Heuristic Crossing Minimisation Algorithms for the Two-page Drawing Problem *

Hongmei He[1], Ondrej Sýkora[1], Ana Sălăgean[1], Imrich Vrťo[2]

[1]Department of Computer Science, Loughborough University
Loughborough, Leicestershire LE11 3TU, The United Kingdom
[2]Department of Informatics, Institute of Mathematics
Slovak Academy of Sciences
Dúbravská 9, 842 35 Bratislava, Slovak Republic
[1]{*H.He, A.M.Salagean*}*@lboro.ac.uk,* [2]*Imrich.Vrto@savba.sk*

Abstract. The minimisation of edge crossings in a book drawing of a graph G is one of the important goals for a linear VLSI design, and the two-page crossing number of a graph G provides an upper bound for the standard planar crossing number. We propose several new heuristics for the two-page drawing problem, and test them on benchmark test suites, Rome graphs and Random Connected Graphs. We also test some typical graphs, and get some exact results. The results for some circulant graphs are better than the one presented by Cimikowski. We have a conjecture for cartesian graphs, supported by our experimental results, and provide direct methods to get optimal solutions for 3- or 4-row meshes and Halin graphs.
Keywords: book drawings, one-page crossing number, two-page crossing number, Hamiltonian cycle, strategies of edge distribution, optimal solutions

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 (κ pages). Such drawings are called κ-page book drawings, and they correspond to the linear VLSI design. Edge crossing minimisation is the most important goal in linear VLSI design, since smaller number of crossings means cheaper design. The minimal number of edge crossings over all κ-page book drawings of a graph is called the κ-page book crossing number [18].
The one-page [18] crossing number corresponds to outerplanar [11] (also called convex [19], or circular [21]) crossing number, which is the minimal possible number of pairs of crossing edges in a drawing where one places vertices of a graph G along a circle, and the edges are drawn as straight lines. Therefore the one-page drawing problem is equivalent to finding an order of the vertices on the circle which minimises the number of edge crossings. We denote the one-page crossing number as $\nu_1(G)$, following the notation in [18]. The problem has been proved to be NP-hard [13].
The two-page drawing problem can be reformulated similarly: one places the vertices of a graph G along a circle and every edge is completely drawn in one of two colours. The two-page crossing number of G is the smallest possible number of crossings of edges with the same colour, and we denote it as $\nu_2(G)$. The problem is also NP-hard [14]. Similarly, a κ-page drawing of a graph is equivalent to a κ-colour circular drawing of the graph.

* This research was supported by the EPSRC grant GR/S76694/01 and by VEGA grant No. 2/3164/23

For a graph G with n vertices 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 distribution of edges to the 2 (or κ) pages is stored in a matrix *d_adj* so that for any two positions i and j on the circle, $d\_adj[i][j]$ equals the number (between 1 and κ) of the page the edge is on, with 0 indicating that no edge exists between the corresponding positions. Given the positions of the two ends of an edge, i and j, and the positions of the two ends of the other edge, k and l, in the current layout, if $i < k < j < l$ and the two edges are in the same page, then they will produce a crossing. So we can calculate the number of crossings in a κ-page drawing of G with the following formula:

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

where

$$d\_adj[i][j] \bigodot d\_adj[k][l] = \begin{cases} 1 & \text{if } d\_adj[i][k] = d\_adj[j][l] \neq 0; \\ 0 & \text{if } otherwise. \end{cases} \tag{2}$$

A lot of heuristic algorithms were designed for the one-page drawing problem, e.g., the algorithm of Mäkinen [15], the CIRCULAR algorithm [21], the algorithm of Baur and Brandes [2], and AVSDF [10]. The two-page crossing number as an approximation to the planar crossing number was first studied in [16, 17] but no thorough testing was done there. The most important paper in this area is [5], 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 distribution of edges between the two pages were designed and tested. The two-page crossing number of a graph G provides an upper bound for the standard planar crossing number of G. Recently, Winterbach [22] proposed heuristics for the two-page crossing numbers and applied them to estimating the planar crossing number of some small complete multipartite graphs.

In this paper we design new two-page drawing heuristic algorithms based on the latest one-page drawing algorithms [2] and [10], and compare different edge distribution strategies. For some structural graphs, we apply the strategy of edge distribution based on an optimal order of vertices [5]. For our experiments we used the following benchmark test suites:

Random Connected Graphs (RCGs) with different densities and sizes, which were used in [10].

Rome Graphs, which are taken from the test suite of GDToolkit [24] and were utilized in [2, 10]. We use two subsets of undirected graphs from this suite:

-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.

Special Graphs: including random 3-regular graphs, Halin graphs, meshes, complete p-partite graphs, complete graphs, circulant graphs, and Cartesian Graphs.

The paper will follow the order: in Section 2 and 3, we mention two latest one-page algorithms and present four two-page strategies; in Section 4, we compare two-page algorithms in different

ways, compare the results of some circulant graphs with Cimikowski's [5], and provide exact results for some structural graphs; in Section 5, we present the direct methods to get optimal solutions for 3- or 4-row meshes and Halin graphs.

2 One-page drawing algorithms

2.1 The algorithm of Baur and Brandes (BB+)

The best algorithm of Baur and Brandes [2] consists of two phases: greedy and sifting.

Greedy phase: 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, and then appended to the end that yields fewer crossings between the open edges and the edges that become closed by placing v, where an open edge is one containing an endpoint that has not been placed. 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 Fig. 1, there are eight such crossings for the left end and only five for the right end. The running time is $O((n+m)\log n)$.

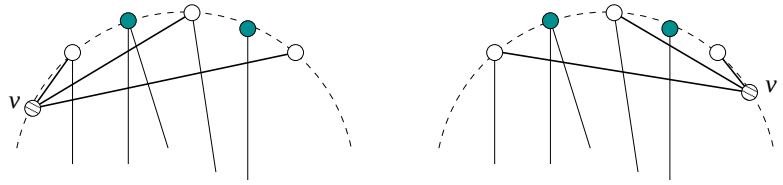


Fig. 1. Incident edges of v cross with open edges.

Sifting phase: every vertex is moved along the circle, while the other vertices stay in their current order. In our implementation, at every step, a vertex will clockwise move from its current position to the next position. The vertex is then placed in its (locally) optimal position. The phase can be run in $O(nm)$ time.
We use BB to denote the greedy phase, and BB+ to denote the combination of the two phases.

2.2 Algorithm AVSDF+

Algorithm AVSDF+ [10] consists of two phases too: greedy and adjusting.

Greedy Algorithm: AVSDF is a variation of the Depth First Search (DFS) algorithm. At 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. The running time of the AVSDF algorithm is $O(m)$.

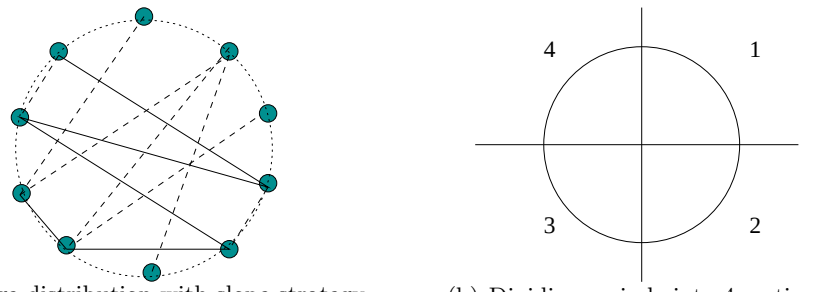
Postprocessing phase(adjusting): in this phase a vertex with the largest number of crossings caused by its incident edges is selected first, and then the best position among the current one and the ones immediately after each of its adjacent vertices is found. The procedure is repeated until no vertex can be adjusted. The running time is $O(mn)$. We denote AVSDF plus adjusting as AVSDF+.

3 Two-page drawing algorithms

Usually the smaller the one-page crossing number is, the smaller is the two-page crossing number. So we try to find an optimal distribution of edges in two pages based on the order given by one-page algorithms. We use the algorithms described above as the first phase to minimise the one-page crossings, and then use the following strategies of edge distribution in the second phase respectively.

3.1 Slope strategy (SLOPE)

In a circular drawing, each edge has a slope to the horizontal line. The SLOPE two-page strategy distributes the edges between pages according to their slopes. If an edge 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 Fig.2 (a)), otherwise the edge is put on page 1 (broken edges in Fig. 2 (a)).



(a) Edge distribution with slope strategy (b) Dividing a circle into 4 sections

Fig. 2. Slope strategy and four sections in a circle

We can divide a circle into four sections as in Fig. 2 (b). The vertex positions are numbered clockwise around the circle, with 0 being at the top. Table 1 describes how the position i of a vertex is related to the four sections.
Considering an edge $e(u, v)$ with u in position i and v in position 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 both located in the section 3, then $n \leq i+j < \frac{3}{2}n$. If i and j are located in section 2 and section 4, respectively, then $n \leq i+j < \frac{3}{2}n$. In these cases, we can say the angle between the edge and horizontal axis is larger than 90°. Considering all the other cases, including those where i and j are in the same section or in adjacent sections, we

Table 1. Vertex positions on each section of a circle

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

obtain the relationship between the slope value and the positions of the two ends of the edge: if $(0 < i+j < \frac{1}{2}n$ or $n \leq i+j < \frac{3}{2}n)$ the edge $e(i,j)$ has a negative slope, otherwise it has positive slope. The vertical edges are considered to have positive slope and the horizontal ones negative slope. Note that the slope strategy is not invariant to rotations of the drawing. We can compute the matrix of a drawing, *d_adj*, according to slope strategy (see Algorithm 1) in $O(n^2)$ time.

Algorithm 1 Slope(*g_adj*, *order*)

```
1: define an array, d_adj;
2: for (i= 0 to |V|) do
3:    for (j=0 to |V|) do
4:       u = order[i]; v = order[j];
5:       if (g_adj[u][v]=0) then
6:          d_adj[i][j] = d_adj[j][i] = 0;
7:       else
8:          if (i + j < 1/2|V| or (i + j ≥ |V| and i + j < 3/2|V|)) then
9:             d_adj[i][j] = d_adj[j][i] = 2;
10:         else
11:            d_adj[i][j] = d_adj[j][i] = 1;
12:         end if
13:      end if
14:   end for
15: end for
16: return d_adj;
```

3.2 Place edges according to their length (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 length of an edge, the larger the probability of its crossing with other edges. Whether the edges with length 1 are on page 1 or page 2, they do not create any crossing. Therefore we create a sorted edge list without the edges with length 1 on non-increasing length, where ties are broken by ordering (i,j) lexicographically. Initially all edges are placed in page 1, and then route each edge in the list in turn to the page where smaller crossings will be produced. The process is iterated until either there is no improvement or there have been 5 traversals of the list of edges (the number 5 was found to be sufficient experimentally).

We denote this strategy as LEN. It is similar with the algorithm E-len by Cimikowski [5]. The difference of LEN and E-len is that LEN starts with a one-page drawing and routes some of the edges to the other page, while E-len is based on a fixed linear order of vertices and adds edges one by one to one of the two pages. The running time is $O(m)$.

3.3 Adjust edges according to descending crossings(CRS)

The strategy of routing edges according to descending crossings is based on an initial one-page drawing with a fixed order and all edges in page 1. We start with the edge which creates most crossings and put it to the other page, then recalculate the crossings created by the routed edge. If the crossing number is larger than that before, the edge is restored to the original page, and next edge will be routed. Otherwise, we modify the numbers of crossings created by other edges, sort edges according to the number of crossings created by each edge, and repeat the whole process until no edge can be adjusted. We denote this strategy CRS. In the worst case the running time is $O(m^2)$, but the running time is often $O(m)$. In the implementation, we keep two lists, *eplist* and *crlist*. The *eplist* stores each edge and its page assignment. In the list *crlist* each element consists of two components, *post* and *cr*, which are the index number of each edge in the *eplist* and the crossing number created by each edge, respectively. During the routing procedure, after every modification of the edge assignment, *crlist* will be sorted on order of descending crossing numbers (see Algorithm 2).

Algorithm 2 Adjusting edges on crossings

```
1: Create an edge list with page attribute, eplist;
2: Create a sorted crossing list with edge index, crlist;
3: Calculate current crossings, currCr;
4: i=0; lastCr=currCr;
5: while (i < m or lastCr > currCr) do
6:    lastCr = currCr;
7:    s = crlist[i].post; u = eplist[s].u; v = eplist[s].v;
8:    if (crlist[i].cr=0) then
9:       i=i+1; // to next edge.
10:   else
11:      eplist[s].page = eplist[s].page mod 2+1; // set the edge to the other page.
12:      cr = calEdgeCr(eplist);
         //calculate the crossing number created by the adjusted edge.
13:      if (cr ≥ crlist[i].cr) then
14:         eplist[s].page = eplist[s].page mod 2+1; i = i+1;
            //restore the edge to the original page, and route the next edge
15:      else
16:         currCr = currCr − crlist[i].cr + cr; crlist[i].cr = cr;
17:         modilist(eplist,crlist,i);
            //Adjust crlist remains sorted on descending crossings.
18:         i = 0;
19:      end if
20:   end if
21: end while
22: return currCr;
```

3.4 Hybrid of one-page algorithm and edge pre-assignment (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 distribution directly for a two-page drawing. A hybrid heuristic algorithm of AVSDF and the edge distribution is presented here, which places an edge incident to the currently placed vertex based on the smallest crossings produced by the edge and already placed edges. We denote it as AVSDF_EP (see Algorithm 3).

Algorithm 3 A hybrid algorithm of AVSDF with distribution of edges

```
1: Create an adjacency list, each vertex with a linked list sorted in descending degree order
2: Define an array order[n], and a stack, S;
3: Get the vertex with the smallest degree from the given graph, and push it into S;
4: while (S is not empty) do
5:    Pop a vertex v, from S;
6:    if (v is not in order) then
7:       Append the vertex v into order;
8:       for (each vertex u adjacent to v, in decreasing order of degree(u)) do
9:          if ( u is not in order) then
10:            push u into S;
11:         else
12:            place the edge e(u, v) to the page where fewer new crossings are created by adding the edge;
13:         end if
14:      end for
15:   end if
16: end while
```

4 Experiments

The experiments of Cimikowski [5] were done based on a fixed Hamiltonian cycle for some special 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 Hamiltonian cycle might not exist, or even if it exists, we might not be able to find it efficiently. Our experiments aim at finding the relationship between one-page drawing and two-page drawing, and what is the most important factor to affect the crossings. So experiments are done on a variety of graphs described previously.

4.1 Test of two page drawing algorithms

Test two-page strategies based on one-page algorithms

We compare pairs of algorithm combinations formed by 2 heuristics for one-page drawings (namely AVSDF+ and BB+) and 3 strategies of edge distribution (namely SLOPE, LEN and CRS), as well as the hybrid algorithm AVSDF_EP, and explore the relationship between one-page and two-page drawings on ALF_CU, RND_BUP, RCGs, 3-regular graphs, etc. In Table 2, the number of times an algorithm gets the best results in all test data is listed, with the highest

values shown in bold. The BB+_CRS performs best on all graphs except for cartesian products and Halin graphs, where AVSDF+_CRS performs best, and circulant graphs $C_{kn}(1, n)$, where AVSDF_EP performs best. We use $X \prec Y$ to express that algorithm Y obtains the best results more times than algorithm X. From Table 2, for most types graphs there exists the relationship: SLOPE $\prec$ LEN $\prec$ CRS both when combined with AVSDF+ or BB+, while AVSDF_EP is better than AVSDF+_SLOPE and BB+_SLOPE.

Table 2. The number of times best results are obtained on each type of graphs with all strategies based on one-page algorithms

Graphs (number)	avsdf+ _slope	avsdf+ _len	avsdf+ _crs	avsdf_ep	bb+ _slope	bb+ _len	bb+ _crs
ALF_CU(268)	52	128	148	92	47	149	**170**
RND_BUP(168)	19	61	82	40	17	84	**85**
RCGs(360)	37	94	141	50	17	115	**170**
3-d RCGs(45)	3	14	12	3	2	17	**25**
CmxCn(49)	2	16	**21**	0	2	10	11
Mesh(49)	0	17	17	1	2	23	**31**
Halin(400)	7	166	**274**	28	8	116	211
$C_n(34)$	0	10	15	9	1	8	19
$C_{kn}(1, n)(68)$	17	21	18	**39**	7	38	34

Test two-page strategies based on a fixed order of vertices

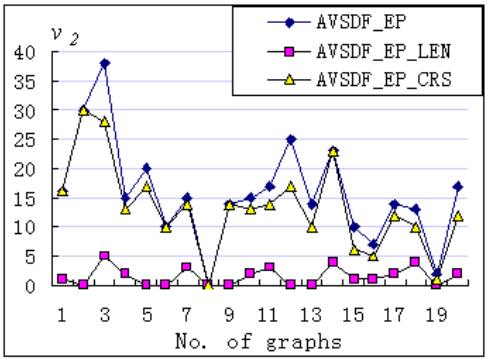
We also test some structural graphs, such as circulant graphs, cartesian graphs, complete graphs and complete p-partite graphs with the method used in [5] – first find an optimal order of vertices for a one-page drawing of a structural graph, and then apply a strategy of edge distribution. Table 3 shows the statistical results for the numbers of times when each strategy gets best results in all tests on each type of graphs. The results indicate that an algorithm has different performance for different structural graphs. For example, SLOPE strategy is the best for Complete graphs and Complete p-partite graphs. In Section 4.3, we will present some exact results.

Table 3. The number of times when best results are obtained for each two-page strategy based on a fixed order for some structural graphs

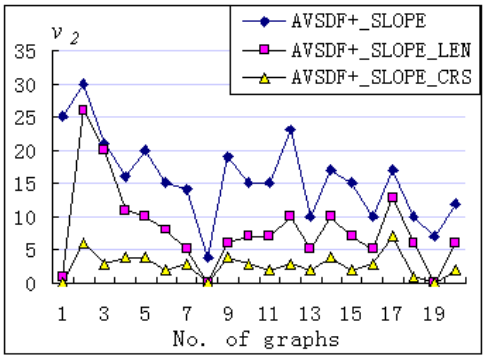
Graphs	C_n[5]	$C_{kn}(1, n)$	$K_n(p)$	K_n	$C_m \times C_n$
slope	0	34	**11**	**28**	11
len	23	36	6	14	24
crs	**25**	**42**	8	21	**43**

Test of two-page strategies based on different two-page preprocessings

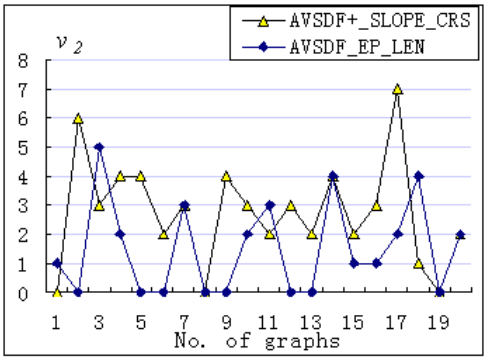
We also investigated the effect of different two-page preprocessings, AVSDF_EP and AVSDF+_SLOPE on the two-page crossing number. Fig. 3 (a) and (b) show the two-page crossing numbers of 20 different graphs with 60 vertices in RND_BUP after using a preprocessing step and then after using a two-page strategy such as LEN or CRS. From Fig. 3 (a), based on AVSDF_EP pre-



(a) AVSEDF_EP as a preprocessing step



(b) AVSDF+_SLOPE as a preprocessing step

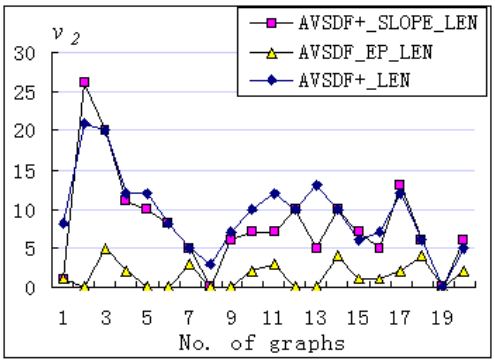


(c) Comparison between AVSDF+_SLOPE_CRS and AVSDF_EP_LEN

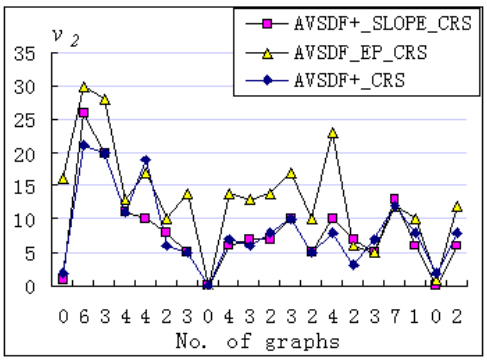
Fig. 3. Two-page strategies based on different preprocessing steps (I)

processing, algorithm LEN achieved a much greater improvement to the results of AVSDF_EP than algorithm CRS. In contrast, from Fig. 3 (b), based on AVSDF+_SLOPE, algorithm CRS achieve greater improvement to results of AVSDF+_SLOPE than algorithm LEN. Furthermore, from Fig. 3 (c), it can be seen that the results by AVSDF_EP_LEN are better than that by AVSDF+_SLOPE_CRS in most cases.

From Fig. 4, we can conclude that for two-page strategy LEN, the best preprocessing is AVSDF_EP, while for two-page strategy CRS, the best preprocessing is the one-page algorithm AVSDF+ without two-page preprocessing.



(a) LEN based on different preprocessing steps



(b) CRS based on different preprocessing steps

Fig. 4. Two-page strategies based on different preprocessing steps (II)

4.2 Test on RCGs with different edge densities

We compare all the combinations, AVSDF+_SLOPE, BB+_SLOPE, AVSDF+_LEN, BB+_LEN, AVSDF+_CRS, and BB+_CRS on RCGs with edge density 1%, 2%, and 5%, where the density is defined as the ratio of edge number to maximal possible edge number. Furthermore we explore the effect of density (Tables. 4-6) by statistical results of crossing numbers for each density of graphs. For each density, 12 groups of graphs with different number of vertices were tested, and every group includes 10 different graphs, for which the average crossing number is calculated. Tables. 4-6 show that AVSDF+_SLOPE gets better results than BB+_SLOPE does for all densities, and the results by AVSDF+_SLOPE and BB+_SLOPE become closer with the rise of density. No matter whether AVSDF+ or BB+ algorithm precedes the two-page algorithms and what density graphs are tested, CRS gets slightly better results than LEN does, and both are significantly better than SLOPE.

Table 4. Two-page crossing numbers obtained by each combination of algorithms on the graphs with density=1%

Vertex Number	avsdf+_slope	avsdf+_len	avsdf+_crs	bb+_slope	bb+_len	bb+_crs
200	**0**	**0**	**0**	3	**0**	**0**
205	5	**1**	**1**	14	3	3
210	14	7	**6**	23	9	9
215	26	16	**15**	52	24	25
220	48	36	33	67	33	**31**
225	76	57	57	96	**53**	**53**
230	107	89	**87**	149	95	90
235	135	111	110	172	112	**107**
240	192	155	147	203	139	**135**
245	224	200	195	269	194	**187**
250	287	246	**241**	360	265	248
255	337	297	297	387	284	**279**

Table 5. Two-page crossing numbers obtained by each combination of algorithms on the graphs with density=2%

Vertex Number	avsdf+_slope	avsdf+_len	avsdf+_crs	bb+_slope	bb+_len	bb+_crs
100	**0**	**0**	**0**	**0**	**0**	**0**
105	2	**1**	**1**	6	**1**	**1**
110	9	6	5	14	5	**4**
115	26	18	19	34	**15**	16
120	41	29	**26**	53	31	28
125	76	59	58	93	60	**57**
130	96	77	78	115	75	**71**
135	140	113	116	158	106	**106**
140	192	162	**155**	229	158	159
145	279	227	217	288	217	**208**
150	351	308	301	374	285	**282**
155	415	389	361	463	373	**359**

Table 6. Two-page crossing numbers obtained by each combination of algorithms on the graphs with density=5%

Vertex Number	avsdf+_slope	avsdf+_len	avsdf+_crs	bb+_slope	bb+_len	bb+_crs
40	**0**	**0**	**0**	**0**	**0**	**0**
45	2	**0**	**0**	2	**0**	**0**
50	10	5	6	14	5	**4**
55	22	17	17	24	**13**	**13**
60	43	39	**33**	59	38	36
65	94	68	68	100	66	**63**
70	146	118	116	153	114	**108**
75	230	193	182	224	179	**169**
80	333	279	269	337	263	**257**
85	439	373	363	449	356	**346**
90	597	534	**515**	651	546	528
95	833	717	704	838	697	**691**

4.3 Typical graph test

We tested some circulant graphs used in [5] and some Halin graphs with two-page strategies based on one-page algorithms. For some typical classes of graphs such as complete p-partite graphs, 3-row meshes, we know optimal one-page drawings, or for some other graphs such as 4-row meshes, cartesian products, we have a hypothesis of optimal one-page drawings, so we first find optimal one-page drawings and then apply a strategy of edge distribution. This method of experimentation was also used in [5].

Circulant graph test

Definition 1. *[4] Circulant graphs of the form* $C_n(a_1, a_2, ..., a_k)$*, where* $0 < a_1 < a_2 < ... < a_k < (n+1)/2$*, are regular Hamiltonian graphs with* n *vertices, and with vertices* $i \pm a_j(\ mod\ n)$*,* $j = 1..k$*, adjacent to each* i*.*

Some circulant graphs were tested based on a fixed Hamiltonian cycle, and the exact results were listed in [5]. We tested these circulant graphs, but first applied a heuristic algorithm to find a good one-page drawing, and then applied a strategy of edge distribution. In the second rightmost column of Table 7, there are listed either the optimal values related to the fixed order of vertices based on a Hamiltonian cycles [5], or theoretical lower and upper bounds (a:b in column Opt.), if the branch-and-bound algorithm of Cimikowski [5] was not applicable. The rightmost column contains the best results obtained with 8 heuristic algorithms based on the Hamiltonian cycle of each circulant graph by Cimikowski [5]. Our results are similar to or better than Cimikowski's [5] (Table 7). For some circulant graphs such as $C_{24}(1,3,5)$, $C_{38}(1,7)$, $C_{40}(1,5)$, $C_{42}(1,4)$, and $C_{46}(1,4)$, our best results (data with star in Table 7) are even better than the optimal values that can be obtained from branch and bound algorithm based on a fixed order of vertices [5]. This is because our algorithms, unlike [5], are not restricted to the fixed order. For example, Fig. 5 presents the best solution for $C_{42}(1,4)$, where $\nu_2(C_{42}(1,4)) = 38$, while the optimal value based on a fixed order of vertices [5] was 42.

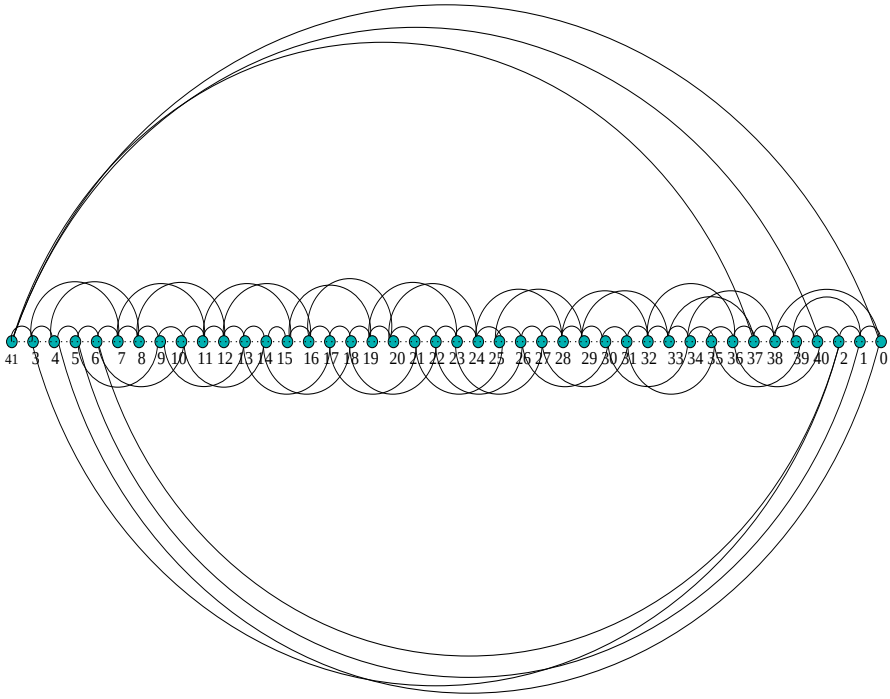


Fig. 5. The best solution for $C_{42}(1,4)$

Table 7. Test results for circulants with two-page strategies based on one-page algorithms (the data with star are even better than the optimal values that can be obtained from branch and bound algorithm based on a fixed order of vertices)

Graphs	avsdf+ _len	avsdf+ _crs	avsdf+ _slope _len	avsdf+ _slope _crs	bb+ _len	bb+ _crs	bb+ _slope _len	bb+ _slope _crs	Opt. [5]	best[5]
$C_{20}(1,2)$	0	0	0	0	0	0	0	0	0	0
$C_{20}(1,2,3)$	22	24	24	24	22	24	24	24	22	22
$C_{20}(1,2,3,4)$	84	70	84	84	84	70	84	84	26:870	70
$C_{22}(1,2)$	0	0	2	2	0	0	2	2	0	0
$C_{22}(1,2,3)$	24	24	30	30	24	24	30	30	24	24
$C_{22}(1,3,5,7)$	216	203	236	210	244	225	229	229	28:1056	200
$C_{24}(1,3)$	14	15	14	13	14	14	14	14	12	12
$C_{24}(1,35)$	88	73	74	82	86	70*	74	78	72	76
$C_{24}(1,3,5,7)$	250	217	242	217	280	237	273	273	30:1260	216
$C_{26}(1,3)$	16	17	18	14	18	19	20	17	14	14
$C_{26}(1,3,5)$	96	83	88	90	94	80	82	82	6:650	82
$C_{26}(1,4,7,9)$	351	365	366	366	344	311	313	313	32:1482	364
$C_{28}(1,3)$	18	19	20	18	19	19	19	19	14	16
$C_{28}(1,3,5)$	104	87	94	92	92	88	92	92	6:756	86
$C_{28}(1,2,3,4)$	126	98	120	116	126	98	120	116	34:1722	98
$C_{28}(1,3,5,7,9)$	561	561	670	561	610	586	655	654	62:3080	560
$C_{30}(1,3,5)$	106	97	97	97	107	96	105	98	6:870	96
$C_{30}(1,3,5,8)$	319	324	324	308	319	324	343	316	36:1980	302
$C_{30}(1,2,4,5,7)$	412	392	410	392	434	411	440	420	66:3540	392
$C_{32}(1,2,4,6)$	160	192	202	208	128	126	126	128	38:2256	160
$C_{34}(1,3,5)$	128	105	107	105	116	110	107	106	6:1122	106
$C_{34}(1,4,8,12)$	575	620	620	620	309	302	324	326	40:2550	574
$C_{36}(1,2,4)$	36	54	62	68	36	36	36	36	6:1260	36
$C_{36}(1,3,5,7)$	348	331	360	340	346	336	356	355	42:2862	328
$C_{38}(1,7)$	76*	77*	72*	72*	58*	53*	53*	53*	84	86
$C_{38}(1,4,7)$	221	187	205	204	190	176	205	186	6:1406	190
$C_{40}(1,5)$	58	61	56	52*	49*	57	57	56	56	58
$C_{42}(1,4)$	42	40*	40*	38*	48	51	50	50	42	42
$C_{42}(1,3,6)$	162	164	157	154	118	115	108	115	6:1722	158
$C_{42}(1,2,4,6)$	228	238	264	266	164	162	158	158	48:3906	210
$C_{44}(1,4,5)$	168	177	184	178	177	123	149	148	6:1892	180
$C_{44}(1,4,7,10)$	674	629	653	627	631	634	717	698	50:4290	632
$C_{46}(1,4)$	46	44*	42*	42*	63	63	64	64	46	46
$C_{46}(1,5,8)$	281	271	281	280	290	267	275	281	6:2070	296

Halin graph test

Definition 2. *A Halin graph H is a plane graph $H = T \cup C$, where T is a plane tree with no vertex of degree two and at least one vertex of degree three or more, and C is a cycle connecting the leaves of T in the cyclic order determined by a plane embedding of T. The edges in T will be called t-edges and the ones in C will be called c-edges.*

We tested 400 Halin graphs with two-page strategies based on one-page algorithms. We use $X \succ Y$ to express that algorithm X obtains 0 crossings more times than algorithm Y (the two-page crossing number of Halin graphs is 0, see Section 5.2). Considering the number of times when each strategy gets 0 crossings, we have: AVSDF+_CRS(214) $\succ$ BB+_CRS(162) $\succ$ AVSDF+_LEN(132) $\succ$ BB+_LEN(88) $\succ$ AVSDF_EP(26) $\succ$ AVSDF+_SLOPE(7) = BB+_SLOPE(7).

Complete p-partite graph and complete graph test

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

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

In [7] we provided exact results for the one-page crossing number of $K_n(p)$.

Theorem 1. *[7] For a complete p-partite graph with n vertices in each partite set,*

$$\nu_1(K_n(p)) = n^4\binom{p}{4} + \frac{1}{2}n^2(n-1)(2n-1)\binom{p}{3} + n\binom{n}{3}\binom{p}{2}.$$

In [7] we also describe the optimal one-page drawing solution for a complete p-partite graph. Namely, 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 (Fig.6). It is easy to get the optimal order for a complete p-partite graph in linear time. The test was done based on this optimal order(see Table 8). It is shown that SLOPE gets the best results for all complete p-partite graphs tested, but the other two strategies, LEN and CRS, are not far behind.

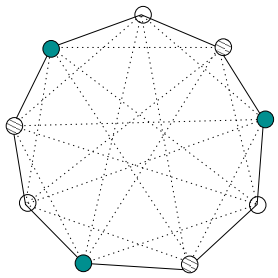


Fig. 6. Optimal order of $K_3(3)$

Guy [9] presented an upper bound for the standard planar crossing number for complete graphs.

Theorem 2. *[9] The crossing number of the complete graph satisfies the inequality*

$$cr(K_n) \leq \frac{1}{4}\left\lfloor\frac{n}{2}\right\rfloor\left\lfloor\frac{n-1}{2}\right\rfloor\left\lfloor\frac{n-2}{2}\right\rfloor\left\lfloor\frac{n-3}{2}\right\rfloor.$$

Table 8. Test results on $K_n(p)$ by two-page strategies based on optimal one-page drawings

Graphs	Opt.(ν_1)	SLOPE	LEN	CRS	Graphs	Opt.(ν_1)	SLOPE	LEN	CRS
$K_3(2)$	3	1	1	1	$K_3(4)$	279	86	90	87
$K_4(2)$	16	4	6	4	$K_4(4)$	1024	336	338	344
$K_5(2)$	50	16	16	16	$K_5(4)$	2725	916	916	916
$K_6(2)$	120	36	36	36	$K_6(4)$	5976	2052	2056	2056
$K_7(2)$	245	81	81	81	$K_7(4)$	11515	4002	4002	4002
$K_8(2)$	448	144	160	144	$K_8(4)$	20224	7104	7110	7108
$K_9(2)$	756	256	256	256	$K_9(4)$	33129	11720	11721	11720
$K_3(3)$	54	16	16	16	$K_3(5)$	885	291	299	291
$K_4(3)$	216	68	80	68	$K_4(5)$	3120	1056	1064	1064
$K_5(3)$	600	196	198	196	$K_5(5)$	8125	2813	2813	2813
$K_6(3)$	1350	450	452	454	$K_6(5)$	17580	6156	6156	6156
$K_7(3)$	2646	900	900	900	$K_7(5)$	33565	11887	11887	11887
$K_8(3)$	4704	1616	1620	1616	$K_8(5)$	58560	20864	20902	20868
$K_9(3)$	7776	2704	2704	2704	$K_9(5)$	95445	34233	34233	34233

The equality has been shown to hold in Theorem 2, for $n \leq 10$ [9]. For complete graphs, we directly apply the strategies of edge distribution on the initial order 0,1,...n-1 (see Table 9). SLOPE gets optimal or conjectured optimal results every time, while the results of the other two strategies LEN and CRS are 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 [5] as well. Those with star are conjectured optimal values, as they are the same as the values from Guy's conjecture for standard planar crossing numbers of complete graphs. Therefore, we can conjecture the two-page crossing number of complete graphs as belows:

Conjecture 1. For any complete graph K_n, $\nu_2(K_n) = \frac{1}{4}\left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor \left\lfloor \frac{n-2}{2} \right\rfloor \left\lfloor \frac{n-3}{2} \right\rfloor$.

Table 9. Test results on K_n by two-page strategies based on a fixed order

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

3- or 4-row mesh test

For 3-row meshes, $P_3 \times P_n$, we knew the optimal one-page crossing number:

Theorem 3. *[7] For any 3-row mesh:*
for any odd $n \geq 3$*:* $\nu_1(P_3 \times P_n) = 2n - 3$*;*
for any even $n \geq 4$*:* $\nu_1(P_3 \times P_n) = 2n - 4$.

Fig. 8(a) and (b) show the Hamiltonian cycles of two 4-row meshes. For all 4-row meshes, we can get Hamiltonian cycles in this way, and each column added will add 4 crossings, which is also supported by our experimental results. Therefore, for 4-row meshes, $P_4 \times P_n$, we have a conjecture for the one-page crossing number as below:

Conjecture 2. For any 4-row mesh: $\nu_1(P_4 \times P_n) = 4n - 8$, $n \geq 4$.

For a 3- or 4-row mesh, it is easy to construct a Hamiltonian cycle or path to get the optimal or conjectured optimal one-page drawing in linear time (Fig. 7(a)(b) and Fig. 8(a)(b)).

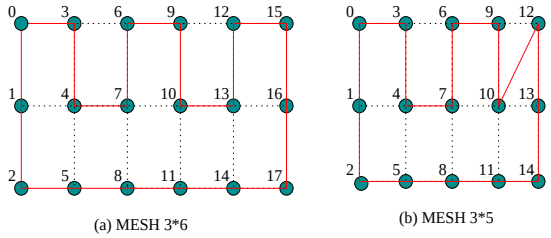


Fig. 7. Optimal order for one-page drawing of 3-row meshes

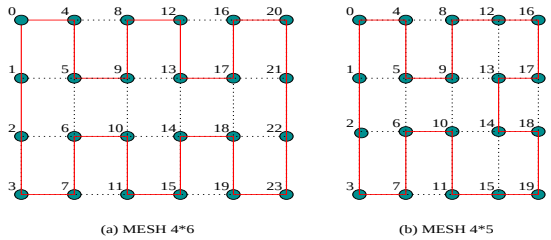


Fig. 8. Conjectured optimal order for one-page drawing of 4-row meshes

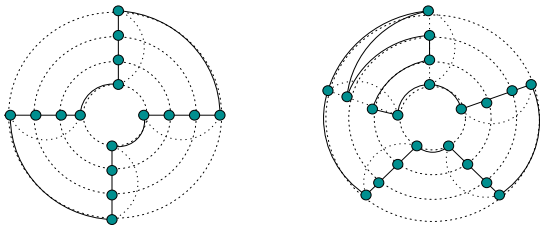
We tested 3- or 4-row meshes with the strategies of edge distribution based on these Hamiltonian Cycles (paths). We also tested a combination of SLOPE followed by either LEN or CRS. Table 10 shows the experimental results. LEN and CRS get 0 crossings for every mesh tested. We show in Section 5.1 that the two-page crossing number of 3- and 4-row meshes is zero.

Table 10. two-page crossing number of 3- or 4-row meshes by two-page strategies based on (conjectured) optimal one-page drawings

Graphs	slope	len	crs	slope_len	slope_crs
$P_3 \times P_4$	1	0	0	0	0
$P_3 \times P_5$	2	0	0	0	0
$P_3 \times P_6$	4	0	0	1	0
$P_3 \times P_7$	7	0	0	2	0
$P_3 \times P_8$	6	0	0	1	0
$P_3 \times P_9$	9	0	0	1	0
$P_4 \times P_4$	2	0	0	0	0
$P_4 \times P_5$	6	0	0	2	2
$P_4 \times P_6$	10	0	0	0	0
$P_4 \times P_7$	12	0	0	4	2
$P_4 \times P_8$	14	0	0	0	0
$P_4 \times P_9$	18	0	0	2	2

Cartesian product $C_m \times C_n$ test

There is a lot of research about Cartesian product graphs $C_m \times C_n$ [1, 8, 20]. There is a natural drawing of $C_m \times C_n$ having $(m-2)n$ crossings, where $m \leq n$: 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[1]. Fig. 9 and Fig. 10 present these drawings for $C_4 \times C_4$, $C_4 \times C_5$ and $C_5 \times C_5$.



(a) C4xC4 (b) C4xC5

Fig. 9. A drawing of $C_4 \times C_4$ and $C_4 \times C_5$

Recall that a graph is subhamiltonian if it is a subgraph of a planar graph that has a hamiltonian cycle.

Lemma 1. *[3, 23] If a graph is a subhamiltonian planar graph, then the two-page crossing number of the graph is 0.*

Obviously, according to the drawings described above, all $P_m \times C_n$ are subhamiltonian planar graphs. According to Lemma 1, we can conclude:

Theorem 4. *for a $P_m \times C_n$ graph, $\nu_2(P_m \times C_n) = 0$.*

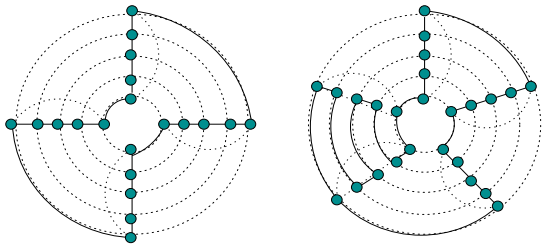


Fig. 10. A drawing of $C_4 \times C_5$ and $C_5 \times C_5$

In Fig. 9 and Fig. 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 Fig. 9(a); When n is odd and m is odd, we can construct a Hamiltonian cycle as in Fig. 10(b); When n is odd but m is even, we can construct the Hamiltonian cycle as in Fig. 10 (a), but it is not optimal for our purpose of two-page drawing. In this case, we also can construct a Hamiltonian path as in Fig. 9(b). So we test Cartesian products $C_m \times C_n$ based on these Hamiltonian cycles (paths) of each graph. Table. 11 shows the test results, and the data with star indicate the results are equal to the conjectured optimal values (see Conjecture 3).
Adamson and Richter proved the following theorem about cartesian product graphs in [1]:

Theorem 5. *For each integer $m \geq 3$, there is an integer $N(m) \leq m(m+1)$ such that $cr(C_m \times C_n)$=(m-2)n, if $n \geq N(m)$.*

For any drawing D of a graph G in a plane, the number of crossings is greater or equal to $cr(G)$, the standard crossing number of G. So for a graph G, the two-page crossing number, $\nu_2(G) \geq cr(G)$. Therefore the two-page crossing number of $C_m \times C_n$ is at least $(m-2)n$. According to our experimental results, we have Conjecture 3.

Conjecture 3. For the Cartesian product $C_m \times C_n$ with $3 \leq m \leq n < 9$, $\nu_2(C_m \times C_n) = (m-2)n$.

5 Optimal solutions for 3- or 4-row meshes and Halin graphs

5.1 Optimal Solutions for 3- or 4-row meshes

Clearly, 3- or 4-row meshes are subhamiltonian planar graphs. According to Lemma 1, we have the following theorems.

Theorem 6. *For any 3-row mesh: $\nu_2(P_3 \times P_n)$=0.*

Theorem 7. *For any 4-row mesh: $\nu_2(P_4 \times P_n)$=0.*

Two-page drawing with 0 crossings can be obtained as follows. In Fig. 7 and Fig. 8, the solid edges form a Hamiltonian cycle (path). For a 3-row mesh, obviously, if we put the broken edges

Table 11. Two-page crossing number of $C_m \times C_n$ by two-page strategies based on the fixed Hamiltonian cycles(paths); the data with star indicate the results are equal to the conjectured optimal values

Graphs	slope	len	crs	slope_len	slope_crs	Conjectured
$C_3 \times C_3$	4	3*	4	4	4	3
$C_3 \times C_4$	4*	4*	4*	4*	4*	4
$C_3 \times C_5$	8	5*	11	5*	5*	5
$C_3 \times C_6$	14	6*	6*	10	10	6
$C_3 \times C_7$	27	15	13	7*	11	7
$C_3 \times C_8$	24	8*	8*	16	16	8
$C_3 \times C_9$	37	15	15	11	11	NA
$C_4 \times C_4$	8*	8*	8*	8*	8*	8
$C_4 \times C_5$	16	15	15	10*	10*	10
$C_4 \times C_6$	28	12*	12*	20	20	12
$C_4 \times C_7$	41	23	19	14*	22	14
$C_4 \times C_8$	48	16*	16*	32	32	16
$C_4 \times C_9$	66	27	23	24	28	NA
$C_5 \times C_5$	22	23	23	15*	15*	15
$C_5 \times C_6$	44	24	18*	32	32	18
$C_5 \times C_7$	63	31	34	21*	35	21
$C_5 \times C_8$	76	32	24*	52	52	24
$C_5 \times C_9$	100	39	40	41	60	NA
$C_6 \times C_6$	62	36	24*	46	46	24
$C_6 \times C_7$	86	41	30	28*	52	28
$C_6 \times C_8$	108	48	32*	76	76	32
$C_6 \times C_9$	137	53	38	58	58	NA
$C_7 \times C_7$	112	41	63	35*	70	35
$C_7 \times C_8$	144	64	40*	104	104	40
$C_7 \times C_9$	179	51	73	77	115	NA
$C_8 \times C_8$	184	48*	48*	136	136	48
$C_8 \times C_9$	224	57	56	98	98	NA
$C_9 \times C_9$	274	69	63*	121	186	63

on the first row of the mesh($e(3,6)$ and $e(9,12)$ in Fig. 7) to page 2, then we can get 0 crossings. For a 4-row mesh, in the same way, if we put the broken edges on the border of the mesh to page 2, we can get 0 crossings. For example, in Fig. 8(a), we can get 0 crossings by putting $e(4,8)$, $e(12,16)$, $e(15,19)$ and $e(7,11)$ to page 2, and in Fig. 8(b), by putting $e(4,8)$, $e(17,18)$ and $e(7,11)$ to page 2.
So we can get optimal solutions for 3- or 4-row meshes using the method above in linear time.

5.2 Optimal solution for Halin graphs

Halin graphs are edge minimal 3-connected, Hamiltonian, and in general have large numbers of Hamiltonian cycles[6]. We can find a Hamiltonian cycle in a Halin graph in polynomial time. In Fig. 11, the solid edges form a Hamiltonian cycle.
We have the following theorem about the two-page crossing number of Halin graphs:

Theorem 8. *For a Halin graph H, $\nu_2(H) = 0$.*

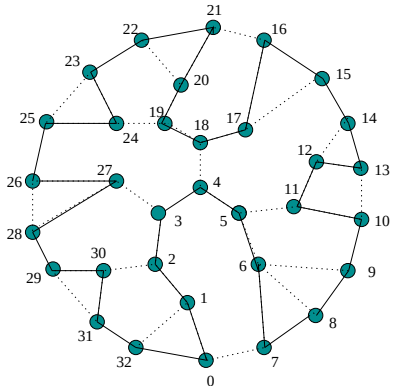


Fig. 11. Optimal order of a Halin graph

Proof. Every Halin graph is planar. Every Halin graph has a hamiltonian cycle[6], thus every Halin graph is a subhamiltonian planar graph. According to Lemma 1, for a Halin graph H, $\nu_2(H) = 0$.

We will now give an effective construction for two-page drawings of Halin graphs with no crossings. Given a Halin graph $H = T \bigcup C$, with l leaves on C, we can cut all c-edges of C from H to get the tree T. We call a vertex that is not a leaf in T as in-vertex. There is only one path between any pair of vertices in a tree, so we can get paths between any pair of neighboring leaves on C. We assume the order of leaves on C as $\{u_0, u_1, ..., u_{l-1}\}$, and denote the path between u_i and u_{i+1} as $P(u_i, u_{i+1})$. Then $V = P(u_0, u_1) \bigcup P(u_1, u_2) \bigcup ... \bigcup P(u_{l-2}, u_{l-1}) \bigcup P(u_{l-1}, u_0)$, and there exists a set of independent paths above, which cover all in-vertices and some leaves that are the ends of the paths, where any two independent paths do not have a common vertex. A point is a central point of a graph if the eccentricity of the point equals the graph radius. Assuming a central point of a tree is the root of the tree (vertex 4 in Figure 11), we can get an embedding with no crossings for a Halin graph by the following steps (initially all edges are in page 1).

- 1. Calculate all paths of neighbouring leaves on C, $P(u_0, u_1)$, $P(u_1, u_2)$, ..., $P(u_{l-2}, u_{l-1})$, and $P(u_{l-1}, u_0)$.
- 2. Find the longest path $P(u_i, u_{i+1})$=$u_i, v_0, v_1, ..., v_{ni-1}, u_{i+1}$ that passes through the root of T.
- 3. Remove all vertices on the path from T. This will separate the tree into several subtrees. The root of each subtree is the vertex that connects to an in-vertex on the removed path. A subtree includes at least one in-vertex and two or more leaves unless the root of the subtree is just a leaf.
- 4. For all subtrees that are not a single leaf, repeat step 2, 3.
- 5. For all independent paths found, replace the c-edge (u_i, u_{i+1}) with $P(u_i, u_{i+1})$ on C, namely, insert the v_0, v_1, ..., v_{ni-1} between u_i and u_{i+1} on C to form a Hamiltonian cycle(see Fig. 12).

– 6. Put the replaced c-edges to page 2.

Proof. There are three possible types of intersecting edge pairs: pair of t-edges, pair of c-edges, and t-edge with c-edge. However,

– During the procedure above, the order of leaves on C is not changed, so any pair of c-edges do not intersect.
– According to the construction of the Hamiltonian cycle, each path comes from a separated subtree, the root of which is connected to a path found previously. In other words, all in-vertices on an independent path connect with at most one in-vertex on another path i.e. there is at most one edge to connect two independent paths. Furthermore, the inserted in-vertices on an independent path are kept in the order of paths, so any pair of t-edges do not intersect.
– the replaced c-edges are put to page 2, so no c-edge intersects with t-edges.

Therefore, we can get an embedding with no crossings for a Halin graph with this method.

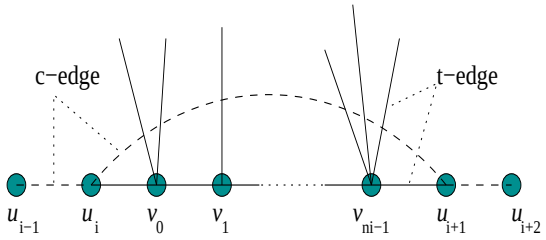


Fig. 12. A path on the Hamilton cycle of a Halin graph

6 Conclusion

For an arbitrary graph, two-page algorithms based on a good one-page algorithm are very useful. From our experiments, the two-page crossing number is not only related to the one-page drawing, but also related to the edge distribution. We investigated combining two possible one-page drawing algorithms (AVSDF+ from [10] and BB+ from [2]) with various strategies for edge assignment according to slope (SLOPE), edge length (LEN) and crossing number (CRS). For Rome graphs, AVSDF+_CRS and BB+_CRS get best results. For RCGs, BB+_CRS gets best results. For Halin graphs and Cartesian graphs, $C_m \times C_n$, AVSDF+_CRS gets best results. For circulant graphs, $C_{kn}(1, n)$, BB+_LEN and BB+_CRS get best results.
Based on a fixed optimal order of vertices, for circulant graphs LEN and CRS get best results, and for complete graphs and complete p-partite graph, SLOPE gets best results. AVSDF_EP and SLOPE are overshadowed by the LEN strategy and the CRS strategy for most graphs.
Different two-page preprocessing steps make the two-page strategies have different effects on solutions. Based on AVSDF_EP, algorithm LEN achieved much greater improvement to the results of

AVSDF_EP than algorithm CRS for RND_BUP graphs. In contrast, based on AVSDF+_SLOPE, algorithm CRS achieve greater improvement to results of AVSDF+_SLOPE than algorithm LEN on the same RND_BUP graphs. For the two-page strategy LEN, the best preprocessing is AVSDF_EP, while for two-page strategy CRS, the best preprocessing is the one-page algorithm AVSDF+ without two-page preprocessing.
For RCGs with different densities, AVSDF+_SLOPE achieves better results than BB+_SLOPE, and the results by them become closer with the rise of density. No matter whether AVSDF+ or BB+ algorithm precedes the two-page algorithms and what density graphs are tested, CRS gets slightly better results than LEN does and both are significantly better than SLOPE.
With our two-page strategies based on a one-page algorithm, the best results of some circulant graphs are even better than the optimal values based on fixed orders of vertices obtained in [5].
We present some test results for complete p-partite graphs, complete graphs, 3-row meshes, 4-row meshes and cartesian graphs, from which we present some theorems and conjectures. The conjectures are supported by our experiments. We also present direct methods to get optimal solutions for 3-row meshes, 4-row meshes and Halin graphs.

References

1. Adamson, J., Richter B.R.: Arrangements, circular arrangements and the crossing number of $C_7 \times C_n$, *Journal of Combinatorial Theory, Series B* **90** (2004) pp. 21-39.
2. Baur, M., Brandes, U., Crossing Reduction in Circular Layouts, *Proc. 30th Intl. Workshop Graph-Theoretic Concepts in Computer-Science (WG '04)*, LNCS **3353**, Springer, (2004), pp. 332-343.
3. Bernhart, F., Kainen, P., The book thickness of a graph, *Journal of Combinatorial Theory, Ser. B.*, **27** (1979), pp. 320-331.
4. Boesch, F., Tindell, R., Circulants and their connectivities, *Journal of Graph Theory* **8**, (1984), pp. 487-499.
5. Cimikowski, Robert: Algorithms for the fixed linear crossing number problem, *Discrete Applied Mathematics* **122** (2002), pp. 93-115.
6. Cornuéjols, G.,Naddef,D., Pulleyblank, W. R., Halin Graphs and the Travelling Salesman Problem. *Math. Prog.* **16** (1983) pp. 287-294.
7. Fulek, R, He, H, Sýkora, O., Vrt'o, I., Outerplanar Crossing Numbers of 3-Row Meshes, Halin Graphs and Complete p-Partite Graphs, *Proc. SOFSEM'05*, LNCS **3381**, Springer, (2005), pp. 376-379.
8. Glebsky, L.Y., Salazar, G., The conjecture $cr(C_m \times C_n) = (m-2)n$ is true for all but finitely n, for each m, *Journal of Graph Theory* **47** (2004), pp. 53-72.
9. Guy, R. K., Crossing number of graphs, *in* Y. Alavi, D. R. Lick and A. T. White, eds, *Graph Theory and Applications: Proc. of the Conference at Western Michigan University, Kalamazoo, Mich.*, New York: Springer-Verlag, (1972), pp. 111-124.
10. He, H, Sýkora, O, New Circular Drawing Algorihtms, *Proc. ITAT'04*, (2004).
11. Kainen, P.C., The book thickness of a graph II, *Congressus Numerantium* **71** (1990), pp. 121-132.
12. Lin, X.;Yang, Y.; Lü, J.; Hao, X.: The Crossing Number of $C(mk; \{1, k\})$, *Graphs and Combinatorics* **21** (2005), no. 1, pp. 89-96.
13. Masuda, S., Kashiwabara, T., Nakajima, K., Fujisawa, T., On the NP-completeness of a computer network layout problem, in: Proc. *IEEE Intl. Symposium on Circuits and Systems 1987*, IEEE Computer Society Press, Los Alamitos, (1987), pp. 292-295.
14. Masuda, S., Kashiwabara, T., Nakajima, K., Fujisawa, T., Crossing minimization in linear embeddings of graphs, *IEEE Trans. Comput.* **39** (1990), pp. 124-127.
15. Mäkinen, E., On circular layouts, *International Journal of Computer Mathematics* **24** (1988), pp. 29-37.
16. Melikov, A.N., Koreičik, V.M., Tiščenko, V.A., Minimization of the number of intersections of edges of a graph (Russian), *Vyčisl. Sistemy Vyp.* **47** (1971), pp. 32-40.
17. Nicholson, T.A.J., Permutation procedure for minimazing the number of crossings in a network, *Proc. Inst. Elec. Engnrs.* **115** (1968), pp. 21-26.

18. Shahrokhi, F., Sýkora, O., Székely, L.A., Vrt'o, I.: The book crossing number of a graph, *Journal of Graph Theory* **21** (1996), pp. 413–424.
19. Shahrokhi, F., Sýkora, O., Székely, L.A., Vrt'o, I., Towards the theory of convex crossing numbers, *Towards a Theory of Geometric Graphs*, ed. J. Pach, Contemporary Mathematics, 342 American Mathematical Society, Providence 2004, pp. 249-258.
20. Shahrokhi, F., Sýkora, O., Székely, L.A., Vrt'o, I.: Intersection of Curves and Crossing Number of $C_m \times C_n$ on Surfaces, *Descrete Comput. Geom.* **19** (1998), No. 2, pp. 237-247.
21. Six, J.M., Tollis, I.G., Circular drawings of biconnected graphs, in: *ALENEX'99*, LNCS **1619**, Springer, (1999), pp. 57-73.
22. Winterbach, W., The crossing number of a graph in the plane, *Master Thesis*, Dept. Appl. Math., University of Stellenbosch, SA, 2005, p. 334.
23. Yannakakis, M.: Four page are neccessary and suffcient for planar graphs(extended abstract). *ACM* (1986), pp. 104-108.
24. GDToolkit: http://www.dia.uniroma3.it/∼gdt/