A GA-Inspired Approach to the Reduction of Edge Crossings in Force-Directed Layouts

Farshad Ghassemi Toosi
Department of CSIS
University of Limerick
Limerick, Ireland
farshad.toosi@ul.ie

Nikola S. Nikolov
Department of CSIS
University of Limerick
Limerick, Ireland
nikola.nikolov@ul.ie

Malachy Eaton
Department of CSIS
University of Limerick
Limerick, Ireland
malachy.eaton@ul.ie

ABSTRACT

We report on our findings using a genetic algorithm (GA) as a preprocessing step for force-directed graph drawings to find a smart initial vertex layout (instead of a random initial layout) to decrease the number of edge crossings in the graph. We demonstrate that the initial layouts found by our GA improve the chances of finding better results in terms of the number of edge crossings, especially for sparse graphs and star-shaped graphs. In particular we demonstrate a reduction in edge-crossings for the class of star-shaped graphs by using our GA over random vertex placement in the order of 3:1.

CCS Concepts

•**Mathematics of computing** → **Combinatoric problems; Combinatorial algorithms; Graph algorithms;**

Keywords

Routing and layout; Genetic algorithms; Fitness evaluation; Combinatorial optimization; Running time analysis

1. INTRODUCTION

The class of force-directed algorithms is the most well-known and accepted approach in the graph visualisation challenge [1]. This class of algorithms is based on the spring-embedder model by Eades [2] which normally starts with a random initial placement of vertices and lets the spring forces move them towards mechanical equilibrium over several iterations. The initial random placement can significantly impact the final layout in terms of edge-crossings [7]. We propose a Genetic Algorithm (AngGA) for preparing smart initial node placements, which lead to reduction in edge crossing in the final layout. This idea was also exploited in previous work [7].

Gecco 2016 July 20–24, 2016, DENVER, COLORADO, USA

ACM ISBN 978-1-4503-4323-7/16/07.

DOI: http://dx.doi.org/10.1145/2908961.2908968

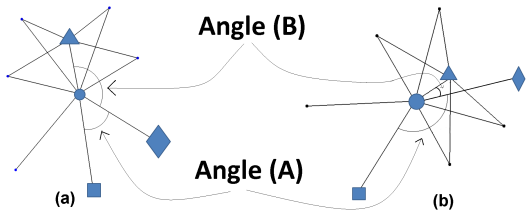

Figure 1: Illustration of angles involved in the fitness function with two different layouts of a star-shaped graph.

2. GENETIC ALGORITHM

AngGA in this paper comprises a fitness function and crossover, mutation, and selection operators. An individual in our GA represents a sequence of cell addresses in a square matrix. Similar to the approach of Eloranta and Mäkinen [3] in their TimGA, we consider a discrete drawing space and represent a layout by an $n \times n$ matrix.

2.1 Fitness

AngGA aims at maximising its fitness function which expresses a weighted sum of angles between adjacent edges. For each pair of adjacent edges, we consider the smaller angle enclosed by these edges. We reserve more space for angles at vertices with high betweenness centrality so that adjacent edges can have longer length. The fitness of layout L of graph $G = (V, E)$ with a set of vertices $V = \{v_1, v_2, \ldots, v_n\}$ and a set of edges $E \subseteq V \times V$ is expressed in Equation (1) where a_{ijk} is the angle between edges $\{v_i, v_j\}$ and $\{v_i, v_k\}$ and the weights b_i, b_j and b_k are the betweenness centralities of vertices v_i, v_j and v_k, respectively; e_{ij} are e_{ik} are the lengths of the two links which create the angle between edges $\{v_i, v_j\}$ and $\{v_i, v_k\}$.

$$f(L) = \sum_{v_i \in V} \sum_{\substack{v_j, v_k \in V \\ \{v_i, v_j\} \in E \\ \{v_i, v_k\} \in E}} a_{ijk} b_i b_j b_k e_{ij} e_{ik} \qquad (1)$$

Fig. 1 shows two different layouts of the same graph taken from the Rome Graphs dataset [1]. The layout in Fig. 1(a) is better in terms of edge crossings than the one in Fig. 1(b). In both layouts the *triangle* and *circle* vertices have higher betweenness centrality than the *diamond* and *square* vertices.

[1] http://www.graphdrawing.org/data/

Table 1: Results from experiments with 512 graphs from Rome repository.

L	No. *Total*	Avg. n	Avg. m	No. Gen	$GA > R$ *Cross%*	$R > GA$ *Cross%*	$GA > R$ *No.Graphs*	$R > GA$ *No.Graphs*	GA time(s)	R time(s)
L_1	512	9	11.55	90	23%	-0.04%	213	53	0.16	0.0048
L_2	532	37.39	37.22	$n10$	13%	-0.09%	135	121	1.43	0.33
L_3	3638	37.15	41.9	$n5$	13%	-1%	1053	954	1.53	0.32
L_4	1503	26.44	36.67	$n5$	23%	-0.04%	990	293	0.63	0.098

Table 2: Results of applying GA as a pre-processor for Fruchterman-Reingold.

Algorithm	Graph	n	m	Gen	GA-planar	R-planar	GA-cross	R-cross	GA-run (s)	R-run (s)
FR	1	7	7	70	100	75	0	0.25	0.07	0.002
FR	2	39	45	200	79	17	0.21	1.23	1.2	0.25
FR	3	68	84	300	96	88	0.11	0.31	4.415	1.2
FR	4	51	63	300	96	84	0.1	0.63	2.6	0.4
KK	1	7	7	70	93	78	0.07	0.27	0.07	0.004
KK	2	39	45	200	70	34	0.355	0.885	2.015	1.11
KK	3	68	84	300	93	71	0.36	1.26	7.67	5.15
KK	4	51	63	200	66	27	0.81	3.37	2.2	0.9

2.2 General Settings

The population size depends on the graph size, we set the population size to 100 for graphs with node count in the range from 10 to 200. We use an *elitist* technique for selecting individuals to be preserved in the next generation. The whole population is sorted from best to worst fitness and the first third of individuals is selected and copied directly into the next generation(**Reproduction**). The first third and the second third in the list are also reserved for the application of crossover and mutation, while the bottom third of the individuals is discarded. Each ordering in the remaining part (the first and the second thirds of orderings) is modified and removed from the current generation and copied to the next generation by either **Crossover** or **Mutation** with equal probability. For a graph with n vertices we consider kn generations, k being set at a value less than or equal to 10. Then the individual with the best fitness from the last generation is selected to be the initial layout for a force-directed graph drawing algorithm. Since our individual represents a set of interconnected vertices we wish to keep a good part of the layout preserved when performing crossover. Because of its effectiveness in this regard *PMX* has been chosen as our crossover operation. It randomly selects v_i such that $v_i \neq v_{|V|}$ and $v_i \neq v_1$. Then it creates the first offspring from the set $\{v_1...v_i\}$ of the first parent and the set $\{v_{i+1}...v_{|V|}\}$ of the second parent, and the second offspring from the set $\{v_1...v_i\}$ of the second parent and the set $\{v_{i+1}...v_{|V|}\}$ of the first parent.

The mutation operator we chose to use in AngGA is *SingleMutate* [5]. This mutation operator randomly picks two different vertices and moves them to two, also randomly picked, empty cells in the layout matrix.

3. EXPERIMENTAL RESULTS

In order to evaluate our AngGA we ran the force-directed algorithm of Fruchterman and Reingold [4] and Kamada-Kawai [6] on a few selected graphs, both with a randomly generated initial layout and with our AngGA, to compare the number of edge crossings in the final force-directed layouts (see Table 2). We have also run Fruchterman and Reingold [4] on a few bigger sets of graphs with both randomly generated initial layouts and with our AngGA. The first dataset L_1 we used in our experimental study consists of 512 planar graphs from the Rome Graphs dataset with a vertex count of 9 the results are listed in Table 1. The second dataset L_2 is comprised of 532 randomly generated sparse graphs. The third set is a set of 3638 *Cactus* graphs which have also been generated randomly. The last dataset L_4 is a list of 1503 star-shaped graphs, again generated randomly, based on the number of branches in the graph (see Table1).

4. REFERENCES

[1] P. Eades. On the future of graph drawing: Invited talk at the 18th International Symposium on Graph Drawing.

[2] P. Eades. A heuristics for graph drawing. *Congressus numerantium*, 42:146–160, 1984.

[3] T. Eloranta and E. Mäkinen. TimGA - a genetic algorithm for drawing undirected graphs. *Divulgaciones Matematicas*, 9(2):155–170, 2001.

[4] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.

[5] L. J. Groves, Z. Michalewicz, P. V. Elia, and C. Z. Janikow. Methodologies for intelligent systems, 5. chapter Genetic Algorithms for Drawing Directed Graphs, pages 268–276. Elsevier North-Holland, Inc., New York, NY, USA, 1990.

[6] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, 31(1):7–15, April 1989.

[7] F. Ghassemi Toosi, N. S. Nikolov, and M. Eaton. Evolving smart initial layouts for force-directed graph drawing. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, pages 1397–1398. ACM, 2015.