THE SUBGRAPH HOMEOMORPHISM PROBLEM
ON REDUCIBLE FLOW GRAPHS

T. Hirata and M. Kimura
Dept. of Information Science
Faculty of Engineering, Tohoku University
Sendai, Japan 980

Abstract We investigate the subgraph homeomorphism problem in which one would like to determine whether a fixed pattern graph is homeomorphic to a subgraph of an input graph. We show that for every fixed pattern graph, there is a polynomial-time algorithm to solve the problem if the input graphs are restricted to reducible flow graphs.

1. Introduction

For a fixed pattern graph H , the subgraph homeomorphism problem (SHP) with respect to H asks to determine whether H is homeomorphic to a subgraph of an input graph G . The graphs G and H are either both directed or both undirected.

There are many graph properties for which the recognition problems can be reduced to the SHPs. For example, the recognition problem of planar graphs can be reduced to the SHP with respect to K_5 and $K_{3,3}$, and that of series-parallel graphs can be reduced to the SHP with respect to K_4 [2]. The SHP arises also in the study of program schema. In fact, many schema properties are characterized by the presence or reachability of certain substructures [7].

"Homeomorphism" between H and a subgraph of G is defined in terms of two mappings h_1 and h_2 : h_1 from vertices of H to vertices of G , and h_2 from edges of H to pairwise disjoint paths in G . If we are concerned with only homeomorphisms with h_1 specified a priori, then we refer to the SHP as the fixed SHP. This restricted problem can be viewed as the problem of deciding whether there is a set of disjoint paths connecting the specified vertices of G . In the

case of undirected graphs, A. S. LaPaugh and R. L. Rivest have shown that there exists a linear time algorithm to solve the fixed SHP if the pattern graph consists of exactly one triangle [8]. Y. Shiloach has given a polynomial-time algorithm for the fixed SHP with a pattern graph consisting of two disjoint edges [10]. When it comes to directed graphs, however, the fixed SHP is known to be NP-complete even if the pattern graph consists of exactly two disjoint edges [3]. This implies that the fixed SHP is NP-complete if a pattern graph contains at least two disjoint edges [3].

In this paper we first present a polynomial-time algorithm for reducible flow graphs to solve the fixed SHP with respect to the pattern graph consisting of only two edges. Extending the algorithm, we next give a polynomial-time algorithm to solve the (fixed) SHP with respect to any pattern graph H for reducible flow graphs.

2. Definitions

Let $G=(V,E)$ be a directed graph, where V is a finite set of vertices and E is a subset of $V \times V$, each member of which is called an edge. For an edge (x,y) , vertex x is called its initial endpoint and y its terminal endpoint. Edge (x,y) is said to leave vertex x and enter vertex y . We say that x is a predecessor of y , and y is a successor of x . A path P of length $k-1$ from x_1 to x_k is a sequence of vertices $(x_1, \dots, x_k)$ with $(x_i, x_{i+1}) \in E$ for $1 \leq i \leq k-1$. If a path consists of a single vertex, the length is 0. P is called a cycle if $x_1 = x_k$ and $x_i \neq x_j$ for $1 \leq i < j \leq k-1$. A cycle of length 1 is called a loop. P is said to be simple if no two vertices in P are identical. Two simple paths are disjoint if they have no vertices in common except for their endpoints. A graph $G'=(V',E')$ is called a subgraph of G if $V' \subset V$ and $E' \subset E$. Furthermore G' is an induced subgraph of G if $E' = E \cap (V' \times V')$. A flow graph, denoted by a triple $F=(V,E,s)$, is a directed graph (V,E) such that there is a path from the initial vertex $s \in V$ to each vertex in G . In a flow graph, vertex x dominates vertex y if every path from the initial vertex to y passes through x .

Given directed graphs G and H , we say that H is homeomorphic to a subgraph of G if there exist two one-to-one mappings h_1 and h_2 , h_1 from the set of vertices of H into the set of vertices of G , and h_2 from the set of edges of H onto a set of pairwise disjoint paths in G , such that for every edge (x,y) of H the path $h_2(x,y)$ starts from $h_1(x)$ and ends at $h_1(y)$. For a fixed pattern graph H , the

subgraph homeomorphism problem with respect to H asks to determine whether H is homeomorphic to a subgraph of an input graph G . (A pattern graph might be allowed to have multi edges.) If we concerned only homeomorphisms with h_1 specified a priori, then we refer to this restricted problem as the fixed subgraph homeomorphism problem. The k disjoint paths problem is to determine, given a directed graph G and k pairs of vertices $(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$ of G , whether there is a set of k disjoint paths $P_1, \dots, P_k$ in G with each P_i running from s_i to t_i , $1 \leq i \leq k$.

3. Reduction of a flow graph

M. S. Hecht and J. D. Ullman defined reducible flow graphs in terms of the following two transformations T_1 and T_2 on flow graphs [5]. Let $F=(V,E,s)$ be a flow graph. Transformation from F to a new flow graph by removing a loop is called the transformation of type T_1 . (We refer to this transformation as T_1 henceforce for simplicity.) Let x and y be vertices in F such that x is the unique predecessor of y , and y is not the initial vertex s . The transformation from F to a new flow graph F' by replacing vertices x, y and edge (x, y) with a new vertex, say $z \notin V$, is called the transformation of type T_2 (We refer to this transformation as T_2 henceforce); predecessors of x become those of z in the new flow graph; and successors of x and y become those of z . F' has a loop (z, z) if and only if F has an edge (x, y) or (x, x) . We illustrate these two transformations in Fig. 1.

Fig. 1 - Transformations T_1 and T_2 .

A given flow graph F is transformed to a flow graph F_ℓ if T_1 and T_2 are applied in any order to F until they can no longer be applied; the resulting flow graph F_ℓ is called the limit graph of F . It is known that the limit graph F_ℓ of F is unique, independent of the sequence of applications of T_1 and T_2 actually chosen [5].

If $F_{\mathcal{L}}$ consists of a single vertex, the original flow graph F is said to be reducible. The class of reducible flow graphs defined above is identical with the class defined by F. E. Allen and J. Cock in terms of "intervals" [5].

As we proceed to apply T_1 and T_2 to a flow graph F , each vertex of an intermediate graph can be viewed to represent a subgraph, called a region, of the original flow graph F , and each edge to represent a set of edges of F , which are defined as follows.

- (i) In the original flow graph, every vertex v represents region $(\{v\}, \emptyset)$, and every edge e represents singleton set $\{e\}$.
- (ii) If T_1 is applied to vertex n with the loop (n, n) and they are replaced by a new vertex n' , then n' represents region $(V', E' \cup E'')$, where (V', E') is the region represented by n , and E'' is the edge set represented by (n, n) .
- (iii) If T_2 is applied to n_1 and n_2 with edge (n_1, n_2) and they are replaced by a new vertex n , then n represents region $(V_1 \cup V_2, E_1 \cup E_2 \cup E_3)$, where (V_1, E_1) is the region represented by n_1 , and (V_2, E_2) the region represented by n_2 , and E_3 is the edge set represented by (n_1, n_2) . If two edges (n_1, n') and (n_2, n') are replaced by edge (n, n') , then (n, n') represents the union of the two sets represented by (n_1, n') and by (n_2, n') .

The following lemma is implicit in [6].

Lemma 1. Let $F=(V, E, s)$ be a flow graph. Let F' be a flow graph formed at some stage of the reduction of F .

- (i) Let n be a vertex of F' . If n represents (V', E') , and $(n_1, n_2) \in E'$, then $n_1 \in V'$ and $n_2 \in V'$ (that is, (V', E') is a graph).
- (ii) Let n_1 and n_2 be (not necessarily distinct) vertices of F' and (n_1, n_2) an edge of F' . If n_1 and n_2 represent (V', E') and (V'', E'') respectively and (n_3, n_4) belongs to the set represented by (n_1, n_2) , then $n_3 \in V'$ and $n_4 \in V''$.
- (iii) The family of the vertex sets of all the regions represented by vertices of F' is a partition of the set of vertices of F .
- (iv) The family of all edge sets represented by edges of F' is a partition of the set of all edges of G with two ends in distinct regions.

A parse of a reducible flow graph F is a sequence of T_1 and T_2 which are applied to F until the limit graph results. Formally

it is defined to be a sequence of four-tuples (T_1, u, v, S) and five-tuples (T_2, u, v, w, S) as follows, where u, v , and w are vertices of F , and S is a set of edges of F .

(i) The parse of a flow graph consisting of a single vertex is an empty sequence.

(ii) If F' is reduced to F'' by an application of T_1 to vertex u with loop (u, u) , and u is replaced with a vertex v in F'' , then (T_1, u, v, S) followed by a parse of F'' is a parse of F' , where S is the set of edges represented by edge (u, u) eliminated from F' .

(iii) If F' is reduced to F'' by applying T_2 to edge (u, v) , and u and v are replaced with a new vertex w , then (T_2, u, v, w, S) followed by a parse of F'' is a parse of F' , where S is the set of edges represented by edge (u, v) of F' .

Generally the parse of a flow graph is not unique. Fig. 2 illustrates a parsing of a flow graph, and Fig. 3 illustrates the regions of the graph.

Fig. 2 - Example of a parsing.

The parse in Fig. 2 is:

$$\begin{aligned} \Pi_F = & (T_2, n_2, n_3, n_5, \{(n_2, n_3)\}), \\ & (T_1, n_5, n_6, \{(n_3, n_2)\}), \\ & (T_2, n_6, n_4, n_7, \{(n_3, n_4), (n_2, n_4)\}), \\ & (T_2, n_1, n_7, n_8, \{(n_1, n_2)\}), \\ & (T_1, n_8, n_9, \{(n_4, n_1)\}) \end{aligned}$$

According to Π_F one can form larger and larger regions by combining smaller regions as illustrated in Fig. 3.

Fig. 3 - Formation of regions.

The following lemma illustrates how regions are combined and how edges are incorporated to regions. The proof can be found in [6].

Lemma 2. Let F be a flow graph and R a region of F . There exists a vertex h in R such that (a) any edge not contained in R but entering some vertex of R has necessarily h as its terminal endpoint, and (b) h dominates every vertex of R . Hence, if R' is R plus all the edges that leave vertices of R and enter h of R , then R' is an induced subgraph of F .

It is clear from the lemma above that every path from a vertex outside a region R to some vertex of R must pass through h of R . Since there is a path in R from h to every vertex of R , R is itself a flow graph with initial vertex h . We call h the header of R .

Suppose that there is a vertex n_1 with edge (n_1, n_1) in some intermediate graph of the reduction of a flow graph F . If n_1 represents a region $R' = (V_1, E_1, h_1)$ of F , each edge in the set represented by (n_1, n_1) has its initial and terminal endpoints in R' by (ii) of Lemma 1, and in particular the terminal endpoint is h_1 by Lemma 2. Fig. 4 shows how a region is expanded by T_1 , where R is the region represented by the new vertex n . We say that R is formed by applying T_1 to R' , or R' is expanded to R .

Fig. 4 - Expansion of a region.

Suppose that there are vertices n_1 and n_2 with edge (n_1, n_2) in some intermediate graph of the reduction of F . If n_1 and n_2 represent regions $R' = (V_1, E_1, h_1)$ and $R'' = (V_2, E_2, h_2)$ respectively, then each edge in the set represented by (n_1, n_2) has its initial endpoint in R' and its terminal endpoint h_2 in R'' by Lemmas 1 (ii) and 2. Fig. 5 illustrates the combination of two regions by T_2 , where R is the region represented by the new vertex n . We say that R is formed by applying T_2 to R' and R'' , or R' and R'' are combined to be R .

Fig. 5 - Union of regions.

4. An algorithm to solve the two disjoint paths problem

The fixed SHP with a pattern graph consisting of two edges is reduced to the two disjoint paths problem. In this section we give an algorithm to solve the two disjoint paths problem for reducible

flow graphs. To begin with, we define three propositional functions $A_R^1(n_1, n_2)$, $A_R^2(n_1, n_2, n_3, n_4)$ and $B_R^2(n_1, n_2, n_3, n_4, n_5)$ for region R , where n_1, n_2, n_3, n_4 , and n_5 range over all the vertices of R .

Definition 1. Let F be a flow graph, $R=(V_1, E_1, h)$ a region of F .

(a) $A_R^1(n_1, n_2)=\text{TRUE}$ iff there is a path in R from n_1 to n_2 .

(b) $A_R^2(n_1, n_2, n_3, n_4)=\text{TRUE}$ iff there are two disjoint paths in R such that one runs from n_1 to n_2 and the other from n_3 to n_4 .

(c) $B_R^2(n_1, n_2, n_3, n_4, n_5)=\text{TRUE}$ iff there are three pairwise disjoint paths in R such that one runs from n_1 to n_2 , another from n_3 to n_4 , and the other from h to n_5 .

In terms of $A_R^2(n_1, n_2, n_3, n_4)$, our aim is to compute the value of $A_R^2(s_1, t_1, s_2, t_2)$ for a given flow graph F with specified vertices s_1, t_1, s_2 , and t_2 . Our algorithm computes the values of these functions recursively for larger and larger regions, using Lemmas 3 and 4. Note that $A_R^1(n, n) = A_R^2(n, n, n, n) = B_R^2(n, n, n, n, n) = \text{TRUE}$ if region R consists of a single vertex.

Lemma 3. Let $R=(V, E, h)$ be a region formed by an application of T_1 from region $R'=(V, E_1, h)$. (See Fig.4.) Then,

$$A_R^1(n_1, n_2) = A_{R'}^1(n_1, n_2) + \sum_{n' \in M} A_{R'}^2(h_1, n_2, n_1, n'), \quad \text{---- (1)}$$

$$\begin{aligned} & A_R^2(n_1, n_2, n_3, n_4) \\ &= A_{R'}^2(n_1, n_2, n_3, n_4) \\ &+ \sum_{n' \in M} B_{R'}^2(n_1, n_2, n_3, n', n_4) \times ((n'=n_3) + (n' \neq n_3) \times (n' \neq n_1) \times (n' \neq n_2)) \\ &\quad \times ((h=n_4) + (h \neq n_4) \times (h \neq n_1) \times (h \neq n_2)) \\ &+ \sum_{n' \in M} B_{R'}^2(n_3, n_4, n_1, n', n_2) \times ((n'=n_1) + (n' \neq n_1) \times (n' \neq n_3) \times (n' \neq n_4)) \\ &\quad \times ((h=n_2) + (h \neq n_2) \times (h \neq n_3) \times (h \neq n_4)) \\ &+ \sum_{n', n'' \in M} A_{R'}^2(n_1, n', n_3, n'') \times (n_2=n_4) \\ &\quad \times ((n'=n_1) + (n' \neq n_1) \times (n' \neq n_3) \times (n' \neq n'')) \\ &\quad \times ((n''=n_3) + (n'' \neq n_3) \times (n'' \neq n_1) \times (n' \neq n'')), \text{ and} \quad \text{--- (2)} \end{aligned}$$

$$B_R^2(n_1, n_2, n_3, n_4, n_5) = B_{R'}^2(n_1, n_2, n_3, n_4, n_5), \quad \text{---- (3)}$$

where $n_i, 1 \leq i \leq 5$, range over all vertices in V , $+$ and Σ denote logical OR, $\times$ denotes logical AND, and M is the set of initial endpoints of all the edges in $E-E_1$.

Proof. We first establish the formula (1). Suppose that the left-

hand side of (1) is TRUE, that is, there is a simple path from n_1 to n_2 in R . Then there are two cases : (i) the path is also in R' , and (ii) it is not in R' . In the former case, the first term of the right-hand side of (1) is TRUE. In the latter case, the path must contain an edge in $E-E_1$. Therefore there are two disjoint paths in R' , one running from n_1 to one of initial endpoints of the edges in $E-E_1$, and the other from the header of R' to n_2 . In this case, the second term becomes TRUE. It is obvious that if the right-hand side of (1) is TRUE, then the left-hand side is TRUE. Thus the formula (1) holds.

We next verify the formula (2). Suppose that the left-hand side of (2) is TRUE, that is, there are two disjoint paths in R , one running from n_1 to n_2 and the other from n_3 to n_4 . Then we have four cases : (i) these paths are also in R' , (ii) the path from n_3 to n_4 contains an edge in $E-E_1$ and the other does not, (iii) the path from n_1 to n_2 contains an edge in $E-E_1$ and the other does not, (iv) both of the two paths contain edges in $E-E_1$. In the case (i), the first term of the right-hand side of (2) is TRUE. In the case (ii), there must be three pairwise disjoint paths in R' such that one runs from n_1 to n_2 , another from n_3 to a vertex n' that is an initial endpoint of an edge in $E-E_1$, and the other from the header of R' to n_4 . Furthermore, if $n' \neq n_3$, then n' can be neither n_1 nor n_2 , and if $h \neq n_4$, then h can be neither n_1 nor n_2 . In this case, the second term of (2) is TRUE. By an argument similar to above, the third term is TRUE for the case (iii) and the fourth term for the case (iv). It is obvious that if the right-hand side of (2) is TRUE, then the left-hand side is TRUE. Thus the formula (2) holds.

Finally, we can easily verify the formula (3) from the fact that h is the terminal endpoint of each edge in $E-E_1$.

Q.E.D.

Lemma 4. Let $R=(V,E,h_1)$ be a region formed by applying T_2 to two regions $R'=(V_1,E_1,h_1)$ and $R''=(V_2,E_2,h_2)$. (See Fig. 5.) (We abbreviate $A_R^1(n_1,n_2)$, $A_R^2(n_1,n_2,n_3,n_4)$ and $B_R^2(n_1,n_2,n_3,n_4,n_5)$ to A_R^1 , A_R^2 and B_R^2 , respectively.) Let $n_i, 1 \leq i \leq 5$, be vertices in V , and M the set of initial endpoints of the edges in $E-E_1-E_2$. Then,
 (a) A_R^1 is computed from $A_{R'}^1$ and $A_{R''}^1$ as follows.

$$\text{Case 1: } n_1, n_2 \in V_1. \\ A_R^1(n_1, n_2) = A_{R^1}^1(n_1, n_2). \quad (4)$$

$$\text{Case 2: } n_1, n_2 \in V_2. \\ A_R^1(n_1, n_2) = A_{R''}^1(n_1, n_2). \quad (5)$$

$$\text{Case 3: } n_1 \in V_1 \text{ and } n_2 \in V_2. \\ A_R^1(n_1, n_2) = \left(\sum_{n' \in M} A_{R^1}^1(n_1, n') \right). \quad (6)$$

$$\text{Case 4: } n_2 \in V_1 \text{ and } n_1 \in V_2. \\ A_R^1(n_1, n_2) = \text{FALSE}. \quad (7)$$

(b) A_R^2 is computed as follows.

$$\text{Case 1: } n_1, n_2, n_3, n_4 \in V_1. \\ A_R^2 = A_{R^1}^2(n_1, n_2, n_3, n_4). \quad (8)$$

$$\text{Case 2: } n_1, n_2, n_3, n_4 \in V_2. \\ A_R^2 = A_{R''}^2(n_1, n_2, n_3, n_4). \quad (9)$$

$$\text{Case 3: } n_1, n_2, n_3 \in V_1 \text{ and } n_4 \in V_2. \\ A_R^2 = \sum_{n' \in M} A_{R^1}^2(n_1, n_2, n_3, n') \\ \times ((n' = n_3) + (n' \neq n_3) \times (n' \neq n_1) \times (n' \neq n_2)). \quad (10)$$

Case 4: $n_1, n_3, n_4 \in V_1$ and $n_2 \in V_2$. Similar to Case 3.

$$\text{Case 5: } n_1, n_2 \in V_1 \text{ and } n_3, n_4 \in V_2. \\ A_R^2 = A_{R^1}^1(n_1, n_2) \times A_{R''}^1(n_3, n_4), \quad (11)$$

Case 6: $n_3, n_4 \in V_1$ and $n_1, n_2 \in V_2$. Similar to Case 5.

$$\text{Case 7: } n_1 \in V_1 \text{ and } n_2, n_3, n_4 \in V_2. \\ A_R^2 = \left(\sum_{n' \in M} A_{R^1}^1(n_1, n') \right) \times A_{R''}^2(h_2, n_2, n_3, n_4) \\ \times ((h_2 = n_2) + (h_2 \neq n_2) \times (h_2 \neq n_3) \times (h_2 \neq n_4)). \quad (12)$$

Case 8: $n_3 \in V_1$ and $n_1, n_2, n_4 \in V_2$. Similar to Case 7.

$$\text{Case 9: } n_1, n_3 \in V_1 \text{ and } n_2, n_4 \in V_2. \\ A_R^2 = \sum_{n', n'' \in M} A_{R^1}^2(n_1, n', n_3, n'') \times (n_2 = n_4 = h_2) \\ \times ((n' = n_1) + (n' \neq n_1) \times (n' \neq n_3) \times (n' \neq n'')) \\ \times ((n'' = n_3) + (n'' \neq n_3) \times (n'' \neq n_1) \times (n'' \neq n'')). \quad (13)$$

$$\text{Case 10: otherwise.} \\ A_R^2 = \text{FALSE}. \quad (14)$$

(c) B_R^2 is computed as follows.

$$\text{Case 1: } n_1, n_2, n_3, n_4, n_5 \in V_1. \\ B_R^2 = B_{R^1}^2(n_1, n_2, n_3, n_4, n_5). \quad (15)$$

$$\text{Case 2: } n_1, n_2, n_3, n_4, n_5 \in V_2. \\ B_R^2 = B_{R''}^2(n_1, n_2, n_3, n_4, n_5) \\ \times ((h_2 = n_5) + (h_2 \neq n_5) \times (h_2 \neq n_1) \times (h_2 \neq n_2) \times (h_2 \neq n_3) \times (h_2 \neq n_4)). \quad (16)$$

Case 3: $n_1, n_3, n_4, n_5 \in V_1$ and $n_2 \in V_2$.

$$B_R^2 = \sum_{n' \in M} B_R^2(n_1, n', n_3, n_4, n_5) \times ((n' = n_1) + (n' \neq n_1) \times (n' \neq h_1) \times (n' \neq n_3) \times (n' \neq n_4) \times (n' \neq n_5)). \quad (17)$$

Case 4: $n_1, n_2, n_3, n_5 \in V_1$ and $n_4 \in V_2$. Similar to Case 3.

Case 5: $n_1, n_2, n_3, n_4 \in V_1$ and $n_5 \in V_2$. Similar to Case 3.

Case 6: $n_3, n_4, n_5 \in V_1$ and $n_1, n_2 \in V_2$.

$$B_R^2 = A_R^2(h_1, n_5, n_3, n_4) \times A_R^1(n_1, n_2). \quad (18)$$

Case 7: $n_1, n_2, n_5 \in V_1$ and $n_3, n_4 \in V_2$. Similar to Case 6.

Case 8: $n_1, n_3, n_5 \in V_1$ and $n_2, n_4 \in V_2$.

$$B_R^2 = \sum_{n', n'' \in M} B_R^2(n_1, n', n_3, n'', n_5) \times (n_2 = n_4 = h_2) \times ((n' = n_1) + (n' \neq n_1) \times (n' \neq h_1) \times (n' \neq n_3) \times (n' \neq n_5) \times (n' \neq n'')) \times ((n'' = n_3) + (n'' \neq n_3) \times (n'' \neq h_1) \times (n'' \neq n_1) \times (n'' \neq n_5) \times (n'' \neq n'')). \quad (19)$$

Case 9: $n_1, n_3, n_4 \in V_1$ and $n_2, n_5 \in V_2$. Similar to Case 8.

Case 10: $n_1, n_2, n_3 \in V_1$ and $n_4, n_5 \in V_2$. Similar to Case 8.

Case 11: $n_3, n_4 \in V_1$ and $n_1, n_2, n_5 \in V_2$.

$$B_R^2 = \sum_{n' \in M} A_R^2(h_1, n', n_3, n_4) \times ((n' = h_1) + (n' \neq h_1) \times (n' \neq n_3) \times (n' \neq n_4)) \times A_R^2(n_1, n_2, h_2, n_5) \times ((h_2 = n_5) + (h_2 \neq n_5) \times (h_2 \neq n_1) \times (h_2 \neq n_2)). \quad (20)$$

Case 12: $n_1, n_2 \in V_1$ and $n_3, n_4, n_5 \in V_2$. Similar to Case 11.

Case 13: $n_3, n_5 \in V_1$ and $n_1, n_2, n_4 \in V_2$. Similar to Case 11.

Case 14: $n_1, n_5 \in V_1$ and $n_2, n_3, n_4 \in V_2$. Similar to Case 11.

Case 15: $n_1, n_3 \in V_1$ and $n_2, n_4, n_5 \in V_2$.

$$B_R^2 = \sum_{n', n'', n''' \in M} B_R^2(n_1, n', n_3, n'', n''') \times (n_2 = n_4 = n_5 = h_2) \times ((n' = n_1) + (n' \neq n_1) \times (n' \neq h_1) \times (n' \neq n_3) \times (n' \neq n'') \times (n' \neq n''')) \times ((n'' = n_3) + (n'' \neq n_3) \times (n'' \neq h_1) \times (n'' \neq n_1) \times (n'' \neq n') \times (n'' \neq n''')) \times ((n''' = h_1) + (n''' \neq h_1) \times (n''' \neq n_1) \times (n''' \neq n_2) \times (n''' \neq n') \times (n''' \neq n'')). \quad (21)$$

Case 16: $n_5 \in V_1$ and $n_1, n_2, n_3, n_4 \in V_2$.

$$B_R^2 = A_R^2(n_1, n_2, n_3, n_4). \quad (22)$$

Case 17: otherwise.

$$B_R^2 = \text{FALSE}. \quad (23)$$

Proof. Omitted.

Now we proceed to construct an algorithm to solve the two disjoint paths problem. We employ a 2-dimensional array $A^1(n_1, n_2)$, a 4-dimensional array $A^2(n_1, n_2, n_3, n_4)$ and a 5-dimensional array $B^2(n_1, n_2, n_3, n_4, n_5)$, and each element of these arrays has a value of $A_R^1(n_1, n_2)$, $A_R^2(n_1, n_2, n_3, n_4)$ or $B_R^2(n_1, n_2, n_3, n_4, n_5)$ with respect to a

current region R.

ALGORITHM A²

Input: A reducible flow graph $F=(V,E,s)$ with specified vertices $s_1, t_1, s_2, t_2 \in V$, and its parse Π_F .

Output: If there are two disjoint paths such that one runs from s_1 to t_1 and the other from s_2 to t_2 , then "YES", otherwise "NO".

Method:

Step 1. $\Pi \leftarrow \Pi_F$. For each vertex $n \in V$, $A^2(n, n, n, n) \leftarrow \text{TRUE}$, $B^2(n, n, n, n, n) \leftarrow \text{TRUE}$, $A^1(n, n) \leftarrow \text{TRUE}$.

Step 2. If the sequence in Π is empty, go to Step 5. Let T_Π be the top of the current sequence in Π . Delete T_Π from the sequence in Π . If T_Π is a four-tuple, that is, a transformation T_1 , then go to Step 3. If T_Π is a five-tuple, that is, a transformation T_2 , then go to Step 4.

Step 3. Compute the values of $A^1(n_1, n_2)$, $A^2(n_1, n_2, n_3, n_4)$ and $B^2(n_1, n_2, n_3, n_4, n_5)$ for the region newly formed by the transformation T_1 using the formulas in Lemma 3, where n_i , $1 \leq i \leq 5$, are vertices of the region. Go to Step 2.

Step 4. Compute the values of $A^1(n_1, n_2)$, $A^2(n_1, n_2, n_3, n_4)$ and $B^2(n_1, n_2, n_3, n_4, n_5)$ for the region newly formed by the transformation T_2 using the formulas in Lemma 4, where n_i , $1 \leq i \leq 5$, are vertices of the region. Go to Step 2.

Step 5. If $A^2(s_1, t_1, s_2, t_2) = \text{TRUE}$, then output "YES", otherwise, output "NO". Halt.

Theorem 1. There exists an algorithm of time complexity $O(n^5 e)$ to solve the two disjoint paths problem for reducible flow graphs, where n is the number of vertices and e is the number of edges in a flow graph.

Proof. It is known that there is an algorithm of $O(e \log e)$ for finding a parse of a reducible flow graph [11], and it is easy to show that the time complexity of algorithm A^2 is $O(n^5 e)$.

5. Algorithms to solve the SHP and the fixed SHP

In this section we construct an algorithm A^k to solve the k disjoint paths problem. As stated in the Introduction, the fixed SHP for any pattern graph with k edges can be reduced to this problem. Since one can easily extend the algorithm A^2 to the case for the problem above, we give only the outline and omit the details.

We define $k+1$ propositional functions as follows: for each j ($1 \leq j \leq k$), $A_R^j(n_1, n_2, \dots, n_{2j}) = \text{TRUE}$ iff there are j disjoint paths in R such that each path runs from n_{2i-1} to n_{2i} for all i ($1 \leq i \leq j$); and $B_R^k(n_1, n_2, \dots, n_{2k+1}) = \text{TRUE}$ iff there are $k+1$ disjoint paths in R such that one runs from the header of R to n_{2k+1} and each of the others runs from n_{2i-1} to n_{2i} for all i ($1 \leq i \leq k$). Then clearly, $B_R^k(n_1, \dots, n_{2k+1}) = \text{TRUE}$ and $A_R^j(n_1, \dots, n_{2j}) = \text{TRUE}$ for all j if region R consists of a single vertex. We can compute these values for larger and larger regions recursively in the same manner as in algorithm A^2 , and finally we compute $A_F^k(n_1, \dots, n_{2k})$ for a given flow graph F . Thus we have algorithm A^k . Clearly the time complexity is $O(n^{2k+1}e)$.

In order to solve the fixed SHP, it is enough to consult $A_F^k(s_1, t_1, \dots, s_k, t_k)$, where s_i and t_i are vertices of F mapped from two endpoints of the i^{th} edge of the pattern graph H , and k is the number of edges of H .

If we are concerned with the SHP, then it is sufficient to know whether there is a one-to-one mapping h_1 from vertices of H to vertices of F such that $A_F^k(h_1(s_1'), h_1(t_1'), \dots, h_1(s_k'), h_1(t_k')) = \text{TRUE}$, where s_i' and t_i' are two endpoints of the i^{th} edge of H . Since there exist at most $O(n^k)$ such mappings, this can be done within $O(n^k)$ time. Thus we obtain the following theorem.

Theorem 2. For any pattern graph H , there exists an algorithm of time complexity $O(n^{2k+1}e)$ to solve the (fixed) SHP for reducible flow graphs, where k is the number of edges of H .

ACKNOWLEDGEMENTS We would like to thank Prof. T. Nishizeki for his careful reading of the manuscript and valuable comments.

References

- [1] Aho, A., Hopcroft, J., and Ullman, J., The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass. (1974).
- [2] Duffin, R. J., "Topology of series parallel networks", J. Math. and appli., 10(1965), 303-318.
- [3] Fortune, S., Hopcroft, J., and Wyllie, J., "The directed subgraph homeomorphism problem", Theor. Comput. Sci., 10(1980), 111-121.
- [4] Harary, F., Graph Theory, Addison-Wesley, Reading, Mass. (1969).
- [5] Hecht, M. S., and Ullman, J. D., "Flow graph reducibility", SIAM J. Computing 1,2(1972), 188-202.
- [6] Hecht, M. S., and Ullman, J. D., "Characterization of reducible flow graphs", J. ACM 21, 3(1974), 367-375.
- [7] Hunt, H. B., and Szymansky, T. G., "Dichotomization, reachability and the forbidden subgraph problem", Proc. Eighth Annual ACM Symposium on Theory of Computing, Hershey, PA (1976), 126-134.
- [8] LaPaugh, A. S., and Rivest, R. L., "The subgraph homeomorphism problem", Proc. Tenth Annual ACM Symposium on Theory of Computing, San Diego, CA (1978), 40-50.
- [9] Perl, T., and Shiloach, Y., "Finding two disjoint paths between two paires of vertices in a graph", J. ACM 24, 1(1978), 1-9.
- [10] Shiloach, Y., "A polinomial solution to the undirected two paths problem", J. ACM 27, 3(1980), 445-456.
- [11] Tarjan, R. E., "Testing flow graph reducibility", J. Comput. and Syst. Sci., 9(1974) 355-365.