Minimize the average mean first passage of random walk in complex networks by genetic algorithm

Yue Yu
Department of Physics
Hong Kong University of Science and Technology
Hong Kong
Email: yyuap@connect.ust.hk

Kwok Yip Szeto
Department of Physics
Hong Kong University of Science and Technology
Hong Kong
Email: phszeto@ust.hk

Abstract—We investigate the methods of rewiring a given connected network so that the average mean first passage time (AMFPT) for random walk process can be reduced. Two rewiring mechanisms are used, which we call the zeroth order method (Z) and the higher order method (H) according the series expansion analysis of the first passage time. We aim at finding the optimal sequence of rewiring action of k units long composed of the Z and H methods that yield the maximal reduction of average mean first passage time. We use both the simple genetic algorithm (SGA) and mutation only genetic algorithm (MOGA) and the results show in general MOGA produces networks with higher reduction of AMFPT for benchmark tests on all three major classes of complex networks (ER, BA, WS). In general, the higher order method appears more often in the networks with large clustering coefficient. The network with small clustering coefficient will develop to regular networks through the zeroth order rewiring method, while the network with initially large clustering coefficient has preference to reduce the number of small loops through the higher order rewiring method. We also apply our rewiring sequence of three steps on real networks (IEEE30, IEEE 57) and produce remarkable reduction on the AMFPT. This indicates that our method of analysis can be of practical importance for engineering.

I. INTRODUCTION

Complex networks are important mathematical representation of complex systems in many areas of science and engineering. [1], describing a wide variety of real systems in sociology [2], biology [3], and transportation [4]. For real-world networks, much research has been done on small world, scale-free, and fractal scaling properties [1], [5], [7]. Recent research efforts have begun to focus on the relation between dynamical processes and the topological properties of the underlying networks. Apart from the topological feature of the networks [8], the dynamical aspect plays an important role in information transmission, diffusion and searching processes [9], [10]. An established method for this purpose is random-walk [11]–[23], because it is simple to implement since nodes only require local information to take routing decisions, with application in modelling the searching (transportation) efficiency of the network. In general, simple random walkers require little computational power for their navigations. A common topic for random walks is the study of the distribution of the first passage time [24], which is a good tool to quantify the transportation speed. The first passage time (FTP) is the first time for the walker to move from the origin node \(i\) to the destination node \(j\) in a given path in a connected network. The mean first passage time is the average of FTP over all the possible paths, describing the average time needed for the walker starting from the origin \(i\) to reach its destination \(j\). Since a walkers movements can be interpreted as spreading signals in the network, an important measurement is the first time the walker visits a given destination, as his arrival may be considered critical in triggering many events, such as neuron firing [25] and the spread of epidemics in social networks [26]. In engineering-related applications, random walk is also very useful in resource allocation, node sampling, community detection [12], [13], [27], [28], tracking of moving object [29], and dynamic routing on networks [30]. Through the analysis of random walk, many features of underlying networks, such as the diameter, centrality, and community structure can be revealed.

A famous study in network search is Milgram’s small-world experiment, which showed that the shortest paths between any pair of nodes can be found using only local information [5]. Later, Kleinberg [14] pointed out that efficient navigability is a fundamental property of only some small-world structures, (in many large networks, search by a simple random walker is inefficient). Improvement of efficiency can be made using local information [12], [13], such as the degree of neighboring nodes [13], and local betweenness centrality [16]. In general, optimal efficiency is unachievable unless all nodes know how to deliver the message along the shortest paths. This condition implies a high cost of computation [17], and large data storage capacity for information on the best paths [18]. Thus, it is a great challenge to design efficient search algorithms that use less power and storage.

The underlying networks that random walkers move come in two categories: spatial network and non-spatial ones. For spatial network, we find many examples of the application of multiple random walkers in physics [27], such as wetting [31], reaction-diffusion processes [32], and the motion of colloidal particles [33]. Traditionally, mathematicians have proved theorems on the dynamics of SRW for simple networks. One of the standard quantities they calculate is the cover time of a graph, which is the expected time taken by a random walker to visit every node of the graph at least once. Beside cover time, the first passage time [27], [34] of multiple random walks have
given us new insight into the dynamics of interacting spins \[35\], and the lifetime of knots in polymer chains \[31\], \[36\]. So far we consider only static networks. In real-world situations, networks are evolving, implying that we should also consider the time scales of different dynamics. If the network evolves only slowly compared to the time taken by the walkers in each step, then the network is approximately static. The interesting question we like to address in this paper is to enquire about the minimum modification of the network topology, such as the use of rewiring links, that will reduce the mean first passage time for random walker starting from one node \(i\) to reach a destination node \(j\). The answer will be trivial if we are only concerned about one particular pair of origin node \(i\) to a particular destination node \(j\), but will be highly nontrivial if we require that the average mean first passage time for all possible pair of nodes in a connected network with \(N\) nodes and \(L\) links to be reduced maximally by a minimal rewiring action. Here, the objective function to minimize is the average mean first passage time (AMFPT, denoted by \(\tau\)) over all possible pairs of origin-destination nodes for the entire network.

For a single simple random walker, the calculation of the mean first passage time has been widely studied in different lattices with an unbounded domain \[24\], a bounded domain with an arbitrary shape of boundary \[37\], and fractal networks with homogeneous degree distribution \[38\], \[39\]. Our recent work has provided a simple approach to obtain the distribution function of the first passage time \[13\] in complex networks. Based on these works, we find that there exist two efficient methods of rewiring that can substantially reduce the AMFPT. These two methods are based on the series expansion for MFPT, which is given by \[24\], \[40\]

\[
T_{ij} = \frac{2L}{d_j} \sum_{t=0}^{\infty} \left[ P(j,t|j) - P(j,t|i) \right]
\]  

(1)

\(L\) is the number of links in the network, and \(d_j/2L\) is the equilibrium probability of the random walker located at destination \(j\). Here \(d_j\) is the degree of the node \(j\) and is the number of nearest neighbours of node \(j\). Clearly, the higher the degree of the node, the higher the probability for the random walker reaching it, thus, this probability is proportional to the ease to find the node \(j\), or its degree. Thus, the time to reach the destination node is inversely proportional to \(d_j/2L\). Furthermore, \(P(j,t|j)\) is the conditional probability for the walker starting from \(j\) to come back after time \(t\). It describes the waste of time in the loops of size \(t\).

To increases the searching efficiency of random walk, we need to minimize the average of MFPT over all possible \(i\) and \(j\). In general, the \(t=0\) term \(2L/d_i\) in the expression for \(T_{ij}\) dominates the summation series if the clustering coefficient is not large, especially for the networks with large number of links. It only depends on the degree distribution. For the network with fixed number of links \(L\), the summation of \(d_i\) is a constant. Therefore, the first rewiring method (zero-th order method) is to reduce the deviation of the degree distribution, which minimizes the average of \(2L/d_i\). It reduces the number of \"difficult\" destinations, while the time to find the \"easy\" destinations is not changed largely. It results in a regular network after multiple rewiring.

We now consider the contribution from higher order terms in the expression for \(T_{ij}\). Detailed expansion shows that \(t=1,2\) term also depends purely on degree distribution. However, the \(t=3\) term depends on the number of triangles in the network, and the \(t=4\) term depends on the number of squares etc. The second method (higher order method) is to minimize the number of small loops in the network, so that the walker is less likely to come back to the starting node in a short time. Then the time waste in the small loops is saved. The large loops actually ensures the walker to reach farther nodes, therefore they do not contribute to the time wasted. After multiple rewiring, the final network has less small loops. As a consequence, the network will have a small clustering coefficients.

The input parameter for each method is the starting node \(n_i\), chosen to be the origin of the link being rewired. The destination node can be any one of the remaining nodes in the network, but in the two rewiring methods we seek to find the destination node so as to maximally reduce the average mean first passage time \(\tau\) of the entire network.

II. CHROMOSOME CONSTRUCTION

We now apply the two possible methods consecutively to rewire \(k\) times. The rewiring sequence

\[
(n_1, m_1, n_2, m_2...n_k, m_k), m_i = 0, 1
\]

(2)

contains all the information about this \(k\)-time modification, including the starting node \(n_i\) and the method \(m_i\) for the \(i\)th rewiring. Our main objective is to find the best rewiring sequence, which leads to the smallest average of mean first passage time \(\langle T_{ij} \rangle\).

We use genetic algorithm to solve this problem, we first digitalize the rewiring sequence into binary code. We take the binary code of \(n_i\) and \(m_i\) separately, and combine them to construct one chromosome.

\[
\text{code}(n_1, n_2...n_k, m_k) = bi(n_1)bi(m_1)...bi(n_k)bi(m_k)
\]

(3)

For example, if the network contains 8 nodes, then the initial nodes are described by 3 digits, while the two methods occupies another 1 digits. For \(k=2\) steps rewiring, totally 8 digits are needed. \((010/1/001/0)\) is a particular chromosome. It is decoded as the rewiring sequence \((3,2,2,1)\), i.e. using second method on node 3 then first method on node 2. In general, the length of the chromosomes needs to be at least

\[
k[\log_2 N + 1]
\]

(4)

where \([x]\) is the smallest integer larger than \(x\). \(N\) is the number of nodes in the network. The length is proportional to the number of rewiring \(k\). The complexity of the genetic algorithm depends on the lengths of the chromosomes and number of chromosomes. If we denote the two rewiring mechanisms involved are either the zeroth order method (called the method
Z) or the higher order method (called the method H) according to the analysis of $T_{ij}$, then our optimization problem involves the finding of the optimal solution of $k$ units long string composed on the alphabets $Z$ or $H$ that denote the sequence of rewiring methods that yield the maximal reduction of average mean first passage time. Thus, there are in general $2^k$ sequences. Since each rewiring method involves the choice of the origin node $i$ and link $(ij)$ to the new endpoint at node $j$, there are $N$ choices for the initial node $i$ and $d_i$ choice for the original endpoint $j$ and $N−2$ choices for the new endpoint at node $j$, therefore each application of the rewiring involves $M_i = N \times d_i \times (N−2)$ possibilities which is of the order of $\mathcal{O}(N^2d)$ where $\langle d \rangle$ is the mean degree of the nodes in the given network. The complexity of the optimization for $k$ steps is of the order of $M^{2^k}$. Genetic algorithm is therefore a good way to handle optimization of this type of problems with such a large solution space.

To evaluate the chromosome for a given network, we need to decode the binary sequence first, and apply the rewiring sequence to the network. The fitness value of a chromosome is inversely proportional to the average of mean first passage time on all possible node pairs. Now the main task is transformed to maximize the highest value of the chromosomes using the Darwinian principle of survival of the fittest.

### III. GENETIC ALGORITHM

We first generate $N_c$ random chromosome. The chromosomes are initially arranged into descending order according to their value. Then we apply the following mutation cycle of $T$ times. One mutation cycle consists of two steps. We first let the chromosomes to mutate according to their mutation rate. The new chromosomes are then reordered. In our formulation, we only use mutation as the genetic operator.

We consider two types genetic algorithm, simple genetic algorithm (SGA) and mutation only genetic algorithm (MOGA) [1], [2]. For simple genetic algorithm, the mutation rate $M_{ij}$ for $j^{th}$ binary code at $i^{th}$ chromosomes depends on $i$ only,

$$M_{ij} = \begin{cases} 0, & 1 \leq i \leq N_1 \\ c, & N_1 < i \leq N_2 \\ 1, & N_2 < i \leq N_c \end{cases}$$

Then there is no difference in mutation rate inside a chromosome. There are three input parameters $N_1$, $N_2$ and $N_c$ for SGA.

MOGA, however, does not require input parameters, but it evolves after each cycle. And the mutation rate depends on the position $j$ of binary code inside the chromosome. We first define the row mutation rate (mutation rate according to chromosomes), which is in descending order $a_i(t) = (i-1)/(N-1)$. It describes the survival of the fittest principle.

The column mutation rate distinguishes different positions on the chromosomes. If a column is mainly dominated by single genus, then it is less likely to evolve. The column mutation rate will protect the dominate (often the better) genes from mutation.

The detailed construction of column mutation rate $b_j(t)$ depends on the weighted average of 0 and 1 in each column $p_{j0}, p_{j1}$.

$$p_{jX} = \frac{\sum_c (N_c + 1 - k)\delta_{kj}(X)}{\sum_m m}$$

$$b_j(t) = 1 - |p_{j0} - 0.5| - |p_{j1} - 0.5|$$

$$b_j(t) = b_j'\sum b_j'$$

The denominator $m$ is for normalization on $p_{jX}$. $\delta_{kj}(X) = 1$ if the jth element on kth chromosome is X, and zero otherwise. Here $X=0.1$. The weight of the average purely depends on the column, for different chromosomes have different importance. The resulted mutation rate is the product of the row and column mutation rate $M_{ij} = a_ib_j$.

### IV. SIMULATION

We use genetic algorithm (SGA or MOGA) on networks generated by ER [43], BA [1] and WS [44] model. There are $N=16$ nodes, and $L=64$ links in each network. In the chromosome, the initial node is then described by 4 binary digits. And for $k=5$ rewiring, the rewiring sequence is then encoded into 25-size binary sequence.

MOGA mutation rate is calculated after each cycle. For SGA, we set $N_1 = 2$, $N_2 = 8$. As a comparison, the mutation probability $c$ is set to the mean value of the MOGA mutation matrix (0.02). Ten chromosomes are randomly generated as the initial state, and 500 cycles are performed for each simulation.

We first measure the percentage difference in the average mean first passage time between the randomly generated rewiring sequence and the rewiring sequence from genetic algorithm. It measures the efficiency of the genetic algorithm. The randomly rewiring sequence is decoded from a random binary sequence corresponding to the network and rewiring steps we use. As a comparison, we first record the percentage difference in AMFPT before and after a random rewiring sequence, which measures the efficiency of a random rewiring sequence.

As for benchmark comparison, we generate 100 random binary codes with 25 digits. The decoded random rewiring methods lead to average 4.4% and 6.6% change in AMFPT on 100 random ER and BA networks, respectively. These numbers are the performance of the random rewiring sequence for three steps rewiring. Compared with random rewiring sequence, the rewiring sequence from genetic algorithm can further lead to an extra 0.8%, and 1.5% change in AMFPT in these two types of networks. That is, the performance for the original rewiring sequence is boosted by $0.8/4.4=18\%$ and $1.5/6.6=23\%$ when we use genetic algorithm find a better rewiring sequence. The results are shown in Fig[1].

As for benchmark comparison, we generate 100 random binary codes with length 25. The decoded random rewiring methods lead to average 1.7% and 3.5% change in AMFPT on a 100 random WS networks with $\beta = 0.1, 0.7$, respectively. Compared with these random rewiring sequence, the rewiring sequence from genetic algorithm can further reduce AMFPT...
by 0.6% for $\beta = 0.1$ network, and 0.8% for $\beta = 0.2$ network. The performance of original rewiring sequence is improved by $0.6/1.7 = 35\%$ and $0.8/3.5 = 23\%$ when we use genetic algorithm to find a better rewiring sequence. The results are shown in Fig. 2.

Starting from the random rewiring sequence, the use of genetic algorithm can boost the performance of the rewiring sequence by more than 18%. For small number of mutation, SGA can achieve larger improvement, but in general, when we run the genetic algorithm for a larger number of generations, MOGA shows better performance in all of the three types of networks after 500 mutations.

Now we study the structure of the eventual network. Recall that the first method Z focuses on the degree distribution of the network, while the second method H reduces the number of small loops in the network. As a result, we can measure the proportion of the two methods in the final rewiring sequence after mutations by analysing the composition of the final rewiring sequence. The results are shown in Table I for ER and BA networks and Table II for WS networks.

In general, the second method appears more often in the networks with large clustering coefficient (BA network). For short rewiring sequence, the final result is mainly dominated by zero-th order (Z) method rewiring, while for longer rewiring sequence, the higher order method appears H more frequently. Similar results are found in small world model, where the second method becomes important if the clustering coefficient is large, or equivalently, small $\beta$ (Table III). We see that for WS networks, the second method H is important even for short rewiring sequence.

To conclude, the zeroth order method (Z) is generally preferred for shorter rewiring sequence, as shown in ER network. As for longer rewiring sequence, the second method (H) becomes more important. As a consequence, the network will first evolve to a regular network, then reduces its clustering coefficient.

The higher order method is particularly important in the
network with large clustering coefficient (BA model and small world model with small $\beta$) even for short rewiring sequence. In this case, the network will evolve in a mixed way of the two methods to achieve the optimal searching speed.

V. COMPARISON WITH EXHAUSTIVE SEARCH

We now study the validity of the genetic algorithm. We want to compare the efficiency of the rewiring sequence obtained from genetic algorithm and exhaustive search.

The network used is ER network with 8 links and 20 nodes. We prepare a 3-step rewiring. Then the length of a chromosome is 12, containing information of the rewiring methods and the initial nodes. We do exhaustive search on all possible $2^{12} = 4096$ chromosomes to find the chromosome with the highest value. This covers all possible combinations of rewiring methods and initial nodes for 3-step rewiring.

The result from exhaustive search provides the rewiring sequence with the highest percentage difference in AMFPT between the rewired network and the original network. On the same network, we perform genetic algorithm to get the mutated rewiring sequence, which is supposed to lead to the same percentage difference in AMFPT. We check this assumption by taking the ratio between the two percentage difference. If the ratio is close to one, then genetic algorithm can achieve optimum satisfactory as we know the optimum from exhaustive search.

VI. APPLICATION TO IEEE30 AND IEEE57 NETWORK

We first show an example of ER network with $N=8$ nodes and $L=20$ links. As discussed above, the exhaustive search takes 4096 steps. We now try 300 mutation cycles with 5 chromosomes. Thus, we need a total of 1500 steps. The adjacency matrix before rewiring is

$$
\begin{pmatrix}
0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0
\end{pmatrix}
$$

(7)

corresponding to Fig. 4

Fig. 4. One of the original network chosen for ER network. The average of mean first passage time is 7.92.

After 300 mutations, simple genetic algorithm cannot provide the optimal rewiring sequence, and the adjacency matrix

In the simulation on 100 ER networks, the exhaustive search on average can reduce the AMFPT by 6.0%. The mutation only genetic algorithm can reach up to 99.4% of this result after 300 mutations, while simple genetic algorithm can reach 99.2%. In the search, we evaluate a total of 1500 chromosomes for both algorithm, compared with 4096 chromosomes used in exhaustive search.

In general, less chromosomes and mutation cycles are needed for small networks with shorter rewiring sequence. This then shows that genetic algorithm already provides substantial improvement over exhaustive search even for a small network. We expect that the improvement will be more important for larger network.
The resulting network is shown in Fig. 5. The rewiring efficiency, defined by the ratio of the change in AMFPT between genetic algorithm result and optimal result by exhaustive search, can reach 90% for SGA algorithm.

The mutation only genetic algorithm, however, can give the best result (Fig 6). The adjacency matrix is rewired to the network with the following adjacency matrix,

\[
\begin{pmatrix}
0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0
\end{pmatrix}
\]

The detailed rewiring sequence is to rewire link (5,4) to (5,1), link (2,8) to (2,7) and link (5,8) to (5,2). However, this is not the optimal network.

We further apply the genetic algorithm to IEEE30, 57 network. Compared with the ER/BA network generated in Fig.1, there are less links in IEEE network. As a result, the initial average mean first passage time is relatively large.
We now apply 3-step rewiring to modify the network. Compared with the original network, the average of mean first passage time is reduced by more than 8%. Both of the zeroth and higher order methods are applied in the rewiring sequence. After rewiring, the network has less nodes with small degree, and the clustering coefficient is reduced.

VII. SUMMARY

In this paper, we study the optimization problem of rewiring sequence in order to reduce the mean first passage time of the networks, i.e. increase the searching speed of random walk. The two rewiring mechanisms involved are either the zeroth order method (method Z) or the higher order method (method H) according the analysis of $T_{ij}$. The optimization problem involves the finding of the optimal sequence of k units long composed of the (Z and H) methods that yield the maximal reduction of average mean first passage time (AMFPT). We use both the simple genetic algorithm (SGA) and mutation only genetic algorithm (MOGA) and the results show in general MOGA produces networks with higher reduction of AMFPT. In general, the efficiency of the rewiring sequence increases by more than 18% on the network with 16 nodes for all three major classes of complex networks investigated (ER, BA, WS) as shown in Table I and II. We also apply our rewiring sequence of three steps on real networks (IEEE30, IEEE 57) and produce remarkable reduction on the AMFPT. This indicates that our method of analysis can be of practical importance for engineering.

We also observe from these studies that the evolution of the network structure largely depends on the initial clustering coefficients. The network with small clustering coefficient
will develop to regular networks, while the network with initially large clustering coefficient has preference to reduce the number of small loops. As for future work, we can consider the following two objectives. First of all, we can further improve the efficiency ratio by means of adaptive genetic algorithm [42] to include cross-over in the mutation matrix. This allows the reordering of rewiring. We can also introduce annealing to the mutation matrix of modified only genetic algorithm to increase the searching speed.

VIII. ACKNOWLEDGEMENT

K. Y. Szeto acknowledges the support of grant FS-GRF13SC25 and FSGRF14SC28.

REFERENCES