2019 International Symposium on Nonlinear Theory and Its Applications
NOLTA2019, Kuala Lumpur, Malaysia, December 2-6, 2019.

A Genetic Algorithm for Maximizing Algebraic Connectivity of Graphs by Adding Multiple Edges

Hiroki Tajiri, Tsuyoshi Migita and Norikazu Takahashi

Graduate School of Natural Science and Technology, Okayama University
3-1-1 Tsushima-naka, Kita-ku, Okayama 700–8530, Japan
Email: tajiri@momo.cs.okayama-u.ac.jp, migita@cs.okayama-u.ac.jp, takahashi@cs.okayama-u.ac.jp

Abstract—The second smallest eigenvalue of the Laplacian matrix, also known as the algebraic connectivity, of a graph represents how well the graph is connected. This paper considers the problem of maximizing the algebraic connectivity of graphs by adding a prescribed number of edges. We propose a genetic algorithm for this problem and evaluate its performance through experiments.

1. Introduction

Distributed computation is an important technique for network systems such as the Internet, sensor networks and multirobot systems. Some distributed algorithms over network systems are characterized by the algebraic connectivity [1, 2], the second smallest eigenvalue of the Laplacian matrix. For example, the convergence speed of a consensus algorithm is determined by the algebraic connectivity [3]. Thus, in order to achieve fast convergence, we need to design the network with a high algebraic connectivity [4].

Recently, Li *et al*. considered the problem of maximizing the algebraic connectivity of undirected graphs by adding one edge, and proposed a simple algorithm called the Minimum Degree and Maximum Distance (MDMD) [5]. Roughly speaking, this algorithm first selects one vertex i with the minimum degree, then selects another vertex j having the maximum distance from i, and finally adds the edge connecting i and j. The MDMD algorithm has a very low computational cost because the computation of the algebraic connectivity is not needed. Nevertheless, it can obtain high algebraic connectivity values as reported in [5].

In this paper, we consider a more general form of the above-mentioned problem, in which the number of edges to add is not restricted to one, and propose a genetic algorithm for finding an approximate solution. Genetic algorithms have proved to be powerful tools for various combinatorial optimization problems [6]. We show through experiments using small graphs that the proposed algorithm can achieve higher algebraic connectivity values than a simple extension of the MDMD algorithm.

2. Notations and Problem Statement

Let $G = (V, E)$ be a simple undirected graph with the vertex set $V = \{1, 2, \ldots, n\}$ and the edge set E. The undirected edge connecting vertices i and j is denoted by $\{i, j\}$. Let $\boldsymbol{A} = (a_{ij}) \in \{0, 1\}^{n\times n}$ be the adjacency matrix of G. Then a_{ij} $(= a_{ji})$ takes 1 if $\{i, j\} \in E$ and 0 otherwise. Note that $a_{ii} = 0$ for all $i \in \{1, 2, \ldots, n\}$ because G is simple. Let $\boldsymbol{D} = \mathrm{diag}(d_1, d_2, \ldots, d_n)$ be the degree matrix of G, where d_i is the degree of vertex i and thus equal to $\sum_{j=1}^{n} a_{ij}$. The Laplacian matrix $\boldsymbol{L} = (\ell_{ij}) \in \mathbb{Z}^{n\times n}$ of G is defined by $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$. Because $\boldsymbol{L}$ is positive-semi definite, all eigenvalues of $\boldsymbol{L}$ are nonnegative real numbers. In addition, because $\boldsymbol{L}\mathbf{1} = \mathbf{0}$ where $\mathbf{1}$ is the vector of all ones, the smallest eigenvalue of $\boldsymbol{L}$ is 0. The eigenvalues of $\boldsymbol{L}$ are denoted by $\lambda_1 (= 0) \le \lambda_2 \le \lambda_3 \le \cdots \le \lambda_n$. The second smallest eigenvalue λ_2 is called the algebraic connectivity of G [1, 2].

In what follows, by a graph, we always mean a simple undirected graph. For a graph $G = (V, E)$ and a positive integer k, the set of all graphs obtained from G by adding k edges is denoted by $\mathcal{G}(G, k)$. In other words,

$$\mathcal{G}(G, k) \triangleq \{G' = (V, E \cup E^+) \,|\, E \cap E^+ = \emptyset,\ |E^+| = k\}. \quad (1)$$

The problem we consider in this paper is stated as follows.

Problem 1 Given a graph $G = (V, E)$ and an integer k satisfying $1 \le k \le |V|(|V|-1)/2-|E|$, find a graph $G^* \in \mathcal{G}(G, k)$ that maximizes the algebraic connectivity among all graphs in $\mathcal{G}(G, k)$.

Problem 1 is a generalization of the problem considered in [5] because the latter corresponds to the special case of the former where the value of k is restricted to 1. The simplest approach to Problem 1 is the brute-force search, but the time complexity of this algorithm is very high even when $k = 1$ [5]. So we need to develop efficient approximation algorithms.

3. Sequential MDMD Algorithm

In this section, we first review the Min-Degree and Max-Distance (MDMD) algorithm developed by Li *et al.* [5] for solving the special case of Problem 1 where $k = 1$. We then presents a simple method called the sequential MDMD algorithm for solving Problem 1.

The MDMD algorithm chooses an edge to be added to G based on the degrees of vertices, the distance between two vertices, and the Extendibility Centralities (EC) defined by

$$c_i \triangleq \sum_{k=1}^{n} a_{ik} d_k, \quad i = 1, 2, \ldots, n. \quad (2)$$

Algorithm 1 MDMD Algorithm [5]

Input: $G = (V, E)$
Output: $G_{\text{out}} = (V, E_{\text{out}})$

1: Set $\mathcal{I} \leftarrow \{i \in V \mid d_i \leq d_{i'} \text{ for all } i' \in V\}$.
2: Select one $i^* \in \mathcal{I}$ such that $c_{i^*} \leq c_i$ for all $i \in \mathcal{I}$.
3: Set $\mathcal{J} \leftarrow \{j \mid j \text{ has the maximum distance from } i^*\}$.
4: Select one $j^* \in \mathcal{J}$ such that $c_{j^*} \leq c_j$ for all $j \in \mathcal{J}$.
5: Set $E_{\text{out}} \leftarrow E \cup \{\{i^*, j^*\}\}$ and return $G_{\text{out}} = (V, E_{\text{out}})$.

Algorithm 2 SMDMD Algorithm

Input: $G = (V, E)$, $k \in \mathbb{N}$
Output: $G_{\text{out}} = (V, E_{\text{out}})$

1: Set $G_0 \leftarrow G$ and $m = 0$.
2: Apply the MDMD algorithm (Algorithm 1) to G_m to get a new graph G_{m+1}.
3: Set $m \leftarrow m + 1$.
4: If $m = k$ then return G_m and stop. Otherwise go back to Step 2.

It first selects one vertex i^* which has the minimum degree and the minimum EC among all vertices with the same degree. It then selects another vertex j^* which has the maximum distance from i^* and the minimum EC among all vertices with the same distance from i^*. It finally returns the edge $\{i^*, j^*\}$. The main advantage of the MDMD algorithm is its low computational cost because the computation of the algebraic connectivity is not needed. The MDMD algorithm is formally described as Algorithm 1.

Although the MDMD algorithm was developed for the special case of Problem 1 where $k = 1$, we can easily extend it to obtain a new algorithm that can be used for the general case. The idea is to apply the MDMD algorithm sequentially. We thus call this new algorithm the sequential MDMD (SMDMD) algorithm. The SMDMD algorithm is formally described as Algorithm 2.

4. Proposed Genetic Algorithm

In this section, we propose a genetic algorithm for finding an approximate solution of Problem 1.

4.1. Overview of Algorithm

In the proposed algorithm, each graph is encoded into a binary sequence called a chromosome by concatenating rows in the strict upper triangular part of the adjacency matrix, as shown in Fig. 1. Also, the fitness value of each chromosome is defined by the algebraic connectivity of the corresponding graph.

Given a graph $G = (V, E)$ and an integer k satisfying $1 \leq k \leq |V|(|V| - 1)/2 - |E|$, the algorithm first generates a population of M chromosomes corresponding to M graphs selected at random from $\mathcal{G}(G, k)$. Each chromosome can be easily generated by randomly selecting k bits with a value of 0 in the chromosome $\boldsymbol{s}_G$ corresponding to G and

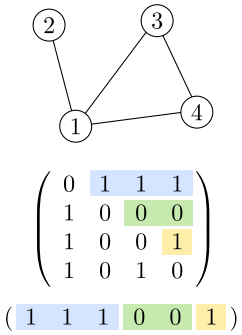

Figure 1: Encoding a graph into a chromosome (upper: graph, middle: adjacency matrix, lower: chromosome).

then flipping the selected bits to 1. Once the initial population of chromosomes is generated, the algorithm then iteratively updates the population in the way explained in the next subsection until some stopping condition is met. In what follows, each iteration is called a generation, and the population in the t-th generation is denoted by $P(t)$ ($P(1)$ is the initial population). Finally the algorithm returns the graph corresponding to the chromosome with the highest fitness value in the final population.

4.2. How to Update Population

Given the population $P(t)$ of M chromosomes, the proposed algorithm performs the following three steps to obtain $P(t + 1)$ of M chromosomes: selection, crossover and mutation. The algorithm also uses the elitist selection.

In the selection step, a new population $P_1(t)$ of M chromosomes is generated from $P(t)$. Suppose that $P_1(t)$ is initialized to $\emptyset$. First, one chromosome with the highest fitness value is selected from $P(t)$ and added to $P_1(t)$. Next, $M - 1$ chromosomes are selected (with repetition allowed) from $P(t)$ by a tournament selection rule and added to $P_1(t)$. The tournament selection rule used in our algorithm is to select one chromosome with the highest fitness value among ℓ chromosomes selected at random from $P(t)$.

In the crossover step, a new population $P_2(t)$ of M chromosomes is generated from $P_1(t)$. Suppose that $P_2(t)$ is initialized to $\emptyset$. First, the chromosome first added to $P_1(t)$ is selected and added to $P_2(t)$. Next, $\lfloor (M - 1)/2 \rfloor$ pairs of chromosomes are randomly made from the remaining $M-1$ chromosomes. For each pair, a single-point crossover is performed with probability $p_c \in (0, 1)$. However, the crossover point cannot be chosen freely because the resulting two chromosomes must have the same number of ones as their parents. So Algorithm 3 is used to perform a single-point crossover and the obtained two chromosomes are added to $P_2(t)$. If M is even then the chromosome in $P_1(t)$ that was not used to make a pair is added to $P_2(t)$.

In the mutation step, a new population $P_3(t) = P(t + 1)$ of M chromosomes is generated from $P_2(t)$. Suppose that $P_3(t)$ is initialized to $\emptyset$. Let $\boldsymbol{s}_G$ be the chromosome

Algorithm 3 Single-Point Crossover

Input: $\boldsymbol{s}_i = (s_{i1}, \ldots, s_{iN}) \in \{0,1\}^N$ $(i = 1, 2)$, $p_c \in (0, 1)$
Output: $\boldsymbol{s}'_i = (s'_{i1}, \ldots, s'_{iN}) \in \{0,1\}^N$ $(i = 1, 2)$

1: Pick a number r from a uniform distribution in $[0, 1]$.
2: If $r > p_c$ then set $\boldsymbol{s}'_i \leftarrow \boldsymbol{s}_i$ for $i = 1, 2$ and go to Step 5. Otherwise go to Step 3.
3: Select j^* uniformly at random from $\{1, 2, \ldots, N\}$.
4: If $\sum_{j=j^*}^{N} s_{1j} = \sum_{j=j^*}^{N} s_{2j}$ then go to Step 6. Otherwise go to Step 5.
5: Set $j^* \leftarrow j^* + 1$ and go to Step 4.
6: Set

$$s'_{1j} \leftarrow \begin{cases} s_{1j}, & \text{if } j \in \{1, 2, \ldots, j^* - 1\}, \\ s_{2j}, & \text{if } j \in \{j^*, j^* + 1, \ldots, N\}, \end{cases}$$

$$s'_{2j} \leftarrow \begin{cases} s_{2j}, & \text{if } j \in \{1, 2, \ldots, j^* - 1\}, \\ s_{1j}, & \text{if } j \in \{j^*, j^* + 1, \ldots, N\}. \end{cases}$$

7: Return $\boldsymbol{s}'_1$ and $\boldsymbol{s}'_2$ and stop.

Algorithm 4 Mutation

Input: $\boldsymbol{s} = (s_1, \ldots, s_N) \in \{0,1\}^N$, $I_G \subset \{1, 2, \ldots, N\}$, $p_m \in (0, 1)$
Output: $\boldsymbol{s}' = (s'_1, \ldots, s'_N) \in \{0,1\}^N$

1: Set $\boldsymbol{s}' \leftarrow \boldsymbol{s}$.
2: Set $f \leftarrow 0$ and $i \leftarrow 1$.
3: If $i \in I_G$ then go to Step 7. Otherwise go to Step 4.
4: Pick a number r from a uniform distribution in $[0, 1]$.
5: If $r \leq p_m$ then set $s'_i \leftarrow 1 - s'_i$ and $f \leftarrow f + 1$ and go to Step 6. Otherwise go to Step 7.
6: Select $j \in \{1, 2, \ldots, N\} \setminus (I_G \cup \{i\})$ such that $s'_j = s'_i$ at random and then set $s'_j \leftarrow 1 - s'_j$.
7: Set $i \leftarrow i + 1$. If $i = N + 1$ or $f = 2$ then return $\boldsymbol{s}'$ and stop. Otherwise go to Step 3.

corresponding to G and let $I_G = \{i \in \{1, 2, \ldots, N\} \mid s_{Gi} = 1\}$. Let $p_m \in (0, 1)$ be the mutation probability. First the chromosome first added to $P_2(t)$ is added to $P_3(t)$. Next, for each of the remaining $M - 1$ chromosomes $\boldsymbol{s}$ in $P_2(t)$, a new chromosome $\boldsymbol{s}'$ is obtained by Algorithm 4 and added to $P_3(t)$. Note here that $\boldsymbol{s}'$ not only satisfies $s'_i = 1$ for all $i \in I_G$ but also has the same number of ones as $\boldsymbol{s}$.

5. Experiments

In order to evaluate the performance of the proposed genetic algorithm in comparison with the SMDMD algorithm, we conducted experiments using 20 synthetic graphs: 10 Watts-Strogatz models [7] with 10 vertices and 20 edges and 10 Barabási-Albert models [8] with 10 vertices and 12 edges. The parameters in the proposed algorithm were set as $M = 50$, $\ell = 5$, $p_c = 0.8$ and $p_m = 0.05$. Also, the proposed algorithm stops when the number of generation t reaches 200. Both algorithms were implemented in Scilab 5.5.2 and executed on a PC with Intel Core i5-4670 and 8GB RAM.

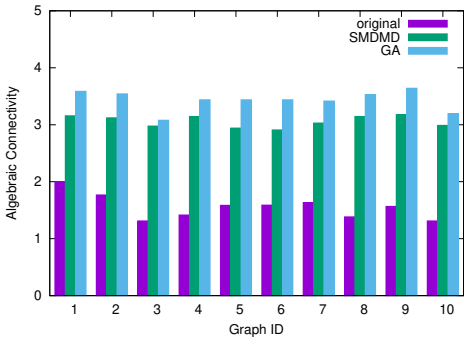

Figure 2: Algebraic connectivity obtained by the two algorithms for Watts-Strogatz models with 10 vertices and 20 edges and $k = 5$. The plots show the average of 10 runs.

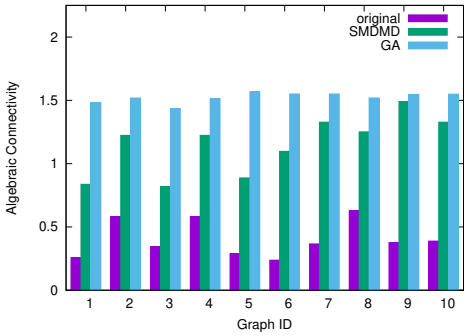

Figure 3: Algebraic connectivity obtained by the two algorithms for Barabási-Albert models with 10 vertices and 12 edges and $k = 5$. The plots show the average of 10 runs.

In the first experiment, we applied the two algorithms to the problem of maximizing the algebraic connectivity of Watts-Strogatz models with 10 vertices and 20 edges by adding five edges. The results are summarized in Fig. 2. It is easy to see that both algorithms increased the algebraic connectivity by about two times and that the proposed algorithm achieved a higher algebraic connectivity than the SMDMD algorithm for all graphs.

In the second experiment, we applied the two algorithms to the problem of maximizing the algebraic connectivity of Barabási-Albert models with 10 vertices and 12 edges by adding five edges. The results are summarized in Fig. 3. The proposed algorithm achieved a higher algebraic connectivity than the SMDMD algorithm for all graphs. In particular, the algebraic connectivity obtained by the former is nearly two times higher than the latter for some graphs. An example is shown in Fig. 4.

In the third and fourth experiments, we applied the two algorithms to the problem of maximizing the algebraic connectivity of Watts-Strogatz models with 10 vertices and 20 edges by adding 15 edges, and the problem of maximizing the algebraic connectivity of Barabási-Albert models with

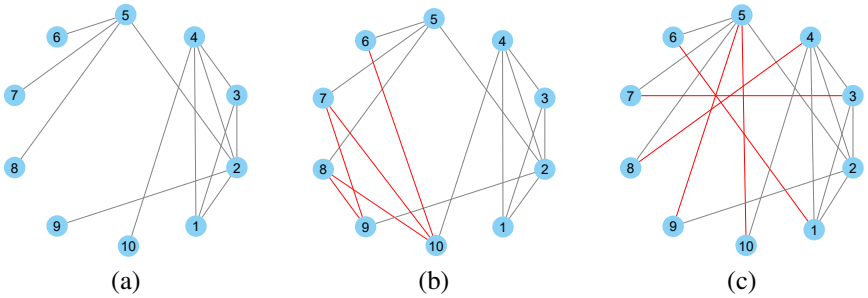

Figure 4: Comparison of the graphs obtained by the two algorithms. (a) Input ($\lambda_2 = 0.2881471$). (b) Graph obtained by the SMDMD algorithm ($\lambda_2 = 0.887342$). (c) Graph obtained by the proposed algorithm ($\lambda_2 = 1.592531$).

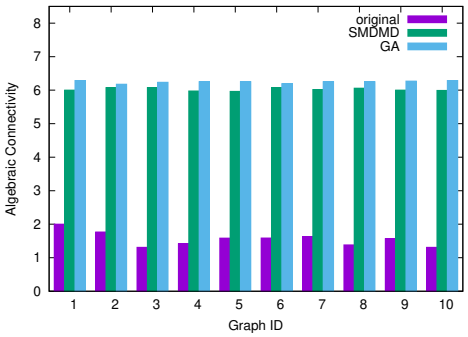

Figure 5: Algebraic connectivity obtained by the two algorithms for Watts-Strogatz models with 10 vertices and 20 edges and $k = 15$. The plots show the average of 10 runs.

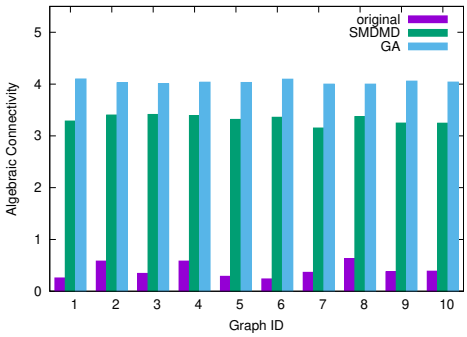

Figure 6: Algebraic connectivity obtained by the two algorithms for Barabási-Albert models with 10 vertices and 12 edges and $k = 15$. The plots show the average of 10 runs.

10 vertices and 12 edges by adding 15 edges, respectively. The results are summarized in Figs. 5 and 6. Both algorithms increased the algebraic connectivity by more than three times for all graphs. Also, the proposed algorithm achieved a higher algebraic connectivity than the SMDMD algorithm for all graphs.

6. Conclusions

We proposed a genetic algorithm for solving the problem of maximizing the algebraic connectivity of a given graph by adding a prescribed number of edges. We then showed experimentally that the proposed algorithm is superior to the SMDMD algorithm in terms of the algebraic connectivity value. However, because only a small number of graphs was used in the experiments, we need to evaluate the performance of the proposed algorithm through further experiments using graphs of various types and sizes. Also, we need to decrease the computational cost of the proposed algorithm so that it can be applied to large graphs.

References

[1] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol.23, no.98, pp.298–305, 1973.

[2] C. Godsil and G. Royle, *Algebraic Graph Theory*, Springer New York, 2001.

[3] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," Proceedings of the IEEE, vol.95, no.1, pp.215–233, January 2007.

[4] K. Ogiwara, T. Fukami and N. Takahashi, "Maximizing algebraic connectivity in the space of graphs with a fixed number of vertices and edges, *IEEE Transactions on Control of Network Systems*, vol.4, no.2, pp.359–368, June 2017.

[5] G. Li, Z. F. Hao, H. Huang and H. Wei, "Maximizing algebraic connectivity via minimum degree and maximum distance," *IEEE Access*, vol.6, pp.41249–41255, 2018.

[6] D. E. Goldberg, *Genetic Algorithms*, Pearson Educational India, 2006.

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

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