Hindawi
Complexity
Volume 2017, Article ID 3203615, 11 pages
https://doi.org/10.1155/2017/3203615

WILEY | Hindawi

Research Article

Optimization of the Critical Diameter and Average Path Length of Social Networks

Haifeng Du,[1] Xiaochen He,[1] Wei Du,[1] and Marcus W. Feldman[1,2]

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

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

Received 29 November 2016; Accepted 13 February 2017; Published 28 March 2017

Academic Editor: Katarzyna Musial

Optimizing average path length (APL) by adding shortcut edges has been widely discussed in connection with social networks, but the relationship between network diameter and APL is generally ignored in the dynamic optimization of APL. In this paper, we analyze this relationship and transform the problem of optimizing APL into the problem of decreasing diameter to 2. We propose a mathematic model based on a memetic algorithm. Experimental results show that our algorithm can efficiently solve this problem as well as optimize APL.

1. Introduction

Following the introduction of models for small-world and scale-free networks, much research has been devoted to analyzing network characteristics [1–5]. In particular, there has been a focus on finding indices to quantify features of network structure such as structural entropy, robustness, or modularity [6–8]. These indices play an important role in measuring specific performance aspects of networks, and optimizing them can help to improve network performance.

Average path length (APL), the average shortest distance between all nodes in a network, is not only a measurement of static characteristics such as connectivity and robustness but also an important control variable in dynamic processes, such as the spread of diseases or target searching [9–11]. Optimizing APL has also attracted attention in the field of structural optimization. Decreasing APL by adjusting nodes or edges can effectively enhance the transfer efficiency and synchronization ability [12–17]. In addition, optimization of APL has also been widely used in urban planning and site selection [14, 18, 19]. Xuan et al. [20] proposed a simulated annealing model to optimize APL in order to speed up convergence. Keren [21] employed a spectral technique to reduce APL in binary decision diagrams.

In order to optimize APL, many scholars focus on adding a given number of edges to produce the largest decrease in APL. These added edges are called "shortcut" edges and the problem of finding the best set of shortcut edges is defined as the "shortcut-selection" problem [22]. A series of methods have been proposed to solve this problem. Meyerson and Tagiku proposed an approximation method, which involved finding a source node and then connecting k other nodes to this node to decrease APL [22]. Parotsidis et al. analyzed the exact effect of a single edge insertion on APL and proposed the EdgeEffect Algorithm to maximize the effect of edge insertion [23]. A greedy algorithm, which adds edges one by one and which makes the maximum reduction of APL for each added edge, has proved to be efficient [24, 25]. These methods have solved the shortcut-selection problem to some extent. However, a common phenomenon has been ignored in the process of adding edges. In experiments to optimize APL by adding edges, we find that no matter which method is used to add edges, there always exists a turning point at which APL begins to decrease linearly as more edges are added. This phenomenon can be related to the network diameter.

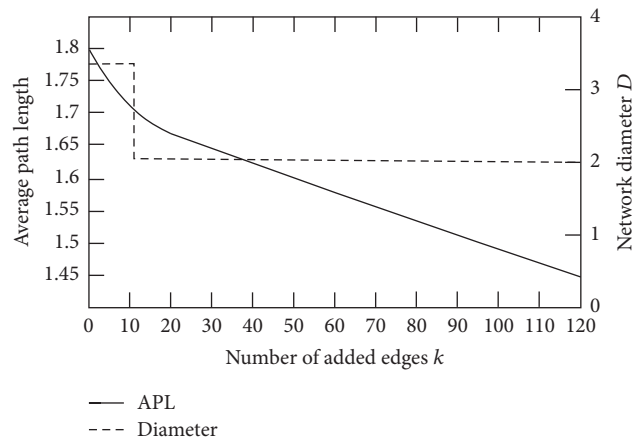


Figure 1: The value of APL and network diameter D as the number of added edges increases.

In this paper, we define the network diameter at the turning point as the "critical diameter," and analyze both this critical diameter and APL in the process of adding edges. We transform the problem of optimizing APL into the problem of optimizing the critical diameter. Specifically, we focus on adding the minimum number of shortcut edges to make the network diameter decrease to 2. Research on predicting missing links has attracted much attention in recent years, the algorithms of which can extract missing information or identify spurious interactions [26–29]. Gao et al. analyzed the feature of predicted network, and they found the network diameter and APL shows a negative linear relation to all of the tested prediction methods [29]. Therefore, our research can also provide some a priori knowledge in designing the method of link prediction. In the next section, we introduce the critical diameter and explore the special relationship between critical diameter and APL; the algorithm for optimizing the critical diameter is proposed in Section 3; Section 4 gives results of testing our method on generated networks; our conclusions and further work are presented in Section 5.

2. Critical Diameter and APL

Network diameter, the maximum path length for all pairs of nodes, is closely related to APL; they both contain information about connectivity and transfer efficiency [30, 31]. Imase and Itoh gave the inequalities $D/2 \leq \text{APL} \leq D$ to describe the static relationship between network diameter, D, and APL [32]. In the dynamic process of adding shortcut edges, there exists a turning point, as shown in Figure 1. APL declines nonlinearly with the number of added edges k until a turning point and then decreases linearly as k increases further. We compute the path length between every pair of nodes and find that the longest path length of the network is larger than 2 before k reaches the turning point (i.e., the network diameter $D > 2$); when k reaches the turning point, the diameter equals 2 ($D = 2$). This is because if $D = 2$, a new added edge between a pair of nodes can only change the path lengths between these two nodes from 2 to 1 but cannot change the path lengths of other pairs of nodes, and the APL can be reduced by just $2/n(n-1)$ for each added edge, which constitutes a linear decline.

In fact, the APL can be computed when the network diameter declines to 2. For a network $G = (V, E)$ with n nodes and m edges, when the diameter equals 2, the path length of every pair of nodes will be equal to or less than 2. If we add k edges to the network, the number of pairs of nodes whose path length equals 1 will be $m + k$ and the number of pairs of nodes whose path length equals 2 will be $n(n-1)/2 - m - k$. Therefore, the value of APL achieved by adding k edges with the network diameter $D = 2$ is

$$\begin{aligned} \text{APL} &= \frac{(m+k) + (n(n-1)/2 - m - k) \times 2}{n(n-1)/2} \\ &= 2 - \frac{2(m+k)}{n(n-1)}. \end{aligned} \tag{1}$$

In the process of adding edges, APL will ultimately decrease linearly and will become equal to the term on the right of (1). Figure 2 shows the results of 20 simulations adding edges randomly to decrease APL; the maximum, mean, and minimum APL of these 20 runs are shown. The three curves become overlapping and linear when the number of added edges becomes large enough. The curve of the minimum APL becomes linear earliest, while the curve of maximum APL is the last to become linear.

In this case, if we attempt to minimize APL by adding a large number of shortcut edges, we can find a solution for which adding a small number of edges has decreased the diameter to 2 and add the remaining edges randomly. Therefore, the problem of optimizing APL can be transformed into the problem of finding shortcut edges that quickly decrease the diameter to 2.

Here, we propose a formal definition for the diameter when APL begins to decline linearly.

Definition 1. In adding edges to a network, the network diameter declines to 2. The network diameter in this case is defined as the "critical diameter," denoted as D_c.

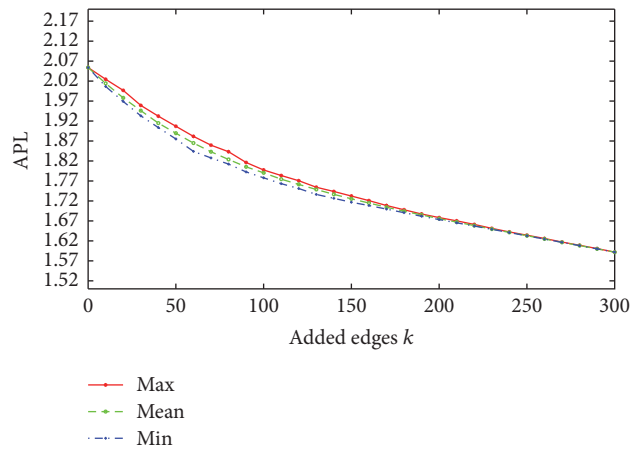
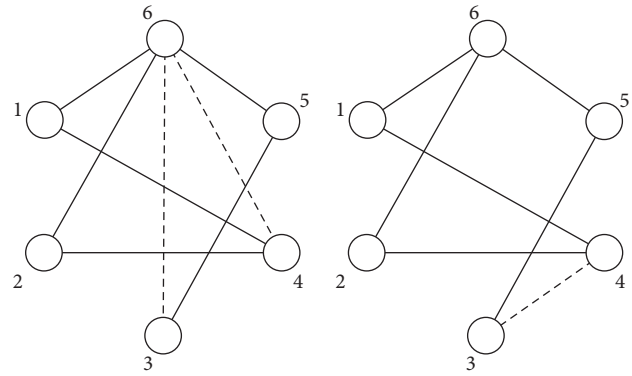


Figure 2: The decrease in APL caused by randomly adding edges.



(a) HDN (b) Other edging method

Figure 3: Different methods to optimize network diameter. The solid lines represent edges of the initial network, while the dashed lines represent added edges.

If we add k shortcut edges to make the network diameter become D_c, the set of these k shortcut edges must be the most optimal solution for minimizing APL by adding k edges. If there exists a solution which can make APL lower by adding another set of k edges, the number of pairs of nodes whose path length equals 1 must be bigger than $m + k$, which cannot be realized by adding only k edges. This kind of relationship between APL and D_c suggests that if we can minimize the number of added edges k to reduce the diameter to D_c, the APL can efficiently decrease to its lowest level.

In this paper, we focus on how to decrease the diameter to 2 by adding the minimum number of edges; this problem is defined as "optimizing the critical diameter." The objective function can be formulated as

$$\begin{aligned} \min \quad & k \\ \text{s.t.} \quad & \forall i \neq j, \quad d_{ij} \leq 2, \end{aligned} \tag{2}$$

where k represents the number of added edges and d_{ij} represents the path length between node i and node j.

It should be noted that the problem of optimizing the critical diameter is an NP-hard problem. Given a connected network $G = (V, E)$ with n nodes and m edges, there can be $\binom{n(n-1)/2-m}{k}$ ways of adding k shortcut edges. Since computing the network diameter costs at least $O(n^3)$, finding the best set of shortcut edges requires $O(n^{2k+3})$, which is high even for a small network.

An efficient way to optimize D_c is to establish connections between the highest-degree node and the rest of nodes, which adds $n - 1 - k_{\max}$ edges (n is the number of nodes and $k_{\max}$ is the highest degree of the network). Then the network will definitely become a Star Network with the highest-degree node at the center. We call this method "HDN" (connecting to the highest-degree node).

However, HDN fails to generate the globally optimal solution. As shown in Figure 3, node 6 is the highest-degree node of the network. To decrease the diameter to 2 by HDN, we should add two edges connecting nodes 3 and 6 and nodes 4 and 6; but if we establish a connection between nodes 3 and 4, the network diameter can also become D_c. Thus we need to design a more efficient method to decrease the network diameter to 2.

(1) Input: the maximum iteration number: $I_{\max}$; population size: S_{pop}; mating pool size: S_{pool}; tournament size: S_{tour}; crossover probability: P_c; mutation probability: P_m; the initial network adjacency matrix: A.
(2) $P \leftarrow$ Initial_Population(S_{pop});
(3) Repeat
(4) $\quad P_{\text{parent}} \leftarrow$ Tournament_Selection($P, S_{\text{pool}}, S_{\text{tour}}$);
(5) $\quad P_{\text{offspring}} \leftarrow$ Genetic_Operation($P_{\text{parent}}, P_c, P_m$);
(6) $\quad P'_{\text{offspring}} \leftarrow$ Local_search($P_{\text{offspring}}$);
(7) $\quad P \leftarrow$ Update_Population($P, P'_{\text{offspring}}$);
(8) Until Termination($I_{\max}$)
(9) Output: the number of added edges, the position of added edges.

Algorithm 1: Framework of our algorithm.

3. The Algorithm for Optimizing Critical Diameter

Memetic algorithms combined with techniques of long-distance and short-distance search have proved to be effective in solving NP-hard problems [33, 34]. In this section, we introduce a memetic algorithm that combines a genetic algorithm and a heuristic local search to optimize critical diameter. We call the method "MA-CD."

3.1. Framework. The framework of MA-CD is shown in Algorithm 1. We first input some necessary parameters such as the maximum iteration number and the population size as well as the adjacency matrix of the network. We generate a population P by the function Initial_Population( ). Next, we repeat the process for optimizing D_c until the number of iterates is $I_{\max}$, or the objective function remains unchanged for 50 iterations. In repeating this process, we first use Tournament_Selection( ) to select the parent population for genetic operations; then we apply two-point crossover and one-point mutation to generate offspring chromosomes by Genetic_Operation( ); we apply some a priori knowledge to carry out a local search on the offspring chromosomes by Local_Search( ); Update_Population( ) is used to construct a new population with better performing chromosomes. Finally, we output the results.

3.2. Representation and Initialization. We aim to find those positions at which we should add edges to optimize D_c. To this end, we find all the positions of these nonexistent edges and encode them as genes $x_i \subseteq X$ in the chromosome $X \in \{0, 1\}$. $x_i = 1$ represents adding a new edge to the corresponding position, while $x_i = 0$ represents not adding an edge. Figure 4 shows an illustration of the representation. We identify the nonexistent edges between nodes 1–3, 1–4, 2–4, 2–5, and 3–5. For the initial network, all the genes are assigned 0 because there are no added edges. If we assign 1 to the first and second gene as shown in Chromosome 2, then the edges between nodes 1–3 and 1–4 will be added. Similarly, when we assign 1 to the first and fourth gene, the edges between 1–3 and 2–5 will be added.

In the initialization, we generate a population of chromosomes and randomly assign 0 or 1 to every gene in the chromosomes.

3.3. The Genetic Operation. The genetic operation consists of two-point crossover and one-point mutation. The crossover operation is described in the appendix. Given two parent chromosomes X_{parent1} and X_{parent2}, we randomly choose two points, and then the parent chromosomes are divided into three parts by two chosen points. Next, we randomly select one part, and all the genes in this part are swapped between X_{parent1} and X_{parent2} with probability P_c (the crossover probability), to generate two offspring chromosomes. Mutation is also described in the appendix, where we choose gene x_i with some probability and reassign $1 - x_i$ to it.

3.4. Local Search. By incorporating some a priori knowledge, a local search can efficiently reduce useless exploration and speed up the convergence of algorithms [35]. We find most optimal networks appear to be disassortative in experiments to optimize APL, and we propose a "disassortativeness-learning" technique to apply this knowledge into local search. Then, we use "Edge-Adding Learning" and "Edge-Dropping Learning" to find the local minimum of our solutions.

3.4.1. Disassortativeness Learning. The detailed algorithm is described in the appendix. We first find the added edge which has the minimum sum of the degrees of the two connected nodes and drop it. Then we find the nonexistent edge which has the maximum difference in degree between the two disconnected nodes and add this new edge to the network.

3.4.2. Edge-Adding Learning. As described in the appendix, we first judge if the diameter of the updated network has decreased to 2. If the diameter exceeds 2, we randomly add edges until the diameter equals 2. Then we output the offspring chromosomes with the updated network diameter equal to 2.

3.4.3. Edge-Dropping Learning. We select every added edge and check whether dropping the added edge will leave the network diameter unchanged. If the drop cannot increase the

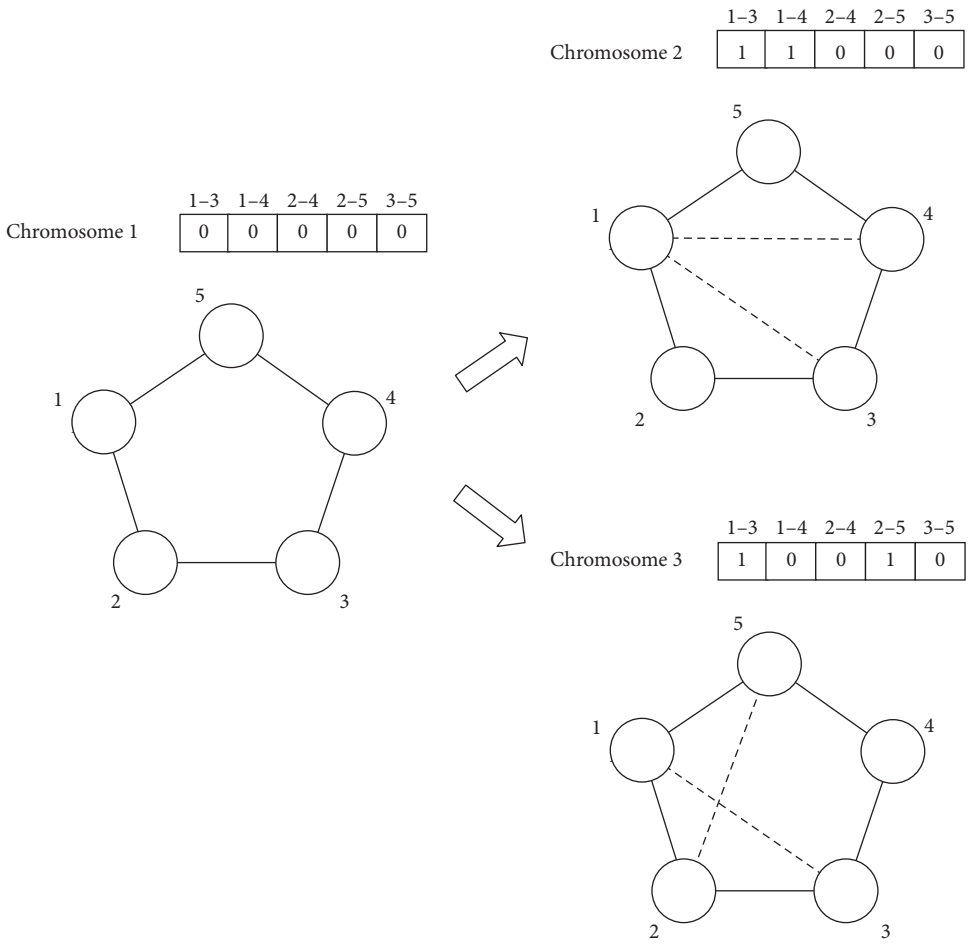


Figure 4: Illustration of representation. The numbers above the chromosomes represent the positions of corresponding nonexistent edges. The solid lines represent initial network edges, while dashed lines represent added edges, and the numbers next to nodes represent the node number.

diameter, we drop the added edge; if the drop increases the diameter, we do not drop the added edge. As a result, some useless added edges may be dropped. The appendix gives the specific procedure of Edge-Dropping Learning.

3.5. Complexity Analysis. The time complexity of MA-CD with the network size n, the number of edges of initial network m, and added edges k can be formulated as follows. Each iteration requires $S_{\text{pool}}/2$ times for crossover and S_{pool} times for mutation, where S_{pool} is the size of mating pool for the genetic operation. Since computing the network diameter costs $O(n^3)$, the total time of genetic operation is $O(S_{\text{pool}}(k + n^3))$. For local search, when executing Disassortativeness Learning, the time to update the matrix is $O(k)$; finding the added edge with the minimum sum of degree requires $O(kn)$; finding the nonexistent edge with the maximum difference in degree requires $O(n^2)$. The time for Disassortativeness Learning is $O(n^2 + kn + k)$. To perform Edge-Adding Learning and Edge-Dropping Learning, we should check at most $n(n-1)/2 - m$ genes for each chromosome and it will cost at most $O(n^5)$ to compute the updated diameter of all changed genes. Therefore, the overall time complexity of MA-CD for each iteration is $O(n^5)$.

Table 1: Parameters of the experiments.

Parameter	Meaning	Value
$I_{\max}$	The maximum iteration number	3000
S_{pop}	Population size	200
S_{pool}	Mating pool size	100
S_{tour}	Tournament size	2
P_c	Crossover probability	0.5
P_m	Mutation probability	0.5

4. Experiments

In this section, we test the performance of MA-CD on different computer-generated networks. The experiments were carried out on a 2.40 GHz CPU, 4.00 GB Memory, and Windows 10 operating system. We use MATLAB to execute the procedure. Table 1 shows the parameters necessary for the experiments.

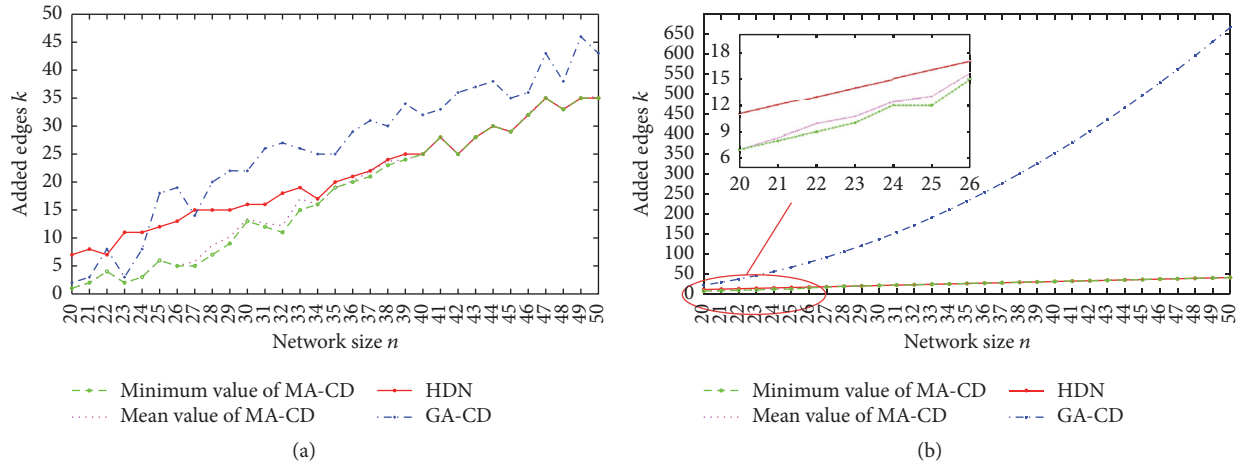


FIGURE 5: Results of optimizing the critical diameter using different methods. (a) is the result for the random network; (b) is the result for the regular network.

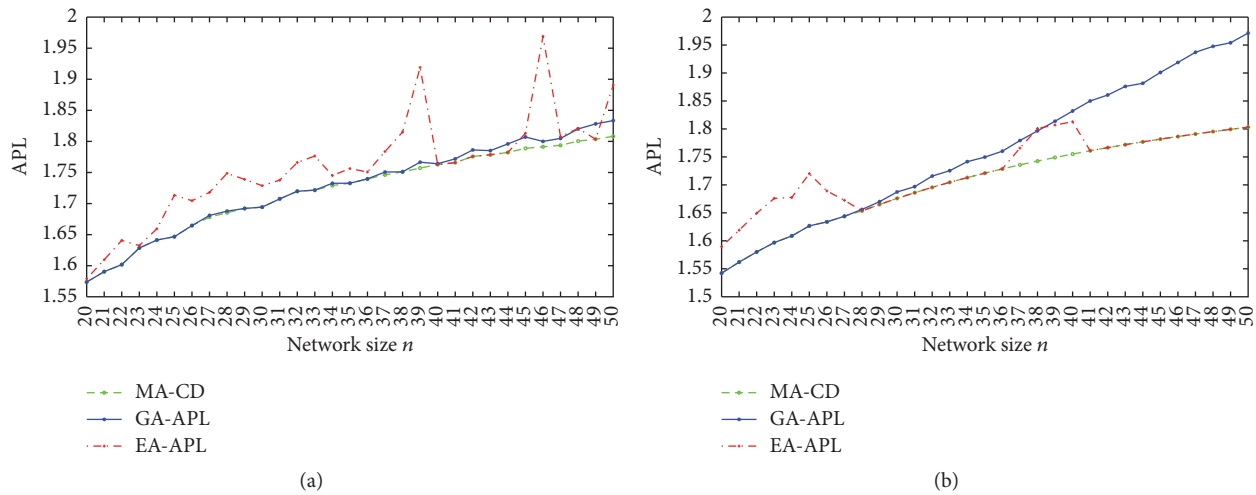


FIGURE 6: Results of optimizing APL using different methods. (a) is the result for the random network; (b) is the result for the regular network.

Our proposed algorithm is carried out ten times on two different network structures: a random network structure and a regular network structure. The detailed information of the two network structures is shown in Appendix. We compare the solution of MA-CD with that of two other methods: (1) adding edges between the highest-degree node and the other nodes as described in Section 2, denoted as HDN; (2) a kind of greedy algorithm which adds edges one by one, with each of the added edges giving the minimum diameter, denoted as "GA-CD" (greedy algorithm for optimizing critical diameter). We compare the minimum and mean value of MA-CD with the minimum value of HDN and GA-CD (the minimum and mean value of these two methods are equal).

Compared with the other two methods for optimizing the critical diameter, MA-CD can always find fewer edges to make the network diameter become the critical diameter, as shown in Figure 5. Further, we find that the results for MA-CD and HDN appear to become the same as the network size becomes larger. In other words, HDN becomes more efficient for larger networks in optimizing D_c. Compared with MA-CD and HDN, GA-CD has worse performance in optimizing D_c, especially for regular networks, even though the greedy strategy performs well in decreasing APL or diameter [23–25].

We show that our proposed method can also efficiently decrease APL. We compute the optimal networks' APL with k shortcut edges added by the MA-CD method, and then this APL is compared with that obtained using the other two methods with the same number k of edges added: (1) the EdgeEffect Algorithm, which maximizes the effect of edge insertion to optimize APL [23], denoted as "EA-APL"; (2) a greedy algorithm adding k edges one by one, with each of the added edges minimizing APL, which has previously been shown to be effective [24], denoted as "GA-APL."

The optimal networks' APL is shown in Figure 6. EA-APL performs worse for random networks, while GA-APL becomes less efficient in regular networks. MA-CD gives the best performance; it can always decrease the APL to its

Table 2: Detailed information of the two network structures.

Network structure	Network size	Number of edges	Density	Diameter	APL
Random networks	20	80	0.42	3	1.58
	21	84	0.40	3	1.62
	22	88	0.38	3	1.66
	23	92	0.36	3	1.64
	24	96	0.35	3	1.67
	25	100	0.33	4	1.74
	26	104	0.32	3	1.72
	27	108	0.31	3	1.73
	28	112	0.30	3	1.77
	29	116	0.29	3	1.78
	30	120	0.28	3	1.80
	31	124	0.27	3	1.80
	32	128	0.26	3	1.80
	33	132	0.25	3	1.84
	34	136	0.24	3	1.87
	35	140	0.24	4	1.88
	36	144	0.23	3	1.86
	37	148	0.22	4	1.94
	38	152	0.22	4	1.95
	39	156	0.21	4	1.98
	40	160	0.21	4	1.97
	41	164	0.20	4	1.98
	42	168	0.20	3	1.97
	43	172	0.19	3	1.97
	44	176	0.19	4	2.01
	45	180	0.18	3	2.02
	46	184	0.18	4	2.04
	47	188	0.17	4	2.03
	48	192	0.17	3	2.03
	49	196	0.17	4	2.05
	50	200	0.16	4	2.04
Regular networks	20	80	0.42	3	1.74
	21	84	0.40	3	1.80
	22	88	0.38	3	1.86
	23	92	0.36	3	1.91
	24	96	0.35	3	1.96
	25	100	0.33	3	2.00
	26	104	0.32	4	2.08
	27	108	0.31	4	2.15
	28	112	0.30	4	2.22
	29	116	0.29	4	2.29
	30	120	0.28	4	2.34
	31	124	0.27	4	2.40
	32	128	0.26	4	2.45
	33	132	0.25	4	2.50
	34	136	0.24	5	2.58
	35	140	0.24	5	2.65

Table 2: Continued.

Network structure	Network size	Number of edges	Density	Diameter	APL
	36	144	0.23	5	2.71
	37	148	0.22	5	2.78
	38	152	0.22	5	2.84
	39	156	0.21	5	2.89
	40	160	0.21	5	2.95
	41	164	0.20	5	3.00
	42	168	0.20	6	3.07
	43	172	0.19	6	3.14
	44	176	0.19	6	3.21
	45	180	0.18	6	3.27
	46	184	0.18	6	3.33
	47	188	0.17	6	3.39
	48	192	0.17	6	3.45
	49	196	0.17	6	3.50
	50	200	0.16	7	3.57

(1) Input: The parent chromosomes X_{parent1} and X_{parent2}.
The number of nonexistent edges of the initial network: N_{non}. Crossover Probability: P_c.
(2) $X_{\text{offspring1}} = X_{\text{parent1}}$;
(3) $X_{\text{offspring2}} = X_{\text{parent2}}$;
(4) randomly generate two positions a and b, which obey: $1 \le a < b \le N_{\text{non}}$;
(5) randomly generate $p \in [0, 1]$;
(6) if $p < P_c$
(7) randomly generate $q \in (0, 1]$;
(8) if $0 < q \le 1/3$
(9) for $i = 1; i \le a; i^{++}$
(10) $temp = X_{\text{offspring1}}(i)$;
(11) $X_{\text{offspring1}}(i) = X_{\text{offspring2}}(i)$;
(12) $X_{\text{offspring2}}(i) = temp$;
(13) end for
(14) end if
(15) if $1/3 < q \le 2/3$
(16) for $i = a + 1; i \le b; i^{++}$
(17) $temp = X_{\text{offspring1}}(i)$;
(18) $X_{\text{offspring1}}(i) = X_{\text{offspring2}}(i)$;
(19) $X_{\text{offspring2}}(i) = temp$;
(20) end for
(21) end if
(22) if $2/3 < q \le 1$
(23) for $i = b + 1; i \le N_{\text{non}}; i^{++}$
(24) $temp = X_{\text{offspring1}}(i)$;
(25) $X_{\text{offspring1}}(i) = X_{\text{offspring2}}(i)$;
(26) $X_{\text{offspring2}}(i) = temp$;
(27) end for
(28) end if
(29) end if
(30) Output: $X_{\text{offspring1}}$ and $X_{\text{offspring2}}$.

Algorithm 2: Crossover operation.

(1) Input: The parent chromosome X_{parent3}.
The number of nonexistent edges of the initial network: N_{non}. Mutation Probability: P_m.
(2) $X_{\text{offspring3}} = X_{\text{parent3}}$;
(3) randomly generate $p \in [0, 1]$;
(4) if $p < P_m$
(5) randomly generate $q \in (0,1]$;
(6) $mt = \text{ceil}(N_{non} * q)$;
(7) for $i = 1$; $i \leq mt$; i++
(8) randomly generate $r \in (0,1]$;
(9) $j = \text{ceil}(N_{non} * r)$;
(10) $X_{\text{offspring3}}(j) = 1 - X_{\text{parent3}}(j)$;
(11) end for
(12) end if
(13) Output: $X_{\text{offspring3}}$

Algorithm 3: Mutation operation.

(1) Input: the offspring chromosome: $X_{\text{offspring}}$. The adjacency matrix of the initial network: A.
(2) $X_{\text{offspring2}} = X_{\text{offspring}}$;
(3) Update the matrix A by decoding $X_{\text{offspring}}$;
(4) Find the element i of $X_{\text{offspring2}}(i) = 1$ with the minimum sum value of nodes pair degrees;
(5) $X_{\text{offspring2}}(i) = 0$;
(6) Find the element j of $X_{\text{offspring2}}(i) = 0$ with the "maximum" difference value of nodes pair degrees;
(7) $X_{\text{offspring2}}(j) = 1$;
(8) Output: $X_{\text{offspring2}}$.

Algorithm 4: Disassortativeness Learning.

(1) Input: the offspring chromosome: $X_{\text{offspring2}}$. The adjacency matrix of the initial network: A.
(2) $X_{\text{offspring3}} = X_{\text{offspring2}}$;
(3) Update the matrix A by decoding $X_{\text{offspring2}}$;
(4) Repeat
(5) randomly choose a gene $X_{\text{offspring3}}(i) = 1$;
(6) Until the diameter of the updated matrix $D = 2$
(7) Output: $X_{\text{offspring3}}$.

Algorithm 5: Edge-Adding Learning.

lowest level compared with the other two algorithms. Thus, we conclude that MA-CD can be used to optimize APL. If we can add a large number of edges to decrease APL, we just need to find a solution for optimizing D_c and then add the remaining edges randomly.

5. Conclusion

In this paper, we find a critical case in which the network diameter declines to 2 when a new edge is added to the network in the process of solving the shortcut-selection problem. Using the relationship between APL and the network diameter, we transform the problem of optimizing APL into the problem of finding shortcut edges to quickly decrease the diameter to 2, which we define as the problem of optimizing the critical diameter. Further, we suggest a method to solve this problem based on a memetic algorithm. The experimental results show that our proposed method can efficiently optimize the critical diameter and is efficient in solving the shortcut-selection problem to decrease the APL.

Appendix

See Algorithms 2, 3, 4, 5, and 6 and Table 2.

(1) Input: the offspring chromosome: $X_{\text{offspring3}}$.
The adjacency matrix of the initial network: A. The number of nonexistent edges of the initial network: N_{non}.
(2) $X_{\text{offspring4}} = X_{\text{offspring3}}$;
(3) Repeat
(4) *islocal* ← TRUE;
(5) rearrange the sequence number of the chromosome *seq*=randperm(N_{non});
(6) for $i = 1$; $i \leq N_{\text{non}}$; i++
(7) if $X_{\text{offspring4}}(seq(i)) = 1$
(8) $X_{\text{offspring4}}(seq(i)) = 0$;
(9) if the diameter of updated network $D = 2$
(10) *islocal* ← FALSE;
(11) else
(12) $X_{\text{offspring4}}(seq(i)) = 1$;
(13) end if
(14) end if
(15) end for
(16) Until *islocal* is TRUE;
(17) Output: $X_{\text{offspring4}}$.

Algorithm 6: Edge-Dropping-Learning.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is jointly supported by Key Project of the National Social Science Foundation of China (Grant no. 12AZD110) and Humanities and Social Science Talent Plan, Fundamental Research Funds for the Central Universities (Grant no. 2011jdgz08).

References

[1] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

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

[3] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, no. 1, pp. 47–97, 2002.

[4] M. E. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.

[5] A.-L. Barabási, "Scale-free networks: a decade and beyond," *Science*, vol. 325, no. 5939, pp. 412–413, 2009.

[6] M. Cai, H.-F. Du, and M. W. Feldman, "A new network structure entropy based on maximum flow," *Acta Physica Sinica*, vol. 63, no. 6, Article ID 060504, 2014.

[7] D. S. Callaway, M. E. J. Newman, S. H. Strogatz, and D. J. Watts, "Network robustness and fragility: percolation on random graphs," *Physical Review Letters*, vol. 85, no. 25, pp. 5468–5471, 2000.

[8] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 12, pp. 7821–7826, 2002.

[9] F. Yu, Y. Li, and T.-J. Wu, "A temporal ant colony optimization approach to the shortest path problem in dynamic scale-free networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 389, no. 3, pp. 629–636, 2010.

[10] H. Ma and A.-P. Zeng, "Reconstruction of metabolic networks from genome data and analysis of their global structure for various organisms," *Bioinformatics*, vol. 19, no. 2, pp. 270–277, 2003.

[11] B. Wang, H. W. Tang, C. H. Guo, Z. L. Xiu, and T. Zhou, "Optimization of network structure to random failures," *Physica A: Statistical Mechanics and Its Applications*, vol. 368, no. 2, pp. 607–614, 2006.

[12] D. J. Ashton, T. C. Jarrett, and N. F. Johnson, "Effect of congestion costs on shortest paths through complex networks," *Physical Review Letters*, vol. 94, no. 5, Article ID 058701, 2005.

[13] L. F. Lago-Fernández, R. Huerta, F. Corbacho, and J. A. Sigüenza, "Fast response and temporal coherent oscillations in small-world networks," *Physical Review Letters*, vol. 84, no. 12, article 2758, 2000.

[14] P. M. Gade and C. Hu, "Synchronous chaos in coupled map lattices with small-world interactions," *Physical Review E*, vol. 62, no. 5, pp. 6409–6413, 2000.

[15] J. Jost and M. P. Joy, "Spectral properties and synchronization in coupled map lattices," *Physical Review E*, vol. 65, no. 1, Article ID 016201, 2002.

[16] H. Hong, M. Y. Choi, and B. J. Kim, "Synchronization on small-world networks," *Physical Review E*, vol. 65, no. 2, Article ID 026139, 2002.

[17] M. Barahona and L. M. Pecora, "Synchronization in small-world systems," *Physical Review Letters*, vol. 89, no. 5, Article ID 054101, 2002.

[18] J. Jost and M. P. Joy, "Spectral properties and synchronization in coupled map lattices," *Physical Review E*, vol. 65, no. 1, part 2, Article ID 016201, 2002.

[19] H. Hong, M. Y. Choi, and B. J. Kim, "Synchronization on small-world networks," *Physical Review E: Statistical Nonlinear & Soft Matter Physics*, vol. 65, no. 2, pp. 95–129, 2002.

[20] Q. Xuan, Y. Li, and T.-J. Wu, "Optimal symmetric networks in terms of minimizing average shortest path length and their sub-optimal growth model," *Physica A: Statistical Mechanics and its Applications*, vol. 388, no. 7, pp. 1257–1267, 2009.

[21] O. Keren, "Reduction of the average path length in binary decision diagrams by spectral methods," *IEEE Transactions on Computers*, vol. 57, no. 4, pp. 520–531, 2008.

[22] A. Meyerson and B. Tagiku, "Minimizing average shortest path distances via shortcut edge addition," in *Proceedings of the International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pp. 272–285, Springer, 2009.

[23] N. Parotsidis, E. Pitoura, and P. Tsaparas, "Selecting shortcuts for a smaller world," in *Proceedings of the SIAM International Conference on Data Mining*, British Columbia, Canada, 2015.

[24] S. H. Lee and P. Holme, "A greedy-navigator approach to navigable city plans," *European Physical Journal: Special Topics*, vol. 215, no. 1, pp. 135–144, 2013.

[25] M. Papagelis, "Refining social graph connectivity via shortcut edge addition," *ACM Transactions on Knowledge Discovery from Data*, vol. 10, no. 2, article no. 12, 2015.

[26] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Proceedings of the 12th International Conference on Information and Knowledge Management*, pp. 556–559, ACM, New Orleans, La, USA, November 2003.

[27] Z. Liu, Q.-M. Zhang, L. Lü, and T. Zhou, "Link prediction in complex networks: a local naïve Bayes model," *EPL*, vol. 96, no. 4, Article ID 48007, 2011.

[28] L. Lü and T. Zhou, "Link prediction in complex networks: a survey," *Physica A: Statistical Mechanics and Its Applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[29] F. Gao, K. Musial, C. Cooper, and S. Tsoka, "Link prediction methods and their accuracy for different social networks and network metrics," *Scientific Programming*, vol. 2015, Article ID 172879, 13 pages, 2015.

[30] E. D. Demaine and M. Zadimoghaddam, "Minimizing the diameter of a network using shortcut edges," in *Proceedings of the Scandinavian Conference on Algorithm Theory*, pp. 3–5, Springer, Bergen, Norway, 2010.

[31] J. Peng and G. Xu, "Average path length for Sierpinski pentagon," *International Journal of Advancements in Computing Technology*, vol. 5, no. 5, p. 724, 2013.

[32] M. Imase and M. Itoh, "Design to minimize diameter on building-block network," *IEEE Transactions on Computers*, vol. 30, no. 6, pp. 439–442, 1981.

[33] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: a literature review," *Swarm and Evolutionary Computation*, vol. 2, pp. 1–14, 2012.

[34] Y.-S. Ong, M. H. Lim, and X. Chen, "Memetic computation-past, present future," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 24–31, 2010.

[35] M. Gong, B. Fu, L. Jiao, and H. Du, "Memetic algorithm for community detection in networks," *Physical Review E*, vol. 84, no. 5, Article ID 056101, 2011.