

# A Genetic Algorithm for Finding the Pagenumber of Interconnection Networks

Nidhi Kapoor

*Nortel Networks, Ottawa, Ontario, Canada*

Mark Russell

*Department of Combinatorics and Optimization, University of Waterloo,  
Waterloo, N2L 3G1 Ontario, Canada*

Ivan Stojmenovic

*SITE, University of Ottawa, Ottawa, K1N 6N5 Ontario, Canada; and  
DISCA, IIMAS, UNAM, Ciudad Universitaria, Coyoacan, Mexico D.F. 04510, Mexico  
E-mail: [ivan@site.uottawa.ca](mailto:ivan@site.uottawa.ca)*

and

Albert Y. Zomaya

*Department of Electrical and Electronic Engineering, University of Western Australia,  
Perth, Western Australia 6907, Australia*

Received August 22, 2000; revised July 12, 2001; accepted August 8, 2001

---

A “book-embedding” of a graph  $G$  comprises embedding the graph’s nodes along the spine of a book and embedding the edges on the pages so that the edges embedded on the same page do not intersect. This is also referred to as the page model. The “pagenumber” of a graph is the thickness of the smallest (in number of pages) book into which  $G$  can be embedded. The problem has been studied only for some specific kind of graphs. The pagenumber problem is known to be NP-complete, even if the order of nodes on the spine is fixed. Using genetic algorithms, we describe the first algorithm for solving the pagenumber problem that can be applied on arbitrary graphs. Experimental results for several kinds of graphs are obtained. We were particularly interested in graphs that correspond to some well-known interconnection networks (such as hypercubes and meshes). We also introduced and experimented with 2-D pagenumber model for embedding graphs. © 2002

Elsevier Science (USA)

## 1. INTRODUCTION

A *book* consists of a spine and a set of *pages* (each a half-plane with the spine as boundary). A *layout* is a linear ordering or permutation of the nodes of an undirected graph  $G = (V, E)$ . A *book embedding* of  $G$  consists of a layout along the spine of a book and an assignment of the edges to pages in such a way that all edges assigned to a page can be drawn without crossing. The minimum number of pages in which a graph can be embedded is its *pagenumber*. Figure 1 shows a four-vertex graph and a corresponding two-page embedding.

Figure 2 shows a one-page embedding of the same graph using a different layout on the spine of the book.

The permutation (1243) yields a better (smaller) pagenumber than the permutation (1234) and is in fact an optimal layout for the graph in Fig. 1. The pagenumber problem has a number of application areas, as follows.

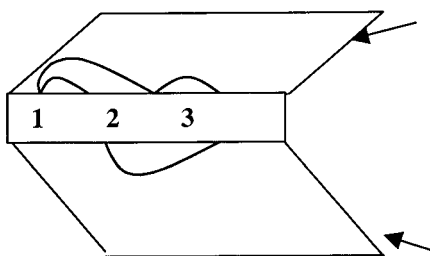
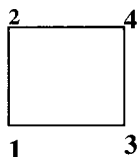
### *Direct Interconnection Networks*

The pagenumber problem can be used to obtain the optimal layout for processors and wires in the VLSI implementation of interconnection networks. The nodes of the graph correspond to processors and the edges correspond to links between processors.

### *Fault-Tolerant Processor Arrays*

Another application is in fault tolerant VLSI design [3, 4]. Processors are arranged on a line and bundles of wires with embedded switches run parallel to the line of processors. Each bundle is capable of implementing a hardware stack of connections among processors. Each connection occurs on exactly one hardware stack (bundle). The line of processors is scanned from left to right and suppose that a good processor  $u$  wants to communicate with another processor  $v$  to its right. Then, at  $u$ , the connection  $(u, v)$  is pushed into one of the stack bundles, that is,  $(u, v)$  occupies the bottom of the stack bundle, while the other connections that are currently in this bundle are shifted up one place. At  $v$ , the connection  $(u, v)$  is popped from the stack. The problem is to realize the desired interconnection graph using the minimum number of stack bundles. This is equivalent to the problem of

page 1



page 2

FIG. 1. Two-page embedding of a square.

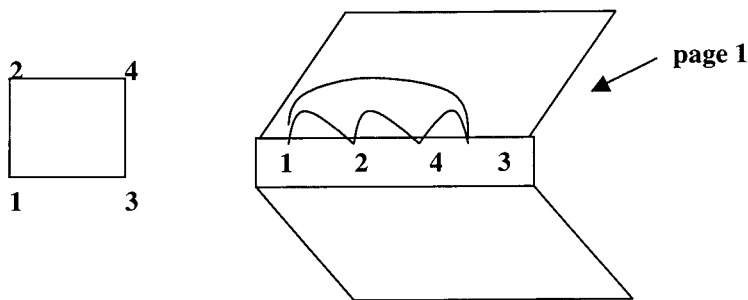


FIG. 2. One-page embedding of a square.

embedding the graph in a book with the minimum number of pages. The pages correspond to bundles. The constraint that edges on the same page do not intersect corresponds to the LIFO property of stacks.

### Sorting with Parallel Stacks

Even and Itai [5] studied the problem of how to realize fixed permutations of  $\{1, \dots, n\}$  with noncommunicating stacks. Initially, each number is pushed in the order 1 to  $n$ , onto any one of the stacks. Once all of the numbers are on stacks, the stacks are popped to form the permutation. The problem can be formulated as follows. Consider the bipartite graph  $G_n$  with vertices  $\{a_1, \dots, a_n, b_1, \dots, b_n\}$  and edges connecting each  $a_i$  to  $b_i$ . The problem of realizing the permutation  $\pi$  on  $\{1, \dots, n\}$  with  $k$  parallel stacks is equivalent to embedding  $G_n$  in a  $k$ -page book, with its vertices embedded in the order  $a_1, \dots, a_n, b_{\pi(1)}, \dots, b_{\pi(n)}$ .

### Single-Row Routing

In an attempt to simplify the problem of routing multilayer printed circuit boards (PCBs). So [24] decomposed the problem in the following way. The circuit elements are arranged in a regular grid, with wiring channels separating rows and columns of elements. The circuit's net lists are then decomposed so that every net connects elements in a single row or in a single column. The PCB can now be routed by routing each of its rows and each of its columns independently. The variant of this scenario that does not allow a net to run from the top of a row around to its bottom nor to change layers en route [23] corresponds directly to the embedding problem applied to small-valence graphs.

### Ordered Sets

In some instances, the underlying structure may be an ordered set [20]. For example, the vertices of the graph may represent steps in a calculation and the edges may represent necessary inputs for that calculation to be executed. If the calculation is done on a machine with a single processor, then each page represents a stack, and the edges indicate the order in which the partial results are entered and removed from the stack. The pagenumber is the smallest number of stacks required to store data in order for the calculation to be carried out.

The book embedding problem has been studied for a variety of graphs relating to common applications. We shall review most important results from literature. In 1975, Berhart and Kainen [2] showed that the pagewidth is  $\leq 1$  if and only if graph is outerplanar (nodes in an outerplanar graph can be placed on the circumference of the circle such that no chords, corresponding to edges, of the circle intersect). They also showed that the pagewidth is  $\leq 2$  if and only if the graph is subgraph of a Hamiltonian planar graph (examples are square grid and X-tree, which is a complete binary tree with additional edges going across each level in left-to-right order). Recall that a planar graph is a graph that can be drawn such that none of the edges intersect, while a Hamiltonian graph is a graph that has a closed path that passes through every vertex exactly once. Yannakakis [28] proved that four pages are sufficient (and sometimes necessary) for embedding an arbitrary planar graph.

A  $d$ -dimensional hypercube has all binary strings as vertices, with two vertices connected if and only if their strings differ in only one bit. An elegant embedding (using Gray codes) of  $d$ -dimensional hypercube using  $d-1$  pages was described in [4]. A complete graph with  $n$  vertices can be embedded into  $n/2$  pages [4]. Games [6] proposed an optimal 3-page book embedding for the FFT (Fast Fourier Transforms), Benes and barrel shifter graphs. Obrenic [21] presented a construction for embedding deBruijn and shuffle-exchange graphs in five pages, while the minimal required number of pages is 3. Definitions of mentioned graphs can be found in [27].

The *bandwidth* of a layout is the length of the longest edge embedded in the layout. The bandwidth of a graph is the smallest bandwidth of any of its layouts. Figure 3 shows a graph and a corresponding layout and embedding. The longest edge is (1, 4), hence the bandwidth for the graph with layout = (1234) is 3.

Swaminathan *et al.* [25] showed that any graph with bandwidth  $k$  can be embedded in  $k-1$  pages using a transformed layout. Since we can determine the bandwidth of a given layout easily in  $O(n^2)$  time (by examining each edge), the effectiveness of genetic algorithms on the bandwidth problem has been investigated in conjunction with the investigation of the pagewidth problem. Unfortunately, the results of this method were not encouraging. This is due to the fact that the minimum pagewidth of a graph is often much smaller than just one less than the minimum bandwidth. For example, the minimum pagewidth of a complete graph is  $\lfloor n/2 \rfloor$  [4], while the minimum bandwidth is  $n-1$ . Next, the minimum pagewidth of any outerplanar graph is 1 [2], while the minimum bandwidth of an outerplanar graph in which one node is adjacent to all of the other nodes is  $\lfloor n/2 \rfloor$ . Gavaille and Hanusse [7] studied the compact shortest path routing tables on

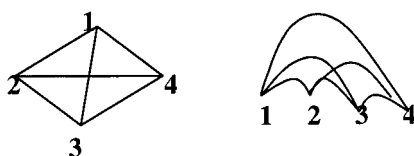


FIG. 3. Graph and its embedding.

weighted graphs with  $n$  nodes and pagenumber  $k$ . In [8] they showed an information-theoretic lower bound of  $kn \log(kn)$  on the minimum number of bits to represent an unlabeled  $n$ -node graph of pagenumber  $k$ . Sharokhi and Shi [26] studied the pagenumber problem for  $t$ -partite graphs.

The pagenumber is a very hard problem. It remains a difficult problem even when the layout is fixed, since determining if a given layout admits a  $k$ -page book embedding is NP-Complete [9]. This makes it difficult to evaluate the layouts in the genetic algorithm since we are trying to find a layout that gives a minimal embedding, but given a layout of the graph there is no guarantee that a particular embedding is minimal. This difficulty in evaluation is also the motivation behind solving a related graph problem (minimal bandwidth) using the genetic algorithm and mapping the solution back to a pagenumber solution. Several different embedding techniques were tested and are discussed in this paper. To test the genetic algorithm, we ran it on graphs with known lower bounds or known minimal pagenumbers.

Section 2 describes mutation and crossover techniques on permutations which are applied in the genetic algorithm in this paper. Section 3 describes our genetic algorithm for pagenumber problem, including the fitness calculation, edge embedding, and edge ordering. Section 4 introduces a new 2-D model for embedding graphs, with several variants. Experimental results and conclusions are given at the end.

## 2. GENETIC ALGORITHMS

The Genetic Algorithm refers to a model introduced and investigated by John Holland [10] and his students. It is a family of computational models whose premise is to find a good solution to a problem by simulating the evolution of a population of individual solutions. These algorithms encode a potential solution to a specific problem on a simple chromosome-like data structure and apply recombination operators to create new solutions while preserving critical information. A genetic algorithm is comprised of the following steps [19]:

1. *Create an initial population of solutions.* This can be done either randomly or heuristically; we have tested the generation of initial solutions using both a random and a heuristic method.
2. *Evaluate each solution in the population to determine its fitness (i.e., how good it is).* In our case, each individual's fitness is simply the pagenumber (or bandwidth) of the layout's embedding. We have also considered some improved fitness functions, taking edge intersections counts on each page into account.
3. *If the termination criterion is met, terminate the algorithm.* We decided to terminate if the best fitness has not been surpassed for a set number of generations. When the algorithm is terminated, the best individual from all generations provides us with our (hopefully optimal or near-optimal) solution.
4. *Produce a new population by evolving the current population, and return to step 2.*

There are two steps to evolving the population:

(i) *Select the individuals who will produce the new population.* Reproductive opportunities are allocated in such a way that chromosomes that represent better solutions are given more chances to “reproduce” than chromosomes that represent poorer solutions.

(ii) *Create the new population by applying genetic operators to the selected individuals.* There are two classes of genetic operators: mutations and crossovers. Generally, a set percentage of individuals in each successive population are produced through mutation and a set percentage are produced through crossover. Some mutation and crossover operators are outlined in [11, 15, 19] as follows.

Mutations create a child from a single parent by randomly changing some aspect of the parent (in our case, changing the layout). Mutations are usually only used to create a small percentage of the new population, as they are not intended to find a better solution; their purpose is to ensure that breadth of search is maintained. We used the following mutation operators in our experiments:

*Swap:* Select two random elements and swap them.

*Move Block:* Select a random block of elements and move it to a random position.

*Swap Blocks:* Select two random blocks and swap them.

*Insert:* Select a random element and insert it at a random position.

*Two-Opt:* Select a random block of elements and reverse the elements in that block.

*Scramble:* Select a random block of elements and randomly reorder the elements in it.

Crossovers create a child from two parents and usually comprise the vast majority of reproductions. Crossovers attempt to create better solutions by mixing the parent solutions while carrying over to the children properties found in the parents. Different crossover operators maintain different properties. We used the following crossover operators in our experiments.

#### *PMX (Partially Matched Crossover)*

Given parent permutations  $P_1$  and  $P_2$ , two child permutations are created by dividing  $P_1$  and  $P_2$  into sections that define mappings between one another (marked by ||), exchanging one section, and then filling the rest of each permutation by replacing corresponding elements. For example, let

$$P_1 = (jhd \parallel efi \parallel gcba) \quad P_2 = (hge \parallel bcj \parallel iadf).$$

Using the mid-sections of  $P_1$  and  $P_2$  ( $efi$  and  $bcj$ , respectively) to define the mapping, we get:  $P'_1 = (ihd \parallel bcj \parallel gfea)$ ,  $P'_2 = (hgb \parallel efi \parallel jadc)$ .

*CX (Cycle Crossover)*

Given parent permutations  $P_1$  and  $P_2$ , two child permutations are created by forming a cycle between  $P_1$  and  $P_2$ . Consider the permutations

$$P_1 = (jhdefigcba) \quad \text{and} \quad P_2 = (hgebcjiadf).$$

Starting from the first element in  $P_1$ , we see that  $j$  in  $P_1$  maps to  $h$  in  $P_2$ ,  $h$  in  $P_1$  maps to  $g$  in  $P_2$ ,  $g$  in  $P_1$  maps to  $i$  in  $P_2$ , and  $i$  in  $P_1$  maps to  $j$  in  $P_2$ , completing the cycle. The elements in the cycle from  $P_1$  are placed in the child, producing

$$P'_1 = (jh***ig***).$$

The empty slots (\*) are filled in by the elements of  $P_2$  at the corresponding positions. Thus,

$$P'_1 = (jhebcigadf)$$

Similarly,  $P'_2 = (hgdefjicba)$ .

*ER (Edge Recombination Crossover)*

Given parent permutations  $P_1$  and  $P_2$ , two child permutations are formed using the node adjacencies of  $P_1$  and  $P_2$ . The first element in the child is set as the first element in  $P_1$  or  $P_2$ . Each subsequent node is selected as the adjacent node with the smallest number of active edges (an edge is active if the element it points to has not yet been placed in the layout). In the case of a tie, the node is randomly selected from the tied elements. Consider the permutations  $P_1 = (cfbade)$  and  $P_2 = (ebacfd)$ . Examining adjacent nodes, we get:  $b, b, c, d$  are adjacent to node  $a$ ;  $a, a, e, f$  are adjacent to node  $b$ ;  $a, e, f, f$  to  $c$ ;  $a, e, e, f$  to  $d$ ;  $b, c, d, d$  to  $e$ ; and  $b, c, c, d$  to  $f$ . A possible child is  $P'_1 = (cfdabe)$ .

*OX (Order Crossover)*

Given parent permutations  $P_1$  and  $P_2$ , two child permutations are created by selecting two cut points (||) in  $P_1$  and  $P_2$ , copying the elements between the cut points into  $P'_1$  and  $P'_2$  and filling the remainder of the permutation with elements not yet used from the alternate parent, starting after the second cut point. Consider the permutations

$$P_1 = (jhd \parallel efi \parallel gcba)$$

$$P_2 = (hge \parallel bcj \parallel iadf).$$

To create the first child, we copy the elements between the cut points from the first parent:  $P'_1 = (** \parallel efi \parallel ***)$ . We then fill in the rest of the child starting from the second cut point with the elements of  $P_2$ :  $iadf hge bcj$ , skipping elements that already appear in the child:  $P'_1 = (bcj \parallel efi \parallel adhg)$ . Similarly,  $P'_2 = (efi \parallel bcj \parallel gahd)$ .

### 3. ALGORITHMIC DETAILS

#### 3.1. Initialization

We used an initialization heuristic that attempts to create a layout in which there are edges between all adjacent nodes. Layouts created in this manner are generally better than randomly created layouts because they have fewer edges which can cross other edges (edges between adjacent nodes can never cross another edges), and so generally they require fewer pages to embed. Genetic algorithms run using the initialization heuristic found three times as many optimal solutions as those run using random initialization.

#### 3.2. Edge Embedding

Initially, we used a very simple edge embedding algorithm that checks the edge to be embedded against all edges on each page until a page on which it will fit is found. We then designed a more efficient algorithm for embedding edges into a book. The new algorithm is based on a layout-splitting idea as follows. Whenever an edge  $uv$  between non-adjacent nodes is embedded on a page, that page's layout can be divided into two separate layouts that will accept any remaining edges:  $L..uv..R$  (which we will denote  $O$ ) and  $u..v$  (which we will denote  $I$ ), where  $L$  and  $R$  are, respectively, the leftmost and rightmost nodes in the layout. Any edge connecting a node in  $O(I)$  to a node in  $I(O)$  will cross the edge  $uv$ , so all edges embedded on the page after  $uv$  may be embedded in either  $O$  or  $I$ . For the algorithm, instead of physically dividing the layout, we place bounds on the nodes between  $u$  and  $v$  when an edge  $uv$  is embedded. The algorithm, then, requires two arrays for each page: one contains the index of the highest neighbor allowed for each node (initialized to  $n$ ), the other contains the index of the lowest neighbor allowed for each node (initialized to 0). Whenever an edge is added to a page, the lowest and highest neighbor indices for the nodes bounded by the new edge are updated (note that either none or both are updated after adding each new edge). When checking if an edge will fit in a page, only the highest and lowest neighbor indices of the nodes incident to the edge need to be checked, which can be done in constant time. If a node  $w$  is bounded by  $u..v$ ,  $uv$  is an edge, and the other end of an edge incident to  $w$  must be in the same interval  $u..v$ . A quick test comparing the old and new embedding algorithms showed that the new algorithm took only a quarter of the time taken by the old algorithm to embed complete graph layouts with 20 nodes and 10 or more pages.

#### 3.3. Edge Ordering

The number of obtained pages greatly depends on the order in which edges are inserted in the book. Our solution space is therefore composed from two permutations: permutation of nodes  $N$  and a permutation (order) of edges  $E$ . The genetic algorithm may be applied on the pair  $(N, E)$  as a solution, meaning that mutation and crossover operators may apply as well. Since the number of edges is possibly

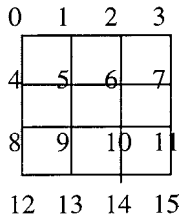


FIG. 4. 4 × 4 grid.

quadratic in number of nodes, the solution space that may be imposed by  $E$  may become too large to allow an efficient search. We assume that the order of nodes is more important than the order of including edges, and therefore our genetic algorithm applies regular mutation and crossover on  $N$  only. For example, consider the (4x4) grid in Fig. 4.

Using the layout (0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15) and selecting the edges in lexicographic order, the pagenumber is 4 (as shown in Fig. 5). A different order may lead to different pagenumber.

For each permutation of nodes tested, several fixed number of edge orders are tried (lexicographic, random, the order of best solution so far, evolve the best order of the best solution). Since it is not practical to test all edge orders, and embedding with only one or two provides no guarantee of closeness to optimality, we developed a new edge ordering algorithm. Our experiments show that it performs as well as or better than the others we tested, and it is optimal in the case of complete graphs. The idea behind the new ordering algorithm is that certain edges are more likely to cross with more edges than others. More specifically, the maximum number of edges an edge may cross with is  $(c-1) * (n-c-1)$ , where  $c$  is the length of the edge. This number is maximized when  $c = \lfloor n/2 \rfloor$  or  $\lceil n/2 \rceil$ , so edges with length  $\lfloor n/2 \rfloor$  are embedded first, followed by edges with length  $\lceil n/2 \rceil$  (if different from  $\lfloor n/2 \rfloor$ ),  $\lfloor n/2 \rfloor - 1$ ,  $\lceil n/2 \rceil + 1$ ,  $\lfloor n/2 \rfloor - 2$ ,  $\lceil n/2 \rceil + 2$ , etc., spiraling out until edges of length 1 and  $n-1$  are embedded (not necessarily in that order). Edges of a given length are embedded in lexicographic order. One would expect it to perform similarly well for near-complete graphs; indeed, it performed significantly better than other ordering methods when dense graphs were being tested.

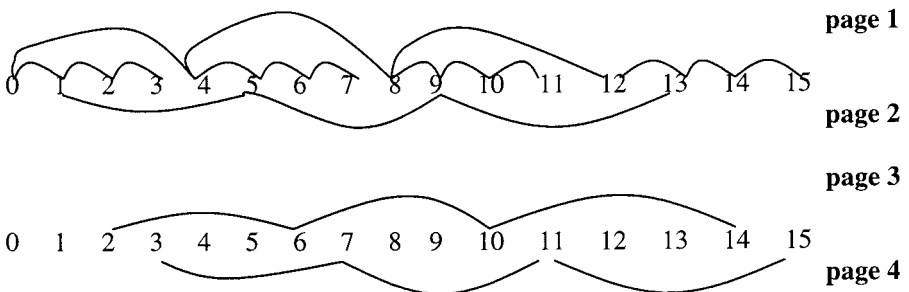


FIG. 5. 4-page embedding.

### 3.4. Fitness Calculation

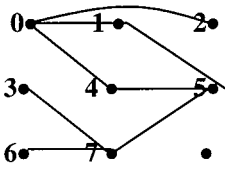
During testing, it became apparent that no significant evolution was occurring for several graph types. One possible reason for this was that the fitness function used was not effectively capturing the “goodness” of sub-optimal solutions—although two solutions with the same pagenumber are equally “good” in the sense that they use the same number of pages, one may in fact be closer to a solution with a smaller pagenumber. This is something that should be recognized by the genetic algorithm, so we tested a few different techniques for fitness calculation.

The original technique simply used pagenumber to determine the allocation of reproduction opportunities. A simple alternative is based on the idea that an embedding with a page that contains only a few edges is closer to having a lower pagenumber than an embedding with the same pagenumber whose pages all have many edges. A simple fitness function based on this is ( $n * \text{pagenumber} + \text{number\_of\_edges\_in\_smallest\_page}$ ); this function ensures that pagenumber still dominates the fitness function, so a graph with smaller pagenumber is always be considered better than a graph with larger pagenumber. We also tested a more complex fitness function based on the same principle that takes into account the distribution of edges throughout all of the pages in the book. The fitness function for this method is ( $0 * \text{number of edges in 1st page} + 1 * \text{number of edges in 2nd page} + \dots + (k-1) * \text{number of edges in } k\text{th page}$ ). We also used a fitness function that orders the pages: ( $0 * \text{number of edges in largest page} + 1 * \text{second largest page} + \dots + (k-1) * \text{smallest page}$ ). A fitness function using increased factors for the edge weights (coefficients changed from  $0, 1, 2, \dots, k-1$  to  $1, 2, 4, \dots, 2^{k-1}$ ) caused a large decrease in the number of best solutions. Another function that had slightly different coefficients ( $1, 2, 3, \dots, k$  instead of  $0, 1, 2, \dots, k-1$ ) gave results similar to the other functions.

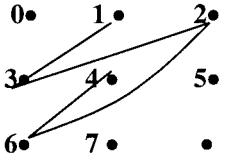
Since all of these techniques depend on the embedding which may not properly represent the properties of a layout, we developed a final fitness function that calculates fitness independently from pagenumber or page distribution. The new fitness is calculated as the number of edge crosses for a tested permutation of nodes, assuming all nodes and edges are embedded on a single page. This method performs as well as or better than any of the other methods tested, with the additional benefit of not depending on imperfect embedding techniques, and so it is an effective and less biased measure of a layout’s fitness.

## 4. 2-DIMENSIONAL PAGENUMBER MODEL

In this section, we introduce a new 2-D model for embedding graphs. The nodes of the graph are placed anywhere on the plane. Edges of the graph are then added by joining corresponding nodes in an arbitrary way. Each edge is placed on a page, such that each page is a planar graph. The problem is to determine the minimum number of pages needed to embed the graph. In order to design an algorithm for an arbitrary graph, it is necessary to include some restrictions on nodes and edges. We introduce square and rook models of node placements in 2-D, and straight-line and horizontal-vertical model of edge drawings.



**Page 1:** Embedded edges are (0,1), (0,2), (0,4), (1,5), (3,7), (4,5), (5,7), (6,7)



**Page 2:** Embedded edges are (1,3), (2,3), (2,6), (4,6)

**FIG. 6.** 2-page embedding of cube.

### 4.1. Square Model

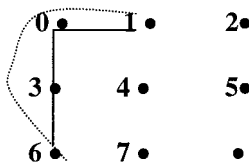
The first layout model that is considered is the 2-D square model, where nodes are arranged in order to form a square grid. A graph with  $n$  nodes is embedded in a grid with  $\lfloor \sqrt{n} \rfloor$  rows and  $\lfloor \sqrt{n} \rfloor$  columns. Each node is placed on a separate grid point. Node placement is specified by their permutation. Nodes are placed in row major order, without skipping any position. For a given graph  $G(V, E)$ , we introduce two methods for embedding its edges.

*Straight-line method.* In this method, edges are embedded by joining nodes with straight-line segments. However, if a straight line passes through another node, a line that avoids these nodes replaces it.

As an example, consider the 2-D embedding of a cube, in two pages. Nodes are placed on a grid, and the edges are embedded as indicated.

*Horizontal-vertical method.* In this method, edges are embedded by using horizontal and vertical line segments only. However, in most cases there will be intermediate nodes that should be avoided. Thus they are replaced by nearby lines on one of two possible sides. For example, edge (1,6) in Fig. 7 consists of a horizontal segment 0-1 and a vertical segment 0-3-6, and is replaced by a dashed line outside (alternative inside line may also be used).

Figure 8 shows a two-dimensional embedding of a four-dimensional hypercube, which is an example for both the straight-line and horizontal-vertical methods.



**FIG. 7.** Horizontal-vertical method.

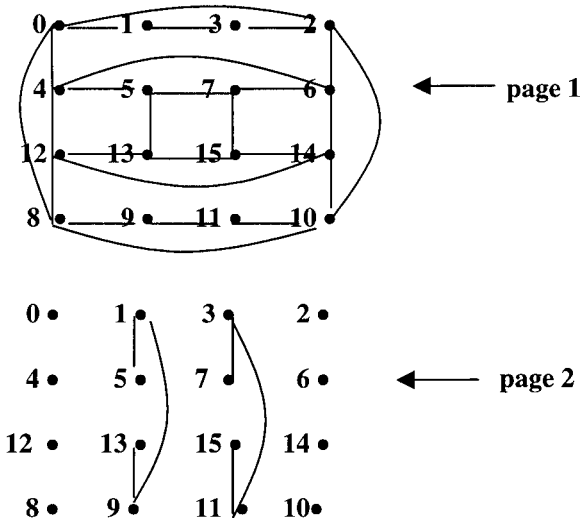


FIG. 8. 2-D embedding of a 4-D hypercube.

#### 4.2. Rook Model

Suppose that  $n$  nodes are placed in a plane. If any two nodes are on the same horizontal or vertical line, slightly rotate the plane so that this is no longer the case. This process does not affect the pagenumber. Thus we may assume that nodes are located on separate vertical and separate horizontal lines. We can further transform the embedding by translating nodes one by one into discrete horizontal or vertical line positions, respecting their mutual order in both directions. This transformation also does not affect pagenumber, since nonintersecting edges will be transformed into corresponding nonintersecting edges (if straight-line method is applied). Thus, position of nodes can be specified by two permutations, one for the horizontal and the other for the vertical direction. For example, Fig. 9 can be characterized by two permutations 62734051 and 02134657.

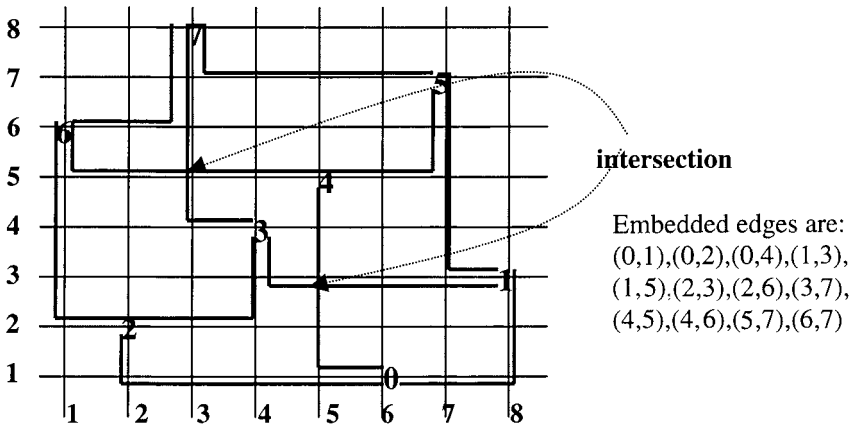


FIG. 9. Embedding of a cube with two edge intersections.

Edges can be inserted using the straight-line method or horizontal-vertical method. In both cases, in order to present a general algorithm, we decided to restrict the ways edges are embedded. For instance, Fig. 9 shows a cube embedding, with edges drawn horizontally and then vertically (horizontal-vertical method). We assume that edges are recorded as  $(a, b)$  with  $a < b$ . For example, edge  $(3, 7)$  in Fig. 9 is drawn horizontally, and then vertically. Edges can also be connected by drawing the vertical line first, and then the horizontal line. However, we have chosen the former method.

There are two edge intersections in Fig. 9, and thus two pages may be needed. Vertical or horizontal line segments having a common node,  $n$ , do not intersect, as each can be placed on a different track that originates from  $n$ . For example, the edges  $(6, 7)$ ,  $(3, 7)$  and  $(5, 7)$  in Fig. 9 have a common node 7.

## 5. EXPERIMENTS

We carried experiments for the following four cases of embedding graphs:

*Case 1.* 1-D model

*Case 2.* 2-D square model with the straight-line method of edge embedding

*Case 3.* 2-D rook model with the straight-line method of edge embedding

*Case 4.* 2-D rook model with the horizontal-vertical method of edge embedding

The solution space in cases 1 and 2 consist of a permutation of nodes  $N$  and an order of edges  $E$ , while cases 3 and 4 require a permutation of nodes  $N_X$  (in the  $X$  direction), a permutation of nodes  $N_Y$  (in the  $Y$  direction), and an order of edges  $E$ , in order to compute the pagenumber. The pair  $N = (N_X, N_Y)$  of two permutations of nodes  $N_X$  and  $N_Y$  is considered as one permutation, with crossover and mutation operators defined as follows. Mutations of  $N = (N_X, N_Y)$  are  $N' = (N'_X, N_Y)$  and  $N'' = (N_X, N'_Y)$ , where  $N'_X$  is a mutation of  $N_X$  and  $N'_Y$  is a mutation of  $N_Y$ . The crossover is defined similarly on two permutation pairs, and it is applied on the first only or second only permutation in each pair. The algorithm that searches for a page to place an edge remains the same for all four cases; however, the condition that determines whether or not two edges cross changes. Thus the main difference in algorithms are criteria for edge crossings, which are described in detail in the full version of this article [14].

Extensive tests were performed on the effectiveness of solving the pagenumber problem using a genetic algorithm. As already discussed, we developed several techniques during these tests that improved the performance of the algorithm. Table 1 shows the largest graphs for which the optimal pagenumber was found by the genetic algorithm. We have also tested a hill climbing and a modified hill climbing algorithm [14].

We discovered that we received the best results when using unusually high percentages of mutations and crossovers (20–40% and 60–80%, respectively, as opposed to the usual 1 and 25%), and in particular when using the insert mutation operator and the  $CX$  crossover operator, with  $OB$  and  $PB$  also performing reasonably well. For larger graph sizes (sizes where our  $GA$  did not produce optimal

**TABLE 1**  
**Largest Graph Sizes for Optimal Pagenumbers Found**

Graph type	Number of nodes	Number of edges	Pagenumber
Outerplanar	50	97	1
Square Grid, $d = 7$	49	84	2
$X$ -Tree, $d = 5$	31	56	2
$D$ -Ary Tree, $d = 3$	13	12	1
Hypercube, $d = 4$	16	32	3
DeBruijn, $k = 4, m = 2$	16	29	3
FFT, $d = 2$	32	48	3
Benes, $d = 2$	16	24	2
Barrel Shifter, $d = 3$	32	48	3

**TABLE 2**  
**Pagenumbers for 2-D Models**

Graph Number of nodes	Square Straight-line	Rook Straight-line	Rook Horizontal-Vertical
<b>Planar</b> 10	2	2	2
<b>Outerplanar</b> 8	2	1	1
<b>Square grid</b> 9	1	2	2
<b>Square grid</b> 16	1	4	3
<b><math>X</math>-tree</b> 15	2	2	2
<b>Binary tree</b> 15	2	2	2
<b>3-D hypercube</b> 8	2	2	1
<b>4-D hypercube</b> 16	3	4	3
<b>5-D hypercube</b> 32	5	8	6
<b>Complete graph</b> 6	3	3	2
<b>DeBruijn</b> 8	2	2	1

pagenumber), some graph types (e.g.,  $d$ -ary tree, hypercube, deBruijn, Benes) were amenable to evolution (meaning that the pagenumber of the last generation was smaller than the one found in the initial population). However, some graph types (e.g., FFT, barrel shifter) did not always show signs of significant evolution.

Although the bandwidth of graphs was explored in the testing, the emphasis remained on results which work on the pagenumber directly. Equivalent or better solutions were always found when running the genetic algorithm directly on the pagenumber problem rather than through the bandwidth problem.

Table 2 gives the best pagenumbers for 2-D models, obtained in our experiments. Master thesis of the first author [14] contains numerous detailed results for the 1-dimensional and three variants of 2-dimensional models, using several variations of GA, as described above, a hill climbing and a modified hill climbing algorithm. The results are available upon request.

## CONCLUSION

Using genetic algorithms, we described the first algorithm for solving the pagenumber problem that can be applied on arbitrary graphs. Therefore no fair comparison with other method is possible, since all existing methods are designed for particular kind of graphs. Experimental results for several kinds of graphs were obtained. We were particularly interested in graphs that correspond to some well-known interconnection networks (such as hypercubes and meshes).

Although much work has been done on the pagenumber genetic algorithm, there remain many ideas that could be tested for possible improvements:

- The edge ordering method works very well in practice. A proof that it always gives an optimal embedding for complete graphs may yield other useful properties.
- Further investigation into the effectiveness of the number-of-edge-crosses method of determining fitness could be done. In particular, how well would this evaluation method aid evolution in different genetic algorithm variants?
- Although the bandwidth problem has been shown to be not particularly helpful in solving the pagenumber problem on its own, perhaps there is some way in which it could be used in a cooperative manner with regular pagenumber layout determination to improve the process of finding an optimal layout.
- A better heuristic for initialization may improve results and place the layouts in a part of the search space more conducive to evolution.
- New genetic operators could be developed to take advantage of pagenumber-specific layout properties, again with the goal of improving evolution. In particular, a crossover operator that uses information on the set of edge crosses could be developed and tested.

The 2-dimensional pagenumber model is introduced in this paper, and no theoretical results exist in literature for that model. It is an open problem to find 2-D pagenumber under any of proposed variants for the same type of graphs for which 1-D pagenumber was determined in literature.

## ACKNOWLEDGMENTS

The authors are grateful to two referees for careful reading of the manuscript, which contributed to the improvement of its presentation. This research is partially supported by NSERC.

## REFERENCES

1. A. Nirwan and E. Hou, "Computational Intelligence for Optimization," Kluwer Academic, Dordrecht/Norwell, MA, 1996.
2. F. Bernhart and P. C. Kainen, The book thickness of a graph, *J. Combin. Theory Ser. B* **27** (1979), 320–331.
3. F. R. K. Chung, F. T. Leighton, and A. L. Rosenberg, Diogenes—A methodology for designing fault-tolerant processor arrays, in "13th Intl. Conference on Fault Tolerant Computing, 1983," pp. 26–32.
4. F. R. K. Chung, F. T. Leighton, and A. L. Rosenberg, Embedding graphs in books—a layout problem with applications to VLSI design, *SIAM J. Algebraic Discrete Methods* **8** (1987), 33–58.
5. S. Even and A. Itai, Queues, stacks, and graphs, in "Theory of Machines and Computations" (Z. Kohavi and A. Paz, Eds.), pp. 71–86, Academic Press, New York, 1971.
6. R. A. Games, Optimal book embeddings of the FFT, Benes, and barrel shifter networks, *Algorithmica* **1** (1986), 233–250.
7. C. Gavoille and N. Hanusse, Compact routing tables for graphs of bounded genus, in "Proc. Int. Coll. on Automata, Languages and Programming ICALP, 1999," pp. 351–360. [*Lecture Notes in Comput. Sci.* **1644**]
8. C. Gavoille and N. Hanusse, On compact encoding of pagenumber  $k$  graphs, *Discrete Math. Theoret. Comput. Sci.*, in press (January 2001).
9. M. R. Garey, D. S. Johnson, G. L. Miller, and C. H. Papadimitriou, The complexity of coloring circular arcs and chords, *SIAM J. Algebraic Discrete Methods* **1** (1980), 216–227.
10. J. Holland, "Adaptation in Natural and Artificial Systems," Univ. Michigan Press, Ann Arbor, MI, 1975.
11. T. Back, D. B. Fogel, and Z. Michalewics (Eds.), "Handbook of Evolutionary Computation," release 97/1, Oxford University Press, London, 1997.
12. L. S. Heath and S. Istrail, The pagenumber of genus  $g$  graphs is  $O(g)$ , *J. Assoc. Comput. Mach.* **39**(3) (July 1992), 479–501.
13. C. D. Keys, Graphs critical for maximum book thickness, *Pi Mu Epsilon J.* **6** (1975), 79–84.
14. N. Kapoor, "Pagenumber Problem," Master's thesis, SITE, University of Ottawa, 1999.
15. M. Mitchell, "An Introduction to Genetic Algorithms," MIT Press, Cambridge, MA, 1996.
16. S. M. Malitz, Genus  $g$  graphs have pagenumber  $O(\sqrt{g})$ , *J. Algorithms* **17** (July 1994), 85–109.
17. A. Moran and Y. Wolfstahl, One page book embedding under vertex-neighborhood constraints, *SIAM J. Discrete Math.* **3** (1990), 376–390.
18. D. J. Muder, M. L. Weaver, and D. B. West, Pagenumber of complete bipartite graphs, *J. Combin. Theory* **12** (1988), 469–489.
19. Z. Michalewics, "Genetic Algorithms + Data Structures = Evolution Programs," 3rd ed., Springer-Verlag, New York, 1996.
20. R. Nowakowski and A. Parker, Ordered sets, pagenumbers, and planarity, *Order* **6** (1989), 209–218.
21. B. Obrenic, Embedding deBruijn and shuffle-exchange graphs in 5 pages, *Proc. ACM SPAA* (1991), 137–146.
22. L. T. Ollmann, On the book thickness of various graphs, in "Proceedings, 4th S.E. Conference on Combinatorics, Graph Theory and Computing, 1973," p. 459.
23. R. Raghavan and S. Sahani, Single row routing, *IEEE Trans. Comput. C* **32** (1983), 209–220.

24. H. C. So, Some theoretical results on the routing of multilayer printed-wiring boards, in "IEEE Intl. Symp. On Circuits and Systems, 1974," pp. 296–303.
  25. R. Swaminathan, D. Giraraj, and D. K. Bhatia, The Pagenumber of the class of bandwidth- $k$  graph is  $k-1$ , *Inform. Process. Lett.* **55** (1995), 71–74.
  26. F. Shahrokhi and W. Shi, On crossing sets, disjoint sets and pagenumber, *J. Algorithms* **34**(1) (2000), 40–53.
  27. I. Stojmenovic, Direct interconnection networks, in "Parallel and Distributed Computing Handbook" (A. Y. Zomaya, Ed.), pp. 537–567, McGraw–Hill, New York, 1996.
  28. M. Yannakakis, Embedding planar graphs in four pages, *J. Comput. System Sci.* **38** (1989), 36–67.
- 

NIDHI KAPOOR KAMRA received the B.Sc. in computer science in 1997 in India and the master's degree in computer science at SITE, University of Ottawa, in 1999, on the subject of this paper. She has been working at Nortel Networks since graduation.

MARK RUSSELL received the B.Sc. in information and management systems at the University of Ottawa in 1999 and the master's degree in mathematics from the Combinatorics and Optimization Department at the University of Waterloo in 2001. He is currently working at Transport Dynamics, Princeton, New Jersey, U.S.A. His research interests are combinatorial optimization, graph and network algorithms, and matching theory.

IVAN STOJMENOVIC received the B.S. and M.S. in 1979 and 1983, respectively, from the University of Novi Sad and the Ph.D. in mathematics in 1985 from the University of Zagreb. He earned a third degree prize at the International Mathematics Olympiad for high school students in 1976. In 1980 he joined the Institute of Mathematics, University of Novi Sad (Yugoslavia). In fall of 1988, he joined the faculty of the Computer Science Department at the University of Ottawa, where currently he holds the position of a full professor in SITE. Since June 2000, he has frequently been in Mexico City, Mexico, as a researcher in DISCA, IIMAS, Universidad Nacional Autónoma de Mexico. He has published three books and over 150 different papers in journals and conferences. His research interests are wireless networks, parallel computing, multiple-valued logic, evolutionary computing, neural networks, combinatorial algorithms, computational geometry, and graph theory. He is currently a managing editor of *Multiple-Valued Logic*, an international journal, and an editor of the following journals: *Parallel Processing Letters*, *IASTED International Journal of Parallel and Distributed Systems*, and *Tangenta*.

ALBERT Y. ZOMAYA is a full professor in the Department of Electrical and Electronic Engineering at the University of Western Australia, where he also leads the Parallel Computing Research Laboratory. He received his Ph.D. from the Department of Automatic Control and Systems Engineering, Sheffield University, United Kingdom. Also, he has held visiting positions at Waterloo University and The University of Missouri-Rolla. He is the author/co-author of five books and more than 130 publications in technical journals, collaborative books, and conferences and the editor of three volumes and three conference proceedings. He is currently an associate editor for the *IEEE Transactions on Parallel and Distributed Systems*, the *Journal of Parallel Algorithms and Applications*, the *Journal of Interconnection Networks*, the *International Journal on Parallel and Distributed Systems and Networks*, the *Journal of Future Generation Computer Systems*, and the *International Journal of Foundations of Computer Science*. He is also the Founding Editor of the Wiley Book Series on Parallel and Distributed Computing. He is the Editor-in-Chief of the *Parallel and Distributed Computing Handbook* (McGraw–Hill, 1996). Professor Zomaya is the Chair of the IEEE Technical Committee on Parallel Processing (1999–Present). He is also a board member of the IFAC Committee on Algorithms and Architectures for Real-Time Control and serves on the executive board of the IEEE Task Force on Cluster Computing. He has been actively involved in the organization of national and international conferences. Professor Zomaya is a Fellow of the Institution of Electrical Engineers (U.K.) and a chartered engineer, a senior member of the IEEE, and a member of the ACM. He received the 1997 Edgeworth David Medal from the Royal Society of New South Wales for outstanding contributions to Australian science. In September 2000 he was awarded the IEEE Computer Society's Meritorious Service Award. His research interests are in the areas of high-performance computing, parallel and distributed algorithms, mobile computing, and networking.