APPLICATIONS OF A PLANAR SEPARATOR THEOREM

by

Richard J. Lipton and Robert E. Tarjan

STAN-CS-77-628
OCTOBER 1977

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

Applications of a Planar Separator Theorem

Richard J. Lipton [*]
Computer Science Department
Yale University
New Haven, Connecticut 06520

Robert Endre Tarjan [**]
Computer Science Department
Stanford University
Stanford, California 94305

August, 1977

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.

Keywords: algorithm, Boolean circuit complexity, divide-and-conquer, geometric complexity, graph embedding, lower bounds, maximum independent set, non-serial dynamic programming, pebbling, planar graphs, separator, space-time tradeoffs.

[*] This research partially supported by the U.S. Army Research Office, Grant No. DAAG 29-76-G-0338.

[**] This research partially supported by National Science Foundation grant MCS-75-22870 and by the Office of Naval Research contract N00014-76-C-0688.

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 [14] the following theorem is proved.

Theorem 1. Let G be any n-vertex planar graph with non-negative 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 $2/3$, 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.

Theorem 1 and its corollary open the way for efficient application of divide-and-conquer to a variety of problems on planar graphs. In this paper we explore a number of such applications. Each section of the paper describes a different use of divide-and-conquer. The results range from an efficient approximation algorithm for finding maximum independent sets in planar graphs to lower bounds on the complexity of planar Boolean circuits. The last section mentions two additional applications whose description is too lengthy to be included in this paper.

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.

Theorem 2. Let G be an n-vertex planar graph with non-negative vertex costs summing to no more than one and let $0 \leq \epsilon \leq 1$. Then there is some set C of $O(\sqrt{n/\epsilon})$ 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. 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 Corollary 1 to K , producing a partition A_1 , B_1 , C_1 of its vertices. Let $C = C \cup C_1$. If one of A_1 and B_1 (say A_1) has cost exceeding two-thirds the cost of K, apply Theorem 1 to the subgraph of G induced by the vertex set A_1 , producing a partition A_2, B_2, C_2 of A_1 . Let $C = C \cup C_2$.

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 and each with no more than two-thirds as many vertices as K. Consider all components which arise during the course of the algorithm. Assign a <u>level</u> 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. It follows that, for $i \geq 1$, each level i component has cost at least $(3/2)^{i-1}$ and contains at least $(3/2)^i$ vertices. Since the total cost of G is at most one, the total number of components of level i is at most $(2/3)^{i-1}/\epsilon$.

The total running time of the algorithm is $O(\sum \{|K| \mid K$ is a component split by the general step$\})$. Since a component of level i contains at least $(3/2)^i$ vertices, the maximum level k must satisfy $(3/2)^k < n$, or $k < \log_{3/2} n$. Since components in each level are vertex-disjoint, the total running time of the algorithm is $O(n \log_{3/2} n) = O(n \log n)$.

The total size of the set C produced by the algorithm is bounded by

$$O(\sum \{\sqrt{|K|} \mid K \text{ is a component split by the general step}\})$$

$$\leq O\left(\sum_{i=1}^{\lfloor \log_{3/2} n \rfloor} \max\left\{\sum_{j=1}^{\lfloor (2/3)^{i-1}/\epsilon \rfloor} \sqrt{n_j} \;\middle|\; \sum_{j=1}^{\lfloor (2/3)^{i-1}/\epsilon \rfloor} n_j \leq n \text{ and } n_j \geq 0\right\}\right)$$

$$\leq O\left(\sum_{i=1}^{\infty} \frac{(2/3)^{i-1}}{\epsilon} \sqrt{\frac{n\epsilon}{(2/3)^{i-1}}}\right) = O\left(\sqrt{n/\epsilon} \sum_{i=0}^{\infty} (2/3)^{i/2}\right)$$

$$= O(\sqrt{n/\epsilon}) \; . \quad \square$$

The following algorithm uses Theorem 2 to find an approximately maximum independent set I in a planar graph $G = (V,E)$.

Step 1. Apply Theorem 2 to G with $\epsilon = (\log \log n)/n$ and each vertex having cost $1/n$ to find a set of vertices C containing $O(n/\sqrt{\log \log n})$ vertices whose removal leaves no connected component with more than $\log \log n$ 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 $|I^*| - |I| = O(n/\sqrt{\log \log n})$. Since G is planar, G is four-colorable, and $|I^*| \geq n/4$. Thus $(|I^*|-|I|)/|I^*| = O(1/\sqrt{\log \log n})$, and the relative error in the size of I tends to zero 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

$$O\left(\max\left\{\sum_{i=1}^{n} n_i 2^{n_i} \;\middle|\; \sum_{i=1}^{n} n_i = n \text{ and } 0 < n_i < \log \log n\right\}\right) =$$

$$O\left(\frac{n}{\log \log n} (\log \log n) 2^{\log \log n}\right) = O(n \log n) .$$ Hence the entire algorithm requires $O(n \log n)$ time.

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 [2,20]. Such a problem is of the following form: minimize the objective function $f(x_1,\ldots,x_n)$, 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 $f_k(\cdot)$.

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 $x_{i_1},\ldots,x_{i_j}$ is the objective function

$f' = \sum \{f_k \mid f_k \text{ depends upon one or more of } x_{i_1},\ldots,x_{i_j}\}$.

Given an objective function $f(x_1, . . ., x_n) = \sum_{k=1}^{m} f_k$ and a subset S of the variables $x_1,\ldots,x_n$ 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 $x_1,\ldots,x_n$ and the corresponding vertices in the interaction graph.

Step 1. If $n < 9$, 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-S which maximizes f and return the appropriate value of f as the solution.

The correctness of this algorithm is obvious. If $n > 9$, 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 $2n/3 + 2\sqrt{2}\sqrt{n} \leq 29n/30$ variables. Thus if $t(n)$ is the running time of the algorithm, we have $t(n) \leq O(n \log n) + 2^{O(\sqrt{n})} \cdot t(29n/30)$ if $n > 9$, $t(n) = O(1)$ if $n \leq 9$. An inductive proof shows that $t(n) \leq 2^{O(\sqrt{n})}$.

4. Pebbling.

The following one-person game arises in register allocation problems [21], the conversion of recursion to iteration [16], and the study of time-space tradeoffs [4,10,18]. 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 [10] (where the constant depends upon the maximum in-degree), and that in general no better bound is possible [18]. We shall show that if the graph is planar, only $O(\sqrt{n})$ pebbles are necessary, generalizing a result of [18]. An example of Cook [4] shows that no better bound is possible for planar graphs.

Theorem 3. 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 $a = 2\sqrt{2}$ and $\beta = 2/3$. Let G be the graph to be pebbled. Use the following recursive pebbling procedure. If $n \leq n_0$, where $n_0 = (\alpha/(1-\beta))^2$, pebble all vertices of G without deleting pebbles. If $n > n_0$, find a vertex partition A, B, C satisfying Corollary 1. Pebble the vertices of G in topological order.[*] To pebble a vertex v, delete all pebbles except those on C. For each predecessor 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

$$p(n) = n \text{ if } n \leq n_0 ,$$

$$p(n) \leq \alpha\sqrt{n} + k + p(2n/3 + \alpha\sqrt{n}) \quad \text{if} \quad n>n_0 .$$

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

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

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

Proof. Let C be a set of $O(n^{2/3})$ vertices whose removal leaves G with no weakly connected component [*] 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 predecessor 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. The total number of pebbling steps is at most $O(k \cdot n \cdot n^{2/3}) = O(kn^{5/3})$. □

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

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 non-planar 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.

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 $1 \le k \le m$.

Theorem 5. 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 $2\sqrt{2}\sqrt{n} - p \ge \min\{m/3 - p/2, m/2 - 3p/4\} \ge m/3 - p$, and $n \ge m^2/72$. □

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 $v_1, v_2,$. ● ☺✝O and m numbered outputs $w_1, w_2, ..$● [illegible] .
G is said to have the shifting property if, for any k in the range $1 \leq k \leq m$, any ℓ in the range $0 < \ell < m-k$, and any subset of k sources $\{v_{i_1}, \ldots, v_{i_k}\}$ such that $i_1, i_2, \ldots, i_k \leq m-\ell$, there are k vertex-disjoint paths joining the set of inputs $\{v_{i_1},$. ▨▨ $V_{i_k}\}$ with the set of outputs $\{v_{i_1+\ell}, \ldots, v_{i_k+\ell}\}$.

Theorem 6. 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 $2\lfloor m/2 \rfloor/3 - p \leq q \leq \lfloor m/2 \rfloor - p/2$. Hence $|A'| \geq q/2 \geq \lfloor m/2 \rfloor/3 - p/2$.
Also

$$\begin{aligned}|A'|\cdot|B'| &\geq |A'|\cdot(\lfloor m/2\rfloor - p - (q - |A'|)) \\ &\geq q/2\cdot(\lfloor m/2\rfloor - p - q/2) \\ &\geq (\lfloor m/2\rfloor/3 - p/2)(\lfloor m/2\rfloor - p - \lfloor m/2\rfloor/3 + p/2) \\ &= (\lfloor m/2\rfloor/3 - p/2)(2\lfloor m/2\rfloor/3 - p/2) \\ &\geq 2\lfloor m/2\rfloor^2/q - p\lfloor m/2\rfloor/2 .\end{aligned}$$

For $v_i \in A'$, $w_j \in B'$, and ℓ in the range $1 \leq \ell \leq \lfloor m/2\rfloor$, call v_i, w_j, ℓ a match if $j-i = \ell$. 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 $|A'|\cdot|B'| \geq 2\lfloor m/2\rfloor^2/q - p\lfloor m/2\rfloor/2$. Since there are only $\lfloor m/2\rfloor$ values of a, some value of ℓ produces at least $2\lfloor m/2\rfloor/q - p/2$ matches. Thus, for $k = 2\lfloor m/2\rfloor/q - p/2$, there is some value of ℓ and some set of k inputs $A'' = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\} \subseteq A'$ such that $B'' = \{w_{i_1+\ell}, w_{i_2+\ell}, \ldots, w_{i_k+\ell}\} \subseteq B'$. 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 $2\sqrt{2}\sqrt{n} - p \geq 2\lfloor m/2\rfloor/q - p/2$, and $n \geq \lfloor m/2\rfloor^2/162$. □

A shifting circuit is a Boolean circuit with m primary inputs $x_1, x_2, \ldots, x_m$, zero or more auxiliary inputs, and m outputs $z_1, z_2, \ldots, z_m$, such that, for any k in the range $0 \leq k \leq m$, 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 $0 \leq i \leq m-k$. The Boolean convolution of two Boolean vectors $(x_1, x_2, \ldots, x_m)$ and $(y_1, y_2, \ldots, y_m)$ is the vector $(z_2, z_3, \ldots, z_{2m})$ 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 [23,24]. $\square$

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 $x_1, \ldots, x_m$ as the primary inputs and $z_2, \ldots, z_{m+1}$ as the outputs, $\square$

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. $\square$

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 $1 \le i,j \le m$ and the outputs are z_{ij} for $1 \le i,j \le m$. 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 [23,24]. For any k in the range $1 \leq k \leq n^2$, any set $\{x_{i_r j_r} \mid 1 \leq r \leq k\}$ of k inputs from X, and any permutation σ of the integers one through n, there exist k vertex-disjoint paths from $\{x_{i_r j_r} \mid 1 \leq r \leq k\}$ to $\{z_{i_r \sigma(j_r)} \mid 1 \leq r \leq k\}$. Similarly, for any k in the range $1 \leq k \leq n^2$, any set $\{y_{i_r j_r} \mid 1 \leq r \leq k)$ of k inputs from Y, and any permutation σ of one through n, there exist k vertex-disjoint paths from $\{y_{i_r j_r} \mid 1 \leq r \leq k\}$ to $\{z_{\sigma(i_r) j_r} \mid 1 \leq r \leq k\}$.

Theorem 7. 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. There is a partition A, B, C of the vertices of G such that neither A nor B has total cost exceeding $1/2$, no edge joins a vertex in A with a vertex in B, and C contains no more than $2\sqrt{2}\sqrt{n}/(1 - \sqrt{2/3}) = c_1\sqrt{n}$ vertices. This is a corollary of Theorem 1; see [14]. 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 $2m^3 \cdot (m^2 + c_1\sqrt{n}) - c_1\sqrt{n} = m^2 - 2c_1\sqrt{n}$ 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 $pm + (m-p)(4m/7) \geq 3m^2/5$, and $p \geq m/15$. Let q be the number of columns of Z which contain at least $4m/9$ elements of B. Then $qm + (m-q)(4m/q) \geq m^2/2 - c_1\sqrt{n}/2$, and $q \geq m/10 - 9c_1\sqrt{n}/(10m)$.

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 $x_{\ell i}$ is in A and $z_{\ell j}$ 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 $km/63 \leq c_1\sqrt{n}$, which means either $m^2/(15\cdot 63) \leq c_1\sqrt{n}$ (i.e., $(m^2/(15\cdot 63c_1))^2 \leq n$) or $m/63(m/10 - 9c_1\sqrt{n}/(10m)) \leq c_1\sqrt{n}$ (i.e., $(m^2/(9\cdot 69c_1))^2 \leq n$).

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 $pm + 4(m-p)m/9 \geq m^2/2 - c_1\sqrt{n}$, and $p \geq m/10 - 9c_1\sqrt{n}/(5m)$. Let q be the number of columns of Z which contain at least $4m/7$ elements of B . Then $qm + 4(m/2 - q)m/7 \geq 3m^2/10$, and $q \geq m/30$. A proof similar to that in Case 1 shows that $n \geq cm^4$ 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 $qm + (m/2 - q)(m/10) \geq m^2/5 - c1\sqrt{n}/2$, and $q \geq m/6 - 5c_1\sqrt{n}/(9m)$. If $0 \geq q \geq m/6 - 5c_1\sqrt{n}/(9m)$, then $(3m^2/(10c_1))^2 \geq n$. 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 $pm + (m-p)(m/25) > 2m^2/5 - 2c_1\sqrt{n}$, and $p \geq 3m/8 - 25c_1\sqrt{n}/(12m)$.

For any input $y_{ij} \in A$ and integer ℓ in the range $-n+1 \leq \ell \leq n-1$, call y_{ij}, ℓ a match if $z_{i+\ell,j} \in B$. By the previous computations, there are at least $2m/3 - 5c_1\sqrt{n}/(9m) + 3m/8 - 25c_1\sqrt{n}/(12m) - m = m/25 - 95c_1\sqrt{n}/(36m) = m/25 - c_1\sqrt{n}/m$ 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 $m^3/6250 - mc_1\sqrt{n}/250$. 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 $k \leq c_1\sqrt{n}$, which means $m^4 /(12{,}500(c_1 + c_2/500))^2 \leq n$.

In **all** cases $n \geq cm^4$ 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.** ☐

The bounds in Theorems 5 - 7 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 $G1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be undirected graphs. An embedding of G_1 in G_2 is a one-to-one map $\phi: V1 \to V_2$. The worst-case proximity of the embedding is $\max\{d_2(\phi(v),\phi(w)) \mid \{v,w\} \in E_1\}$, where $d_2(x,y)$ denotes the distance between x and y in G_2 . The average proximity of the embedding is $\frac{1}{|E_1|} \Sigma \{d_2(\phi(v),\phi(w)) \mid \{v,w\} \in E_1\}$. 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 [5,13,19] for research in this area.

Theorem 8. Any planar graph with maximum degree k can be embedded in a binary tree so that the average proximity is a constant depending only upon k .

Proof. Let G be an n-vertex planar graph. 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 any vertex). 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

$$h(n) \le 9 \quad \text{if } n \le 9 ,$$

$$h(n) \le h(2n/3 + 2\sqrt{2}\sqrt{n} - 1) < h(29n/30) \quad \text{if } n > 9 .$$

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 Gl be the subgraph of G induced by $A \cup C$, and let G_2 be the subgraph induced by B . If $s(G) = \sum \{d_2(\phi(v),\phi(w)) \mid (v,w) \in E\}$, then s(G) = 0 if n = 1 , and $s(G) \le s(G_1) + s(G_2) + k\,|C|\,h(n)$ if n > 1 . This follows from the fact that any edge of G not in Gl 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

$$s(1) = 0 ;$$

$$s(n) \le \max(s(i) + s(n-i) + ck\sqrt{n} \log n \mid n/3 - 2\sqrt{2}\sqrt{n} \le i \le 2n/3 + 2\sqrt{2}\sqrt{n} \}$$

if n > 1 , for some positive constant c .

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; the star of Figure 2 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 $|A|, |B| \leq \alpha n$, $|C| \leq \beta f(n)$, and no vertex in A is adjacent to any vertex in B.

Let S be any class of graphs of bounded degree closed under the subgraph relation (i.e., if $G_2 \in S$ and G1 is a subgraph of G_2, then $G_1 \in S$). Suppose S satisfies an $ng(n)/(\log n)^2$ separator theorem for some non-decreasing function $g(n)$. Using the idea in the proof of Theorem 8, it is not hard to show that any graph in S can be embedded in a binary tree with $O(g(n))$ average proximity. Conversely, suppose any graph in a class S can be embedded in a binary tree with $O(g(n))$ average proximity. Then S satisfies an $ng(n)/\log n$ separator theorem. In particular, if S satisfies no $o(n)$ -separator theorem, then embedding the graphs of S in binary trees requires $\Omega(\log n)$ average proximity. Erdős, Graham, and Szemerédi [7] have shown the existence of a class of graphs of bounded degree having no $o(n)$ -separator theorem.

7. The Post Office Problem.

In [11], Knuth mentions the following problem: given n points (post offices) in the plane; determine, for any new point (house), which post office it is nearest. Any preprocessing of the post offices is allowed before the houses are processed. Shamos [22] gives an $O(\log n)$ -time, $O(n^2)$ -space algorithm and an $O((\log n)^2)$ -time, $O(n \log n)$ -space algorithm. See also [6]. Using Theorem 2 we can give a solution which requires $O(\log n)$ time and $O(n)$ space, both minimum if only binary decisions are allowed,

A polygon is a connected, open planar region bounded by a finite set of line-segments. (For convenience, we allow the point at infinity to be an endpoint of a line segment; thus a line is a line segment.) A polygon partition of the plane is a partition of the plane into polygons and bounding line segments. A triangulation of the plane is a polygon partition, all of whose polygons are bounded by three line segments. A triangulation of a polygon partition is a refinement of the partition into a triangulation, Two polygons in a polygon partition are adjacent if their boundaries share a line segment. A set of polygons is connected if any two polygons in the set are joined by a sequence of adjacent polygons.

We shall solve the following triangle problem: given an n-triangle triangulation and a point, determine which triangle or line segment of the triangulation contains the point. The post office problem can be reformulated as a triangle problem; the set of points closest to each post office forms a polygon [22]. We shall make use of the following lemma, which we do not prove.

Lemma 1. Any n-polygon partition has a refinement whose total number of triangles is bounded by n plus the number of line segments bounding non-triangles plus a constant (a line segment bounding two non-triangles counts twice in this bound).

We shall build up a sequence of more and more complicated (but more and more efficient) algorithms, the last of which is the desired one.

Theorem 9. Given an $O(\log n)$ -time, $O(n^{1+\epsilon})$ -space algorithm for the triangle problem with $\epsilon > 0$, one can construct an $O(\log n)$ -time, $O(n^{1+2\epsilon/3})$ -space algorithm.

Proof. Let T be a triangulation and v be a vertex for which the triangle problem is to be solved. By Theorem 2 there is a set of $O(n^{2/3})$ triangles C_0 whose removal from T leaves no connected set of more than $O(n^{2/3})$ triangles.

Merge pairs of adjacent triangles which are not in C_0 to form a polygon partition P_0 . P_0 contains at most $O(n^{2/3})$ line segments, since each such line segment must be a bounding segment of a triangle in T . Find a triangulation T_0 of P_0 with $O(n^{2/3})$ triangles, which exists by Lemma 1. Using the given algorithm, determine which triangle or line segment of T_0 contains v .

If v is in some triangle of C_0 , the problem is solved. Otherwise, v is known to be in some connected set C_i of triangles in T minus C_0 . Merge pairs of adjacent triangles which are not in C_i to form a polygon partition Pi . Since P_i contains at most $O(n^{2/3})$ line segments, there is a triangulation T_i of P_i with $O(n^{2/3})$ 'triangles. Using'" the given algorithm, determine which triangle or line segment of T_i contains v . This solves the problem.

The sets C_i , polygon partitions P_i , and triangulations T_i are all precomputed. Thus the time required by the algorithm is $O(\log n^{2/3})$ to discover which triangle of T_0 contains v , plus $O(\log n^{2/3})$ to discover which triangle of T_i contains v . The total time is thus $O(\log n)$. The total space is $\sum_{*} O(|T_i|^{\alpha}) \leq O(n^{1+2\epsilon/3})$. □

Corollary 5. For any $\epsilon > 0$ there is an $O(\log n)$ -time, $O(n^{1+\epsilon})$ -space algorithm for the triangle problem.

Proof. Immediate from Theorem 9, using the $O(\log n)$ -time, $O(n^2)$ -space algorithm of [22] as a starting point. □

Theorem 10. There is an $O(\log n)$ -time, $O(n)$ -space solution to the triangle problem.

Proof. Let T be a triangulation and v a vertex for which the triangle problem is to be solved. If T contains no more than n_0 triangles, where n_0 is a sufficiently large constant, determine which triangle contains v by testing v against each line segment bounding a triangle of T . Otherwise, let C be a set of $O(n^{3/5})$ triangles whose removal from T leaves no connected set of more than $O(n^{4/5})$ triangles. Group the connected sets of triangles in T minus C_0 into sets C_i , each containing within a constant factor of $n^{4/5}$ triangles.

Merge pairs of adjacent triangles which are not in C_0 to form a polygon partition P_0 . P_0 contains at most $O(n^{3/5})$ line segments.

Find a triangulation T_0 of P_0 with $O(n^{3/5})$ triangles. Using an $O(\log n)$ -time, $O(n^{7/6})$ -space algorithm, determine which triangle of T_0 contains v .

If v is some triangle of C_0 , the problem is solved. Otherwise v is known to be in some set C_i . Merge pairs of adjacent triangles which are not in C_i to form a polygon partition P_i . Each line segment bounding a non-triangular polygon of P_i must bound a triangle of C_0 . Thus there is a triangulation T_i of P_i containing $|C_i| + O(n^{3/5})$ triangles. Apply the algorithm recursively to discover which triangle of T_i contains v . This solves the problem.

The sets C_i , polygon partitions P_i , and triangulations T_i are all precomputed. If t(n) is the worst-case time required by the algorithm on an n-triangle triangulation, then

$$t(n) = O(1) \quad \text{if} \quad n \le n_0,$$

$$t(n) = t(O(n^{4/5})) + O(\log n) \quad \text{otherwise.}$$

An inductive proof shows that t(n) is $O(\log n)$ if n_0 is chosen sufficiently large.

If s(n) is the worst-case storage space required by the algorithm on an n-triangle triangulation, then

$$s(t) = O(1) \quad \text{if} \quad n \le n_0 \quad ,$$

$$s(n) \le O(n^{7/10}) + \max\{\Sigma\ s(n_i + O(n^{3/5}))\ |\ \Sigma\, n_i \le n \ \text{ and } \ c_1 n^{4/5} \le n_i \le c_2 n^{4/5}\}$$

for some positive constants cl and c_2 .

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

The preprocessing time required by the algorithm of Theorem 10 is $O(n \log n)$. See [22]. We do not advocate this algorithm as a practical one, but its existence suggests that there may be a practical algorithm with an $O(\log n)$ time bound and an $O(n)$ space bound.

8. Other Applications.

As illustrated in this paper, Theorem 1 and its corollaries have many interesting applications, and the paper does not exhaust them. We have obtained two additional results which require fuller discussion than is possible here. One is the application of Theorem 1 to Gaussian elimination. George [8] has proposed an $O(n \log n)$ -space, $O(n^{3/2})$ -time method of carrying out Gaussian elimination on a system of equations whose sparsity structure corresponds to a $\sqrt{n} \times \sqrt{n}$ square grid. We can generalize his method so that it applies to any system of equations whose sparsity structure corresponds to a planar or almost-planar graph. Such systems arise in the solution of two-dimensional finite-element problems [15]. We shall discuss this application in a subsequent paper; we hope that it will prove of practical, as well as theoretical, value.

Another application involves the power of non-determinism in one-tape Turing machines. We can prove that any non-deterministic $t(n)$ -time-bounded one-tape Turing machine can be simulated by a $t(n)^\gamma$ alternating one-tape Turing machine with a constant number of alternations, where $y < 1$ is a suitable constant and $t(n)$ satisfies certain reasonable restrictions. Alternation generalizes the concept of non-determinism and is discussed in [3,12]. Our result strengthens Paterson's space-efficient simulation of one-tape Turing machines [17].

References

[1] A, V. Aho, J. E. Hopcroft, and J. D. Ullman, The Design and Analysis of Efficient Computer Algorithms, Addison-Wesley, Reading, Mass., 1974.

[2] U. Bertele and F. Brioschi, Nonserial Dynamic Programming, Academic Press, New York, 1972,

[3] A. K. Chandra and L. J.Stockmeyer, "Alternation," Proc. Seventeenth Annual Symp. on Foundations of Computer Science (1976),98-108.

[4] S. A. Cook, "An observation on time-storage tradeoff," Proc. Fifth Annual ACM Symp. on Theory of Computing (1973), 29-33.

[5] R. A. DeMillo, S. C. Eisenstat, and R. J. Lipton, *'Preserving average proximity in arrays," School of Information and Computer Science, Georgia Institute of Technology (1976).

[6] D. Dobkin and R. J. Lipton, "Multidimensional searching problems," SIAM J. Comput. 5 (1976), 181-186.

[7] P. Erdös, R. L. Graham, and E. Szemerédi, "On sparse graphs with dense long paths," STAN-CS-75-504, Computer Science Dept., Stanford University (1975).

[8] J. A. George, "Nested dissection of a regular finite element mesh," SIAM J. Numer. Anal. 10 (1973), 345-363.

[9] L. Goldschlager, "The monotone and planar circuit value problems are log space complete for P," ACM SIGACT News 9, 2 (1977), 25-29.

[10] J. Hopcroft, W. Paul, and L. Valiant, "On time versus space," Journal ACM 24 (1977), 332-337.

[11] D. E. Knuth, The Art of Computer Programming, Volume 3: Sorting and Searching, Addison-Wesley, Reading, Mass., 1973.

[12] D. Kozen, "On parallelism in Turing machines," Proc. Seventeenth Annual Symp. on Foundations of Computer Science (1976), 89-97.

[13] R. J. Lipton, S. C. Eisenstat, and R. A. DeMillo, "Space and time hierarchies for control structures and data structures," Journal ACM 23 (1976), 720-732.

[14] R. J. Lipton and R. E. Tarjan, "A separator theorem for planar graphs," to appear.

[15] H. C. Martin and G. F. Carey, Introduction to Finite Element Analysis, McGraw-Hill, New York, 1973.

[16] M. S. Paterson and C. E. Hewitt, "Comparative schematology," Record of Project MAC Conf. on Concurrent Systems and Parallel Computation (1970), 119-128.

[17] M. S. Paterson, "Tape bounds for time-bounded Turing machines," Journal Computer and System Sciences 6 (1972), 116-124.

[18] W. J. Paul, R. E. Tarjan, and J. R. Celoni, "Space bounds for a game on graphs," Math. Systems Theory 10 (1977), 239-251.

[19] A. L. Rosenberg, "Managing storage for extendible arrays," SIAM J. Comput. 4 (1975), 287-306.

[20] A. Rosenthal, "Nonserial dynamic programming is optimal," Proc. Ninth Annual ACM Symp. on Theory of Computing (1977), 98-105.

[21] R. Sethi, "Complete register allocation problems," SIAM J. Comput. 4 (1975), 226-248,

[22] M. J. Shamos, "Geometric complexity," Proc. Seventh Annual ACM Symp. on Theory of Computing (1975), 224-233.

[23] L. G. Valiant, "On non-linear lower bounds in computational complexity," Proc. Seventh Annual ACM Symp. on Theory of Computing (1975), 45-53 .

[24] L. G. Valiant, "Graph-theoretic arguments in low-level complexity," Computer Science Dept., University of Edinburgh (1977).

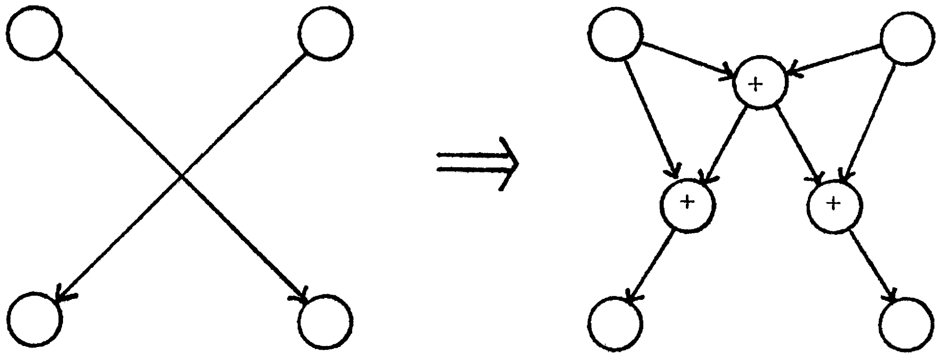

Figure 1. Elimination of a crossover by use of three "exclusive or" gates. Reference [9] contains a crossover circuit which uses only "and" and "not" .

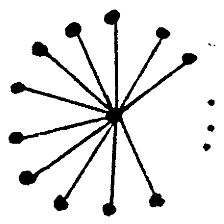

Figure 2. A star.