I Krajowa Konferencja
Algorytmy Ewolucyjne
Murzasichle, 12-15.06.1996

Results of evolutionary maximization of the number of spanning trees in directed circulant graphs

Jarosław Arabas, Jacek Wojciechowski
Institute of Electronics Fundamentals,
Warsaw University of Technology
{jarabas, jwojc}@ipe.pw.edu.pl

Abstract The paper is devoted to a fundamental problem arising during the design of optimal networks - maximization of the number of spanning trees. To make the problem tractable, we considered the class of circulant directed graphs. We formulated the design problem as a discrete optimization task. To solve the problem, we used an Evolutionary Algorithm with Varying Population Size, and introduced problem-specific genetic operators. To test the introduced approach, we performed number of experiments with different graph structures; the results are reported in this paper.

1. Introduction

The problem of counting spanning trees in a graph is a fundamental problem in enumerative combinatorics. The number of spanning trees is considered to be a measure of graph structural complexity [3]. The aim of a graph synthesis is to design a graph with the fixed number of points and lines that maximizes the number of spanning trees.

Synthesis of optimal graph is a preliminary step in the design of optimal networks. Let us assume that a certain attribute is assigned to each of the lines of a graph. This attribute could be, for example, the probability of line failure. The objective of a network design is to find such a placement of lines that the probability that a graph representing the network is connected (i.e. the probability that there exists a connection between any pair of points), is maximized. If we assume that the probabilities of line failures are independent and identical, then the problem of designing of optimal network is reduced to the problem of synthesis of a graph maximizing the number of spanning trees [2].

Synthesis of a graph with the maximum number of spanning trees is, in general case, the NP hard problem. In this work we restrict ourselves to the class of regular graphs: the eigenvalue technique provides effective means for counting the number of spanning trees in such graphs. The class of regular graphs is further limited to the class of circulant graphs to make the problem of synthesis numerically tractable. Synthesis of optimal circulant graphs is formulated in this paper as a discrete optimization problem, and solved with the use of the Evolutionary Algorithm with Varying Population Size.

2. Definitions and Notation

We shall use the basic terminology as in [5]. $G = (V,X)$ denotes an undirected or directed graph without loops having p points and q lines. Points are identified by the numbers $0, 1, \ldots, p-1$. The adjacency matrix $\mathbf{A}$ of a graph G is a $p \times p$ matrix with the entry a_{ij} equals 1 if there is a line directed from the ith to the jth point and 0 otherwise. If G is an undirected graph then obviously, $\mathbf{A}$ is a symmetrical matrix, i.e. $\mathbf{A} = \mathbf{A}^T$. The characteristic polynomial of G is defined as $F(\lambda) = \det(\lambda\mathbf{I}-\mathbf{A})$, where $\mathbf{I}$ is the identity matrix. Roots $\lambda_1, \lambda_2, \ldots, \lambda_p$ of $F(\lambda)$ are called the eigenvalues of G.

The degree d_l of the *lth* point of an undirected graph, the out-degree d_l^+ and the in-degree d_l^- of the *lth* point of a directed graph are defined in the following manner:

$$d_l = \sum_{j=1}^{p-1} a_{lj}, \qquad d_l^+ = \sum_{j=1}^{p-1} a_{lj} \qquad d_l^- = \sum_{j=1}^{p-1} a_{jl}, \tag{1}$$

i.e. the degree of a point is the number of lines incident with the point, the out- and in- degrees are, respectively, the numbers of lines directed away, and to the point. If degrees of all points of an undirected graph are the same and equal d, then G is called the *regular undirected* graph of degree d. If p is even then d can be either even or odd; if p is odd then the value of d must be even. If the degrees of a directed graph satisfy the condition: $d_l^+ = d_l^- = d$ ($l=0,1,...,p-1$), then G is called the *regular directed* graph of degree d.

The spectrum $\{ \lambda_1, \lambda_2, \ldots, \lambda_p \}$ of a regular graph has the following properties [3]:

- $$\sum_{k=1}^{p} \lambda_k = 0, \tag{2}$$
- the degree d of G is a c-fold eigenvalue, c - the number of connected components of G. If G is connected, then d is a simple eigenvalue,
- magnitudes of all eigenvalues do not exceed d, i.e.
  $$|\lambda_k| \le d, \qquad k = 1,2,...,p. \tag{3}$$

If G is an undirected graph then all eigenvalues are real; if it is directed then eigenvalues are either real or appear in complex conjugate pairs.

A spanning tree of an undirected connected graph is a connected subgraph of G spanned over all points of G and containing no cycles. A spanning in-tree of a directed connected graph with respect to the *lth* point is a directed connected graph spanned over all points of G such that the *lth* point has out-degree *0* and all others have out-degrees *1* [5]. Similarly, a spanning out-tree with respect to the *lth* point is a connected subgraph spanned over all points of G such that the *lth* point has in-degree *0* and all others have in-degrees *1*. If G is a regular directed graph then the numbers of spanning in-trees and out-trees with respect to any point are the same, i.e. both classes of regular graphs, undirected and directed , have similar tree counting properties.

In this paper we consider a special class of regular graphs, i.e. circulant graphs. A graph is said to be circulant if its adjacency matrix **A** is circulant. A circulant matrix is obtained by taking an arbitrary first row $a_0, a_1, ..., a_{p-1}$ and shifting it cyclically to the right to obtain the successive rows. We consider graphs without loops, and $a_0 = 0$. Obviously, circulant graph is regular of degree $d = a_1 + a_2 + ... + a_{p-1}$. If G is undirected then the following symmetry condition holds: $a_l = a_{p-l}$, $l = 1,2,...,p-1$. Moreover, if d is odd then $a_{\frac{p}{2}} = 1$.

3. Tree counting formula

The number of spanning trees of a directed or undirected regular graph can be determined with the use of a closed form formula in terms of the graph eigenvalues. Let us assume that the eigenvalues of G are ordered in such a way that $\lambda_p = d$. The number of spanning trees can be calculated with the use of the formula [3,6]:

$$t(G) = \frac{1}{p} \prod_{k=1}^{p-1} (d - \lambda_k). \tag{4}$$

In general case the problem of finding eigenvalues of a matrix is at least as complex as finding the number of spanning trees with the use of any other tree counting formula, e.g. the matrix-tree theorem [7].

Formula (4) becomes numerically effective for circulant graphs. There exists a closed form formula for calculating eigenvalues of circulant graphs [8,9]:

$$\lambda_k = \sum_{l=1}^{p-1} a_l \exp(\frac{2\pi i}{p} kl), \qquad k = 1,2,...,p \tag{5}$$

where i is the imaginary unit on the complex plane. The formula can be simplified for undirected graphs by exploiting the property: $a_l = a_{p-l}$:

$$\lambda_k = \begin{cases} 2\sum_{l=1}^{r} a_l \cos\left(\frac{2\pi}{p} kl\right) & \text{if } d \text{ is even} \\ 2\sum_{l=1}^{r} a_l \cos\left(\frac{2\pi}{p} kl\right) + (-1)^k & \text{if } d \text{ is odd} \end{cases} \tag{6}$$

where $r = \frac{p}{2} - 1$ if p is even, and $r = \frac{p-1}{2}$ if p is odd.

Formulae (5,6) together with (4) provide means for effective computation of the number of spanning trees in circulant graphs.

4. Synthesis of optimal graph - problem formulation

The aim of synthesis of the optimal graph is to design a circulant graph that maximizes the number of spanning trees. Let us assume that p and d are given and fixed numbers, and $p \geq 2$. The first row of the adjacency matrix contains exactly d elements with the value *1*. The problem of designing the optimal graph is indeed a problem of combinatorial optimization: how to place d elements with the value *1* in the vector $a = [a_1, a_2, ..., a_{p-1}] \in Z_2^{p-1}$, such the resulting graph maximizes the number of spanning trees. It can be formulated in the following manner:

$$\max\left\{ t(G) = \frac{1}{p} \prod_{k=1}^{p-1} (d - \lambda_k) \right\}, \tag{7}$$

with respect to $a = [a_1, a_2, ..., a_{p-1}] \in Z_2^{p-1}$,

subject to the constraint $\sum_{j=1}^{p-1} a_j = d$,

where λ_k is given by (5,6).

The problem of synthesis of the optimal graph is, in general, NP hard. Several methods can be tried to attack it. One can perform an exhaustive search and find the best solution. This method, although guarantees finding an optimum, is extremely computationally intensive. The number of different graph structures equals $\binom{p-1}{d}$ for the directed graphs. Therefore, an exhaustive search can be effectively applied only when p is small and $d \approx 2$ or $d \approx p-1$. Another possible approach is to apply a nondeterministic algorithm, which no more guarantees finding an optimum, however, the probability to hit the optimum is nonzero. In practice, we could accept a suboptimal solutions with the objective function value reasonably close to the value at the optimum.

In our research, we have chosen an Evolutionary Algorithm with Varying Population Size (EVaPS) as a nondeterministic optimization method. The algorithm maintains population of a variable size, allowing to tune the size to the needs of the problem being solved [1]. We

have also introduced the problem-specific genetic operators to improve the EVaPS performance.

5. Evolutionary algorithm

A sketch of the evolutionary Algorithm with Varying Population Size is given below.

```
procedure EVaPS
begin
    t:=0
    initialize Pop(0)
    evaluate Pop(0)
    while not termination_condition do
    begin
        Off(t):=reproduce (Pop(t))
        recombine (Offs(t))
        evaluate (Offs(t))
        for all a ∈ Pop(t)
            age(a):=age(a)+1
        Pop(t+1):=Pop(t) ∪ Offs(t)
        Pop(t):=Pop(t)\{a ∈ Pop(t) age(a)>lt(a)}
        t:=t+1
    end
end
```

The algorithm maintains population *Pop* of chromosomes. The size of the population can vary over time. In the „initialize *Pop(0)*" step, an initial population is randomly created. During the „evaluate *Pop(t)*" step, an objective function value for each chromosome *a* in the population *Pop(t)* is evaluated. Then, the chromosome lifetime *lt(a)* is calculated according to the formula, called the *lifetime allocation strategy*. The detailed comparative study of different lifetime allocation strategies is provided in [1]. According to the suggestions in [1], we have adopted a bilinear strategy for the experiments. The chromosome is kept in the population until its age *age(a)* exceeds its lifetime *lt(a)* - then the chromosome dies off. During the „reproduce *(Pop(t))*" step, an intermediate population *Offs(t)* is created by copying a constant number of randomly selected chromosomes (probability of selection is the same for each chromosome in the population).

During the „recombine *(Offs(t))*" step, chromosomes in the intermediate population undergo crossover and mutation. Crossover is a binary operator - it takes (randomly selected) two parents, and produces two children using the parental genetic pool. During mutation, a randomly selected parental chromosome undergoes a slight perturbance, yielding one offspring. The detailed description of the genetic operators is provided below.

5.1 Chromosome encoding

A chromosome is encoded as a string of zeros and ones, which is a natural representation of the problem. Let us consider an example of the circulant adjacency matrix:

$$\mathbf{A}=\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

For this matrix, the chromosome equals the first row of the adjacency matrix A (except of its first element):

$a=[1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0]$

The above description fully defines an encoding method for any directed circulant graphs.

5.2 Crossover

We used a special type of crossover, which preserves the number of ones in the chromosome (see [4]). The crossover is performed in the following way:

Firstly, each of the chromosomes being crossed-over is divided into blocks - substrings of adjacent bits. For each block, the number of ones equals the number of ones in the corresponding block in the second parental chromosome.

To make it more formal, let us assume that l and r stand for the leftmost and rightmost position of a block. Each parental chromosome a can be written down as follows:

$a=[a_1,\ldots,a_{l-1},a_l,\ldots,a_r,a_{r+1},\ldots,a_{p-1}]$

Then the above condition can be formulated as

$$\sum_{i=l}^{r} a_i^1=\sum_{i=l}^{r} a_i^2,$$

where $a^1{}_i$ and $a^2{}_i$ denote the i-th bit of the first and second parental chromosome, respectively.

Secondly, corresponding blocks are exchanged with probability *0.5*.(see Fig. 1 for an example).

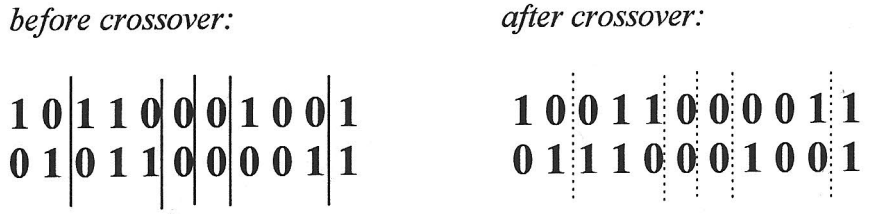

Fig. 1 Example of crossover. Perpendicular lines reveal borders between blocks.

5.3 Mutation

During mutation, two bits from the chromosome are randomly chosen, and their values are interchanged, which guarantees preservation of the number of ones (see [4]).

before mutation: **1 0 1 1 0 0 0 1 0 0 1**

after mutation: **1 0 1 0 0 0 0 1 1 0 1**

Fig. 2. Example of mutation. Arrows point to the bits undergoing mutation.

6. Experiments and results

For the purpose of testing of the evolutionary approach, we performed a series of experiments. In each experiment 10 independent runs of the evolutionary algorithm were performed, and additionally we did an exhaustive search through the whole space of graphs (if it was possible because of the problem dimensionality). The aim was to maximize the number of spanning trees for the directed circulant graphs.

In the following tables, the results of the evolutionary algorithm are listed. These data come from 10 independent EVaPS runs. Value of D equals the number of all possible solutions, whereas value of F equals the number of the objective function calls to obtain the solution, divided by the value of D. Value of E equals the relative error of the solution found by the evolutionary algorithm

$$E = \frac{f_{best} - f_{EVaPS}}{f_{best}},$$

where f_{EVaPS} and f_{best} denote the objective function value for the solution found by the EVaPS and the true optimum. The value of M equals

$$M = \frac{f_{MC}}{f_{EVaPS}},$$

where f_{MC} denotes the best objective function value found by the Monte Carlo method.

For $p=20$ and $p=21$, an additional, exhaustive search through all solutions was performed in order to obtain a full spectrum of the objective function values just to compare the results of EVaPS and of the enumerative method. In all cases not less than 15 first significant digits of the objective function value were identical for the true optimum, and for the solution found by EVaPS. From the results in Tab.1 and Tab.2 it evidences that for a large problem space, it is enough for EVaPS to search through less than 10% solutions to find a very good one (or even the best one).

d	2	4	6	8	10	12	14	18
D	190	4845	38760	125970	184756	125970	38760	190
F	7.45	0.285	0.0972	0.0506	0.0401	0.0178	0.0662	19.04
E	0	0	0	10^{-15}	10^{-16}	10^{-15}	10^{-15}	10^{-16}

Tab. 1 Simulation results for $p=20$.

d	3	5	7	9	11	13	15	17
D	1330	20349	116280	293930	352716	203490	54246	5985
F	0.354	0.101	0.0339	0.0131	0.0146	0.0123	0.0508	0.566
E	0	0	10^{-3}	10^{-6}	10^{-5}	0	0	0

Tab. 2 Simulation results for $p=21$.

Now, let us look at the results for $p \geq 30$. For these values of p, it was computationally too expensive to perform an exhaustive search, and the capacity needs to store the data were extremely high. Therefore, as a comparative data we took the results of a Monte Carlo analysis. We performed a random sampling using the number of samples equal to the number of the objective function calls generated by the EVaPS algorithm. For high values of d, the number of spanning trees exceeded the floating point range. Therefore, the data are incomplete.

d	2	4	6	8	10	12	14	16	18	20
D	435	2.74×10^{4}	5.94×10^{5}	5.85×10^{6}	3.01×10^{7}	8.65×10^{7}	1.45×10^{8}	1.45×10^{8}	8.65×10^{7}	3.01×10^{7}
F	1.15	5.29×10^{-2}	6.64×10^{-3}	1.77×10^{-3}	3.02×10^{-4}	8.39×10^{-5}	9.32×10^{-5}	8.92×10^{-5}	1.27×10^{-4}	2.99×10^{-4}
M	1.0	1.0	1.0	0.95	0.91	0.97	0.99	0.99	0.98	0.97

Tab.3 Simulation results for $p=30$.

d	3	5	7	9	11	13	15	17	19
D	4.49×10^{3}	1.69×10^{5}	2.63×10^{6}	2.02×10^{7}	8.47×10^{7}	2.06×10^{8}	3.01×10^{8}	2.65×10^{8}	1.41×10^{7}
F	0.47	1.37×10^{-2}	2.05×10^{-3}	4.10×10^{-4}	1.12×10^{-4}	5.18×10^{-5}	3.93×10^{-5}	5.20×10^{-5}	7.87×10^{-5}
F	1.0	0.88	0.90	0.89	0.98	0.97	0.99	0.98	0.98

Tab.4 Simulation results for $p=31$.

d	2	4	6	8	10
D	780	9.14×10^{4}	3.84×10^{6}	7.69×10^{7}	8.47×10^{8}
F	2.53	3.02×10^{-2}	1.25×10^{-3}	9.04×10^{-5}	1.18×10^{-5}
M	1.0	1.0	0.84	0.78	0.80

Tab.5 Simulation results for $p=40$.

d	3	5	7	9
D	2128	7.49×10^{5}	2.25×10^{7}	3.50×10^{8}
F	0.19	7.41×10^{-3}	3.08×10^{-4}	2.82×10^{-5}
M	1.0	1.0	0.80	0.80

Tab.5 Simulation results for $p=41$.

d	2	4	6
D	1.22×10^{3}	2.30×10^{5}	1.59×10^{7}
F	0.99	1.34×10^{-2}	2.92×10^{-4}
M	1.0	0.77	0.74

Tab.6 Simulation results for $p=50$.

d	3	5
D	2.08×10^{4}	2.35×10^{6}
F	0.87	1.74×10^{-3}
M	1.0	1.0

Tab.6 Simulation results for $p=51$.

7. Conclusions

The percentage of possible solutions taken into account by the EVaPS algorithm decreases with the increase of the number of solutions for this particular type of a problem. This allows to deal with the great number of solutions within a reasonable time, and indicates that EVaPS is not an enumerative algorithm.

While the number of possible solutions increases, the Monte Carlo method reveals inferior performance than EVaPS. This indicates a better EVaPS convergence.

For simple problems (small p and $d \approx 2$ or $d \approx p-1$) it seems unnecessary to apply EVaPS - either enumerative, or a Monte Carlo method will be powerful enough to find the global optimum.

It should be reminded that solutions obtained by EVaPS are not guaranteed to be the optimal ones, however it seems that for the problem under consideration there exists a set of suboptimal solutions with the objective function value approximately equal to the global maximum.

Acknowledgments

This work was partially supported by the Program in Control, Information Theory and Automation (PATIA) at the Warsaw University of Technology.

References

[1] J. Arabas, „Evolutionary algorithms with varying population size and variable crossover range", Ph.D. dissertation, Warsaw, 1995 (in Polish).

[2] F.Boesch, „Is reliable network synthesis a graph theory problem", Stevens Inst. of Technology, Computer Science Dept., Techn. Report No 8601, 1986.

[3] D.Cvetkovic, M.Doob, and H.Sachs, "Spectra of Graphs'", Academic Press, New York, 1980.

[4] P.D. Domański, J. Arabas, „A Genetic Approach to the Linguistic Modelling", *Proc. 8th Int. Symp. Systems, Modelling, Control*, Zakopane, pp. 224-229, 1995.

[5] F.Harary, "Graph Theory," Addison-Wesley, Reading, MA,1969.

[6] A.Schwenk, and R.Wilson, "On the eigenvalues of a graph," in "Selected topics in Graph Theory" (Edited by Beineke and Wilson), Academic Press, New York, pp.307-336, 1978.

[7] G.Kirchhoff, "Über die Auflösung der Gleichungen, auf welche man bei Untersuchung der linearen Vertulung galvanischer Strome geführt wird," *Ann. Phys. Chem.*, vol.72, pp.497-508, 1847.

[8] J.Wojciechowski, and M.Fellows, "Counting spanning trees in directed regular multigraphs," *J. Franklin Inst.*, vol.326, pp.889-896, 1989.

[9] J.F.Wang, and C.S.Yang, "On the number of spanning trees of circulant graphs," *Int. J. Computer Math.*, vol.16, pp.229-241, 1984.