アルゴリズム 2－2
（1988. 7. 25）

平面グラフで内素な道を求めるアルゴリズム

鈴木均　　赤間長浩　　西関隆夫

東北大学 工学部 通信工学科

本報告では平面無向グラフで内素な道を求める効率の良いアルゴリズムを与える．平面グラフGとG上の2点sとtが与えられたときに，本アルゴリズムは，まずsとtを結ぶG上の内素な道の本数の最大値を求め，次に内素な道を実際に求める．アルゴリズムの計算時間はO(n log n)である．ここでnはグラフGの点数である．

Algorithms for Finding Internally Disjoint Paths in Planar Graphs

Hitoshi SUZUKI, Takehiro AKAMA, and Takao NISHIZEKI

Department of Electrical Communications, Faculty of Engineering
Tohoku University, Sendai 980, Japan

This paper gives an efficient algorithm for finding internally disjoint paths in planar undirected graphs. Given a planar graph G together with two specified vertices s and t, the algorithm first determines the maximum number of internally disjoint paths between s and t in G. Then the algorithm actually finds these paths by using divide-and-conquer method. The algorithm runs in $O(n \log n)$ time, where n is the number of vertices in G.

1. Introduction

Let s and t be two specified vertices in a graph G. Paths connecting s and t are defined to be *internally disjoint* if they have only vertices s and t in common. One can find a maximum number of internally disjoint paths in a general graph G in $O(\text{MIN}\{k_M m, m\sqrt{n}\})$ time by solving the maximum flow problem for a graph obtained from G [ET], where m is the number of edges in G, n is the number of vertices, and k_M is the maximum number of internally disjoint paths.

In this paper we present an efficient algorithm for finding a maximum number of internally disjoint paths in a planar undirected graph G. The algorithm runs in $O(n\log n)$ time. We do not use any flow algorithm, but use the divide-and-conquer method based on properties of planar graphs.

In section 2 we outline an $O(n\log n)$ algorithm which determines the maximum number k_M of internally disjoint paths. Then in section 3, using a divide-and-conquer strategy, we design an $O(n\log n)$ algorithm which actually find k disjoint paths for any $k \leq k_M$.

2. Determining the maximum number

Reif [Rei] and Hassin and Johnson [HJ] have given $O(n\log^2 n)$ algorithms which find a maximum s-t (edge-)cut and a maximum flow in an undirected planar graph. We employ their ideas, but need to introduce several new ideas.

Let $G = (V, E)$ be an undirected simple planar graph with vertex set V and edge set E. We denote by $V(G)$ the vertex set of G, and by $E(G)$ the edge set of G. Let n be the number of vertices in G, that is, $n = |V|$. We assume that G is connected and embedded in the plane R^2. The image of G on R^2 is denoted by $Image(G)$. A face of planar graph G is a connected component of $R^2 - Image(G)$. Denote by $V(f)$ the set of vertices on the boundary of a face f, and by $\Gamma(G)$ the set of faces of G. We say that a vertex set $X \subset V$ *separates* two vertices s and t or X is an s-t *separating set* if s and t are in different components of $G - X$. The following theorem, due to Menger, is well-known.

THEOREM 1 [Men]. Let s and t be two non-adjacent vertices of an undirected graph G. Then G has k internally disjoint paths if and only if there is no s-t separating set X with $|X| < k$. ■

We may assume without loss of generality that t lies on the outer face boundary of G. Furthermore we may assume that s and t are not adjacent. If s and t are adjacent, then a maximum number of internally disjoint paths in G can be constructed from a maximum number of internally disjoint paths in graph $G - \{(s,t)\}$ by adding the path of a single edge (s,t).

Let $\Gamma(G)$ be the set of faces of G. We construct from G a new planar graph $G^* = (V^*, E^*)$ as follows (See Fig. 1):

(1) for each face $f \in \Gamma(G)$, introduce a new vertex f^* and embed f^* in face f; and

(2) join vertex f^* with every vertex in $V(f)$.

Thus $V^* = V(G) \cup \Gamma^*(G)$ and $E^* = \{(f^*, v) | f^* \in \Gamma^*(G), v \in V(f)\}$, where $\Gamma^*(G) = \{f^* | f \in \Gamma(G)\}$. Note that none of the edges in G remains in G^* and that G^* is not a dual of G unlike in [Rei] and [HJ].

We say that a cycle (i.e. a closed walk) C in G^* *separates* two vertices $s, t \in V(G^*)$ if $Image(C)$ separates two points s and t in R^2, that is, s and t are in different connected components of $R^2 - Image(C)$. We often call C an s-t *separating cycle*. We denote by $leng(W)$ the length (i.e. the number of edges) of a walk W in a graph G^*. Since G^* is a bipartite graph, walks connecting $v, v' \in V(G)$, walks connecting $f^*, f'^* \in \Gamma^*(G)$, and cycles all have even lengths in G^*. From Menger's theorem one can easily derive the following lemma.

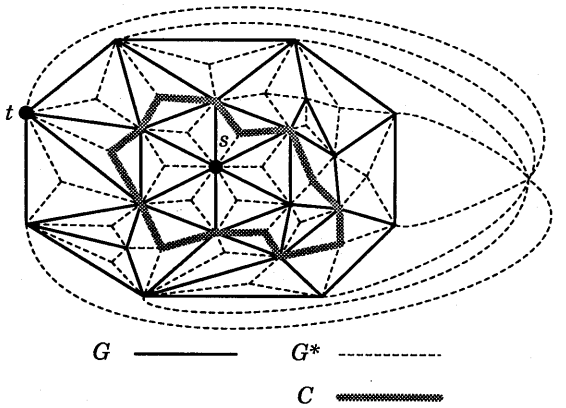

Fig. 1 G, G' and an s-t separating cycle C in G^*.

Lemma 1. Let $k \geq 2$. A planar graph G has k internally disjoint s-t paths if and only if all the s-t separating cycles in G^* have lengths at least $2k$.

Proof. By Theorem 1 it suffices to show that a vertex set $V \subset V(G)$ with $|X| < k$ separates s and t if and only if G^* has a cycle C of $leng(C) < 2k$ separating s and t.

If: Suppose that G^* has a cycle C such that $leng(C) < 2k$ separating s and t. We may assume that C is a simple cycle. Then $|V(C) \cap V(G)| = leng(C)/2 < k$, and $V(C) \cap V(G)$ is the vertex set which separate s and t in G.

Only if: Suppose that a vertex set $X \subset V(G)$ with $|X| < k$ separates s and t in G. We may assume that X is minimal among these sets. There exists a closed curve $J \subset R^2$ such that:

(1) J separates s and t on R^2;

(2) $J \cap Image(G) = Image(X)$; and

(3) Let $v_0, v_1, ..., v_{|X|-1}$ be the vertices in X, and assume that the vertices appear in this order clockwise going on J. Then for any $1 \leq i \leq |X| - 1$, v_i and v_{i+1} lie on the boundary of the same face f_i of G and the interval of J between v_i and v_{i+1} is in f_i.

Clearly, the cycle $C = v_0, f_0^*, v_1, f_1^*, ..., v_0$ separates s and t, and $leng(C) = 2|X| < 2k$. Q.E.D.

Thus it suffices to find a shortest s-t separating cycle in G^*. We now use Reif's idea to efficiently find such a cycle in a planar graph. Find a shortest s-t paths P_{st} in G^*, and let $Q = P_{st} - \{s, t\}$. Construct from $G^* - \{s, t\}$ a new planar graph G^*_{slit}. G^*_{slit} is generated by slitting apart the path Q into two paths Q' and Q'', duplicating the vertices and edges of Q as follows. Each vertex v on Q is replaced by new vertices v' and v''. Any edge (v, w) that is not in Q but is incident with vertex v in Q is replaced by (v', w) if (v, w) is to the left of Q in G^*, and by (v'', w) if (v, w) is to the right of Q in G^*. Once G^*_{slit} is generated, it is searched as follows. Let v be a midpoint of the original path Q. A shortest path computation in G^*_{slit} is performed using v' as the source and v'' as the destination. The shortest v'-v'' path P_v corresponds to the shortest s-t separating cycle in G^* containing v. The graph G^*_{slit} is split into two subgraphs along P_v including vertices and edges on P_v in both subgraphs. Our algorithm recurses on each half. Among the $r = |V(Q)|$ paths so identified, the shortest path will correspond to the shortest separating cycle. The work performed across each level of recursion is dominated by the shortest path computation, which are $O(n)$ time since the breadth-first search simply suffices in our case. The time to handle all levels of recursion is $O(n \log n)$, since there are

$O(\log r) = O(\log n)$ levels of recursion. However we need two new ideas here to make the recursive algorithm run actually in $O(n \log n)$ time. If all the paths P_v were listed, then the algorithm would require $O(rn) = O(n^2)$ time. Instead our algorithm finds a forest F in G^*_{slit} containing path P_v for every vertex v on Q. At each level of recursion the breadth-first search is performed in each of the subgraphs of G^*_{slit} bounded by the forest F obtained at the preceding level, and hence spends $O(n)$ time. Once such a forest F is found, one can find the distance between v' and v'' for all v on Q total in $O(n)$ time by using a linear algorithm to solve "the nearest common ancestor problem for trees" in [GT]. Thus we have:

THEOREM 2. The maximum number k_M of internally vertex-disjoint s-t paths in a planar graph G can be determined in $O(n \log n)$ time. ■

3. Finding disjoint paths

We now outline an algorithm PATH for finding k internally disjoint s-t paths in G for any positive integer $k \leq k_M$.

STEP 1: Find a vertex set $X \subset V(G)$ such that X separates s and t and $|X| = k$; {if there is no such a vertex set $X \subset V$, then one can reduce G to a smaller graph having such a vertex set X by deleting "redundant" vertices and collapsing "redundant" faces.}

STEP 2: Split G into two graphs, the "inside" containing s and the "outside" containing t (see Fig. 2);

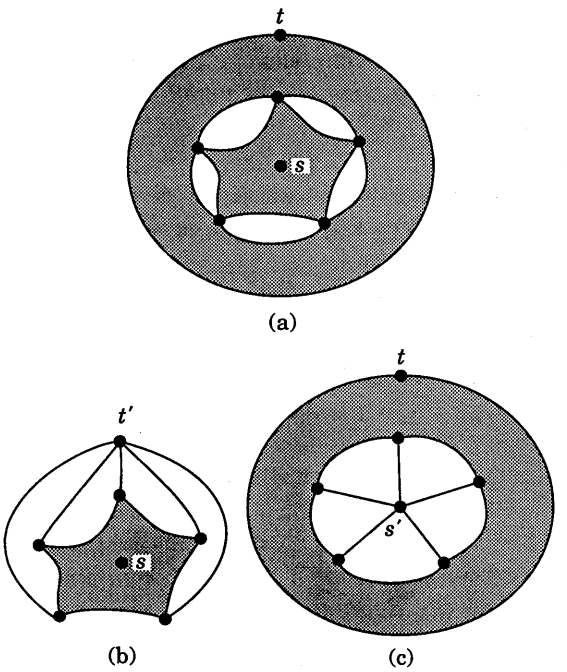

Fig. 2 Graph G and its split graphs G_1 and G_2.

STEP 3: Construct a new planar graph G_1 from the "inside" by adding new vertex t' and joining t' with every vertex in X. Similarly construct G_2 from the "outside" by adding a new vertex s' and joining s' with every vertex in X. We call G_1 and G_2 above *split graphs of* G; and

STEP 4: Find k internally disjoint s-t' paths in G_1 and k internally disjoint s'-t paths in G_2 by recursively applying Steps 1, 2, and 3 above. Construct k internally disjoint s-t paths by combining these two sets of paths.

One can prove the following two lemmas.

LEMMA 2. Let $v \in V(G^*) - \{s,t\}$ be any vertex which is adjacent with neither s nor t. Let C be a shortest s-t separating cycle in G^* containing v. Let G_v^* be the graph obtained from G^* by contracting all edges incident with v, and let C' be the cycle in G_v^* corresponding to C. Then C' is a shortest s-t separating cycle in G_v^* containing v, and $leng(C') = leng(C) - 2$.

PROOF. Since neither s nor t is identified with v in G_v^*, cycle C' of G_v^* separates s and t.

Since C is a shortest cycle, C passes through vertex v exactly once. The two edges on C incident with v are contracted in G_v^*. Therefore $leng(C) = leng(C') + 2$.

We next show that $leng(C_v') \geq leng(C')$ for any shortest s-t separating cycle C_v' in G_v^* which contains v. By adding to C_v' two edges which were incident with v in G^*, one can construct an s-t separating cycle C_v in G^* which contains v is induced. Therefore

$$leng(C_v) = leng(C_v') + 2.$$

On the other hand, by the assumption

$$leng(C_v) \geq leng(C).$$

Therefore $leng(C_v') \geq leng(C')$. Q.E.D.

LEMMA 3. Assume that a vertex $v \in V(G^*) - \{s,t\}$ is adjacent with neither s nor t, that every cycle in G^* containing v and separating s and t has length$\geq 2k+2$, and that G' is obtained from G by applying the following operation (1) or (2):

(1) if $v \in V(G)$, then delete v from G; or

(2) if $v = f^* \in \Gamma^*(G)$, then contract all vertices in $V(f)$ into a single vertex v.

Then G has k internally disjoint s-t paths if and only if G' has k internally disjoint s-t paths.

PROOF. Note that both (1) and (2) for G correspond to an operation for G^* to contract all edges incident with v. Suppose that G has k internally disjoint s-t paths. Then by Lemma 1, every s-t separating cycle in G^* has length$\geq 2k$. Let C be a shortest s-t separating cycle in G'^*. If C contains v, then by Lemma 2 $leng(C) \geq 2k$. If C does not contain v, then C is a separating cycle also in G^*, and hence has length $\geq 2k$. Therefore by Lemma 1 G' has k internally disjoint s-t paths.

For the case in which operation (1) is applied, k internally disjoint s-t paths in G' are internally disjoint also in G. On the other hand, for the case operation (2) is applied, k internally disjoint paths in G can be obtained from those in G' by adding some edges in $E(f)$. Q.E.D.

As known from Lemmas 2 and 3, once one finds a shortest separating cycle C in G^* containing a vertex v, one can repeat the contraction of all edges incident with v until either the length of C becomes $2k$ or the distance between s and t in G^* becomes at most four, with preserving k internally disjoint s-t paths in G. For the case that $leng(C)$ becomes $2k$, G^* has a separating set X with $|X| = k$, and hence we can apply to G the divide-and-conquer strategy illustrated in Fig. 2. For the case that the distance beween s and t in G^* becomes

2 or 4, we can find internally disjoint paths in $O(n)$ time by using a simple method. The details will be presented later.

In order to balance the "sizes" of split graphs G_1 and G_2, we choose as v a midpoint on a shortest s-t path P_{st} in G^*. Then both the distance between s and t' in G_1^* and the distance between s' and t in G_2^* are at most about one half of the distance between s and t in G^*. Repeating this operation, we split G into several graphs for which the distance is 2 or 4. Note that the distance between s and t in G^* is even, and that if the distance is at most 4, then the distances for split graphs do not necessarily decrease.

The detail of the algorithm PATH is as follows.

```
procedure PATH(G, s, t);
  begin
    construct G* from G;
    find a shortest s-t path P_st in G*;
    if leng(P_st) > 4 then
      begin
        find a shortest s-t separating cycle C which contains a midpoint v on P_st;
        while leng(P_st) > 4 and leng(C) > 2k do
          contract all edges incident with v in G*, and update G, P, and C; {see Lemmas 2 and 3}
      end;
    if leng(P_st) ≤ 4 then
      find k internally disjoint s-t paths in G {the detail will be given later}
    else {leng(P_st) > 4 and leng(C) = 2k}
      begin
        split G into G_1 and G_2;
        PATH(G_1, s, t');
        PATH(G_2, s', t);
        construct k internally disjoint s-t paths in G by concatenating s-t' paths in G_1 and s'-t paths in
        G_2
      end
  end;
```

Since the paths obtained by procedure PATH may pass through vertices which are surrogates for collapsed faces, they are not necessarily complete s-t paths in the original graph G. However, one can immediately construct k internally disjoint paths in G by adding some edges on the boundary of collapsed faces.

Algorithm for the case $leng(P_{st}) \leq 4$.

We next show how to find k internally disjoint s-t paths for the case $leng(P_{st}) \leq 4$. First consider the case in which $leng(P_{st}) = 2$. In this case one may assume that s and t lie on the outer face boundary. Then one can find k internally disjoint paths in G simply by the "uppermost path algorithm" [FF]. That is, find the uppermost path P in G, i.e., the clockwise s-t path in the outer face boundary, delete the vertices in $V(P) - \{s,t\}$ from G, and let G be the resulting graph. Repeating this operation k times, one can find k paths in $O(n)$ time.

Next consider the case in which $leng(P_{st}) = 4$. In this case exactly one vertex v of G lies on path $P_{st} - \{s,t\}$ in G^*. Therefore one may assume that t and v lie on the outer face boundary and v and s lie on

an inner face boundary. Then one may further assume that one of the four s-t paths P_1, P_2, P_3 and P_4 drawn by thick lines in Fig. 3 is included in the k disjoint paths. Each P_i is a concatination of an s-v path in the inner face boundary and a v-t path in the outer boundary. For some i, $G-(V(P_i)-\{s,t\})$ has $k-1$ internally disjoint s-t paths. These $k-1$ paths can be found by the uppermost path algorithm since s and t lie on the new outer boundary. Thus k internally disjoint paths in G can be found in $O(n)$ time.

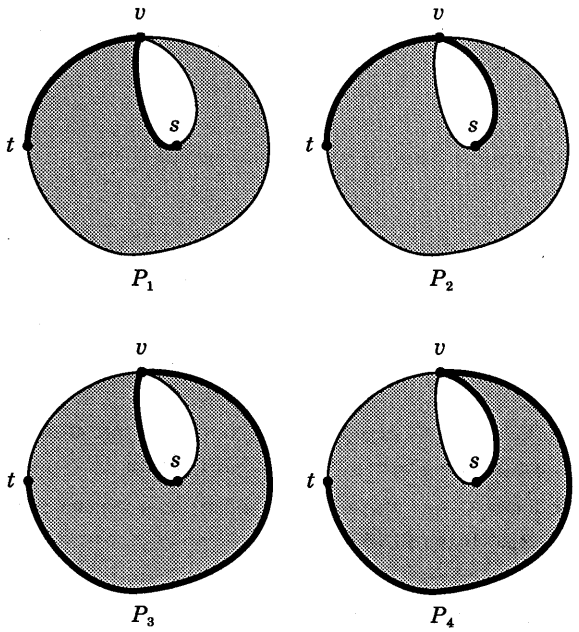

Fig. 3 Four s-t paths P_1, P_2, P_3 and P_4.

Since the description of PATH has been completed, we next analyze the execution time $T(m,r)$ of PATH(G,s,t), where m is the number of edges in G and r is the distance between s and t in G^*. A shortest separating cycle containing v can be found in $O(m)$ time simply by the breadth-first search for G^*_{slit}. The time needed to update G, P_{st}, and C when deleting redundant vertices and collapsing redundant faces is proportional to the number of deleted or contracted edges. Therefore the execution time of PATH, excluding the time required by the recursive calls, is $O(m)$. Thus, if $r>4$, then

$$T(m,r) \leq T(m_1,r_1)+T(m_2,r_2)+O(m),$$

where m_1 and m_2 are the number of edges in G_1 and G_2 respectively, and r_1 and r_2 are the distance between s and t in G_1^* and G_2^*, respectively. As shown above, $T(m,4), T(m,2)=O(m)$. Furthermore we claim

(a) $m_1+m_2 \leq m+2k$;

(b) $rk/2 \leq m$; and

(c) $r_1, r_2 \leq r/2+2$.

Clearly (a) and (c) hold. Let r' be the length of a shortest s-t path in G, then $r'k \leq m$ since G has k internally disjoint paths. Furthermore $r/2 \leq r'$. Hence (b) holds. Solving the recursion above by using (a), (b), and (c), we have $T(m,r)=O(m\log r)=O(n\log n)$.

On the other hand, a standard technique known in network flow theory can reduce the problem for finding k internally vertex-disjoint s-t paths in an undirected graph to the problem for finding a flow of value k in a directed graph. Therefore one can find k internally disjoint paths in $O(kn)$ time by using a flow algorithm. Thus we can conclude:

THEOREM 3. Given a planar graph G with two specified vertices s and t, a set of k internally vertex-disjoint s-t paths can be found in $O(\mathrm{MIN}\{kn, n\log n\})$ time for any positive integer $k \leq k_M$, where k_M is the maximum number of paths. ■

Acknowledgment

We would like to thank Professor N. Saito for many stimulating discussions.

Rererences

[ET] S. Even and R. E. Tarjan, Network flow and testing graph connectivity, SIAM J. Compt. 4, 4, pp. 507-518 (1975).

[FF] L. R. Ford and D. R. Fulkerson, Maximal flow through a network, Canad. J. Math., 8, pp. 399-404 (1956).

[GT] H. N. Gabow and R. E. Tarjan, A linear-time algorithm for a special case of disjoint set union, Journal of Computer and System Sciences, 30, pp. 209-221 (1985).

[HJ] R. Hassin and D. B. Johnson, An $O(n\log^2 n)$ algorithm for maximum flow in undirected planar networks, SIAM J. Compt., 14, pp. 612-624 (1985).

[Men] K. Menger, Zur allgemeinen Kurventheorie, Fund. Math.10, pp. 95-115 (1927).

[Rei] J. H. Reif, Minimum s-t cut of a planar undirected network in $O(n\log^2(n))$ time, SIAM J. on Compt., 12, pp. 71-81 (1981).