Discussiones Mathematicae – Graph Theory 15 (1995) 59–72

EDGE–DISJOINT PATHS IN 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

In this paper we consider the following problem. 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 edge–disjoint paths P_1 and P_2 connecting s_1 with t_1 and s_2 with t_2, respectively. We give a linear $(O(|V| + |E|))$ algorithm to solve this problem on a *permutation* graph.

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

1991 Mathematics Subject Classification: 058C85, 05C38

1. Introduction

One essential element in parallel computation models, which extremely influences the execution speed of the entire system, is information exchange between individual processors. The standard solution for performing data transmissions between processors is to use a sparse communication network, in which each processor is connected by a bidirectional line to few other processors. One basic method for performing communications between processors is *open–line communication.* Open–line communication is a suitable method when data of variable length have to be transferred in bidirectional way. If a pair of processors wants to communicate, the network satisfies the request by establishing a path between this pair. The nodes/lines on such a path are exclusively assigned for this purpose and will not be released until the communication finishes. If two pairs of processors wish to communicate simultaneously, the network reserves two paths that are *node-* or *line-* disjoint in order to ensure that messages between these pairs do not interfere. If we model processor networks by undirected graphs, in which vertices and edges represent processors and bidirectional communication lines, respectively we get the following problem:

> Given an undirected graph $G = (V, E)$ and four distinct vertices $s_1, t_2; s_2, t_2$ find two vertex/edge–disjoint paths P_1 and P_2 connecting s_1 with t_1 and s_2 with t_2, respectively.

The vertex–disjoint version of this problem i.e., finding two vertex–disjoint paths between two given pairs of vertices has known algorithms. For example, Shiloach [S 80] and Ohtsuki [O 80] gave an $O(|V| * |E|)$ algorithm for solving this problem on general undirected graphs. The complexity of the problem becomes linear on special classes of graphs like chordal ([KPS 91]), planar ([RP]) and circular–arc graphs ([SP 91]). Recently, we have shown that this problem admits a linear solution on permutation graphs also ([GP]).

As far as the edge–disjoint version is concerned, LaPaugh and Rivest ([LR 78]) give a polynomial reduction to get edge–disjoint paths using vertex–disjoint algorithms after suitably modifying the input graph.

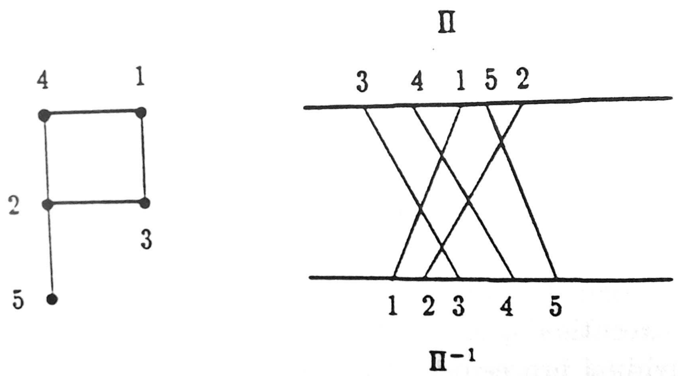


Figure 1. A permutation graph and its permutation diagram

More specifically, this transformation increases the size of an input graph $G = (V, E)$ as follows: the number of vertices become squared and the number of edges become $|V|^3$ in the worst case. Hence, even when a linear vertex–disjoint algorithm is applied the complexity becomes cubic in the number of vertices. Also the following results are know. Cypher ([C 80]) has proposed polynomial algorithms that determine n edge–disjoint paths

$(n \leq 5)$ connecting n pairs of vertices in $(n+2)$–connected graphs. Frank ([F 85]) developed an $O(|V|^3 * log(|V|))$ algorithm to compute n edge–disjoint paths between n pairs of vertices if the graph is planar and the $2n$ vertices lie on the outer face of a planar embedding of the graph and all vertices not on the outer face have even degree. [KPS 91] gives a simple linear algorithm using greedy techniques of [S 90] to find edge–disjoint paths between two pairs of vertices in chordal graphs.

2. Preliminaries

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

A permutation graph has a geometric representation called the *permutation diagram.* Consider two parallel line segments A and B and mark off n points on each segment. Label them as $1...n$ in that order on each segment. Let π denote a permutation of $(1...n)$. Now draw n line segments, connecting point i in A to point $\pi^{-1}(i)$ in B, for $1 < i < n$. 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].

Let $P = [v_0, v_1, ..., v_{n-1}, v_n]$ be a *path* of lenght n in a permutation graph. That is, $(v_i, v_{i+1}) \in E$, for all $0 \leq i < n$. We sometimes use the notation $P(v_0, v_n)$ to indicate the above path between the source vertex v_0 and sink v_n. 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 $0 \leq i < j \leq n$. The operation '.' *concatenates* two paths, i.e., if $P = [v_0, ..., v_n]$ and $Q = [u_0, ..., u_m]$ are two paths and $v_n = u_0$, then $P.Q$ denotes the path $[v_0, ...v_n = u_0, ..., u_m]$.

A path $P = [v_0, v_1, ..., v_n]$ is a *cycle* of length n if $n \geq 3, v_n = v_0$ and $P = [v_0; v_{n-1}]$ 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.*

Let $\delta_G(x,y)$ be the *distance,* i.e., the length of a minimal path (if any), between two vertices $x,y \in V$. An *edge separator* for a pair of vertices $x,y \in V$ is a set of edges $S \subseteq E$, such that $\delta_G(x,y) < \delta_{G-S}(x,y) = \infty$. S is a *minimal* edge separator if no proper subset of S is an edge separator for x and y.

In this paper, we use the concept of *bridges* to design our algorithm. Like the depth first search tree, the bridges offer clean solutions to a variety of graph problems. The graph viewed as a suitable cycle or a path together with a collection of its bridges offers a good insight into its structure. We define a bridge formally as follows. Here we use the definition as in [O 80]. Other equivalent definitions may be found in [BM 76], [KPS 91].

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 $E(J)$ by 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.

3. The Edge–Disjoint Path Algorithm

Let $G = (V,E)$ be a given undirected permutation graph. Let s_1, t_1 and s_2, t_2 be the two pairs of vertices (all distinct) between which two edge–disjoint paths have to be found. We assume that the Edge–Disjoint Path problem (henceforth abbreviated as EDP) is 'true' if the two required edge–disjoint paths exist, otherwise it is 'false'. We first state some definitions and lemmas.

Let P be a shortest path between s_1 and t_1 in G. Obviously P is *chordless.* Let $\boldsymbol{B} = \{B_1, B_2, ...\}$ be the set of bridges with respect to P. It is obvious that all bridges are non–degenerate.

We state a few definitions regarding the structure of the bridgess in $\boldsymbol{B}$.

Definition 3.1. The endpoints of a bridge $B \in \boldsymbol{B}$ are its extreme vertices of attachment on P. See Figure 2. Let the endpoints be x and y for a bridge B. Then $P[x;y]$ is called as the *width* of B. We sometimes use the same term to denote the *length* of the subpath $P[x;y]$. The width of a bridge is zero, if it has only one vertex of attachment on P. Starting from x we can number the vertices of attachment on P consecutively till y. For a bridge B, two vertices of attachment are called *consecutive* if they are numbered consecutively.

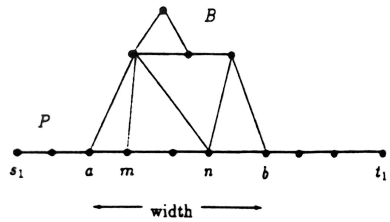
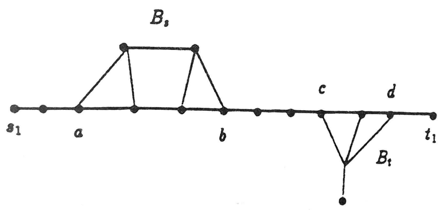
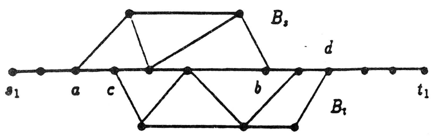


Figure 2. Structure of a bridge B. a and b are the endpoints of the bridge. m and n are consecutive vertices.



(a) Non-overlapping bridges



(b) Overlapping bridges

Figure 3. Non–overlapping and Overlapping bridges. In overlapping bridges $P[c;b]$ is the interval of overlap. Notice how the vertices of attachment of both the bridges alternate in the interval of overlap.

Definition 3.2. Two distinct bridges B_1 and B_2 are said to be *mutually traversable* if they share at least one common vertex of attachment. If B_1 and B_2 do not share any common vertex of attachment then they can be further classified as *overlapping* or *non–overlapping* depending on whether their widths overlap or not. For overlaping bridges we define *width of overlap* as the subpath common to both widths. We also use the same term to denote the length of such a subpath. See Figure 3.

For all the lemmas that follow we assume the following:

- P is a shortest path between s_1 and t_1, the two distinguished vertices.
- $\boldsymbol{B}= \{B_1, B_2, ...\}$ are the set of bridges with respect to P.
- Two distinguished bridges in $\boldsymbol{B}$, B_s and B_t contain s_2 and t_2, respectively.
- The endpoints of B_s are a and b and those of B_t are c and d.

Lemma 3.1. *If $B_s = B_t$ (i.e., B_s and B_t refer to the same bridge) then EDP is true.*

Proof. Easy. ■

Lemma 3.2. *Let B_s and B_t be mutually traversable. Then EDP is true.*

Proof. This implies that B_s and B_t share a common vertex of attachment. Hence it is obvious that a path from s_2 to t_2 can be constructed without using any vertices of P. ■

Lemma 3.3. *Let a path exists between s_2 and t_2 in $G-E(P)$. Then EDP is true.* ■

Proof. This path and P are the two required edge–disjoint paths. ■

Lemma 3.4. *Let B_s and B_t be non–overlapping. Let $E(P)$ be an edge separator for s_2 and t_2. Then EDP is false if and only if there exists no path between s_1 and t_1 in $G - E(P[b; c])$.*

Proof. Since B_s and B_t are non–overlapping, without loss of generality let the order of occurence of the endpoints be a, b, c, d in P. Refer 3(a).

First we prove the 'only if' part. Let there exist a path, say Q between s_1 and t_1 in $G - E(P[b; c])$. Then $CC_{B_s}(s_2, b).P[b; c].CC_{B_t}(c, t_2)$ and Q are the required edge–disjoint paths and hence EDP is true.

The 'if part'. Let there exists no path between s_1 and t_1 in $G - E(P[b; c])$. This also implies that there can exist no path between s_2 and t_2 in $G - E(P[b; c])$ for the following reason. If possible let there exist a path

S between s_2 and t_2 in $G-E(P[b;c])$. Let p and q be the vertices of S on P which are nearest to s_1 and t_1, respectively. Then $P[s_1;p].S[p;q].[q;s_2]$ is a path between s_1 and t_1 in $G-E(P[b;c])$, a contradiction.

It is now clear that $E(P[b;c])$ is an edge–separator for both s_1, t_1 and s_2, t_2. Let $T \subseteq E(P[b;c])$ be a minimal edge separator for s_2 and t_2. If $|T|=1$ then EPD is false, because any pair of paths connecting s_1, t_1 and s_2, t_2 respectively has to pass through $e \in T$. We show that only this case is possible, or in other words, $|T|$ cannot be greater than one.

Let if possible $|T| \geq 2$. Let $e_1 = \{x, y\} \in T$. Clearly, $\{x, y\}$ is connected in $G - \{e_1\}$. Let $R[x, y]$ be a minimal path in $G - \{e\}$. By Lemma 2.1, $|R|$ can be 3 or 4.

Case 1. $|R| = 3$. Let $R = [x, z, y]$ say. Obviously, either the edge $\{x, z\}$ or $\{z, y\}$ lies in T, otherwise T is not an edge separator for s_2 and t_2. Now, $[x, y, z, x]$ is a cycle of length 3 and $x, y, z \in V(P)$. A contradiction to the minimality of P. So, $|R|$ cannot be three.

Case 2. $|R| = 4$. Let $R = [x, z_1, z_2, y]$. Clearly, at least one of the three edges (other than $\{x, y\}$) should belong to T for T to be the edge separator. Also, not more than one edge of these should belong to T, because that would contradict the minimality of P. For the same reason, $\{z_1, z_2\}$ cannot belong to T. Hence, either $\{x, z_1\}$ or $\{y, z_2\}$ should belong to T. Let the former (call the edge as e_2) belong to T. That is, we have e_1 and e_2 (adjacent edges which belong to $P[b;c]$ and also to T) and the other two edges (say e_3 and e_4–we call them as *parallel edges*) forming a four–cycle. This means that we can use the parallel edges to find a path between s_2 and t_2 in $G - E(P[p;c])$, a contradiction. Hence EDP is false. ■

Before proceeding further we state a lemma about the vertices of attachment of a bridge in $\boldsymbol{B}$.

Lemma 3.5. *For a bridge $B \in \boldsymbol{B}$, the maximum distance between any two consecutive vertices of attachment is two.*

Proof. If the distance between two consecutive vertices is greather than two, then a chordless cycle including the subpath between these two vertices and some vertices in B would violate Lemma 2.1. ■

Corollary 3.5.1. *Let B_s and B_t be overlapping. Then they will have their respective vertices of attachment on alternate vertices on P in the width of overlap.*

Proof. Otherwise the two bridges will become mutually traversable. Refer Figure 3(b). ■

Corollary 3.5.2. *More than two bridges cannot overlap in the same interval.*

Remark 3.1. The above corollaries imply that when B_s and B_t are overlapping we have to construct the edge–disjoint paths between the two pairs of vertices using only the edges of these two bridges and the edges of their widths. The other bridges will not be of any use.

Definition 3.3. Let B_s and B_t be overlapping. Let $P'(P'')$ be a chordless path between $a(c)$ and $b(d)$ in $B_s(B_t)$. Let $C(C')$ be the set of chords connecting vertices of P and $P'(P'')$. A vertex belonging to $P'(P'')$ or P is said to be *cross–over* vertex if it has at least two chords emanating from it.

For the following lemmas we assume P', P'', C, C'' as defined above and B_s and B_t as overlapping.

Lemma 3.6. *EDP is true in each of the following cases:*

1. *If $s_2(t_2)$ belongs to P.*
2. *If there is a path connecting $s_2(t_2)$ to P in $B_s(B_t) - E(P')(E(P''))$.*

Proof. Easy. ∎

Lemma 3.7. *EDP is true in each of the following cases:*

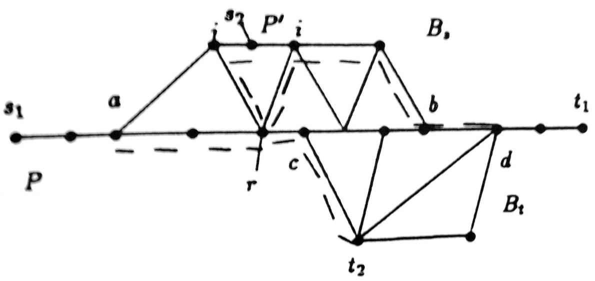
1. *Let $s_2(t_2)$ belong to $P'(P'')$ and further let it be connected by a chord to P.*
2. *Let $s_2(t_2)$ does not belong to $P'(P'')$ and there exists a path from $s_2(t_2)$ to a vertex say $l_s(l_t)$ in $P'(P'')$. Further let $l_s(l_t)$ be connected by a chord to P.*

Proof. Easy. ∎

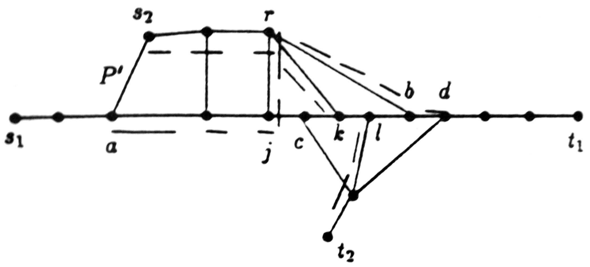
Remark 3.2. Hereafterwords, we refer to $l_s(l_t)$ as quoted in the above lemma i.e., it is the endpoint of a path from $s_2(t_2)$ to $P'(P'')$. Note that $l_s(l_t)$ comes into the picture only when $s_2(t_2)$ does not belong to $P'(P'')$.

Definition 3.4. s_2 is called a *special* vertex if either of the following is true:

- Let s_2 be adjacent to a or b (or both) in P' and further there exists no chord between s_2 to any vertex in P.
- Let $s_2 \notin P'$ and l_s be adjacent to a or b (or both) in P'. Further l_s ($\in P'$) be a cutvertex and there exists no chord between l_s to any vertex in P.



(a) s_2 is not adjacent to a.



(b) s_2 is next to a in P'

Figure 4. Extraction of paths when interval of overlap > 1. r is a cross–over vertex. The dotted lines indicate the two edge–disjoint paths:
(a) $P[s_1; r].\{r, i\}.P'[i; b].P[b; t_1]$ and $P'[s_2; j].\{j, r\}.P[r; c].CC_{B_t}(c, t_2)$.
(b) $P[s_1; j].\{j, r\}.P'[r; b], P[b; t_1]$ and $P'[s_2; r].\{r, k\}.CC_{B_t}(k, t_2)$.

A similar definition can be stated for t_2 with respect to B_t.

Lemma 3.8. *Let the width of overlap between B_s and B_t be greater than one. Then EDP is true.*

Proof. Let EDP be false by Lemmas 3.6. and 3.7. Then it implies that $s_2(t_2)$ or $l_s(l_t)$ belongs to $P'(P'')$ and there is no chord connecting either of these vertices to P. Refer Figure 4. We consider B_s. Let $s_2(l_s)$ not be a special vertex. Let $r_1(r_2)$ be the vertices adjacent to s_2 in P' and $r \in P$. Since the width of overlap is greater than one and by Lemma 2.1 r_1, s_2, r_2, r form a 4–cycle. Clearly r is a cross–over vertex. We extract paths as shown in Figure 4(a). If $s_2(l_s)$ is a special vertex we extract paths as shown in Figure 4(b). Again, since the width of overlap being greater than one guarantees a cross–over vertex. Equivalently, we can extract the paths working in B_t. ∎

Lemma 3.9. *Let width of overlap between B_s and B_t be one or zero. If $s_2(t_2)$ is not a special vertex then EDP is true.*

Proof. The extraction of paths is same as given in the previous lemma when $s_2(t_2)$ is not a special vertex. ∎

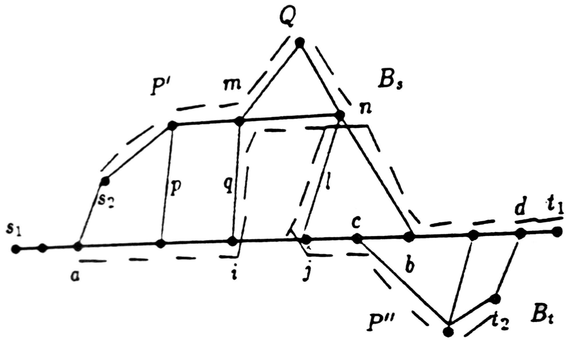


Figure 5. Extraction of paths when interval of overlap is one and both s_2 and t_2 are special vertices. Q is a path connecting two vertices (m and n) of P'. $C = \{p, q, l\}$ is the set of chords connecting vertices of P and P'. The presence of Q ensures the presence of edge–disjoint paths when there are no cross–over vertices. The edge–disjoint paths are $P[s_1; i].\{i, m\}.P'[m; b].P[b; t_1]$ and $P'[s_2; m].Q(m, n).\{n, j\}.P[j; c].P''[c; t_2]$.

Lemma 3.10. *Let width of overlap between B_s and B_t be one or zero. Let s_2 and t_2 be both special vertices. Then EDP is true only if at least one of the following exists:*

1. *If there is a cross–over vertex in P or P' or P''.*
2. *If there exist a path in $B_s(B_t) - E(P')(E(P'')) - C(C')$ between any two vertices of P or $P'(P'')$.*

Proof. If there exists a cross–over vertex, we can extract the paths as indicated in Figure 4(b).
For the second case refer Figure 5. Since s_2 and t_2 are both special vertices we can assume without loss of generality that s_2 is adjacent to a in P' and t_2 is adjacent to d in P''. Consider the cycle $P'(a, b).P[b; d].P''(d, c).P[c; a]$. We have to find edge–disjoint paths between s_2, t_2 and a, d (this can be extended to s_1, t_1). Let there be a path Q between vertices $x \in P$ and $y \in P'$. Other cases where both belong to P or P' or P'' etc. can be dealt similarly. There exist a chord $\in C$ between x and some vertex say, x' (can be y too) in P', since otherwise it would imply the presence of a cross–over vertex in P' by Lemma 2.1.

Let x' be a nearer to s_2 on P'. Then, the required edge–disjoint paths are $P[a;x].Q(x,y).P'[y;b].P[b;d]$ and $P'[s_2;x'].\{x',x\}.P[x;c].P''[c;t_2]$.

It is obvious that edge–disjoint paths cannot be extracted when no path like Q exists. ■

We now give the algorithm for finding out the edge–disjoint paths between the two pairs of vertices based on the above lemmas.

Algorithm: Two edge–disjoint paths for a general undirected permutation graph
Input: A graph $G=(V,E)$ and four distinct vertices $s_1, t_1, s_2, t_2 \in V$.
Output: EDP is *true* or *false.*

begin
1) Find a shortest path P between s_1 and t_1;
2) Find all bridges $\boldsymbol{B}=\{B_1, B_2,...\}$ with respect to P;
3) Find B_s and B_t;
/*B_s contains s_2 and B_t contains t_2*/
4) **If** $B_s = B_t$ **then return** EDP is *true*;
/*$B_s \neq B_t$*/
5) **If** B_s and B_t are mutually traversable **then return** EDP is *true.*
/*B_s and B_t do not share any common vertex */
6) **If** there is a path between s_2 and t_2 in $G-E(P)$ **then return** EDP is *true.*
/*$E(P)$ is an edge separator for s_2 and t_2*/
/* Henceforth we assume that B_s has endpoints a, b and B_t has endpoints c, d.*/
7) **If** B_s and B_t are non–overlapping **then**
begin
/* The order of endpoints in P, without loss of generality is a, b, c, d*/
  7.1) **If** there exists no path between s_1 and t_1 in $G-E(P[b;c])$
    then return EDP is *false;*
  else return EDP is *true;*
  /* Lemma 3.4. gives the extraction of paths */
end
8) **else**
/*B_s and B_t are overlapping */
begin
  8.1) Find P', P'', C, C' as defined in Definition 3.3.
  8.2) **If** width of overlap >1
  then return EDP is *true;*

/* extract paths by Lemma 3.8. */
8.3) **else**
/* width of overlap is one or zero */
begin
8.3.1) **If** (s_2 is not a special vertex) **or**
(t_2 is not a special vertex)
then return EDP is *true;*
/* extract paths by Lemma 3.9. */
8.3.2) **else**
/* both s_2 and t_2 are special vertices */
begin
8.3.2.1) **If** there is a cross–over vertex in P or P' or P''
then return EDP is *true;*
8.3.2.2) **If**
(there is a path in $B_s - P' - C$ between any two vertices of P or P') **or**
(there is a path in $B_t - P'' - C'$ between any two vertices of P or P'')
then return EDP is *true;*
/* extract paths by Lemma 3.10. */
8.3.2.3) **return** EDP is *false;*
end
end
end
end

The proof of coreectness of the algorithm follows directly from the corresponding lemmas. We discuss the implementation and complexity of the algorithm in the next section.

4. Implementation and Complexity

We assume linear as $O(|V|+|E|)$. We show that all steps in the algorithm can be implemented in linear time.

We assume the adjacency list representation of the graph G.

Step 1 is trivial and can be found in linear time.

In step 2 for finding the bridges with respect to P, we search for the connected components of $G - P$. This can be implemented using the standard depth first search technique. Then using the adjacency list of G the attachments of the bridges are found as vertices adjacent to a vertex in a connected component. By scanning P and knowing the attachments of a bridge we can easily find its endpoints and hence its width.

Step 3, which involves identifying B_s and B_t is trivial as we have to merely see in which connected component(s) s_2 and t_s belong(s).

Steps 4,5 and 6 are trivial.
In step 7 we check whether B_s and B_t are overlapping or not. This can be done by scanning their endpoints in P. Step 7.1 is also easy.

Step 8 is performed only if the bridges are overlapping. Finding $P', P''C, C'$ (step 8.1) can be done in a straightforward way. By scanning vertices of $P, P'P''$ and from the chords of C and C' we can identify the cross–over vertices. We can also easily check the width of overlap.

Step 8.3.1 checks whether s_2 or t_2 are special vertices. This can be done in a straightforward way from the definition of a special vertex.

In Step 8.3.2.2 for finding out the existence of a path in $B_s - P' - C$ or in $B_t - P'' - C'$ we simply have to check for an empty graph. Outputting the paths if EDP is true can also be done easily.

We have shown that all steps can be done linearly and hence the complexity of the algorithm is $O(|V| + |E|)$.

5. CONCLUSION

The approach used in this paper to find two edge–disjoint paths is quite general and hence can by tried on other classes of special graphs to obtain efficient algorithms. Particularly, graphs having some special property with respect to their cycles can benefit from using this approach.

We feel that this approach also lends itself to efficient parallelization, especially if the algorithm for finding the bridges has an efficient parallel implementation as is in the case of permutation graphs [AKP].

REFERENCES

[AKP] K. Arvind, V. Kamakoti, C. Pandu Rangan, *Efficient Parallel Algorithms for Permutation Graphs*, to appear in Journal of Parallel and Distributed Computing.

[BM 80] J. A. Bondy, U.S.R. Murty, Graph Theory with Applications, (Macmillan Press, 1976).

[C 80] A. Cypher, *An approach to the* k *paths problem*, Proc. of the 12th STOC (1980) 211–217.

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

[F 85] A. Frank, *Edge–disjoint paths in planar graphs*, J. Combin. Theory (B) **39** (1985) 164–178.

[GP] C. P. Gopalakrishnan, C. Pandu Rangan, *The two paths problem on permutation graphs*, (submitted).

[LR 78] A. LaPaugh, R. L. Rievest, *The subgraph homeomorphism problem*, Proc. of the 10th STOC (1978) 40–50.

[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 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*, to appear.

[S 90] A. Schwill, *Nonblocking graphs*: *Greedy algorithms to compute disjoint paths*, Proc. of the 7th STACS (1990) 250–262.

[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*, Proc. of Fifteenth ACM Symposium on the Theory of Computing (1983) 457–466.

[SP 91] A. Srinivasa Rao, C. Pandu Rangan, *Linear algorithms for parity path and two path problems on circular arc graphs*, BIT **31** (1991) 182–193.

[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 4 May 1994