong WWK, Yan X (2013) Searching for optimal network topology with best possible synchronizability. IEEE Circuits Syst Mag 13(1):66–75. https://doi.org/10.1109/msacm.2012.2237145
12. Sun Y, Du H, Gong M, Ma L, Wang S (2014) Fast computing global structural balance in signed networks based on memetic algorithm. Physica A 415:261–272. https://doi.org/10.1016/j.physa.2014.07.071
13. Hwang SW, Kweon SJ, Ventura JA (2020) An alternative fuel refueling station location model considering detour traffic flows on a highway road system. J Adv Transp 2020. https://doi.org/10.1155/2020/9473831
14. Davoodi M, Ghaffari M (2021) Shortest path problem on uncertain networks: an efficient two phases approach. Comput Indus Eng 157. https://doi.org/10.1016/j.cie.2021.107302
15. Wang T, Cheng H, Wang X (2020) A link addition method based on uniformity of node degree in interdependent power grids and communication networks. Physica A 560:125112
16. Kulakovski K, Gawroński P, Gronke P (2005) The heider balance: a continuous approach. Int J Mod Phys C 16(05):707–716. https://doi.org/10.1142/s012918310500742x
17. Antal T, Krapivsky PL, Redner S (2005) Dynamics of social balance on networks. Phys Rev E Stat Nonlinear Soft Matter Phys 72(3 Pt 2):036121. https://doi.org/10.1103/PhysRevE.72.036121
18. Facchetti G, Iacono G, Altafini C (2011) Computing global structural balance in large-scale signed social networks. Proc Natl Acad Sci USA 108(52):20953–8. https://doi.org/10.1073/pnas.1109521108
19. Yan X, Li C, Zhang L, Hu Y (2016) A new method optimizing the subgraph centrality of large networks. Physica A 444:373–387
20. Barahona F (1982) On the computational complexity of ising spin glass models. J Phys A Math Gen 15(10):3241–3253. https://doi.org/10.1088/0305-4470/15/10/028
21. Wang S, Gong M, Du H, Ma L, Miao Q, Du W (2016) Optimizing dynamical changes of structural balance in signed network based on memetic algorithm. Social Netw 44:64–73. https://doi.org/10.1016/j.socnet.2015.06.004
22. Yang X, Xiao-hua W (2007) Decentralized small-world optimization strategy. J Suzhou Univ 27(3):41–46. https://doi.org/10.3969/j.issn.1673-047X.2007.03.010
23. Schleich J, Danoy G, Dorrorsoro B, Bouvry P (2014) Optimising small-world properties in vanets: centralised and distributed overlay approaches. Appl Soft Comput 21:637–646. https://doi.org/10.1016/j.asoc.2014.03.045
24. Du H, Fan J, He X, Feldman MW (2018) A genetic simulated annealing algorithm to optimize the small-world network generating process. Complexity 2018:1–12. https://doi.org/10.1155/2018/1453898
25. Alstott J, Klymko C, Pyzza PB, Radcliffe M (2019) Local rewiring algorithms to increase clustering and grow a small world. J Compl Netw 7(4):564–584. https://doi.org/10.1093/comnet/cny032
26. Yu F, Li Y, Wu T-J (2010) A temporal ant colony optimization approach to the shortest path problem in dynamic scale-free networks. Physica A 389(3):629–636. https://doi.org/10.1016/j.physa.2009.10.005
27. Ma H, Zeng AP (2003) Reconstruction of metabolic networks from genome data and analysis of their global structure for various organisms. Bioinformatics 19(2):270–277. https://doi.org/10.1093/bioinformatics/19.2.270
28. Wang B, Tang H, Guo C, Xiu Z, Zhou T (2006) Optimization of network structure to random failures. Physica A 368(2):607–614. https://doi.org/10.1016/j.physa.2005.12.050
29. Ashton DJ, Jarrett TC, Johnson NF (2005) Effect of congestion costs on shortest paths through complex networks. Phys Rev Lett 94(5):058701. https://doi.org/10.1103/PhysRevLett.94.058701
30. Lago-Fernández LF, Huerta R, Corbacho F, Sigüenza JA (2000) Fast response and temporal coherent oscillations in small-world networks. Phys Rev Lett 84(12):2758–2761. https://doi.org/10.1103/physrevlett.84.2758
31. Gade PM, Hu C-K (2000) Synchronous chaos in coupled map lattices with small-world interactions. Phys Rev E 62(5):6409–6413. https://doi.org/10.1103/physreve.62.6409
32. Jost J, Joy MP (2001) Spectral properties and synchronization in coupled map lattices. Phys Rev E 65(1). https://doi.org/10.1103/physreve.65.016201
33. Hong H, Choi MY, Kim BJ (2002) Synchronization on small-world networks. Phys Rev E Stat Nonlinear Soft Matter Phys 65(2 Pt 2):026139. https://doi.org/10.1103/PhysRevE.65.026139
34. Barahona M, Pecora LM (2002) Synchronization in small-world systems. Phys Rev Lett 89(5). https://doi.org/10.1103/physrevlett.89.054101
35. Donetti L, Hurtado PI, Munoz MA (2005) Entangled networks, synchronization, and optimal network topology. Phys Rev Lett 95(18):188701. https://doi.org/10.1103/PhysRevLett.95.188701
36. Xuan Q, Li Y, Wu T-J (2009) Optimal symmetric networks in terms of minimizing average shortest path length and their sub-optimal growth model. Physica A 388(7):1257–1267. https://doi.org/10.1016/j.physa.2008.12.020
37. Keren O (2008) Reduction of average path length in binary decision diagrams by spectral methods. IEEE Trans Comput 57(4):520–531. https://doi.org/10.1109/tc.2007.70811
38. Garijo D, Márquez A, Rodríguez N, Silveira RI (2019) Computing optimal shortcuts in scale-free networks. Eur J Oper Res 279(1):26–37. https://doi.org/10.1016/j.ejor.2019.05.018
39. Adamic LA, Adar E (2003) Friends and neighbors on the web. Social Netw 25(3):211–230. https://doi.org/10.1016/s0378-8733(03)00009-1
40. Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. J Am Soc Inform Sci Technol 58(7):1019–1031. https://doi.org/10.1002/asi.20591
41. Meyerson A, Tagiku B (2009) Minimizing average shortest path distances via shortcut edge addition. Springer, pp 272–285. https://doi.org/10.1007/978-3-642-03685-9_21
42. Parotsidis N, Pitoura E, Tsaparas P (2015) Selecting shortcuts for a smaller world. In: Proceedings of society for industrial and applied mathematics, pp 28–36. https://doi.org/10.1137/1.9781611974010.4
43. Papagelis M, Bonchi F, Gionis A (2011) Suggesting ghost edges for a smaller world. Assoc Comput Mach. https://doi.org/10.1145/2063576.2063952
44. Lazaridou K, Semertzidis K, Pitoura E, Tsaparas P (2015) Identifying converging pairs of nodes on a budget
45. Lee SH, Holme P (2013) A greedy-navigator approach to navigable city plans. Eur Phys J Special Topics 215(1):135–144. https://doi.org/10.1140/epjst/e2013-01720-8
46. Papagelis M (2015) Refining social graph connectivity via shortcut edge addition. ACM Trans Knowl Discov Data 10(2):1–35. https://doi.org/10.1145/2757281
47. Gozzard A, Ward M, Datta A (2018) Converting a network into a small-world network: Fast algorithms for minimizing average path length through link addition. Inf Sci 422:282–289
48. Papagelis M (2015) Refining social graph connectivity via shortcut edge addition. ACM Trans Knowl Discov Data 10(2). https://doi.org/10.1145/2757281
49. Brusco M, Steinley D (2011) A tabu-search heuristic for deterministic two-mode blockmodeling of binary network matrices. Psychometrika 76(4):612–33. https://doi.org/10.1007/s11336-011-9221-9
50. Ross BJ, Zuviria E (2006) Evolving dynamic bayesian networks with multi-objective genetic algorithms. Appl Intell 26(1):13–23. https://doi.org/10.1007/s10489-006-0002-6
51. Norman MG, Moscato P et al (1991) A competitive and cooperative approach to complex combinatorial search. In: Proceedings of the 20th informatics and operations research meeting. Citeseer, pp 3–15
52. Neri F, Cotta C (2012) Memetic algorithms and memetic computing optimization: a literature review. Swarm Evol Comput 2:1–14. https://doi.org/10.1016/j.swevo.2011.11.003
53. Moscato P (1989) On evolution, search, optimization, genetic algorithms and martial arts-towards memetic algorithms. Caltech Concurrent Computation Program
54. Du H, He X, Du W, Feldman MW (2017) Optimization of the critical diameter and average path length of social networks. Complexity 2017:1–11. https://doi.org/10.1155/2017/3203615
55. Du H, Wang J, He X, Du W (2019) A memetic algorithm to optimize critical diameter. Swarm Evol Comput 47:56–65. https://doi.org/10.1016/j.swevo.2017.10.001
56. Niu J, Wang L (2016) Structural properties and generative model of non-giant connected components in social networks. Science China Inform Sci 59(12). https://doi.org/10.1007/s11432-015-0790-x
57. Newman MEJ (2002) Assortative mixing in networks. Phys Rev Lett 89(20). https://doi.org/10.1103/physrevlett.89.208701
58. Lusseau D, Schneider K, Boisseau OJ, Haase P, Slooten E, Dawson SM (2003) The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations: can geographic isolation explain this unique trait? Behav Ecol Sociobiol 54(4):396–405
59. Hayes B (2006) Connecting the dots. can the tools of graph theory and social-network studies unravel the next big plot? Am Sci 94(5):400–404
60. Sundaresan SR, Fischhoff IR, Dushoff J, Rubenstein DI (2007) Network metrics reveal differences in social organization between two fission-fusion species. Grevy’s zebra and onager. Oecologia 151(1):140–149

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.