Hybrid GA for Straight-Line Drawings of Level Clustered Planar Graphs

Ahmed A. A. Radwan [1], Mohamed A. El-Sayed [2*] and Nahla F. Omran [3]

[1] Department of Computer Science, Faculty of Science, Minia University, Minia, Egypt

[2] Department of Mathematics, Faculty of Science, Fayoum University, Fayoum, Egypt

[3] Department of Mathematics, Faculty of Science, South Valley University, Qena, Egypt

Abstract
In this paper we introduce an application of genetic algorithms (GAs) with the problem of drawing of level planar graph or hierarchical planar graph, and explore the potential use of GAs to solve this particular problem. Given a l evel planar graph, we want to find a geometric position of every vertex (layout) in a straight-line grid drawing without any edge-intersection. Here we introduce a simple hybrid GA, which nicely draws level planar graph of moderate size. The paper shows that the GAs can help find an layout of levels and hierarchical planar graphs without any crossing edges.

Keywords: *Genetic algorithms, graph drawing, hierarchical graphs, level graph, clustered graph.*

1. Introduction

Hierarchical planar graph embedding (sometimes called level planar graphs) is widely recognized as a v ery important task in diverse fields of research and development. Examples include VLSI Design and plant layout [9], graphical user interfaces for visualization for information [2], software and information engineering, project management, visual languages [5], subroutine-call graphs, Interpretative Structural Modeling [13], organization charts, hierarchical relationships, system theory and other research fields. Designing web sites, and visualizing the content of the World Wide Web [4]. See An example of a hierarchical graph of UML level- cluster diagram in Figure 1.

Many applications imply a partition of the vertices into levels that have to be visualized by placing the vertices that belonging to the same level on a horizontal line. The corresponding graphs are called level graphs, and the drawing of the networks that correspond to this category of graphs means the drawing of level graphs [17]. In fact, early articles in this area state that "the most crucial problem as far as readability of a graph is that of edge crossing" [7].

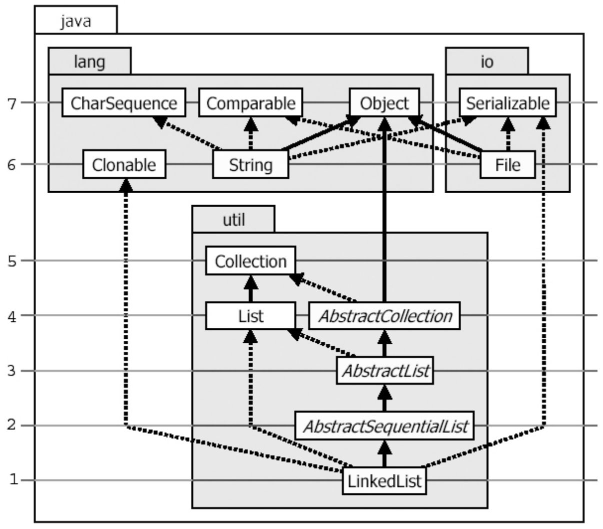


Fig. 1 An example of a hierarchical graph, UML level-cluster diagram

Usually, we use one of some aesthetic criteria (such as drawing area minimization, minimizing the number of edge crossings, symmetry, bends minimization or distributing the vertices uniformly) in order to make the layout of a

[*] Corresponding author.

graph readable and understandable [20, 26]. Reducing the number of edge crossings or distributing the vertices uniformly have been proposed, and evaluating goodness of drawing based on these criteria has been reported [3, 19].

Numerous deterministic heuristics for the static hierarchical layout problem follow the layer-by-layer sweep scheme: the vertices of each layer are reordered to reduce crossings while holding the vertex orderings on the other layers. Various strategies have been proposed for reordering [18]. The most commonly used are the sorting methods and the averaging heuristics which include the popular barycenter heuristics from Sugiyama [25], the median heuristics [14], and their variants. Sorting methods exchange vertices using crossing numbers in a way similar to classical sorts. Averaging heuristics are based on the idea that edge crossings tend to be minimized when connected vertices are placed facing each other. Consequently, vertices are arranged according to their neighbour average positions e.g. the arithmetic mean or the median. Abello [1] highlighted the main tasks behind the computation of hierarchical graph maps and provided several examples.

The use of integer coordinates in embedding a graph on the grid has many advantages such as speed, accuracy, and it guarantees automatically that the resultant picture has fairly good properties. A straight line drawing is a grid drawing if each vertex is at a grid point, and the edges are represented as straight-line segments between their endpoints without any edge-intersection. See example of straight-line drawings by Eades, Feng, and Lin [12] in Figure 2.

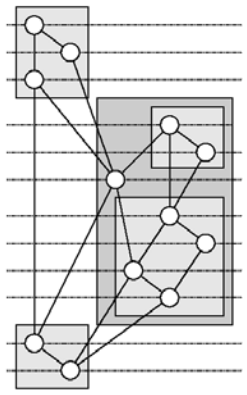


Fig. 2 Straight-line drawings of hierarchical clustered graph [12].

Previously, there have been two main concepts for combining leveling and clustering. Sugiyama and Misue [24] present an algorithm for drawing *compound graphs* on horizontal levels. Compound graphs are a generalization of clustered graphs that also allow edges between two clusters or between a cluster and a vertex. A similar algorithm is proposed by Sander [23]. Both algorithms extend the classical level drawing algorithm of Sugiyama, Tagawa, and Toda [25]. The vertices are drawn on horizontal levels, and the clusters are drawn as nested rectangles. After partitioning the vertices into levels, permutations of each level are computed for minimizing edge crossings. Finally, horizontal coordinates are assigned to the vertices and clusters.

This paper introduces application of genetic algorithms (GAs) with the problem of planar hierarchical graph drawing in a rectangular grid and explore the potential use of GAs to solve this particular problem. Our algorithm has been inspired by the ideas from Refs.[14, 21, 22].

The remainder of the paper is organized as follows. In section 2, displays the problem with some definitions and preliminaries. In section 3, introduces proposed hybrid GA. The representations of chromosomes, the selection, and the evaluation function are investigated in section 4. In section 5, introduces the genetic operations. Section 6 is abstract the parameters and results. Finally, conclusions in section 7.

2. Preliminaries

If the given graph $G = (V, E)$ is a directed acyclic graph (DAG), or after all cycles have been removed, the vertex set V is partitioned into levels $V_1, V_2, \ldots, V_k$. All vertices of the same level V_i are later drawn on the horizontal line $l_i = \{ (x, i) \mid x \in \mathbb{N}\}$. The level number is identical to the y-coordinate of the vertex. The result of the level assignment step is a level graph: A k-level graph $G = (V, E, \varphi)$ is a graph (V, E) with a leveling $\varphi: V \rightarrow \{1, \ldots, k\}$ that partitions the vertex set into k disjoint levels $V_1, V_2, \ldots, V_k$, $V_i = \varphi^{-1}(i)$, such that each edge $(u,v) \in E$ has a positive span $\varphi(v) - \varphi(u) > 0$, i. e., all edges point downwards. Edges are called proper if their span is 1 and long span edges otherwise. G is proper if all its edges are proper. Figure 4 is proper level graphs of G in Figure 3, where dummy vertices are drawn black.

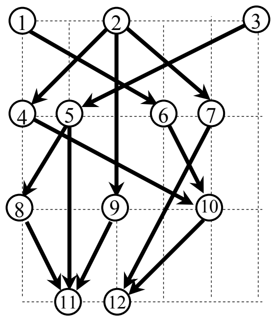


Fig. 3 Example of level graphs.

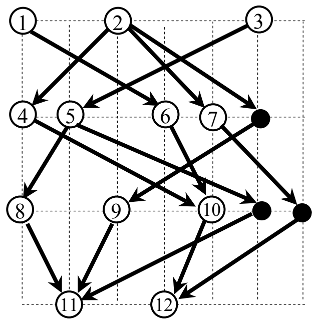


Fig. 4 Proper level graphs.

Hierarchical acyclic digraph $H=(V,E,\lambda,k)$, consists of a directed graph $G=(V,E)$ with a vertex set V and an edge set E and $L = \{ L_1, L_2, \ldots, L_k\}$ be a set of k layers and a given partition $V_1, V_2, \ldots, V_k$ of V on L in classes with respectively $n_1, n_2, \ldots, n_k$ vertices. A positive number k, and, for each vertex u, an integer $\lambda(u)\in\{1,2,\ldots,k\}$, with the property that if $(u,v)\in E$, then $\lambda(v)>\lambda(u)$. For $1\le i\le k$ the set $\{u:\lambda(u)=i\}$ is the i^{th} *layer* of G and denoted by L_i .

The *span* of an edge (u,v) is $\lambda(v)-\lambda(u)$. An edge of span greater than one is called *long edge*; and a hierarchical graph with no long edges is called *proper hierarchical graph*. The layers L_1 and L_k are called *boundary layers* of *H*. If all source vertices are in layer L_1 and all sink vertices are in layer L_k, in this case *H* is called a *boundary s-t hierarchical graph*.

A hierarchical graph is conventionally drawn with layer L_i on the horizontal line *y*=*i*. If t he graph is proper, then edges are drawn as closed line segments. Thus a p roper hierarchical graph is assigns a point $P(u)=(p_x(u), p_y(u))$ to each vertex $u\in V$. The drawing convention implies that $p_y(u)=\lambda(u)$ and so effectively the only role of a d rawing algorithm is to choose $p_x(u)$.

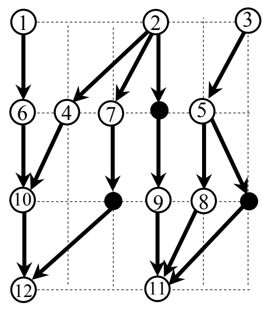


Fig. 5 Proper level planar graphs embedding.

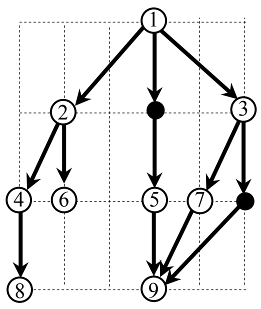


Fig. 6 Proper hierarchical planar graphs embedding.

In a level graph, we define the *longest level* that is the level in which the number of vertices is greater than or equal the number of vertices belong to any other level. Note that the longest level is not necessarily unique. For example, Figure 3, shows a level graph with four levels, and second level is the longest levels, it h as four vertices. but in Figure 4, shows a l evel graph with two levels are the longest levels, each one has five vertices.

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 favour 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 mutations. 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 aesthetic 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.

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. Proposed hybrid genetic algorithm

Let $G=(V^1,V^2,...,V^k;E)$ be a given proper level graph with *n* vertices. An embedding of a level graph can be aesthetic if we draw the graph by making the following two main steps:

First step: we are divide the graph into set of paths $\{W_1, W_2, \dots, W_{np}\}$, where *np* is the number of the individual paths of *G*, each path is contains at most one node in level V^i, $1\le i\le k$. If found more than one node in the next (previous) level are neighbored of the current node in the current level, we select one from them such that it was not selected as before. If there are dummy vertices among these nodes, one of them is favored to select. Remove this path from the graph and repeat this process until remove all nodes. The processes of dividing and clustering are begin according to longest level which contain largest number of dummy vertices. The number *np* of clusters is greater than or equal to the number of nodes in longest level. Figure 7, shows a six paths, *np*=5 of proper level graph in Figure 4 .

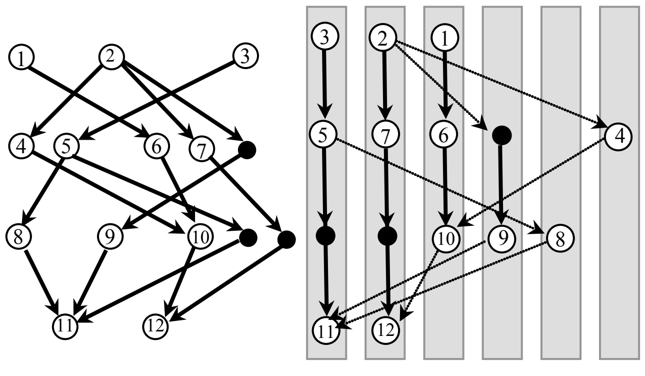


Fig. 7 Proper level graph and it's a partial paths.

Second step: We are rearrangement the paths using GA, where no pair of edges intersect. therefore the out put is proper level planar graph on grid with size *k*-1 × *np*-1. Figure 8, shows a solution of proper level graph in Figure 7, and Figure 9 is show the output level planar graphs embedding on grid.

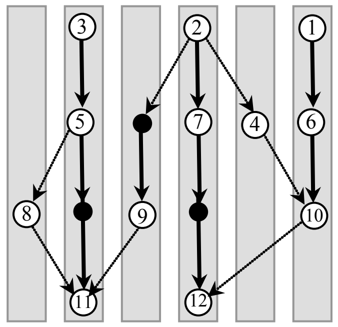


Fig. 8. a GA solution of *G* in Figure 7.

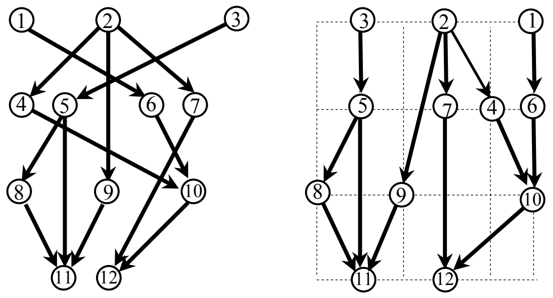


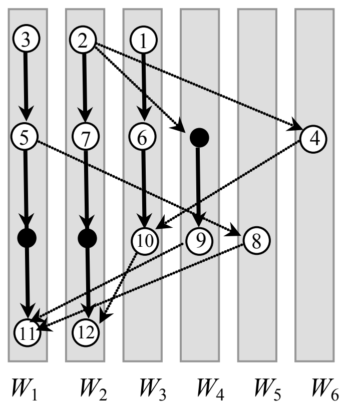
Fig. 9 Input level planar graphs and its output embedding on grid.

4. Representations of chromosomes and fitness function

Our algorithm draws hierarchical planar graphs in a rectangular grid with area *k*-1 × *np*-1. Each vertex is located in a Cartesian point of the grid and all edges are drawn as straight lines.

To represent a graph with *n* vertices and *m* edges, we use a *k*-1 × *np*-1 matrix to indicate the positions of the vertices and a 2 × *m* matrix to indicate the edges by storing pairs of vertices. The corresponding end points are then found from the vertices matrix. Figure 10, shows the representation used of a sample example, Figure 7, *P*(3)=(0, 0), *P*(4)=(5, 1) and *P*(10)=(2, 2) .

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 *np*, a gene, g_j is represent the position $0 \le x \le np\text{-}1$ of path W_j in grid, $1 \le j \le np$, and the genes have different values in the chromosome. For example ,in Figure 10, the set of nodes { v_3, v_5, v_{14}, v_{11}} belong to Path W_1 , *x*-coordinate values are equal to 0, but *y*-coordinate value is fixed of each vertex in *G* through our algorithm according to its level. Hence , the chromosome depend on *x*-value which changed between the paths, where each Path W_j take certain value for its nodes. Figure 11 is show the general form of the chromosome.



Vertices matrix, n=15

	0	1	2	3	4	5
0	3	2	1			
1	5	7	6	13		4
2	14	15	10	9	8	
3	11	12				

Edges matrix, m=16 (directed from-to)

e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13	e14	e15	e16
3	2	2	2	1	5	5	7	6	13	4	14	15	10	9	8
5	7	13	4	6	14	8	15	10	9	10	11	12	12	11	11

Fig. 10 The representation of *G* in Figure 7.

Here We describe the following operations: selection process, and fitness function, 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 [11] together with *elitism*. 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 less 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.

0	1	2	. . .	*np*-2	*np*-1
W_1	W_2	W_3	. . .	$W_{np\text{-}1}$	W_{np}
g_1	g_2	g_3		$g_{np\text{-}1}$	g_{np}

(a) general form of the chromosome

0	1	2	3	4	5
W_1	W_2	W_3	W_4	W_5	W_6
g_1	g_2	g_3	g_4	g_5	g_6

(b) initial chromosome of example in figure 7

Fig. 11 The representation of chromosome

Evaluation function f (also called the fitness function) based on well-known measurable aesthetic criteria for graphs: Minimize edge crossings. The number of edge crossings is minimized to zero in the drawing grid, α=0.

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 [10, pp. 889-890]. More advanced methods are introduced by Bentley and Ottmann [6] and Chazelle and Edelsbrunner [8]. 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 paths, and update the number of edge crossings only when a path is moved. This method outperforms the Bentley and Ottman's algorithm in the present situation.

5. Crossover and mutation operations

The crossover operation transforms two chromosomes into two new chromosomes. The algorithm has two types of crossover operations. *Crossover*1 works as follows. Let the parent chromosome P_1: g_1, g_2,, …, g_{np} , which is stores the paths numbers. First it randomly chooses an integer number or more of path W_i , $1 \le i \le np$, of the parent chromosome P_1. Similarly, the genes in P_2 , and a path W^*_j of the parent chromosome P_2, $i \ne j$. Exchange paths numbers W_i with W^*_j and W_j with W^*_i, and the rest of the paths are kept unchanged of its orders in genes of the parent chromosomes P_1 and P_2 to obtain children: Ch_1 and Ch_2. The sample Crossover1 operation of Figure 12, uses 4 levels, with six paths. Consider the permutations :

$P_1 = (W_1, W_3, W_2, W_6, W_5, W_4)$ and $P_2 = (W^*_3, W^*_6, W^*_5, W^*_4, W^*_2, W^*_1)$. If it randomly chooses i=2 , j= 5. Thus, Ch_1=(W_1, W_3, W^*_5, W_6, W^*_2, W_4). Similarly, Ch_2=(W^*_3, W^*_6, W_2, W^*_4, W_5, W^*_1). Clear that, the Child-2 is represent a solution of our case problem and good chromosome.

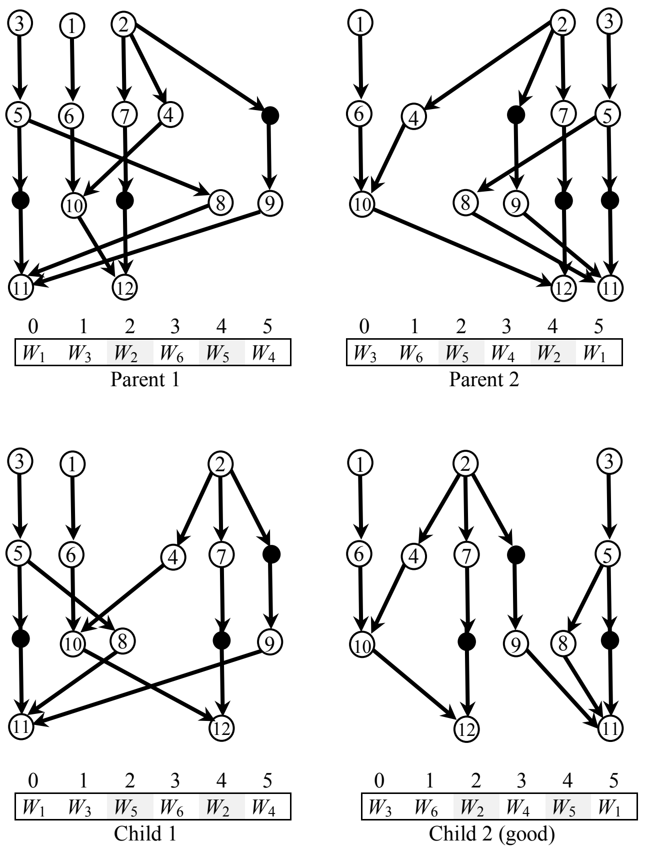

0 1 2 3 4 5
W_1 W_3 W_2 W_6 W_5 W_4
Parent 1
0 1 2 3 4 5
W_3 W_6 W_5 W_4 W_2 W_1
Parent 2
0 1 2 3 4 5
W_1 W_3 W_5 W_6 W_2 W_4
Child 1
0 1 2 3 4 5
W_3 W_6 W_2 W_4 W_5 W_1
Child 2 (good)

Fig. 12 A Sample of *Crossover*1 operation.

The other crossover operation in the algorithm is called *Crossover2* or *cycle crossover*: Given parent permutations P_1 and P_2 , two child permutations are created by forming a cycle between P_1 and P_2. Consider the permutations : P_1 = (*jhdefigcba*) and P_2 = (*hgebcjiadf*). Starting from the first element in P_1 , we see that *j* in P_1 maps to *h* in P_2 , *h* in P_1 maps to *g* in P_2 , *g* in P_1 maps to *i* in P_2 , and *i* in P_1 maps to *j* in P_2 , completing the cycle. The elements in the cycle from P_1 are placed in the child, producing Ch_1=(*jh***ig****). The empty slots (*) are filled in by the elements of P_2 at the corresponding positions. Thus, Ch_1=(*jhebcigadf*) Similarly, Ch_2=(*hgdefjicba*). See Figure 13.

Groves et al. [16] introduced about a d ozen different mutation operations. We have used t wo different mutations performed best in our tests. First, *mutation*1, choose a two genes randomly and exchange their position. Second, *mutation*2, the order of the genes is inverted between two random vertices.

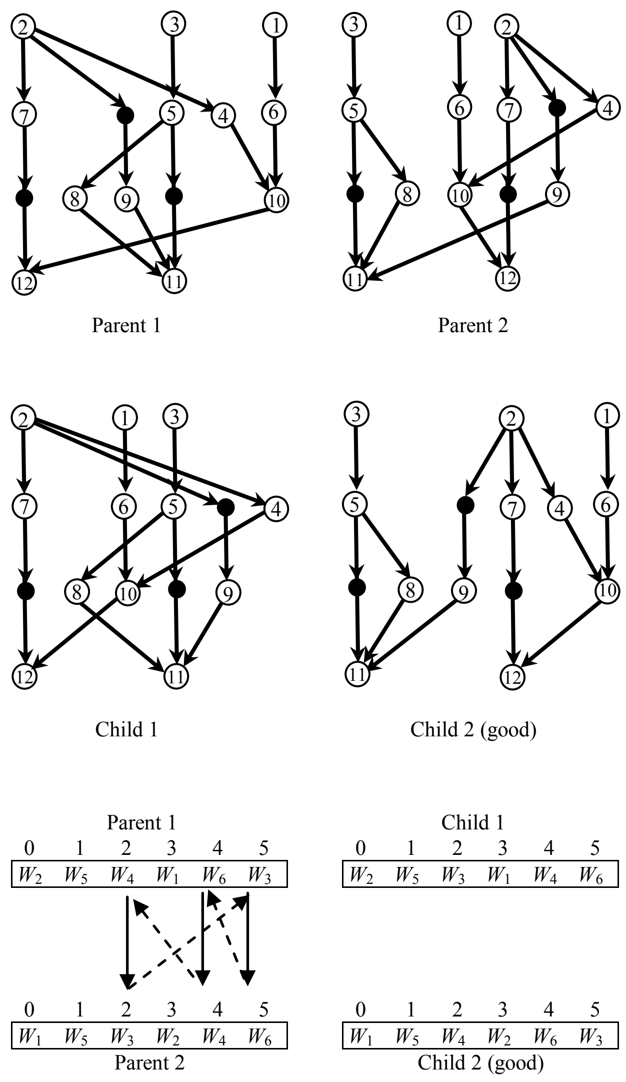

Parent 1
Parent 2
Child 1
Child 2 (good)
Parent 1
0 1 2 3 4 5
W_2 W_5 W_4 W_1 W_6 W_3
0 1 2 3 4 5
W_1 W_5 W_3 W_2 W_4 W_6
Parent 2
Child 1
0 1 2 3 4 5
W_2 W_5 W_3 W_1 W_4 W_6
0 1 2 3 4 5
W_1 W_5 W_4 W_2 W_6 W_3
Child 2 (good)

Fig. 13 A Sample of *Crossover2* operation.

6. Parameters and results

Selection: Our test advice to use large steps in the linear normalization. This means that the best chromosomes are strongly favored. This selection is decrease computation time to compute the fitness function with paths rather than the vertices in graph.

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(*MG*): 1000, Population Size (*PS*): 10, Crossover Rate (*CR*): 0.2 and Mutation Rate (*MR*): 0.2 . Our genetic algorithm was able to find layouts with no edge crossings in all the test level planar graphs.

7. Conclusion

In this study, an attempt is made to develop a new hybrid GA for straight-line grid drawings of level clustered planar graph drawing without any edge-intersection and explore the potential use of GAs to solve this particular problem. It is nicely draws level planar graph of moderate size. The operations of crossover and mutation are described, and tested on several level planar graphs.

References

[1] J. Abello, “Hierarchical graph maps”, Computers & Graphics, Volume 28, Issue 3, June 2004, pp. 345-359.
[2] G. Di Battista, P. Eades, R. Tamassia and I. Tollis, “Algorithms for Drawing Graphs: Annotated Bibliography”, Computational Geometry, Theory Applications, 4, 1994, pp. 235-282.
[3] C. Batini, L. Furlani and E. Nardelli, “What is a Good Diagram? A Pragmatic Approach”, Proceeding Of the 4th International Conference on Entity-Relationship Approach, 1985 , pp. 312-319.
[4] U. Brandes, V. Kääb, A. Läh, and D. Wagner, Dynamic WWW structures in 3d. Journal of Graph Algorithms and Applications 4(3), 2000, pp. 183–191.
[5] S. Bhatt and F. Leighton, “A Framework for Solving VLSI Graph Layout Problems”, Journal of Computer and System Systems Sciences 28, 1984, pp. 300-343.
[6] J. L. Bentley and T. A. Ottmann, “Algorithms for reporting and counting geometric intersections”. IEEE Trans. Comput. C-28,1979, pp. 643-647.
[7] M-J. Carpano, Automatic display of hierarchized graphs for computer-aided decision analysis. IEEE Transactions on Systems, Man and Cybernetics. SMC-10, 11. 1980, pp. 705-715.
[8] B. Chazelle and H. Edelsbrunner, “An optimal algorithm for intersecting line segments in the plane”. J. ACM 39, 1 , 1992, pp. 1-54.
[9] F. Chung, F. Leighton and A. Rosenberg, “Embedding Graphs in Books: A Layout Problem with Applications to VLSI Design”, SIAM Journal Discrete Mathematics, 8, 1987, pp. 33-58.
[10] T. H. Cormen, C. E. Leiserson and R. L. Rivest,” Introduction to Algorithms”. The MIT Press, 1990.
[11] L. Davis, “A genetic algorithms tutorial”. Handbook of Genetic Algorithms, L. Davis (ed.), Van Nostrand Reinhold,1991, pp. 1-101.
[12] P. Eades, Q.-W. Feng, and X. Lin. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. S. North, editor. Proceedings Graph Drawing, GD 1996, volume 1190 of Lecture Notes in Computer Science. Springer, 1997, pp. 113–128.
[13] P. Eades and X. Lin, “How to Draw a Directed Graph”, Proceeding IEEE on Visual Languages, (VL ’89) , 1989, pp. 13-17.
[14] T. Eloranta and E. Makinen. “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
[15] P. Eades and N. Wormald. Edge crossings in drawings of bipartite graphs. Algorithmica, 11, 1994, pp. 379-403.
[16] L. Groves, Z. Michalewicz, P. Elia and C. Janikow, Genetic algorithms for drawing directed graphs. Proceedings of the Fifth International Symposium on Methodologies for Intelligent Systems, Elsevier North-Holland,1990, pp. 268-276.
[17] M. Jünger and S. Liepert, "Level Planar Embedding in Linear Time", Technical Report 99-374, Institut für Informatik, Unversität zu Köln, 1999.
[18] M. Laguna, R. Marti, and V. Valls. Arc crossing minimization in hierarchical digraphs with tabu search. Computers and Operation Research, 24(12), 1997, pp.1175-1186.
[19] H. Purchase, R. Cohen and M. James, “Validating Graph Drawing Aesthetics”, Proceeding of Symposium on Graph Drawing, GD ’95 (Lecture Notes in Computer Sciences, 1027), Springer, 1996, pp. 435-446.
[20] A. A. A. Radwan, “A new Algorithm for Drawing Level Graphs on a Grid with Minimum Width”, International Journal of Applied Mathematics, vol. 12 No. 4, 2004, pp. 367-385.
[21] A. A. A. Radwan, M. A. El-Sayed, “Using Genetic Algorithm for Drawing Triangulated Planar Graphs”, Jour. Inst. Math. & Computer Sciences, (Comp. Sc. Ser.)Vol. 15, No. 1 , 2004, pp. 137-147.
[22] A. Rosete, A. Ochoa, “G enetic Graph Drawing”, Proceeding of the 13th International Conference of Applications of Artificial Intelligence in Engineering, Galway, 1998, pp.37-41, citeseer.nj.nec.com/ rosete98genetic.html
[23] G. Sander. Graph layout for applications in compiler construction. Theoretical Computer Science, 217, 1999, pp. 175–214.
[24] K. Sugiyama and K. Misue. Visualization of structural information: Automatic drawing of compound digraphs. IEEE Transactions on S ystems, Man, and Cybernetics, 21(4), July 1991, pp. 876–892.
[25] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical systems. IEEE Trans. Syst., Man, Cybern., 11(2), 1981, pp.109-125.
[26] A. Yamaguchi and A. Sugimoto, “An Approximation Algorithm for the Two-Layered Graph Drawing Problems”, COCOON ’99, (Lecture Notes in Computer Sciences, 1627) , 1991 , pp. 81-91.