SIAM J. ALG. DISC. METH.
Vol. 8, No. 2, April 1987

EMBEDDING OUTERPLANAR GRAPHS IN SMALL BOOKS*

LENWOOD S. HEATH†

Abstract. We investigate the problem of embedding graphs in books. A *book* is some number of half-planes (the *pages* of the book), which share a common line as boundary (the *spine* of the book). A *book embedding* of a graph embeds the vertices on the spine in some order and embeds each edge in some page so that in each page no two edges intersect. The *pagenumber* of a graph is the number of pages in a minimum-page embedding of the graph. The *pagewidth* of a book embedding is the maximum cutwidth of the embedding in any page. A practical application of book embeddings is in the realization of a fault-tolerant array of VLSI processors.

Our result is an $O(n \log n)$ time algorithm for embedding an n-vertex outerplanar graph with small pagewidth. The algorithm embeds any d-valent outerplanar graph in a two-page book with $O(d \log n)$ pagewidth. This result is optimal in pagewidth to within a constant factor for the class of outerplanar graphs. As there are trivalent outerplanar graphs that require $\Omega(n)$ pagewidth in any one-page embedding, the pagenumber of our embedding is exactly optimal for the stated pagewidth. The significance for VLSI design is that any outerplanar graph can be implemented in small area in a fault-tolerant fashion.

Key words. outerplanar graphs, book embedding, algorithm, hamiltonian cycles

AMS(MOS) subject classifications. 05C45, 68Q35, 94C15

1. The problem.

We study embeddings of graphs in structures called *books*. A *book* consists of a *spine* and some number of *pages*. The spine of a book is a line. For simple exposition, view the spine as being horizontal. Each page of the book is a half-plane that has the spine as its boundary. Thus any half-plane is a one-page book, and a plane with a distinguished horizontal line is a two-page book.

The embedding of an undirected graph in a book consists of two steps. The first step places the vertices of the graph on the spine in some order. The second step assigns each edge of the graph to one page of the book in such a way that on each page, the edges assigned to that page *do not cross*. Whether two edges cross is determined by the order of the vertices. If (s, t) and (u, v) are edges of the graph with $s < u < v$ and $s < t$, then the edges cross if and only if $s < u < t < v$. The resulting embedding is called a *book embedding* of the graph.

There are two measures of the quality of a book embedding for G.

The first measure is the *pagenumber* of the embedding, which is the number of pages in the book.

The *pagenumber* of the graph G is the minimum pagenumber of any book embedding of G. The *pagenumber* of a class of graphs is the minimum number of pages that every member of the class can be embedded in, as a function of graph size. The *width* of a page is the maximum number of edges that intersect any half-line perpendicular to the spine in the page.

The second measure is the *pagewidth* of the embedding, which is the maximum width of any page.

The *pagewidth* of the graph G is the minimum pagewidth of any book embedding of G in a book having a minimum number of pages. The *pagewidth* of a class of graphs is the minimum pagewidth that every member of the class can be embedded in, as a function of graph size. The *book embedding problem* is to find good book embeddings for a graph family with respect to one or both of these measures.

* Received by the editors November 13, 1985; accepted for publication (in revised form) July 31, 1986. This research was supported by National Science Foundation grant MCS-83-01213.

† Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139.

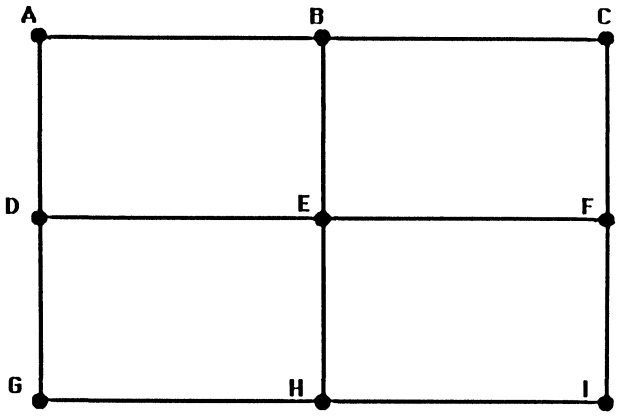

FIG. 1. *Grid graph G.*

As an example, consider the grid graph G of Fig. 1. A two-page embedding of G is shown in Fig. 2. The vertices of G are placed on the spine in the order *A-B-C-F-E-D-G-H-I*. The first page consists of the upper half-plane, and the second page consists of the lower half-plane. Edge (B, E) of the first page crosses edge (F, I) of the second page, so these two edges cannot be assigned to the same page of this book. The pagenumber of the book embedding is two, and the pagewidth is three as witnessed by the nested edges (A, D), (B, E), and (C, F) (both measures are optimal for G).

The book embedding problem is of interest because it models problems in several areas of computer science and VLSI theory. We mention here only problems arising from the DIOGENES approach of Rosenberg [9]. For further motivations, see Heath [5] or Chung, Leighton and Rosenberg [2].

Rosenberg [9] proposes the DIOGENES approach to the design of fault-tolerant arrays of VLSI processors. The elements of the approach are sketched here. One lays out some number of identical processors in a (conceptual) line. One provides sufficiently many processors so that one expects (probabilistically) that enough good processors exist to implement the desired array.

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). For any processor, a connection to a processor on its right is pushed on a stack; each connection to a processor on its left is popped from a stack. In this way, each connection to a good processor requires one stack operation at that processor. No stack operations occur at a bad processor. Since the state of a processor as good or bad is a binary value, a single control signal can cause the shift (push or pop) of many connections. Thus, fault tolerance is achieved by switching in only good processors.

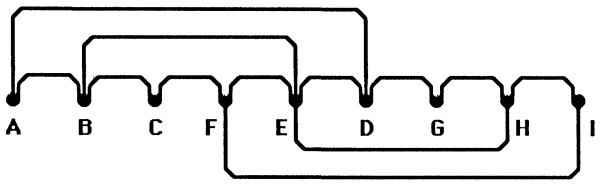

FIG. 2. *Two-page embedding of G.*

The desired array of processors is modeled as a *connection graph*; the vertices represent the processors, and the edges represent the desired connections between processors. The DIOGENES design problem is to determine the number of stacks and the *stackwidths* (the number of connections carried by each stack) required to implement the array of processors. In a way analogous to a hardware stack, it is possible to view one page of a book embedding as a stack of edges. For any vertex, each incident edge that connects it to a vertex to its right is pushed on a stack; each incident edge that connects it to a vertex to its left is popped from a stack. The DIOGENES design problem for an array of processors is exactly the book embedding problem for the corresponding connection graph. The number of stacks is exactly the number of pages. The stackwidths are the widths of the pages.

In this paper, we consider the problem of simultaneously attaining small pagenumber and small pagewidth. We consider the class of *outerplanar* graphs (an outerplanar graph is one that has a planar embedding with all vertices on the exterior face). There exist outerplanar graphs of size n that have pagewidth $\Omega(n)$ in any one-page embedding but have pagewidth $O(1)$ in an optimal two-page embedding. These graphs exhibit a tradeoff between pagenumber and pagewidth. We present an algorithm that produces a two-page embedding of small pagewidth for any outerplanar graph. The pagewidth that is attainable depends partly on the *valence* (maximum degree) of the outerplanar graph. Let G be a d-valent outerplanar graph with n vertices. Our algorithm embeds G in a two-page book having pagewidth less than $Cd \log n$ where $C = 8/(\log 3/2)$ (all logarithms are to the base two). This result is within a constant factor of optimal in pagewidth for the class of outerplanar graphs. The algorithm executes in time $O(n \log n)$.

The remainder of the paper consists of seven sections. In the next section, we survey previous results on book embeddings relevant to our algorithm. In § 3, we discuss tradeoffs between pagenumber and pagewidth and give an example of such a tradeoff. Section 4 presents the essential ideas of the algorithm, while § 5 gives the detailed statement. Section 6 proves the correctness of the algorithm, and § 7 establishes its performance. In the last section, we conclude with a discussion of the significance of our result and suggest an area for further research.

2. Previous results. The original statement of book embedding is a linear embedding performed in two parts. First, the vertices of a graph are placed on a line in some order. Second, each edge of the graph is embedded in one page so that no edges in the same page cross.

The resulting linear embedding can be transformed into a circular embedding in three steps. First, choose a distinct color for each page of the book, and assign each edge the color of its page. Second, "close" the book by projecting all pages (and their edges) into a single page. In this one-page book, if two edges cross, then the two edges have different colors. Third, curve the spine into a circle so that the "ends" at infinity are identified.

The result of the transformation is an alternate two-part formulation of the book embedding problem. First, order the vertices of the graph on a circle. Second, draw the edges of the graph as chords of the circle. Color the chords (edges) so that if two chords intersect in the interior of the circle, the chords have different colors. The number of colors in the circular embedding is exactly the number of pages in the corresponding linear embedding.

A useful consequence of the circular formulation is that any p-page graph is a subgraph of a p-page hamiltonian graph. (A graph is *hamiltonian* if it has a cycle containing all its vertices; such a cycle is called a *hamiltonian cycle*.) Moreover, the order of the

vertices in the circular embedding is exactly the order of the vertices in the hamiltonian cycle. To see this, let $v_1, v_2, \cdots, v_n$ be the vertices of the p-page graph in the cyclic order of the circular embedding. Add each of the edges (chords) (v_k, v_{k-1}), $1 \leqq k \leqq n$ (where $k-1$ is taken modulo n) that are not already present. Since these edges connect vertices adjacent on the circle, they cannot intersect any other edges. Therefore, each of the edges can legitimately be assigned to any page. The resulting edge-augmented graph is a p-page graph, with hamiltonian cycle $v_1, \cdots, v_n$.

The idea of adding edges to a graph to obtain a hamiltonian cycle is a strategy for obtaining the vertex order of a book embedding. We will call a cycle obtained in this fashion *superhamiltonian.* The following heuristic for book embedding a graph G is proposed in [2]:

(1) Obtain a superhamiltonian cycle for G and place the vertices of G on the circle in the order of the cycle;

(2) Color the edges of G by coloring the associated circle graph.

Finding an optimal solution to the second step in the heuristic is an NP-complete problem (Garey et al. [3]). The first step can be done in a number of ways; in fact, any ordering of the vertices can be obtained for a superhamiltonian cycle by adding the right edges. Thus, the problem of finding good book embeddings can be approached as that of finding a superhamiltonian cycle in an intelligent fashion so that a good (but not necessarily optimal) edge coloring can be produced.

2.1. One-page graphs. Any one-page graph can be embedded in the plane so that its vertices are on the spine and its edges are in the first page (the upper half-plane). Then all its vertices are exposed to the lower half-plane, which is a subset of the exterior face of the embedding. Thus the graph is outerplanar.

One characterization of an outerplanar graph is that its vertices can be embedded on a circle so that all its edges are inside the circle and no two edges intersect. This is just the condition that the graph be one-page embeddable under the circular formulation. We have the following:

PROPOSITION 1 (Bernhart and Kainen [1]). *G is one-page embeddable if and only if it is outerplanar.*

In fact, a k-page embedding of a graph G yields a decomposition of G into k outerplanar subgraphs, one for each page. The subgraphs share the vertices of G but are edge-disjoint. The outerplanarity of each subgraph is witnessed by the same circular ordering as that of the original book embedding.

2.2. Two-page graphs. Each two-page graph is a subgraph of a two-page hamiltonian graph. Every two-page graph is planar since the two half-planes (pages) together form a plane. Thus a two-page graph is a subgraph of a planar Hamiltonian graph.

Define a graph to be *subhamiltonian* if it is the subgraph of a planar hamiltonian graph. Given a subhamiltonian graph G, it is easy to show that G has a two-page embedding ([1]). Edge-augment G to obtain a superhamiltonian cycle in a planar graph. Order the vertices of G on a circle according to the superhamiltonian cycle. The edges of G interior to the cycle form an outerplanar graph. The edges exterior to the cycle form another outerplanar graph with its vertices in the same order as those of the interior one. A two-page embedding of G results. Thus we have the following:

PROPOSITION 2 [1]. *G is two-page embeddable if and only if it is subhamiltonian.*

Propositions 1 and 2 are the results we use in our algorithm to obtain two-page embeddings of outerplanar graphs with small pagewidth. From Proposition 1, an outerplanar graph G has a one-page embedding with all edges embedded in the upper half-plane (page). Our algorithm adds edges to G in the lower half-plane so that a planar,

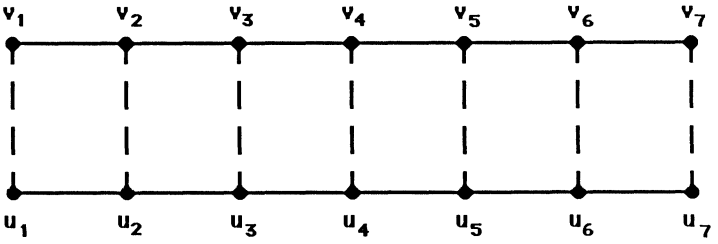

FIG. 3. *The 7-ladder* L_7.

hamiltonian supergraph results. By Proposition 2, the superhamiltonian cycle yields a two-page embedding of G with the vertices of G in cycle order.

3. Tradeoffs. We investigate the problem of tradeoffs between pagenumber and pagewidth in book embeddings. Motivation is best provided by an example from Chung, Leighton and Rosenberg [2]. The example is a sequence of outerplanar graphs $\{L_m\}$ for which any one-page embedding requires large pagewidth $\lceil m/2 \rceil$, but for which there exist two-page embeddings with pagewidth 2. The sequence consists of *m-ladders* (in [2], an m-ladder is called a *depth-m K_2-cylinder*). The m-ladder L_m has vertex set

$$\{u_1, \cdots, u_m\} \cup \{v_1, \cdots, v_m\}$$

and edge set

$$\{(u_k, u_{k+1}) | 1 \leqq k < m\} \cup \{(v_k, v_{k+1}) | 1 \leqq k < m\} \cup \{(u_k, v_k) | 1 \leqq k \leqq m\}.$$

The first two components of the edge set constitute the two *sides* of the ladder while the last component constitutes its *rungs*. Figure 3 illustrates L_7. The sides are solid and the rungs are dashed.

The m-ladder is clearly outerplanar and biconnected. By biconnectivity, L_m has a unique outerplanar embedding (Syslo [10]). Therefore, L_m has a unique one-page embedding up to reflection and circular permutation. Figure 4 illustrates a one-page embedding of L_7 of minimal pagewidth over all one-page embeddings. The rungs $\{(u_4, v_4), (u_5, v_5), (u_6, v_6), (u_7, v_7)\}$ nest over the interval (u_7, v_7). Hence the pagewidth is $\geqq 4$. A moment's reflection generalizes this observation: In any one-page embedding for L_m, at least $\lceil m/2 \rceil$ rungs nest over some interval; hence pagewidth is $\geqq \lceil m/2 \rceil$.

Figure 5 illustrates a two-page embedding for L_7 that has pagewidth 2. The corresponding superhamiltonian cycle is illustrated in Fig. 6. This superhamiltonian cycle is easily generalized, giving a two-page embedding of any L_m with pagewidth 2.

We now discuss tradeoffs in the general setting of an arbitrary graph G. Let P be the pagenumber of G. For each $p \geqq P$, there exist one or more embeddings of G in a p-

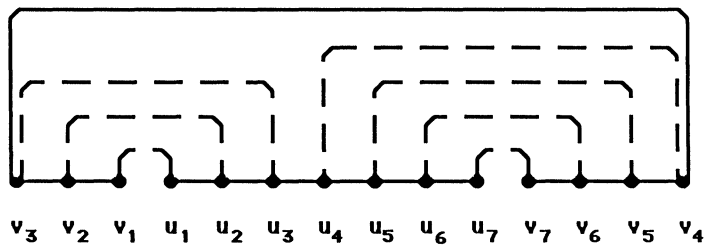

FIG. 4. *One-page embedding for* L_7.

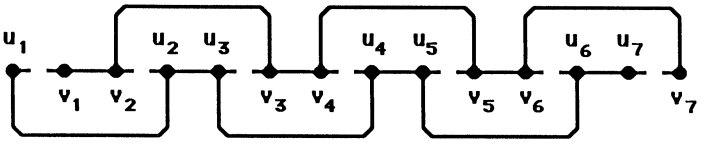

FIG. 5. *Two-page embedding for* L_7.

page book. Among all those p-page embeddings of G, let w_p denote the pagewidth of one having minimum pagewidth. These pagewidths are nonincreasing:

$$w_P \geqq w_{P+1} \geqq \cdots \geqq w_p, p \geqq P.$$

In the extreme case that $p \geqq |E|$, $w_p = 1$, as each edge may be assigned to a distinct page.

We are particularly interested in the product pw_p. We seek cases where pw_p is within a constant factor of the cutwidth of G. Note that pw_p is an upper bound on the cutwidth of the best p-page embedding of G. In the context of the DIOGENES approach, pw_p is an upper bound on the height of a p-stack DIOGENES layout of G. Hence, we seek DIOGENES layouts of G that are within a constant factor of optimal in area over all linear layouts and within a small additive constant of optimal in stacknumber.

Our result is for the class of one-page (i.e., outerplanar) graphs. The m-ladder exhibits an extreme pagewidth tradeoff between one-page and two-page embeddings. For general outerplanar graphs, we do not expect such an extreme tradeoff. Since there exist outerplanar graphs that have one-page embeddings of minimal pagewidth, e.g., complete binary trees, the tradeoff in going from one page to two pages can be arbitrarily small, even zero.

An n-vertex complete $(d-1)$-ary tree has cutwidth $\geqq (d/2) \log n$ (Lengauer [6]). (All logarithms are to the base 2.) Hence, any book embedding of a complete $(d-1)$-ary tree in a constant number of pages requires pagewidth $\Omega(d \log n)$. In general, we cannot assume that outerplanar graphs have pagewidth $o(\log n)$.

4. Overview of the algorithm. The tradeoff result we show is that any d-valent outerplanar graph G can be embedded in a two-page book with pagewidth $Cd \log n$, where $C = 8/(\log 3/2)$. From the observations in the preceding section regarding m-ladders and complete $(d-1)$-ary trees, this result is optimal in pagenumber and within a constant factor of optimal in pagewidth for the class of d-valent outerplanar graphs. We prove our result via a recursive algorithm.

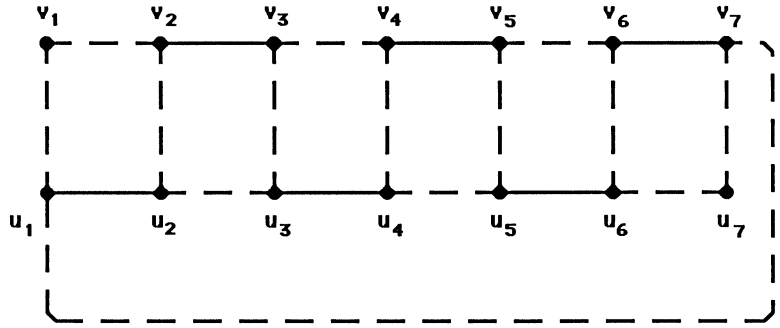

FIG. 6. *Superhamiltonian cycle for* L_7.

We aim for an algorithm that, when given an n-vertex d-valent outerplanar graph, returns a two-page embedding with pagewidth logarithmic in n. The input and output requirements of such an algorithm are a useful place to start.

The input to the algorithm is a d-valent outerplanar graph $G = (V, E)$. The manner of representing this input should witness the outerplanarity of G. Hence, a one-page embedding of G is the required form for the input. The linearization of V orders the vertices and provides names $1, 2, \cdots, n$ for the vertices. The order of the vertices in the two-page embedding will *not* be the original order, but we shall continue to use the *names*. Since the algorithm is recursive, the same vertex will have different names at different levels of recursion. Figure 7 illustrates a possible form of the input when $G = L_7$.

The output of the algorithm is a two-page embedding of G with logarithmic pagewidth. To give a two-page embedding for G, it is sufficient to give a superhamiltonian cycle H in a supergraph G' of G (Proposition 2). $G' = (V, E)$ is actually a *multigraph* (i.e., it may contain loops and multiple copies of edges) that contains all the edges of G plus possibly edges added to obtain H. $H \subset E$ is a set of n edges; since H is superhamiltonian, each of $1, \cdots, n$ appears exactly twice among these edges. H represents $2n$ different book embeddings for G: there are n choices for the leftmost vertex, and there are two directions to the cycle. The algorithm fixes the desired book embedding by returning the leftmost (x) and rightmost (y) vertices of the two-page embedding. We call x and y the *vertices of attachment* for G', for reasons that will become clear. The output of the algorithm is then the ordered triple $(G', H, (x, y))$.

We imagine the one-page embedding of G as follows. The vertices are on a horizontal line in a plane, and the edges are drawn in the upper half-plane. In general, there are many sets of edges that can be added to G without destroying planarity. We restrict ourselves to two types of edges, *upper edges* and *lower edges*, depending on which half-plane the edges are embedded in. (Thus our restriction is that no edge uses both half-planes in its embedding.) The original edges of G are always upper edges. The algorithm may add an upper edge if it will not cross an existing upper edge. The algorithm may add a lower edge if it will not cross an existing lower edge. In particular, we may (and shall) assume that the upper edges $(i, i + 1)$, $1 \leqq i < n$ are always present in G; they can always be added with no affect on pagenumber and at most unit increase in pagewidth.

The algorithm uses the divide-and-conquer paradigm. It determines subgraphs of G to work on separately before the results are joined together to obtain G'. Each subgraph is induced by a subinterval of $[1, n]$. We define the closed subinterval $[i, j]$ to be the set of integers $\{i, i + 1, \cdots, j\}$. We define two types of half-closed, half-open subintervals: $[i, j)$ denotes $[i, j - 1]$ and $(i, j]$ denotes $[i + 1, j]$. For any subinterval α, size $\{\alpha\}$ denotes the number of vertices in the subinterval; hence, size $\{[i, j]\} = j - i + 1$. Define $G[i, j]$

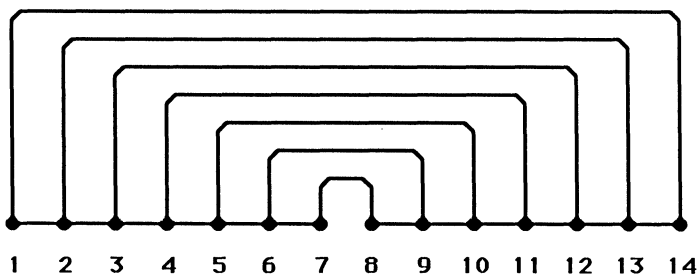

FIG. 7. *Input representation for* L_7.

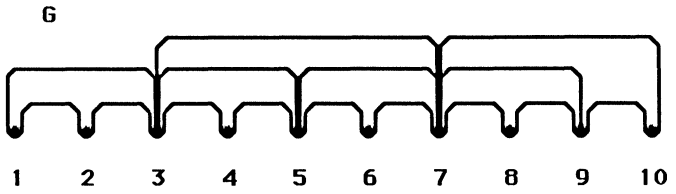

FIG. 8. *Sample G for divide-and-conquer.*

to be the subgraph of G induced on the vertices in the interval $[i, j]$. If the algorithm is applied to $G[i, j]$ the result is $(G'[i, j], H, (x, y))$, where (x, y) determines the first and last vertices of a two-page embedding of $G[i, j]$ with pagewidth logarithmic in size $\{[i, j]\}$.

The choice of subintervals depends on the structure of the one-page embedding of G. Define an *exposed vertex* w of G to be one for which G contains no (upper) edge (u, v) satisfying $u < w < v$. Thus an exposed vertex w is one that is "visible" from the infinite region of the upper half-plane. Each exposed vertex of G except 1 and n is a cutpoint of G whose removal separates G into left and right subgraphs.

An example will illustrate the divide-and-conquer paradigm. Figure 8 shows a sample G in a one-page embedding. The exposed vertices of G are 1, 3, 7 and 10. The algorithm recognizes that each of the edges (1, 3), (3, 7) and (7, 10) is "highest" in the sense that no other edge passes over it. These three edges determine three *nondisjoint* subintervals [1, 3], [3, 7] and [7, 10]. In order to decompose the interval into *disjoint* subintervals, the algorithm chooses the largest, [3, 7], to remain intact, and removes one vertex from each of the other two subintervals. The resulting subintervals are [1, 2], [3, 7] and [8, 10]. The algorithm recursively applies itself to each of the subintervals. The result to this point is shown in Fig. 9. Each subproblem displays a superhamiltonian cycle of its subgraph and the first and last vertices of the corresponding two-page embedding. In Fig. 10, these three superhamiltonian cycles are replaced by a superhamiltonian cycle for the entire graph. Lower edges (1, 2), (4, 6) and (8, 10) are deleted and lower edges (2, 4), (6, 8) and (1, 10) are added.

If two exposed vertices i and j are joined by an (upper) edge (i, j), then there are no other exposed vertices in the interval $[i, j]$. In this case, we call $G[i, j]$ a *block*, denoted $B[i, j]$. When the interval $[1, n]$ is partitioned into subintervals, there will be edges with endpoints in different subintervals. Such *dangling* edges are exactly those edges of G not in any of the subgraphs generated by the subintervals. In the case of a block $B[i, j]$, these dangling edges can be incident to only i or j. The total number of such edges incident to i or j is called the *edge deficit* of $B[i, j]$, denoted def $\{[i, j]\}$. It is always true that

$$\text{def}\,\{[i, j]\} \leqq 2(d-1).$$

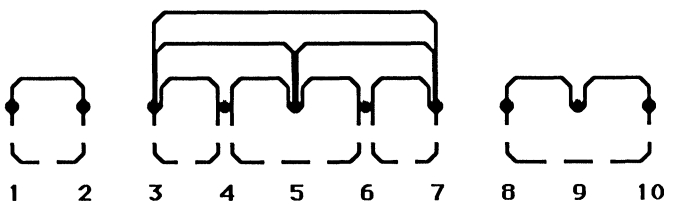

FIG. 9. *Results of subproblems.*

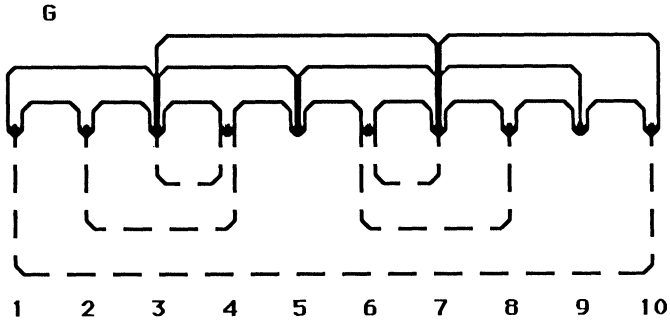

FIG. 10. *Superhamiltonian cycle for G.*

In Fig. 8, $B[1, 3]$, $B[3, 7]$ and $B[7, 10]$ are the blocks of G, and def $\{[3, 7]\} = 5$ (because of edges (1, 3), (2, 3), (7, 8), (7, 9), and (7, 10)).

The example of Figs. 8–10 illustrate the execution of the algorithm in the case that G has two or more blocks. There is another possible case: G has only one block. In that case, the divide-and-conquer construction is more complex. The two divide-and-conquer constructions corresponding to these two cases are developed in turn in the next two subsections.

4.1. String construction. We now describe one of the two constructions used to obtain a superhamiltonian cycle for G from superhamiltonian cycles for the graphs induced by subintervals. It is called the *string construction.* (The name suggests that the superhamiltonian cycles for the subintervals are *strung together* to obtain a superhamiltonian cycle for the entire interval.) It is employed when the number of exposed vertices is greater than two, i.e., when G is not one block. The partition into subintervals keys on the largest block, say $B[i, j]$: $B[i, j]$, is taken to be one of the subintervals.

A precise description of the partition into subintervals requires more notation. Let $m_1, m_2, \cdots, m_q$ be the exposed vertices of G in ascending order. Suppose $B[m_k, m_{k+1}]$ is the largest block in G. Figure 11 illustrates the situation. The partition into $q - 1$ subintervals is

$$\{[m_1, m_2), [m_2, m_3), \cdots, [m_{k-1}, m_k), [m_k, m_{k+1}], (m_{k+1}, m_{k+2}], \cdots, (m_{q-1}, m_q]\}.$$

Note that $B[m_k, m_{k+1}]$ is the only block of G in the partition. It is called the *key* block of the partition. The other subintervals are called *side* subintervals. Figure 12 illustrates the partition of G.

The algorithm is recursively applied to the jth subinterval to obtain

$$(G'_j, H_j, (x_j, y_j)).$$

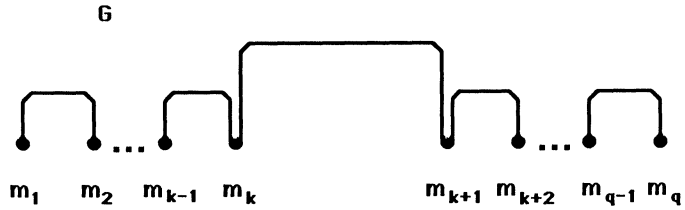

FIG. 11. *Exposed vertices.*

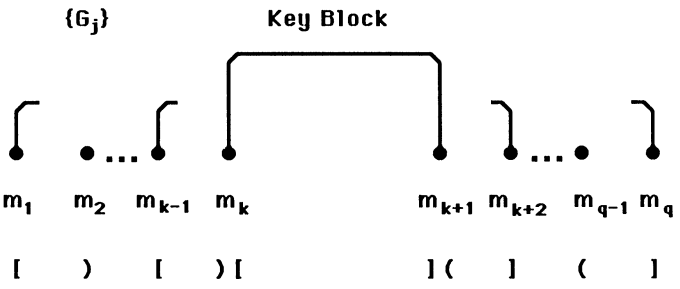

FIG. 12. *Partition into subintervals.*

G' is obtained in two steps. First, all edges added to the G_j' are added to G (Fig. 13). Second, the lower edges $\{(x_j, y_j)\}$ are deleted and the lower edges

$$\{(y_j, x_{j+1}) | 1 \leqq j \leqq q-2\} \cup \{(x_1, y_{q-1})\}$$

are added. H is obtained from $\cup H_j$ by deleting and adding the same edges (Fig. 14). Assigning $(x, y) = (x_1, y_{q-1})$ completes the string construction. The correctness of the construction is proved in Lemma 5 (§ 6).

4.2. Ladder construction. In this section, we consider the case when G has only one block, so we are unable to divide G into subintervals based on blocks. To reach a solution, we first focus on the problem of obtaining logarithmic pagewidth. To obtain logarithmic pagewidth, it is clearly sufficient that the linear layout corresponding to the two-page embedding have logarithmic cutwidth. An approach to small cutwidth is the recursive application of a *separator theorem* (see Lipton and Tarjan [7]). A separator theorem states that the removal of some number of vertices from a graph will partition the remainder of the graph into two subgraphs of approximately equal size. For outerplanar graphs, a two-vertex separator always exists.

LEMMA 3. *Let G be an outerplanar graph containing at least 3 vertices. There exist vertices x and y whose removal separates G into disjoint subgraphs G_1 and G_2 such that $\frac{1}{3}n < |G_k| < \frac{2}{3}n$, $k = 1, 2$. If (x, y) is not an upper edge of G, then it can be added to G as an upper edge without inducing a crossing.*

Proof. Since G is outerplanar, we can use the circular formulation of book embedding to embed G in a circle. The vertices of G are placed equally spaced on the circle. The edges of G are chords of the circle with no two chords intersecting. If the center of the circle lies on an edge, let x and y be the endpoints of the edge; in this case, $|G_k| < \frac{1}{2}n$, $k = 1, 2$, and the result follows. Otherwise, let F be the face of G containing the center. If two vertices on F are on a diameter, let them be x and y, and the result follows.

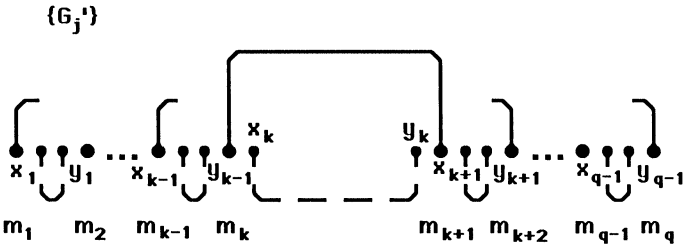

FIG. 13. *Results for subintervals.*

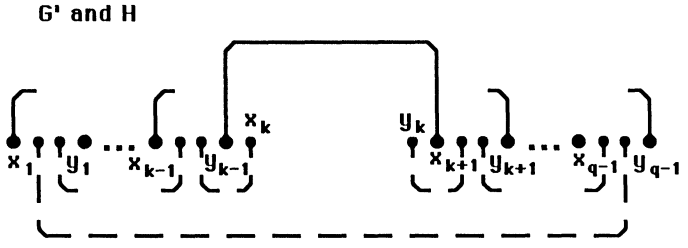

FIG. 14. *Subintervals strung together.*

Otherwise, triangulate F within the circle. The center of the circle lies within some resulting triangle (u, v, w). Let the angle $\angle uvw$ be the largest of the triangle. This angle is easily seen to be between 60° and 90°. Let $x = u$ and $y = w$. Let G_1 be the graph induced by the vertices within the angle $\angle uvw$, and let G_2 be the graph induced by the vertices outside the angle $\angle uvw$. Then the removal of x, y separates G into G_1 and G_2 where

$$\tfrac{1}{3}n < |G_1| < \tfrac{1}{2}n.$$

Note that the edge (x, y) can be added to G without destroying the outerplanar embedding. The lemma follows. □

If (x, y) is not already an edge of G, it can be added without destroying outerplanarity. An edge (x, y) that satisfies Lemma 3 is called a *separating edge.* An algorithm to obtain logarithmic cutwidth for a d-valent outerplanar graph G can select a separating edge (x, y) and apply itself recursively to the resulting G_1 and G_2. However, it is unclear how to obtain a superhamiltonian cycle for G from superhamiltonian cycles for G_1 and G_2.

Our algorithm uses separating edges in another way so as to make it possible to derive a superhamiltonian cycle from superhamiltonian cycles for the pieces. The key is the following definition. Let G be an outerplanar graph, and let (x, y) be a separating edge for G. A set of edges $P \subset E$ is *parallel* to (x, y) if

(1) $(x, y) \in P$;

(2) if $(u, v), (w, z) \in P$, then $\{u, v\} \cap \{w, z\} = \varnothing$ (there are no shared endpoints);

(3) P can be ordered as $\{(u_1, v_1), (u_2, v_2), \cdots, (u_k, v_k)\}$ in such a way that

$$u_1 < u_2 < \cdots < u_k < v_k < \cdots < v_2 < v_1$$

(the edges of P nest).

A sample set of parallel edges for a graph G is shown in Fig. 15 by dashed lines. A set P of parallel edges is *maximal* if no edge of G can be added to P to obtain a larger set of parallel edges.

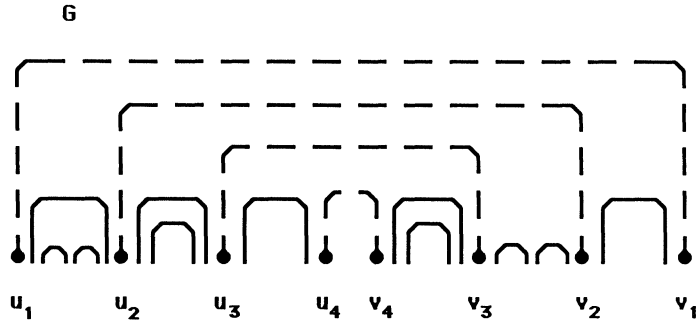

FIG. 15. *Parallel edges in G.*

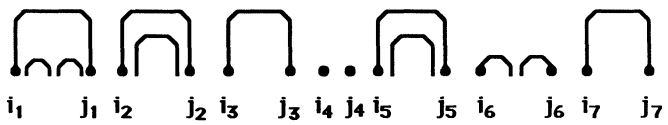

FIG. 16. *Removal of* V_P.

Suppose P is a maximal set of parallel edges for G containing the separating edge (x, y). Let V_P be the set of endpoints of edges in P. The removal of the vertices V_P from G separates the interval $[1, n]$ into some number of subintervals. Let G_P be the subgraph of G resulting from the removal of V_P and all incident edges. Let $[i_1, j_1], \cdots, [i_s, j_s]$ be these subintervals in left-to-right order. The planarity of G and the maximality of P guarantees that there is no edge of G between two vertices in different subintervals. This in turn guarantees that G_P can be obtained an alternate way: G_P is the (disjoint) union of the induced subgraphs $G[i_k, j_k]$, $1 \leqq k \leqq s$. By Lemma 3, the presence of a separating edge in P guarantees that

$$\text{size } \{[i_k, j_k]\} < \tfrac{2}{3}n, \qquad 1 \leqq k \leqq s.$$

Figure 16 shows the graph of Fig. 15 after the removal of V_P.

The algorithm is applied recursively to each $G[i_k, j_k]$ to obtain a superhamiltonian cycle for each. To obtain a superhamiltonian cycle for G, one need only reintroduce the endpoints of the parallel edges V_P. A second look at Fig. 15 provides inspiration. If each subinterval $[i_s, j_s]$ were replaced by an edge $(i_s - 1, j_s + 1)$ between two vertices in V_P, the result is the one-page embedding of a ladder where *all* the rungs nest. The construction of a superhamiltonian cycle H for G is patterned after the superhamiltonian cycle for a ladder, as illustrated in Fig. 6. Appropriately, we name the construction of H the *ladder construction.*

There are two cases to consider, depending on whether or not the edge $(1, n)$ is in P. The case $(1, n) \in P$ illustrates all the ideas and is simply modified to cover the case $(1, n) \notin P$.

Start with the picture of the parallel edges alone in Fig. 17. Some lower edges are added to obtain a supercycle containing exactly the vertices in V_P. This supercycle is indicated in Fig. 18 by arrows. It remains to place all the subintervals within this supercycle. To accomplish this, each lower edge is replaced by new lower edges that connect two subintervals into its place in the supercycle. For a right arrow (u_k, u_{k+1}), the result is as in Fig. 19. For a left arrow (v_{k-1}, v_k), the result is as in Fig. 20; t is chosen so that $[i_t, j_t]$ is the subinterval between v_{k+1} and v_k.

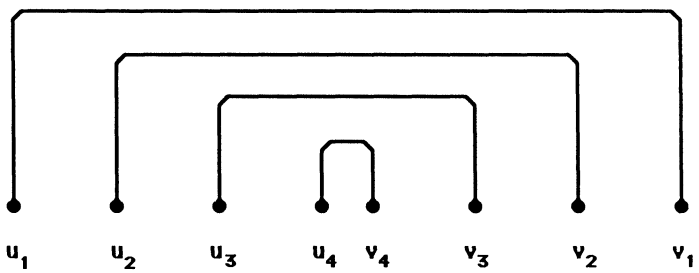

FIG. 17. *Parallel edges.*

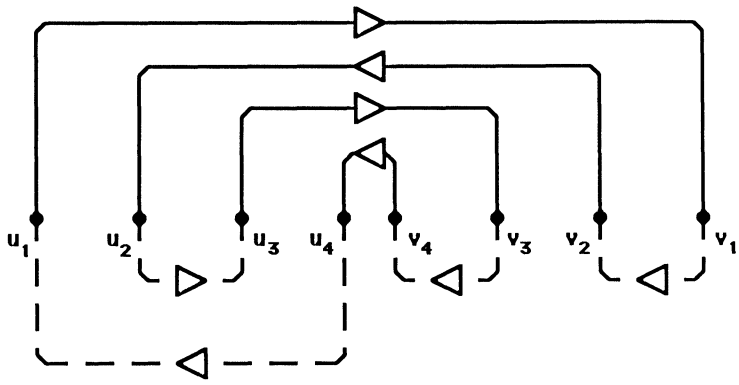

FIG. 18. *Supercycle for parallel edges.*

For the case $(1, n) \notin P$, $[i_1, j_1]$ is to the left of the ladder and $[i_s, j_s]$ is to the right of the ladder. The connection of $[i_1, j_1]$ into H is shown in Fig. 21. The connection of $[i_s, j_s]$ into H is shown in Fig. 22.

5. The algorithm. This section describes our algorithm for embedding a d-valent outerplanar graph in a two-page book with logarithmic pagewidth. The correctness of the algorithm, embodied in Theorem 4, is given in the next section. Section 8 analyzes the performance of the algorithm.

For the statement of our algorithm, see Algorithm 1. As the algorithm is recursive, it is useful to give it a name. The name is TRADEOFF. TRADEOFF is a recursive function which has as input the d-valent outerplanar graph G and as output the planar supergraph G' having hamiltonian cycle H and vertices of attachment x and y.

It is to be noted that, for simplicity, certain trivial cases are not included in the statement of TRADEOFF. These cases occur when a recursive invocation of TRADEOFF returns an empty G'. This cannot occur in step 5, as each subinterval contains at least one vertex. However, it can occur in step 9 when some G_k is empty. In that case, the ladder construction merely skips the empty interval $[i_k, j_k]$ (which is caused by two adjacent elements of V_P).

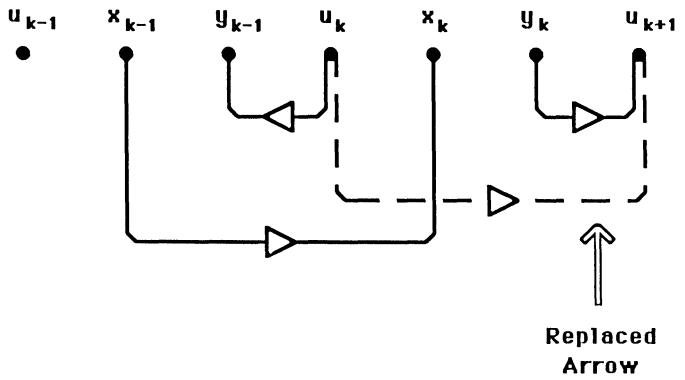

FIG. 19. *Replacing a right lower edge.*

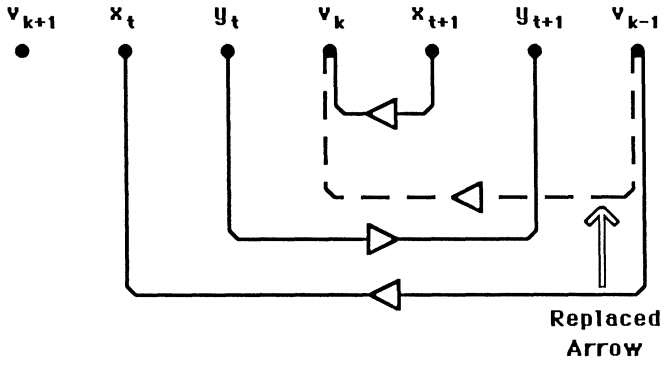

FIG. 20. *Replacing a left lower edge.*

ALGORITHM 1. *The Tradeoff Algorithm.*
Function TRADEOFF (G), **returns** (G', H, (x, y)).

(1) (Trivial cases)
If $G = \varnothing$, then assign $G' = \varnothing$, $H = \varnothing$ and $(x, y) = \textit{undefined}$.
If $V = \{1\}$, then assign $G' = (\{1\}, \{(1, 1)\})$, $H = \{(1, 1)\}$ and $(x, y) = (1, 1)$.
If $V = \{1, 2\}$, then assign $G' = (\{1, 2\}, \{(1, 2), (1, 2)\})$, $H = \{(1, 2), (1, 2)\}$ and $(x, y) = (1, 2)$.
Return $(G', H, (x, y))$.

(2) Let $S = \{m_j | 1 \leqq j \leqq q\}$ be the set of exposed vertices of G in increasing order.

(3) Choose k, $1 \leqq k \leqq q - 1$ such that $B[m_k, m_{k+1}]$ is the key block of G.

(4) If $B[m_k, m_{k+1}] = G$, then go to step 7.

String Construction

(5) (G has more than one block.) For $1 \leqq j < k$, assign

$$(G'_j, H_j, (x_j, y_j)) = \text{TRADEOFF } (G[m_j, m_{j+1}]).$$

For $j = k$, assign

$$(G'_j, H_j, (x_j, y_j)) = \text{TRADEOFF } (G[m_k, m_{k+1}]).$$

For $k < j < q$, assign

$$(G'_j, H_j, (x_j, y_j)) = \text{TRADEOFF } (G(m_j, m_{j+1}]).$$

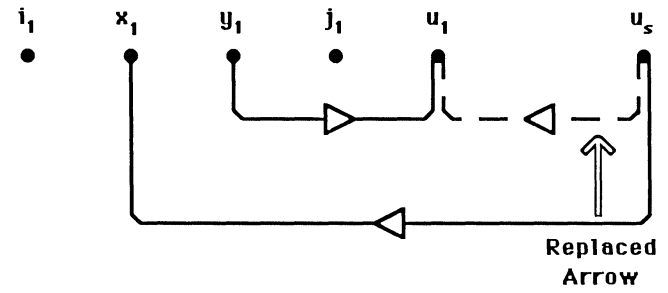

FIG. 21. *Adding a subinterval on the left.*

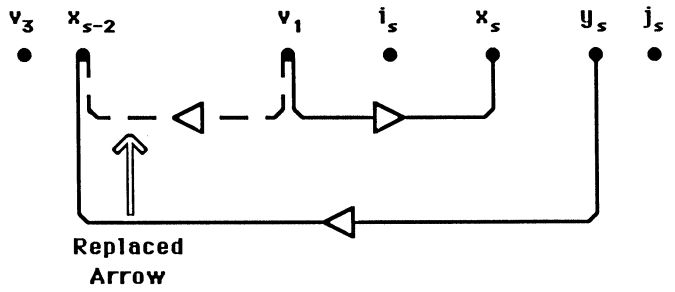

FIG. 22. *Adding a subinterval on the right.*

(6) Use the string construction to obtain G', H and (x, y). Return $(G', H, (x_1, y_q))$.

Ladder Construction

(7) ($B[m_k, m_{k+1}] = G$.) Choose a separating edge (u, v) for G. If (u, v) is not already an edge of G, then add it as an upper edge.

(8) Choose P a maximal set of edges parallel to (u, v). Let V_P be the set of endpoints of edges in P.

(9) $V - V_P$ determines a sequence of disjoint subintervals $[i_1, j_1]$, $[i_2, j_2]$, $\cdots$, $[i_s, j_s]$. For $1 \leqq k \leqq s$, make the assignment:

$$(G'_k, H_k, (x_k, y_k)) = \text{TRADEOFF } (G[i_k, j_k]).$$

Construct G' from G and $\{G'_k\}$ using the ladder construction. Return $(G', H, (x, y))$.

We now describe TRADEOFF step by step.

(1) These are the trivial cases when $n \leqq 2$. If G is empty, return $G' = \varnothing$. If G is a single vertex, return G' having a single loop. If G has two vertices, then it has one edge (1, 2). Return G' having the added lower edge (1, 2) which is distinct from the upper edge (1, 2).

(2) From the one-page embedding of G, determine the exposed vertices of G. It is straightforward to accomplish this step in linear time; see Algorithm 2. Algorithm 2 requires time $O(dn)$ and generates the elements of S in increasing order.

(3) Choose the key block of G, $B[m_k, m_{k+1}]$. Clearly, this can be accomplished in time linear in $|S|$.

(4) This step determines which of two cases is current. If G is a single block, then the ladder construction is applied (steps 7 through 9). If G has more than one block, then the string construction is applied (steps 5 and 6).

(5) Decompose the interval $[1, n]$ into subintervals so that the key block $B[m_k, m_{k+1}]$ is one of the subintervals. Note that each side subinterval contains fewer than $\frac{1}{2}n$ vertices. Apply TRADEOFF to the graphs induced by each subinterval to obtain supergraphs G'_j, $1 \leqq j \leqq q - 1$.

ALGORITHM 2. *Determining exposed vertices in linear time.*

(1) Assign $S = \{1\}$ and $i = 1$.

(2) If $i > n$, then halt.

(3) Assign $i = \max \{i + 1, \max_{(i,k) \in E} k\}$.

(4) Assign $S = S \cup \{i\}$. Go to step 2.

(6) Apply the string construction to obtain the planar hamiltonian supergraph G' and the hamiltonian cycle H for G'. Assign $(x, y) = (x_1, y_q)$. Return $(G', H, (x, y))$.

(7) We know that G is entirely covered by the edge $(1, n)$. We show that it is then safe to add a separating edge to G. If this is the initial call to TRADEOFF, we can always add a separating edge. If this is a deeper recursive call to TRADEOFF, we imagine that there are intervals to the left and right of $[1, n]$ with dangling edges incident to vertices in $[1, n]$. Since these dangling edges can only be incident to exposed vertices (in this case, 1 and n), any upper edge added to G at this recursive level cannot cross an edge at a higher recursive level. The determination of a suitable separating edge is accomplished in linear time by Algorithm 3. (Note that the triangulated G has a linear number of edges.)

(8) Select a maximal set of parallel edges. The construction of P is accomplished in linear time by Algorithm 4.

(9) This step completes the ladder construction. TRADEOFF is invoked recursively for each subinterval disjoint from V_P. G' and H are obtained by the ladder construction described in the previous section.

ALGORITHM 3. *Finding a separating edge.*

(1) Triangulate the interior faces of G.

(2) Examine each edge (u, v) of the triangulated G to find one such that $\frac{1}{3}n \leqq (v - u) \leqq \frac{2}{3}n$.

ALGORITHM 4. *Generating a maximal set of parallel edges.*

(1) Assign $P = \{(u, v)\}$, $s = u - 1$ and $t = v$.

(2) If $s < 1$, then go to step 4.

(3) Assign $r = \max\{1, \max_{(s,k)\in E} k\}$. If $r \leqq t$, then assign $s = s - 1$ and go to step 2. Else assign $P = P \cup \{(s, r)\}$, $s = s - 1$ and $t = r$ and go to step 2.

(4) Assign $s = u + 1$, and $t = v$.

(5) If $s \geqq t$, then halt.

(6) Assign $r = \min\{n, \min_{(s,k)\in E} k\}$. If $r < t$, then assign $P = P \cup \{(s, r)\}$, $s = s + 1$ and $t = r$ and go to step 5. Else, assign $s = s + 1$ and go to step 5.

6. Correctness. In this section, we demonstrate the correctness of algorithm TRADEOFF via the following theorem.

THEOREM 4. *Let G be a d-valent outerplanar graph. Let $(G', H, (x, y))$ result from applying TRADEOFF to G. Then H is a superhamiltonian cycle for G with the following property: following H from x to y yields a two-page embedding of G with pagewidth $< Cd \log n$, where C is a constant that can be chosen to have any value $\geqq 8/(\log 3/2)$.*

Proof. The proof decomposes naturally into the proof of pagenumber (Lemma 5) and the proof of pagewidth (Lemma 6). □

LEMMA 5. *Given the assumptions of Theorem 4, following H from x to y yields a two-page embedding of G.*

Proof. The proof is by induction on n. The inductive hypothesis is

(H.1) $G \subset G'$;

(H.2) G' is planar;

(H.3) H is a hamiltonian cycle of G';

(H.4) $(x, y) \in H$ is a lower edge of G' such that there is no lower edge (u, v) of

G' with $u < x \leqq y < v$ (i.e., x and y are on the unbounded region of the lower half-plane).

Step 1 of Algorithm 1 guarantees that the inductive hypothesis is satisfied when $n \leqq 2$.

For purposes of induction, assume that the inductive hypothesis is true for graphs of size less than n and that $n > 2$. There are two cases determined by the cardinality of the set S of exposed vertices of G: (1) $|S| > 2$ and (2) $|S| = 2$.

(1) $|S| > 2$. TRADEOFF applies the string construction in steps 5 and 6. The inductive hypothesis guarantees that after the applications of TRADEOFF to all the subintervals, each x_j and each y_j is on the unbounded region of the lower half-plane. Therefore, the lower edges (y_j, x_{j+1}), $1 \leqq j \leqq q - 1$ and (x_1, y_q) can be added while maintaining planarity (H.2). Clearly, $G \subset G'$ (H.1), and H is a hamiltonian cycle of G' (H.3). Finally $x = x_1$ and $y = y_q$ satisfy (H.4).

(2) $|S| = 2$. The edge $(1, n)$ is in G and covers all other upper edges. TRADEOFF applies the ladder construction to G in steps 7 through 9. In § 2.2, the addition of the separating upper edge (u, v) was shown to maintain planarity. In step 9, the application of TRADEOFF to each $G[i_k, j_k]$ yields $(G'_k, H, (x_k, y_k))$ that satisfies the inductive hypothesis. In particular, (H.4) applies to each (x_k, y_k). Since each x_k and y_k is on the unbounded region of the lower half-plane, the ladder construction yields a planar result (H.2). The ladder construction also makes $G \subset G'$ (H.1) and H a hamiltonian cycle of G' (H.3). Finally, (x, y) is explicitly chosen to satisfy (H.4).

This extends the induction for arbitrary G. Since H is a hamiltonian cycle of a planar supergraph of G, it yields a two-page embedding of G [1]. □

To complete the proof of Theorem 4, we must bound the pagewidth of the two-page embedding. It is sufficient to bound the cutwidth of the underlying linear embedding. We use the notation cw (H) to mean the cutwidth of the linear embedding obtained by following H from x through y. (G, x and y will be clear from context.) If i and j are vertices in H such that i comes before j in the linear embedding, define cw $([i, j])$ to be the cutwidth of the linear subembedding from i to j.

LEMMA 6. *Given the assumptions of Theorem* 4, $\text{cw}(H) < Cd \log n$, *where* $C = 8/(\log 3/2)$.

Proof. The proof is by induction on n. The statement of the inductive hypothesis mirrors the two cases of the algorithm. The inductive hypothesis is

(I.1) If G has more than one block, then

$$\text{cw}(H) < Cd \log n;$$

(I.2) If G is a single block, then

$$\text{cw}(H) \leqq \max(1, Cd \log n) - \text{def}\{[1, n]\}.$$

Some explanation of the presence of the edge deficit in (I.2) is in order. In the string construction, a large key block $[m_k, m_{k+1}]$ must be able to absorb def $\{[m_k, m_{k+1}]\}$ additional cutwidth, as its cutwidth will dominate the cutwidth of the entire string construction. The precise meaning of this statement will be clear from the proof. The max $(1, Cd \log n)$ takes care of the case $n = 1$. Note that a G with a single vertex can never be the key block in a string construction.

For the basis of the induction, it is easy to check the inductive hypothesis for $n = 1$ and $n = 2$.

For purposes of induction, assume that the inductive hypothesis is true for graphs of size less than n and that $n > 2$. There are two cases: (1) G has more than one block and (2) G is a single block.

(1) *G has more than one block.* In this case, the string construction is applied (steps 5 and 6). Let us examine the linear order induced by H and (x, y) on V. H_1 is a super-hamiltonian cycle for $G([m_1, m_2))$ that begins at $x = x_1$ and ends at y_1. As such, H_1 can be viewed as a permutation on $[m_1, m_2)$. The string construction places the vertices of $[m_1, m_2)$ first in H, in this permuted order. Similarly, the vertices of $[m_2, m_3)$ come next in H, in the permuted order given by H_2. In general, the $q - 1$ subintervals appear in the same order in H as they do in the partition, though H permutes the vertices within each subinterval. The permutation of the jth subinterval is always that of H_j.

It is now possible to bound cw (H) based on $\{\text{cw}(H_j)\}$. First, consider the cutwidth of H between two subintervals, that is, cw $([y_j, x_{j+1}])$, $1 \leqq j \leqq q - 2$. Suppose $j < k$. Then the only edges that pass over the interval $[y_j, x_{j+1}]$ are dangling edges from m_{j+1} to $[m_j, m_{j+1})$. Hence,

$$\text{cw}([y_j, x_{j+1}]) \leqq d - 1 < Cd \log n.$$

If $j \geqq k$, by a similar argument, we have

$$\text{cw}([y_j, x_{j+1}]) \leqq d - 1 < Cd \log n.$$

Second, consider the cutwidth over a side subinterval. Consider the jth subinterval in H, $[x_j, y_j]$. If $j < k$, then there are at most $(d - 1)$ dangling edges from m_{j+1} to $[m_j, m_{j+1})$ that can contribute to cw $([x_j, y_j])$ and at most d dangling edges from m_j to $[m_{j-1}, m_j)$ that can contribute to cw $([x_j, y_j])$. Hence by (I.1)

$$\begin{aligned}\text{cw}([x_j, y_j]) &< 2d + Cd \log (\text{size}\{[m_j, m_{j+1})\}) \\ &< Cd \log n\end{aligned}$$

since size $\{[m_j, m_{j+1})\} < \frac{1}{2}n$. If $j > k$, we have similarly

$$\begin{aligned}\text{cw}([x_j, y_j]) &< 2d + Cd \log (\text{size}\{(m_j, m_{j+1}]\}) \\ &< Cd \log n.\end{aligned}$$

Third and finally, consider the cutwidth over the key block, $B[m_k, m_{k+1}]$. By (I.2),

$$\text{cw}([x_k, y_k]) \leqq (Cd \log (\text{size}\{[m_k, m_{k+1}]\})) - \text{def}\{[m_k, m_{k+1}]\}.$$

The only dangling edges that can contribute to the cutwidth over $[x_k, y_k]$ are those incident to m_k and m_{k+1}. There are def $\{[m_k, m_{k+1}]\}$ of these. Hence

$$\begin{aligned}\text{cw}([x_k, y_k]) &\leqq Cd \log (\text{size}\{[m_k, m_{k+1}]\}) \\ &< Cd \log n.\end{aligned}$$

Putting these three results together yields cw $(H) < Cd \log n$. Thus G satisfies (I.1).

(2) *G is a single block.* In this case, the ladder construction is applied (steps 7 through 9). The subintervals are $[i_1, j_1], \cdots, [i_s, j_s]$. Let

$$P = \{(u_1, v_1), (u_2, v_2), \cdots, (u_t, v_t)\}$$

where

$$u_1 < u_2 < \cdots < u_t < v_t < \cdots < v_2 < v_1 \quad \text{and} \quad t = \lfloor (s+1)/2 \rfloor.$$

We first consider the case $(1, n) \in P$. We can represent the order in H of the vertices of V_P and of the subintervals by the following string:

$$u_1 v_1 [i_{s-1}, j_{s-1}][i_s, j_s] v_2 u_2 [i_1, j_1][i_2, j_2] u_3 v_3 [i_{s-3}, j_{s-3}][i_{s-2}, j_{s-2}] v_4 u_4 [i_3, j_3][i_4, j_4] u_5 v_5 \cdots.$$

Of course, the vertices of the subintervals are permuted with H as they were in case (1).

From the ladder construction, there are four recognizable *types* of subintervals, two types on the left and two types on the right. While we could write down subscript formulas for each of the four types, for the cutwidth argument it is sufficient to consider the following four representatives of the four types: $[i_3, j_3]$, $[i_4, j_4]$, $[i_{s-3}, j_{s-3}]$ and $[i_{s-2}, j_{s-2}]$. The only edges that add to cw (H_k) are edges incident to vertices in V_P that pass over the kth subinterval in H. The diagram in Fig. 23 illustrates the *potential* for a vertex in V_P to have edges incident to some subinterval. For example, u_2 or v_2 might have one or more edges to subintervals $[i_1, j_1]$, $[i_s, j_s]$, $[i_2, j_2]$, and $[i_{s-1}, j_{s-1}]$. Since we are interested only in an upper bound on cutwidth, we ignore the possibility that the existence of some edge may preclude the existence of other edges.

We start with the type represented by subinterval $[i_3, j_3]$. An examination of the string for H together with Fig. 23 reveals the potential for edges passing over $[x_3, y_3]$ from u_3, v_3, u_4, v_4, u_5 and v_5 only. Hence, by inductive hypothesis,

$$\begin{aligned}\text{cw}([x_3, y_3]) &\leqq 6d + \text{cw}(H_3)\\ &\leqq 6d + Cd \log(\text{size}\{[i_3, j_3]\})\\ &\leqq 6d + Cd \log \tfrac{2}{3}n\\ &\leqq (Cd \log n) - 2d\\ &< (Cd \log n) - \text{def}\{[1, n]\}\end{aligned}$$

since each subinterval contains at most $\frac{2}{3}n$ vertices.

Similarly, consideration of the three types represented by $[i_4, j_4]$, $[i_{s-3}, j_{s-3}]$ and $[i_{s-2}, j_{s-2}]$ reveals that at most 6 vertices in V_P can have incident edges adding to the cutwidth of a subinterval. Hence, for all subintervals $[x_k, y_k]$ in H,

$$\text{cw}([x_k, y_k]) \leqq (Cd \log n) - \text{def}\{[1, n]\}.$$

Consideration of intervals in H between the subintervals (e.g., $[u_5, v_5]$) yields no worse an upper bound. Hence we conclude that cw $(H) \leqq (Cd \log n) - \text{def}\{[1, n]\}$.

The case in which $(1, n) \notin P$ is similar to the preceding case. The additional left or right subinterval cannot boost the cutwidth above $(Cd \log n) - \text{def}\{[1, n]\}$. Hence in all cases, (I.2) holds.

This completes the induction and the proof of the lemma. □

7. Performance. In this section, we analyze the time and space complexity of TRADEOFF. Of course, the complexity depends on the representation of data. While

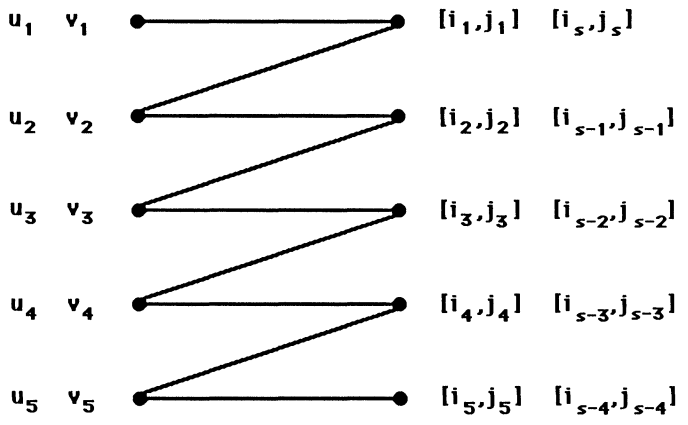

FIG. 23. *Proof of Lemma* 9.

we do not prescribe the details of the data representation, we do require that the representation make elementary operations efficient (i.e., constant time per edge or vertex). A place where this requirement is crucial is Algorithm 3 for finding a separating edge. To accomplish step 1 in linear time, it is necessary to be able to recognize the next (counterclockwise) edge of an interior face in constant time. It is easy to represent G so that this is possible. A reasonable representation puts all the edges adjacent to a vertex of G in a circular list in counterclockwise order; for each edge (u, v), there is a link from its position in the u-list to its position in the v-list. In this representation, the edges of an interior face can be traversed in constant time per edge.

First, we note that all operations of TRADEOFF performed on G except the recursive calls require linear time. From the description of the steps in § 3, all steps are clearly linear time except steps 6 and 9. From the description of the string construction, G' can be constructed in linear time from the $\{G_j\}$ (step 6). Similarly, the ladder construction can be accomplished in linear time (step 9). Hence, the entire algorithm excluding recursive calls can be implemented in linear time.

Let $T(n)$ be the time complexity of TRADEOFF. Let $n_1, n_2, \cdots, n_p$ be the sizes of the subintervals either in step 5 or in step 9, depending on which case holds. Then, $\sum_{k=1}^{p} n_k \leqq n$ and $n_k \leqq \frac{2}{3}n$, $1 \leqq k \leqq p$. By the result of the previous paragraph, there exists a constant c such that

$$T(n) \leqq cn + \sum_{k=1}^{p} T(n_k).$$

LEMMA 7. *If $T(1)$ is one unit of time, then for all $n > 1$,*

$$T(n) \leqq (c/\log \tfrac{2}{3})n \log n.$$

Proof. By induction on n. The lemma is certainly true for $n = 2$. Assume $n > 2$ and assume the truth of the lemma for values smaller than n. Then,

$$\begin{aligned}
T(n) &= cn + \sum_{k=1}^{p} T(n_k) \\
&\leqq cn + \sum_{k=1}^{p} \left(c \Big/ \log \frac{3}{2}\right) n_k \log n_k \\
&\leqq cn + \left(c \Big/ \log \frac{3}{2}\right) \sum_{k=1}^{p} n_k \log \frac{2}{3} n \\
&= cn + \left(c \Big/ \log \frac{3}{2}\right) n \log \frac{2}{3} n \\
&= cn + \left(c \Big/ \log \frac{3}{2}\right) n \log n - \left(c \Big/ \log \frac{3}{2}\right) n \log \frac{3}{2} \\
&= \left(c \Big/ \log \frac{3}{2}\right) n \log n.
\end{aligned}$$

The lemma follows by induction. □

The space requirements of TRADEOFF are clearly n times some small constant. We thus have the following.

THEOREM 8. TRADEOFF *has time complexity at most $C_1 n \log n$ and space complexity at most $C_2 n$, for small constants C_1, C_2.*

8. Conclusion. We have investigated tradeoffs between pagenumber and pagewidth that are significant in a VLSI context. Our main result is an algorithm for obtaining a book embedding for outerplanar graphs that is within a constant factor of optimal in VLSI area for the class of outerplanar graphs. While this near-optimality is not guaranteed for *individual* outerplanar graphs, we know of no example of an outerplanar graph for which our algorithm fails to obtain near-optimal area. Our algorithm embeds any d-valent n-vertex outerplanar graph in a two-page book with at most $Cd \log n$ pagewidth, $C = 8/(\log 3/2)$; the algorithm executes in time $O(n \log n)$. We show that at the cost of one additional page above optimal pagenumber, layouts of near-optimal cutwidth for outerplanar graphs can be obtained constructively.

Our result is applicable to the motivating DIOGENES design problem. The result bounds the area of a two-stack DIOGENES layout for a circuit represented by an outerplanar graph.

A fruitful area for further research is tradeoffs between pagenumber and pagewidth. It is not known how prevalent such tradeoffs are or whether dramatic tradeoffs exist for any pagenumber. In the context of VLSI problems, algorithms for embedding a graph in a bounded number of pages with pagewidth close to the cutwidth of the graph could be most practical. We do not know of any example where adding one or two pages above the pagenumber of G does not give us an embedding whose pagewidth is within a small constant factor of the pagewidth of G; there is hope that such algorithms exist for important classes of graphs. In particular, we believe that such an algorithm is possible for planar graphs. The algorithm would embed any d-valent planar graph in a B-page book with $Cd\sqrt{n}$ pagewidth where B and C are small constants. Our planar graph algorithm [4] or the similar algorithm of Yannakakis [11] might serve as the starting point for obtaining bounded pagenumber. The small pagewidth would depend on a version of Lipton and Tarjan's [7] planar separator theorem tailored to this problem. The result of Miller [8] on separating *cycles* in planar graphs is relevant here, though it is not sufficient by itself.

Acknowledgments. The author wishes to thank Arnold Rosenberg for suggesting the problem and for helpful discussions. He also wishes to thank the referee whose detailed report improved the presentation of the paper greatly.

REFERENCES

[1] F. BERNHART AND P. C. KAINEN, *The book thickness of a graph*, J. Combin. Theory Ser. B, 27 (1979), pp. 320–331.

[2] F. R. K. CHUNG, F. T. LEIGHTON AND A. L. ROSENBERG, *Embedding graphs in books: a layout problem with applications to VLSI design*, this Journal, 8 (1987), pp. 33–58.

[3] M. R. GAREY, D. S. JOHNSON, G. L. MILLER AND C. H. PAPADIMITRIOU, *The complexity of coloring circular arcs and chords*, this Journal, 1 (1980), pp. 216–227.

[4] L. S. HEATH, *Embedding planar graphs in seven pages*, 25th IEEE Symposium on Foundations of Computer Science, 1984, pp. 74–83.

[5] ———, *Algorithms for embedding graphs in books*, Ph.D. dissertation. Univ. of North Carolina at Chapel Hill, Dept. of Computer Science Technical Report TR 85-028, 1985.

[6] T. LENGAUER, *Upper and lower bounds on the complexity of the min-cut linear arrangement problem on trees*, this Journal, 3 (1982), pp. 99–113.

[7] R. J. LIPTON AND R. E. TARJAN, *A separator theorem for planar graphs*, SIAM J. Appl. Math., 36 (1979), pp. 177–189.

[8] G. L. MILLER, *Finding small simple cycle separators for 2-connected planar graphs*, 16th ACM Symposium on Theory of Computing, 1984, pp. 376–382.

[9] A. L. ROSENBERG, *The DIOGENES approach to testable fault-tolerant arrays of processors*, IEEE Trans. Comput., C-32 (1983), pp. 902–910.

[10] M. M. SYSLO, *Characterizations of outerplanar graphs*, Discrete Math., 26 (1979), pp. 47–53.

[11] M. YANNAKAKIS, *Four pages are necessary and sufficient for planar graphs*, 18th ACM Symposium on Theory of Computing, 1986, pp. 104–108.