SIAM J. COMPUT.
Vol. 9, No. 3, August 1980

APPLICATIONS OF A PLANAR SEPARATOR THEOREM*

RICHARD J. LIPTON† AND ROBERT ENDRE TARJAN‡

Abstract. Any n-vertex planar graph has the property that it can be divided into components of roughly equal size by removing only $O(\sqrt{n})$ vertices. This separator theorem, in combination with a divide-and-conquer strategy, leads to many new complexity results for planar graph problems. This paper describes some of these results.

Key words. algorithm, Boolean circuit complexity, divide-and-conquer, graph embedding, lower bounds, matching, maximum independent set, nonserial dynamic programming, pebbling, planar graphs, separator, space-time tradeoffs

1. Introduction. One efficient approach to solving computational problems is "divide-and-conquer" [1]. In this method, the original problem is divided into two or more smaller problems. The subproblems are solved by applying the method recursively, and the solutions to the subproblems are combined to give the solution to the original problem. Divide-and-conquer is especially efficient when the subproblems are substantially smaller than the original problem. In this paper we explore the efficient application of divide-and-conquer to a variety of problems on planar graphs. We employ the following theorem.

THEOREM 1 [20]. *Let G be any n-vertex planar graph with nonnegative vertex costs summing to no more than one. Then the vertices of G can be partitioned into three sets A, B, C, such that no edge joins a vertex in A with a vertex in B, neither A nor B has total vertex cost exceeding $\frac{2}{3}$, and C contains no more than $2\sqrt{2}\sqrt{n}$ vertices. Furthermore A, B, C can be found in $O(n)$ time.*

In the special case of equal-cost vertices, this theorem becomes

COROLLARY 1. *Let G be any n-vertex planar graph. The vertices of G can be partitioned into three sets A, B, C, such that no edge joins a vertex in A with a vertex in B, neither A nor B contains more than $2n/3$ vertices, and C contains no more than $2\sqrt{2}\sqrt{n}$ vertices.*

Corollary 1 verifies a conjecture of Ungar [32], who obtained a similar result but with a bound of $O(\sqrt{n}\log n)$ on the size of C. It is easy to construct examples to show that Corollary 1 is tight to within a constant factor in the worst case [20].

Each section of this paper describes a different use of Theorem 1. The results range from an efficient algorithm for finding maximum independent sets in planar graphs to lower bounds on the complexity of planar Boolean circuits. In each case, the only property of planar graphs that we use is Theorem 1, and our results generalize easily to any class of graphs which can be separated into small components by removing a small number of vertices. For instance, by employing the following result of Sider, we can extend our results to graphs of arbitrary genus.

LEMMA 1 [2], [30]. *Let G be any n-vertex graph of genus $g>0$. Then there exists a subset of no more than $\sqrt{2n}$ vertices whose removal reduces the genus of G by at least one.*

* Received by the editors August 9, 1978, and in final revised form September 11, 1979.

† Computer Science Department, Yale University, New Haven, Connecticut. Now at Computer Science Division, University of California, Berkeley, California 94720. This research was partially supported by the U.S. Army Research Office, Grant No. DAAG 29-76-G-0338.

‡ Computer Science Department, Stanford University, Stanford, California 94305. This research was partially supported by National Science Foundation Grant MCS-75-22870 and by the Office of Naval Research Contract N00014-76-C-0688.

THEOREM 2. *If G is an n-vertex graph of genus $g > 0$, there is a subset of no more than $g\sqrt{2n}$ vertices whose removal leaves a planar graph.*

Proof. The proof is by induction on g employing Lemma 1. □

We state our results only for the planar case, since it seems the most interesting, and leave as an exercise the extension of these results to graphs of higher genus.

2. Approximation algorithms for *NP*-complete problems.

Divide-and-conquer in combination with Theorem 1 can be used to rapidly find good approximate solutions to certain *NP*-complete problems on planar graphs. As an example we consider the *maximum independent set* problem, which asks for a maximum number of pairwise non-adjacent vertices in a planar graph. We need the following generalization of Theorem 1.

THEOREM 3. *Let G be an n-vertex planar graph with nonnegative vertex costs summing to no more than one and let $0 \leqq \varepsilon \leqq 1$. Then there is some set C of $O(\sqrt{n/\varepsilon})$ vertices whose removal leaves G with no connected component of cost exceeding ε. Furthermore the set C can be found in $O(n \log n)$ time.*

Proof. If $\varepsilon \leqq 1/\sqrt{n}$, let C contain all the vertices of G. Then the theorem holds. Otherwise, apply the following algorithm to G.

Initialization. Let $C = \emptyset$.

General Step. Find some connected component K in G minus C with cost exceeding ε. Apply Theorem 1 to K, producing a partition A_1, B_1, C_1 of its vertices. Let $C = C \cup C_1$.

Repeat the general step until G minus C has no component with cost exceeding ε.

The effect of one execution of the general step is to divide the component K into smaller components, each with no more than two-thirds the cost of K. Consider all components that arise during the course of the algorithm. Assign a *level* to each component as follows. If the component exists when the algorithm halts, the component has level zero. Otherwise the level of the component is one greater than the maximum level of the components formed when it is split by the general step. With this definition, any two components on the same level are vertex-disjoint.

Each level one component has cost greater than ε, since it is eventually split by the general step. Thus, for $i \geqq 1$, each level i component has cost at least $(\frac{3}{2})^{i-1}\varepsilon$. Since the total cost of G is at most one, the total number of components of level i is at most $(\frac{2}{3})^{i-1}/\varepsilon$. In particular, the maximum level k must satisfy $1 \leqq (\frac{2}{3})^{k-1}/\varepsilon \leqq (\frac{2}{3})^{k-1}\sqrt{n}$, which means $k \leqq (\log_{3/2} n)/2 + 1$. Since the time to split a component is linear in its number of vertices, and since any two components on the same level are vertex-disjoint, the total running time of the algorithm is $O(n \log n)$.

It remains for us to bound the size of the set C produced by the algorithm. Let $K_1, K_2, \ldots, K_l$, of sizes $n_1, n_2, \ldots, n_l$, respectively, be the components of some level $i \geqq 1$. The number of vertices added to C by splitting $K_1, K_2, \cdots, K_l$ is bounded by $2\sqrt{2}\sum_{j=1}^{l}\sqrt{n_j}$. We have $l \leqq (\frac{2}{3})^{i-1}/\varepsilon$ and $\sum_{j=1}^{l} n_j \leqq n$. For fixed l, the sum $\sum_{j=1}^{l}\sqrt{n_j}$ subject to $\sum_{j=1}^{l} n_j \leqq n$ is maximized by setting $n_j = n/l$ for $1 \leqq j \leqq l$; thus $2\sqrt{2}\sum_{j=1}^{l}\sqrt{n_j} \leqq 2\sqrt{2}\sqrt{nl} \leqq 2\sqrt{2}\sqrt{n/\varepsilon}(\frac{2}{3})^{(i-1)/2}$. It follows that $|C| \leqq \sum_{i=1}^{\infty} 2\sqrt{2}\sqrt{n/\varepsilon}(\frac{2}{3})^{(i-1)/2} = O(\sqrt{n/\varepsilon})$. □

The following algorithm uses Theorem 3 to find an approximately maximum independent set I in a planar graph $G = (V, E)$. The algorithm uses a function $k(n)$ to be chosen later.

Step 1. Apply Theorem 3 to G with $\varepsilon = k(n)/n$ and each vertex having cost $1/n$ to find a set of vertices C of size $O(n/\sqrt{k(n)})$ whose removal leaves no connected component with more than $k(n)$ vertices.

Step 2. In each connected component of G minus C, find a maximum independent set by checking every subset of vertices for independence. Form I as a union of maximum independent sets, one from each component.

Let I^* be a maximum independent set of G. The restriction of I^* to one of the connected components formed when C is removed from G can be no larger than the restriction of I to the same component. Thus $|I^*|-|I|=O(n/\sqrt{k(n)})$. Since G is planar, G is four-colorable, and $|I^*|\geqq n/4$. Thus $(|I^*|-|I|)/|I^*|=O(1/\sqrt{k(n)})$, and the relative error in the size of I tends to zero with increasing n as long as $k(n)$ tends to infinity with increasing n.

Step 1 of the algorithm requires $O(n \log n)$ time by Theorem 2. Step 2 requires $O(n_i\, 2^{n_i})$ time on a connected component of n_i vertices. The total time required by Step 2 is thus

$$O\left(\max\left\{\sum_{i-1}^{n} n_i 2^{n_i} \,\middle|\, \sum_{i=1}^{n} n_i=n \text{ and } O\leqq n_i\leqq k(n)\right\}\right)=O\left(\frac{n}{k(n)}k(n)2^{k(n)}\right)=O(n2^{k(n)}).$$

Hence the entire algorithm requires $O(n\cdot\max\{\log n, 2^{k(n)}\})$ time. If we shoose $k(n)=\log n$, we get an $O(n^2)$-time algorithm with $O(1/\sqrt{\log n})$ relative error. If we choose $k(n)=\log\log n$, we get an $O(n\log n)$ algorithm with $O(1/\sqrt{\log\log n})$ relative error.

3. Nonserial dynamic programming.

3. Nonserial dynamic programming. Many *NP*-complete problems, such as the maximum independent set problem, the graph coloring problem, and others, can be formulated as *nonserial dynamic programming problems* [3], [27]. Such a problem is of the following form: maximize the objective function $f(x_1,\cdots,x_n)$, where f is given as a sum of terms $f_k(\cdot)$, each of which is a function of only a subset of the variables. We shall assume that all variables x_i take on values from the same finite set S, and that the values of the terms $f_k(\cdot)$ are given by tables. Associated with such an objective function f is an interaction graph $G=(V,E)$, containing one vertex v_i for each variable x_i in f, and an edge joining x_i and x_j for any two variables x_i and x_j which appear in a common term $f_k(\cdot)$.

We can formulate the maximum independent set problem as a nonserial dynamic programming problem as follows. Let $G=(V,E)$ be an undirected graph. For each vertex v_i, $1\leqq i\leqq n$, let x_i be an associated variable which can assume a value of either zero or one. Let the objective function $f(x_1,x_2,\cdots,x_n)$ be defined by

$$f(x_1,x_2,\cdots,x_n)=\sum_{(v_i,u_j)\in E} f_e(x_i,x_j)+\sum_{i=1}^{n} x_i,$$

where $f_e(x_i,x_j)=-\infty$ if $x_i=x_j=1$, $f_e(x_i,x_j)=0$ otherwise. Then a maximum independent set in G corresponds to an assignment of values 0, 1 to $x_1,x_2,\cdots,x_n$ which maximizes f; $x_i=1$ means x_i is in the independent set, $x_i=0$ means x_i is not in the independent set. Other graph problems can be formulated similarly. Note that G is the interaction graph of f.

By trying all possible values of the variables, a nonserial dynamic programming problem can be solved in $2^{O(n)}$ time. We shall show that if the interaction graph of the problem is planar, the problem can be solved in $2^{O(\sqrt{n})}$ time. This means that substantial savings are possible when solving typical *NP*-complete problems restricted to planar graphs. Note that if the interaction graph of f is planar, no term $f_k(\cdot)$ of f can contain more than four variables, since the complete graph on five vertices is not planar.

In order to describe the algorithm, we need one additional concept. The *restriction* of an objective function $f=\sum_{k=1}^{m} f_k$ to a set of variables $x_{i_1},\cdots,x_{i_j}$ is the objective function $f'=\sum\{f_k|f_k$ depends only upon $x_{i_1},\cdots,x_{i_j}\}$.

Given an objective function $f(x_1, \cdots, x_n) = \sum_{k=1}^{m} f_k$ and a subset S of the variables $x_1, \cdots, x_n$ which are constrained to have specific values, the following algorithm solves the problem: maximize f subject to the constraints on the variables in S. In the presentation, we do not distinguish between the variables $x_1, \cdots, x_n$ and the corresponding vertices in the interaction graph.

Step 1. If $n < 100$, solve the problem by exhaustively trying all possible assignments to the unconstrained variables. Otherwise, go to Step 2.

Step 2. Apply Corollary 1 to the interaction graph G of f. Let A, B, C be the resulting vertex partition. Let f_1 be the restriction of f to $A \cup C$ and let f_2 be the restriction of f to $B \cup C$. For each possible assignment of values to the variables in $C - S$, perform the following steps:

(a) Maximize f_1 with the given values for the variables in $C \cup S$ by applying the method recursively;

(b) maximize f_2 with the given values for the variables in $C \cup S$ by applying the method recursively;

(c) combine the solutions to (a) and (b) to obtain a maximum value of f with the given values for the variables in $C \cup S$.

Choose the assignment of values to variables in $C \cup S$ which maximizes f and return the appropriate value of f as the solution.

The correctness of this algorithm is obvious. If $n \geqq 100$, the algorithm solves at most $2^{O(\sqrt{n})}$ subproblems in Step 2, since C is of $O(\sqrt{n})$ size. Each subproblem contains at most $2n/3 + 2\sqrt{2}\sqrt{n} \leqq 29n/30$ variables. Thus if $t(n)$ is the running time of the algorithm, we have $t(n) \leqq O(n) + 2^{O(\sqrt{n})} \cdot t(29n/30)$ if $n \geqq 100$, $t(n) = O(1)$ if $n < 100$. An inductive proof shows that $t(n) \leqq 2^{O(\sqrt{n})}$.

4. Pebbling.

The following one-person game arises in register allocation problems [28], the conversion of recursion to iteration [23], and the study of time-space tradeoffs [4], [12], [25]. Let $G = (V, E)$ be a directed acyclic graph with maximum in-degree k. If (v, w) is an edge of G, v is a *predecessor* of w and w is a *successor* of v. The game involves placing pebbles on the vertices of G according to certain rules. A given step of the game consists of either placing a pebble on an empty vertex of G (called *pebbling* the vertex) or removing a pebble from a previously pebbled vertex. A vertex may be pebbled *only* if all its predecessors have pebbles. The object of the game is to successively pebble each vertex of G (in any order) subject to the constraint that at most a given number of pebbles are ever on the graph simultaneously.

It is easy to pebble any vertex of an n-vertex graph in n steps using n pebbles. We are interested in pebbling methods which use fewer than n pebbles but possibly many more than n steps. It is known that any vertex of an n-vertex graph can be pebbled with $O(n/\log n)$ pebbles [12] (where the constant depends upon the maximum in-degree), and that in general no better bound is possible [25]. We shall show that if the graph is planar, only $O(\sqrt{n})$ pebbles are necessary, generalizing a result of [25]. An example of Cook [4] shows that no better bound is possible for planar graphs.

THEOREM 4. *Any n-vertex planar acyclic directed graph with maximum in-degree k can be pebbled using $O(\sqrt{n} + k \log_2 n)$ pebbles.*

Proof. Let $\alpha = 2\sqrt{2}$ and $\beta = \frac{2}{3}$. Let G be the graph to be pebbled. Use the following recursive pebbling procedure. If $n = 1$, pebble the single vertex of G. If $n > 1$, find a vertex partition A, B, C satisfying Corollary 1. Pebble the vertices of G in topological order.[1] To pebble a vertex v, delete all pebbles except those on C. For each predecessor

[1] That is, an order such that if v is a predecessor of w, v is pebbled before w.

u of v, let $G(u)$ be the subgraph of G induced by the set of vertices with pebble-free paths to u. Apply the method recursively to each $G(u)$ to pebble all predecessors of v, leaving a pebble on each such predecessor. Then pebble v.

If $p(n)$ is the maximum number of pebbles required by this method on any n-vertex graph, then

$$p(1)=1,$$
$$p(n)\leqq \alpha\sqrt{n}+k+p(\lfloor 2n/3\rfloor) \quad \text{if } n>1.$$

An inductive proof shows that $p(n)$ is $O(\sqrt{n}+k \log_2 n)$. □

It is also possible to obtain a substantial reduction in pebbles while preserving a polynomial bound on the number of pebbling steps, as the following theorem shows.

THEOREM 5. *Any n-vertex planar acyclic directed graph with maximum in-degree k can be pebbled using $O(n^{2/3}+k)$ pebbles in $O(n^{5/3})$ time.*

Proof. Let C be a set of $O(n^{2/3})$ vertices whose removal leaves G with no weakly connected component[2] containing more than $n^{2/3}$ vertices. Such a set C exists by Theorem 2. The following pebbling procedure places pebbles permanently on the vertices of C. Pebble the vertices of G in topological order. To pebble a vertex v, pebble each predecessor u of v and then pebble v. To pebble a predessor u, delete all pebbles from G except those on vertices in C or on predecessors of v. Find the weakly connected component in G minus C containing u. Pebble all vertices in this component, in topological order.

The total number of pebbles required by this strategy is $O(n^{2/3})$ to pebble vertices in C plus $n^{2/3}$ to pebble each weakly connected component plus k to pebble predecessors of the vertex v to be pebbled. We can bound the number of pebbling steps as follows. To pebble a vertex v requires $d_I(v)n^{2/3}+1$ steps, where $d_I(v)$ is the in-degree of vertex v. The total pebbling time is thus $n+\sum_{v\in V} d_I(v)n^{2/3}\leqq n+(3n-3)n^{2/3}=O(n^{5/3})$. □

5. Lower bounds on Boolean circuit size.

A *Boolean circuit* is an acyclic directed graph such that each vertex has in-degree zero or two, the predecessors of each vertex are ordered, and corresponding to each vertex v of in-degree two is a binary Boolean operation b_v. With each vertex of the circuit we associate a Boolean function which the vertex computes, defined as follows. With each of the k vertices v_i of in-degree zero (inputs) we associate a variable x_i and an identity function $f_{v_i}(x_i)=x_i$. With each vertex w of in-degree two having predecessors u, v we associate the function $f_w=b_w(f_u, f_v)$. The circuit computes the set of functions associated with its vertices of out-degree zero (outputs).

We are interested in obtaining lower bounds on the size (number of vertices) of Boolean circuits which compute certain common and important functions. Using Theorem 1 we can obtain such lower bounds under the assumption that the circuits are planar. Any circuit can be converted into a planar circuit by the following steps. First, embed the circuit in the plane, allowing edges to cross if necessary. Next, replace each pair of crossing edges by the crossover circuit illustrated in Figure 1. It follows that any lower bound on the size of planar circuits is also a lower bound on the total number of vertices and edge crossings in any planar representation of a nonplanar circuit. In a technology for which the total number of vertices and edge crossings is a reasonable measure of cost, our lower bounds imply that it may be expensive to realize certain commonly used functions in hardware.

[2] A *weakly connected component* of a directed graph is a connected component of the undirected graph formed by ignoring edge directions.

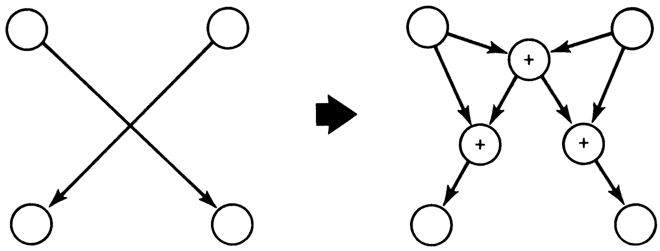

FIG. 1. *Elimination of a crossover by use of three "exclusive or" gates. Reference* [11] *contains a crossover circuit which uses only "and" and "not".*

A *superconcentrator* is an acyclic directed graph with m inputs and m outputs such that any set of k inputs and any set of k outputs are joined by k vertex-disjoint paths, for all k in the range $1 \leqq k \leqq m$.

THEOREM 6. *Any m-input, m-output planar superconcentrator contains at least* $m^2/72$ *vertices.*

Proof. Let G be an m-input, m-output planar superconcentrator. Assign to each input and output of G a cost of $1/(2m)$, and to every other vertex a cost of zero. Let A, B, C be a vertex partition satisfying Theorem 1 on G (ignoring edge directions). Suppose C contains p inputs and outputs. Without loss of generality, suppose that A is no more costly than B, and that A contains no more outputs than inputs. A contains between $2m/3-p$ and $m-p/2$ inputs and outputs. Hence A contains at least $m/3-p/2$ inputs and at most $m/2-p/4$ outputs. B contains at least $m-p-(m/2-p/4)=m/2-3p/4$ outputs. Let $k=\min\{\lceil m/3-p/2\rceil, \lceil m/2-3p/4\rceil\}$. Since G is a superconcentrator, any set of k inputs in A and any set of k outputs in B are joined by k vertex-disjoint paths. Each such path must contain a vertex in C which is neither an input nor an output. Thus $2\sqrt{2}\sqrt{n}-p \geqq \min\{m/3-p/2,\ m/2-3p/4\} \geqq m/3-p$, and $n \geqq m^2/72$. □

The property of being a superconcentrator is a little too strong to be useful in deriving lower bounds on the complexity of interesting functions. However, there are weaker properties which still require $\Omega(m^2)$ vertices. Let $G=(V,E)$ be an acyclic directed graph with m numbered inputs $v_1, v_2, \cdots, v_m$ and m numbered outputs $w_1, w_2, \cdots, w_m$. G is said to have the *shifting property* if, for any k in the range $1 \leqq k \leqq m$, any l in the range $0 \leqq l \leqq m-k$, and any subset of k sources $\{v_{i_1}, \cdots, v_{i_k}\}$ such that $i_1, i_2, \cdots, i_k \leqq m-l$, there are k vertex-disjoint paths joining the set of inputs $\{v_{i_1}, \cdots, v_{i_k}\}$ with the set of outputs $\{w_{i_1+l}, \cdots, w_{i_k+l}\}$.

THEOREM 7. *Let G be a planar acyclic directed graph with the shifting property. Then G contains at least* $\lfloor m/2 \rfloor^2/162$ *vertices.*

Proof. Suppose that G contains n vertices. Assign a cost of $1/m$ to each of the first $\lfloor m/2 \rfloor$ inputs and to each of the last $\lfloor m/2 \rfloor$ outputs of G, and a cost of zero to every other vertex of G. Call the first $\lfloor m/2 \rfloor$ inputs and the last $\lfloor m/2 \rfloor$ outputs of G *costly*. Let A, B, C be a vertex partition satisfying Theorem 1 on G (ignoring edge directions).

Without loss of generality, suppose that A is no more costly than B, and that A contains no more costly outputs than costly inputs. Let A' be the set of costly inputs in A, B' the set of costly outputs in B, p the number of costly inputs and outputs in C, and q the number of costly inputs and outputs in A. Then $2\lfloor m/2 \rfloor/3-p \leqq q \leqq \lfloor m/2 \rfloor - p/2$. Hence $|A'| \geqq q/2 \geqq \lfloor m/2 \rfloor/3-p/2$. Also

$$\begin{aligned}|A'|\cdot|B'| &\geqq |A'|\cdot(\lfloor m/2 \rfloor - p-(q-|A'|))\\ &\geqq q/2\cdot(\lfloor m/2 \rfloor - p - q/2).\end{aligned}$$

The function $x(\lfloor m/2\rfloor - p - x)$ for $\lfloor m/2\rfloor/3 - p/2 \leqq x \leqq \lfloor m/2\rfloor/2 - p/4$ is minimized either at $x = \lfloor m/2\rfloor/3 - p/2$ or at $x = \lfloor m/2\rfloor/2 - p/4$. If $x = \lfloor m/2\rfloor/3 - p/2$, we have $x(\lfloor m/2\rfloor - p - x) = 2\lfloor m/2\rfloor^2/9 - p\lfloor m/2\rfloor/2 + p^2/4$. If $x = \lfloor m/2\rfloor/2 - p/4$, we have $x(\lfloor m/2\rfloor - p - x) = \lfloor m/2\rfloor^2/4 - p\lfloor m/2\rfloor/2 + 3p^2/16$. It follows that $|A'| \cdot |B'| \geqq 2\lfloor m/2\rfloor^2/9 - p\lfloor m/2\rfloor/2$.

For $v_i \in A'$, $w_j \in B'$, and l in the range $1 \leqq l \leqq \lfloor m/2\rfloor$, call v_i, w_j, l a *match* if $j - i = l$. For every $v_i \in A'$ and $w_j \in B'$ there is exactly one value of l which produces a match; hence the total number of matches for all possible v_i, w_j, l is $|A'| \cdot |B'| \geqq 2\lfloor m/2\rfloor^2/9 - p\lfloor m/2\rfloor/2$. Since there are only $\lfloor m/2\rfloor$ values of l, some value of l produces at least $2\lfloor m/2\rfloor/9 - p/2$ matches. Thus, for $k = 2\lfloor m/2\rfloor/9 - p/2$, there is some value of l and some set of k inputs $A'' = \{v_{i_1}, v_{i_2}, \cdots, v_{i_k}\} \subseteq A'$ such that $B'' = \{w_{i_1+l}, w_{i_2+l}, \cdots, w_{i_k+l}\} \subseteq B'$. Since G has the shifting property, there must be k vertex-disjoint paths between A'' and B''. But each such path must contain a vertex of C which is neither an input nor an output. Hence $2\sqrt{2}\sqrt{n} - p \geqq 2\lfloor m/2\rfloor/9 - p/2$, and $n \geqq \lfloor m/2\rfloor^2/162$. □

A *shifting circuit* is a Boolean circuit with *m primary inputs* $x_1, x_2, \cdots, x_m$, zero or more *auxiliary inputs*, and m outputs $z_1, z_2, \cdots, z_m$, such that, for any k in the range $0 \leqq k \leqq m$, there is some assignment of the constants 0, 1 to the auxiliary inputs so that output z_{i+k} computes the identity function x_i, for $0 \leqq i \leqq m - k$. The *Boolean convolution* of two Boolean vectors $(x_1, x_2, \cdots, x_m)$ and $(y_1, y_2, \cdots, y_m)$ is the vector $(z_2, z_3, \cdots, z_{2m})$ given by $z_k = \sum_{i+j=k} x_i y_j$.

COROLLARY 2. *Any planar shifting circuit has at least* $\lfloor m/2\rfloor^2/162$ *vertices.*

Proof. Any shifting circuit has the shifting property. See [31], [33]. □

COROLLARY 3. Any planar circuit for computing Boolean convolution has at least $\lfloor m/2\rfloor^2/162$ vertices.

Proof. A circuit for computing Boolean convolution is a shifting circuit if we regard $x_1, \cdots, x_m$ as the primary inputs and $z_2, \cdots, z_{m+1}$ as the outputs. □

COROLLARY 4. Any planar circuit for computing the product of two m bit binary integers has at least $\lfloor m/2\rfloor^2/162$ vertices.

Proof. A circuit for multiplying two m-bit binary integers is a shifting circuit. □

The last result of this section is an $\Omega(m^4)$ lower bound on the size of any planar circuit for multiplying two $m \times m$ Boolean matrices. We shall assume that the inputs are x_{ij}, y_{ij} for $1 \leqq i, j \leqq m$ and the outputs are z_{ij} for $1 \leqq i, j \leqq m$. The circuit computes $Z = X \cdot Y$, where $Z = (z_{ij})$, $X = (x_{ij})$, and $Y = (y_{ij})$. We use the following property of circuits for multiplying Boolean matrices, called the *matrix concentration property* [31], [33]. For any k in the range $1 \leqq k \leqq m^2$, any set $\{x_{i_r j_r} | 1 \leqq r \leqq k\}$ of k inputs from X, and any permutation σ of the integers one through m, there exist k vertex-disjoint paths from $\{x_{i_r j_r} | 1 \leqq r \leqq k\}$ to $\{z_{i_r \sigma(j_r)} | 1 \leqq r \leqq k\}$. Similarly, for any k in the range $1 \leqq k \leqq m^2$, any set $\{y_{i_r j_r} | 1 \leqq r \leqq k\}$ of k inputs from Y, and any permutation σ of one through m, there exist k vertex-disjoint paths from $\{y_{i_r j_r} | 1 \leqq r \leqq k\}$ to $\{z_{\sigma(i_r) j_r} | 1 \leqq r \leqq k\}$.

THEOREM 8. *Any planar circuit G for multiplying two $m \times m$ Boolean matrices contains at least cm^4 vertices, for some positive constant c.*

Proof. This proof is somewhat involved, and we make no attempt to maximize the constant factor. Suppose G contains n vertices, and that m is even. Assign a cost of $1/(4m^2)$ to each input x_{ij} and each input y_{ij}, a cost of $1/(2m^2)$ to each output z_{ij}, and a cost of zero to every other vertex. By Theorem 2, there is a partition A, B, C of the vertices of G such that neither A nor B has total cost exceeding $\frac{1}{2}$, no edge joins a vertex in A with a vertex in B, and C contains no more than $c_1\sqrt{n}$ vertices. Without loss of generality, suppose that B contains no fewer outputs than A, and that A contains no fewer inputs x_{ij} than inputs y_{ij}. Then B contains at least $(m^2 - c_1\sqrt{n})/2$ outputs, which contribute at least $1/4-c_1\sqrt{n}/(4m^2)$ to the cost of B. Thus inputs contribute at most $1/4-c_1\sqrt{n}/(4m^2)$ to the cost of B, and B contains at most $m^2+c_1\sqrt{n}$ inputs. A contains at least $2m^2-(m^2+c_1\sqrt{n})-c_1\sqrt{n}=m^2-2c_1\sqrt{n}$ inputs, of which at least $m^2/2-c_1\sqrt{n}$ are inputs x_{ij}. One of the following cases must hold.

Case 1. A contains at least $3m^2/5$ inputs x_{ij}. *Let* p be the number of columns of X which contain at least $4m/7$ elements of A. Then $pm+(m-p)(4m/7)\geqq 3m^2/5$, and $p\geqq m/15$. Let q be the number of columns of Z which contain at least $4m/9$ elements of B. Then $qm+(m-q)(4m/9)\geqq m^2/2-c_1\sqrt{n}/2$, and $q\geqq m/10-9c_1\sqrt{n}/(10m)$.

Let $k=\min\{m/15,\ m/10-9c_1\sqrt{n}/(10m)\}$. Choose any k columns of X, each of which contains at least $4m/7$ elements of A. Match each such column of X with a column of Z which contains at least $4m/9$ elements of B. For each pair of matched columns x_{*i}, z_{*j}, select a set of $4m/7+4m/9-m=m/63$ rows l such that x_{li} is in A and z_{lj} is in B. Such a selection gives a set of $km/63$ elements in $X\cap A$ and a set of $km/63$ elements in $Z\cap B$ which must be joined by $km/63$ vertex-disjoint paths, since G has the matrix concentration property. Each such path must contain a vertex of C. Thus $km/63\leqq c_1\sqrt{n}$, which means either $m^2/(15\cdot 63)\leqq c_1\sqrt{n}$ (i.e., $(m^2/(15\cdot 63c_1))^2\leqq n$) or $m/63(m/10-9c_1\sqrt{n}/(10m))\leqq c_1\sqrt{n}$ (i.e., $(m^2/(9\cdot 71c_1))^2\leqq n$).

Case 2. A contains fewer than $3m^2/5$ inputs x_{ij}. Then A contains at least $2m^2/5-2c_1\sqrt{n}$ inputs y_{ij}. Let S be the set of $m/2$ columns of Z which contain the most elements in B.

Subcase 2a. S contains at least $3m^2/10$ elements in B. Let p be the number of columns of X which contain at least $4m/9$ elements of A. Then $pm+4(m-p)m/9\geqq m^2/2-c_1\sqrt{n}$, and $p\geqq m/10-9c_1\sqrt{n}/(5m)$. Let q be the number of columns of S which contain at least $4m/7$ elements of B. Then $qm+4(m/2-q)m/7\geqq 3m^2/10$, and $q\geqq m/30$. A proof similar to that in Case 1 shows that $n\geqq \mathrm{c}m^4$ for some positive constant c.

Subcase 2b. S contains fewer than $3m^2/10$ elements in B. Then the $m/2$ columns of Z not in S contain at least $m^2/5-c_1\sqrt{n}/2$ elements in B. Let q be the number of columns of Z not in S which contain at least $m/10$ elements in B. Then $qm+(m/2-q)(m/10)\geqq m^2/5-c_1\sqrt{n}/2$, and $q\geqq m/6-5c_1\sqrt{n}/(9m)$. If $0\geqq q\geqq m/6-5c_1\sqrt{n}/(9m)$, then $(3m^2/(10c_1))^2\geqq n$. Hence assume $q>0$. Then all columns in S must contain at least $m/10$ elements in B, and $2m/3-5c_1\sqrt{n}/(9m)$ columns of Z must contain at least $m/10$ elements in B.

Let p be the number of columns of Y which contain at least $m/25$ elements of A. Then $pm+(m-p)(m/25)\geqq 2m^2/5-2c_1\sqrt{n}$, and $p\geqq 3m/8-25c_1\sqrt{n}/(12m)$.

For any input $y_{ij}\in A$ and integer l in the range $-m+1\leqq l\leqq m-1$, call y_{ij}, l a *match* if $z_{i+l,j}\in B$. By the previous computations, there are at least $2m/3-5c_1\sqrt{n}/(9m)+3m/8-25c_1\sqrt{n}/(12m)-m=m/25-95c_1\sqrt{n}/(36m)=m/25-c_2\sqrt{n}/m$ columns j such that y_{*j} contains $m/25$ elements of A and z_{*j} contains $m/10$ elements of B. Each such column produces $m^2/250$ matches; thus the total number of matches is at least $m^3/6250-mc_2\sqrt{n}/250$. Since there are only $2m-1$ values of l, some value of l produces at least $k=m^2/12{,}500-c_2\sqrt{n}/500$ matches. Since G has the matrix concentration property, this set of matches corresponds to a set of k elements in $Y\cap A$ and a set of k elements in $Z\cap B$ which must be joined by k vertex-disjoint paths. Each such path must contain a vertex in C. Thus $k\leqq c_1\sqrt{n}$, which means $m^4/(12{,}500(c_1+c_2/500))^2\leqq n$.

In all cases $n\geqq cm^4$ for some positive constant c. Choosing the minimum c over all cases gives the theorem for even m. The theorem for odd m follows immediately. $\square$

The bounds in Theorems 6–8 and Corollaries 2–4 are tight to within a constant factor. We leave the proof of this fact as an exercise.

6. Embedding of data structures. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be undirected graphs. An *embedding* of G_1 in G_2 is a one-to-one map $\phi: V_1 \to V_2$. The *worst-case proximity* of the embedding is $\max\{d_2(\phi(v), \phi(w)) | \{v, w\} \in E_1\}$, where $d_2(x, y)$ denotes the distance between x and y in G_2. The *average proximity* of the embedding is $(1/|E_1|)/\sum\{d_2(\phi(v), \phi(w)) | \{v, w\} \in E_1\}$. These notions arise in the following context. Suppose we wish to represent some kind of data structure by another kind of data structure, in such a way that if two records are logically adjacent in the first data structure, their representations are close together in the second. We can model the data structures by undirected graphs, with vertices denoting records and edges denoting logical adjacencies. The representation problem is then a graph embedding problem in which we wish to minimize worst-case or average proximity. See [5], [18], [26] for research in this area.

THEOREM 9. *Any planar graph with maximum degree k can be embedded in a binary tree so that the average proximity is $O(k)$.*

Proof. Let G be an n-vertex graph of maximum degree k. Embed G in a binary tree T by using the following recursive procedure. If G has one vertex v, let T be the tree of one vertex, the image of v. Otherwise, apply Corollary 1 to find a partition A, B, C of the vertices of G. Let v be any vertex in C (if C is empty, let v be a vertex in A). Embed the subgraph of G induced by $A \cup C - \{v\}$ in a binary tree T_1 by applying the method recursively. Embed the subgraph of G induced by B in a binary tree T_2 by applying the method recursively. Let T consist of a root (the image of v) with two children, the root of T_1 and the root of T_2. Note that the tree T constructed in this way has exactly n vertices.

Let $h(n)$ be the maximum depth of a tree T of n vertices produced by this algorithm. Then

$$\begin{aligned} &h(n) < 100 && \text{if } n < 100,\\ &h(n) \leqq h(2n/3 + 2\sqrt{2}\sqrt{n} - 1) + 1 \leqq h(29n/30) + 1 && \text{if } n \geqq 100. \end{aligned}$$

It follows that $h(n)$ *is* $O(\log n)$.

Let $G = (V, E)$ be an n-vertex graph to which the algorithm is applied, let G_1 be the subgraph of G induced by $A \cup C$, and let G_2 be the subgraph induced by B. If $s(G) = \sum\{d_2(\phi(v), \phi(w)) | (v, w) \in E\}$, then $s(G) = 0$ if $n = 1$, and $s(G) \leqq s(G_1) + s(G_2) + 2k|C|h(n)$ if $n > 1$. This follows from the fact that any edge of G not in G_1 or G_2 must be incident to a vertex of C.

If $s(n)$ is the maximum value of $s(G)$ for any n-vertex graph G, then

$$\begin{aligned} &s(1) = 0;\\ &s(n) \leqq \max\{s(i) + s(n - i - 1) + ck\sqrt{n}\log n \,|\, n/3 - 2\sqrt{2}\sqrt{n} \leqq i \leqq 2n/3 + 2\sqrt{2}\sqrt{n}\}\\ &\qquad\qquad \text{if } n > 1, \text{ for some positive constant } c. \end{aligned}$$

An inductive proof shows that $s(n)$ is $O(kn)$.

If G is a connected n-vertex graph embedded by the algorithm, then G contains at least $n - 1$ edges, and the average proximity is $O(k)$. If G is not connected, embedding each connected component separately and combining the resulting trees arbitrarily achieves an $O(k)$ average proximity. □

It is natural to ask whether *any* graph of bounded degree can be embedded in a binary tree with $O(1)$ average proximity. (Graphs of unbounded degree cannot be so embedded; a star consisting of a single vertex adjacent to $n - 1$ other vertices requires $\Omega(\log n)$ proximity.) Such is not the case, and in fact the property of being embeddable in a binary tree with $O(1)$ average proximity is closely related to the property of having a good separator. To make this statement more precise, let S be a class of graphs. The class S has an *$f(n)$-separator theorem* if there exist constants $\alpha < 1$, $\beta > 0$ such that the vertices of any n-vertex graph in S can be partitioned into three sets A, B, C such that $|A|$, $|B| \leqq \alpha n$, $|C| \leqq \beta f(n)$, and no vertex in A is adjacent to any vertex in B.

THEOREM 10. *Let S be any class of graphs of maximum degree k closed under the subgraph relation (i.e., if $G_1 \in S$ and G_2 is a subgraph of G_1, then $G_2 \in S$). Suppose S satisfies an $n/(\log n)^{2+\varepsilon}$ separator theorem for some fixed ε. Then any graph in S can be embedded in a binary tree with $O(k)$ average proximity.*

Proof. Similar to the proof of Theorem 9.

THEOREM 11. *Let $G = (V, E)$ be any graph of n vertices and m edges which is embeddable in a binary tree T with average proximity p. Then V can be partitioned into three sets A, B, C such that $|A|$, $|B| \leqq 2n/3$, $|C| \leqq cmp/\log n$ for some positive constant c, and no edge joins a vertex in A with a vertex in B.*

Proof. We can assume $m \geqq 2n/3$; otherwise the theorem is immediate. Let v be a vertex whose removal divides T into two or three connected components, each containing fewer than $2n/3$ vertices. Such a vertex can be found by initializing v to be the root of T and repeating the following step until it is no longer applicable: if some child w of v has at least $2n/3$ descendants, replace v by w. Let A be the set of vertices in G corresponding to the largest component of T when v is removed, let C be the set of vertices in $V - A$ adjacent to at least one vertex in A, and let $B = V - A - C$. By the choice of v and A, $|A| \leqq 2n/3$ and $|B| \leqq 2n/3$. Let $T(A)$, $T(B)$, $T(C)$ be the sets of vertices in T corresponding to A, B, C respectively. Since T is a binary tree, the number of vertices in $T(B) \cup T(C)$ within a distance of i from at least one vertex in $T(A)$ is at most $2^i - 1$. Thus the average proximity of the embedding of G in T is at least $|C| \cdot \lfloor \log_2 |C| \rfloor / (2m)$. This means $|C| \log |C| = O(mp)$, and $|C| = O(mp/\log n)$. □

Erdös, Graham, and Szemerédi [7] have shown that for c a large enough constant, almost all graphs of cn edges cannot be separated into small components without removing $\Omega(n)$ vertices. It follows from Theorem 11 that almost all graphs of cn edges require $\Omega(\log n)$ average proximity when embedded in binary trees.

7. Maximum matching. Let $G = (V, E)$ be an undirected graph. A *matching* $M \subseteq E$ is a set of edges no two of which have a common endpoint. A *maximum cardinality* matching is a matching M such that $|M|$ is maximum. If each edge $e \in E$ has an associated real-valued *weight* $w(e)$, a *maximum weight matching* is a matching M such that $\sum_{e \in M} w(e)$ is maximum. By using Corollary 1, we can find maximum cardinality matchings in planar graphs in $O(n^{3/2})$ time and maximum weight matchings in $O(n^{3/2} \log n)$ time. For arbitrary graphs, the best known algorithms require $O(\sqrt{n}\, m \log \log n)$ time to find maximum cardinality matchings [14] and $O(mn \log n)$ time to find maximum weight matchings [8], where $m = |E|$. For planar graphs, these bounds are $O(n^{3/2} \log \log n)$ and $O(n^2 \log n)$, respectively.

To describe the method, we need a few ideas from matching theory. If M is a matching in a graph C, an *unmatched vertex* is a vertex incident to no edge of M. An *alternating path* is a simple path or simple cycle whose edges are alternately in M and not in M. The *net weight* of an alternating path is the total weight of its unmatched edges minus the total weight of its matched edges. An alternating path is *augmenting* if its net weight is positive and it is either a cycle or each of its first and last edges is either in M or incident to an unmatched vertex. Given an augmenting path, we can increase the weight of the matching by adding to M all previously unmatched edges on the path and deleting from M all previously matched edges on the path. Conversely, if there is no augmenting path, then M is of maximum weight. The next lemma provides a way to update a maximum weight matching when a single vertex is added to a graph.

LEMMA 2. *Let $G=(V, E)$ be an undirected graph with edge weights $w(e)$, let $v \in V$, and let $G-v$ be the subgraph of G induced by the vertex set $V-\{v\}$. Suppose M is a maximum weight matching in $G-v$. If G contains no augmenting path (with respect to M) with v as one endpoint, then M is a maximum weight matching of G. Otherwise, let P be the edge set of an augmenting path of maximum net weight. Then $M \oplus P = M \cup P-(M \cap P)$ is a maximum weight matching in G.*

Proof. Let M_0 be a maximum weight matching in G. Consider $M \oplus M_0 = M \cup M_0-(M \cap M_0)$. Every vertex in G is incident to at most two edges of $M \cup M_0$; thus $M \oplus M_0$ consists of a set of simple cycles and simple paths in G, each of which is an alternating path with respect to M_0. Any augmenting path in $M \oplus M_0$ must have v as an endpoint, or else M would not be of maximum weight in $G-v$. (Note that v is incident to at most one edge in $M \oplus M_0$). Thus $M \oplus M_0$ contains at most one augmenting path, and such a path has v as one endpoint. The lemma follows. □

Given a suitable representation of a maximum weight matching M in $G-v$, a maximum weight matching M_0 in G can be found in $O(m \log n)$ time by applying Lemma 2; see [8] for details. Thus applying Lemma 2 to a planar graph requires $O(n \log n)$ time. In the maximum cardinality case, all the weights are one, and application of Lemma 2 requires $O(m)$ time on an arbitrary graph, $O(n)$ time on a planar graph [13].

The following recursive algorithm makes use of Corollary 1 and Lemma 2 to find maximum weight matchings.

Step 1. If G contains at most one vertex, return the empty set as a maximum weight matching.

Step 2. Otherwise, apply Corollary 1 to G. Let A, B, C be the resulting vertex partition and let G_A, G_B be the subgraphs of G induced by the vertex sets A, B, respectively. Apply the algorithm recursively to find maximum weight matchings M_A in G_A, M_B in G_B. Let $M=M_A \cup M_B$, $S=A \cup B$.

Step 3. Add C one vertex at a time to S. Each time a vertex is added to S, apply Lemma 2 to replace M by a maximum weight matching in G_S, the subgraph of G induced by the vertex set S. Stop when $S=V$.

After Step 2, $M=M_A \cup M_B$ is a maximum weight matching of $G_{A \cup B}$. It follows from Lemma 2 that after Step 3, M is a maximum weight matching of G_V. If $t(n)$ is the running time of the algorithm on an n-vertex graph, then

$$t(l)=c_1;$$

$$t(n) \leqq \max\{t(n_1)+t(n_2)+c_2 n^{3/2} \log n \mid n_1+n_2 \leqq n;\ n_1, n_2 \leqq 2n/3\} \qquad \text{if } n>1,$$

where c_1 and c_2 are suitable positive constants, since $|C|=O(\sqrt{n})$. An inductive proof shows that $t(n)=O(n^{3/2} \log n)$. In the maximum cardinality case, the algorithm requires only $O(n^{3/2})$ time.

8. Remarks.

Theorem 1 and its corollaries have applications beyond those in this paper. For instance, the planar separator theorem can be used to generalize George's "nested dissection" method [10] for carrying out sparse Gaussian elimination on a system of linear equations whose sparsity structure corresponds to a square grid. The generalized method solves any linear system whose sparsity structure corresponds to an n-vertex planar graph in $O(n^{3/2})$ time and $O(n \log n)$ space [19]. Theorem 2 can be employed to give a rather complicated $O(\log n)$ time, $O(n)$-space solution [21] to the closest-point searching problem in two dimensions, sometimes called the post office problem [16]. The previously best solutions to this problem required either $O(\log n)$ time and $O(n^2)$ space [29], or $O((\log n)^2)$ time and $O(n)$ space [6], [29]. Recently Kirkpatrick [15] has discovered a simple $O(\log n)$-time, $O(n)$-space solution which does not use the separator theorem. We leave further applications of the separator theorem to the reader.

Although most sparse graphs do not have good separators, there are other classes besides planar graphs and graphs of fixed genus which do (see e.g. [19]). The results discussed in this paper generalize to any such class. In some of the problems we have examined, such as graph embedding (§ 6) and sparse Gaussian elimination [19], the existence of good separators is not only a sufficient but also a necessary condition for efficient solution of the problem. This phenomenon deserves more study, and suggests that for certain graph problems it may be valuable to define the concept of "usefully sparse" as meaning that a graph has good separators.

Acknowledgment. We would like to thank Hal Gabow for his very perceptive comments and for suggesting the results in § 7. A preliminary version of this paper was presented at the Eighteenth Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, 1977.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Efficient Computer Algorithms*, Additon-Wesley, Reading, MA., 1974.
[2] M. O. Albertson and J. P. Hutchinson, *On the independence ratio of a graph*, J. Graph Theory, 2 (1978), pp. 1–8.
[3] U. Bertele and F. Brioschi, *Nonserial Dynamic Programming*, Academic Press, New York, 1972.
[4] S. A. Cook, *An observation on time-storage tradeoff*, Proc. Fifth Annual ACM Symp. on Theory of Computing (1973), pp. 29–33.
[5] R. A. DeMillo, S. C. Eisenstat and R. J. Lipton, *Preserving average proximity in arrays*, Comm. ACM, 21(1978), pp. 228–230.
[6] D. Dobkin and R. J. Lipton, *Multidimensional searching problems*, this Journal, 5 (1976), pp. 181–186.
[7] P. Erdös, R. L. Graham and E. Szemerédi, *On sparse graphs with dense long paths*, Comp. and Math. with Appl., 1 (1975), pp. 365–369.
[8] H. Gabow, *An efficient implementation of Edmonds' algorithm for maximum weight matching on graphs*, Technical Report CU-CS-075-75, University of Colorado, Boulder, Colorado (1975).
[9] M. R. Garey, D. S. Johnson, F. P. Preparata, and R. E. Tarjan, *Triangulating a simple polygon*, Informat. Processing, Letters, 7 (1978), pp. 175–179.
[10] J. A. George, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345–363.
[11] L. Goldschlager, *The monotone and planar circuit value problems are log space complete for P*, ACM SIGACT News 9, 2 (1977), pp. 25–29.
[12] J. Hopcroft, W. Paul and L. Valiant, *On time versus space*, J. Assoc. Comput. Mach., 24 (1977), pp. 332–337.
[13] T. Kameda and I. Munro, *A $O(VE)$ algorithm for maximum matching of graphs*, Computing 12 (1974), pp. 91–98.
[14] O. Kariv, *An $O(n^{2.5})$ algorithm for finding a maximum matching on a general graph*, Ph.D dissertation, Weizmann Institute of Science, Rehovot, Israel, 1976.
[15] D. Kirkpatrick, private communication, 1979.
[16] D. E. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching*, Addison-Wesley, Reading, MA., 1973.
[17] D. Kozen, *On parallelism in Turing machines*, Proc. Seventeenth Annual Symp. on Foundations of Computer Science, 1976, pp. 89–97.

[18] R. J. LIPTON, S. C. EISENSTAT, AND R. A. DEMILLO, *Space and time hierarchies for control structures and data structures*, J. Assoc. Comput. Mach., 23 (1976), pp. 720–732.
[19] R. J. LIPTON, D. J. ROSE, AND R. E. TARJAN, *Generalized nested dissection*, SIAM J. Numer. Anal., 16 (1979), pp. 346–358.
[20] R. J. LIPTON AND R. E. TARJAN, *A separator theorem for planar graphs*, SIAM J. Appl. Math., 36(1979), pp. 177–189.
[21] ———, *Applications of a planar separator theorem*, Proc. 18th Annual Symp. on Foundations of Computer Science (1977), pp. 162–170.
[22] H. C. MARTIN AND G. F. CAREY, *Introduction to Finite Element Analysis*, McGraw-Hill, New York, 1973.
[23] M. S. PATERSON AND C. E. HEWITT, *Comparative schematology*, Record of Project MAC Conf. on Concurrent Systems and Parallel Computation (1970), pp. 119–128.
[24] M. S. Paterson, *Tape bounds for time-bounded Turing machines*, J. Comput. System Sci., 6 (1972), pp. 116–124.
[25] W. J. PAUL, R. E. TARJAN, AND J. R. CELONI, *Space bounds for a game on graphs*, Math. Systems Theory 10 (1977), pp. 239–251.
[26] A. L. ROSENBERG, *Managing storage for extendible arrays*, this Journal, 4 (1975), pp. 287–306.
[27] A. ROSENTHAL, *Nonserial dynamic programming is optimal*, Proc. Ninth Annual ACM Symp. on Theory of Computing (1977), pp. 98–105.
[28] R. SETHI, *Complete register allocation problems*, this Journal, 4 (1975), pp. 226–248.
[29] M. J. SHAMOS, *Geometric complexity*, Proc. Seventh Annual ACM Symp. on Theory of Computing (1975), pp. 224–233.
[30] N. SIDER, *Partial colorings and limiting chromatic numbers*, Ph.D. dissertation, Syracuse University, Syracuse, NY (1971).
[31] L. G. VALIANT, *On non-linear lower bounds in computational complexity*, Proc. Seventh Annual ACM Symp. on Theory of Computing (1975), pp. 45–53.
[32] P. UNGAR, *A theorem on planar graphs*, J. London Math. Soc., 26 (1951), pp. 256–262.
[33] L. G. VALIANT, *Graph-theoretic arguments in low-level complexity*, Computer Science Dept., University of Edinburgh, 1977.