Discussiones Mathematicae
Graph Theory 15(1995) 147–166

A LINEAR ALGORITHM FOR THE TWO PATHS PROBLEM ON PERMUTATION GRAPHS

C.P. Gopalakrishnan and C. Pandu Rangan

Department of Computer Science
Indian Institute of Technology
Madras 600 036, India
e-mail: rangan@iitm.ernet.in

Abstract

The *'two paths problem'* is stated as follows. Given an undirected graph $G = (V, E)$ and vertices $s_1, t_1; s_2, t_2$, the problem is to determine whether or not G admits two vertex-disjoint paths P_1 and P_2 connecting s_1 with t_1 and s_2 with t_2 respectively. In this paper we give a linear ($O(\mid V \mid + \mid E \mid)$) algorithm to solve the above problem on a *permutation graph.*

Keywords: algorithm, bridge, connectivity, disjoint paths, permutation graph, two paths problem.

1991 Mathematics Subject Classification: 05C38, 05C85.

1. Introduction

Given an undirected graph G and distinct vertices s_1, t_1, s_2, and t_2, the *'two paths problem'* (abbreviated as TPP) is to determine whether there exists two vertex-disjoint paths connecting s_1 with t_1 and s_2 with t_2 and to find such paths if they exist.

For the TPP on general undirected graphs, Shiloach [S 80] and Ohtsuki [O 80] gave an $O(\mid V \mid \star \mid E \mid)$ algorithm independently. Their methods were totally different. In [S 80] a complicated reduction process is used to obtain a solution of the problem for an arbitrary graph from the solutions of its tri-connected components. In [O 80] the concept of bridges and a recursive

strategy is used to solve the problem on a biconnected graph. For the TPP on chordal graphs [PS 78] and [KPS 91] give an $O(\mid V \mid + \mid E \mid)$ algorithm by different methods. For planar graphs the problem can be solved in linear time ([RP]).

In this paper we present a linear algorithm for the TPP on permutation graphs. Our linear algorithm takes its spirit from the work of [O 80] and uses certain vital properties of bridges of paths in permutation graphs.

2. PRELIMINARIES

A *permutation graph* is a graph for which there is a labelling $\{v_1, \ldots, v_n\}$ of the vertices and a permutation π of $\{1, \ldots, n\}$ for which $(i-j)(\pi(i) - \pi(j)) < 0$ if and only if (v_i, v_j) is an edge.

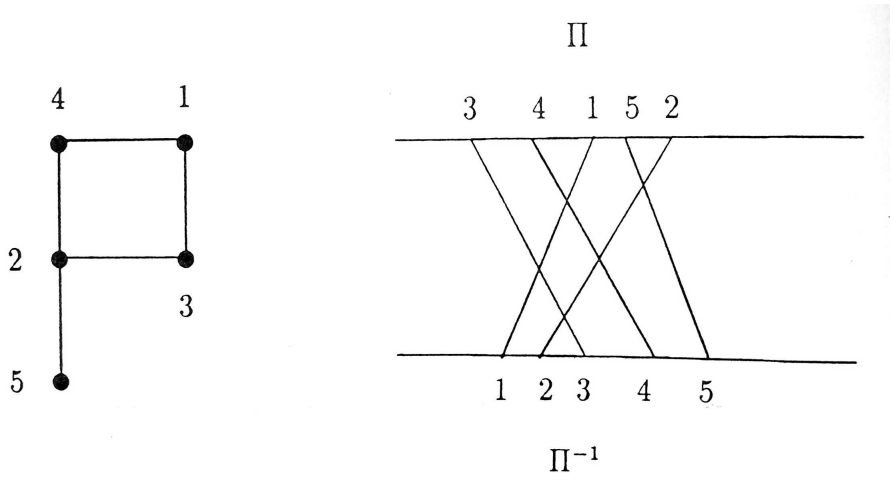

Figure 1. A permutation graph and its permutation diagram.

In this paper, we make use of the geometric representation of a permutation graph called the *permutation diagram*. Consider two parallel line segments A and B and mark off n points on each segment. Label them as $1 \ldots n$ in that order on each segment. Let π denote a permutation of $(1 \ldots n)$. Now draw n line segments, connecting point i in A to point $\pi^{-1}(i)$ in B, for $1 \leq i \leq n$. These n line segments will serve as the underlying intersection model for the input permutation graph G. The permutation graph represented is the one obtained by taking the line segments as vertices and the line crossings

as edges. That is, (i, j) is an edge iff the line segments $(i, \pi^{-1}(i))$ and $(j, \pi^{-1}(j))$ intersect. See Figure 1 for an example of a permutation graph and its corresponding permutation diagram. Given any permutation graph, Spinrad [S 83] shows how to construct a corresponding permutation diagram in $O(n^2)$ time. Applications of permutation graphs are discussed in [G 80].

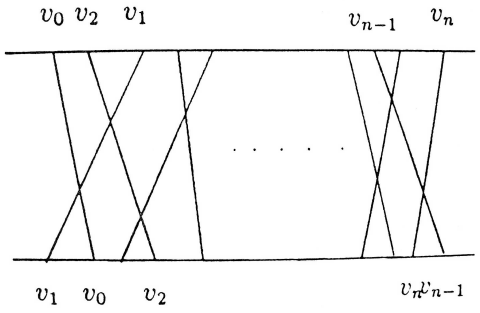

Figure 2. Chordless path in a permutation diagram.

Let $P = [v_0, v_1, \ldots, v_{k-1}, v_k]$ be a *path* of length k in a permutation graph. That is, $(v_i, v_{i+1}) \in E$, for all $0 \leq i < k$. We sometimes use the notation $P(v_0, v_k)$ to indicate the above path between the source vertex v_0 and sink v_k. The set of vertices of the path P is denoted by $V(P)$ and the set of edges constructing it is denoted by $E(P)$. For two vertices $v_i, v_j \in P$, the subpath of P between these two vertices is denoted by $P[v_i; v_j]$. An edge $\{v_i, v_j\} \in E - E(P)$ is called a *chord* of P. A *chordless* path has no chords. P is called *simple* if $v_i \neq v_j$ for $0 \leq i < j \leq k$. The operation '.' *concatenates* two paths, i.e., if $P = [v_0, \ldots, v_p]$ and $Q = [u_0, \ldots, u_q]$ are two paths and $v_p = u_0$, then $P.Q$ denotes the path $[v_0, \ldots, v_p = u_0, \ldots, u_q]$.

A path $P = [v_0, v_1, \ldots, v_k]$ is a *cycle* of length k if $k \geq 3$, $v_k = v_0$ and $P[v_0, v_{k-1}]$ is a simple path. A *chordless cycle* has no chords. We state the following lemma about chordless cycles in a permutation graph.

Lemma 2.1. *In a permutation graph, the length of a chordless cycle can be at most four.*

Consider the geometric representation of a simple chordless path $P = [v_0, v_1, \ldots, v_{k-1}, v_k]$ $(k > 3)$ in a permutation graph as shown in Figure 2 (we are not showing the other vertices of the graph). Barring isomorphisms, this is the only possible representation of the path. We call the vertices v_0 and v_1 as *outside-head* vertices and v_{k-1} and v_k as *outside-tail* vertices. All

the other vertices of the path are called as *inside* vertices. If the path consists of only four vertices or less, then they are all outside vertices. We observe that any vertex not belonging to $V(P)$ and adjacent to an inside vertex should also be adjacent to atleast one other vertex of the path. Whereas outside vertices need not satisfy this criterion i.e., some other vertex not belonging to the path can be adjacent to an outside vertex without being adjacent to any other vertex of the path. All these can be easily verified by inspecting Figure 2.

Definition 2.1. Let J be a fixed subgraph of G. Let $V(J)$ be the set of all vertices which belong to the subgraph J. Let $E(J)$ be the set of edges which constitute J. We define a *bridge* B of J as either of the following:

- a single edge $e = (x, y) \in E - E(J)$ and $x, y \in V(J)$. This is called as a *degenerate* bridge.
- a maximal subgraph of $G' = (V, E - E(J))$ with at least one vertex $x \in V - V(J)$ such that, for every other vertex y of B, there exists a path $R(x, y)$ without intersecting any vertex in V(J) except at y. This path is usually called a *cross-cut* from x to y and is denoted by $CC_B(x, y)$.

The vertices of B which also belong to J are called the *vertices of attachment* of B with respect to J.

In the following definitions let $G = (V, E)$ be a graph. Without loss of generality G is assumed to be bi-connected. Let i, j, k, l be four distinct vertices of G and $M(i, j)$ and $N(k, l)$ be disjoint (henceforth disjoint means vertex-disjoint) paths between i, j and k, l respectively.

Definition 2.2. (See Figure 5.) Let $J = M \cup N$. Let there exist two vertex-disjoint paths $P_1(a, d)$ and $P_2(b, c)$ without traversing $E(J)$, where a, b, c, d are distinct bridge attachments such that i, a, b, j are in this order on M and k, c, d, l are in this order on N. A pair of bridges B_1 and B_2 of J is called *alternating* if B_1 and B_2 include $P_1(a, d)$ and $P_2(b, c)$, respectively.

Definition 2.3. Let $J = M \cup N$. A bridge of J having two or more attachments on each of M and N is called an *eligible* bridge.

Definition 2.4. (See Figure 6.) Let $J = M \cup N$. For each eligible bridge B of J, let a and b (c and d) be its attachments on $M(N)$ closest to i and j (k and l), respectively. Then we call the subgraph of G determined

by the vertices and edges of $B \cup M[a;b] \cup N[c;d]$ as the *section graph* $G_J(B)$ or simply $G(B)$. a, b, c, d are called the *endpoints* of the section graph.

Definition 2.5. Let $J = M \cup N$. Let G have a single bridge with respect to J containing attachments i, j, k, l. A binary recursive function $f = (G, M, N)$ is defined as follows. f returns true (false) if two disjoint paths $M(i, l)$ and $N(j, k)$ exist (do not exist). We refer to M as the *top* path and N as the *bottom* path.

Definition 2.6. (See Figure 6.) Let P and Q be vertex-disjoint paths between s_1 and t_2 and s_2 and t_1, respectively. Let $J = P \cup Q$ and $\mathbf{B}$ be the set of bridges with respect to J. The set $\mathbf{B}$ can be partitioned into the following subsets:

$\mathbf{B}^P$: The set of bridges with every attachment on P.
$\mathbf{B}^Q$: The set of bridges with every attachment on Q.
$\mathbf{B}^{PQ}$: The set of bridges with attachments on both P and Q.

The next section begins with a outline of the $O(\mid V \mid\mid E \mid)$ time algorithm of [O 80] for undirected graphs. Then we present our linear algorithm for permutation graphs. Conclusions are offered in the last section.

3. The Two Disjoint Path Algorithm

Let $G_0 = (V_0, E_0)$ be a given general undirected graph. The two pairs of vertices between which the disjoint paths have to be found are s_1, t_1 and s_2, t_2. We assume that TPP is true if the two required vertex-disjoint paths exist, otherwise it is false. Without loss of generality, we can assume that G_0 is bi-connected. The algorithm is outlined below in a sequence of steps.

Step 1. Find two disjoint paths between $\{s_1, t_1\}$ and $\{s_2, t_2\}$. (This can be performed by applying flow techniques [ET 75]). If no two disjoint paths are found then TPP is clearly false. If two paths $P_1(s_1, t_1)$ and $P_2(s_2, t_2)$ are found then TPP is trivially true. Hence non-trivially, we assume that two paths $P(s_1, t_2)$ and $Q(s_2, t_1)$ are found and we proceed to the next step.

Step 2. Find all bridges $\mathbf{B} = (B_1, B_2, \ldots)$ with respect to $P \cup Q$.

Step 3. If there exist three vertices a, b, c on P in this order and two bridges $B_i \in \mathbf{B}^{PQ}$ and $B_j \in \mathbf{B}^P$ 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 Figure 4 and

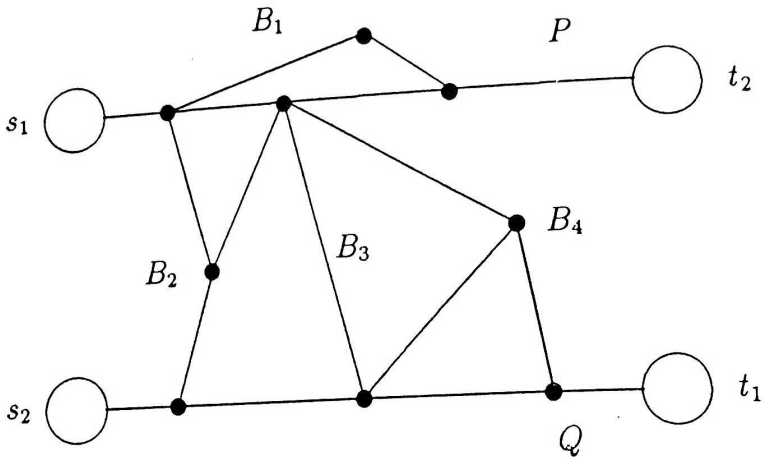

Figure 3. Set of bridges.

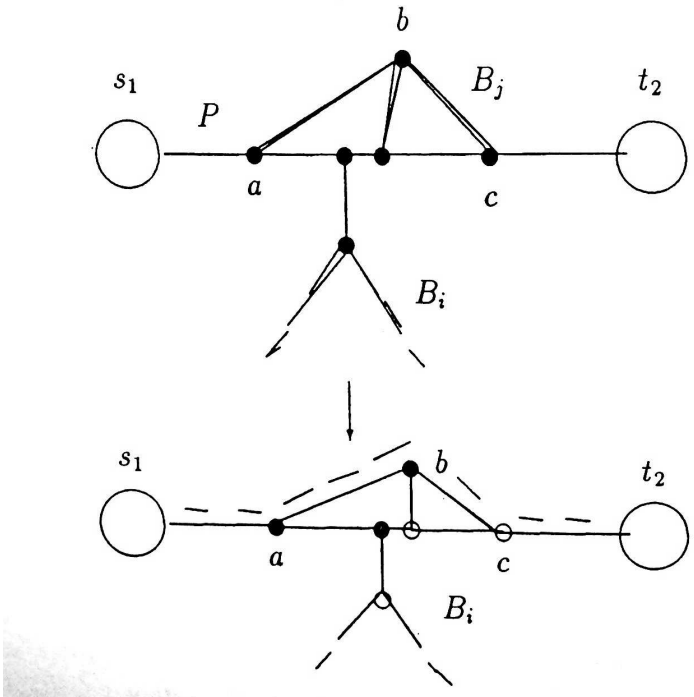

Figure 4. Elimination of $\mathbf{B}^P$. Note after elimination, the modified path P passes through b.

update the set $\mathbf{B}$ 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. Now make the paths $P(Q)$ chordless and update $\mathbf{B}^P(\mathbf{B}^Q)$. (This step involves finding the ambitus and for a linear time algorithm and its efficient implementation see [MT 89]).

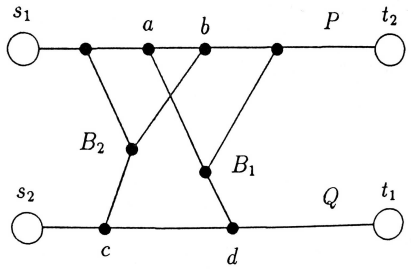

Figure 5. Alternating pair of bridges.

Step 4. Discard all the bridges in $\mathbf{B}^P$ and $\mathbf{B}^Q$. Henceforth we may assume that all bridges belong to $\mathbf{B}^{PQ}$. If there exists an alternating pair of bridges, then TPP is true. Otherwise go to the next step.

Step 5. If there is no eligible bridge in $\mathbf{B}$ then TPP is obviously false. Discard all non-eligible bridges. For each eligible bridge 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. Now call the binary recursive function $f(G_0(B), P[a;b], Q[c;d])$ and let $\mathbf{B} \leftarrow \mathbf{B} - \{B\}$. TPP is true if f returns true and false if f returns false for every $G_0(B)$.

The following steps (Steps f1–f8) are performed in $f(G', P[a;b], Q[c;d])$, where G' is a section graph and P, Q, a, b, c, d are as shown in Figure 6. Here P is the top path and Q is the bottom path and a, b, c, d are the endpoints of the section graph. Without loss of generality, G' is assumed to be bi-connected.

Step f1. Find a path $P'(a, b)$ which is disjoint with $P[a;b]$ and $Q[c;d]$. Existence of P' is clear from the definition of bridge. We refer to P' as the *complementary* path.

Step f2. Find all the bridges $\mathbf{B}' = \{B_1, B_2, \ldots\}$ with respect to $P \cup Q \cup P'$ by means of the same operation as in Step 3.

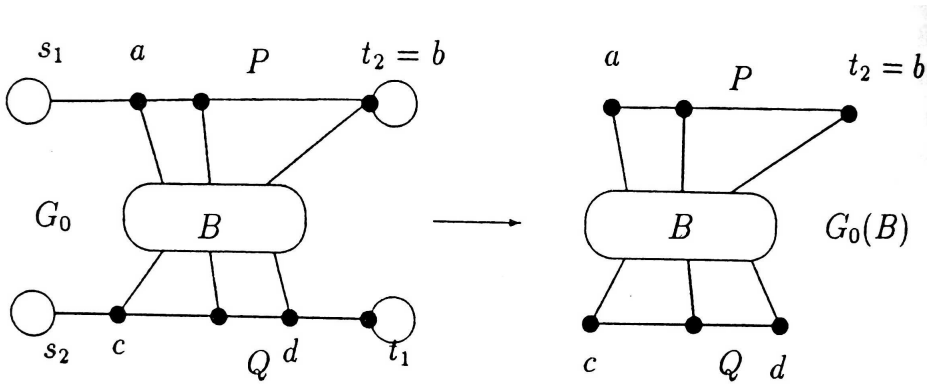

Figure 6. Section graph $G_0(B)$.

Step f3. Perform the same operations as in Step 3 with respect to P', and pertinently update the path P' and the set B'. Make P' chordless. Note that each bridge must have an attachment on $P' - \{a, b\}$ and another on $P \cup Q - \{a, b\}$.

Step f4. If there exists a bridge $B_i \in \mathbf{B}'$ having an attachment $l \in P - \{a, b\}$ and another attachment $m \in Q$, then return true, otherwise go to the next step. This can be proved by recalling that B_i also includes another attachment $n \in P' - \{a, b\}$ and the bi-connectivity assumption. Figure 7 illustrates how to extract a desired pair of paths.

Step f5. Now we may assume that each bridge $B \in \mathbf{B}'$ has either all the attachments on $P' \cup Q$ or all of them on $P \cup P'$. The set of bridges of the former (latter) type is denoted by $\mathbf{B}^I$ ($\mathbf{B}^{II}$).
If there exists an alternating pair of bridges in $\mathbf{B}^I$, then return true. Otherwise go to the next step.

Step f6. For each eligible bridge $B \in \mathbf{B}^I$ 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 a and b (c and d), respectively, and return true if f returns true. If f returns false for every eligible bridge or if no eligible bridge exists, go to the next step.

Step f7. Let x and y be attachments of bridges of $\mathbf{B}^I$ closest to a and b on P'. (It is necesary that a, b, x, y are distinct). If there exists an alternating pair of bridges $B_i, B_j \in \mathbf{B}^{II}$ such that B_i has an attachment $u \in P'[x; b] - \{x, b\}$ and B_j has an attachment $v \in P'[a; y] - \{a, y\}$, then return true. Otherwise go to the next step. The extraction of desired paths is illustrated

in Figure 8. It should be noted here that some bridges of $\mathbf{B}^I$ include two disjoint paths $M(x, c)$ and $N(y, d)$.

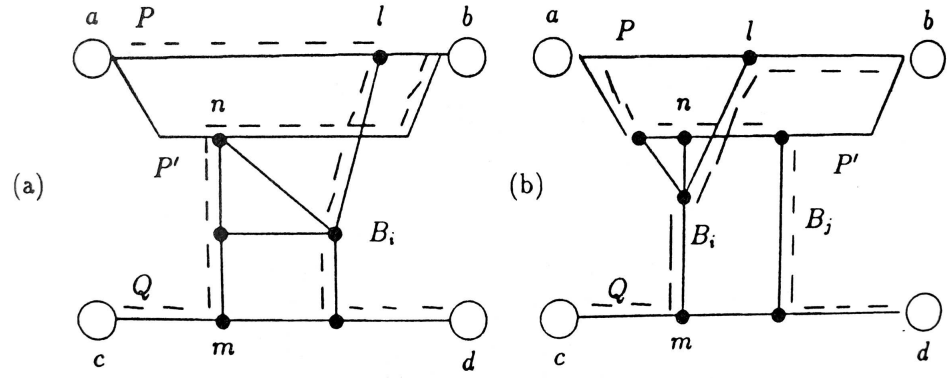

Figure 7. Extraction of two paths in Step 3. The dotted lines indicate the two paths.

Step f8. Let x and y be as in Step f7. For each bridge $B \in \mathbf{B}^{II}$ having an attachment $u \in P'[x; b] - \{x, b\}$ and another attachment $v \in P'[a; y] - \{a, y\}$, perform the following operations, or if no such bridge exists return false.

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

Consider the section graph G'' of G' determined by the vertices of $P[a^*, b^*], P'[x^*, y^*]$ and B. Call $f(G'', P[a^*, b^*], P'[x^*, y^*])$ and return true if f returns true. If f returns false for every such bridge then return false. The above algorithm consists of operations in the main routine (Steps 1–5), which are performed once and those in the recursion procedure f (Steps f1–f8). If we ignore recursion all steps are performed in $O(\mid V \mid + \mid E \mid)$ time. The depth of the recursive calls can be shown to be bounded by $O(\mid V \mid)$. 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 a subgraph

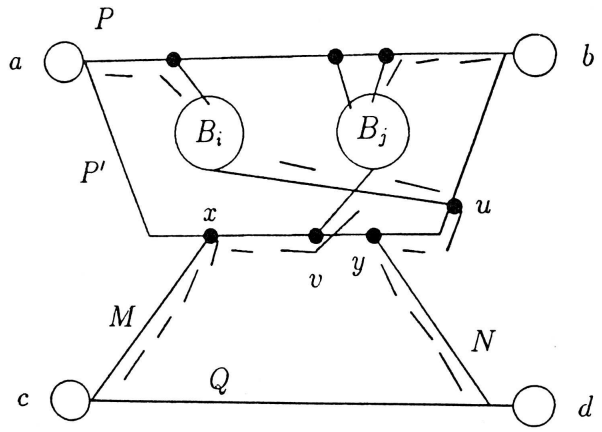

Figure 8. Extraction of two paths in Step 3. The dotted lines indicate the two paths.

$G'' = (V'', E'')$ of G' which is extracted in Step f6 or Step f8, 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 whole algorithm runs in $O(| V | | E |)$ time.

4. The Linear Algorithm for Permutation Graphs

In this section we show that when the input graph is restricted to the class of permutation graphs, we can bound the number of recursive calls to a constant and hence the above algorithm becomes linear. First we state some lemmas.

For all the following lemmas it is assumed that the input graph G_0 is a permutation graph and hence the section graphs are permutation graphs as well. In the general algorithm outlined in the previous section recursive calls are made on section graphs constructed in Steps f6 and f8. In order to arrive at a linear algorithm we have to eliminate recursive calls in these two steps.

Although we use the permuation diagram of the permutation graph extensively in the proofs to show the linear algorithm, we do not need the permutation diagram to be given as input i.e., we do not require to run Spinrad's transitive orientation algorithm initially as a preprocessing step. The input of the permutation graph in the form of an adjacency list is adequate, just like any other graph.

We define the following special cases of section graphs which admit a linear solution to the TPP. This will be proved later.

Definition 4.1. (See Figure 10). We call the section graphs which have their top and bottom paths of length at most two as *reduced cases*. Let a', b', c', d' be the endpoints of the section graph. In particular, we define the following:

Reduced Case 1 (RC1): Section graph with top and bottom paths of length one and a' is adjacent to c' and b' is adjacent to d'.
Reduced Case 2 (RC2): Section graph with top and bottom paths of length one and a' is not adjacent to c' and b' is not adjacent to d'.
Reduced Case 3 (RC3): Section graph with only one of the top or bottom path of length two and the other of length one.
Reduced Case 4 (RC4): Section graph with both top and bottom paths of length two.

Our strategy now is to show that all the recursive subproblems (these subproblems are of solving the TPP on the section graphs) generated during the general algorithm after a constant number of recursive calls fall under the reduced cases just defined which admit linear solutions. We give the following lemma.

Lemma 4.1. *Let P and P' be as defined in the recursive function f. Then the inside vertices of the path P' should be adjacent to at least one vertex belonging to $V(P)$.*

Proof. It follows directly from Lemma 2.1 and from the defintion of inside vertices and from the fact that P and P' are chordless. ∎

Lemma 4.2. *Consider Step f6 of the recursive function f. After at most two iterations the top path in the recursive call of this step has length at most two.*

Proof. Let P'' denote the path $P'[a; b]$ after one iteration, that is after one call of f in step f6. We have to show that after one iteration the length of P'' is at most two. If $P'[a; b]$ is of length one or two we are done, i.e., we do not need another iteration.
In the first call of f in Step f6, $P'[a; b]$ is the top path of a section graph of a bridge $B \in \mathbf{B}^I$. B, by defintion has all its attachments on $P' \cup Q$, where P' is as defined in Step f1. In the next iteration, it is clear that P'' is the path (the new P') which is found in Step f1. If P'' has length greater than

two (recall that P' etc. are chordless), by Lemma 4.1, there would be an inside vertex of P'' and this vertex will be adjacent to some vertex of P. This violates the property that the bridge B belongs to of $\mathbf{B}^I$. ■

Corollary 4.2.1. *After at most four iterations of the recursive function f, we have at Step f6 subproblems with the following property: all the section graphs have top and bottom paths of length atmost two (Reduced cases).*

Proof. After two iterations, by Lemma 4.2 we have reduced the length of the top path to be at most two. We now interchange the top and bottom paths for the next iteration and reduce the length of the bottom path as well. ■

Lemma 4.3. *In Step f8 of the recursive function f all the subproblems fall under the category of RC1.*

Proof. It is clear that Steps f7 and f8 of f are performed only if there is a negative result in Step f6, i.e., all the section graphs formed from the bridges belonging to $\mathbf{B}^I$ yield a negative result. In Step f7 presence of alternating bridges are checked in $\mathbf{B}^{II}$. Step f8 is performed if there are no such alternating bridges. In particular there are no alternating chords in $\mathbf{B}^{II}$. This means that we have to solve for TPP on section graphs which have their endpoints (a', b', c', d') as follows:

- The endpoints form a 4-cycle. See Figure 9.
- a', b' belongs to P and c', d' belongs to P'.
- Let x, y be as defined in Step f8. Then $c' \in P'[x; b] - \{x, b\}$, $b \in P'[a; y] - \{a, y\}$.

From Definition 4.1 it is clear this type of section graph falls under RC1. ■

4.1. TPP on Reduced Cases of Section Graphs

For the sake of uniformity, we assume that a section graph G_S is named as follows: the top and bottom paths are called P and Q respectively with endpoints a, b for P and c, d for Q. Below we give algorithms for solving TPP on each of the reduced cases of section graphs. TPP is true for a section graph if there exist two vertex-disjoint paths between a, d and b, c, respectively.

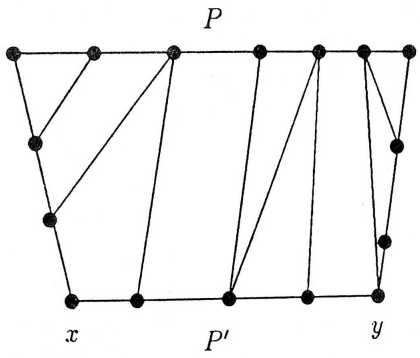

Figure 9. Non-alternating chords between P and P'. The section graphs formed from these fall under the Reduced Case 1 (RC1).

4.1.1. Reduced Case 1 (RC1)

Algorithm 4.1. begin
1) Find a shortest path M between a and d (not using the b and c vertices) and a shortest path N between b and c.
2) **If** (M or N is of length 1) **then return** TPP is true.
/* M and N is of length ≥ 2 */
3) Find which of M or N is of length 2.
(/* Without loss of generality let M be of length 2 and x be its intermediate vertex */)

3.1) **If** (x is a cut vertex between a and d) **then**
    3.1.1) **if** (there exists a path between b and c with this vertex excluded)
        then return TPP is true.
    3.1.2) **else**
        return TPP is false.
3.2) **else**
    3.2.1) **if** (N is of length 2)
        then return TPP is true.
    3.2.2) **else**
    begin
        3.2.2.1) Find bridges with respect to $P \cup Q \cup N$.
        3.2.2.2) **If** (any bridge has a and d as its vertices of attachment)

then return TPP is true.
3.2.2.3) **else**
return TPP is false.
end
end

For proving the correctness of the above algorithm we state the following lemma.

Lemma 4.4. *Let M and N be as defined in Algorithm* 4.1. *If the length of N is greater than* 2 *then the length of M should be at most* 2 *and vice versa.*

Proof. Let $N = [b, x, y, c]$, i.e., of length 3. By Lemma 2.1, N forms a 5-cycle with d as well as a. Hence, at least there should be a chord in both the 5-cycles. Since, N is chordless, the only possible chords are $\{x, d\}$ (say e_1) or $\{y, d\}$ (e_2) for one cycle and $\{x, a\}$ (e_3) and $\{y, a\}$ (e_4) for the other. But only e_1 and e_4 or e_2 and e_3 cannot exist as they give rise to a chordless 5-cycle. Hence, the presence of at least one of the two combinations ensures that a path of length 2 exists between a and d. This can be similarly shown for lengths of N greater than 3. Since M and N are complementary, the lemma holds the other way also. ■

Corollary 4.4.1. *Let the length of N be greater than* 2. *Then all inside vertices of N have chords connecting both a and d. If N is of length* 3, *at least one of the intermediate vertices will have the above property.*

Lemma 4.5. *Algorithm* 4.1 *correctly solves for TPP in RC1 in linear time* ($O(| V | + | E |)$).

Proof. In RC1, $[a, b, d, c, a]$ form a 4-cycle. To solve the TPP, we have to find two disjoint paths between a, d and c, b, respectively.

In Step 2, TPP is true since there exists a path between b and c. (Recall that the section graph is eligible.)

By Lemma 4.4, it is clear that at least one of M or N should have a length 2. Without loss of generality let it be M and let its intermediate vertex be x. In Step 3.1 we check whether x is a cut vertex for a and d, by checking whether there exists any other path between a and d excluding this vertex. Obviously if this vertex is also a cutvertex for b and c, then TPP is false, otherwise it is true.

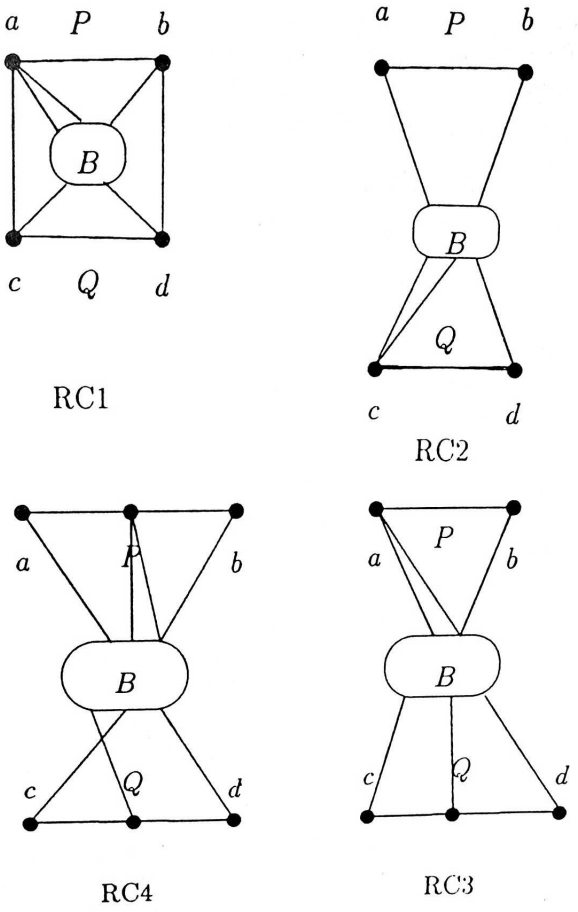

Figure 10. Reduced Cases of Section Graphs. In RC1 $[a, b, c, d, a]$ form a 4-cycle. RC2 is a general version of RC1. In RC3, only one of top or bottom paths is of length two. In RC4 both the top and bottom paths are of length one.

In Step 3.2 x is not a cut vertex and hence if the length of N is also 2 we have TPP to be true.

Step 3.2.2 handles the general case when the length of N is greater than 2. We find bridges with respect to $P \cup Q \cup N$. By Corollary 4.4.1 it follows that Step 3.2.2.2 correctly yields the solution.

We assume an adjacency list representation of S_G. Steps 1, 2, 3 are all trivial and can be implemented in a straightforward way in linear time. In Step 4, for finding the bridges, we just check for connected components in $G_S - P - Q - N$ and can be implemented in linear time. ∎

Thus we have solved the TPP for RC1. RC1 is a fundamental case to which the other Reduced cases will be reduced.

4.1.2. Reduced Case 2 (RC2)

The difference between RC1 and RC2 is, here there is no edge between both a, c and b, d. As in the previous case, our aim is to find two disjoint paths between a to d and b to c, respectively.

Algorithm 4.2. begin
1) Find two disjoint paths between $\{a, b\}$ and $\{b, c\}$.
/* (We assume that two chordless paths $M(a, c)$ and $N(b, d)$ are found, otherwise TPP turns out to be either true or false.) Obviously M and N are not both of length 1, otherwise it boils down to RC1. Hence at least one of them should be of length greater than 1. Or in other words $M(a, c).[c, d].N(b, d).[a, b]$ form a cycle C of length at least 5. */
2) Find bridges with respect to $P \cup Q \cup M \cup N$.
If (there exists alternating bridges between M and N) **then return** TPP is true.
3) Find the set of chords of cycle C. /* These set of chords of C divides the cycle into 4-cycles and 3-cycles. */ Solve for TPP for each eligible section graph which has endpoints in a 4-cycle. TPP is true if at least one of them returns true.
end

Lemma 4.6. *Algorithm* 4.2. *correctly solves for TPP in RC2 in linear time.*

Proof. Let us assume that alternating bridges do not exist between M and N. In particular no alternating chords exist. Hence the set of chords of C divides the cycle into 4-cycles and 3- cycles which are subcyles within C. Now obviously, TPP is true only if its true for atleast one eligible section graph which has its endpoints in one 4-cycle. But this is nothing but a RC1 problem which can be solved in linear time. Subgraphs which are within 3-cycles need not be considered as they do not give rise to an eligible section graph.

Step 1 can be peformed in linear time in the same way as Step 3 of the general algorithm by applying flow techniques [ET 75]. In Step 2, checking for alternating bridges can also be done in linear time as in Step 3 of the general algorithm. In [O 80] this step is viewed as part of the linear planarity test algorithm of [HT 74]. The time taken for Step 3 is also linear because time taken for a RC1 problem is linear and an edge of the original graph is considered at most once, by the obvious property of bridges. Hence the whole algorithm is linear. ∎

4.1.3. Reduced Case 3 (RC3)

In RC3, one of the top or bottom paths is of length two and the other is of length one. We assume that P is of length 2 and Q is of length one.

Lemma 4.7. *Consider a RC3 subproblem. Let the length of P be two and that of Q be one. Then a shortest path from a to b is of length* 2.

Proof. Follows from the fact that RC3 is a subproblem generated by the recursive function f at Step f6 and from Lemma 4.2. ∎

Lemma 4.8. *Consider a RC3 subproblem. Let the length of P be two and that of Q be one. Let the set $\mathbf{P} = \{P_1, P_2, \ldots\}$ represent the set of all vertex-disjoint shortest paths from a to b. Then $\mathbf{P}$ is the set of only possible chordless paths and the length of each P_i is two.*

Proof. Follows from the previous lemma. ∎

Lemma 4.9. *Consider a RC3 subproblem. Let $P, Q, \mathbf{P}$ be as defined above. Let $\mathbf{B}$ be the set of bridges with respect to $\mathbf{P} \cup Q$. Then there exist no bridge in $\mathbf{B}$ with attachments to both a and b.*

Proof. Follows from the previous lemma. ∎

Lemma 4.10. *Consider a RC3 subproblem. Let $\mathbf{P}$ be as defined above. Since the length of each path is of length two, let the intermediate vertex of a path P_i be p_i. Let $\mathbf{B}$ be the set of bridges with respect to $\mathbf{P} \cup Q$. If there exists two bridges such that one has an attachment on p_i and c and the other on p_j and d, then TPP is true.*

Proof. Easy. ∎

Lemma 4.11. *Let $\mathbf{P}$, $\mathbf{B}$ be as defined previously. Also let the intermediate vertex of a path P_i be p_i. Let there exist a bridge with attachments on c,*

d, p_i and p_j, two different intermediate vertices. Then TPP is true if there exists two vertex-disjoint paths in this bridge from $\{p_i, p_j\}$ and $\{c, d\}$.

Proof. Easy. ∎

Lemma 4.12. *Let* **P**, **B** *be as defined previously. Let there exist a bridge with attachments on c, d and two attachments on the same path P_i (It has to either a and p_i or p_i and b). Then TPP is true if its true on the section graph formed from this bridge.*

Proof. Easy. ∎

Remark 4.1. *By the lemma just stated, we have succeeded in reducing the RC3 problem to a RC2 problem, because the length of both the top and bottom paths is one.*

Algorithm 4.3. begin
1) Find the set of all vertex-disjoint shortest paths from a to b. Let it be $\mathbf{P} = \{P_1, P_2, \ldots\}$. Let the intermediate vertices of a path P_i in **P** be p_i.
2) Find the set of bridges with respect to $\mathbf{P} \cup Q$. Let it be **B**.
3) **If** there exists two bridges in **B** such that one has an attachment on p_i and c and the other on p_j and d, where p_i and p_j are two different intermediate vertices, **then return** TPP is true.
4) Find all bridges in **B** with attachments on c, d, and p_i, p_j, two different intermediate vertices. **If** there exists two vertex-disjoint paths in such a bridge from $\{p_i, p_j\}$ and $\{c, d\}$ **then return** TPP is true.
5) Find all bridges with attachments on c, d and two attachments on the same path P_i. **If** TPP is true on the section graph formed from any one such bridge **then return** TPP is true.
end

Lemma 4.13. *Algorithm* 4.3 *correctly solves for TPP in RC3 in linear time.*

Proof. The correctness follows from the lemmas stated before the algorithm.

The algorithm like the algorithms for RC1 and RC3 can be easily implemented in linear time. Step 1 can be solved easily in linear time because the length of each shortest path is only two. Step 3 can be implemented in a way similar to that of finding alternating bridges. Step 4 is easy. In Step 5, the problem is reduced to that of a RC2 and the linearity of this step is guaranteed because each edge is present in only one of the subproblems. ∎

4.1.4. Reduced Case 4(RC4)

As stated in Remark 4.1, we reduced the length of the top path to one and hence an RC2 subproblem resulted from an RC3. In a RC4 problem where the length of both P and Q is two, after reducing the top path to one, we interchange top and bottom paths to get a RC3 subproblem.

Thus we have succeeded in solving the TPP on the Reduced cases by reducing one problem to a smaller one. That is RC4 is reduced to RC3 which in turn was reduced to RC2, which was in turn reduced to RC1.

5. Conclusion

General graphs require $O(\mid V \mid \star \mid E \mid)$ time for the TPP. Chordal graphs and planar graphs admit linear time solutions. We have shown now that permutation graphs too admit a linear time solution to the TPP. The main difficulty in deriving a linear time algorithm comes with handling different levels of recursion. We have successfully eliminated the recursion by referring to the special properties of permutation graphs. Thus by exploiting some structural properties of a certain restricted class of graphs we have obtained efficient algorithms tailor-made for that class.

It would be interesting to study whether comparability or co-comparability graphs — both supersets of permutation graphs, have linear algorithms for TPP. We feel that this general approach can be used for solving this problem on these classes of graphs also.

References

[BM 76] J.A. Bondy, U.S.R. Murthy, Graph Theory with Applications (Academic Press, 1976).

[ET 75] S. Even, R.E. Tarjan, *Network flow and testing graph connectivity*, SIAM J. Comput. **4** (1975) 507–518.

[HT 74] J.E. Hopcroft, R.E. Tarjan, *Efficient planarity testing*, J. ACM **21** (1974) 549–568.

[G 80] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs (Academic Press, 1980).

[MT 89] B. Mishra, R.E. Tarjan, A linear time algorithm for finding an ambitus (Technical Report 464, August 1989, New York University).

[O 80] T. Ohtsuki, *The two disjoint path problem and wire routing design*, In: Proc. of the 17th Symp. of Res. Inst. of Electrical Comm. (1980) 257–267.

[PS 78] Y. Perl, Y. Shiloach, *Finding two disjoint paths between two pairs of vertices in a graph*, J. of the ACM **25** (1978) 1–9.

[RP] P.B. Ramprasad, C. Pandu Rangan, A new linear time algorithm for the two path problem on planar graphs (Technical Report, Department of Computer Science, IIT, Madras, 1991).

[S 80] Y. Shiloach, *A polynomial solution to the undirected two paths problem*, J. of the ACM **27** (1980) 445–456.

[S 83] J. Spinrad, *Transitive orientation in $O(n^2)$ time*, In: Proc. of Fifteenth ACM Symposium on the Theory of Computing (1983) 457–466.

[KPS 91] S.V. Krishnan, C. Pandu Rangan, S. Seshadri, A. Schwill, Two Disjoint Paths in Chordal graphs (Technical Report, 2/91, February 1991, University of Oldenburg, Germany).

Received 9 May 1994