THE TWO DISJOINT PATH PROBLEM AND WIRE ROUTING DESIGN

T. Ohtsuki

Dept. of Electronics and Communication Engineering
School of Science and Engineering
Waseda University, Tokyo, Japan 160

Abstract. A new polynomial time algorithm is presented for finding two vertex-disjoint paths between two specified pairs of vertices on an undirected graph. An application of the two disjoint path algorithm to the automatic wire routing is also discussed.

1. Introduction

This paper considers the problem of finding, given an undirected graph $G = (V, E)$ and four vertices $s_1, t_1, s_2, t_2 \in V$, two vertex-disjoint paths, P_1 from s_1 to t_1 and P_2 from s_2 to t_2 . The more general problem, called disjoint connecting path or discrete multi commodity flow, of finding pairwise disjoint paths between pairs of vertices is NP-complete [1] and remains NP-complete for planar graphs [2]. The complexity of this problem is open for any fixed $k \geq 3$, but can be solved in $O(|E|)$ operations if $k=2$ and G is planar or chordal (triangulated) [3]. Also, an $O(|V| \cdot |E|)$ time algorithm for the general two path problem has been presented very recently [4].

In this paper, a different $O(|V| \cdot |E|)$ time algorithm is presented for solving the two disjoint path problem on general undirected graphs. The outline of the algorithm is described in Section 2 and its complexity is considered in Section 3. Finally, an application of this algorithm to the computer-aided wire routing design is discussed in Section 4.

2. The Two Disjoint Path Algorithm

In what follows, the two disjoint path problem under consideration is abbreviated by TPP. The convention "TPP is true (false)" is also used to indicate the existence (nonexistence) of the desired pair of paths, respectively. The set of vertices constructing a path P is also denoted by P and the set of edges constructing it is denoted by $E(P)$. Sometimes the notation $P(s,t)$ is used to indicate a path with source s and sink t . For two vertices $u, v \in P$, the subpath of P between u and v is denoted by $P[u;v]$. Throughout this paper, disjoint means vertex-disjoint.

Let $G_0 = (V_0, E_0)$ be a given undirected graph. Without loss of generality, we assume that G_0 is 2-connected. The outline of the presented algorithm is as follows.

Step 1. Find two disjoint paths between $\{s_1, t_1\}$ and $\{s_2, t_2\}$. We assume that two paths $P(s_1, t_2)$ and $Q(s_2, t_1)$ are found, otherwise TPP turns out to be either true or false.

Step 2. Find all the bridges $B = (B_1, B_2, \dots)$ with respect to $P \cup Q$, where a bridge B is either i) a single edge $e = (x, y) \in E_0 - E(P) - E(Q)$ with $x, y \in P \cup Q$ or ii) a maximal subgraph of $G = (V, E_0 - E(P) - E(Q))$ with at least one vertex $x \in V_0 - P - Q$ such that, for every other vertex y of B , there exists a path $R(x, y)$ without intersecting P or Q except at y . An example of the set of bridges is shown in Fig. 1.

An attachment of a bridge B is its vertex which belongs to $P \cup Q$. The set of bridges with every attachment on P or on Q is denoted by B^P or B^Q , respectively. The set of bridges with attachments on both P and Q is denoted by B^{PQ} .

Step 3. If there exist three vertices a, b, c on P in this order and two bridges $B_i \in B^{PQ}$ and $B_j \in B^P$ such that $b \in B_i$ and $a, c \in B_j$, replace the subpath $P[a; c]$ of P so as to go through B_j as shown in Fig. 2 and update the set B with respect to the new P . This operation is repeated until no such pair of B_i and B_j exists. Perform the same operation for path Q .

Fig. 1. Set of bridges.

Step 4. Discard all the bridges in B^P and B^Q . Henceforce we may assume that all the bridges belongs to B^{PQ} . Therefore TPP is true if there exist two disjoint paths $P_1(a,d)$ and $P_2(b,c)$ without traversing $E(P) \cup E(Q)$, where a, b, c, d are distinct bridge attachments such that s_1, a, b, t_2 are in this order on P and s_2, c, d, t_1 are in this order on Q . A pair of bridges (B_1, B_2) is alternating if B_1 and B_2 include $P_1(a,d)$ and $P_2(b,c)$ as above, respectively (see Fig. 3).

Step 5. If there exists an alternating pair of bridges, then TPP is true. Otherwise go to the next step.

Step 6. It is assumed that there exists at least one bridge having two or more attachments on each of P and Q , since otherwise TPP is obviously false. Discard all the bridges not possessing the above property.

For each bridge $B \in B$, let a and b (c and d) be its attachments on $P(Q)$ closest to s_1 and t_2 (s_2 and t_1), respectively, and $G_0(B)$ be the section graph of G_0 determined by the vertices of $B \cup P[a;b] \cup Q[c;d]$ (see Fig. 4). For each bridge B , call the binary recursive function $f(G_0(B), P[a;b], Q[c;d])$ and let $B \leftarrow B - \{B\}$. TPP is true if $f = \text{"yes"}$ and false if $f = \text{"no"}$ for every $G_0(B)$.

The binary recursive function $f = (G, P, Q)$ is defined as follows. G is a graph with two disjoint paths $P(s_1, t_2)$ and $Q(s_2, t_1)$ specified. Furthermore G has a single bridge with respect to $P \cup Q$ containing attachments s_1, s_2, t_1, t_2 . f returns "yes" ("no") if two disjoint paths $P_1(s_1, t_1)$ and $P_2(s_2, t_2)$ exist (do not exist). Without loss of generality we assume that G is 2-connected.

Fig. 2. Elimination of B^P .

Fig. 3. Alternating pair of bridges.

Fig. 4. Section graph $G_O(B)$.

The following steps (Steps 7 - 14) are performed in f .

Step 7. Find a path $P'(s_1, t_2)$ which is disjoint with $P(s_1, t_2)$ and $Q(s_2, t_1)$. Existence of P' is clear from the definition of bridge.

Step 8. Find all the bridges $B = \{B_1, B_2, \dots\}$ with respect to $P \cup Q \cup P'$ by means of the same operation as in Step 2.

Step 9. Perform the same operations as Steps 3, 4 with respect to P' , and pertinently update the path P' and the set B . Note that each bridge must have an attachment on $P' - \{s_1, t_2\}$ and another on $P \cup Q - \{s_1, t_2\}$.

Step 10. If there exists a bridge $B_i \in B$ having an attachment $a \in P - \{s_1, t_2\}$ and another attachment $b \in Q$, then return "yes", otherwise go to the next step. Note that TPP is true if such a bridge exists. This can be proved by recalling that B_i also includes another attachment $c \in P' - \{s_1, t_2\}$ and the 2-connectivity assumption. Fig. 5 illustrates how to extract a desired pair of paths.

Fig. 5 Extraction of two paths in Step 10.

Step 11. Now we may assume that each bridge has either all the attachment on $P' \cup Q$ or all of them on $P \cup P'$. The set of bridges of the former (latter) type is denoted by $B^I(B^II)$.

If there exists an alternating pair of bridges in B^I , then return "yes". Otherwise go to the next step.

Step 12. For each bridge $B \in B^I$ having two or more attachments on each of P' and Q , call $f(G(B), P'[a;b], Q[c;d])$, where a and b (c and d) are the attachments on P' (Q) closest to s_1 and t_2 (s_2 and t_1), respectively, and return "yes" if $f = \text{"yes"}$. If $f = \text{"no"}$ for every such bridge or if no such bridge exists, go to the next step.

Step 13. Let x and y be attachments of bridges of B^I closest to s_1 and t_2 , on P' (It is necessary that s_1, t_2, x, y are distinct). If there exists an alternating pair of bridges $B_i, B_j \in B^II$ such that B_i has an attachment $u \in P'[x; t_2] - \{x, t_2\}$ and B_j has an attachment $v \in P'[s_1; y] - \{s_1, y\}$, then return "yes". Otherwise go to the next step. The extraction of desired paths is illustrated in Fig. 6. It should be noted here that some bridges of B^I include two disjoint paths $M(x, s_2)$ and $N(y, t_1)$.

Step 14. Let x and y be as in Step 13. For each bridge $B \in B^II$ having an attachment $u \in P'[x; t_2] - \{x, t_2\}$ and another attachment $v \in P'[s_1; y] - \{s_1, y\}$, perform the following operations, or if no such bridge exists, return "no".

If B has an attachment in $P'[s_1; x] - \{s_1, x\}$, let $s_1^* \leftarrow s_1$ and $x^* \leftarrow x$. Otherwise let $s_1^*(x^*)$ be the attachment on $P(P')$ closest to $s_1(x)$ and update bridge B by discarding subpaths $P[s_1; s_1^*]$ and $P'[s_1; x^*]$ except $\{s_1^*, x^*\}$. By means of the same argument for $P'[y; t_2]$, we specify vertices t_2^* and y^* corresponding to s_1^* and x^* , respectively.

Consider the section graph G' of G determined by the vertices of $P[s_1^*; t_2^*]$, $P[x^*; y^*]$ and B (see Fig. 7). Call $f(G', P[s_1^*; t_2^*], P[x^*; y^*])$ and return "yes" if $f = \text{"yes"}$. If $f = \text{"no"}$ for every such bridge, then return "no".

Fig. 6 Extraction of two paths in Step 13.

Fig. 7. Section graph G' .

3. Complexity Analysis

We have not explicitly discussed computer implementation of the algorithm, which is a long and tedious description. However its complexity can be easily evaluated unless we insist on very strict analysis.

Henceforce linear means $O(|E|+|V|)$. The algorithm consists of operations in the main routine (Steps 1 - 6), which are performed once and those in the recursive procedure f (Steps 7 - 14). First we show that all the steps are performed in linear time if we neglect the statements of recursive calls.

Step 1 is performed by applying flow techniques [5]. It is linear since only two augmentation paths are required. Step 2 is performed by means of an obvious modification of the well-known graph search technique, which is linear. Step 3 is also done by means of a modification of the graph search technique. Its straightforward implementation requires $O(|V| \cdot |E|)$ time. To guarantee linearity of Step 3, we need a careful implementation so that each edge is not traversed no more than some constant time in the graph search. The implementation and its linearity proof are complicated and are not given here. Step 4 requires trivial operations only and obviously is linear.

The term "bridge" is the same as that appears in the planarity testing [6],[7], except that it is defined here with respect to some paths instead of a cycle. Thus the operation in Step 5 to find a pair of alternating bridges can be viewed as a part of the linear planarity test algorithm [8].

Step 6 is obviously linear except for the recursive call of f , and the linear 2-connectivity test algorithm [9] can be used. To find a path $P'(s_1, t_2)$ in Step 7 can also be done in linear time by using the graph search. We do not give the complexity analysis of Steps 8 - 14, since the operations therein are obvious modifications of those in Steps 1 - 6.

What remains to be shown is that the depth of recursive calls is bounded by $O(|V|)$, which yields the conclusion that the whole algorithm terminates in $O(|V| \cdot |E|)$ time. We only consider the edges or vertices of graph $G = (V, E)$ at the first call of f , since the number of operations related to the other edges is bounded by some constant. If an edge or a vertex of G belongs to an subgraph $G' = (V', E')$ of G which is extracted in Step 12 or Step 14, then it is clear that $|V'| < |V|$. Hence the number of such subgraphs including a specific edge or vertex is of $O(|V|)$,

and so is the depth of recursive calls, which implies that the presented algorithm runs in $O(|V| \cdot |E|)$ time.

4. An Application to Wire Routing

The TPP is related to "wire routing", which is an important portion of the layout design of LSI's, printed circuit boards (PCB's) or hybrid IC's. In computer aided wire routing design, the constraints on the way in which the connections of signal nets should be carried out are very often modeled by an underlying graph, usually a grid, such that the only routes allowed are those using its edges. Graph-theoretically, the routing is equivalent to or even harder than the Disjoint Connection Path (Discrete Multi-Commodity) Problem, which has been shown to be NP-complete [1].

The almost all existing layout design systems [e.g., 10] are based on the following strategy. First we try to connect as many signal nets as possible by using an automatic router, which very rarely gives a complete (100 %) connection. The so-called "maze-type" algorithms [11] are often used in automatic routers, in which the signal nets (or pin pairs) are connected one by one by means of a graph search technique until no more signal net can be connected without intersecting already-routed ones. Next the partial design is improved by means of manual interaction through graphic displays or digital plotters, where the designers' experience and intuition play an essential role. The complete connection, if succeeded, goes through to a digitizer and a design rule check program. It goes without saying that such a design methodology costs much design turnaround time and can involve errors due to manual operations.

Recently a new interactive design system for PCB, which provides us with error-free wiring pattern, has been proposed [12]. The design turnaround time of wire routing can be remarkably decreased if the TPP algorithm is implemented on such a design system. Suppose we have connected k nets $\{N_1, N_2, \dots, N_k\}$ using a maze-type algorithm and $(k+1)$ th net N_{k+1} cannot be connected. This implies the existence of "boundary" which separates a pin (terminal) s from another pin t of net N_{k+1} . Graph-theoretically, "boundary" is a minimal set S of vertices, called minimal s, t separator, such that every path between s and t includes at least one vertex in S . With an obvious modification of a maze-type algorithm, we can list up the vertices in the boundary S [12]. Now it

is clear that we must change the route of at least one net in $\{N_1, N_2, \dots, N_k\}$ in order to obtain a complete connection, if it exists. The TPP algorithm is used to check whether $(k+1)$ nets $\{N_1, N_2, \dots, N_{k+1}\}$ can be connected by rerouting only one of $\{N_1, N_2, \dots, N_k\}$. Namely, the TPP algorithm with respect to the pair (N_i, N_{k+1}) is applied for each $N_i \in \{N_1, N_2, \dots, N_k\}$, where all the vertices included in the $(n-1)$ nets $\{N_1, \dots, N_{i-1}, N_{i+1}, \dots, N_k\}$ are prohibited to be used.

In the routing design of linear bipolar IC's and thic film hybrid IC's, usually a single layer is provided for wire connection. So the linear TPP algorithm for planar graphs [3] can be used. On the other hand, we should use the TPP algorithm for a general graph when LSI's or PCB's with multi-layer technology are concerned.

Example. Fig. 8(a) shows a graph (single-layer grid) representing the constraints for connecting nets $N_1, N_2, \dots, N_6$. This example is far simpler than those appear in practise, but suitable to demonstrate how the TPP algorithm is applied. The number, say i , in a circle indicates a pin of net N_i . A typical automatic router connects nets $N_1 \sim N_5$ as indicated by the solid lines. It is observed that net N_6 cannot be connected unless we reroute some of $N_1 \sim N_5$. Now the graph search algorithm for connecting N_6 extracts a boundary consisting of six vertices indicated by $\times$ in Fig. 8(b), where the vertices corresponding to pins are neglected. By removing the route for N_5 (removing N_4 fails) and applying the TPP algorithm with respect to net pair (N_5, N_6) on the subgraph shown in Fig. 8(c), we discover the connection as indicated by

Fig. 8. Example of applying TPP algorithm.

the solid lines. Note that the vertices on the connection routes for $N_1 \sim N_4$ and their incident edges are removed when considering the pair (N_5, N_6) . The complete connection for $N_1 \sim N_6$ is shown in Fig. 8(d).

Acknowledgment. This work was supported by the Grant in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan under Grant: Cooperative Research (A) 435013 (1980).

References

- [1] Karp, R.M., "On the complexity of combinatorial problems", Networks 5, pp. 45-68 (1975).
- [2] Lynch, J.F., "The equivalence of theorem proving and the interconnection problem", ACM SIGDA Newsletter 5:3 (1975).
- [3] Perl, Y. and Shiloach, Y., "Finding two disjoint paths between two pairs of vertices in a graph", J. Assoc. Comput. Mach. 25, pp. 1-9 (1978).
- [4] Shiloach, Y., "A polynomial solution to the undirected two paths problem", ibid. 27, pp. 445-56 (1980).
- [5] Even, S. and Tarjan, R.E., "Network flow and testing graph connectivity", SIAM J. Comput. 4, pp. 507-18 (1975).
- [6] Auslander, L. and Parter, S.V., "On imbedding graphs in the sphere", J. Math. Mech. 10, pp. 517-23 (1961).
- [7] Fisher, G.J. and Wing, O., "Computer recognition and extraction of planar graphs from the incidence matrix", IEEE Trans. CT-13, pp. 154-63 (1966).
- [8] Hopcroft, J.E. and Tarjan, R.E., "Efficient planarity testing", J. ACM 21, pp. 549-68 (1974).
- [9] Tarjan, R.E., "Depth-first search and linear graph algorithms", SIAM J. Comput. 1, pp. 146-60 (1972).
- [10] Johnson, D.R., "PC board layout techniques", Proc. 16th Design Automation Conf., pp. 337-43 (1979).
- [11] Lee, C.Y., "An algorithm for path connections and its applications", IRE Trans. EC-10, pp. 346-65 (1961).

- [12] Mori, H., et. al., "BRAIN: An advanced interactive layout design system for printed wiring boards", to be presented at the IEEE Internat. Conf. on Circuits and Computers (1980).