Research Article **Open Access**

Hongmei He*, Ana Sălăgean, Erkki Mäkinen, and Imrich Vrt'o

Various heuristic algorithms to minimise the two-page crossing numbers of graphs

DOI 10.1515/comp-2015-0004

Received March 16, 2012; accepted July 16, 2015

Abstract: We propose several new heuristics for the two-page book crossing problem, which are based on recent algorithms for the corresponding one-page problem. Especially, the neural network model for edge allocation is combined for the first time with various one-page algorithms. We investigate the performance of the new heuristics by testing them on various benchmark test suites. It is found out that the new heuristics outperform the previously known heuristics and produce good approximations of the planar crossing number for several well-known graph families. We conjecture that the optimal two-page drawing of a graph represents the planar drawing of the graph.

Keywords: heuristic algorithm; two-page book crossing number; one-page book crossing number; Hamiltonian cycle; planar drawing

1 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 in κ half planes (called pages). Such drawings are called κ-page book drawings, and they are used in the linear VLSI design. The minimal number of edge crossings over all κ-page book drawings of a graph is called the κ-page book crossing number [1].

The one-page [1] book crossing number corresponds to outerplanar [2] (also called convex [3], or circular [4]) crossing number, which is the minimal number of edge crossings in a drawing where one places the vertices of a graph G along a circle, and the edges are drawn as straight lines. Therefore, the one-page book crossing problem is equivalent to finding the order of the vertices on the circle which minimises the number of edge crossings. We denote the one-page book crossing number as $\nu_1(G)$, following the notation in [1]. The problem to determine $\nu_1(G)$ for a given graph G is NP-hard [5].

The two-page book crossing problem can be formulated analogously: one places the vertices of a graph G along a circle and each edge is drawn in one of two colours. The two-page book crossing number $\nu_2(G)$ of G is the smallest number of crossings of edges with the same colour. Naturally, this problem is also NP-hard [6]. Similarly, a κ-page drawing of a graph is equivalent to a κ-colour circular drawing of the graph.

For a graph G with n vertices labeled $0, 1, \ldots, n-1$ and m edges, we denote by g_adj the usual adjacency matrix. A vector $order$ will be defined such that $order[u]$ denotes the position of vertex u on the circle in the current drawing (positions being denoted $0, 1, \ldots, n-1$). The allocation of edges between the two pages is stored in a matrix d_adj. Matrix entry (i, j) indicates the page (colour) of the edge between vertices at positions i and j, with 0 meaning that no edge exists between these vertices. If (i, j) and (k, l) are edges on the same page, and $i < k < j < l$, then they produce a crossing. So we can calculate the number of crossings in a κ-page drawing of G with the following formula:

$$\nu_\kappa(G) = \sum_{i=0}^{n-4} \sum_{j=i+2}^{n-2} \sum_{k=i+1}^{j-1} \sum_{l=j+1}^{n-1} d\_adj(i,j) \bigodot d\_adj(k,l), \quad (1)$$

where

$$d\_adj(i,j) \bigodot d\_adj(k,l) = \begin{cases} 1, & \text{if } d\_adj(i,k) \\ & = d\_adj(j,l) \neq 0; \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

The crossing minimisation problem is of outmost importance in the field of graph drawings. The aesthetical properties and readability of graphs (such as diagrams or

***Corresponding Author: Hongmei He:** Cranfield University, Manufacturing Informatics Centre, Cranfield, MK43 0AL, UK, E-mail: h.he@cranfield.ac.uk
Ana Sălăgean: Loughborough University, Department of Computer Science, Loughborough, LE11 3TU, UK, E-mail: a.m.salagean@lboro.ac.uk
Erkki Mäkinen: Third institution, School of Information Sciences, University of Tampere, Tampere, FIN-33014, Finland, E-mail: em@sis.uta.fi
Imrich Vrt'o: Slovak Academy of Sciences, Department of Informatics, Institute of Mathematics, Bratislava, Slovak Republic, E-mail: imrich.vrto@savba.sk

maps) are heavily dependent on the number of crossings in a drawing. Crossing minimisation is also one of the most important goals in VLSI circuit design [7, 8] and Quantum-dot Cellular Automata (QCA) [9, 10], where a smaller crossing number means lower implementation cost. From the theoretical point of view, the one-page and two-page crossing number of a graph provide an upper bound of the general crossing number of the graph. Naturally, the two-page book crossing number of a graph provides a closer upper bound for the general crossing number of the graph than the corresponding one-page crossing number.

Several heuristic algorithms are designed for the one-page book crossing problem, e.g., the algorithm of Mäkinen [11], the CIRCULAR algorithm [4], the algorithm of Baur and Brandes [12], and AVSDF [13]. The two-page book crossing number as an approximation to the conjectured planar crossing number was first studied in [14, 15], but no thorough testing was done there. An important paper in this area is that by Cimikowski [16], where first an order of vertices for some structural graphs is given by a Hamiltonian cycle, and then eight different heuristic algorithms to find a good allocation of edges between the two pages were designed and tested. Winterbach [17] proposed heuristics for the two-page crossing numbers and applied them to estimate the planar crossing number of some small complete multipartite graphs. de Klerk and and Pasechnik [18] have recently obtained new lower bounds of two-page crossing number for $k_{m,n}$ and K_n. We have studied the two-page book crossing number problem using simulated annealing [19], genetic algorithm [20], and neural network [21]. However, these methods require long running time. Hence, we also investigated the parallelisation of genetic algorithms for the two-page crossing number problem [22, 23].

In this paper, we further study heuristic algorithms that combine the latest one-page drawing algorithms (BB and AVSDF) [12, 13] and four edge allocation algorithms, such as SLOPE, LEN, CRS and the neural network (NN) model [21], for the two-page crossing number problem, and compare the performance of edge allocation algorithms. Especially the NN as an edge allocation algorithm is combined for the first time with heuristic one-page algorithms. The experiments are carried out on the benchmark test suites Rome Graphs [24] and Random Connected Graphs (RCGs) [13]. In order to further examine the performance of edge allocation algorithms, we apply the edge allocation algorithms based on a fixed vertex order, which is either the optimal or a Hamiltonian order for the one-page drawing of some typical structural graphs.

This paper is organised as follows. In Sections 2 and 3 we introduce two known heuristic one-page algorithms and four new two-page algorithms. In Section 4, we examine two-page algorithms in different ways, compare the results for some circulant graphs with Cimikowski's results [16], and provide exact results for some structural graphs. In Section 5, we conclude our findings.

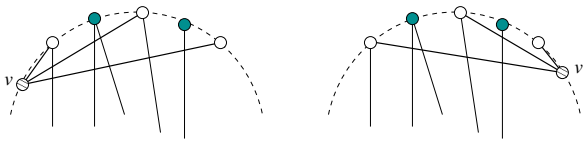


Figure 1: Incident edges of v cross with open edges.

2 One-page drawing algorithms

2.1 The algorithm of Baur and Brandes (BB+)

The best algorithm of Baur and Brandes [12] consists of two phases: greedy and sifting.

2.1.1 Greedy

The vertices are placed one by one on the circle as follows: at each step a vertex v with the largest number of already placed neighbours is selected, where ties are broken in favour of vertices with fewer unplaced neighbours. This node is placed to the end that yields fewer crossings between the open edges and the edges that become closed by placing v, where an open edge refers to an edge which has one endpoint unplaced. Crossings with closed edges not incident to the currently inserted vertex do not need to be considered because they are the same for both sides. In Figure 1, there are eight such crossings for the left end and only five for the right end. The running time of this phase of the algorithm is $O((n + m) \log n)$.

2.1.2 Sifting

Each vertex is moved along the circle and the number of crossings is computed, while the other vertices stay in their current order. The vertex is then placed in its (locally) optimal position. The running time for one round is $O(nm)$. We use BB to denote the greedy phase, and BB+ to denote the combination of the two phases.

2.2 Algorithm AVSDF+

Algorithm AVSDF+ [13] consists of two phases too: greedy and adjusting.

2.2.1 Greedy

AVSDF is a variation of CIRCULAR algorithm [4], but it is implemented in a depth-first manner. First, one places the vertex with the smallest degree, and then visits the adjacencies of the current vertex, which have not been visited yet, such that the smallest degree vertex has the highest priority for visiting. The vertices are placed on the circle in the order they are visited. AVSDF runs in time $O(m)$.

2.2.2 Adjusting

In each round of the adjusting phase, the vertices are sorted in non-increasing order of the number of crossings created by their incident edges. The vertices are then considered, in this order, and for each vertex the best position among the current one and the ones immediately after each of its adjacent vertices is found and the vertex is moved to that position. The round is repeated until no improvements are obtained.

For each vertex v, only $deg(v)$ positions are considered during a round of the adjusting phase, while sifting considers all n positions. So, for the n vertices, the adjusting algorithm tries different positions $\sum_{v \in V} deg(v) = 2m$ times compared to n^2 times by sifting. Since the vertices causing larger number of crossings have higher priority to be adjusted, the crossing number is reduced quickly. In the experiments it was observed that the number of iterations is much lower than the vertex number n. Analogically to BB, we denote the basic phase as AVSDF, and the one augmented with adjusting phase as AVSDF+.

3 Two-page drawing algorithms

The goal of one-page algorithms is to *order* the vertices of a given graph to minimise the one-page book crossing number. Two-page algorithms have the additional task to *divide* the edges into two pages. Our new two-page algorithms start with an order produced by a one-page algorithm, and divide the edges into two pages with small crossings as possible.

3.1 Slope algorithm (SLOPE)

In a circular drawing, each edge has a slope to the horizontal line. SLOPE divides the edges into the pages according to their slopes. If an edge is horizontal or has a negative slope (the angle between the edge and the horizontal axis is larger than 90°), the edge is put on page 2 (solid edges in Figure 2). Otherwise, the edge is put on page 1 (dashed edges in Figure 2). Note that SLOPE is not invariant to rotations of the drawing.

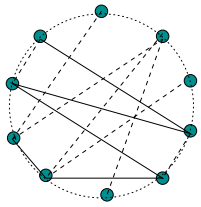


Figure 2: SLOPE divides the edges based on their slopes.

The vertex positions are numbered clockwise around the circle, with 0 being at the top. If an edge has the vertices in positions i and j, it is not hard to see that it is horizontal, if $(i + j) \bmod n = 0$, and vertical, if $(i + j) \bmod n = n/2$. An edge has a negative slope if $0 < (i+j) \bmod n < n/2$, and a positive slope, if $(i + j) \bmod n > n/2$. Hence, SLOPE puts an edge in page 2, if $(i + j) \bmod n < n/2$, and in page 1, if $(i+j) \bmod n \geq n/2$. SLOPE computes the matrix d_adj in time $O(n^2)$.

3.2 Descending length order (LEN)

The length of an edge between positions i and j in a circular drawing is defined as $\min\{|i - j|, n - |i - j|\}$. Intuitively, the longer the edge, the larger the probability to cross with other edges. Edges of length one create no crossings. Therefore, we sort the edges of length two or more in non-increasing order by length, where ties are broken lexicographically. Initially all edges are placed in page 1, and then each edge is either left in page one or moved to page two depending on the number of edge crossings produced. The process is iterated until either no improvement happens or five iterations are carried out (the number of iterations was found experimentally). LEN is similar to the algorithm E-len by Cimikowski [16]. The difference is that LEN starts with all edges in page one and moves some of the edges to page two, while E-len adds edges one by one to one of the two pages. The running time for one round is $O(m)$ excluding the calculation of crossings.

3.3 Descending crossing order (CRS)

The algorithm of edge allocation according to descending number of crossings is based on an initial fixed order of

edges in page one. CRS sorts the edges in non-increasing order of the number of crossings they produce, then it finds the first edge e in the list which would create fewer crossings on the other page, and moves it. CRS recalculates the number of crossings for all other edges, and re-sorts the list of edges. Note that the new number of crossings of any edge except e will differ by at most 1 from the old one. So, re-sorting the list can be done in linear time. Hence, the time complexity of a round is $O(m)$ excluding the computation of the number of crossings. CRS repeats this process (starting again at the top of the list) until no improvements are possible or the iteration number arrives m. Our preliminary experiments show that the number of iterations is much smaller than m.

CRS maintains two lists of m elements, *eplist* and *crlist*. Each element $eplist(i)$ stores the positions of the two endpoints of an edge, $eplist(i).u$ and $eplist(i).v$, and the page allocation $eplist(i).page$ of the edge. In the list *crlist* each element consists of two components, $crlist(i).post$ and $crlist(i).cr$, which are the index number of an edge in the *eplist* and the crossing number created by the edge, respectively. CRS is presented as Algorithm 1.

```
Create an edge list with page attribute, eplist;
Create a sorted crossing list with edge index, crlist;
Calculate current crossings, currCr;
i=0; lastCr=currCr;
while (i < m or lastCr > currCr) do
  lastCr = currCr;
  s = crlist(i).post; u = eplist(s).u;
  v = eplist(s).v;
  if (crlist(i).cr=0) then
    i=i+1;
  else
    eplist(s).page = eplist(s).page mod 2+1;
    cr = calEdgeCr(eplist, i);
    if (cr ≥ crlist(i).cr) then
      eplist(s).page = eplist(s).page mod 2+1;
      i = i+1;
    else
      currCr = currCr − crlist(i).cr + cr;
      crlist(i).cr = cr;
      modilist(eplist,crlist,i);
      i = 0;
    end if
  end if
end while
return currCr;
```

Algorithm 1: CRS

3.4 The neural network algorithm (NN)

A Hopfield network is a fully connected recurrent single layer and unsupervised network. Cimikowski and Shope [25] used a Hopfield neural network to model a two-page drawing of a graph with n vertices and m edges. In the model, each edge is associated with an "up" and a "down" neuron, representing an embedding of the edge in the upper or lower half-plane. Hence, the neurons Vup_{ij} and $Vdown_{ij}$ represent the upper and lower connections between vertices i and j, respectively. Therefore, the number of neurons is $2m$.

In the improved neural network [21], we used a model of binary neurons [26], each of which associated to an edge. The fired neurons indicate that the edges are in the upper side of the spine, and the unfired neurons indicate that the edges are in the down side of the spine. Hence, the model uses only m neurons.

A function $\Delta(x, y, z, w)$ to express the relationship of two edges in one page is defined as:

$$\Delta(x, y, z, w) = \begin{cases} 1, & if\ (x < z < y < w) \vee (z < x < w < y), \\ 0, & otherwise. \end{cases} \tag{3}$$

The energy function is defined as

$$E = A \sum_{i=0}^{m-1} \sum_{j=0, j \neq i}^{m-1} \Delta(x, y, z, w)(v_i v_j + \bar{v}_i \bar{v}_j). \tag{4}$$

The motion function is written as

$$\Delta u_i = -A \sum_{j=0, j \neq i}^{m-1} \Delta(x, y, z, w)(2v_j - 1). \tag{5}$$

We suppose that an order of vertices $\pi = 0, 1, \ldots, n-1$, has been obtained by a heuristic one-page algorithm. Moreover, we suppose that an ordered edge list $e_0, e_1, \ldots, e_{m-1}$ is created. At the startup, the network is randomly given an initial input $u_i \in (-1, 1)$. Correspondingly, the neuron states of the network are randomly initialized with 0 or 1.

According to the value of the motion Equation (5) in each iteration, the state of each neuron is updated, and consequently, the edge corresponding to the updated neuron is moved between upper and lower half planes, i.e., between page one and two. When the neural network arrives a stable state, the two-page book crossing number is calculated. The learning procedure is repeated with different initial states of the neurons for several times, and the best embedding is output. This algorithm is much faster than the Cimikowski and Shope algorithm. According to the motion function of the new model, the calculation in

each neuron takes time $O(m)$ in the worst case. Therefore, a sequential traversal of all m neurons takes time $O(m^2)$. Full details of the heuristic are given in [21].

3.5 Hybrid of one-page algorithm and edge pre-allocation (AVSDF_EP)

In addition to the methods based on a fixed linear vertex order or on a one-page drawing of the graph, we can synthetically consider vertex order and edge division directly for a two-page drawing. A hybrid heuristic algorithm AVSDF_EP is based on AVSDF. At edge division phase, it places an edge incident to the currently placed vertex based on the smallest crossings produced by the edge and already placed edges (see Algorithm 2). The time complexity is $O(m)$ excluding the computation of the number of crossings.

4 Experiments

Our main idea is to combine the order of vertices fixed by a one-page algorithm and edge division between pages. We also examine the effect of fixed order of vertices. After that we test different combinations of two-page algorithms. In Section 4.2 we investigate how edge density affect the performance of our heuristics. Finally, in Section 4.3 we test our heuristics on some typical graph families.

The experiments were done for a variety of graphs, which included two subsets in the benchmark test suites Rome Graphs (GDToolKit) [24]:

- RND BUP: A set of biconnected, undirected and planar graphs generated randomly, of which, 200 graphs were tested.
- ALF CU: A set of connected and undirected graphs, of which, 380 graphs were tested.

The experiments were also carried out for some sets of graphs in the graph library we created, including Random Connected Graphs (RCGs), regular degree 3 random connected graphs (3-d RCGs), and some typical graphs, such as Circulant graphs ($C_n(1, a_1, ..., a_k)$, $C_{kn}(1, n)$), Halin graphs, mesh graphs ($P_m \times P_n$), Cartesian production graphs ($C_m \times C_n$), complete p-partite graphs ($K_n(p)$), and complete graphs (K_n).

In the experiments of two page algorithms based on the optimal vertex order (e.g., a Hamiltonian cycle) or on the vertex order obtained with a one-page algorithm, the edges of the tested graphs were initially set in one page for SLOPE, LEN and CRS, while they are randomly assigned in two pages for NN.

The experiments were run on a laptop with dual-core CPU 3.0GHZ and 2GB memory.

4.1 Tests of two page algorithms

4.1.1 Tests based on one-page algorithms

We compared pairs of algorithm combinations formed by two heuristics for one-page drawings (AVSDF+ and BB+) and four two-page algorithms (SLOPE, LEN, CRS and NN), and AVSDF_EP as well. The experiments were run on RND_BUP, ALF_CU, RCGs, 3-d RCGs, $C_m \times C_n$, $P_m \times P_n$, Halin, $C_n(1, a_1, ..., a_k)$, and $C_{kn}(1, n)$ graphs.

Table 1 lists the numbers of times when each algorithm combination got the best results in all test data, with the highest values shown in bold face. The number in parentheses in the first column represents the number of graphs of that type used in the experiments. No matter which one-page algorithm is used to find the optimal order of vertices for all types of tested graphs, NN dominates the performance in all two-page algorithms, and BB+_NN performs better than AVSDF+_NN for all graphs except on circulant graphs $C_n(1, a_1 \dots a_k)$, and Cartesian products, for which, BB+_NN ties AVSDF+_NN. We use $X \prec Y$ to express that algorithm Y obtains the best results more times than algorithm X. From Table 1, for most types of graphs there exists the relationship: SLOPE $\prec$ LEN $\prec$ CRS $\prec$ NN when they are combined with AVSDF+ or BB+, while AVSDF_EP is slightly better than AVSDF+_SLOPE and BB+_SLOPE, but worse than AVSDF+_LEN and BB+_LEN. The last row of Table 1 shows the sum of numbers of times when an algorithm obtained the best results for all types of tested graphs. Therefore, we have the integrated performance of all algorithms on all types of tested graphs: BB+_SLOPE$\prec$ AVSDF+_SLOPE$\prec$ AVSDF_EP$\prec$ AVSDF+_LEN$\prec$ BB+_LEN$\prec$ AVSDF+_CRS$\prec$ BB+_CRS$\prec$ AVSDF+_NN$\prec$ BB+_NN.

4.1.2 Tests based on a fixed order of vertices

The experiments of Cimikowski [16] were done based on a fixed Hamiltonian cycle for some typical structural graphs. However, not every Hamiltonian cycle corresponds to an optimal vertex order for a two-page drawing, and an optimal two-page drawing might not correspond to a Hamiltonian cycle. Moreover, for an arbitrary graph, a Hamilto-

```
Create an adjacency list, each vertex with a linked list sorted in descending degree order
Define a list L, and a stack S;
Get the vertex with the smallest degree from the given graph, and push it into S;
while (S is not empty) do
    Pop a vertex v from S;
    if (v is not in L) then
        Append the vertex v into L;
        for (each vertex u adjacent to v, in decreasing order of deg(u)) do
            if (u is not in L) then
                push u into S;
            else
                place the edge (u, v) to the page where fewer new crossings are created by adding the edge;
            end if
        end for
    end if
end while
```

Algorithm 2: AVSDF_EP

Table 1: The numbers of times when best results were obtained on each type of graphs with the hybrid algorithm and all two-page algorithms based on one-page algorithms avsdf+ and bb+.

Graphs (number)	AVSDF+ _SLOPE	AVSDF+ _LEN	AVSDF+ _CRS	AVSDF+ _NN	AVSDF _EP	BB+ _SLOPE	BB+ _LEN	BB+ _CRS	BB+ _NN
ALF_CU(380)	47	113	124	189	87	47	132	147	**247**
RND_BUP(200)	45	82	88	91	57	46	90	91	**93**
RCGs(360)	36	62	80	142	46	17	85	102	**179**
3-d RCGs(46)	2	6	8	22	1	1	7	8	**30**
$C_m \times C_n$(49)	1	8	13	**28**	0	2	10	9	**28**
$P_m \times P_n$(49)	0	19	19	19	1	2	15	22	**40**
Halin(400)	12	139	229	284	26	7	91	173	**300**
$C_n(1, a_1 \ldots a_k)$(34)	0	7	7	**19**	3	0	5	13	16
$C_{kn}(1, n)$(68)	17	20	18	32	4	7	31	26	**40**
Total:	160	456	586	826	225	129	466	591	**973**

nian cycle might not exist, or even if it exists, we might not be able to find it efficiently.

We tested some structural graphs, such as $C_n(a_1, .., a_k)$, $C_{kn}(1, n)$, $C_m \times C_n$, $P_m \times P_n$, $K_n(p)$, and K_n, with the method used in [16] – first find an optimal order of vertices for a one-page drawing, and then apply an algorithm for the edge division. Circular graphs are regular Hamiltonian graphs. Therefore, the vertices of circulant graphs were placed along the node line in Hamiltonian order for the testing. The vertices of Cartesian graphs $C_m \times C_n$ were placed in Hamiltonian cycles or paths. For 3-row and 4-row mesh graphs, vertices were placed in the optimal orders as described in [27], but for other meshes, the natural orders in rows were kept as they were produced. For complete p-partite graphs, the vertices were placed in the optimal order [28] (see Section 4.3.3).

Table 2 shows the numbers of times when each two-page algorithm achieves the best results on each type of graphs in all tests. It can be seen that NN achieves the best results for all types of structural graphs. It is also shown that certain algorithms have different performance for different structural graphs. For example, SLOPE is the best for complete graphs, but for circulants $C_n(a_1, \ldots, a_k)$, Cartesian graphs $C_m \times C_n$ and mesh graphs $P_m \times P_n$, SLOPE performs worst. LEN has the worst performance for complete graphs K_n and complete p-partite graphs $K_n(p)$. This might be because the edges of a complete graph or a complete p-partite graph have the same lengths, and thus the lengths of edges do not make sense for edge division.

4.1.3 Tests based on different two-page preprocessings

We investigated the effect of using AVSDF_EP and AVSDF+_SLOPE as a two-page preprocessing steps, followed by LEN, CRS or NN. Figure 3 (a) and (b) show the two-page book crossing numbers of 60 different graphs with vertex numbers 80, 90, 100 in RND_BUP of Roman Graphs by using the algorithm combinations above.

As shown in Figure 3 (a) and (b), CRS achieved greater improvement to the results of the preprocessing algorithms AVSDF_EP and AVSDF+_SLOPE than LEN and NN. It can be seen that NN did not improve any results of AVSDF_EP, and occasionally improved results of AVSDF+_SLOPE. This might be because in the experiments the initial states of neurons correspond to the initial edge division obtained by the two preprocessing algorithms, respectively, and the initial two-page drawings obtained could be local optimums for NN. Thus, NN cannot improve the drawings any further. Note that for the experiments of the combinations of NN and one-page algorithms, the initial states of neurons were set randomly, which left a large search space for NN. Figure 3 (c) shows the results of the two best combinations AVSDF_EP_CRS and AVSDF+_SLOPE_CRS. Obviously, the later is better than the former.

We also observed the effect of different first stage algorithms on a two-page algorithm. The experiments were done on the same graph set as in the experiments above. From Figure 4 (a),(b) and (c), we can conclude that BB+ is the best preprocessing algorithm for LEN, CRS and NN in the experiment on RND_BUP of Roman Graphs. Figure 4 (d) shows that BB+_NN achieved the best results most times in the three best combinations BB+_LEN, BB+_CRS and BB+_NN.

4.2 Test on RCGs with different edge densities

We compared the eight combinations of one-page algorithms (AVSDF+, BB+) and two-page algorithms (SLOPE, LEN, CRS, NN) on RCGs with edge densities 1%, 2%, and 5%, where the density is defined as the ratio of the number of edges to the maximum possible number of edges. The maximal possible number of edges of a graph with n vertices is $n(n-1)/2$, thus the density can be written in the form of

$$d = \frac{2|E|}{n(n-1)}. \tag{6}$$

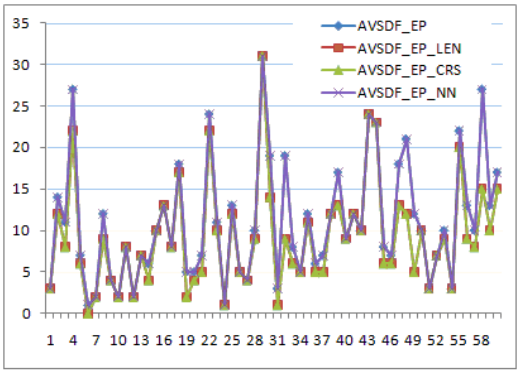
Trees are the sparsest connected graphs with $n-1$ edges, given a specific vertex number n. Therefore, a connected graph with edge density 1% has at least 200 vertices, with edge density 2% has at least 100 vertices, and with edge density 5% has at least 40 vertices. To create a RCG, we first created a random tree, and then add rest edges between any-pair of vertices picked randomly.

Twelve groups of graphs with vertex numbers from 200, 100, 40 were created for densities 1%, 2% and 5%, respectively. For the twelve groups of graphs, vertex numbers are changed with step 5, and every group includes 10 different graphs, for which the average crossing number is calculated.

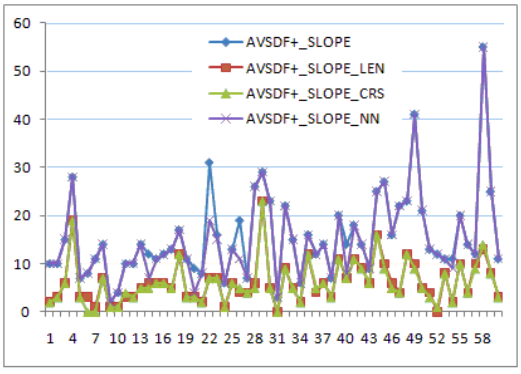
Tables 3, 4 and 5 show that BB+_NN achieved the dominating results for all densities of graphs, the second best combination was AVSDF+_NN for the RCGs. In the three tables, the best results are shown in bold face. AVSDF+_NN achieved the best results for the density 1% of graphs with lower vertex numbers, while BB+_NN achieved the best results for the density 1% of graphs with larger vertex num-

Table 2: The numbers of times when best results were obtained for each two-page algorithm based on a fixed order for some structural graphs.

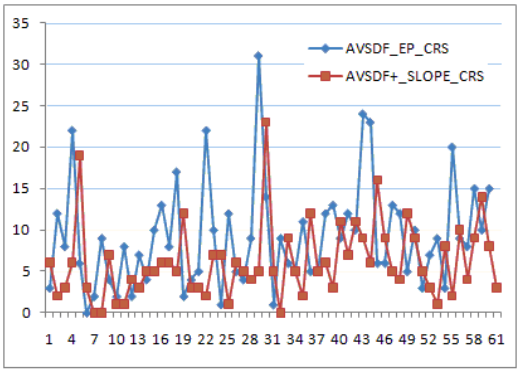
Graphs	SLOPE	LEN	CRS	NN
$C_n(a_1, \ldots, a_k)(34)$ [16]	0	14	14	**30**
$C_{kn}(1, n)(68)$	34	45	21	**68**
$K_n(p)(28)$	23	11	16	**27**
$K_n(26)$	**26**	17	22	**26**
$C_m \times C_n(49)$	9	19	34	**49**
$P_m \times P_n$ (49)	1	14	17	**48**



a) AVSEDF_EP as a preprocessing



b) AVSDF+_SLOPE as a preprocessing



c) Two best combinations

Figure 3: A two-page preprocessing following with different two-page algorithms.

bers. For densities 2% and 5% of graphs, BB+_NN achieved the best results most times in all combinations.

We also observed the running times of one-page algorithms (AVSDF+ and BB+) and two-page algorithms (SLOPE, LEN, CRS, NN) on RCGs with densities 1%, 2% and 5%. The running time of a two-page algorithm was recorded when the combination of the two-page algorithm and AVSDF+ was applied. The running time of a two-page algorithm based on BB+ was similar to that when it was based on AVSDF+, although they were different due to different orders obtained by different one-page algorithms. Tables 6, 7 and 8 list the running times (ms) of one-page algorithms and two-page algorithms for the three densities of graphs RCGs.

From Tables 6, 7 and 8, it can be seen that the one-page algorithm BB+ took much longer time than AVSDF+. The running times of SLOPE and CRS were basically less than 10 ms. For all densities of graphs, LEN took longer time than NN when vertex number went larger for all densities of graphs. This indicates that the running time of LEN increased as the rise of vertex number faster than that of NN. However, the running time of BB+ was much longer than that of all two-page algorithms. Therefore, we can conclude that BB+_NN achieved dominating results with high time cost due to BB+'s time complexity. Hence, more efficient one-page algorithms are needed.

4.3 Tests on some typical graphs

We tested some circulant graphs used by Cimikowski [16] and some Halin graphs with two-page algorithms based on one-page algorithms. For some typical classes of graphs such as complete p-partite graphs, 3-row meshes, 4-row meshes and Cartesian graphs, the optimal one-page drawings were first constructed, and then a two-page algorithm was applied.

Table 3: Two-page book crossing numbers obtained by algorithm combinations on the graphs with density of 1%, where V_n is the number of vertices.

V_n	AVSDF+ _SLOPE	AVSDF+ _LEN	AVSDF+ _CRS	AVSDF+ _NN	BB+ _SLOPE	BB+ _LEN	BB+ _CRS	BB+ _NN
200	**0**	**0**	**0**	**0**	3	**0**	**0**	**0**
205	5	**1**	**1**	**1**	12	2	2	2
210	14	7	6	**5**	26	8	9	8
215	26	16	15	**14**	45	22	22	19
220	48	36	33	**29**	69	36	35	30
225	77	58	58	**53**	103	59	58	57
230	109	91	88	**82**	141	**82**	83	84
235	137	110	110	106	185	117	115	**103**
240	191	155	147	145	222	147	**141**	**141**
245	223	200	194	191	286	203	194	**176**
250	289	253	246	241	358	250	236	**228**
255	336	302	295	289	392	278	**270**	276

Table 4: Two-page book crossing numbers obtained by algorithm combinations on the graphs with density of 2%, where V_n is the number of vertices.

V_n	AVSDF+ _SLOPE	AVSDF+ _LEN	AVSDF+ _CRS	AVSDF+ _NN	BB+ _SLOPE	BB+ _LEN	BB+ _CRS	BB+ _NN
100	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
105	2	**1**	**1**	**1**	6	**1**	**1**	**1**
110	9	6	5	**4**	14	5	**4**	**4**
115	25	17	19	16	34	15	16	**14**
120	41	29	26	24	53	31	28	**22**
125	78	60	58	**54**	93	60	57	55
130	95	77	78	74	115	75	**71**	75
135	139	113	116	107	158	**106**	**106**	110
140	195	161	157	155	229	158	159	**147**
145	272	226	220	217	288	217	208	**197**
150	349	310	307	304	374	285	282	**262**
155	427	388	369	365	463	373	359	**342**

Table 5: Two-page book crossing numbers obtained by algorithm combinations on the graphs with density of 5%, where V_n is the number of vertices.

V_n	AVSDF+ _SLOPE	AVSDF+ _LEN	AVSDF+ _CRS	AVSDF+ _NN	BB+ _SLOPE	BB+ _LEN	BB+ _CRS	BB+ _NN
40	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
45	2	**0**	**0**	**0**	2	**0**	**0**	**0**
50	10	5	6	5	12	6	5	**4**
55	22	17	17	**14**	26	**14**	15	15
60	44	39	33	**30**	51	35	32	**30**
65	89	68	65	64	95	68	66	**56**
70	140	118	115	111	155	113	**109**	**109**
75	221	187	182	177	223	172	173	**160**
80	328	274	273	265	329	265	264	**256**
85	429	363	355	**347**	463	376	371	349
90	605	522	517	**502**	620	536	525	504
95	828	711	699	690	826	696	689	**672**

Table 6: Running times of one-page algorithms and two-page algorithms based on AVSDF+ for density of 1%, where V_n is the number of vertices, and time unit is ms.

V_n	AVSDF+	BB+	SLOPE	LEN	CRS	NN
200	703	20759	1	46	3	155
205	822	26434	1	151	0	195
210	1065	34922	1	238	3	223
215	1661	42273	0	315	1	268
220	2296	57579	3	444	3	332
225	3202	87395	0	639	3	383
230	3681	87037	1	937	6	439
235	4898	104079	1	1028	1	480
240	5572	138003	1	1443	3	528
245	6882	158404	4	1421	4	620
250	9172	191772	1	1928	4	647
255	11247	211673	7	2092	6	733

Table 7: Running times of one-page algorithms and two-page algorithms based on AVSDF+ for density of 2%, where V_n is the number of vertices, and time unit is ms.

V_n	AVSDF+	BB+	SLOPE	LEN	CRS	NN
100	132	1120	1	4	0	43
105	118	1530	0	15	0	45
110	182	2377	0	28	0	59
115	321	3672	0	48	0	73
120	426	4835	0	70	0	90
125	648	6960	1	109	0	118
130	974	9099	0	140	0	143
135	1158	11555	0	181	3	177
140	1427	16086	3	310	1	229
145	2213	20025	1	430	3	277
150	3004	27459	0	601	4	316
155	3117	33719	3	731	1	380

Table 8: Running times of one-page algorithms and two-page algorithms based on AVSDF+ for density of 5%, where V_n is the number of vertices, and time unit is *ms*.

V_n	AVSDF+	BB+	SLOPE	LEN	CRS	NN
40	15	28	0	0	0	12
45	11	51	0	1	0	11
50	23	98	0	1	0	14
55	53	177	0	4	0	22
60	96	333	0	14	0	34
65	135	575	0	23	1	54
70	249	943	0	46	0	72
75	474	1346	0	84	0	106
80	722	2160	1	146	1	143
85	1092	3232	0	224	4	208
90	1480	4631	1	349	1	262
95	2262	6506	0	575	12	365

4.3.1 Circulant graph test

Circulant graphs of the form $C_n(a_1, a_2, \ldots, a_k)$, where $0 < a_1 < a_2 < \ldots < a_k < (n+1)/2$, are regular Hamiltonian graphs with n vertices, and with vertices $i \pm a_j (\bmod n)$, $j = 1, \ldots, k$, adjacent to each i [30].

Some circulant graphs had been tested based on a fixed Hamiltonian cycle by Cimikowski [16]. We tested these circulant graphs as well, but first applied a heuristic algorithm to find a good one-page drawing, and then applied a two-page algorithm.

In Table 9, the middle eight columns list the crossing numbers of all tested circulants obtained with eight algorithm combinations. The third rightmost column contains either the optimal values related to the fixed order of vertices based on Hamiltonian cycles [16], or theoretical lower and upper bounds from [16] (written as $a : b$ with a the lower and b the upper bound), if the branch-and-bound algorithm of Cimikowski [16] was not applicable. The second rightmost column contains the best results obtained with Cimikowski's eight heuristic algorithms based on the Hamiltonian cycle of each circulant graph [16]. The rightmost column lists the best results obtained with genetic algorithms that we previously developed [20].

In Table 9, the best results of our eight heuristic algorithms for all circulant graphs are shown in bold face. Almost all best results are the same as or better than the best results obtained with Cimikowski's eight heuristic algorithms [16]. For some circulant graphs such as $C_{24}(1, 3, 5)$, $C_{26}(1, 3)$, $C_{38}(1, 7)$, $C_{40}(1, 5)$, $C_{42}(1, 4)$, and $C_{46}(1, 4)$, the best results are better than the optimal values that can be obtained from branch-and-bound algorithm based on a fixed order of vertices [16]. This is because our algorithms, unlike [16], are not restricted to the fixed order. Although the fixed Hamiltonian order is usually viewed as a favorable order for two-page drawings [16], it restricts the search space. For example, Figure 5 presents the best known two-page drawing for $C_{42}(1, 4)$ (which we found using AVSDF+_NN). It has 38 crossings on a fixed Hamiltonian order of vertices, while the optimal solution based on the same order had 42 crossings [16]. However, the best two-page crossing number for $C_{42}(1, 4)$ is 23 [19].

However, comparing with the best results achieved with the genetic algorithms, which are listed in the rightmost column in Table 9, the best results of our eight heuristic algorithm combinations are worse. This might be because genetic algorithms search a larger space of vertex order and edge division synchronously.

4.3.2 Halin graph tests

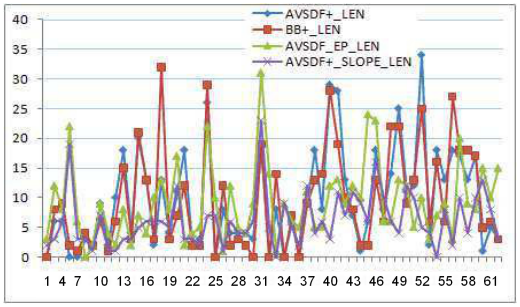
A Halin graph H is a planar graph $H = T \cup C$, where T is a tree with no vertex of degree two and at least one vertex of degree three or more. T is embedded in the plane and C is a cycle connecting the leaves of T in the cyclic order determined by the embedding of T. The edges in T will be called t-edges and the ones in C will be called c-edges.

The two-page book crossing number of a subhamiltonian planar graph is zero [31, 32]. A Halin graph is a subhamiltonian planar graph, and thus the two-page book crossing number of Halin graphs is zero [27]. We have presented an algorithm to obtain planar two-page drawing of a Halin graph based on an optimal Hamiltonian order [27].

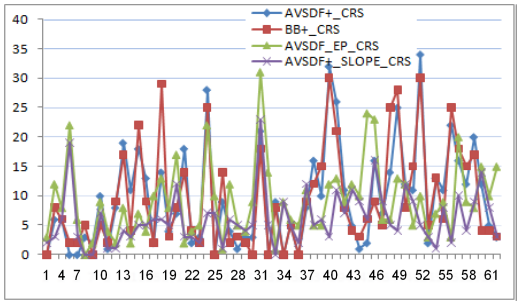
We tested 400 Halin graphs with the hybrid algorithm AVSDF_EP and the four two-page algorithms based

Table 9: Test results for circulants obtained with the eight algorithm combinations, ranges or optimal values on fixed order denoted as Opt._C [16], best results obtained with Cimikowski's eight heuristic algorithms denoted as best_C [16], and the best results obtained with genetic algorithms denoted as best_GA [20].

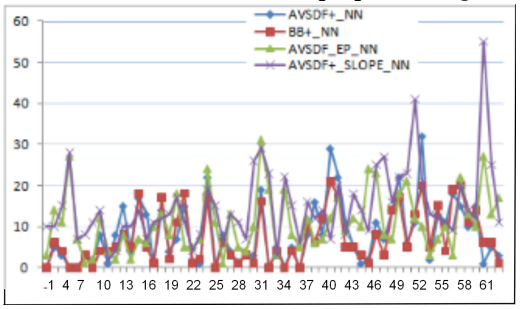
Graphs	AVSDF+ _SLOPE	AVSDF+ _LEN	AVSDF+ _CRS	AVSDF+ _NN	BB+ _SLOPE	BB+ _LEN	BB+ _CRS	BB+ _NN	Opt. _C	best _C	best _GA
$C_{20}(1,2)$	16	**0**	**0**	**0**	16	**0**	**0**	**0**	0	0	0
$C_{20}(1,2,3)$	72	**22**	24	**22**	72	**22**	24	**22**	22	22	18
$C_{20}(1,2,3,4)$	180	84	**70**	**70**	180	84	**70**	72	26:870	70	68
$C_{22}(1,2)$	18	**0**	**0**	2	18	**0**	**0**	**0**	0	0	0
$C_{22}(1,2,3)$	82	**24**	**24**	**24**	82	**24**	**24**	**24**	24	24	20
$C_{22}(1,3,5,7)$	266	216	203	**201**	238	231	225	227	28:1056	200	166
$C_{24}(1,3)$	34	14	15	**12**	14	14	14	15	12	12	9
$C_{24}(1,3,5)$	142	88	73	73	140	86	**70**	73	72	76	60
$C_{24}(1,3,5,7)$	318	246	217	**216**	309	280	237	223	30:1260	216	193
$C_{26}(1,3)$	38	16	17	14	36	18	19	**12**	14	14	10
$C_{26}(1,3,5)$	162	96	83	82	160	94	**80**	83	6:650	82	63
$C_{26}(1,4,7,9)$	382	320	318	318	323	339	**311**	336	32:1482	364	290
$C_{28}(1,3)$	42	18	19	**14**	22	19	19	19	14	16	11
$C_{28}(1,3,5)$	182	104	87	**86**	188	92	88	**86**	6:756	86	75
$C_{28}(1,2,3,4)$	292	126	**98**	100	292	126	**98**	**98**	34:1722	98	97
$C_{28}(1,3,5,7,9)$	754	**561**	**561**	**561**	751	602	586	633	62:3080	560	508
$C_{30}(1,3,5)$	202	106	97	**90**	208	107	96	**90**	6:870	96	83
$C_{30}(1,3,5,8)$	466	319	324	**296**	466	319	324	**296**	36:1980	302	226
$C_{30}(1,2,4,5,7)$	728	410	**392**	394	711	434	411	**392**	66:3540	392	346
$C_{32}(1,2,4,6)$	408	160	192	160	206	**128**	126	160	38:2256	160	124
$C_{34}(1,3,5)$	242	128	105	**104**	251	116	110	107	6:1122	106	96
$C_{34}(1,4,8,12)$	648	561	620	561	371	309	302	**295**	40:2550	574	286
$C_{36}(1,2,4)$	172	**36**	54	**36**	60	**36**	**36**	**36**	6:1260	36	36
$C_{36}(1,3,5,7)$	654	**329**	331	**329**	657	346	336	**329**	42:2862	328	301
$C_{38}(1,7)$	114	76	77	64	82	58	**53**	75	84	86	36
$C_{38}(1,4,7)$	364	221	187	187	337	190	**176**	187	6:1406	190	149
$C_{40}(1,5)$	108	58	61	**48**	82	49	57	58	56	58	29
$C_{42}(1,4)$	96	42	40	**38**	90	48	51	40	42	42	24
$C_{42}(1,3,6)$	338	162	164	151	259	118	115	**111**	6:1722	158	106
$C_{42}(1,2,4,6)$	598	228	238	210	250	164	162	**158**	48:3906	210	168
$C_{44}(1,4,5)$	431	168	177	165	328	124	123	**107**	6:1892	180	99
$C_{44}(1,4,7,10)$	1094	639	629	**627**	820	631	634	631	50:4290	632	491
$C_{46}(1,4)$	108	46	44	**42**	72	63	63	44	46	46	29
$C_{46}(1,5,8)$	521	289	280	281	479	290	**267**	271	6:2070	296	246



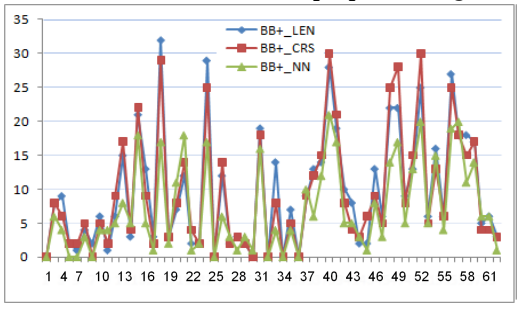
a) LEN based on different preprocessings



b) CRS based on different preprocessings



c) NN based on different preprocessings



d) Comparison of the best combinations

Figure 4: Effect of different preprocessings on two-page algorithms.

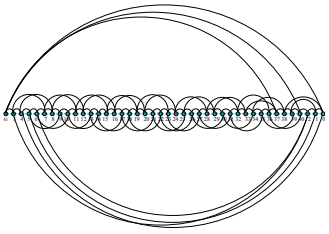


Figure 5: The best known solution for $C_{42}(1, 4)$.

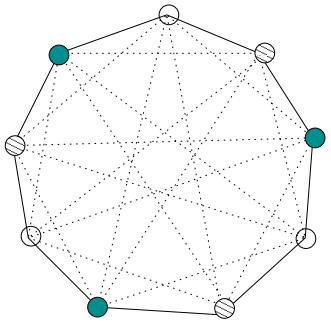


Figure 6: Optimal order of $K_3(3)$.

on one-page algorithms. We use $X \succ Y$ to express that algorithm X obtains 0 crossings more times than algorithm Y. We have: BB+_NN(278) $\succ$ AVSDF_NN(259) $\succ$ AVSDF+_CRS(213) $\succ$ BB+_CRS(162) $\succ$ AVSDF+_LEN(132) $\succ$ BB+_LEN(88) $\succ$ AVSDF_EP(26) $\succ$ AVSDF+_SLOPE(11) $\succ$ BB+_SLOPE(7), with the numbers in brackets denoting the number of times the algorithm obtained a planar drawing.

4.3.3 Complete p-partite graph and complete graph tests

We denote a complete p-partite graph with equal size (n) of the partite sets as

$$K_n(p) = K_{\underbrace{n, n, \ldots, n}_{p}}.$$

For a complete p-partite graph with n vertices in each partite set, the one-page book crossing number is $\nu_1(K_n(p)) = n^4\binom{p}{4} + \frac{1}{2}n^2(n-1)(2n-1)\binom{p}{3} + n\binom{n}{3}\binom{p}{2}$ [28]. We know the optimal one-page drawing solution for $K_n(p)$ [28]. All vertices of the partite sets are evenly placed around a cycle, i.e., the vertices of every partite set form a regular n-gon (Figure 6). It is easy to get the optimal order for a complete p-partite graph in linear time. The test was done based on the optimal orders of 28 complete p-partite graphs (see Table 10). NN got the best results for all complete p-partite graphs tested except for $K_3(4)$, for which SLOPE obtained the best result (86 two-page crossings). SLOPE also achieved the best results in most cases (23/28), but the results of LEN and CRS were not far behind.

For complete graphs, we directly applied the two-page algorithms on the natural order 0,1,...,n-1 (see Table 11). NN and SLOPE achieved optimal or conjectured optimal results every time, while the results of LEN and CRS were close to or the same as the conjectured optimal ones. The values for the complete graphs from K_5 to K_{13} were presented by Cimikowski [16] as well. Data in bold face match

Table 10: Test results on $K_n(p)$ by the four two-page algorithms based on optimal one-page drawings.

Graphs	Opt.(v_1)	SLOPE	LEN	CRS	NN
$K_3(2)$	3	1	1	1	1
$K_4(2)$	16	4	6	4	4
$K_5(2)$	50	16	16	16	16
$K_6(2)$	120	36	36	36	36
$K_7(2)$	245	81	81	81	81
$K_8(2)$	448	144	160	144	144
$K_9(2)$	756	256	256	256	256
$K_3(3)$	54	16	16	16	16
$K_4(3)$	216	68	80	68	66
$K_5(3)$	600	196	198	196	196
$K_6(3)$	1350	450	452	454	450
$K_7(3)$	2646	900	900	900	900
$K_8(3)$	4704	1616	1620	1616	1616
$K_9(3)$	7776	2704	2704	2704	2704
$K_3(4)$	279	86	90	87	87
$K_4(4)$	1024	336	338	344	336
$K_5(4)$	2725	916	916	916	916
$K_6(4)$	5976	2052	2056	2056	2052
$K_7(4)$	11515	4002	4002	4002	4002
$K_8(4)$	20224	7104	7110	7108	7104
$K_9(4)$	33129	11720	11721	11720	11720
$K_3(5)$	885	291	299	291	289
$K_4(5)$	3120	1056	1064	1064	1056
$K_5(5)$	8125	2813	2813	2813	2811
$K_6(5)$	17580	6156	6156	6156	6156
$K_7(5)$	33565	11887	11887	11887	11885
$K_8(5)$	58560	20864	20902	20868	20864
$K_9(5)$	95445	34233	34233	34233	34231

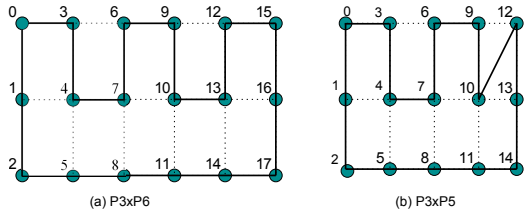


Figure 7: Optimal order for one-page drawing of 3-row meshes [28].

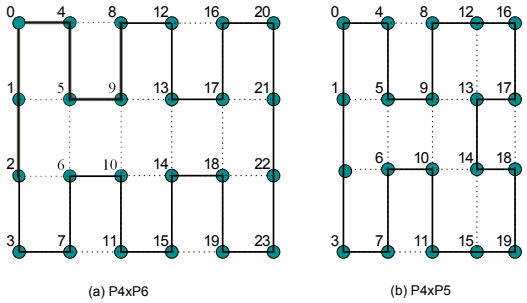


Figure 8: Optimal order for one-page drawing of 4-row meshes [27].

the (conjectured) optimal values, as they are the same as the upper bounds from Guy's conjecture for general planar crossing numbers of complete graphs [29].

We define an approximate degree η of a test result to the optimum value as

$$\eta = v_2/cr_2. \tag{7}$$

The closer to 1 the approximate degree η is, the closer to the optimal two-page crossing number is the result. We calculated the average approximate degree of each two-page algorithm for the tested complete graphs from K_4 to K_{29}. The approximate degrees of SLOPE, LEN, CRS and NN were 1, 0.985894787, 0.995485922 and 1, respectively.

Actually, we have tested complete graphs from K_4 to K_{200} with NN, and all the results of the tested complete graphs are the same as the upper bound of the general planar crossing numbers [21]. Here, we also tested K_4 to K_{200} with SLOPE, and the results are the same as those obtained by NN.

4.3.4 3- or 4-row mesh tests

For 3-row meshes, $P_3 \times P_n$, we know the optimal one-page book crossing number [28], for any odd $n \geq 3$: $v_1(P_3 \times P_n) = 2n - 3$, and for any even $n \geq 4$: $v_1(P_3 \times P_n) = 2n - 4$. For 4-row meshes with n columns, $P_4 \times P_n$, we have $v_1(P_3 \times P_n) = 4n - 8$ [27].

For a 3- or 4-row mesh, it is easy to construct a Hamiltonian cycle to get the optimal one-page drawing in linear time [27] (Figure 7 (a) (b) and Figure 8 (a)(b)). The one-page drawing of each mesh is obtained by placing the vertices on a circle in the order given by the Hamiltonian cycle drawn with solid lines.

Clearly, 3-row and 4-row meshes are subhamiltonian graphs, and thus the two-page book crossing numbers of all 3-row and 4-row meshes are zeros [27]. Exact solutions for two-page drawings of 4-row meshes were presented in [27]. We tested 3- or 4-row meshes with the four two-page algorithms based on these Hamiltonian cycles (paths). Table 12 shows the results. LEN, CRS and NN produced planar drawings for every 3-row or 4-row mesh tested.

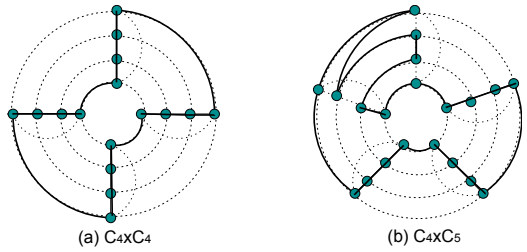


Figure 9: A drawing of $C_4 \times C_4$ and $C_4 \times C_5$.

4.3.5 Cartesian product $C_m \times C_n$ tests

There is a lot of research about Cartesian product graphs $C_m \times C_n$ [33–35]. A natural planar drawing of $C_m \times C_n$ having $(m-2)n$ crossings, where $m \leq n$, is as the following: draw $P_m \times C_n$ with no crossings (the cycles are taken to be concentric) and then to each path add one edge, crossing $m-2$ of the concentric cycles [33]. Figure 9 and Figure 10 present these drawings for $C_4 \times C_4$, $C_4 \times C_5$ and $C_5 \times C_5$.

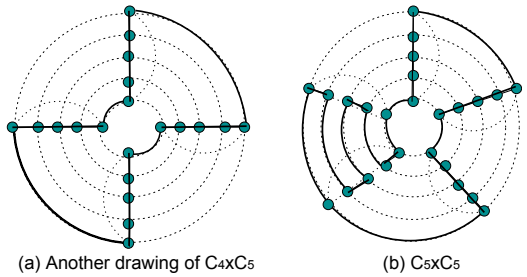


Figure 10: A drawing of $C_4 \times C_5$ and $C_5 \times C_5$.

Table 11: Test results on K_n by the four two-page algorithms.

Graphs	SLOPE	LEN	CRS	NN	Graphs	SLOPE	LEN	CRS	NN
K_4	**0**	**0**	**0**	**0**	K_{17}	**784**	786	**784**	**784**
K_5	**1**	**1**	**1**	**1**	K_{18}	**1008**	**1008**	1018	**1008**
K_6	**3**	**3**	**3**	**3**	K_{19}	**1296**	**1296**	**1296**	**1296**
K_7	**9**	**9**	**9**	**9**	K_{20}	**1620**	**1620**	**1620**	**1620**
K_8	**18**	20	19	**18**	K_{21}	**2025**	**2025**	**2025**	**2025**
K_9	**36**	38	**36**	**36**	K_{22}	**2475**	**2475**	**2475**	**2475**
K_{10}	**60**	65	62	**60**	K_{23}	**3025**	**3025**	**3025**	**3025**
K_{11}	**100**	**100**	**100**	**100**	K_{24}	**3630**	**3630**	**3630**	**3630**
K_{12}	**150**	155	154	**150**	K_{25}	**4356**	**4356**	**4356**	**4356**
K_{13}	**225**	227	**225**	**225**	K_{26}	**5148**	**5148**	**5148**	**5148**
K_{14}	**315**	**315**	**315**	**315**	K_{27}	**6084**	**6084**	**6084**	**6084**
K_{15}	**441**	473	**441**	**441**	K_{28}	**7098**	7126	**7098**	**7098**
K_{16}	**588**	606	**588**	**588**	K_{29}	**8281**	**8281**	**8281**	**8281**

Table 12: Two-page book crossing number of 3- or 4-row meshes by the four two-page algorithms based on (conjectured) optimal one-page drawings.

Graphs	SLOPE	LEN	CRS	NN
$P_3 \times P_4$	1	0	0	0
$P_3 \times P_5$	2	0	0	0
$P_3 \times P_6$	4	0	0	0
$P_3 \times P_7$	7	0	0	0
$P_3 \times P_8$	6	0	0	0
$P_3 \times P_9$	9	0	0	0
$P_4 \times P_4$	2	0	0	0
$P_4 \times P_5$	6	0	0	0
$P_4 \times P_6$	10	0	0	0
$P_4 \times P_7$	12	0	0	0
$P_4 \times P_8$	14	0	0	0
$P_4 \times P_9$	18	0	0	0

Table 13: Two-page book crossing number of $C_m \times C_n$ by the four two-page algorithms based on the fixed Hamiltonian cycles (paths); the data in bold face indicate the results are equal to the conjectured optimal values.

Graphs	SLOPE	LEN	CRS	nn
$C_3 \times C_3$	4	**3**	**3**	**3**
$C_3 \times C_4$	**4**	**4**	**4**	**4**
$C_3 \times C_5$	**5**	**5**	**5**	**5**
$C_3 \times C_6$	8	**6**	**6**	**6**
$C_3 \times C_7$	10	**7**	**7**	**7**
$C_3 \times C_8$	12	**8**	**8**	**8**
$C_3 \times C_9$	16	**9**	**9**	**9**
$C_4 \times C_4$	**8**	**8**	**8**	**8**
$C_4 \times C_5$	26	**10**	14	**10**
$C_4 \times C_6$	16	**12**	**12**	**12**
$C_4 \times C_7$	42	18	18	**14**
$C_4 \times C_8$	24	**16**	**16**	**16**
$C_4 \times C_9$	58	22	22	**18**
$C_5 \times C_5$	22	23	23	**15**
$C_5 \times C_6$	44	24	18	**18**
$C_5 \times C_7$	63	31	34	**21**
$C_5 \times C_8$	76	32	**24**	**24**
$C_5 \times C_9$	100	39	40	**27**
$C_6 \times C_6$	62	36	**24**	**24**
$C_6 \times C_7$	86	41	30	**28**
$C_6 \times C_8$	108	48	**32**	**32**
$C_6 \times C_9$	137	53	38	**36**
$C_7 \times C_7$	112	41	63	**35**
$C_7 \times C_8$	144	64	**40**	**40**
$C_7 \times C_9$	179	51	73	**45**
$C_8 \times C_8$	184	**48**	**48**	**48**
$C_8 \times C_9$	224	57	56	**54**
$C_9 \times C_9$	274	69	**63**	**63**

In Figure 9 and Figure 10, the solid edges form a Hamiltonian cycle or path. For a Cartesian graph $C_m \times C_n$ with $m \leq n$, when n is even, we can construct a Hamiltonian cycle as in Figure 9 (a). When n is odd and m is odd, we can construct a Hamiltonian cycle as in Figure 10 (b). When n is odd but m is even, we can construct the Hamiltonian cycle as in Figure 10 (a), but it is not optimal for our purpose of two-page drawings. In this case, we can also construct a Hamiltonian path as in Figure 9 (b). Hence, we tested Cartesian products $C_m \times C_n$ based on these Hamiltonian cycles (paths) of each graph. Table 13 shows the results, and the data in bold face indicate that the results are equal to the planar crossing numbers $(m-2)n$. It can be seen that NN achieved the best results, and all results are equal to the planar crossing numbers.

We know the exact crossing numbers or upper bounds of all tested graphs except the tested circulant graphs. From the experimental results, we can see that the best results obtained by our edge allocation algorithms show excellent performance in approximation to the planar crossing number, as they are always equal to either the exact crossing numbers (e.g., for Halin graphs, mesh graphs, complete p-partite graphs and Cartesian products), or the (conjectured) upper bounds of the tested graphs (e.g., complete graphs). This indicates that we can completely use the two-page crossing number to approximate the planar crossing number of a graph, and the optimal two-page drawing provides an easy approach to a planar drawing. Actually, according to the experimental results, we may have the following conjecture:

The optimal two page drawing of a graph represents the planar drawing of the graph.

5 Conclusion

We designed several heuristics for the two-page book crossing problem by combining two of one-page drawing algorithms (BB+[12] and our AVSDF+ [13]) with four two-page algorithms (SLOPE, LEN, CRS, and NN). We also proposed a hybrid algorithm (AVSDF_EP), which directly assign edges into pages while placing the vertices on the circle in the AVSDF algorithm.

We investigated the performance of our algorithms by testing them on benchmark test suites: Rome graphs (RND_BUP and ALF_CU), Random Connected Graphs (RCGs) and some typical graphs. For all types of graphs tested, BB+_NN achieved dominating results, but it took much time due to BB+'s time complexity. AVSDF+_NN is the second best combination for all graphs.

We investigated the effect of one-page algorithms on a two-page algorithm. The experiments show that BB+ is the best preprocessing algorithm for all two-page algorithms.

We also investigated the effect of using AVSDF_EP or AVSDF+_SLOPE as a preprocessing step, followed by the LEN, CRS, or NN. Starting with AVSDF_EP, LEN achieved much greater improvement to the results than CRS for RND_BUP graphs. In contrast, starting from AVSDF+_SLOPE, CRS achieved greater improvements than LEN on the same RND_BUP graphs. For LEN, the best preprocessing is AVSDF_EP, while for CRS, the best preprocessing is the AVSDF+. However, when starting with an initial two-page distribution provided by AVSDF_EP or AVSDF+_SLOPE, NN was in the shadow of other two-page algorithms.

For all densities of RCGs, BB+_NN achieved the best results for most cases in all algorithm combinations. We examined the running times of one-page and two-page algorithms. It is shown that BB+ took much longer time than AVSDF+ for all densities of RCGs. LEN took longer time than NN when the vertex number of a graph increased for all densities of RCGs.

Based on the optimal order of vertices for one-page drawing, we tested several typical structured graphs. NN almost achieved the optimal results for all tested structured graphs. For complete graphs and complete p-partite graphs, SLOPE obtained excellent results as well, especially for complete graphs, SLOPE achieved the same results as NN, and the results are equal to the planar crossing numbers for complete graphs. NN also achieved the planar crossing numbers for Cartesian products graphs. LEN, CRS and NN achieved planar drawings for all tested 3-row and 4-row mesh graphs.

With our two-page algorithms based on a one-page algorithm, the best results of some circulant graphs are better than the optimal values based on fixed orders of vertices obtained by Cimikowski [16]. A possible reason is that a fixed order of the vertices constrains the search space of solutions. However, the best results are worse than those obtained with genetic algorithms that we previously developed. It might be because genetic algorithms search a larger space of vertex order and edge distribution synchronously. In other words, the constraints of vertex order could lead to the limitation of finding a good two-page drawing.

The experimental results of these typical graphs, especially complete graphs, strongly support that two-page book crossing number presents an excellent upper bound for general crossing number of graphs, and the fact that the two-page book crossing numbers of 3-row and 4-row meshes and Halin graphs are zero further highlights this point, as these graphs are planar graphs. Further more, we may conjecture that the optimal two-page crossing number of a graph is the planar crossing number of the graph.

Acknowledgement: This work is based on the research when the first author was supported by the EPSRC grant GR/S76694/01 and by VEGA grant No. 2/3164/23 in the Department of Computer Science, Loughborough University.

References

[1] F. Shahrokhi, O. Sýkora, L.A. Székely, I. Vrt'o, The book crossing number of a graph, J. Graph Theory 21, 413–424, 1996

[2] P.C. Kainen, The book thickness of a graph II, Congressus Numerantium 71, 121–132, 1990

[3] F. Shahrokhi, O. Sýkora, L.A. Székely, I. Vrt'o, The gap between crossing numbers and the convex crossing numbers, Towards a Theory of Geometric Graphs, In: J. Pach (Ed.), Contemporary Mathematics, 342, American Mathematical Society, Providence, RI USA, 2004, 249–258

[4] J.M. Six, I.G. Tollis, Circular drawings of biconnected graphs, Proc. of ALENEX'99, LNCS 1619, 57–73, 1999 pp. 57-73.

[5] M. Masuda, T. Kashiwabara, K. Nakajima, T. Fujisawa, On the NP-completeness of a computer network layout problem, Proc. of IEEE Intl. Symposium on Circuits and Systems 1987, IEEE Computer Society Press, Los Alamitos, 1987, 292–295

[6] S. Masuda, T. Kashiwabara, K. Nakajima, T. Fujisawa, Crossing minimization in linear embeddings of graphs, IEEE Trans. Comput. 39, 124–127, 1990

[7] F.R.K. Chung, F.T. Leighton, A.L. Rosenberg, Diogenes: a methodology for designing fault-tolerant VLSI processor arrays, Proc. the 13th Annu. Symp. Fault-Tolerant Comput, June 1983 Milan, Italy, 26–32

[8] F.R.K. Chung, F.T. Leighton, A.L. Rosenberg, Embedding graphs in books: a layout problem with applications to VLSI design, SIAM J. Algebra. Discr. 8(1), 33-58, 1987

[9] W.J. Chung, B.S. Smith, S.K. Lim, QCA Physical Design With Crossing Minimization, Proc. IEEE Conference on Nanotechnology, 11-15 July 2005, Nagoya Congress Center Nagoya, Japan, 2005, 262–265

[10] B.S. Smith, S.K. Lim, QCA Channel Routing With Wire Crossing Minimization, Proc. of The 15th ACM Great Lakes symposium on VLSI, Chicago, Illinois, USA., 2005, 217–220

[11] E. Mäkinen, On circular layouts, Int. J. Comput. Math. 24, 29–37, 1988

[12] M. Baur, U. Brandes, Crossing Reduction in Circular Layouts, Proc. 30th Intl. Workshop Graph-Theoretic Concepts in Computer-Science (WG '04), LNCS 3353, 332–343, 2004

[13] H. He, O. Sýkora, New Circular Drawing Algorihtms, Proc. ITAT'04, 15-19 Sept. 2004, High Tatras, Slovakia, 2004

[14] A.N. Melikov, V.M. Koreičik, V.A. Tiščenko, Minimization of the number of intersections of edges of a graph (Russian), Vyčisl. Sistemy Vyp. 47, 32–40, 1971

[15] T.A.J. Nicholson, Permutation procedure for minimazing the number of crossings in a network, Proc. Inst. Elec. Engnrs. 115, 21–26, 1968

[16] R. Cimikowski, Algorithms for the fixed linear crossing number problem, Discrete App. Math. 122, 93–115, 2002

[17] W. Winterbach, The crossing number of a graph in the plane, Master's Thesis, Dept. Appl. Math., University of Stellenbosch, SA, 2005

[18] E. de Klerk, D.V. Pasechnik, Improved lower bounds for the 2-page crossing numbers of $K_{m,n}$ and K_n via semidefinite programming, http://arxiv.org/abs/1110.4824v1, October 2011.

[19] T. Poranen, E. Mäkinen, H. He, A simulated annealing algorithm for the 2-page crossing number problem, Proc. of the International Network Optimization Conference, 22-25 Apr. 2007, Spa, Belgium, 2007

[20] H. He, O. Sýkora, E. Mäkinen, Genetic algorithms for the 2-page drawing problem of graphs, J. Heuristics 13(1), 77–93, 2007

[21] H. He, O. Sýkora, E. Mäkinen, An Improved Neural Network Model for the 2-page Crossing Number Problem, IEEE Trans. Neural Netw. 17(6), 1642–1646, 2006

[22] H. He, O. Sýkora, A. Salagean, Various Island-based Parallel Genetic Algorithms for the 2-page Drawing Problem, Proc. of IASTED International Conference on Parallel and Distributed Computing and Networks, 14-16 February 2006, Innsbruck, Austria (PDCN2006), 2006, 316–323

[23] H. He, O. Sýkora, A. Salagean, E. Mäkinen, Parallelization of Genetic Algorithm for the 2-page Crossing Number Problem, J. Paral. Distr. Com. 67(2), 229–241, 2007

[24] GDToolkit: http://www.dia.uniroma3.it/∼b:24/

[25] R. Cimikowski, P. Shope, A neural network algorithm for a graph layout problem, IEEE Trans. Neural Netw. 7(2), 341–346, 1996

[26] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, B. Math. Biophys. 5, 115–133, 1943

[27] H. He, A. Salagean, E. Mäkinen, One- and two-page crossing numbers for some types of graphs, International J. Computer Mathematics 87(8), 1667–1679, 2010

[28] R. Fulek, H. He, O. Sýkora, I. Vrt'o, Outerplanar Crossing Numbers of 3-Row Meshes, Halin Graphs and Complete p-Partite Graphs, Proc. SOFSEM'05, LNCS 3381, 376–379, 2005

[29] R.K. Guy, Crossing number of graphs, In Y. Alavi, D.R. Lick, A.T. White (Eds), Graph Theory and Applications: Proc. of the Conference at Western Michigan University, Kalamazoo, Mich., Springer-Verlag, New York, 1972, 111–124

[30] F. Boesch, R. Tindell,Circulants and their connectivities, J. Graph Theory 8, 487–499, 1984

[31] F. Bernhart, P. Kainen,The book thickness of a graph, J. Combin. Theory Ser. B. 27, 320–331, 1979

[32] M. Yannakakis, Four page are neccessary and suffcient for planar graphs (extended abstract), Proc. of the Eighteenth Annual ACM Symposium on Theory of Computing, Berkeley, California, United States, 1986, 104–108

[33] J. Adamson, B.R. Richter, Arrangements, circular arrangements and the crossing number of $C_7 \times C_n$, J. Combin. Theory Ser. B 90, 21–39, 2004

[34] L.Y. Glebsky, G. Salazar, The conjecture $cr(C_m \times C_n) = (m-2)n$ is true for all but finitely n, for each m, J. Graph Theory 47, 53–72, 2004

[35] F. Shahrokhi, O. Sýkora, L.A. Székely, I. Vrt'o, Intersection of Curves and Crossing Number of $C_m \times C_n$ on Surfaces, Descrete Comput. Geom. 19(2), 237–247, 1998