Egyptian Informatics Journal (2012) **13**, 9–17

Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

ORIGINAL ARTICLE

GA for straight-line grid drawings of maximal planar graphs

Mohamed A. El-Sayed *

Mathematics department, Faculty of Science, Fayoum University, Fayoum, Egypt
College of Computers and Information Technology, Taif University, Taif, Saudi Arabia

Received 13 August 2011; revised 24 November 2011; accepted 8 January 2012
Available online 15 February 2012

KEYWORDS

Genetic algorithms;
Graph drawing;
Maximal planar graphs

Abstract A straight-line grid drawing of a planar graph G of n vertices is a drawing of G on an integer grid such that each vertex is drawn as a grid point and each edge is drawn as a straight-line segment without edge crossings. Finding algorithms for straight-line grid drawings of maximal planar graphs (MPGs) in the minimum area is still an elusive goal. In this paper we explore the potential use of genetic algorithms to this problem and various implementation aspects related to it. Here we introduce a genetic algorithm, which nicely draws MPG of moderate size. This new algorithm draws these graphs in a rectangular grid with area $\lfloor 2(n-1)/3 \rfloor \times \lfloor 2(n-1)/3 \rfloor$ at least, and that this is optimal area (proved mathematically). Also, the novel issue in the proposed method is the fitness evaluation method, which is less costly than a standard fitness evaluation procedure. It is described, tested on several MPG.

1. Introduction

Graph drawing problem can be found in many areas of computer sciences, such as in the computer networks, data networks, information visualization, maps [1–5], class interrelationship diagrams in object oriented databases and object oriented programs, visual programming interfaces, database design systems, project management, visual languages and web sites [6–8]. Graph drawing addresses the problem of finding a layout of a graph that satisfies given esthetic and understandability objectives. The most important objective in graph drawing is minimization of the number of crossings in the drawing, as the esthetics and readability of graph drawings depend on the number of edge crossings. VLSI layouts with fewer crossings are more easily realizable and consequently cheaper [9–11].

* Corresponding author at: Mathematical Department, Faculty of Science, Fayoum University, Egypt; College of Computers and Information Technology, Taif University, Taif, Saudi Arabia.
E-mail addresses: mas06@fayoum.edu.eg, m.sayed@tu.edu.sa

Peer review under responsibility of Faculty of Computers and Information, Cairo University.
doi:10.1016/j.eij.2012.01.003

Production and hosting by Elsevier

Graph drawing – also known as network visualization – deals with all aspects of representing relational structures. The automatic generation of graph drawings is of relevance not only in computer science, but in virtually every area concerned with graphical data analysis or visual communication of information. Research field of graph drawing encompasses design and analysis of algorithms and algorithm engineering, as well as modeling aspects, topics from graph theory and combinatorics, and the development of software tools [12]. In what follows we assume that the reader is familiar with the basics of graph drawing as given e.g. in [13]. It is suitable for use in advanced undergraduate and graduate level courses on algorithms, graph theory, graph drawing, information visualization and computational geometry.

Esthetics plays a fundamental role, their optimization aims at facilitating both readability and memorization of the information contained in the graphs. Frequently adopted esthetic criteria include: minimization of the drawing area, minimization of the edge crossing number, minimization of the sum of the arc lengths, etc. However, recent experiments have confirmed that the minimization of edge crossing is by far the most important criterion [14,15].

Many algorithms for graph visualization have been proposed, and an extensive survey was done by Di Battista, Eades and Tamassia. Heuristic approaches were introduced to graphic drawing in earlier researches [16,17], and of which, genetic algorithm (GA) show its excellent flexibility to complex constrains. GA can be a viable alternative to more traditional approaches for graph drawing, but at the same time encounters serious performance issues, which made it hard to be applied to large scale applications [18,19].

GA in [20–22] has been described for solving many problems of graph, for example, graph partitioning problem, and the page number problem. Evolutionary computation techniques especially GAs based on planar graph have been successfully applied to multiple sequence alignment problem [23–26].

A maximal plane graph is one, which cannot have any additional edges without destroying its planarity. Such a graph is also called triangulated since all the faces are triangles. Since every planar graph can be triangulated by adding additional (dummy) edges. A straight-line embedding of a plane graph G is a plane embedding of G in which edges are represented by straight-line segments joining their vertices without any edge-intersection [27]. An example of a planar straight-line drawing is illustrated in Fig. 1.

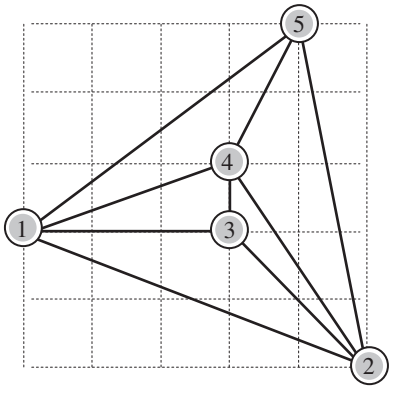


Figure 1 A planar straight-line drawing.

This problem was solved by Chrobak and Payne [28] who proved that, for $n \geqslant 3$, each n-vertex planar graph could be drawn on the $(2n-4)\times(n-2)$ grid. They also presented an linear time algorithm for constructing such embedding. Then, it is shown that every plane graph with $n \geqslant 3$ vertices has a planar straight-line drawing in a rectangular grid with area $(n-2)\times(n-2)$ by two methods. Schnyder's realizer [29,30], "realizer concept", for plane triangulation was invented for compact straight-line drawing of plane graph. Researchers [28,31–36] also obtained similar and other graph drawing results using "the canonical ordering concept". Each method is a constructive proof and yields a simple linear-time algorithm to find such a drawing. Nakano [37] attempted to explain the hidden relation between these two concepts.

The following question "what is an algorithm for the minimum area of the rectangular grid for planar straight-line drawings" remains open up to now. Thus there still exists a gap between upper and lower bounds for the area of the rectangular grid.

This paper introduces a GA which nicely draws planar graphs in a rectangular grid with area at least $\lfloor 2(n-1)/3 \rfloor \times \lfloor 2(n-1)/3 \rfloor$. Our algorithm has been inspired by the ideas from Refs. [8,38–40].

The remainder of the paper is organized as follows. In Section 2, presents the principles of GA, then it is introduce the problem representation and the genetic operators we used. In Section 3, the representations of chromosomes are explained. The selection and the evaluation function are investigated, the genetic operations are explained in Section 4. The parameters of our algorithm, the results are given in Section 5. Finally, conclusions are in Section 6.

2. Genetic algorithms

GAs are a class of randomized optimization heuristics based loosely on the biological paradigm of natural selection. While the exact mechanisms behind natural evolution are not very well known, some aspects have been studied in considerable depth. The general principle underlying GAs is that of maintaining a population of possible solutions, which are often called chromosomes. It is believed that chromosomes are the information carriers and that the evolution process works at the chromosome level through reproduction. The reproduction can be made by either combining chromosomes from the parents to produce offspring, a process called crossover, or by a random change occurring in the chromosome pattern, termed mutation.

Population size is the number of chromosomes used to represent a set of solutions to the problem. In our problem a population is a set of graph layouts. The population undergoes an evolutionary process that imitates the natural biological evolution. In each generation better chromosomes have greater possibilities to reproduce, while worse chromosomes have greater possibilities to die and to be replaced by new individuals.

A GA first creates an initial population of solutions. The solutions are then evaluated, using an application-specific criteria of fitness, to characterize them from most fit to least fit. A subset of the population is selected, using criteria that tend to favor the most fit solutions. This subset is then used to produce a new generation of offspring solutions. Finally, a number of solutions in this new generation are subjected to random muta-

tions. The processes of selection, crossover and mutation are then repeated. A drawback of GAs is that the optima of these problems are generally unknown and it is therefore difficult to assess their performance. Another drawback is that GAs need a simple fitness function with a reasonably fast evaluation to distinguish between "good" and "bad" chromosomes, but this is often not possible. GAs are usually slow, especially because the fitness function evaluation takes a long time.

In graph drawing the evaluation function depends on the esthetic criteria used, our evaluation function is discussed in greater detail in the next section.

A genetic algorithm must have the following five basic components:

1. A genetic representation of solutions to the problem.
2. A way to create an initial population of solutions.
3. An evaluation function rating solutions in terms of their fitness.
4. Genetic operators that alter the genetic composition of children fitness.
5. Values for the parameters of genetic algorithms.

The general structure of a GA [see Fig. 2] is as follows:

```
procedure GA
{
  t: = 0;
  create the initial population P0;
  evaluate the initial population;
  while not Termination-condition do
  {
  t: = t + 1;
  select individuals to be reproduced;
  perform crossover and mutation operations to create the
new population Pt;
  evaluate (Pt)
  }
}
```

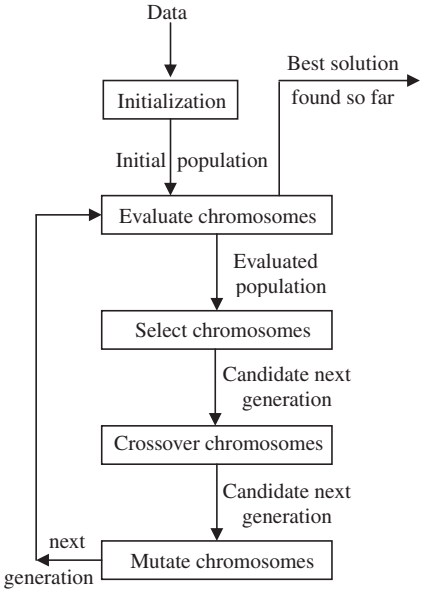


Figure 2 procedure GA and it architecture.

There are several parameters to be fixed. First, we have to decide how to represent the set of possible solutions. In "pure" genetic algorithms only bit string representations were allowed, but we allow any representation that makes efficient computation possible. Second, we have to choose an initial population. We use initial populations created by random selection. Third, we have to design the genetic operations, which alter the composition of children during reproduction. The two basic genetic operations are the mutation operation and the crossover operation. Mutation is an unary operation, which increases the variability of the population by making point wise changes in the representation of the individuals. Crossover combines the features of two parents to form two new individuals by swapping corresponding segments of parent's representations. It turns out that the main problem in genetic graph drawing algorithms is to find efficient crossover operations.

3. Representations of chromosomes

Our algorithm draws planar graphs in a rectangular grid with area $L \times L$. Each node is located in a Cartesian point of the grid and all edges are drawn as straight lines. Chrobak and Nakano [41] obtained mathematically $L = \lfloor 2(n-1)/3 \rfloor$ in the following theorem:

Theorem 1. For each $n \geqslant 3$ there is an n-vertex plane graph G_n such that the width and height of each grid drawing of G_n is at least $\lfloor 2(n-1)/3 \rfloor$.

Proof [41]. It is sufficient to consider the width only. We construct H_n recursively. H_3 is the triangle (v_1, v_2, v_3) and for $n \geqslant 4$, H_n is obtained by adding vertex v_n to the outer face of H_{n-1}, and connecting it to v_{n-3}, v_{n-2}, v_{n-1}, in such a way that the outer face of H_n is (v_n, v_{n-1}, v_{n-2}).

First, notice that for $n = 3$, 4, and 5, H_n requires width 1, 2, 2 respectively, which equals $\lfloor 2(n-1)/3 \rfloor$. The theorem follows by induction, since adding v_{n+1}, v_{n+2} and, v_{n+3} to H_n forces us to use at least two more x-coordinates, see Fig. 3. □

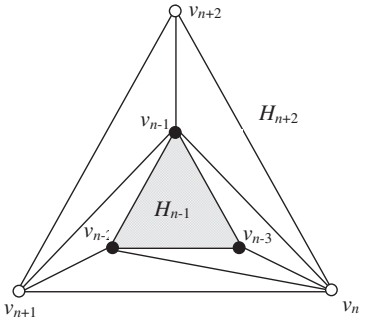


Figure 3 The construction of graph H_n from Theorem 1.

Nodes matrix, n=5

	v_1	v_2	v_3	v_4	v_5
x	0	5	3	3	4
y	2	0	2	3	5

Edges matrix, m=9

e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
1	1	1	1	2	2	2	3	4
2	3	4	5	3	4	5	4	5

Figure 4 The representation of a sample maximal planar graph in Fig. 1.

To represent a graph with n nodes and $m = 3(n-2)$ edges, we use a $2 \times n$ matrix to indicate the positions of the nodes and a $2 \times m$ matrix to indicate the edges by storing pairs of nodes. The corresponding end points are then found from the node matrix. Fig. 4, shows the representation used of a sample example, Fig. 1. Groves et al. [42] have used similar representation for nodes.

Representing a solution of planar graph drawing problem into a chromosome is a key issue when using genetic algorithms. Chromosomes are the strings or arrays of genes (a gene is the smallest building block of the solution). A chromosome can be represented by a string of integers with length n, where n is the number of nodes in planar graph, a gene, g_i is represent the position (x, y) of v_i in grid, and the genes have different values in the chromosome. See Fig. 5.

We select optional face in G, say $(v_1–v_2–v_n)$ to represent the outer face of output layout graph in the grid. Hence, v_1, v_2 and v_n be the nodes to form a largest triangulated face. See Fig. 5. Take $P(v_1) = (0,0)$, $P(v_2) = (L,0)$ and $P(v_n) = (|L/2|, L)$. A gene, g_i of the chromosome, $3 \leqslant i \leqslant n-1$, is represent a grid point $P(v_i) \in D$, where D is a set of Cartesian points, p_1, $p_2, \ldots, p_k$, are lies inside the triangle $(0,0)$-$(L,0)$-$(|L/2|, L)$.

The domain D can be generated using the following simple code:

```
k = 0;
for y = 1 to L − 1
{ if L even then c = |y/2| + 1
        else c = |(y − 1)/2| + 1;
  for x = c to L − |y/2| − 1
    { k = k + 1; p_k = (x,y);};
};
D = {(1,1), (2,1), (3,1),...  (L −3,1), (L −2,1), (L −1,1),
     (2,2), (3,2), (4,2),...  (L −2,3)
     (2,3), (3,3), (4,3),...  (L −2,3)
     (3,4),...  (L −3,4)
     (3,5),...  (L −3,4)
     ...
     (|L/2|, L −2)
     (|L/2|, L −1)}.
```

4. Genetic operations

Here we describe the following operations: selection process, fitness function, crossover and mutation operations in our algorithm. The selection process directs the genetic algorithm towards promising regions in the search space. One of the selection method is used the *linear normalization* suggested by Davis [32] together with *elitism*. The linear normalization works as follows. The chromosomes are sorted in decreasing order by their evaluation function values. The best chromosome gets a certain constant value (e.g. 100) and the other chromosomes get stepwise decreasing constant values (e.g. 98, 96, 94, etc.). Chromosomes are then selected to the genetic operations proportionally to the values so obtained. In our problem instead of using stepwise decreasing constant our selection depends on the number of crossing and coincide edges, i.e. the chromosome which has a small number will be in the top. This method can be parameterized to give a desired emphasis to the best chromosomes. It uses elitist selection, i.e. the best chromosome is always chosen as such to the next generation.

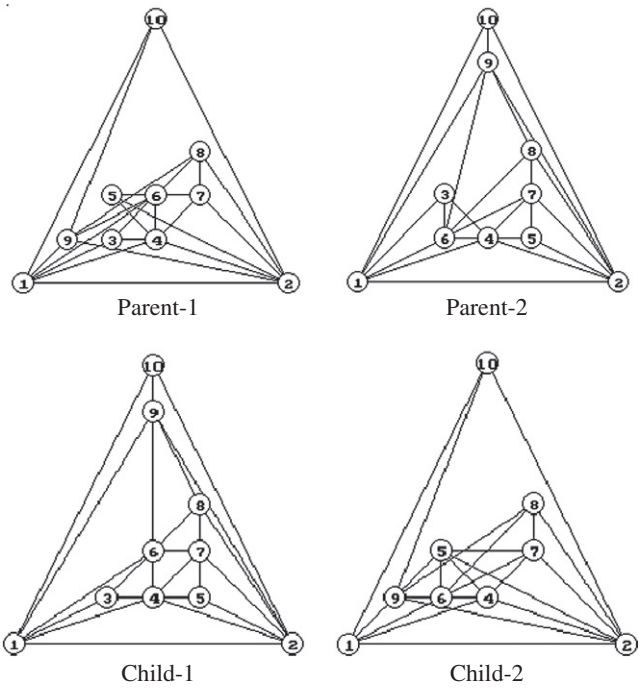


Figure 6 A sample of *Crossover*1 operation.

The esthetic criteria used are imported to genetic graph drawing algorithms in the form of the evaluation function (also called the fitness function). The algorithm tries to minimize the number of edge crossings, to distribute the nodes

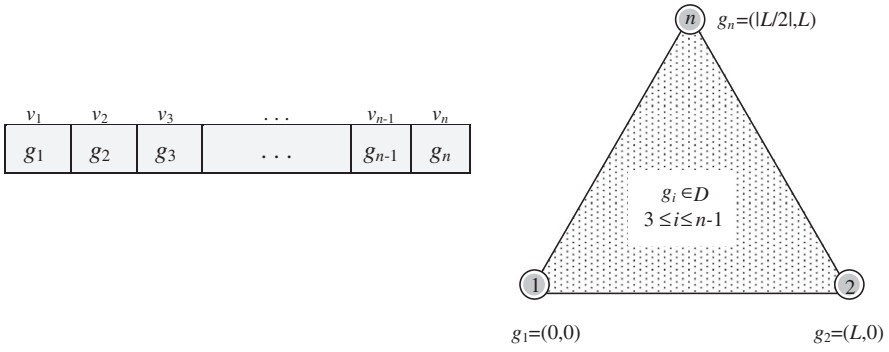


Figure 5 The representation of chromosome, and domain of the genes.

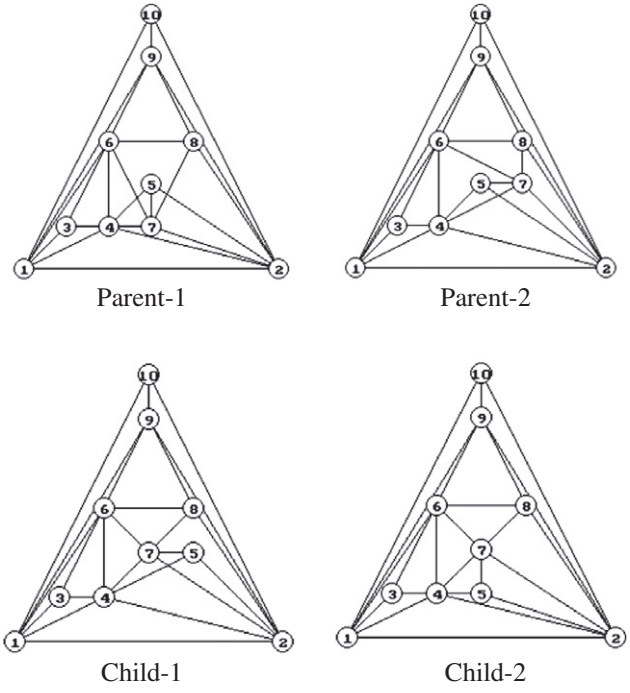


Figure 7 A sample of *Crossover*2 operation.

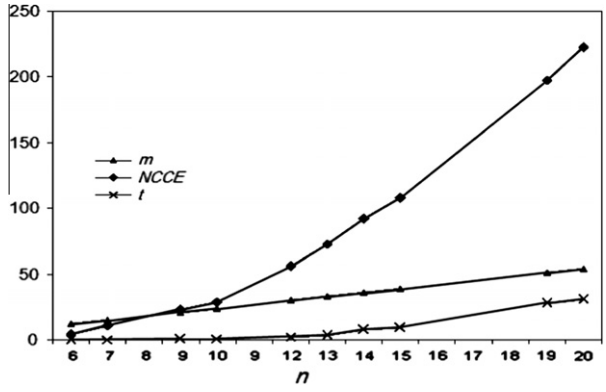


Figure 8 Chart of vertices n with m, $NCCE$ and average performance over time t.

Table 1 Test problems.

Graph No.	n	m	$NCCE$	GN	t	Area
1	6	12	4	1	0.04	3×3
2	7	15	11	9	0.064	4×4
3	9	21	23	78	0.357	5×5
4	10	24	29	301	0.987	6×6
5	12	30	56	547	2.402	7×7
6	13	33	69	620	3.381	8×8
7	14	36	92	1034	7.524	8×8
8	15	39	108	1133	9.142	9×9
9	19	51	198	1684	28.706	12×12
10	20	54	223	1890	32.231	12×12

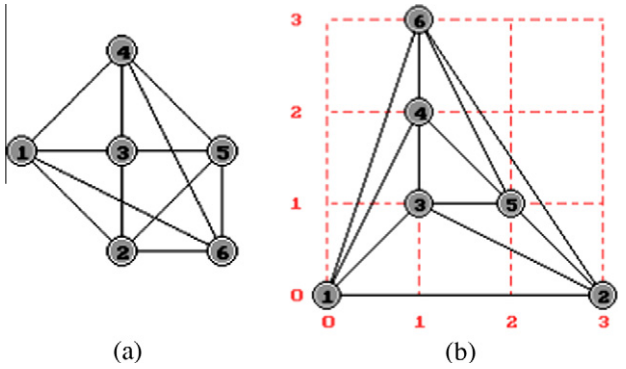


Figure 9 Graph No. 1, a random chromosome and a solution chromosome, respectively.

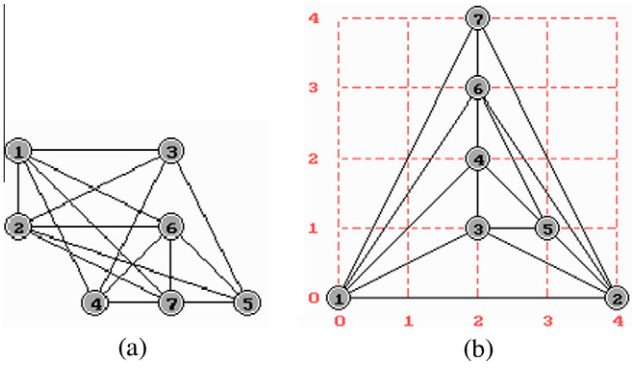


Figure 10 Graph No. 2, a random chromosome and a solution chromosome, respectively

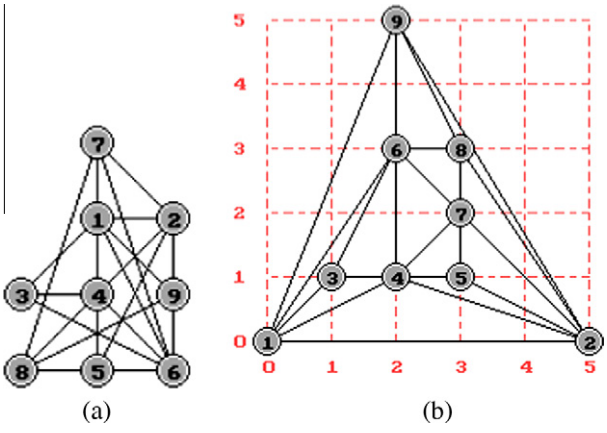
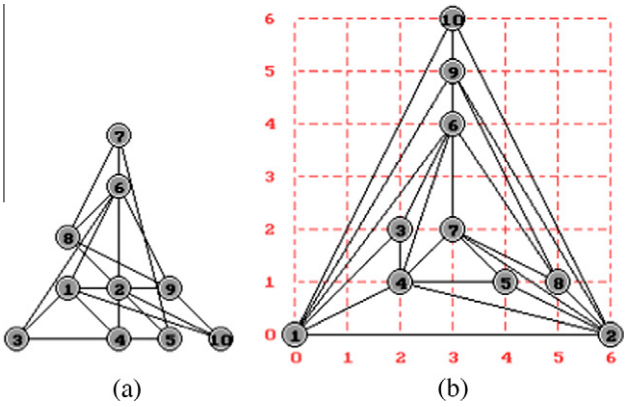


Figure 11 Graph No. 3, a random chromosome and a solution chromosome, respectively.

evenly over the drawing area. Fitness function based on two well-known measurable esthetic criteria for graphs:

1. Minimize graph area. This is a calculation of area of the bounding rectangle of the nodes in the graph on grid.



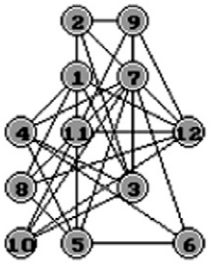
(a) (b)

Figure 12 Graph No. 4, a random chromosome and a solution chromosome, respectively.

2. Minimize edge crossings. The number of edge crossings is minimized to zero in the drawing grid.

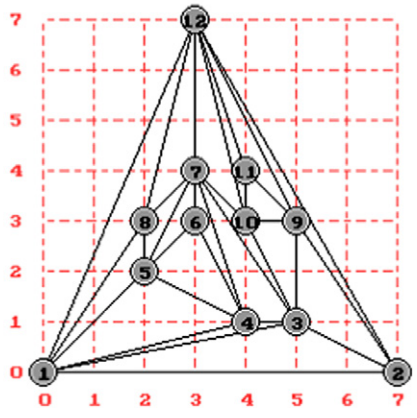
GAs are usually slow, especially because the fitness function evaluation takes a long time. The algorithm spends most of its computation time in evaluating the chromosomes. One of the problematic issues is the counting of the number of edge crossings. There is a well-known method based on cross productions to check whether two line segments intersect [[43], pp. 889–90]. More advanced methods are introduced by Bentley and Ottmann [44] and Chazelle and Edelsbrunner [45]. In order to reduce the run time of the execution our GA, we have used a method of our own for counting the number of edge crossing. We keep track of the movements of the nodes, and update the number of edge crossings only when a node is moved. This method outperforms the Bentley and Ottman's algorithm in the present situation.

The crossover operation transforms two chromosomes into two new chromosomes. The algorithm has two types of crossover operations. *Crossover*1 works as follows. First it randomly chooses an integer number or more, $3 \leqslant i \leqslant n-1$ and, say g_i is gene of the parent chromosome Parent-1, and a gene g_i^* of the parent chromosome Parent-2. The parent



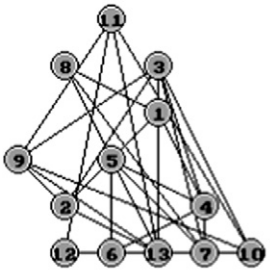
Input MPG with the edges:
E = { (1,2),(1,3),(1,4),(1,5),(1,8),(1,12),(2,3),(2,9), (2,12), (3,4),(3,7),(3,9),(3,10),(4,5),(4,6),(4,7),(5,6), (5,7), (5,8),(6,7),(7,8),(7,10),(7,12),(8,12),(9,10), (9,11), (9,12),(10,11), (10,12),(11,12) }

(a) a random chromosome.



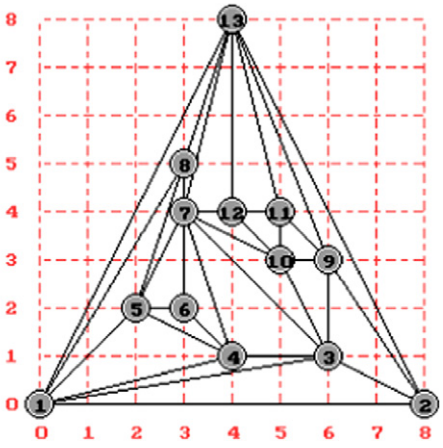
(b) a solution chromosome of G

Figure 13 Graph No. 5, a random chromosome with 12 vertices and the esthetically optimized result.



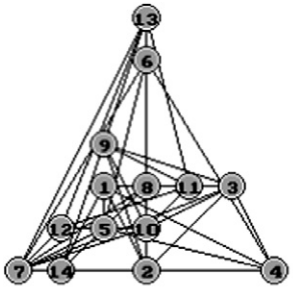
Input MPG with the edges:
E= { (1,2),(1,3),(1,4),(1,5),(1,8),(1,13),(2,3),(2,9), (2,13),(3,4),(3,7),(3,9),(3,10),(4,5),(4,6),(4,7),(5,6), (5,7),(5,8),(6,7),(7,8),(7,10),(7,12),(7,13),(8,13),(9,10), (9,11),(9,13),(10,11),(10,12),(11,12),(11,13),(12,13) }

(a) a random chromosome.



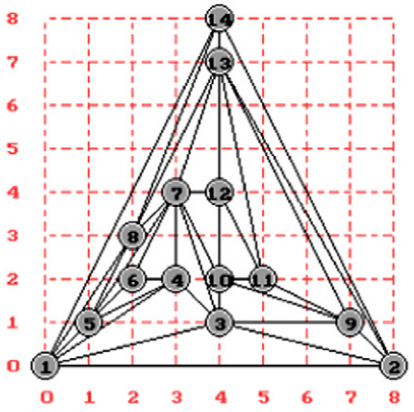
(b) a solution chromosome of G

Figure 14 Graph No. 6, a random chromosome with 13 vertices and the esthetically optimized result.



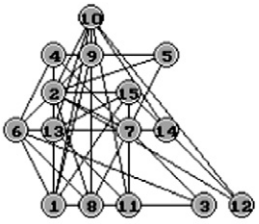
Input MPG with the edges:
E= { (1,2),(1,3), (1,4),(1,5),(1,8),(1,14), (2,3), (2,9),(2,13), (2,14),(3,4),(3,7),(3,9),(3,10),(4,5),(4,6),(4,7),(5,6),(5,7),(5,8), (6,7),(7,8),(7,10),(7,12),(7,13),(8,13),(8,14),(9,10),(9,11), (9,13),(10,11),(10,12),(11,12),(11,13),(12,13), (13,14) }

(a) a random chromosome.



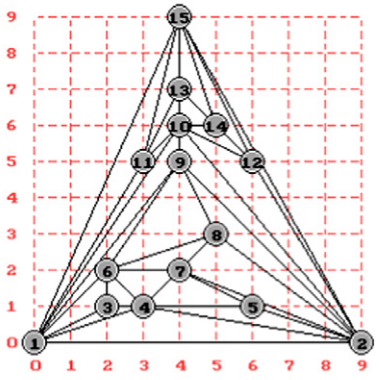
(b) a solution chromosome of G

Figure 15 An initial MPG of Graph No. 7, with 14 vertices and the esthetically optimized result.



Input MPG with the edges:
E= { (1,2),(1,3),(1,4),(1,6),(1,9),(1,10),(1,11),(1,15),(2,5),(2,4), (2,7),(2,8),(2,9),(2,10),(2,12),(2,15),(3,4),(3,6),(4,5),(4,6),(4,7), (5,7),(6,7),(6,8),(6,9),(7,8),(8,9),(9,10),(10,12),(10,14),(10,13), (10,11),(11,13),(11,15),(12,14),(12,15),(13,14),(13,15),(14,15) }

(a) a random chromosome.



(b) a solution chromosome of G

Figure 16 An initial MPG of Graph No. 8, with 15 vertices and the esthetically optimized result.

chromosomes exchange the genes g_i and g_i^* to produce new offspring, Child-1 and Child-2. The rest of the genes are kept unchanged, if possible. i.e. In Crossover1, an exchange happens in each gene position with a certain probability. For example if Parent-1: $g_1, g_2, \ldots, g_i, \ldots, g_j, \ldots, g_n$ and Parent-2: $g_i^*, g_2^*, \ldots, g_i^*, \ldots, g_j^*, \ldots, g_n^*$, then the new offspring after Crossover1 at positions i and j are Child-1: $g_1, g_2, \ldots, g_i^*, \ldots, g_j^*, \ldots, g_n$ and Child-2: $g_1^*, g_2^*, \ldots, g_i, \ldots, g_j, \ldots, g_n^*$.

The sample Crossover1 operation of Fig. 6 uses rectangles of size 6×6, with $n = 10$. Since genes $g_5 = (2,2)$, and $g_9 = (1,1)$ of Parent-1, $g_5 = (4,1)$, and $g_9 = (3,5)$ of Parent-2. The parents change the genes 5 and 9 (from Parent-1) and genes 5 and 9 (from Parent-2). then genes $g_5 = (2,2)$, and $g_9 = (1,1)$ of Child-2. Similarly, the genes $g_5 = (4,1)$, and $g_9 = (3,5)$, of Child-1. Clear that, the Child-1 is represent a solution of our case problem and good chromosome.

The other crossover operation in the algorithm is called *Crossover*2. It works as follows. First it randomly chooses a two integers $i, j \in \{3,4,\ldots,n-1\}$ and, say g_i and g_j are genes of the parent chromosome Parent-1. Similarly of two genes g_i^* and g_j^* of the parent chromosome Parent-2. Genes g_i and g_j in Parent-1 are exchange with g_j^* and g_i^* in Parent-2, respectively, to produce new offspring, Child-1: $g_1, g_2, \ldots, g_j^*, \ldots, g_i^*, \ldots, g_n$ and Child-2: $g_1^*, g_2^*, \ldots, g_j, \ldots, g_i, \ldots, g_n^*$. The rest of the genes are kept unchanged, if possible.

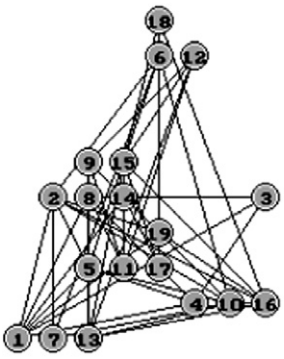
The sample Crossover2 operation of Fig. 7 uses rectangles of size 6×6, with $n = 10$. Since genes $g_5 = (3,2)$, and $g_7 = (3,1)$ of Parent-1, $g_5 = (3,2)$, and $g_7 = (4,2)$ of Parent-2. The parents change the genes 5 and 7 (from Parent-1) and genes 5 and 7 (from Parent-2). Then genes $g_5 = (4,2)$, and $g_7 = (3,2)$ of Child-1. Similarly, the genes $g_5 = (3,1)$, and $g_7 = (3,2)$, of Child-2. Here, the Child-2 is represent a solution of our case problem and good chromosome.

Groves et al. [42] introduced about a dozen different mutation operations. We have used two different mutations performed best in our tests. First, single mutation, choose a random node and move it to a random empty position. Second, inversion mutation, the order of the genes is inverted between two random nodes.

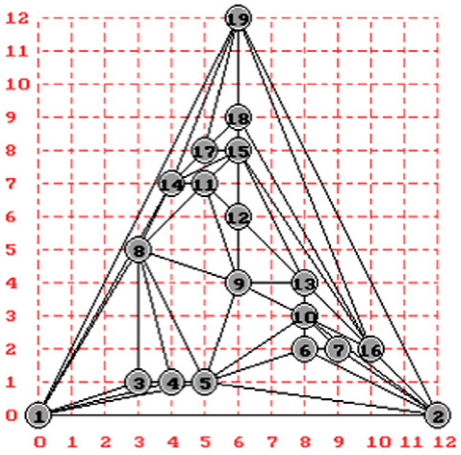
5. Parameters and results

The size of the grid: what is the optimal size of the drawing area for our test graphs? The optimum size was $\lfloor 2(n-1)/3 \rfloor \times \lfloor 2(n-1)/3 \rfloor$ of

Input MPG with the edges:
E={ (1,2),(1,3),(1,4),(1,5),(1,8),
(1,14),(1,19),(2,5),(2,6),(2,7),
(2,10),(2,16),(2,19),(3,8),(3,4),
(4,8),(4,5),(5,6),(5,8),(5,10),
(5,9),(6,10),(6,7),(7,10),(8,9),
(8,11),(8,14),(9,11),(9,10),
(9,13),(9,12),(10,13),(10,16),
(11,14),(11,15),(11,12),(12,13),
(12,15),(13,15),(13,16),(14,15),
(14,17),(14,19),(15,16),(15,17),
(15,18),(16,18),(16,19),(17,18),
(17,19),(18,19) }



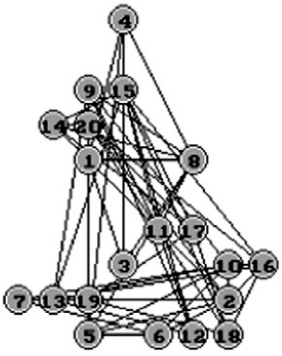
(a) a random chromosome.



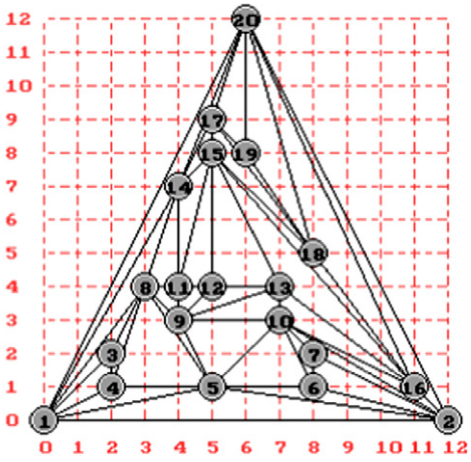
(b) a solution chromosome of *G*

Figure 17 An initial MPG of Graph No. 9, with 19 vertices and the esthetically optimized result.

Input MPG with the edges:
E={
(1,2),(1,3),(1,4),(1,5),(1,8),(1,14),
(1,20),(2,5),(2,6),(2,7),(2,10),
(2,16),(2,20),(3,8),(3,4),(4,8),
(4,5),(5,6),(5,8),(5,10),(5,9),
(6,10),(6,7),(7,10),(8,9),(8,11),
(8,14),(9,11),(9,10),(9,13),
(9,12),(10,13),(10,16),(11,14),
(11,15),(11,12),(12,13),(12,15),
(13,15),(13,16),(14,15),(14,17),
(14,20),(15,16),(15,17),(15,18),
(16,18),(16,20),(17,20),(17,18),
(17,19),(18,20),(18,19),(19,20) }



(a) a random chromosome.



(b) a solution chromosome of *G*

Figure 18 An initial MPG of Graph No. 10, with 20 vertices and the esthetically optimized result.

MPG where tested, grids smaller than this area were clearly inferior. Our algorithm was tested for different grids from $\lfloor 2(n-1)/3 \rfloor \times \lfloor 2(n-1)/3 \rfloor$ to $n-2) \times n-2)$. Selection: Our test advice to use large steps in the linear normalization. This means that the best chromosomes are strongly favored.

Crossover and mutation rates: Increasing the mutation rate makes the search more efficient all the way to the level 20–50%. Still increasing the mutation rate over 50% again makes the results worse. The crossover rate 20% and mutation rate 20% are default values. The values of the parameters of the algorithm are as follows: Maximum Generations (*MGs*): 2000, Population Size (*PS*): 10, Crossover Rate (*CR*): 0.3 and Mutation Rate (*MR*): 0.2.

Our genetic algorithm was able to find layouts with no edge crossings in all tested MPGs. We use the following test problems which a randomly generated with the number of crossing and coincide edges (*NCCEs*) shown in Table 1. We use the following some test problems which a randomly positions generated with *NCCE*, average Generation Number (*GN*) and average run time (*t*) shown in Table 1. Note that, the computation times given are averaged over several runs (15 times of each test problem). This means that randomly selected initial populations may distort the results. In order to the readers can be reproduce the experiments, the graphs structure is presenting in Figs. 9–18 of each test problem.

Fig. 8 shows the average crossing number *NCCE* evolution for the whole set of graphs. Increasing the number of nodes *n* makes increasing the average of *NCCE* in random chromosomes. For example, Fig. 18 shows a case of test problem number 10, which a randomly generated graph with the number of crossing = 302 and coincide edges = 7, i.e. $NCCE = 309$. Running the GA with it's parameters, $CR = 0.3$ $CM = 0.2$, produces the final population which has no edge crossings or coincide edges at $GN = 1788$ and $t = 30.9$ on 12×12 grid. The computation time was about half minute.

6. Conclusion

In this study, an attempt is made to develop a new GA for straight-line grid drawings of MPG in $\lfloor 2(n-1)/3 \rfloor \times \lfloor 2(n-1)/3 \rfloor$ at least, and that this is minimum area. Our GA nicely draws most MPG of moderate size. Also, the novel

issue in the proposed method is the fitness evaluation method, which is less costly than a standard fitness evaluation procedure. We make experiments with simple MPG and results allow us to think that GA technology is a strong candidate to solve this kind of problems.

References

[1] Eades Peter, Cohen Robert F, Huang Mao Lin. Online animated graph drawing for web navigation. In: Di Battista G, editor. Graph drawing '97. Lecture notes in computer science, vol. 1653. Rome, Italy: Springer-Verlag; 1998.

[2] Farrugiaa M, Quigley A. Effective temporal graph layout: a comparative study of animation versus static display methods. Inform Visual 2011;10(1):47–64.

[3] Gansner E, Hu Y, Kobourov SG. Visualizing graphs and clusters as maps. IEEE Comput Graphics Appl 2010;30(6):44–66.

[4] Six Janet, Tollis Ioannis. A framework for circular drawings of networks. In: Proc symp graph drawing GD'99. Lecture notes in computer science, 1731. Springer-Verlag; 2000. p. 107–16.

[5] Tubaishat M, Madria S. Sensor networks: an overview. IEEE Potentials 2003;22:20–3.

[6] Didimo W, Liotta G, Romeo Salvatore A. A graph drawing application to web site traffic analysis. J Graph Algorithms Appl 2011;15(2):229–51.

[7] Williams C, Rasure J, Hansen C. The state of the art of visual languages for visualization. In: Visualization '92, proc., IEEE computer society press; 1992. p. 202–9.

[8] Radwan Ahmed A, El-Sayed Mohamed A, Omran Nahla F. Hybrid GA for straight-line drawings of level clustered planar graphs. Int J Comput Sci Issues (IJCSI) 2011;8(4):229–35, No. 2.

[9] Dornheim Christoph. Planar graphs with topological constraints. J Graph Algorithms Appl (JGAA) 2002;6(1):27–66.

[10] Fowler J, Juenger M, Kobourov SG, Schulz M. Characterizations of restricted pairs of planar graphs allowing simultaneous embedding with fixed edges. Comput Geom: Theory Appl 2011;44(8):385–98.

[11] Harel D. On visual formalisms. Commun ACM 1988;31(5):514–30.

[12] Kaufmann Michael, Wagner Dorothea. Drawing graphs: methods and models. Lecture notes in computer science, 2025. Springer-Verlag; 2001.

[13] Nishizeki Takao, Rahman Md Saidur. Planar graph drawing, world scientific books. Lect Notes Ser Comput 2004;12.

[14] Bridgeman S, Tamassia R. A user study in similarity measures for graph drawing. J Graph Algorithms Appl 2002;6(3):225–54.

[15] Purchase H. Effective information visualization: a study of graph drawing aesthetics and algorithms. Interact Comput 2000;13(2).

[16] Di Battista G, Eades P, Tamassia R, Tollis IG. Annotated bibliography on graph drawing algorithms. Comput Geom Theory Appl 1994;4:235–82.

[17] Eades P. A heuristic for graph drawing. Congressus Numerantium 1984;42:149.

[18] Vrajitoru D, DeBoni J. Hybrid real-coded mutation for genetic algorithms applied to graph layouts. In: Proceeding of the genetic and evolutionary computation conference (GECCO'05 and SIGEVO 1), Washington, DC; June 25–29 2005. p. 1563–4.

[19] Vrajitoru D. Hybrid multi-objective optimization genetic algorithms for graph drawing. In: Proc genetic and evolutionary computation conference (GECCO'07); 2007.

[20] Bui TN, Moon BR. Genetic algorithm and graph partitioning. IEEE Trans Comput 1996;45(7):841–55.

[21] Kapoor N, Russell M, Stojmenovic I, Zomaya AY. A genetic algorithm for finding the page number of interconnection networks. J Parallel Distrib Comput 2002;62:267–83.

[22] Pinaud B, Kuntz P, Lehn R. Dynamic graph drawing with a hybridized genetic algorithm. Automat Comput Des Manuf 2004;VI:365–75.

[23] Chen WBL, Zhu W, Xiang X. Multiple sequence alignment algorithm based on a dispersion graph and ant colony algorithm. J Comput Chem 2009;30:2031–8.

[24] Lee ZJ, Su SF, Chuang CC, Liu KH. Genetic algorithm with ant colony optimization (GA–ACO) for multiple sequence alignment. Appl Soft Comput J 2008;8:55–78.

[25] Wallace PM, Blackshields G, Higgins DG. Multiple sequence alignments. Curr Opin Struct Biol 2005;15:261–6.

[26] Xiang X, Zhang D, Qia J, Fu y. Ant colony with genetic algorithm based on planar graph for multiple sequence alignment. Inform Technol J 2010;9(2):274–81.

[27] de Fraysseix H, Pach J, Pollack R. Small sets supporting straight-line embeddings of planar graphs. In: Twentieth annual ACM symposium on theory of computing; 1988. p. 426–33.

[28] Chrobak M, Payne T. A liner-time algorithm for drawing a planar graphs on a grid. Inform Process Lett 1995;54:241–6.

[29] Schynder W. Planar graphs and poset dimension. Order 1989;5:323–43.

[30] Schynder W. Embedding planar graphs on the grid. In: Proceedings of the first annual ACM-SIAM symposium on discrete algorithms, San Franciso; 1990. p. 138–47.

[31] Chiba N, Yamanouchi T, Nichireki T. Linear algorithms for convex drawings of planar graphs. In: Bondy JA, R Murty US, editors. Progress in graph theory. New York: Academic Press; 1984. p. 153–73.

[32] Davis L. A genetic algorithms tutorial. In: Davis L, editor. Handbook of genetic algorithms. Van Nostrand Reinhold; 1991. p. 1–101.

[33] He X. A simple linear time algorithm for proper box rectangular drawings of plane graphs. J Algorithms 2001;40(1):82–101.

[34] Kant G. Drawing planar graphs using the canonical ordering. Algorithmica 1996;16(1):4–32.

[35] Kant G, He X. Regular edge labeling od 4-connected plane graphs and its applications in graph drawing problems. Theor Comput Sci 1997;172(1–2):175–93.

[36] Radwan AAA, El-Sayed MA. New algorithm for drawing planar straight-line graphs. Int J Appl Math 2003;11(3):265–82.

[37] Nakano S. Planer drawings of plane graphs. IEICE Trans Fundam 2000;E00-A(3).

[38] Eloranta T, Makinen E. TimGA – a genetic algorithm for drawing undirected graphs. Report series A-1996-10. University of Tampere, Department of Computer Science; 1996. <citeseer.nj.nec.com/eloranta96timga.html>.

[39] Radwan AAA, El-Sayed MA. Using Genetic Algorithm for Drawing Triangulated Planar Graphs. J Inst Math Comput Sci (Comp Sci Ser) 2004;15(1):137–47.

[40] Rosete A, Ochoa A. Genetic graph drawing. In: Proceeding of the 13th international conference of applications of artificial intelligence in engineering, Galway; 1998. p. 37–41.

[41] Chrobak M, Nakano S. Minimum-width grid drawings of plane graphs. Comput Geometry: Theory Appl 1998;10:29–54.

[42] Groves L, Michalewicz Z, Elia P, Janikow C. Genetic algorithms for drawing directed graphs. In: Proceedings of the fifth international symposium on methodologies for intelligent systems. Elsevier North-Holland; 1990. p. 268–76.

[43] Cormen TH, Leiserson CE, Rivest RL. Introduction to algorithms. The MIT Press; 1990.

[44] Bentley JL, Ottmann TA. Algorithms for reporting and counting geometric intersections. IEEE Trans Comput 1979;C-28:643–7.

[45] Chazelle B, Edelsbrunner H. An optimal algorithm for intersecting line segments in the plane. J ACM 1992;39(1):1–54.