MATCH
Communications in Mathematical and in Computer Chemistry

MATCH Commun. Math. Comput. Chem. **71** (2014) 437-443

ISSN 0340 - 6253

Computing Frustration Index Using Genetic Algorithm

Z. Seif , M. B. Ahmadi

Department of Mathematics, College of Sciences,

Shiraz University 71454, Shiraz, Iran

zseif@shirazu.ac.ir , mbahmadi@shirazu.ac.ir

(Received June 10, 2012)

Abstract

Edge frustration index of a graph is defined as the smallest number of edges that have to be deleted from the graph to obtain a bipartite spanning subgraph. Bipartite subgraph problem has many remarkable applications in various fields. It was claimed that the chemical stability of fullerenes is related to the minimum number of vertices or edges that need to be deleted to make a fullerene graph bipartite. The aim of this article is to find the frustration index of different graphs using a mathematical programming model and genetic algorithm.

1. Introduction

A graph $G = V, E)$ is a combinatorial object consisting of an arbitrary set $V = \prime(G)$ of vertices and a set $E = \Sigma(G)$ of unordered pairs $i, j = i$ of distinct vertices of G called edges. A simple graph is a graph that does not have more than one edge between any two vertices and no edge starts and ends at the same vertex. Two vertices are adjacent if there is an edge between them. The adjacency matrix of a simple graph is a matrix with rows and columns labeled by graph vertices, with a 1 or 0 in position (i, j) according to whether vertices v_i and v_j are adjacent or not. G is bipartite if V can be partitioned into two subsets V_1 and V_2 such that all edges have one endpoint in V_1 and the other in V_2. A subgraph S of G is a graph

whose set of vertices and set of edges are all subsets of G. A spanning subgraph is a subgraph that contains all the vertices of the original graph.

An edge $e \in$? is frustrated with respect to a given bipartition (V_1, V_2) of V if both endpoints of e belong to the same class of the bipartition. Frustration index of a graph G, denoted by $\varphi\ G)$, is the minimum number of frustrated edges over all possible bipartitions of V. Alternatively, $\varphi\ G)$ is the smallest cardinality of a set of edges of G that need to be deleted to obtain a bipartite spanning subgraph. Finding the edge frustration index is hard, in fact, it is NP-hard. The edge frustration index is important in a model of spin glasses, the mixed ising model. Also, this index was introduced and considered in the context of complex networks [1]. The edge frustration index has application in chemistry. In [2] Fajtlowicz claimed that the chemical stability of fullerenes is related to the minimum number of vertices/edges that need to be deleted to make a fullerene graph bipartite. There is some correlate studies in [3, 4].

A genetic algorithm (GA) is a search technique used in computing exact or approximate solutions to optimization and search problems. Its original idea comes from Darwinian's evolution theory. The concept of GA was given by Holland [9]. It was first used to solve optimization problem by De-Jong's [10]. A detailed implementation of GA could be found in Goldberg [11].

The focus of this article is upon the formation of an integer-linear mathematical programming model and a genetic algorithm (GA) for computing the edge frustration index of a graph G. Section 2 presents a binary integer linear mathematical programming model. There is a genetic algorithm approach to edge frustration index in section 3. Section 4 contains some numerical examples.

2. Binary integer linear program

A Binary Integer Linear Program is a linear program where the decision variables can take on only two values, 0 and 1 [5]. This section presents a binary integer mathematical programming model whose objective function is equal to the edge frustration index of a graph G. The point of departure is the mixed integer linear programming model developed by Klein and Aronson [6].

Let $G(V,E)$ be a simple graph on n vertices labeled with $v_1,\ldots,v_n$. Let C be its adjacency matrix with c_{ij} in position (i,j). The problem is to find bipartition (V_1,V_2) of V whose total number of frustrated edges is minimized. To formulate the problem, let us denote the decision variables as follows:

for $1 \le i \le n$ and $1 \le k \le 2$, let

$$x_{ik} = \begin{cases} 1 & v_i \in V_k \\ 0 & v_i \notin V_k \end{cases} \tag{1}$$

for $1 \le i \le n$ and $i < j \le n$ define

$$y_{ij} = \begin{cases} 1 & \text{if } v_i \text{ and } v_j \text{ belong to the same partition} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The objective is to find the minimum number of frustrated edges over all possible bipartitions of V. This objective can be stated mathematically as

$$\text{Minimize} \sum_{i=1}^{n} \sum_{j>i} c_{ij}\, y_{ij} \tag{3}$$

The objective function incorporates the total number of frustrated edges with respect to the bipartition (V_1,V_2).

We should now define the constraints of the problem, which are the restrictions imposed upon the values of the decision variables by the characteristics of the problem under study. In order to enforce y_{ij} on assuming value 1 when v_i and v_j are in the same partition and 0 otherwise, we can use constraint

$$y_{ij} \ge x_{ik} + x_{jk} - 1 \tag{4}$$

where $1 \le i \le n$, $i < j \le n$ and $1 \le k \le 2$. It should be noticed that each vertex must belong to one of the partitions. This restriction can be expressed as follows:

$$x_{i1} + x_{i2} = 1 \qquad \text{for } 1 \le i \le n. \tag{5}$$

The model will be

$$\text{Minimize} \sum_{i=1}^{n} \sum_{j>i} c_{ij}\, y_{ij}$$

$$\text{Subject to}:$$

$$y_{ij} \ge x_{ik} + x_{jk} - 1 \qquad \text{for } 1 \le i \le n,\ j > i \text{ and } 1 \le k \le 2$$

$$\sum_{k=1}^{2} x_{ik} = 1 \qquad for\ 1 \le i \le n$$

$$x_{ik} \in \{0,1\} \qquad for\ \ 1 \le i \le n\ \ and\ 1 \le k \le 2$$

$$y_{ij} \in \{0,1\} \qquad for\ \ 1 \le i \le n\ and\ \ j > i$$

Although this model can be entered in different optimization softwares, solving the model to optimality may require an enormous amount of computer time when n increases. In practice that amount of computer time is not always available. In the next section we will describe a genetic algorithm to compute the edge frustration index. Its goal is to find a solution in a relatively short time.

3. The Proposed Genetic Algorithm

In this section, there is a genetic algorithm approach to edge frustration index. This technique works by taking an initial population of individuals and applying genetic operators in each production. In optimization terms, each individual in the population is encoded into a string or chromosomes or some float number which represents a possible solution to a given problem. The fitness of an individual is evaluated with respect to a given objective function. Highly fit individuals or solutions are given opportunities to reproduce by exchanging some of their genetic information, in a crossover procedure, with other highly fit individuals. This produces new offspring solution, which share some characteristics taken from both parents. Mutation is often applied after crossover by altering some genes or perturbing float numbers. The offspring can either replace the whole population or replace less fit individuals. This evaluation-selection-reproduction cycle is repeated until some stopping criteria are met [11].

3. 1. Details of the proposed genetic algorithm

Let $G(V, E)$ be a simple graph on n vertices labeled with $v_1, \ldots, v_n$. Let C be its adjacency matrix with c_{ij} in position (i, j). The optimization problem is to find the bipartition (V_1, V_2) of V with minimum number of frustrated edges. So, each bipartition (V_1, V_2) of V is a feasible solution of our optimization problem. Each potential solution for this problem can be represented as a binary array consisting of n bits. Each bit belongs to a vertex of G and the value of the bit (0 or 1) shows whether the vertex is in the first partition of V or in the second one (figure 1).

1	2	3	4	5	6	…	*n*
0	1	1	0	1	0		1

Fig. 1. A sample solution that shows the vertex belongs to the first or second partition.

The fitness of a solution is the total number of its frustrated edges, i.e. the number of edges whose both endpoints have the same value. Assume that array A is a chromosome that represents bipartition (V_1, V_2) of V. In this case, for $1 \le i \le n$ we have

$$A[i] = \begin{cases} 1 & v_i \in V_1 \\ 0 & v_i \in V_2 \end{cases}$$

the fitness of A can be compute as follows:

$$f(A) = \frac{1}{2}\left(\sum_{i \in B_1} \sum_{j \in B_1} c_{ij} + \sum_{i \in B_2} \sum_{j \in B_2} c_{ij} \right)$$

where $B_1 = \{1 \le i \le n \mid v_i \in V_1\}$ and $B_2 = \{1 \le i \le n \mid v_i \in V_2\}$. It is obvious that $f(A)$ is equal to the total number of frustrated edges with respect to the bipartition (V_1, V_2).

In this genetic algorithm, individuals are selected based upon the tournament selection, and then a single point crossover is performed. In the One-point crossover a single crossover point on both parents' organism strings is selected. All data beyond that point in either organism string is swapped between the two parent organisms. The resulting organisms are the children.

The mutation operator is accomplished by random selection of a set of genes and flipping their values from 1 to 0 or vice versa. The mutation rate is 0.15.

The algorithm terminates when

$$iter > iter\max$$

$iter$ is the current number of generation and $iter\max$ denotes the maximum number of generation allowed.

This genetic algorithm is coded in MATLAB. Some numerical examples are available in the next section.

4. Computational Results

We have prepared a MATLAB code for computing the edge frustration index of an arbitrary graph using genetic algorithm. This program ran for the following graphs (Fig. 2). The population size at each generation was 100. The maximum number of generation was equal to 1000. The results are available in table 1.

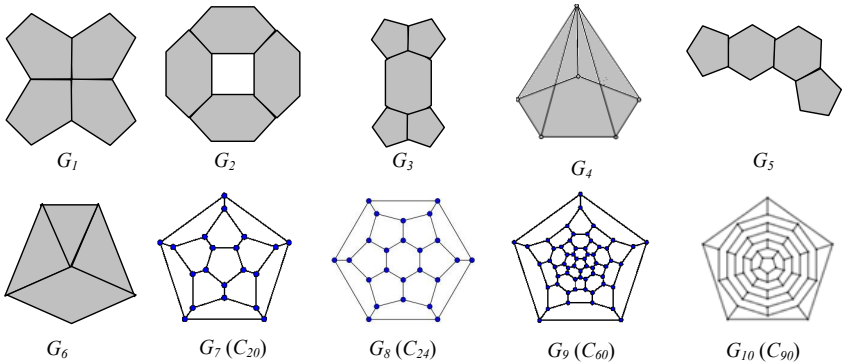

Fig. 2. Graphs considered in this section

Table 1

Graph	$\varphi(G)$ computed using GA
G_1	2
G_2	0
G_3	2
G_4	3
G_5	2
G_6	2
G_7	6
G_8	6
G_9	12
G_{10}	6

References

[1] P. Holme, F. Liljeros, G. R. Edling, B. J. Kim, Network bipartivity, *Phys. Rev. E* **68** (2003) 056107.

[2] S. Fajtlowicz, C. E. Larson, Graph–theoretic independence as a predictor of fullerene stability, *Chem. Phys. Lett*. **377** (2003) 485–490.

[3] T. Došlić, D. Vukičević, Computing the bipartite edge frustration of fullerene graphs, *Discr. Appl. Math.* **155** (2007) 1294–1301.

[4] J. Yazdani, A. Bahrami, Frustration index of HAC_5C_7 nanotorus, *Digest J. Nanomat. Biostruct.* **4** (2009) 881–883.

[5] M. S. Bazara, J. J. Jarvis, H. D. Sherali, *Linear Programming and Network Flows,* Wiley, New York, 2005.

[6] J. E. Aronson, G. Klein, A clustering algorithm for computer-assisted process organization, *Dec. Sci.* **20** (1989) 730–745.

[7] J. Bisschop, M. Roelofs, *AIMMS Language Reference,* Paragon Decision Technology, 2008.

[8] J. Bisschop, M. Roelofs, *AIMMS User's Guide*, Paragon Decision Technology, 2008.

[9] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Univ. Michigan Press, Ann Arbor, 1975.

[10] K. A. De-Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Univ. Michigan, PhD thesis, 1975.

[11] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, New York, 1989.

[12] R. L. Haupt, S. E. Haupt, *Practical Genetic Algorithm*, Technology & Engineering, 2004.