PARALLEL ALGORITHMS FOR MINIMUM CUTS AND MAXIMUM FLOWS IN PLANAR NETWORKS

by

D. B. Johnson and S. M. Venkatesar

Memorandum No. UCB/ERL M82/12

25 February 1982

PARALLEL ALGORITHMS FOR MINIMUM CUTS AND MAXIMUM FLOWS IN PLANAR NETWORKS

by

Donald B. Johnson and Shankar M. Venkatesan

Memorandum No. UCB/ERL M82/12

25 February 1982

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94820

PARALLEL ALGORITHMS FOR MINIMUM CUTS AND MAXIMUM FLOWS IN PLANAR NETWORKS†

Donald B. Johnson

University of California, Berkeley††
The Pennsylvania State University†††

Shankar M. Venkatesan

The Pennsylvania State University†††

February, 1982

††Computer Science Division
573 Evans Hall
University of California
Berkeley, California 94720

†††Department of Computer Science
Whitmore Laboratory
The Pennsyalvania State University
University Park, PA 16802

† This work partially supported by the National Science Foundation under grants MCS-8002684 and MCS-8105217.

PARALLEL ALGORITHMS FOR MINIMUM CUTS AND MAXIMUM FLOWS IN PLANAR NETWORKS[1]

Donald B. Johnson

University of California, Berkeley[2]
The Pennsylvania State University[3]

Shankar M. Venkatesan

The Pennsylvania State University[3]

1. Introduction

We give algorithms which find a minimum capacity s-t cut and therefore the value of a maximum s-t flow in a planar network with n nodes in $O((\log n)^2)$ time on a parallel computer with a polynomial number of processors which share a common memory. One algorithm solves the problem on undirected networks; the other solves it on directed networks, using polynomially more processors. An alternative algorithm solves the problem on directed networks using no more processors asymptotically than in the undirected case but at the expense of an additional log factor in time. We also show how to find a maximum flow itself in $O(n(\log n)^2)$ parallel time for each case, undirected and directed planar networks, and again in the directed case exhibit a tradeoff of a log factor in time against a polynomial factor in number of processors. When networks are $(s-t)$-planar, a running time of $O((\log n)^2)$ can be achieved.

These results are relevant by themselves and in the context of parallel algorithms for general combinatorial optimization problems. They are more interesting, however, in view of the result by Goldschlager *et al* [7]

[1]This research partially supported by the National Science Foundation under grants MCS-8002684 and MCS-8105217.

[2]Computer Science Division, 573 Evans Hall, Berkeley, CA 94720

[3]Computer Science Department, Whitmore Laboratory, University Park, PA 16802

where it is shown that finding the value of a maximum flow in a general network is log space complete for P. It is therefore unlikely, in the light of [6], that an algorithm for general networks can be found that runs in $O((\log n)^k)$ parallel time for any given constant k.

Our results employ the model of unbounded parallelism where it is assumed, among other things, that

(1) an unbounded number of processors is available,

(2) memory is common to all processors, and

(3) a memory location can be read from, but not written into, simultaneously by more than one processor.

The best serial algorithms known for planar networks (*i*) finds a minimum (s,t)-cut in a directed (s,t)-planar network in $O(n\log n)$ time [4,5], (*ii*) finds a maximum flow in a directed (s,t)-planar network in $O(n\log n)$ time [9], and (*ii*) finds a minimum (s,t)-cut in an undirected planar network in $O(n(\log n)^2)$ time [17]. There is a parallel algorithm by Shiloach and Vishkin [19] which computes maximum flows in general networks in $O(n^3(\log n)/p)$ time with $p \le n$ processors. When capacities are polynomial in n, the flow problem in a general network can be reduced to the solution of $O(\log n)$ perfect matching problems, using a construction to be found in [13]. The results in [2,15,18] imply a nonuniform algorithm which solves the perfect matching problem in $O((\log n)^2)$ parallel time.

2. Preliminaries

Let $G=(V,E)$ be either an undirected or a directed graph. The network $N=(G,c)$ has a *capacity function* $c : E \to \mathbf{R}^+$. The *capacity* of an edge set $X \subseteq E$ is the sum of the capacities of its edges. The distinguished vertices s and t are the *source* and *sink*, respectively. A set $X \subseteq E$ is a *disconnecting set* if the undirected version of $(V,E-X)$ has no path from

s to t. The set $X \subseteq E$ is an (s,t)-*cut* if no proper subset of X is disconnecting. Let (V_s,E_s) and (V_t,E_t) be the weakly connected subgraphs of $(V,E-X)$ for some (s,t)-cut X. Then X is a minimum (s,t)-cut if $c\{e \mid e=(v,w)\in X \text{ and } v\in V_s, w\in V_t\}$ is minimized over all (s,t)-cuts where (v,w) is taken either as directed or undirected as the case may be. A *flow* f in a directed network is a function $f : E\rightarrow \mathbf{R}$ which satisfies conservation at each vertex except s and t and for which $0\leq f(e)\leq c(e)$ for each $e\in E$. The *value* $v(f)$ of a flow f is the sum of $f(e)$ for edges e incident from s. To consider flows in undirected networks, replace undirected edge (v,w) by a pair of directed edges $<v,w>$ and $<w,v>$, and call the resulting directed graph $G'=(V,A)$. If f is a flow function for G', the flow in the undirected edge (v,w) is obtained as $|f(<v,w>)-f(<w,v>)|$, with proper orientation. For additional definitions and terminology in graph and network flow theory, see [1,13].

> PROPOSITION 1 [4]. The value of a maximum flow is equal to the capacity of a minimum (s,t)-cut in any network. ▪

For a *plane network* Π, that is, a plane embedding of a planar network N, let $D(\Pi)$ be the dual plane network of Π and $D(N)$ be a network consistent with the plane network $D(\Pi)$. Let the dual edge $D(e)$ have the same cost as e for each edge $D(e)\in D(N)$. To capture (s,t)-cuts in terms of cycles (which throughout this paper mean vertex-simple closed paths) in a directed planar network, let $\hat{D}(N)$ be $D(N)$ augmented with edge $\hat{e}=(w,v)$ of cost $c(\hat{e})=0$ for each edge $e=(v,w)$ in $D(N)$. For undirected plane networks, simply define $\hat{D}(N)=D(N)$. Clearly $\hat{D}(N)$ corresponds to a plane network $\hat{D}(\Pi)$ in each case. A cycle q in $\hat{D}(\Pi)$ is a *cut-cycle* when the region of the plane bounded by q includes exactly one of s or t. In the case of directed networks, q is a *forward* cut-cycle if q

is directed clockwise when q encloses s and is directed anticlockwise when q encloses t. Cut-cycles with opposite orientation are called *reverse* cut-cycles. Using the terminology of Reif [17], we have the following additional definitions. A cut-cycle or forward cut-cycle q in $\hat{D}(\Pi)$ is F-minimum if q is incident on dual vertex F (*i.e.* face F of Π) and has the smallest cost $c(q)$ of all cut cycles incident on F. While cut-cycles are defined in terms of plane networks, we apply the terminology to the corresponding cycles in the underlying planar networks.

PROPOSITION 2 [11]. The cost of a minimum-cost (s,t)-cut in N is equal to the cost of a minimum (forward) cut-cycle in $\hat{D}(N)$. ▪

Continuing the definitions, a *$\mu(s,t)$-path*, *μ-path* or μ for short, is a path of minimum cost in $\hat{D}(N)$ from any vertex of $\hat{D}(N)$ adjoining s to any vertex of $\hat{D}(N)$ adjoining t. A vertex of $\hat{D}(N)$ is said to *adjoin* a vertex v of N if v is on the perimeter of the face in Π corresponding to F in $\hat{D}(\Pi)$. Let μ traverse the vertices $F_1, F_2, \cdots, F_d$ in $\hat{D}(N)$, and let q_i be an F_i-minimum cut-cycle (forward cut-cycle in the directed case) of $\hat{D}(N)$ for $i=1, \cdots, d$. Let i_0 satisfy $c(q_{i_0}) \leq c(q_i)$ for $i=1, \cdots, d$.

PROPOSITION 3 [11]. $\hat{D}^{-1}(q_{i_0})$ is a minimum (s,t)-cut of N. ▪

For a given plane embedding Π, if μ is traversed in the order $(F_1, \cdots, F_d)$, then every edge incident on μ enters μ from the left or from the right. In our case, since the undirected version of N is triconnected, no self-loops exist in $\hat{D}(\Pi)$. If indeed N is not triconnected, it can be made so as follows. Find the triconnected components, which can be done in $O((\log n)^2)$ parallel time using $O(n^4)$ processors [12], and then add dummy edges of zero capacity to triconnect the original network.

In the undirected case, the following construction is known. Given $D(N)$, a planar dual of N, let μ' be a copy of μ and let F_i', lying on μ', be the corresponding copies of F_i, $i=1,\cdots,d$. Reconnect to F_i' all edges incident on F_i from the left on a traversal of μ from F_1 to F_d. Call the resulting undirected network D'. A path p' in D' corresponds to a path p in $\hat{D}(N)$ by replacing every vertex of μ' with the corresponding vertex on μ.

> PROPOSITION 4 [11]. Let p' be a minimum cost simple path between F_i and F_i' in undirected D'. Then p is an F_i-minimum cut-cycle of $\hat{D}(N)$.

Proof: Every subpath of μ is of minimum cost since μ is minimum. So if a subpath of an F_i-minimum cut-cycle contains two or more vertex-disjoint subpaths from μ, the lowest and highest numbered vertices from all of these subpaths being F_x and F_y, respectively, then the subpath from F_x to F_y in the F_i-minimum cut-cycle can be replaced by the subpath of μ from F_x to F_y. Hence a simple path p' in D' from F_i to F_i' is minimum if and only if p is a minimum cost F_i-cut-cycle in $D(N)$. ▪

3. Computing a Minimum Capacity Cut in an Undirected Network

The strategy of the algorithm should be clear from the results just given, namely, to search the space of F_i-minimal cut cycles for a globally minimal cut cycle. To realize this strategy efficiently in parallel computation we must obtain efficiently a representation for a plane dual $\hat{D}(N)$. Then we must find a μ-path, split it to yield D' defined above, and search the path space induced by this construction for a shortest path.

The first step in obtaining a plane dual of $N=(G,c)$ is to obtain a plane mesh for G. A *cycle basis* for G is a set of cycles with the property

that any cycle of G can be expressed as the the symmetric difference of cycles in the cycle basis. A cycle C is *peripheral* if the number of bridges in G relative to C is 1. Let $\mathbf{C}$ be a peripheral cycle basis of G. Then $M = \mathbf{C} \cup \{$ symmetric difference of all $C \in \mathbf{C}\}$ is called a *plane mesh* of G if every edge of G appears in exactly two cycles of M. It has been shown by McLane [16] that G is planar if and only if G has a plane mesh.

Ja'Ja' and Simon [12] define a combinatorial object which they call a *pseudo-embedding* of G. They show a construction which yields a plane mesh for G, given its pseudo-embedding, whenever G is planar. The construction must, by McLane's result, fail when G is nonplanar. Their construction gives us the following lemma.

> LEMMA 1 [12]. A plane mesh can be constructed for any planar graph in $O((\log n)^2)$ parallel time using $O(n^4)$ processors. ▪

The set of cycles of the plane mesh is output as a set of 0–1 vectors over the edge set of G; the presence or absence of an edge in a cycle is denoted by a 1 or a 0 in the corresponding vector. Because of this representation, the following result is immediate.

> LEMMA 2. A planar dual $D(N)$ can be constructed from a plane mesh for N in $O(\log n)$ parallel time using $O(n^4)$ processors. ▪

A construction given later describes an $O((\log n)^2)$ parallel time shortest path algorithm. We use this algorithm to identify a μ-path. To construct D', the left or right incidence of dual edge edges to the μ-path in an embedding of $D(N)$ must be determined. To do this efficiently a clockwise ordering of edges around every face f of Π is obtained independently for each face. Note that this ordering corresponds exactly

to the clockwise ordering of the dual edges around the vertex of $\hat{D}(\Pi)$ that stands for f.

LEMMA 3. Clockwise ordering of primal edges around each face of Π can be obtained in $O(\log n)$ parallel time using $O(n)$ processors.

Proof: Split each vertex v on the face into two nodes $[v,1]$ and $[v,2]$ and arbitrarily assign these two nodes to the two edges of the face incident on v. Let $\mathrm{CL}[v,x]=[w,y]$ if $[w,y]$ follows $[v,x]$ in a clockwise ordering of the edges, where $x,y\in\{1,2\}$. Similarly, let $\mathrm{ACL}[w,y]=[v,x]$. See Figure 1. The algorithm that computes CL and ACL is in essence similar to the one in [14] for canonical edge coloring of 2-way graphs, CL and ACL playing almost identical roles to the colors in [14]. The details of the algorithm will be omitted. See also [10]. ▪

Let edge (x,y) be common to faces F_{i_1} and F_{i_2} in Π. Then going from x to y must be clockwise on one of F_{i_1} and F_{i_2} and anticlockwise in the other; if not, the ordering of either F_{i_1} or F_{i_2} must be reversed to be consistent. For the ordering of the entire graph to be consistent, consistency must exist for every pair of adjacent faces. However, it may be seen that, for our purposes, consistency needs to exist only for the faces on the μ-path.

LEMMA 4. A consistent clockwise ordering of the enclosing edges in Π for each face f_i corresponding to a vertex F_i on the μ-path in $\hat{D}(\Pi)$ can be obtained in $O((\log n)^2)$ parallel time using $O(n\log n)$ processors.

Proof: At step i of each of $\lceil \log d \rceil$ steps, consistent sets of 2^i faces are obtained from pairs of consistent sets of size 2^{i-1}, where this operation is

performed in parallel on each interval of size 2^i in a partition of the μ-path. If the two subsets of size 2^{i-1} in some interval of size 2^i are already consistent, nothing is done. Otherwise, the orientations of all the faces in the second subset of the pair are reversed. Although propagating reversals when necessary to each of the 2^{i-1} faces involved in one subset of a pair takes $i-1$ steps, these steps can occur in parallel with further adjustments, the adjustments from successive steps being pipelined through the levels of a binary tree constructed with the faces of μ as leaves. We omit further details except to note that $d=O(n)$. ▪

When consistency along the μ-path is obtained, it is known exactly which one of the following is true for each face f_i: either CL represents correctly a clockwise ordering or CL must be reversed to be consistent.

Finally, the following lemma is needed.

> LEMMA 5. The dual edges of $\hat{D}(\Pi)$ which enter face F_i from the left can be computed from a consistent clockwise ordering for f_i in $O(\log n)$ parallel time using $O(n^2)$ processors.

Proof: Let the dual vertex F_i be defined by the primal cycle $C_i=(y,x,\cdots,x',y',\cdots y)$, where we denote C_i in clockwise order, and let the dual edges (F_{i-1},F_i) and (F_i,F_{i+1}) be defined by the primal edges (x,y) and (x',y'), respectively. Without loss of generality let ACL$[x]=y$ and CL$[x']=y'$ in the consistent ordering (We condense our previous notation.). See Figure 2. The primal edges on the path $(x,\cdots,x')$ (not going through either y or y') define dual edges entering F_i from the left. Disconnect the cyclic ordering at y and y', set the index of x or x' to be the lowest around F_i, and run a component-finding algorithm on C_i. This step takes $O(\log n)$ parallel time. If an edge belongs to the lowest num-

bered component in F_i, then its dual enters F_i from the left. This at once defines edges incident on the path μ'. (For F_1 and F_d, s and t act as F_{i-1} and F_{i+1}, respectively.) ▪

> THEOREM 1. A minimum capacity cut and the maximum flow value in an undirected planar network can be computed in $O((\log n)^2)$ parallel time using $O(n^4)$ processors.

Proof: Execute the following algorithm. Given a network, find a plane mesh for it (if none exists, the network is non-planar); this can be done in $O((\log n)^2)$ time with $O(n^4)$ processors [12]. Find the planar dual; this takes $O(\log n)$ time and at most $O(n^3)$ processors, since there are at most $O(n^2)$ pairs of faces, each made up of $O(n)$ edges. Compute a μ-path; this is done in $O((\log n)^2)$ time with $O(n^3)$ processors as described in Section 5. Compute clockwise orderings for all faces in $O(\log n)$ time with $O(n^2)$ processors. Obtain consistency in the clockwise orderings on the μ-path and, from this, the left-incidence of the dual edges to the μ-path; this needs at most $O((\log n)^2)$ time and $O(n^2)$ processors. Construct network D' and compute a minimum cut-cycle using the shortest path algorithm of Section 5 in $O((\log n)^2)$ time with $O(n^3)$ processors. The cost of the minimum cut-cycle yields the maximum flow value. The proofs of correctness of the individual steps can be found in the previous propositions and lemmas. The algorithm takes $O((\log n)^2)$ time and $O(n^4)$ processors. ▪

4. Computing a Minimum Capacity Cut in a Directed Network

Proposition 3 allows us to use the same strategy as in the undirected case, namely to search the space of F_i-minimal forward cut-cycles for a globally minimum cycle. However, the analogue of Proposition 4 does not hold if D' is directed. A more difficult construction is required to force a

path-finding algorithm to find forward cut-cycles.

Let q be a path of the plane network $\hat{D}(\Pi)$ which begins at some μ-vertex F_l and ends at a μ-vertex. Any such q has a *canonical representation* $q=(p_1,\cdots,p_k)$, where $p_1=(F_l,\cdots)$, in which each p_i begins and ends on μ and is extremal in a sense to be described shortly. When q is a closed path, we require that F_l of the denotation be chosen so that q intersects μ at no vertex F_i, $i<l$. In this case, vertex F_l is called the *earliest intersection* with μ. If earliest intersection F_l appears exactly once in closed path q, then q is *admissible*.

When a subpath p_i of the canonical representation with no internal vertices on μ leaves μ from one side, say the left, and returns to μ on the other, say the right, then p_i is *enclosing*. An enclosing subpath is *left enclosing* or *right enclosing* if it leaves μ on the left or on the right, respectively. If a subpath p_i leaves and reenters on the same side and contains no enclosing subpath, it is *nonenclosing*. A subpath is *coincident* if it is nonenclosing and has all its internal vertices on μ. Any subpath that is not coincident (and therefore has at least one edge) is *noncoincident*. Each coincident subpath in the canonical representation $q=(p_1,\cdots,p_k)$ is required to be maximal while each noncoincident subpath in the canonical representation must be minimal. These conditions are sufficient to uniquely define the canonical representation. Since we admit the trivial single-vertex coincident subpath, coincident and noncoincident subpaths alternate on the canonical representation, and in particular p_1 is coincident.

Let the number of right enclosing subpaths minus the number of left enclosing subpaths in the canonical representation of a path q be the *excess* of q, which can be written *excess*(q). We observe that the closed

curve in the plane, constructed by connecting the undirected version of any noncoincident p_i in a canonical representation with the undirected version of the segment of μ joining the endpoints of p_i, separates s and t if and only if p_i is enclosing, as just defined. We extend this observation to coincident subpaths (which may not be simple and therefore may induce many regions in the plane) by noticing that no region so induced can enclose exactly one of s and t.

A closed path $q = (v_1, \cdots, v_m)$ in a plane digraph is *pseudosimple* if every intersection of q with itself is noncrossing in the following sense. For every pair of maximal subpaths $(v_{i_0}, \cdots, v_{i_0+h})$ and $(v_{j_0}, \cdots, v_{j_0+h})$ for which either $v_{i_0+l} = v_{j_0+l}$, $l = 0, \cdots, h$, or $v_{i_0+l} = v_{j_0+h-l}$, $l = 0, \cdots, h$, edge (v_{j_0-1}, v_{j_0}) enters on the same side of $(v_{i_0-1}, v_{i_0}, \cdots, v_{i_0+h}, v_{i_0+h+1})$ as the side from which edge (v_{j_0+h}, v_{j_0+h+1}) leaves $(v_{i_0-1}, v_{i_0}, \cdots, v_{i_0+h}, v_{i_0+h+1})$. It is clear that a pseudosimple closed path is topologically equivalent to a Jordan curve in the plane. Because of this equivalence, we may extend the definitions of *forward cut*, *backward cut*, and *noncut* directly to pseudosimple closed paths. We call a pseudosimple closed path *flat* if each of its canonical subpaths is either coincident or enclosing.

> LEMMA 6. If q is a pseudosimple closed path in plane graph $\hat{D}(\Pi)$, then $excess(q) \in \{-1, 0, +1\}$. Furthermore, q is a forward cut exactly when $excess(q) = +1$, q is a backward cut exactly when $excess(q) = -1$, and q is not a cut exactly when $excess(q) = 0$.

Proof: Let q be pseudosimple and have canonical representation $(p_1, \cdots, p_k)$. We may without loss of generality assume that q is flat, for if it is not there will be some noncoincident, nonenclosing subpath of q

for which the Jordan curve it induces with μ properly encloses no vertex of q. This subpath may therefore be replaced by a subpath which follows μ, either forward or backward, without changing the excess or the cut properties of q. Furthermore, the edges of the subpath left unused by this replacement can be deleted from $\hat{D}(\Pi)$. A sequence of such transformations yields a digraph, and a flat pseudosimple closed path within it, which have the same properties relevant to the lemma as do the given q and $\hat{D}(\Pi)$.

When there is exactly one enclosing subpath it is clear that the lemma holds. Thus the lemma holds for $k \leq 2$. Assume that the lemma holds for all digraphs and for all pseudosimple closed paths q for which $k \leq k_0$, $k_0 \geq 2$, in the canonical representation. Let q' be given for which $k' = k_0 + 1$, and let q' have more than one enclosing subpath.

If we assume that the (multiple) enclosing subpaths of q' are either all right enclosing or all left enclosing we obtain a contradiction as follows. Consider the case where all are assumed to be right enclosing and, in particular, p_2 is right enclosing. We picture the main cases in Figure 3. It may be seen that for q' to be a pseudosimple closed path there must appear in q' at some time a left enclosing subpath from the segment of μ labeled μ_a to the segment labeled μ_b, and this subpath cannot cross either p_2 or p_4. The left enclosing case yields a similar contradiction.

Thus we may assume that there are two enclosing subpaths p_l and p_m, $m = l+2$, where if p_l is right enclosing then p_m is left enclosing, and *vice versa*, and if $p_l = (F_{i_l}, \cdots, F_{j_l})$ and $p_m = (F_{i_m}, \cdots, F_{j_m})$ then no portion of the Jordan curve for q is contained within the Jordan curve defined by $(p_l, p_{l+1}, p_m, F_{j_m}, \cdots, F_{i_l})$ where the last segment $(F_{j_m}, \cdots, F_{i_l})$ is coincident. It is clear that the excess and the cut properties of q'

remain unaltered if (p_l, p_{l+1}, p_m) is replaced by the coincident segment $(F_{i_1}, \cdots, F_{j_m})$, adding edges as required. But this derived pseudosimple closed path has fewer than k_0+1 subpaths, and therefore the lemma holds for it. It follows that the lemma holds for the given graph and path, and therefore by induction the lemma holds for any given graph and admissible pseudosimple closed path. ▪

We now derive from $\hat{D}(N)$ a network $\tilde{D}$ in which every path has an image in $\hat{D}(N)$. In particular, the images of a sufficient set of shortest paths will be F_i-minimum forward cut-cycles for vertices F_i on μ. The construction involves many copies of $\hat{D}(N)$. For a given integer i, copies $\hat{D}^{i\mathbf{L}}(N)$ and $\hat{D}^{i\mathbf{R}}(N)$ are formed by deleting all edges entering or leaving μ on the left in $\hat{D}(\Pi)$ to form $\hat{D}^{i\mathbf{L}}(N)$ and deleting all edges entering or leaving μ on the right in the case of $\hat{D}^{i\mathbf{R}}(N)$. The superscript "**L**" stands for "Left incidence diverted." The superscript "**R**" has an analogous meaning. See Figure 4.

A *unit* U_i of $\tilde{D}$ is then constructed from $\hat{D}^{i\mathbf{L}}(N)$ and $\hat{D}^{i\mathbf{R}}(N)$, and additional vertices $\{F_1^{(i-1)}, \cdots, F_d^{(i-1)}, F_1^{(i+1)}, \cdots, F_d^{(i+1)}\}$ as follows. For every edge (u, F_j) in $\hat{D}(\Pi)$, with F_j on μ, construct the following edges:

if (u, F_j) enters μ on the left construct $(u^{i\mathbf{L}}, F_j^{(i+1)})$, and

if (u, F_j) enters μ on the right construct $(u^{i\mathbf{R}}, F_j^{(i-1)})$.

Then, for every vertex F_j on μ in $\hat{D}(\Pi)$, identify vertices $F_j^{i\mathbf{L}}$ and $F_j^{i\mathbf{R}}$, giving the resulting vertex the name F_j^i. As may be seen, the unit U_i contains vertices $\{F_j^{i-1}, F_j^i, F_j^{i+1} | j=1, \cdots, d\}$ as well as all vertices not on the μ-path from both copies $\hat{D}^{i\mathbf{L}}(N)$ and $\hat{D}^{i\mathbf{R}}(N)$. In general, there are edges from both of the copies, as well as the new edges.

The network $\tilde{D}$ is formed as the union of U_i for all i, the union being

obtained by identifying vertices with identical names. (These vertices are exactly the vertices F_j^i which appear in U_{i-1}, U_i, and U_{i+1} for all i and j.) The network $\tilde{D}$ is therefore an infinite network with the property that the superscript i will be equal to the excess of any path in $\hat{D}(\Pi)$ which is the image of a path $(F_j^0, \cdots, F_j^i)$ in $\tilde{D}$. This completes the construction of $\tilde{D}$.

Let $\Psi_l(n)$ be any set of simple paths $(F_l^0, \cdots, F_l^1)$ in $\tilde{D}$ which contains no path with an internal vertex from the set $\{F_j^i \mid 1 \le j \le l \text{ and all } i\}$ and, subject to this restriction on internal vertices, contains but is not restricted to all simple paths $(F_l^0, \cdots, F_l^1)$ in $\tilde{D}$ with n or fewer vertices. (A set $\Psi_l(n)$ may be described alternatively in terms of $\tilde{D} - \{F_j^i \mid 1 \le j < l \text{ or } (j = l \text{ and } i \ne 0,1)\}$.) If we let $h(p)$ be the extension to paths of the homomorphism $h(u^x) = u$ for any superscript "x", then $h(\Psi_l(n))$ is a set of admissible paths in $\hat{D}(\Pi)$ with earliest intersection F_l. Our algorithm will require paths p, from some set $\Psi_l(n)$, for which $h(p)$ is simple.

> LEMMA 7. Let p be a path in any $\Psi_l(n)$ which is shortest subject to the constraint that $h(p)$ is simple. Then $h(p)$ is an F_l-minimum cut-cycle in $\hat{D}(\Pi)$.

Proof: For any $\Psi_l(n)$, let p' be any path in $\Psi_l(n)$ for which $h(p')$ is simple. The path p' is of the form $(p_1', \cdots, p_k')$ where $h(p_i')$ is either maximal nonenclosing or enclosing, for $i = 1, \cdots, k$. Let subpath p_i' be *enclosing* exactly when $h(p_i')$ is enclosing. It may be shown by induction on the traversal of p' over the edges introduced in the construction of $\tilde{D}$ that $h(p')$ has excess equal to 1. Thus, by Lemma 6, $h(p')$ is a forward cut-cycle.

Since every edge in $\hat{D}(\Pi)$ is the image of one or more edges in $\tilde{D}$, it may be shown by induction on the subpaths p_i that any forward cut-cycle

$q = (F_l, \cdots, F_l)$ in $\hat{D}(\Pi)$ is the image of some path p which is in every $\Psi_l(n)$. It follows that if p is of minimum length in any $\Psi_l(n)$, subject to the restriction that $h(p)$ is simple, then $h(p)$ is of minimum length among all forward cut-cycles with earliest intersection F_l. This in turn implies F_l-minimality. ▪

The preceding lemma does not as yet yield our result because, while it is relatively easy to find shortest paths over some $\Psi_l(n)$, the requirement that the image $h(p)$ be simple is difficult to enforce. If this condition is not enforced, then a shortest path we find at F_l may be of lower cost than an F_l-minimum cut-cycle. We attack this problem globally over all $l = 1, \cdots, d$ by means of the next lemma.

For any closed path q denoted as $(v_1, v_2, \cdots, v_m)$, $v_m = v_1$, we define an *initial cycle* q_I to be any closed path $(v_1, \cdots, v_m)$ which is simple and is a subgraph of q. For example, one initial cycle is the path generated by a traversal of q from vertex v_1 where, at each i-th path vertex v_i at which q intersects itself, the edge (v_j, v_{j+1}) is taken, where j is the largest index for which $v_i = v_j$ in the vertex set of the graph. It is clear that the closed path $q_I = (v_1, \cdots, v_m)$, $v_m = v_1$, so generated is simple. Thus at least one initial cycle q_I exists for any q. We note that the definition of initial cycle depends on the initial vertex in the denotation of the path and, consequently, different initial cycles of the same path may have different start vertices. However, the initial cycles of an admissible path denoted as such all have the same start vertex.

> LEMMA 8. Let p be any closed path in $\hat{D}(\Pi)$ in which some vertex v_i of the graph appears more than once. For any partition of p into closed paths $p_j = (v_i, \cdots, v_i)$ and $p_l = (v_i, \cdots, v_i)$,

$$excess(p) = excess(p_j) + excess(p_l).$$

Proof: We observe that $p = (p_j, p_l)$ and that p_j and p_l are of the form $p_j = (v_i, \cdots F_{j_1}, \cdots, F_{j_2}, \cdots, v_i)$ and $p_l = (v_i, \cdots F_{l_1}, \cdots, F_{l_2}, \cdots, v_i)$, where F_{j_1} is the first μ-vertex on p_j, F_{j_2} is the last μ-vertex on p_j, and similarly for F_{l_1} and F_{l_2} on p_l. When, for example, the subpath $(F_{j_2}, \cdots, v_i, \cdots, F_{j_1})$ of p_j and the subpath $(F_{l_2}, \cdots, v_i, \cdots, F_{l_1})$ of p_l are both nonenclosing, then if $excess(F_{j_2}, \cdots, v_i, \cdots, F_{l_1}) = 1$ it must be that $excess(F_{l_2}, \cdots, v_i, \cdots, F_{j_1}) = -1$ (where these last two subpaths are on p), and the result follows in this case. The remaining seven cases can be demonstrated similarly. ▪

LEMMA 9. For any $\Psi_l(n)$, let p be a shortest path in $\Psi_l(n)$. If $h(p)$ is not simple, then one of two cases occurs:

(i) All initial cycles q_I of $h(p)$ are forward cut-cycles, or

(ii) There exists an index $h > l$ and a path p' which belongs to every $\Psi_h(n)$ and which is no longer than p.

Proof: When some initial cycle q_I of $h(p)$ is not a forward cut-cycle, then by Lemma 6 it has excess equal to 0 or -1. It follows by induction on the result in Lemma 8 that the sum of the excesses of the paths in $h(p) - q_I$, which is a collection of directed closed paths, is greater than or equal to 1. Therefore at least one closed path in the collection has a strictly positive excess. Repeated application of this construction to such a closed path, choosing any initial cycle at each step, will eventually yield a closed path q_I' with excess equal to 1. Since all edge lengths are nonnegative and a simple path can have no more than n vertices, it follows that $h(q_I')$ satisfies the conditions set forth in (ii). ▪

LEMMA 10. Let p_l be shortest in some $\Psi_l(n)$ for each $l=1,\cdots,d$, let Q be any set containing one arbitrary initial cycle from each path in the set $\{p_l \mid l=1,\cdots,d\}$, and let $Q_C \subseteq Q$ be the forward cut-cycles in Q. Any cycle of minimum length in Q_C corresponds to a minimum cut in N.

Proof: Since all edge weights are nonnegative, subpaths are never longer than the paths in which they are contained. Thus Lemmas 7 and 9 guarantee that the minimum cycle is a minimum forward cut-cycle for $\hat{D}(\Pi)$, which in turn by Propositions 2 and 3 gives our result. ▪

Lemma 10 establishes the correctness of the following algorithm for our problem.

ALGORITHM DIRECTED-MIN-CUT

1. Construct a unit U_i of $\tilde{D}$.
2. For each $l = 1,\cdots,d$, find a shortest path in some $\Psi_l(n)$.
3. In each path obtained in Step 2, find an initial cycle. Find the shortest cut-cycle among the subset Q_C of these initial cycles which happen to be forward cut-cycles.

THEOREM 2. Algorithm DIRECTED-MIN-CUT can be implemented to run in $O((\log n)^2)$ parallel time using $O(n^6)$ processors, and in $O((\log n)^3)$ time using $O(n^4)$ processors.

Proof: We may use the results of the previous section to construct U_i in $O((\log n)^2)$ parallel time using $O(n^4)$ processors. Given the results of Step 2, it is possible to implement Step 3 in $O((\log n)^2)$ time and $O(n^4)$ processors. To do this, in each of d paths it is necessary to find an initial cycle and test if it is a forward cut-cycle.

Let $p=(v_1,v_2,\cdots,v_m)$ be a closed path where $v_1=v_m\neq v_i$ for $i=2,\cdots,m-1$. To find an initial cycle in p, we proceed as follows. Let the *last mate* of a vertex v_i in p be the vertex defined as $lastmate(v_1)=v_2$, and for $i=2,\cdots,m-1$ $lastmate(v_i)=v_j$ where $j=\max\{i+1,l\,|\,v_i=v_l\}$. Given p, last mates for all vertices v_i, $i<m$, in p can be found in $O(\log n)$ parallel time and $O(mn)$ processors, when there are no more than n distinct vertices in p (which is the case in our application). The graph induced on $\{v_1,\cdots,v_m\}$ by directed edges $(v_i,lastmate(v_i))$ has exactly one path p_I of the form $(v_1,\cdots,v_m)$, and this path p_I is an initial cycle of p. This initial cycle p_I may be found in an additional $O(\log n)$ parallel time using $O(m)$ processors.

To find one initial cycle for each of d paths p_l in $O(\log n)$ parallel time using $O(n^4)$ processors we must therefore guarantee that $m=O(n^2)$. As will be seen, we may only be able to guarantee in our application that $m=O(n^3)$, but when $m=\omega(n^2)$ we will be able to apply the *lastmate* reduction twice, first to paths with $O(n^2)$ vertices, reducing the number of vertices on these paths to $O(n)$, then expanding these paths again to $O(n^2)$ vertices, and applying the *lastmate* reduction once more. Any initial cycle of any closed subgraph $(v_1,\cdots,v_m)$ of some closed path $p=(v_1,\cdots,v_m)$ is also an initial cycle of p.

We now show how to implement Step 2. For a given l, let us take $\tilde{D}_l$ to be $\tilde{D}-\{F_j^i\,|\,j=1,\cdots,l \text{ and all } i\}$. Let Δ_l be the matrix of shortest distances in $\tilde{D}_l$ between vertices in the set $\{F_j^{-1},F_j^0,F_j^1,F_j^2\,|\,j=l+1,\cdots,d\}$. By the shortest path results in Section 5 we can in $O((\log n)^2)$ time and $O(n^3)$ processors construct distance matrices A_l (of dimensions $1\times 4(d-l)$) and B_l (of dimensions $4(d-l)\times 1$) and evaluate the "plus-min" matrix product $A_l\Delta_l B_l$ which yields (by the bookkeeping of Section 5) a

shortest path $(F_l^0, \cdots, F_l^1)$ with no internal vertices from the set $\{F_j^i \mid 1 \le j \le l \text{ and all } i\}$. We now show how to obtain Δ_l from a single unit U_i of $\tilde{D}_l$.

Let M_l be a matrix of numbers on $S \times S$, where $S = \{F_i^j \mid i = l+1, \cdots, d, \text{ and all } j\}$, a subset of the vertices of $\tilde{D}_l$. Let S be ordered so that F_i^j precedes F_k^j when $i<k$, and F_i^j precedes F_k^m when $j<m$. With this ordering, square submatrices M_l^{ij} of M_l are induced by $\{F_{l+1}^i, \cdots, F_d^i\} \times \{F_{l+1}^j, \cdots, F_d^j\}$. These $(d-l)\times(d-l)$ submatrices are *blocks* of M_l, and we may represent M_l as a *block matrix*, the elements of which are not numbers but $(d-l)\times(d-l)$ matrices of numbers. A block M_l^{ij} is on the *main block diagonal* when $i=j$, and is on the same *minor* block diagonal whenever $i-j=c$, for a fixed constant $c \neq 0$. We determine the elements of M_l by assigning to blocks $M_l^{i,(i-1)}$, M_l^{ii}, and $M_l^{i,(i+1)}$ the shortest distances in unit U_i of $\tilde{D}_l$ (which are over paths with at most n vertices). Any M_l^{ij} not so assigned for some i is set to ∞. The resulting matrix M_l does not represent the edges in $\tilde{D}_l$. Rather it is an edge-length matrix of a network on the vertex set S for which intervertex distances on the set S are identical with those in $\tilde{D}_l$. We observe that M_l is block tridiagonal, with identical blocks throughout each of the diagonals. Since $d-l = O(n)$, the results of Section 5 therefore show that we may obtain Δ_l in $O((\log n)^2)$ time using $O(n^5)$ processors or in $O((\log n)^3)$ time using $O(n^3)$ processors. For all $l = 1, \cdots, n$, computations of Δ_l, A_l, B_l, and then of $A_l \Delta_l B_l$, are done in parallel, yielding one shortest path $(F_l^0, \cdots, F_l^1)$ for each l. Each of these paths is on M_l, and by the results in Section 5 contains $O(n^2)$ vertices. As indicated previously, we may therefore apply the homomorphism h to each path and then apply the *lastmate* construction, yielding paths of vertex-length $O(n)$. Then these

paths can be expanded by replacing the edges of M_t with the shortest paths of U_i from which M_t was obtained. This gives paths of with vertex-length $O(n^2)$ from which one more application of the *lastmate* construction yields an initial cycle for the true shortest paths in $\tilde{D}_t$. ▪

5. Shortest Paths

As is indicated in [3], shortest paths between all pairs of vertices in a network, directed or undirected, can be computed in $O((\log n)^2)$ parallel time by straightforward parallelization of the $O(n^3 \log n)$ serial-time shortest path algorithm which is based on repeated *plus-min* multiplication of the edge-length matrix [13]. To do so, initialize the matrix $K^{(1)}$ to zero, and $U^{(1)}$ to the edge cost matrix and compute as follows.

for $k=2$ **to** $p=\lceil \log(n-1) \rceil$ **do**

$U_{ij}^{(2^k)} = \min_m \{ U_{im}^{(2^{k-1})} + U_{mj}^{(2^{k-1})} \}$

if $U_{ij}^{(2^k)} < U_{ij}^{(2^{k-1})}$ **then**

set $K_{ij}^{(2^k)}=m$ for m satisfying $U_{ij}^{(2^k)}= U_{im}^{(2^{k-1})}+ U_{im}^{(2^{k-1})}$

else $K_{ij}^{(2^k)} = K_{ij}^{(2^{k-1})}$

endfor

When there are no cycles of negative length, $U_{ij}^{(2^l)}$ may be interpreted as the length of a shortest path from i to j such that the path contains no more than 2^l edges. Thus the construction below can be used to identify all the edges in a path from i to j.

IDENTIFY $(i,j,K^{(2^p)})$

Let $m = K_{ij}^{(2^p)}$

if $m=0$ **then** edge (i,j) is in the shortest path

else IDENTIFY $(i,m,K^{(2^p)})$ **and IDENTIFY** $(m,j,K^{(2^p)})$

END_IDENTIFY

Shortest paths between all pairs of vertices in a general directed graph with n vertices can be found in $O((\log n)^2)$ parallel time using $O(n^3)$ processors by means of the above procedures.

It is convenient to observe that plus-min multiplication may be extended to block matrices, matrices in which each element is a square matrix, say, $m \times m$. In the extension, *plus* is plus-min multiplication on the blocks, and *min* is element-wise minimization. When two matrices which are arguments for plus-min multiplication can be viewed as block matrices, the product is the same whether it is found by plus-min multiplication on the matrices or by extended plus-min multiplication on the block matrices.

Now, let a shortest path problem be posed as a nonnegative edge-length matrix M of infinite dimensions which is block tridiagonal on $m \times m$ blocks, and all blocks within any single block diagonal are identical. The problem is to find some $cm \times cm$ block, for positive integer c, in the distance matrix which is the k-fold plus-min multiple of M for some $k \geq m-1$. For example, solving this problem for $c=4$ and $m=n-l$ on M_l for the network $\tilde{D}_l$ of the previous section will yield the matrix Δ_l.

To compute M^{2h} by squaring M^h, h a nonnegative power of two, we proceed as follows. It may be shown by induction that M^h has $2h+1$ distinct blocks which are not ∞. Therefore, producing the $4h+1$ blocks sufficient to describe M^{2h} can be done by $4h+1$ inner product

computations, each one requiring m^2 inner products over vectors of length $m(2h+1)$. Producing M^{2h} from M^h therefore takes $O(\log m)$ parallel time and uses $O(hm^2mh) = O(m^5)$ processors. Since $\log(m-1)$ such multiplications suffice to solve the problem, the overall parallel running time is $O((\log m)^2)$ with $O(m^5)$ processors. We observe that paths represented in the k-fold product have at most $k+1$ vertices.

A second approach to this problem produces a $cm \times cm$ block which is an element-wise lower bound on the answer, taking $O((\log m)^3)$ parallel time but using only $O(m^3)$ processors. Each element produced in this second approach corresponds to some path length in the given network, so the bounds we obtain are sufficient for the shortest path computations in the min-cut algorithm of the preceding section.

For this second method we consider finite submatrices of M, where the matrix containing blocks with indices (i,j) for $i_1 \le i \le i_2$ and $j_1 \le j \le j_2$ is denoted $M(i_1{:}i_2, j_1{:}j_2)$. (If $i=i_1=i_2$ or $j=j_1=j_2$, we condense the notation, for example writing $M(i,j)$ when all these equalities hold.) Our consideration of finite submatrices is motivated by the following result.

LEMMA 11. For any positive integer h,

$$M^h(i,j) = M(i_a{:}i_b, j_a{:}j_b)^h(i,j),$$

where $i_a = j_a = \lceil (i+j-h)/2 \rceil$ and $i_b = j_b = \lfloor (i+j+h)/2 \rfloor$.

Proof: Since the lemma holds immediately for $h=1$, we proceed by induction. For any matrix X, $X^h = XX^{h-1} = X^{h-1}X$ for $h \ge 1$. Let X be block tridiagonal. Then

$$\begin{aligned} XX^{h-1}(i,j) &= \min\{X(i,i-1)X^{h-1}(i-1,j), X(i,i)X^{h-1}(i,j), X(i,i+1)X^{h-1}(i+1,j)\} \\ &= X^{h-1}X(i,j) \\ &= \min\{X^{h-1}(i,j-1)X(j-1,j), X^{h-1}(i,j)X(j,j), X^{h-1}(i,j+1)X(j+1,j)\}. \end{aligned}$$

By hypothesis,

$$M^{h-1}(r,s) = M(r_a:r_b,s_a:s_b)^{h-1}(r,s),$$

for all $r_a = s_a \le \lceil (r+s-(h-1))/2 \rceil$, $r_b = s_b \le \lfloor (r+s+(h-1))/2 \rfloor$. Thus

$$M^{h-1}(r,s) = M(r_a^*:r_b^*,s_a^*:s_b^*)^{h-1}(r,s)$$

where

$$r_1^* = s_1^* \le \lceil (i+j-1-(h-1))/2 \rceil = \lceil (i+j-h)/2 \rceil ,$$

$$r_2^* = s_2^* \ge \lfloor (i+j+1+(h-1))/2 \rfloor = \lfloor (i+j+h)/2 \rfloor$$

for $(r,s) \in \{(i-1,j),(i,j),(i+1,j),(i,j-1),(i,j+1)\}$, which gives our result. ▪

Our first objective is to compute $M^k(i,i)$ for k a power of two for which $k \ge m-1$. To this end, for any h a positive power of two let $a(h) = i-h/2$, $b(h) = i+h/2$ (these values are compatible with the definitions of i_a and i_b of the preceding lemma) and let $t(h)$ be the least power of two for which $t(h) \ge (h+1)m$. Define the 3×3 block matrix $Q_2(1:3,1:3) = M(i-1:i+1,i-1:i+1)$. Then define Q_{2h} as follows.

$$\begin{aligned}
Q_{2h}(1,1) &= Q_h^{t(h)}(1,1),\\
Q_{2h}(1,2) &= Q_h^{t(h)}(1,3),\\
Q_{2h}(1,3) &= \infty,\\
Q_{2h}(2,1) &= Q_h^{t(h)}(3,1),\\
Q_{2h}(2,2) &= \min\{Q_h^{t(h)}(3,3), Q_h^{t(h)}(1,1)\},\\
Q_{2h}(2,3) &= Q_h^{t(h)}(1,3),\\
Q_{2h}(3,1) &= \infty,\\
Q_{2h}(3,2) &= Q_h^{t(h)}(3,1),\\
Q_{2h}(3,3) &= Q_h^{t(h)}(3,3),
\end{aligned}$$

where minimization over matrices is element-wise.

LEMMA 12. For h a positive power of two and all $s(h) \le t(h)$,

$$M(a(h):b(h),a(h):b(h))^{s(h)}(a(h),a(h)) \ge Q_h^{t(h)}(1,1),$$

$$M(a(h):b(h),a(h):b(h))^{s(h)}(b(h),a(h)) \ge Q_h^{t(h)}(3,1),$$

$$M(a(h):b(h),a(h):b(h))^{s(h)}(a(h),b(h)) \geq Q_h^{t(h)}(1,3),$$
$$M(a(h):b(h),a(h):b(h))^{s(h)}(b(h),b(h)) \geq Q_h^{t(h)}(3,3),$$
$$M(a(h):b(h),a(h):b(h))^{s(h)}(i,i) \geq Q_h^{t(h)}(2,2),$$

and furthermore every entry in $Q_h^{t(h)}$ represents a path length in the graph for which $M(a(h):b(h),a(h):b(h))$ is the edge length matrix. ▪

Proof: Since $M(a(2):b(2),a(2):b(2)) = Q_2$, the lemma is true for $h=2$. Assume the lemma holds for some $h=h_0 \geq 2$. By the properties of plus-min multiplication we may view the matrix $M(a(h_0):b(h_0),a(h_0):b(h_0))^{s(h_0)}$ as containing shortest distances among all paths with no more than $s(h_0)$ edges in the directed graph for which $M(a(h_0):b(h_0),a(h_0):b(h_0))$ is the edge length matrix. By hypothesis we may assert that the relevant submatrices in $Q_h^{t(h_0)}$ contain shortest distances among a set of paths which contains all paths with $s(h_0)$ or fewer edges.

The key observation for the induction step is that $M(a(2h_0):b(2h_0),a(2h_0):b(2h_0))$ has no more than $t(2h_0)$ vertices, and therefore every shortest distance is over a path with no more than $t(2h_0)-1$ edges. Thus the $t(2h_0)$-fold product of Q_{2h_0} must also satisfy the lemma, as a case analysis on each $m \times m$ block shows. We omit further details. ▪

It follows from the above that we may generate the sequence $Q_2, Q_4, Q_8, \cdots, Q_k, Q_k^{t(k)}$ in $O((\log m)^3)$ parallel time using $O(m^3)$ processors. Lemmas 11 and 12 then allow us to bound $M^k(i,i)$ by $Q_k^{t(k)}(2,2)$.

Our computation of $M^k(i:i+c-1,i:i+c-1)$ is completed as follows. Define the infinite block tridiagonal matrix R as follows for all i.

$$R(i,i) = Q_k^{t(k)}(2,2) \le M^k(i,i)\ ,$$
$$R(i,i+1) = M(i,i+1)\ ,$$
$$R(i+1,i) = M(i+1,i)\ ,$$
$$R(i,j) = \infty \text{ for } j>i+1 \text{ or } j<i-1\ .$$

LEMMA 13.

$$M^k(i{:}i+c-1,i{:}i+c-1) \ge R(i{:}i+c-1,i{:}i+c-1)^k,$$

and furthermore every entry in $R(i{:}i+c-1,i{:}i+c-1)^k$ is a path length in M.

Proof: The k-fold multiple of $R(i{:}i+c-1,i{:}i+c-1)^k$ represents shortest paths in a set of paths which contains all paths of vertex length $k+1$, which is sufficient for the result. ▪

6. Computing a Maximum Flow in Any Planar Network

The computation involves $O(n)$ serial executions of the maximum flow value algorithms of the earlier sections, using the appropriate algorithms for the cases of undirected and of directed networks.

LEMMA 14. Let e be an edge of the network N. Let v be the maximum flow value in N. Let v' be the maximum flow value in the network $(N-e)$ obtained by removing the edge e from N. Then,

(i) for any maximum flow f in N, $f(e) \ge v-v'$, and

(ii) there exists a maximum flow f for which $f(e) = v-v'$.

Proof: (i) Assume there exists a maximum flow f in N, for which $f(e) < v-v'$. Since f is of value v, there is a flow $\hat{f}$ of value $\hat{v} = v-f(e)$ for which $\hat{f}(e)=0$. Then $v'<\hat{v}$, and v' cannot be the maximum flow value in $N-e$, a contradiction.

(ii) Let f' realize a flow of value v' such that $f'(e)=0$. Clearly , such a

flow exists in N, since v' is the maximum flow value in $(N-e)$. Augment f' in N to obtain a maximum flow f. The total amount of augmentation over $(s-t)$ paths cannot exceed $v-v'$. Therefore, $f(e)-f'(e) \leq v-v'$. From (i), we then get $f(e) = v-v'$. ▪

Let e, v, and v' be as in the lemma above. Let N' be obtained from N by assigning a capacity of $(v-v')$ to edge e. Then the above lemma implies the following:

COROLLARY 1. The value of a maximum flow in N' is equal to v.

COROLLARY 2. The edge e is saturated in every maximum flow in N'.

LEMMA 15. Let edge e be saturated for every maximum flow f in N. Then the direction of $f(e)$ is the same for all f.

Proof: Assume in the undirected case that two maximum flows f and f' have opposite directions for $f(e)$ and $f'(e)$. Then $f/2+f'/2$ is a maximum flow with zero flow in edge e, a contradiction. ▪

For any l, let the networks $N_1, N_2, \cdots, N_l$ be defined as follows:

(i) $N_1 = N$.

(ii) Let v_i be the maximum flow value in N_i.

Let v_i' be the maximum flow value in the network $N_i - e_{j_i}$, for some $e_{j_i} \in E$.

N_{i+1} is obtained from N_i by assigning a capacity of $(v_i - v_i')$ to edge e_{j_i}.

(iii) Let f_i be a maximum flow in N_i.

Using these definitions we obtain the following.

LEMMA 16.

(i) $v_1=v_2=\cdots=v_l$, and

(ii) $f_{i+1}(e_{j_i})=f_k(e_{j_i})$ for $i<k\leq l$.

Proof: (i) Follows from Corollary 1.

(ii) For $k>i$, f_k is also a maximum flow in N_{i+1} since capacities have only been reduced in going from N_{i+1} to N_k. The rest follows from Corollary 2 and Lemma 15. ▪

LEMMA 17. Let e, v, and v' be as in Lemma 14. Let $(v-v')>0$. Let N^1 and N^2 be obtained from N by assigning positive capacities $c(e)=v-v'$ and $c(e)<v-v'$ respectively, to edge e. Then the direction of $f(e)$ in a maximum flow f in N^1 can be computed in $O((\log n)^{2})$ parallel time using $O(n\log n)$ processors.

Proof: The result is trivial in the directed case, so let N be undirected. The value of a min-cut in N^2 is less than v. Otherwise, a flow of value greater than v' can be obtained in $(N-e)$. Since all $(s-t)$-cuts which do not involve edge e in N^2 have value not less than v, a minimum cut in N^2 must involve edge e. Thus we may compute as follows. As described in Section 3, obtain consistency in a clockwise ordering of all faces on the minimum cut-cycle, which corresponds to a shortest path p_i' from some face F_i on the μ-path to F_i' in the network D'. Let p_i' cross edge e from face F_r to face F_s. If $e=(p,q)$ and going from p to q is clockwise in F_s, then the direction of flow in edge e in N' is from p to q. Otherwise, it is from q to p. ▪

THEOREM 3. A maximum flow in a planar network can be constructed in $O(n(\log n)^2)$ parallel time using a number of processors proportional to the number required to find a cut of minimum capacity.

Proof: Let $e_1, e_2, \cdots, e_{|E|}$ be the edges of the given network N. Let $e_{j_i}=e_i$

for $1 \le i \le |E|$ in Lemma 16. Construct N_{i+1} from N_i, $1 \le i \le |E|$, by application of the flow value algorithm. This can be done in $O((\log n)^2)$ parallel time. Then obtain the direction of a maximum flow f_{i+1} in edge e_i is obtained as described in Lemma 17 (in the case of undirected networks). This takes $O((\log n)^2)$ extra time. So by Lemmas 15 and 16, $f_{|E|+1}$ saturates every edge in $N_{|E|+1}$ and is a maximum flow in N. The overall running time is $O(n(\log n)^2)$. ▪

In [9], Hassin shows a new method of computing a maximum flow function in (s,t)-planar networks. This method does not appear to generalize to planar networks in general. For an (s,t)-planar network, we can find a μ-path of zero length, consisting of just one face F_1. A shortest path tree rooted at F_1 or F_1' in network D' defines not only the value of a maximum flow but also, by an elegant transformation, a maximum flow itself. Hence we can obtain the following lemma.

> LEMMA 18. A maximum flow can be computed in either an undirected or a directed (s,t)-planar network in $O((\log n)^2)$ parallel time using $O(n^4)$ processors. ▪

References

[1] Bondy, J. A. and U. S. R. Murty, *Graph Theory with Applications*, American Elsevier, 1977.

[2] Csanky, L., Fast parallel matrix inversion algorithms, *SIAM J. Comput.* **5**, 4(Dec. 1976) 618-623.

[3] Dekel, E., D. Nassimi, and S. Sahni, Parallel matrix and graph algorithms, *SIAM J. Comput.* **10**, 4(Nov. 1981) 657-675.

[4] Ford, L. R. and D. R. Fulkerson, Maximal flow through a network, *Canadian J. Math.* **8**, 3(1956) 399-404.

[5] Gomory, R. and T. C. Hu, Multiterminal network flows, *SIAM J. Appl. Math.* **9**, 4(Dec. 1961) 551-570.

[6] Goldschlager, L. M., A unified approach to models of synchronous parallel machines, *Proc. Tenth ACM Symp. on Th. Comp. Sci.*, May, 1978.

[7] Goldschlager, L., R. Shaw, and J. Staples, *The Maximum Flow Problem is Log Space Complete for P*, (manuscript) Dept. of Comp. Sci., Univ. of Queensland, Australia, May 1981.

[8] Harary, F., *Graph Theory*, Addison-Wesley, 1969.

[9] Hassin, R., Maximum flow in (s,t)-planar networks, *Inf. Proc. Letters* **13**, 3(Dec. 1981) 107.

[10] Hirschberg, D. S., A. K. Chandra, and D. V. Sarwate, Computing connected components on parallel computers, *C.ACM* **22**, 8(1977) 461-464.

[11] Itai, A. and Y. Shiloach, Maximum flow in planar networks, *SIAM J. Comput.* **8**, 2(May 1979) 135-150.

[12] Ja'Ja', J., and J. Simon, *Parallel Algorithms in Graph Theory: Planarity Testing*, Tech. Rept. CS 80-14, Dept. of Comp. Sci., The Penn. State Univ., 1980.

[13] Lawler, E., *Combinatorial Optimization, Networks and Matroids*, Holt, Rinehart, and Winston, 1976.

[14] Lev, G. F., N. Pippenger, and L. G. Valiant, A fast parallel algorithm for routing in permutation networks, *IEEE Trans. on Cmptrs.* **C-30**, 2(Feb. 1981) 93-100.

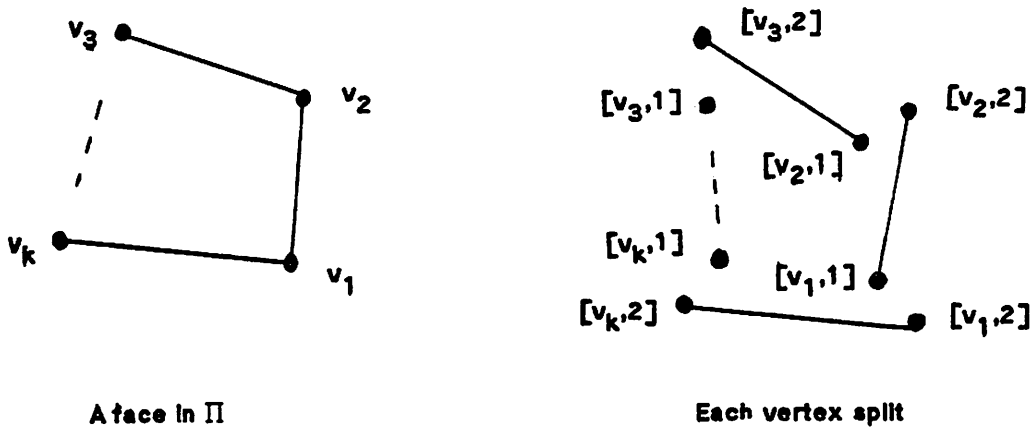
[15] Lovàsz, L., *On determinants, matchings, and random algorithms* (manuscript).

[16] McLane, S., A combinatorial condition for planar graphs, *Fund. Math.* **28** (1937) 22-32.

[17] Reif, J., Minimum $(s-t)$-cut of a planar undirected network in $O(n\log^2 n)$ time, *Proc. Eighth Int. Colloq. on Automata, Languages, and Programming*, July 1981, Acre, Israel, (published as *Lecture Notes on Computer Science*, 115, Springer-Verlag, 1981) 56-67.

[18] Schwartz, J. T., Probabilistic algorithms for verification of polynomial identities, in *Lecture Notes in Computer Science*, 72, Springer-Verlag, 1979, 200-215.

[19] Shiloach, Y., and U. Vishkin, *IBM Israel Tech. Rept.*, Feb. 1981.



vertex	CL	ACL
$[v_1,1]$	-	$[v_2,2]$
$[v_1,2]$	$[v_k,2]$	-
$[v_2,1]$	-	$[v_3,2]$
$[v_2,2]$	$[v_1,1]$	-
$[v_3,1]$	-	$[v_4,.]$
$[v_3,2]$	$[v_2,1]$	-
$[v_k,1]$	$[v_{k-1},.]$	-
$[v_k,2]$	-	$[v_1,2]$

Figure 1. Example of the ordering of edges in a face $(v_1,\cdots,v_k,v_1)$ in terms of CL and ACL.

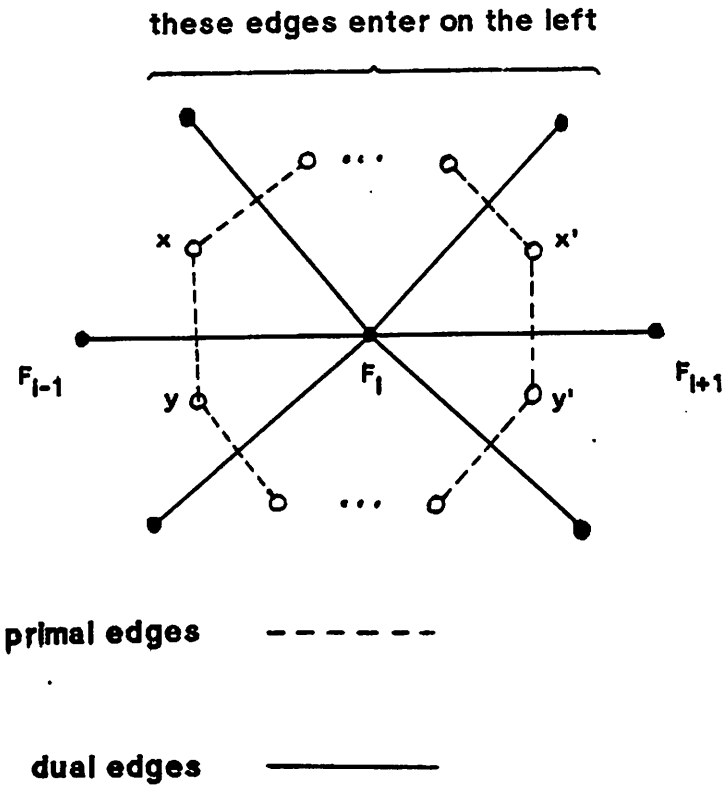


Figure 2. Identifying left-entering dual edges. The dual edges that enter dual vertex F_i on the left of the μ-path $(\cdots,F_{i-1},F_i,F_{i+1},\cdots)$ are defined by precisely those edges in the primal face $(x,\cdots,x',y',\cdots,y,x)$ containing F_i which are in the segment $(x,\cdots,x')$.

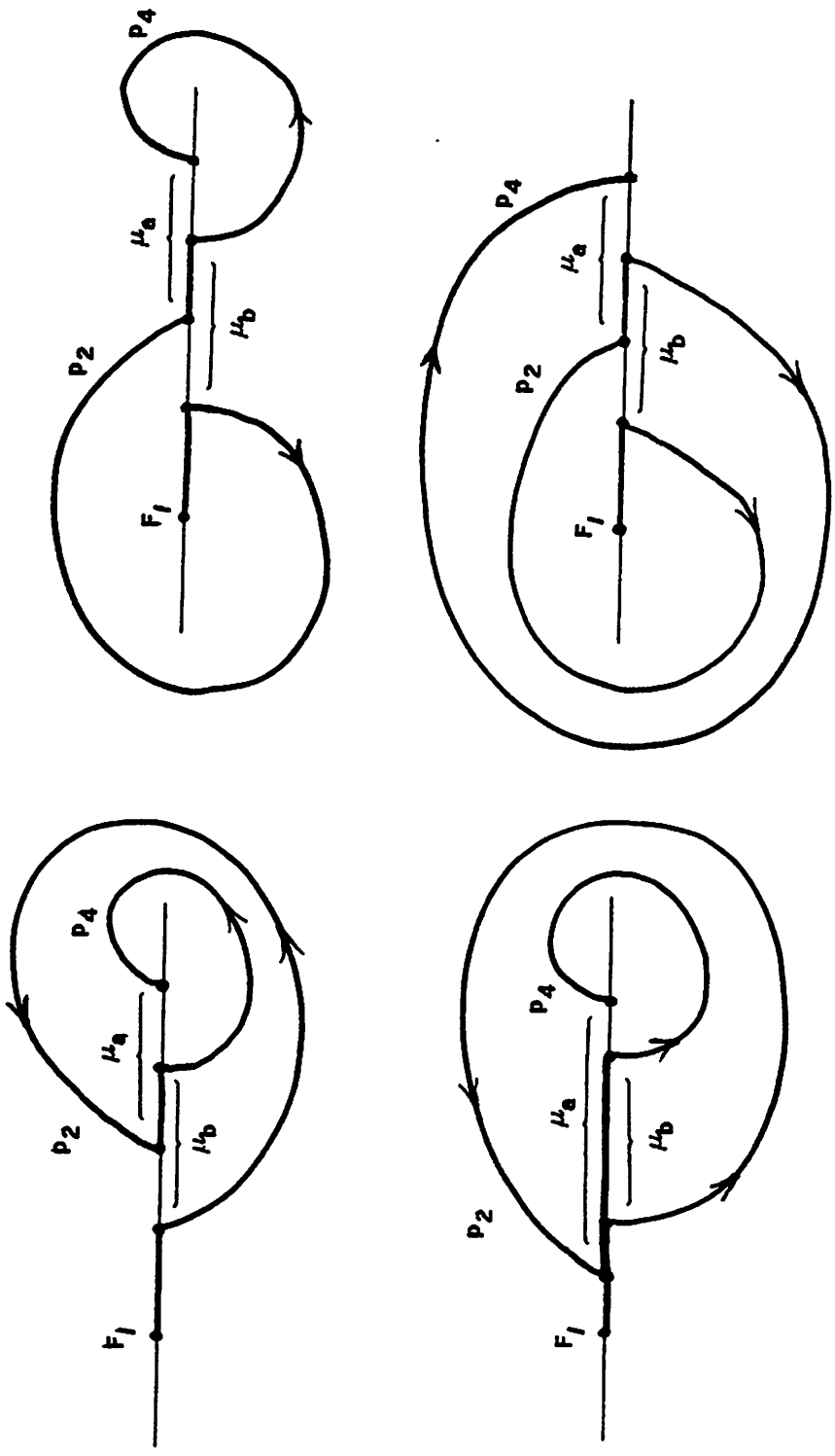


Figure 3. Four cases of a pseudosimple path $(p_1,p_2,p_3,p_4,\cdots)$ beginning from μ-vertex F_i. In each case, if the path is to be closed and not to cross either p_2 or p_4, there must ensue some enclosing subpath which leaves μ in the region μ_a and reenters μ in the region μ_b.

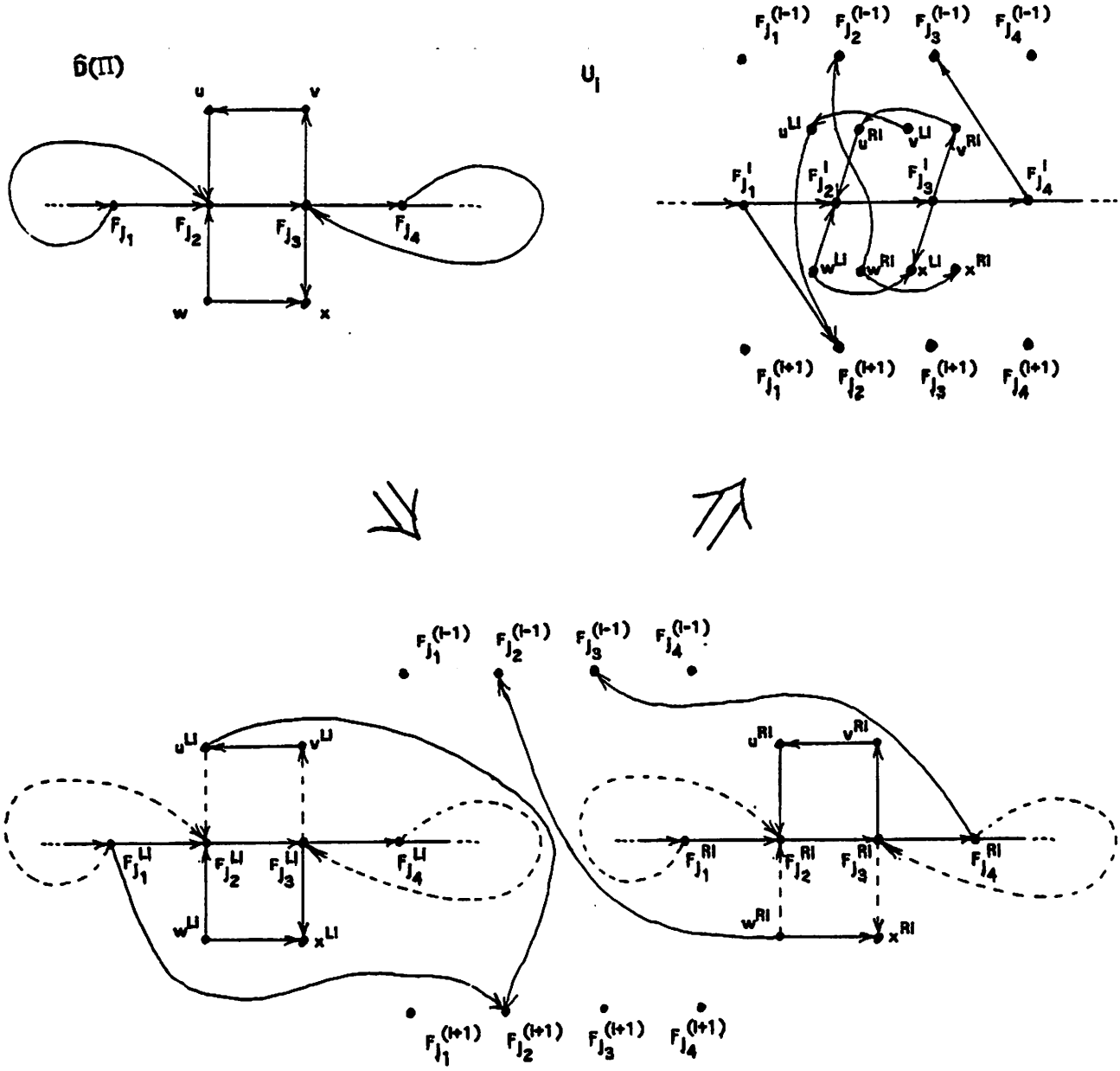


Figure 4. The construction of *unit* U_i of $\tilde{D}$ from $\hat{D}(\Pi)$. In the first step, $\hat{D}$ is replicated, the edges shown as dashed are removed, and new edges are introduced which connect to new copies of the μ-vertices. The construction is completed by identifying the μ-vertices (but no others) in the replicas.