Genetic algorithms for the 2-page book drawing problem of graphs *

Hongmei He[1], Ondrej Sýkora[1], Erkki Mäkinen[2]
[1] Department of Computer Science, Loughborough University
Loughborough, Leicestershire LE11 3TU, The United Kingdom
[2] Department of Computer Sciences, P.O. Box 607,
FIN-33014 University of Tampere, Finland
[1]*H.He@lboro.ac.uk,* [2]*em@cs.uta.fi*

Abstract

The minimisation of edge crossings in a book drawing of a graph is one of the important goals for a linear VLSI design, and the 2-page crossing number of a graph provides an upper bound for the standard planar crossing number. We design genetic algorithms for the 2-page drawings, and test them on the benchmark test suits, Rome graphs and Random Connected Graphs. We also test some circulant graphs, and get better results than previously presented in the literature. Moreover, we formalise three conjectures for certain kinds of circulant graphs, supported by our experimental results.

Keywords: genetic algorithms, 2-page crossing number, Hamiltonian cycle, order of vertices, edge distribution, optimal values.

Introduction

The simplest graph drawing method is that of putting the vertices of a graph on a line and drawing the edges as half-circles. Such drawings are called "book drawings", and they correspond to the linear VLSI design. Edge crossing minimisation is the most important goal in linear VLSI design, since a smaller number of crossings means cheaper design. The minimal number of edge crossings in a book drawing is called the book crossing number (Shahrokhi et al. 1996).

*This research was supported by the EPSRC grant GR/S76694/01, accepted by the Journal of Heuristics.

In the 2-page drawing one places the vertices of a graph G along a line called a 'spine' and every edge is completely drawn in one of two pages. The smallest number of crossings in such a drawing of G is called the 2-page crossing number of G, denoted by $\nu_2(G)$. Equivalently, the vertices can be put on a circle and the edges can be drawn as straight lines and coloured by two colours. The 2-page crossing number is the same as the smallest number of crossings of edges with the same colour. The problem is NP-hard (Masuda et al. 1990).

Genetic algorithms (GAs) have shown to be good global optimizers for a broad range of optimisation problems, and they have been used successfully for drawing graphs (Barreto & Barbosa 2000, Eloranta & Mäkinen 2001, Huang & Kang 1998, Radwan & El-Sayed 2004).

In this paper, we design genetic algorithms to find an ordering of vertices and a distribution of edges in the two pages in order to minimise the 2-page crossing number. Our experiments are based on the benchmark test suits: *Random Connected Graphs* (RCG) (He et al. 2005a, 2005b) and *Rome Graphs* (GDToolKit).

The 2-page crossing number of a graph G provides an upper bound for the standard planar crossing number. Recently, Winterbach (2005) proposed heuristics for the 2-page crossing numbers and applied them to estimating the plane crossing number of some small complete multipartite graphs. Cimikowski (2002) tested eight different heuristic algorithms where the order of vertices was determined by finding a Hamiltonian cycle. Hence, contrary to our approach, in his tests the order of vertices was always *fixed*. We also give some explorations to circulant graphs, and compare our results with results of other authors, e.g., Cimikowski (2002), and the theoretical results published by Lin et al. (2005).

1 Solving the 2-page drawing problem with GA

In genetic algorithms, the first population of solutions is often generated randomly. Fitness, as a measure of quality of solutions, usually expressed in the form of one or multiple functions, is used to select the better solutions from the current population. The selected solutions undergo the operators of crossover and mutation in order to create a population of new solutions (the offspring population). The procedure is repeated until the termination criteria given by the user are met.

One of the most important questions is that of determining the characteristics of a problem, which makes it well-fitted for the genetic approach. The 2-page drawing problem is to find a good order of vertices and a good distribution of edges for a graph so that the 2-page crossing number of the graph is as small as possible. According to the problem we need to solve, we define four important aspects of genetic algorithms for the 2-page drawing problem: chromosome, fitness function, genetic operations, and termination criteria.

1.1 The chromosome

TimGA (Eloranta & Mäkinen 2001) drew graphs in an $N \times N$ matrix, and the authors presented a graph with n vertices and m edges by using a $2 \times n$ matrix to indicate the positions of vertices, and a $2 \times m$ matrix to indicate the edges by storing pairs of vertices. We use a list of edges, $e(u, v)$, to present a graph $G = (V, E)$, where $u, v \in V$ and $e(u, v) \in E$. For a 2-page drawing, the vertices of a graph are placed along a spine and edges are allocated in two pages. The key point of the 2-page drawing problem is to find an ordering of vertices and a distribution of edges minimising the 2-page crossing number. So, a chromosome should include two parts, a permutation of all vertices, $\pi = (v_0, v_1, \ldots, v_{n-1})$, and a string, $S = (b_0, b_1, \ldots, b_{m-1})$, where $b_i \in \{1, 2\}$. Each bit corresponds to one edge, i.e. b_i=1 indicates the corresponding edge is in page 1, and b_i=2 indicates the corresponding edge is in page 2. If we consider a κ page drawing, it is easy to extend our genetic algorithms by setting $b_i \in \{1, 2, \ldots, \kappa\}$.

The size of population, *popSize*, is an important parameter. A small population indicates that the variation of chromosomes is small, and that the search time is short, but it may lead to a premature convergence of solutions. A large population indicates that the variation of chromosomes is large, and that the search time is longer, but it may get better solutions. Since the rate of the number of individuals searched and the search space is larger, the chance of getting the best solution is larger. A larger graph has a larger search space. During the procedure, the total number of individuals that GA generates is *popSize* $\times$ *generations*. Considering the chance of getting the best solution for each graph to be as equal as possible, we fix a small size of population, and set *popSize*=16 in our experiments. The number of generations is decided by the termination criteria, which are related to the size of graphs as described in Section 1.4.

1.2 The fitness function

Fitness functions are used to evaluate the quality of solutions. Our goal is to minimise the 2-page crossing number, so we can directly define the 2-page crossing number ν_2 as the objective function (to be minimised) and the inverse of one plus the square of the 2-page crossing number as the fitness function (to be maximised). The reason for using the square of crossing number is to enlarge the difference of fitness values for similar crossing numbers. Our preliminary experiments showed that the results with squared crossing numbers were better than those with linear crossing numbers. The fitness function, $f(G, \pi, S)$, depends on the vertex order, π, and the edge distribution, S, in the current layout of a graph G. We use a table, *adj*, as an adjacent matrix in the current drawing $D(\pi, S)$ of G. If an element of S, which corresponds to an edge $e(u, v)$, has the value x (i.e. the edge $e(u, v)$ is drawn in page x, with $x = 1, \ldots, \kappa$), and the vertex u is in position i and the vertex v in position j in the current permutation π, then we set $adj[i][j] = adj[j][i] = x$. Otherwise, we set $adj[i][j] = adj[j][i] = 0$. We can calculate the number of crossings in a κ-page drawing of G with the following formula (He et al. 2005b):

$$v_\kappa(G) = \sum_{i=0}^{n-4} \sum_{j=i+2}^{n-2} \sum_{k=i+1}^{j} \sum_{l=j+1}^{n-1} adj[i][j] \bigodot adj[k][l] \qquad (1)$$

where

$$adj[i][j] \bigodot adj[k][l] = \begin{cases} 1 & \text{if } adj[i][k] = adj[j][l] \neq 0 \\ 0 & \text{if } \textit{otherwise}. \end{cases} \qquad (2)$$

The calculation takes time $O(n^2)$.

1.3 The genetic operators

Selection

Various selection criteria can be used so that sufficiently good individuals are picked for mating (and subsequent crossover). In our problem, we set a probability, *prob*, to decide the selection operator, and the probability of each individual in the current population is in proportion to the fitness value, which is defined as $\frac{1}{1+\nu_2^2(D)}$. The smaller the crossing number is, the larger is the probability. The probability can be calculated by the following formula, where D_i is the i-th drawing corresponding to the i-th individual in a population:

$$prob(D_i) = \frac{\frac{1}{\nu_2^2(D_i)+1}}{\sum_{k=0}^{popSize-1} \frac{1}{\nu_2^2(D_k)+1}} \times 100\%. \tag{3}$$

When a random number is produced, and it is located in the probability range of a chromosome, the chromosome will be selected. The selection operator is similar with the one used by He et al. (2005a) for the outerplanar drawing problem (Algorithm 1). The running time is $O(popSize)$.

Algorithm 1 Select($pops$)

```
1: $r$=random() mod 100;
2: $lastprob = pops[0].prob$;
3: $i$ =0;
4: while ($r > lastprob$) do
5:     $i = i + 1$;
6:     $lastprob = lastprob + pops[i].prob$;
7: end while
8: return $i$;
```

Crossover

The purpose of crossover is to create new solutions by combining previous solutions that have shown to be good temporary solutions. Depending on the presentation of chromosomes, different crossover operators are used. Eloranta and Mäkinen (2001) used two types of crossover operators, *RectCrossover* and *ThreeNodeCrossover*. For the 2-page drawing problem, the chromosome includes two parts, permutation of vertices and distribution of edges, so the crossover will act on both parts. Our preliminary tests showed that the following crossover types for the two parts of a chromosome are adequate. In the implementation, we use two circle queues to maintain the permutation and edge distribution, respectively, so that the variation of permutation and edge distribution by crossover operators is double as that by using normal queues. Indeed, no matter what the presentation of chromosome, if a circle queue is used, then any segment in the circle queue can be chosen randomly for crossover, including the connection of head and tail of the circle queue.

Crossover on permutation, π

Here we use Multi-point Order crossover(MOX) (Michalewicz 1994): two parental permutations, π_1 and π_2, are chosen randomly depending on the probability of being chosen. A continuous interval of the permutation π_1 is chosen, and also an interval starting at the same position and of the same length from π_2. Two new permutations, π_1' and π_2', are created such that π_1' contains the interval from π_2 with the rest being the other elements of π_1 in the order as they were in π_1. π_2' contains the interval from π_1 with the rest being the other elements of π_2 in the order as they were in π_2. The crossover operator on π is similar with the one of He et al. (2005a) (Algorithm 2). An example is given in Fig. 1.

Algorithm 2 Crossover(π_1,π_2)

```
1: $start = random()$ mod $n$;
2: $len = random()$ mod $n$;
3: Swap parts from $start$ to $(start + len)$ mod $n$ in $\pi_1$ and $\pi_2$;
4: Complete the permutation with the rest of the original permutation
   (i.e. the other elements are put into the unused space according to the
   original ordering);
```

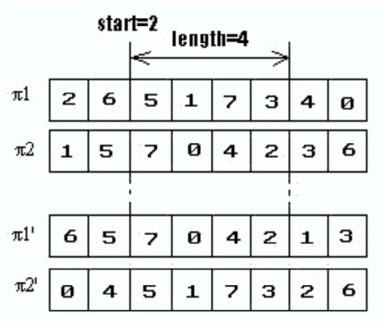

Figure 1: An example of the crossover operation

Crossover on page distribution of edges, S

We use Multi-Point crossover (MPX) (Michalewicz 1994) on two parental strings, s_1 and s_2. A continuous interval of the string s_1 is chosen, and also an interval starting at the same position and of the same length from s_2. Two parents, s_1 and s_2, swap the two selected intervals to get two new distributions of edges, s_1' and s_2' (see Algorithm 3).

Algorithm 3 Crossover(s_1,s_2)

```
1: $start = random()$ mod $m$;
2: $len = random()$ mod $m$;
3: Swap the parts from $start$ to $((start + len)$ mod $m)$ in $s_1$ and $s_2$;
```

Mutation

After creating *popSize* children by crossover operator, the step of mutation is executed. The mutation is done with some probability on each child in the population. Actually, there is a small chance that the crossover on π gets synchronous performance as the crossover on S. Our preliminary experiments show that a large probability of mutation is needed. Good results are achieved where the probability of mutation is as high as 40%. The mutation operator acts on both parts of the chromosome, π and S. On π, the mutation is the swap of two randomly picked elements in a permutation. On S, the mutation is the change of a randomly picked element in a string, which indicates that the corresponding edge is changed to the opposite page from its current page. Finally, the historically best individual will replace the worst one in the new generation.

1.4 The terminate criteria

The termination criteria are important parameters, which determine the running time and the final result of the algorithm. For the 2-page drawing problem, one of the termination criteria is usually related to the number of edges and vertices. We terminate the GA procedure when the chance of improvement is close to 0. In the implementation, we define: $duration = min(3n + 3m + 100, 1000)$.

The evolution procedure will be repeated until the best solution shows no improvement up to *duration* generations or the minimal crossing number is 0. In this way, if the *duration* is large, the evolution will run for a long time, but might get better solutions. The *duration* parameter is fixed for a graph, but the exact number of generations is not only related to evolution procedure, but also related to the randomness of evolution in each run. Our preliminary tests showed that this termination criterion allows sufficiently long running time for obtaining good solutions.

2 Different Implementations of GA

2.1 GA with SLOPE 2-page strategy

The SLOPE 2-page strategy (He et al. 2005b) distributes the edges between pages according to their slopes. If the angle between an edge and the horizontal axis is larger than 90°, the edge is put on page 2 (solid edges in Fig.2), otherwise the edge is put on page 1 (broken edges in Fig. 2).

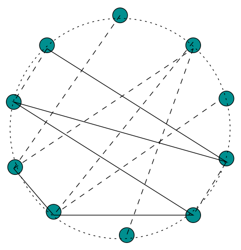

Figure 2: SLOPE distribution strategy

We can divide a circle into four sections as in Fig. 3. Suppose a vertex's position on the circle is i, Table 1 describes how the positions of vertices are related to the four sections.

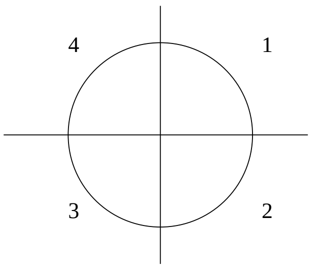

Figure 3: Dividing a circle into 4 sections

Considering an edge $e(u, v)$, u's position is i and v's position is j, if i and j are both located in the section 1, then $0 < i + j < \frac{1}{2}n$. If i and j are

Table 1: Vertex positions on each section of a circle

Section	Positions
1	$0 \le i < \frac{n}{4}$
2	$\frac{n}{4} \le i < \frac{n}{2}$
3	$\frac{n}{2} \le i < \frac{3}{4}n$
4	$\frac{3}{4}n \le i < n$

both located in the section 3, then $n \le i + j < \frac{3}{2}n$. If i and j are located in section 2 and section 4, respectively, then $n \le i + j < \frac{3}{2}n$. In these cases, we roughly say the angle between the edge and horizontal axis is larger than 90°. Therefore, according to the positions of vertices incident to the edge, we obtain the rough relationship between the slope value and the positions of two ends of the edge, if ($0 < i + j < \frac{1}{2}n$ or $n \le i + j < \frac{3}{2}n$) the angle between the edge and horizontal axis is larger than 90°. This resembles the GA for outerplanar drawings defined by He et al. (2005a). The chromosome includes only one part, permutation π. Crossover and mutation operators only act on π. Just before the fitness calculation for each individual, edges will be arranged with the SLOPE strategy. We denote this as GA-2Ptg.

2.2 GA operators run in different ways

Normally in the GA for the 2-page drawing problem, the search space is implicitly defined by the order of vertices and the distribution of edges. We assume that the search space related to the ordering of vertices is $SP_v = \{0..n-1\}^n$, where $n = |V|$, and that the search space related to the edges is $SP_e = \{1, 2\}^m$, where $m = |E|$. $|SP_v| = n!$, $|SP_e| = 2^m$. If the degree of a vertex is d, then the change of one vertex position, is equivalent to the possible change of 2^d edge distributions. So the vertex order evolution has more effect on the crossings than the edge distribution does. We use four models, in which the GA operators run in different ways.

Crossover and mutation on π and S in each generation

For this model, in every generation, the crossover operator includes two sub-crossovers, operating on π and S, respectively. The crossover operator and mutation run on both π and S. We denote this as GA_2P. Obviously, the crossover and mutation run on the space $SP_v \times SP_e$.

Algorithm 4 *Slope*(*pops*[*k*])

```
{
define a 2-dim array, $adj$
for ($i$=0 to $|V|$) do
  for ($j$=0 to $|V|$) do
    $u = pops[k].order[i]$; $v = pops[k].order[j]$;
    if $e(u,v) \in E$ then
      if ($i + j < \frac{1}{2}|V|$) or ($|V| \leq i + j < \frac{3}{2}|V|$) then
        $adj[i][j] = adj[j][i] = 2$;
      else
        $adj[i][j] = adj[j][i] = 1$;
      end if
    else
      $adj[i][j] = adj[j][i] = 0$;
    end if
  end for
end for
return $adj$;
}
```

Crossover and mutation on π and S alternatively

In this model, the crossover and mutation run on π and S alternately. Namely, first crossover and mutation operate on π until the best solution shows no improvement up to L generations, which is an experimental parameter ($L = 20$), and then crossover and mutation operate on S until the best solution shows no improvement up to L generations. The whole procedure will be repeated until the termination criteria are met, the search space alternating between SP_v and SP_e. We denote this as GA_2Px.

π has priority over S

In this model, the crossover and mutation mainly run on π, and the search space is decided by the order of vertices. When the best solution shows no improvement up to L generations, the crossover and mutation operate on S once, so that the search jumps out of the local SP_v space. The procedure is repeated until the termination criteria are met. We denote this as GA_2Pxv.

S has priority over π

This model is reverse as the model above, crossover and mutation mainly run on S, and the search space is decided by the distribution of edges. When the best solution shows no improvement up to L generations, the crossover and mutation are done on π once, so that the search jumps out of the local SP_e space. The procedure is repeated until the termination criteria are met. We denote this as GA_2Pxe.

3 Experimental Results

3.1 Rome Graphs and RCGs

We use two sets of undirected graphs from Rome graphs (GDToolKit):

* **RND_BUP:** this graph set contains about 200 graphs generated randomly. Each graph in the set is biconnected, undirected and planar.

* **ALF_CU:** this graph set contains about 10,000 connected and undirected graphs.

RCGs is a library of undirected random connected graphs with different sizes and different densities. In Table 2, the number of times that each GA model achieves the best results on the benchmark graph libraries are listed, and in Table 3, the numbers of sample graphs for which each GA model runs the smallest number of generations on the benchmark graph libraries are listed.

Table 2: The number of times that each GA model achieves the best results on benchmark graph libraries

Graphs	2P	2Ptg	2Px	2Pxe	2Pxv
ALF_CU (268)	147	134	127	20	137
RND_BUP(169)	74	63	65	13	78
RCG(360)	67	171	26	0	133
total	288	368	218	33	348

For Rome graphs, models GA_2P, GA_2Ptg, GA_2Px, and GA_2Pxv have similar performance both in results and in the number of generations. Our other tests indicate that for RCG graphs, GA_2Ptg gets the best results, and for all graphs, GA_2Pxe gets the worst results. We can conclude that GA_2Pxe converges prematurely, and we exclude it from our further tests.

Table 3: The numbers of sample graphs for which each GA model runs the smallest number of generations on benchmark graph libraries

Graphs	2P	2Ptg	2Px	2Pxe	2Pxv
ALF_CU	37	58	32	116	43
RND_BUP	25	26	21	80	22
RCG	13	19	9	300	19
total	75	103	62	496	84

3.2 Test on RCGs with different densities and sizes

Next we examine the effect and efficiency of all GA models by statistical results of crossing number and number of generations on RCGs with different sizes and different densities, where density of a graph G with n vertices and m edges is $m/|E(K_n)|$, and K_n is the n-vertex complete graph. We use three densities: 1%, 2%, and 5%. For each density, 12 groups of graphs with different number of vertices are tested, and for every group 10 different graphs are generated and average number of generations and average number of crossings are calculated.

As shown in Figs. 4, 5 and 6, for all densities, model GA_2Ptg always gets the best results. Usually, from the best to the worst, we find the order: GA_2Ptg $\succ$ GA_2Px $\succ$ GA_2P $\succ$ GA_2Pxv. With the rise of density, the results become closer, but the numbers of generations needed differ substantially:

- When density is 1%, the numbers of generations of the four models are quite similar. But occasionally, GA_2Ptg has the largest number of generations, and GA_2Pxv has the smallest number of generations.
- When density is 2%, GA_2P and GA_2Ptg have the smallest numbers of generations. GA_2Px and GA_2Pxv have about the same numbers of generations, but occasionally, they have about the same number of generations as GA_2Ptg and GA_2P do.
- When density is 5%, GA_2P and GA_2Ptg remain similar and have the smallest number of generations. With the rise of vertex number, GA_2Px has a much larger number of generations than the others.

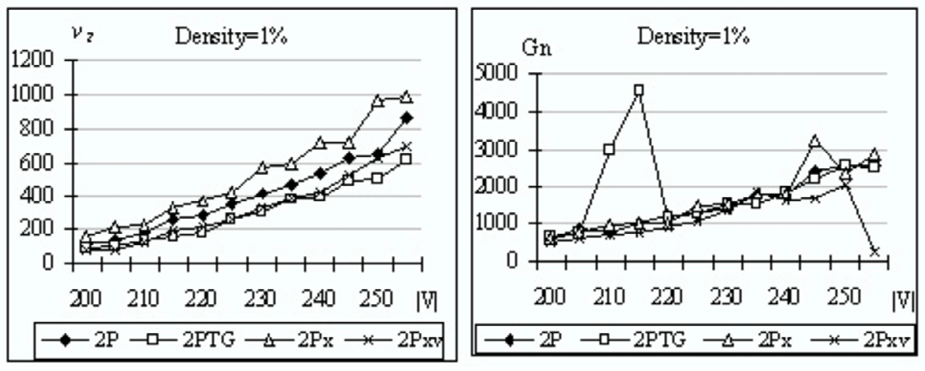

Figure 4: Average numbers of 2-page crossings and numbers of generations on RCG(1%)

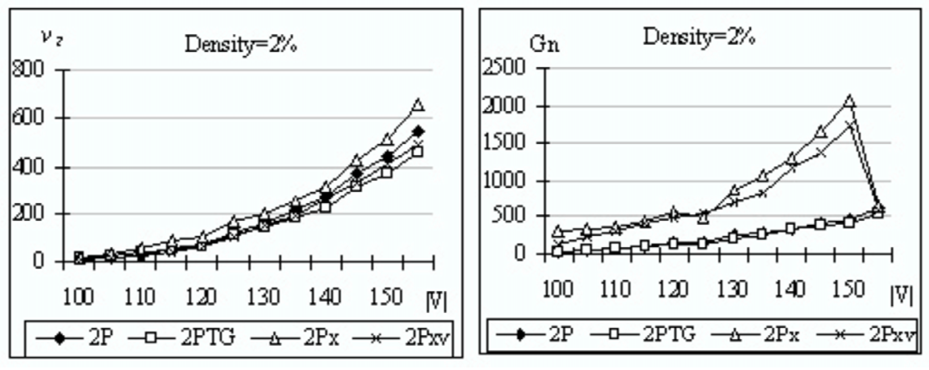

Figure 5: Average numbers of 2-page crossings and numbers of generations on RCG(2%)

3.3 Test on circulant graphs

Circulant graphs with the form, $Cn(a_1, a_2, ..., a_k)$, where $0 < a_1 < a_2 < ... < a_k < (n+1)/2$, are regular Hamiltonian graphs with n vertices, and with vertices $i \pm a_k(\bmod n)$ adjacent to each i (Fig. 7).

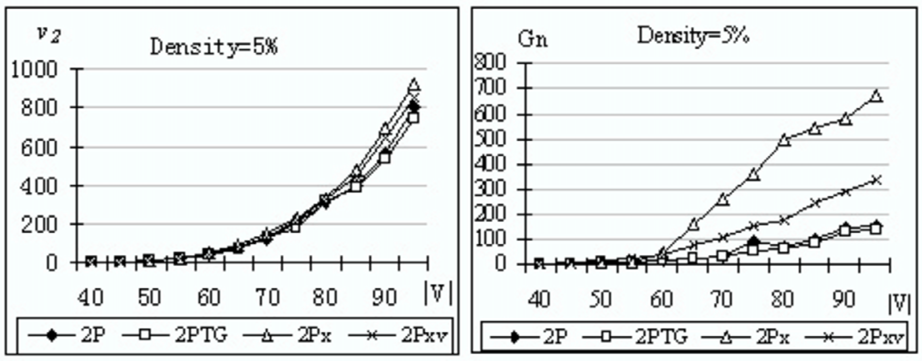

Figure 6: Average numbers of 2-page crossings and numbers of generations on RCG(5%)

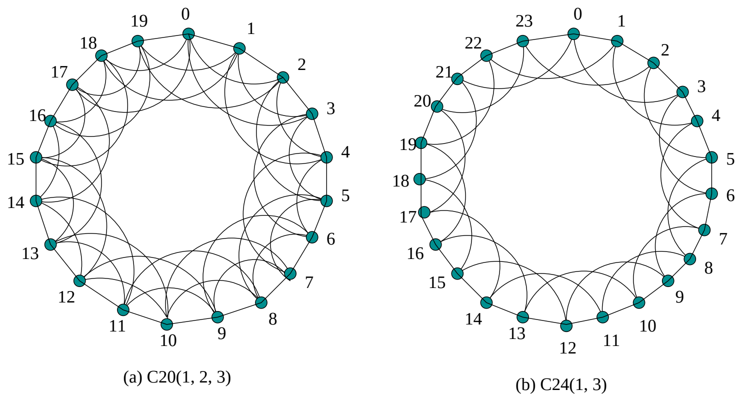

Figure 7: Two circulant graphs

The experiments of Cimikowski (2002) were done based on a fixed Hamiltonian cycle for some special structural graphs. However, not every Hamiltonian cycle corresponds to an optimal vertex order, and an optimal drawing might not correspond to a Hamiltonian cycle. Moreover, for an arbitrary graph, a Hamiltonian cycle might not exist, or even if it exists, we might not be able to find it. Our experiments aim at finding a 2-page drawing with 2-page crossings as small as possible for a graph. We test the circulant graphs used by Cimikowski (2002) with the four GA models.

According to the percentage of the results, which are the same as or better than the best results listed by Cimikowski (2002), accounting for all test results, we get the order of the four models: GA_2Ptg(94%) $\succ$ GA_2P(91%)

$\doteq$ GA_2Px(91%) $\succ$ GA_2Pxv(26%). According to the number of times that each model gets the best results, we get the same order: GA_2Ptg(21) $\succ$ GA_2P(18) $\succ$ GA_2Px(8) $\succ$ GA_2Pxv(0). In the second rightmost column of Table 4, there are listed either the optimal values related to the fixed order of vertices based on the Hamiltonian cycles (Cimikowski 2002), or theoretical lower and upper bounds, if the branch-and-bound algorithm of Cimikowski (2002) was not applicable. The rightmost column contains the best results obtained by Cimikowski. According to the data, we get the chart shown in Fig. 8. All of our best results are better than the best results of Cimikowski related to the fixed orders of vertices. Fig. 9 and Fig. 10 show the best solutions for Fig. 7 (a) and (b), respectively, both crossing numbers of which are better than the optimal values related to fixed orders of vertices (Cimikowski 2002). In the drawing of Fig. 9, the order of vertices is a Hamiltonian cycle, but in Fig. 10 is not.

Table 4 shows that our heuristics overcome the fundamental problem of Cimikow-ski's method: a heuristic based on a fixed vertex order seems to be too restrictive for the present problem. The circulant graphs that Cimikowski used were relatively small, and since the termination criterion of the genetic algorithm is directly proportional to the graph size, the running time of the genetic algorithm is quite short on a high-speed computer.

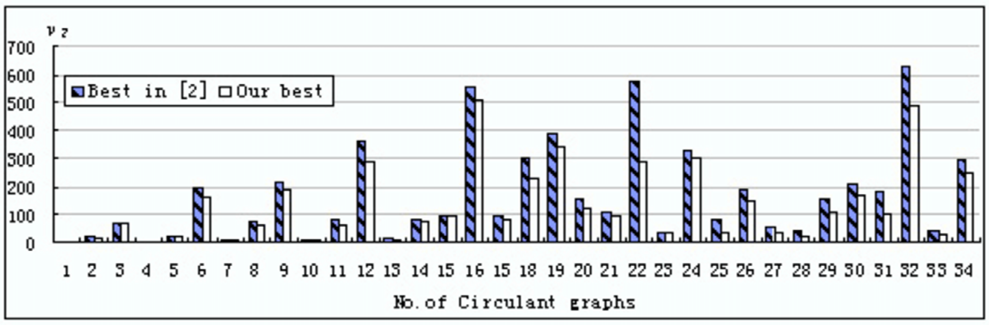

Figure 8: Our best results and Cimikowski's best results

Table 4: 2-page crossings for *circulants* used by Cimikowski (2002). The column C-opt contains the optimal values (if available) related to a fixed order of vertices or theoretical lower and upper bounds, and the column C-best contains the best values obtained by Cimikowski (2002)

No.	Graphs	GA_2P	GA_2Px	GA_2Pxv	GA_2Ptg	C-opt.	C-best
1	$C_{20}(1,2)$	2	0	2	0	0	0
2	$C_{20}(1,2,3)$	18	18	31	18	22	22
3	$C_{20}(1,2,3,4)$	68	70	102	68	26 : 870	70
4	$C_{22}(1,2)$	0	0	2	0	0	0
5	$C_{22}(1,2,3)$	22	20	35	20	24	24
6	$C_{22}(1,3,5,7)$	166	199	252	172	28 : 1056	200
7	$C_{24}(1,3)$	9	9	10	9	12	12
8	$C_{24}(1,3,5)$	60	64	83	62	72	76
9	$C_{24}(1,3,5,7)$	219	204	255	193	30 : 1260	216
10	$C_{26}(1,3)$	10	10	12	10	14	14
11	$C_{26}(1,3,5)$	63	68	92	64	6 : 650	82
12	$C_{26}(1,4,7,9)$	316	319	391	290	32 : 1482	364
13	$C_{28}(1,3)$	12	12	13	11	14	16
14	$C_{28}(1,3,5)$	79	77	97	75	6 : 756	86
15	$C_{28}(1,2,3,4)$	97	103	140	103	34 : 1722	98
16	$C_{28}(1,3,5,7,9)$	524	555	746	508	62 : 3080	560
17	$C_{30}(1,3,5)$	84	84	109	83	6 : 870	96
18	$C_{30}(1,3,5,8)$	252	254	332	226	36 : 1980	302
19	$C_{30}(1,2,4,5,7)$	352	361	492	346	66 : 3540	392
20	$C_{32}(1,2,4,6)$	124	129	234	134	38 : 2256	160
21	$C_{34}(1,3,5)$	96	98	122	96	6 : 1122	106
22	$C_{34}(1,4,8,12)$	286	314	394	302	40 : 2550	574
23	$C_{36}(1,2,4)$	36	42	67	42	6 : 1260	36
24	$C_{36}(1,3,5,7)$	317	320	406	301	42 : 2862	328
25	$C_{38}(1,7)$	36	42	54	37	84	86
26	$C_{38}(1,4,7)$	149	157	221	155	6 : 1406	190
27	$C_{40}(1,5)$	32	29	42	36	56	58
28	$C_{42}(1,4)$	24	25	31	28	42	42
29	$C_{42}(1,3,6)$	109	113	174	106	6 : 1722	158
30	$C_{42}(1,2,4,6)$	168	215	334	174	48 : 3906	210
31	$C_{44}(1,4,5)$	99	118	141	112	6 : 1892	180
32	$C_{44}(1,4,7,10)$	505	514	762	491	50 : 4290	632
33	$C_{46}(1,4)$	31	29	45	29	46	46
34	$C_{46}(1,5,8)$	268	274	303	246	6 : 2070	296

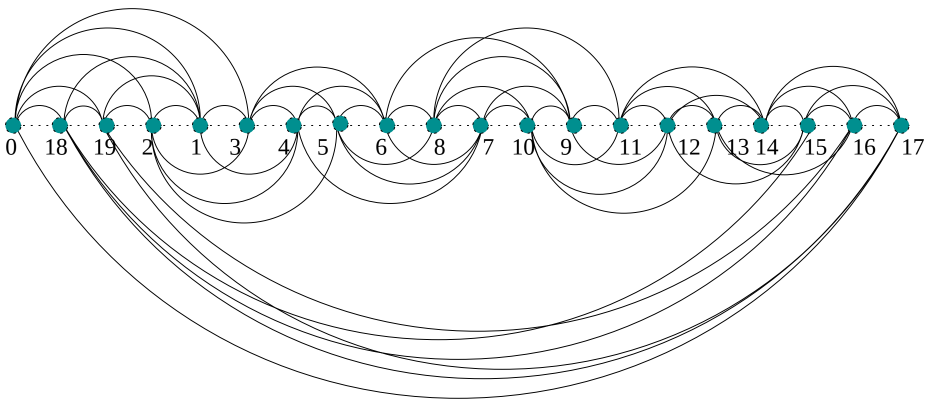

Figure 9: The best solution of $C_{20}(1, 2, 3)$ (Fig. 7 (a)), $\nu_2 = 18$

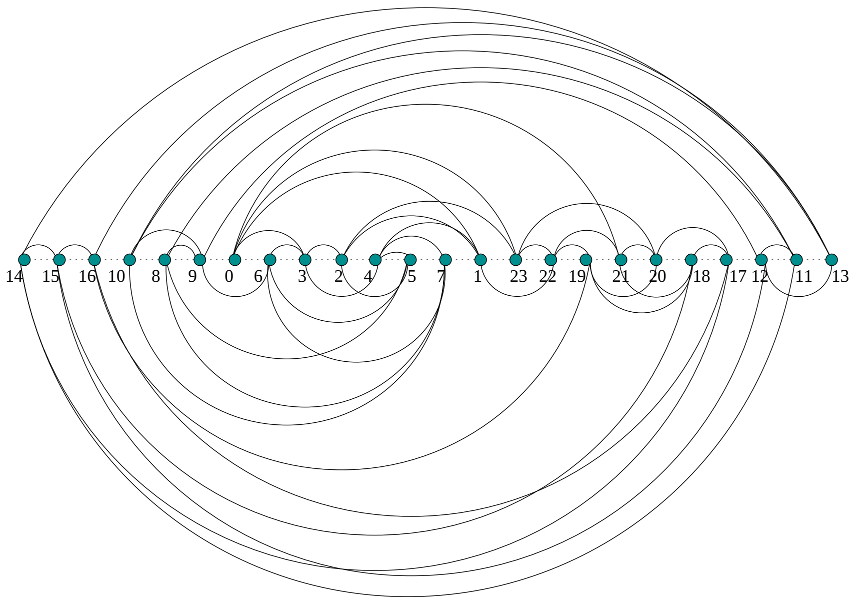

Figure 10: The best solution of $C_{24}(1, 3)$ (Fig. 7 (b)), $\nu_2 = 9$

3.4 Three conjectures

Winterbach (2005) proposed heuristics for the 2-page crossing number and applied them to estimate the planar crossing number of some small complete multipartite graphs. In contrast, we use the standard crossing number of some graphs with our experimental results to conjecture the optimal 2-page crossing number of the graphs.

Winterbach's test was based on the complete multipartite graphs with $|V| = 6 \sim 13$, except for the complete multipartite graphs that were isomorphic to the corresponding complete graphs with the same vertex number. A tabu algorithm was used to produce a good arrangement of vertices, and then either GreedySide algorithm or the neural network of Cimikowski and Shope (1996) was used to find a good distribution of edges. Our experiments use the genetic algorithm to find a good vertex order and edge distribution directly, and we get exactly the same results for those complete multipartite graphs tested by Winterbach with the genetic algorithm GA_2Ptg, except for the graph $K_{11}(2, 2, 2, 2, 2, 1)$, for which we obtained 61 and not 60.

We also tested another family of circulant graphs $C_{mk}(1, k)$(see Fig. 11), which are regular Hamiltonian graphs with $n = km$ vertices, $V \longrightarrow \{0, 1, 2, ..., km-1\}$, where $E \longrightarrow \{(i, i + 1), (i, (i + k) \bmod n)|i \in V\}$. Table 5 shows the results of our experiments.

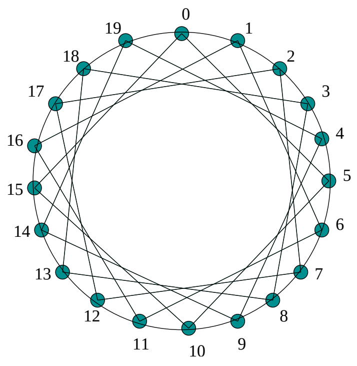

Figure 11: $C_{4\times 5}(1, 5)$

We know the crossing number of circulant graphs, $cr(C_n(1, \lfloor n/2 \rfloor)) = 1$ (Yang & Zhao 2001). Therefore, for this family of circulant graphs, $C_{2k}(1, k)$(see

Fig. 12), the 2-page crossing number is at least 1. From our experimental results in Table 5, we obtain the following conjecture:

Conjecture 3.1 $\nu_2(C_{2k}(1,k)) = 1$.

Similarly, we know the crossing number of circulant graphs, $cr(C_{3k}(1,k)) = k$, for k > 3 (Lin et al. 2005). So we have $\nu_2(C_{3k}(1,k)) \geq k$, for k > 3 (see Fig. 13). And from our experimental results in Table 5, we have the following conjecture:

Conjecture 3.2 $\nu_2(C_{3k}(1,k)) = k$.

Moreover, from our experimental results, we have a conjecture for $C_{4k}(1,k)$, $k > 4$, as follows (see Fig. 14):

Conjecture 3.3 $\nu_2(C_{4k}(1,k)) = 2k + 1$.

Table 5: 2-page crossing number of $C_{mk}(1,k)$ by all GAs

No.	Graphs	GA_2P	GA_2Px	GA_2Pxv	GA_2Ptg
1	$C_{2\times3}(1,3)$	1	1	1	1
2	$C_{2\times4}(1,4)$	1	1	1	1
3	$C_{2\times5}(1,5)$	1	1	1	1
4	$C_{2\times6}(1,6)$	1	1	1	1
5	$C_{2\times7}(1,7)$	1	1	1	1
6	$C_{2\times8}(1,8)$	1	1	1	1
7	$C_{2\times9}(1,9)$	1	1	1	1
8	$C_{3\times3}(1,3)$	3	3	3	3
9	$C_{3\times4}(1,4)$	4	4	4	4
10	$C_{3\times5}(1,5)$	5	5	5	5
11	$C_{3\times6}(1,6)$	6	6	6	6
12	$C_{3\times7}(1,7)$	7	7	7	7
13	$C_{3\times8}(1,8)$	8	8	8	8
14	$C_{3\times9}(1,9)$	9	9	10	9
15	$C_{4\times3}(1,3)$	4	4	5	4
16	$C_{4\times4}(1,4)$	9	9	9	8
17	$C_{4\times5}(1,5)$	11	11	14	11
18	$C_{4\times6}(1,6)$	13	13	14	13
19	$C_{4\times7}(1,7)$	17	15	18	15
20	$C_{4\times8}(1,8)$	17	17	24	18
21	$C_{4\times9}(1,9)$	19	21	24	19

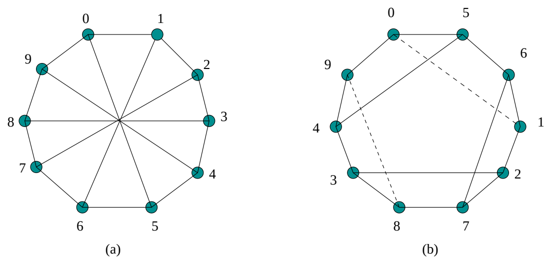

Figure 12: Drawings of $C_{2\times 5}(1,5)$, $\nu_2 = 1$

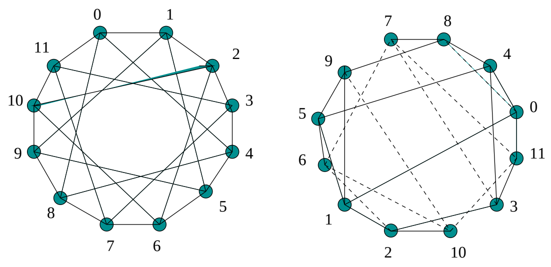

Figure 13: Drawings of $C_{3\times 4}(1,4)$, $\nu_2 = 4$

4 Conclusions

Models GA_2P, GA_2Pxv, and GA_2Ptg are the best for the graph sets ALF_CU, RND_BUP, and RCG, respectively. When the density of graphs increases from 1 % to 5 %, the difference between the results remains quite modest for RCGs. However, the numbers of generations needed are much more unbalanced when the density of graphs varies. For sparse graphs (density 1 %), the number of generations needed by GA_2Ptg has the largest fluctuation. With denser graphs (density 5 %), GA_2Px needs considerably more generations than the other models. The fluctuation on sparse graphs might indicate that the edge distribution limits the possible vertex orders very little, and that the evolution process is more random than with denser

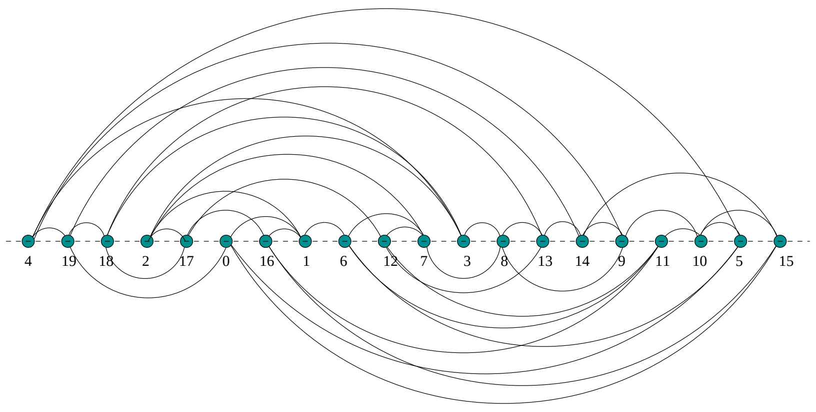

Figure 14: A drawing of $C_{4\times5}(1,5)$, $\nu_2 = 11$

graphs. It is also worth noticing that GA_2P is stable on all RCG graphs. In most cases, GA_2Ptg has as many generations as GA_2P, and when density is 2% and 5%, their numbers of generations needed are smaller than with the other models.

For circulant graphs, from the best to the worst, there is the relationship: GA_2Ptg $\succ$ GA_2P $\succ$ GA_2Px $\succ$ GA_2Pxv. Moreover, our experiments of GAs even improve optimal values based on fixed orders of vertices of some circulant graphs used by Cimikowski (2002). We have three conjectures for $C_{2k}(1,k)$, $C_{3k}(1,k)$ and $C_{4k}(1,k)$, supported by our experimental results. We will give proofs for the conjectures in our further work.

References

Barreto, A. M. S. & Barbosa, H. J. C. (2000), 'Graph layout using a genetic algorithm', *in* Proc. the VI Brazilian Symposium on Neural Networks (SBRN'00), pp. 179–184.

Cimikowski, R. (2002), 'Algorithms for the fixed linear crossing number problem', *Discrete Applied Mathematics* **122**, pp. 93–115.

Cimikowski, R. & Shope, P. (1996), 'A neural network algorithm for a graph layout problem', *IEEE Trans. on Neural Networks* **7**(2), pp. 341–346.

Eloranta, T. & Mäkinen, E. (2001), 'Timga: a genetic algorithm for drawing undirected graphs', *Divulg. Mat.* **9**(2), pp. 155–170.

He, H., Newton, M. C. & Sýkora, O. (2005a), ‘Genetic algorithms for bipartite and outerplanar graph drawings are best!’, *in* Communications, SOFSEM’05, pp. 51–60.

He, H., Sykora, O. & Vrt’o, I. (2005b), ‘Heuristic crossing minimisation algorithms for 2-page drawings(extended abstract)’, *in* Proc. ICGT’05. The Electronic Notes in Discrete Mathematics (ENDM) **22**.

Huang, J. W. & Kang, L. S. (1998), ‘A graph drawing algorithm for general undirected graphs based on genetic algorithms’, *J. Math. (Wuhan)* **18**(suppl.), pp. 68–72.

Lin, X., Yang, Y., Lŭ, J. & Hao, X. (2005), ‘The crossing number of $c(mk; \{1, k\})$’, *Graphs and Combinatorics* **21**(1), pp. 89–96.

Masuda, S., Kashiwabara, T., Nakajima, K. & Fujisawa, T. (1990), ‘Crossing minimization in linear embeddings of graphs’, *IEEE Trans. Comput.* **39**, pp. 124–127.

Michalewicz, Z. (1994), ‘*Genetic Algorithms + Data Structures = Evolution Programs*’, second, extended edition edn, Springer.

Radwan, A. A. & El-Sayed, M. A. (2004), ‘Using genetic algorithm for drawing triangulated planar graphs’, *J. Inst. Math. Comput. Sci., Comput. Sci. Ser.* **15**(1), pp. 137–147.

Shahrokhi, F., Sýkora, O., Székely, L. & Vrt’o, I. (1996), ‘The book crossing number of a graph’, *Journal of Graph Theory* **21**, pp. 413–424.

Winterbach, W. (2005), ’The crossing number of a graph in the plane’, Master’s thesis, Dept. Appl. Math, University of Stellenbosch, SA.

Yang, Y. & Zhao, C. (2001), ‘The crossing number of $C_n(1, k)$’, *in* Proc. National Symposium on Software Technology, Chinese Computer Institute’, pp. 134–136.

GDToolKit: `http://www.dia.uniroma3.it/~gdt/`.