数理解析研究所講究録
397巻 1980年 50-85

AN ALMOST-LINEAR-TIME ALGORITHM FOR SOLVING THE GRAPH-REALIZATION PROBLEM

Satoru FUJISHIGE

Institute of Socio-Economic Planning

University of Tsuksba

Abstract

A (0,1)-matrix is called graphic if it is a fundamental circuit matrix of a graph. Given a (0,1)-matrix N, the graph-realization problem is

(i) to determine whether N is graphic and

(ii) if graphic, to realize a graph which has N as its fundamental circuit matrix.

We propose a data structure called a PQ-graph based on PQ-trees and then present an efficient algorithm for solving the graph-realization problem by means of PQ-graphs. A running time required for the algorithm is $O(\nu\alpha(\nu,k))$, where ν is the number of nonzero elements of a given (0,1)-matrix N, k is the number of rows of N and $\alpha(\cdot,\cdot)$ is a function defined in terms of Akermann's function. Since the value of $\alpha(\nu,k)$ is not more than 3 for all practical values of ν and k, we can solve the graph-realization problem in a running time almost proportional to ν, the number of nonzero elements of N.

1. Introduction

A (0,1)-matrix is called graphic if it is a fundamental circuit matrix of a graph. (The precise definition of a fundamental circuit matrix of a graph will be given in the next section.) Given a (0,1)-matrix N, The graph-realization problem is

(i) to determine whether N is graphic and

(ii) if graphic, to realize a graph which has N as its fundamental circuit matrix.

The practical importance of the graph-realization problem was recognized about twenty years ago in the theory of electric networks [9], while the problem of determining whether a given linear programme is reducible to a network problem can also be formulated as the graph-realization problem (cf. [6], [7] and [2]).

A considerable number of methods have been proposed for solving the graph-realization problem up to now (cf. [2], [5], [6], [8], [11], [12] and [13]). However, from the point of view of computational complexity, most of these methods are not so efficient and seem to be improved by employing recently developed data structures [1]. Denoting by m the number of columns of a (0,1)-matrix N, Iri's algorithm [6] requires an $O(m^6)$ worst-case running time, while N. Tomizawa has recently proposed an algorithm by describing Tutte's algorithm [12] in a more complete and efficiently computable form and asserts that the worst-case running time is $O(m^3)$. Also, R. E. Bixby and W. H. Cunningham [2] have recently proposed an $O(m\nu)$ algorithm along the idea of [12], where ν is the number of nonzero elements of N.

In the present paper, we propose a data structure called a PQ-graph based on PQ-trees due to K. S. Booth and G. S. Lueker [3]. Using PQ-graphs, we present an efficient algorithm for solving the graph-realizztion problem which requires a running time almost proportional to the number of nonzero elements of a given (0,1)-matrix N. A summary of this paper was presented in [4].

2. Definitions and Assumptions

Let $G(V,A;\partial^+,\partial^-)$ be a graph with a vertex set V, an arc set A and functions ∂^+, ∂^-: $A \to V$. Here, for an arc $a \in A$, ∂^+a and ∂^-a are end-vertices of a and they are, respectively, called the initial vertex and the terminal vertex of a. Such a graph is sometimes called a directed graph. The graph is sometimes denoted by $G(V,A)$ or more simply by G if there is no possibility of confusion. If, for each $a \in A$, we are concerned with the set $\{\partial^+a,\partial^-a\}$ but not with the ordered pair $(\partial^+a,\partial^-a)$ of the end-vertices of a, then we call the graph G an undirected graph and we use the term "edge" instead of "arc".

A path in a (directed or undirected) graph $G(V,A)$ is a sequence $(v_0,a_1,v_1,a_2,\cdots,a_n,v_n)$, with possible repetition, of vertices v_i $(0 \leqq i \leqq n)$ and arcs or edges a_i $(1 \leqq i \leqq n)$ such that

$$\{\partial^+ a_i, \partial^- a_i\} = \{v_{i-1}, v_i\} \qquad (1 \leqq i \leqq n),$$

where v_0 and v_n are called end-vertices of the path. When $n = 0$, the path is degenerate. A closed path is a path whose end-vertices coincide with each other. A path is called elementary if it traverses each vertex at most once. Furthermore, suppose that the sequence $(v_0, a_1, v_1, a_2, \cdots, a_n, v_n)$ is a path in a directed graph G. For each $i = 1, 2, \cdots, n$, if $\partial^+ a_i = v_{i-1}$ and $\partial^- a_i = v_i$, then we say that a_i is in the positive direction of the path, or else, a_i is in the negative direction. If, for each $i = 1, 2, \cdots, n$, a_i is in the positive direction of the path, then the path is called a directed path from v_0 to v_n, and v_0 and v_n are, respectively, called the initial vertex and the terminal vertex of the path. Also, when $\partial^+ a = v^+$ and $\partial^- a = v^-$ for an arc a and vertices v^+ and v^-, we say that v^+ is adjacent to v^- and v^- is adjacent from v^+. A directed graph with no directed closed paths is called acyclic.

Let $G(V,E)$ be an undirected graph with a vertex set V and an edge set E. A set of edges of an elementary closed path in G is called a circuit in G. A tree in G is a maximal set of edges which does not contain any circuits. The complement, in the edge set E, of a tree is called a cotree. When a tree or a cotree is given, an edge of the tree or the cotree is sometimes called a tree-edge or a cotree-edge. For a tree T in G and an edge $e \in E - T$, there exists a unique circuit in $T \cup \{e\}$, which is denoted by $C^*(T|e)$ and called a fundamental circuit with respect to the tree T and the edge e in the cotree $E - T$. The system of circuits $(C^*(T|e) : e \in E-T)$ is called the fundamental system of circuits with respect to the tree T. Suppose that $T = \{e_1, e_2, \cdots, e_m\}$ and $E - T = \{e_{m+1}, e_{m+2}, \cdots, e_{m+k}\}$. Then the fundamental circuit matrix with respect to the tree T is a $k \times m$ matrix with the (i,j)-element c_{ij} given by

$$\begin{aligned} c_{ij} &= 1 \quad \text{if } e_j \in C^*(T|e_{m+i}), \\ &= 0 \quad \text{otherwise} \qquad (i=1,2,\cdots,k;\ j=1,2,\cdots,m). \end{aligned}$$

Each column of a fundamental circuit matrix corresponds to an edge of the tree with respect to which the fundamental circuit matrix is defined, and each row corresponds to an edge of the associated cotree or to a fundamental circuit.

A graph $G(V,E)$ is connected if for any vertices v_1 and v_2 in V there is a path with its end-vertices v_1 and v_2. A graph $G(V,E)$ is 2-connected if, for any proper dissection $\{E_1, E_2\}$ of the edge set E, there is a circuit C such that $C \cap E_1 \neq \emptyset$ and $C \cap E_2 \neq \emptyset$, where for any set D a proper dissection $\{D_1, D_2\}$ of D is a partition of the set D into two subsets D_1 and D_2 such that $D_1 \neq \emptyset$, $D_2 \neq \emptyset$, $D_1 \cup D_2 = D$ and $D_1 \cap D_2 = \emptyset$.

Let $G_1(V,E)$ and $G_2(V,E)$ be 2-connected graphs with the same vertex set V and the same edge set E. The graphs G_1 and G_2 are said to be 2-isomorphic with each other if the set of all the circuits in G_1 and the set of all the circuits in G_2 are the same.

For a 2-connected graph $G(V,E)$, let $\{E_1, E_2\}$ be a proper dissection of the edge set E and let V_i be the set of end-vertices of edges in E_i for $i = 1, 2$. The set $V_1 \cap V_2$ is called the set of attachment vertices of the subgraph $G_1(V_1,E_1)$ (and $G_2(V_2,E_2)$) of the graph $G(V,E)$. If $|V_1 \cap V_2| = 2$, then G_1 and G_2 are called two-terminal subgraphs of G, where $|\cdot|$ denotes the cardinality.

A fundamental path in an acyclic graph is a path $P = (P_1,P_2^*)$ composed of its subpaths P_1 and P_2^* with possible repetition of arcs, where, letting P_2 be the path in reverse order of P_2^*, P_1 and P_2 are directed (possibly degenerate) paths whose terminal vertices coincide with each other. The terminal vertex is called the turning vertex of P. As shown in Figure 1, let P_1 and P_2 be directed paths from the vertex u_1 to u^* and from the vertex u_2 to u^*, respectively, and let P_2^* be the reversion of P_2. Then the composition $P = (P_1,P_2^*)$ of the paths P_1 and P_2^* is a fundamental path and u^* is the turning vertex of P. The notion of a fundamental path will be used for a data structure called a PQ-graph. (PQ-graphs will be defined in Section 3.) A fundamental path corresponds to a fundamental circuit in a graph expressed by the PQ-graph.

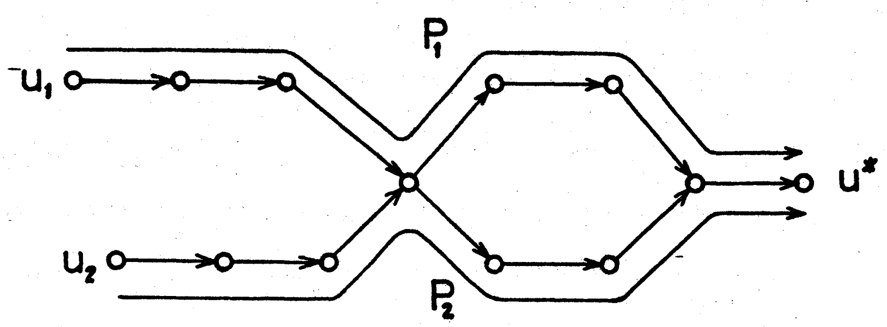


Fig. 1. An example of a fundamental path.

A <u>rooted directed tree</u> T is a directed graph with a distinguished vertex, called the <u>root</u> of T, such that the root is adjacent to no vertex and each vertex except for the root is adjacent to one and only one vertex. A vertex of T which is adjacent from no vertex is called a <u>leaf</u> of T. A roooted directed tree will be used for expressing a PQ-tree [3].

Now, let N be a $k \times m$ (0,1)-matrix whose graph-realizability should be decerned. We suppose that there is at least one nonzero element in each row and each column of N. Let R and S be the set of rows and columns of N, respectively. Denote the (r,s)-element of N by $N(r,s)$, $r(s)$ or $s(r)$ for $r \in R$ and $s \in S$. We regard each row $r \in R$ as a fundamental circuit or the corresponding cotree-edge and each column $s \in S$ as a tree-edge, even if N is non-graphic. A fundamental circuit $r \in R$ will be simply called a circuit. Also, we suppose that the information about the (0,1)-matrix N is expressed by a bipartite graph, where every nonzero element of N corresponds to an edge of the bipartite graph.

A subset K of R is called <u>connected</u> if for every proper dissection $\{K_1,K_2\}$ of K we have

$$\{s \mid s\in S, \exists r\in K_1 : s(r)=1\} \cap \{s \mid s\in S, \exists r\in K_2 : s(r)=1\} \neq \emptyset.$$

A sequence $(r_0,r_1,\cdots,r_n)$ of circuits $r_i \in R$ is called <u>sequentially connected</u> if, for each $i = 0, 1, \cdots, n$, $\{r_0,r_1,\cdots,r_i\}$ is connected. We suppose without loss of generality that $(r_0,r_1,\cdots,r_{k-1})$ is a sequentially connected sequence of all the circuits of R, since we can decompose R into maximal connected subsets and generate sequentially connected sequences for the connected subsets in a total running time proportional to the number of nonzero elements of N and since N is graphic if and only if all the submatrices corresponding to maximal connected subsets of R are graphic.

Moreover, we adopt the following notations. For each $i = 0, 1, \cdots, k-1$,

$$S^*(r_i) = \{s \mid s \varepsilon S, r_i(s)=1\}, \tag{2.1}$$

$$\pi(r_i) = S^*(r_i) \cap (\bigcup_{j<i} S^*(r_j)), \tag{2.2}$$

$$\sigma(r_i) = S^*(r_i) - \pi(r_i), \tag{2.3}$$

$$U(r_i, r_j) = S^*(r_i) \cap \sigma(r_j) \quad (j < i). \tag{2.4}$$

Here, $S^*(r_i)$ is the set of tree-edges of the circuit r_i, $\pi(r_i)$ is the set of tree-edges, of the circuit r_i, contained in at least one circuit r_ℓ $(\ell < i)$, $\sigma(r_i)$ is the set of tree-edges, of the circuit r_i, contained in no circuits r_ℓ $(\ell < i)$, and $U(r_i, r_j)$ is the set of tree-edges, of the circuit r_i, contained in the circuit r_j but in no circuits r_ℓ $(\ell < j)$.

3. PQ-Graph

The PQ-tree data structure is proposed in [3] and effectively applied to several combinatorial problems. In the present paper, definitions and terminology concerned with PQ-trees almost follow [3], though we use the term "vertex" instead of "node" used in [3].

Given a universal set U, a PQ-tree over U is a rooted directed tree whose leaves are elements of U and whose nonleaf vertices are labeled either P or Q. A vertex labeled P is called a P-vertex and a vertex labeled Q a Q-vertex. Vertices v_i's adjacent to a vertex v are called children of v and v is called a parent of v_i's. The root has no parent and the leaves have no child. For each nonleaf vertex, admissible linear arrangements (or permutations) of the children are specified as follows:

(i) for a P-vertex, every linear arrangement of the children is admissible,

(ii) for a Q-vertex, only two linear arrangements defined on the children, one being the reversion of the other, are admissible.

If we choose an admissible linear arrangement of the children of each nonleaf vertex, then it induces, in a natural manner, a linear arrangement of the elements of the universal set U, the set of the leaves. A linear arrangement of the elements of U induced in such a way is called admissible for the PQ-tree. A PQ-tree thus represents a class of admissible linear arrangements of the elements of U efficiently.

Furthermore, given a subset W of U and a PQ-tree T over U, K. S. Booth and G. S. Lueker [3] provide an efficient method

(i) for determining whether there exists at least one linear arrangement which is admissible for the PQ-tree T and in which elements of W are consecutive and

(ii) (if such a linear arrangement exists) for constructing a new PQ-tree T' such that the set of all the linear arrangements admissible for T' coincides with that of all the linear arrangements which are admissible for T and in which the elements of W are consecutive.

The new PQ-tree T' is called the W-reduction of T and we say that T' is obtained by reducing T by W.

We propose a data structure, called a PQ-graph, based on PQ-trees, which provides a foundation for an efficient algorithm for solving the graph-realization problem. A PQ-graph G over an universal set U is a directed graph satisfying (A1)-(A4):

(A1) G consists of disjoint PQ-trees T_i $(i=0,1,\cdots,n)$ and arcs connecting distinct PQ-trees. Each element of U is a leaf of some PQ-tree.

(A2) The leaves of each PQ-tree T_i $(i=0,1,\cdots,n)$ are distinguished vertices called branching vertices, two distinguished vertices called heads and some elements of U except that T_0 does not contain heads. Two heads of a PQ-tree are always consecutive for any linear arrangements admissible for the PQ-tree.

(A3) Each head of a PQ-tree is adjacent to one and only one branching vertex in the other PQ-tree, while each branching vertex is adjacent from at least one head.

(A4) The directed graph obtained by shrinking every PQ-tree into a single vertex is acyclic. (Such a shrunk graph is denoted by $\hat{G}$ against the original PQ-graph G and the labels of the vertices of $\hat{G}$ are those of corresponding PQ-trees in G.)

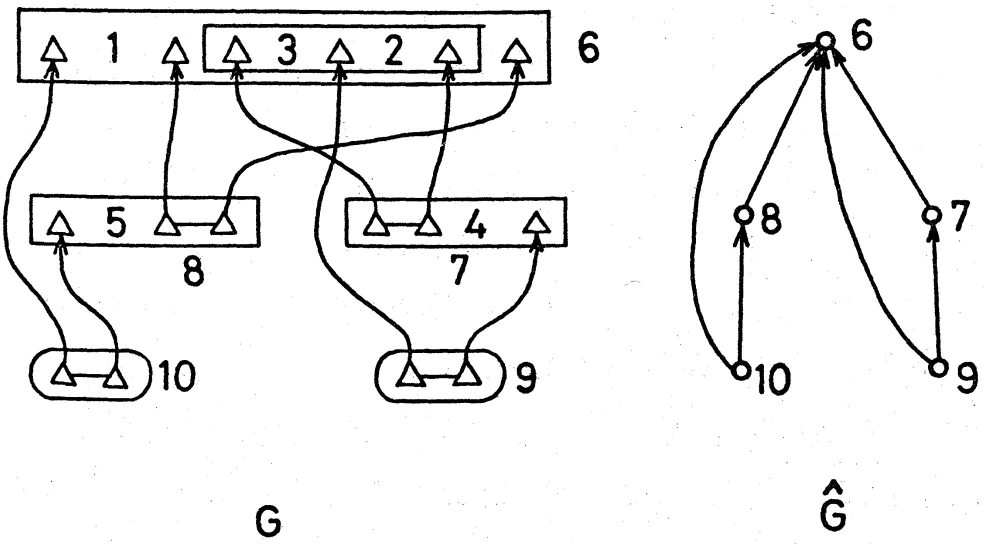


Fig. 2. A PQ-graph G and its shrunk graph $\hat{G}$, where parallel arcs are replaced by a single arc.

Figure 2 shows a PQ-graph G and the shrunk graph $\hat{G}$, where a P-vertex is denoted by ⊂⊃ and a Q-vertex by ▭ , and their children are written inside them. Also, a branching vertex is denoted by △ and a pair of heads by △—△. This way of representing PQ-graphs will be also adopted for the examples in Section 6.

We call a Q-vertex q of a PQ-tree a <u>neutral</u> Q-vertex if q has three children two of which are branching vertices at the both ends of q. (See Figure 3(a).) Also, we call a PQ-tree T a <u>two-terminal tree</u> if T has a root q being a Q-vertex, q has three children two of which are branching vertices at the both ends of q and one of which is a P-vertex p, and p has two children, the heads of T. (See Figure 3(b).)

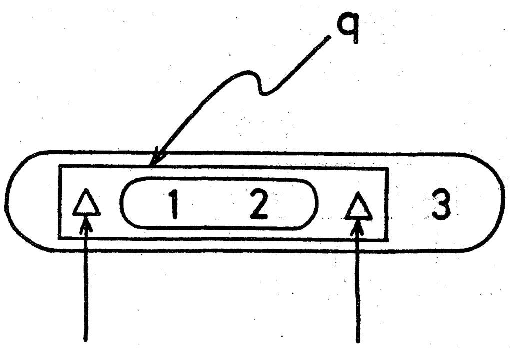


Fig. 3(a). A neutral Q-vertex q.

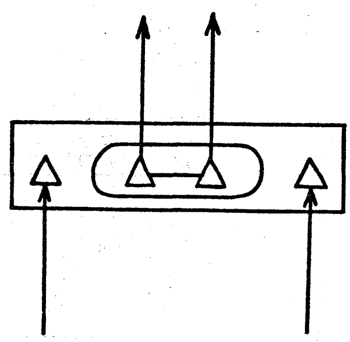


Fig. 3(b). A two-terminal tree.

For the sake of reducing the required running time, we adopt the following (B1)-(B3):

(B1) Branching vertices which are consecutive children of a Q-vertex are replaced by a single new branching vertex and all the arcs incident to those branching vertices are made incident to the new one.

(B2) For a branching vertex b and a head h which are consecutive children of a Q-vertex, the vertex b is removed and all the arcs incident to b are made incident to the branching vertex adjacent from the head h.

(B3) If a parent of a neutral Q-vertex is also a neutral Q-vertex, these neutral Q-vertices should be replaced by a single neutral Q-vertex.

(See Figure 4.)

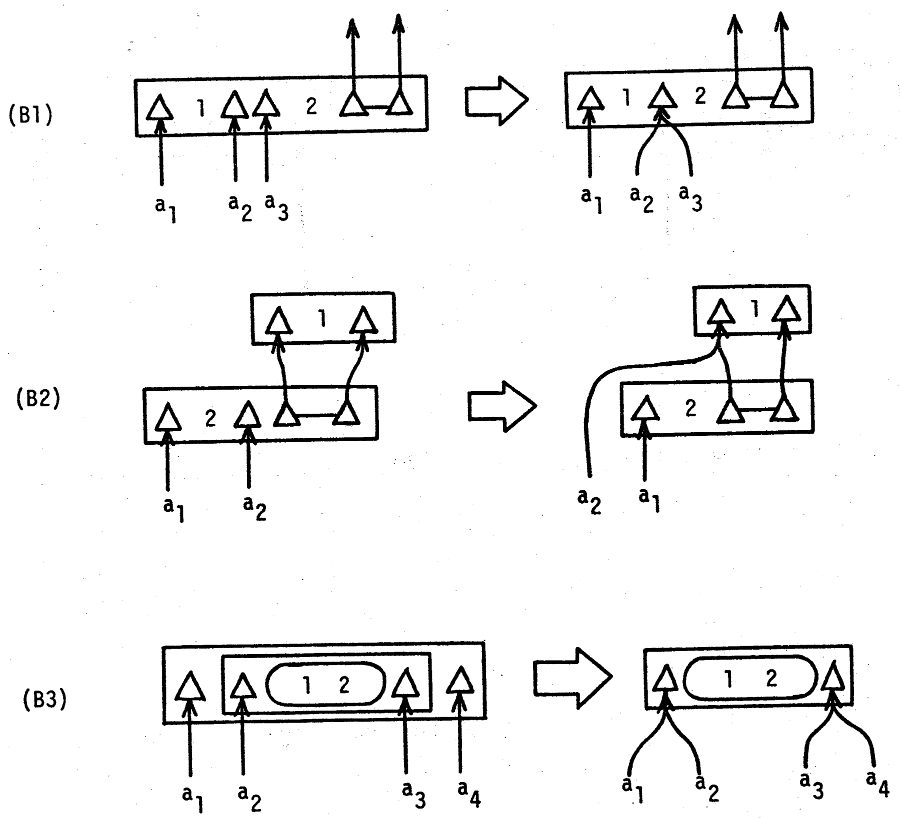


Fig. 4. Operations (B1), (B2) and (B3).

Moreover, let T be a PQ-tree in the PQ-graph G and W be a union of a subset of U in the PQ-tree T and a (possibly empty) set of branching vertices and heads in T. In this case, W-reduction T' of T is a new PQ-tree, linear arrangements admissible for which are exactly those which are admissible for the original T and in which the elements of W are consecutive, where any branching vertices not in W are allowed to be among the elements of W.

By the following (C1) and (C2), a PQ-graph G over U determines a class of graphs $\tilde{G}$ having a tree-edge set U, each corresponding to a choice of a set of admissible linear arrangements for the PQ-trees in G.

(C1) For each PQ-tree in G, carry out the following (I) and (II).

(I) Choose a linear arrangement (denote it by L) of the leaves which is admissible for the PQ-tree. If the linear arrangement L contains heads, then insert a new distinguished element between the heads in L. Then replace the consecutive branching vertices by a single branching vertex and, if a branching vertex and a head are consecutive, replace them by the head and, according to the replacement, make each arc (as a pointer) incident to a replaced branching vertex be incident to the replacing branching vertex or head. Denote the resultant linear arrangement by $\tilde{L}$. The $\tilde{L}$ can be expressed as a sequence

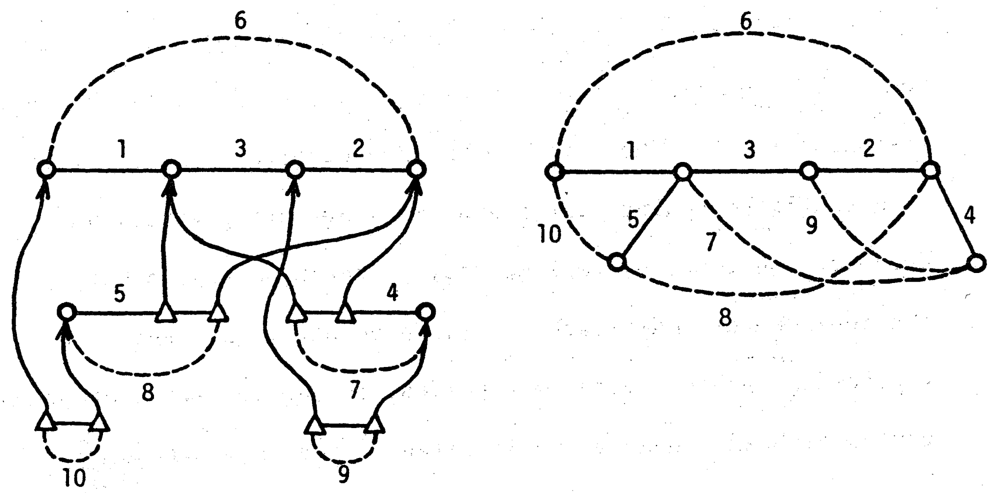
$$\tilde{L} = (v_0, a_1, v_1, a_2, \cdots, a_\ell, v_\ell), \tag{3.1}$$

where v_i $(i=0,1,\cdots,\ell)$ are branching vertices or a head, though some v_i's may be missing in $\tilde{L}$, and a_i $(i=1,2,\cdots,\ell)$ are elements of U in L or the new distinguished element between heads. If v_i is missing in $\tilde{L}$ of (3.1) for some $i = 0, 1,$

$\cdots,\ell$, then insert a new element as v_i.

(II) Construct a path which is represented by the sequence (3.1), where $\{v_0,v_1,\cdots,v_\ell\}$ is the set of vertices and $\{a_1,a_2,\cdots,a_\ell\}$ is the set of edges of the path. If the PQ-tree under consideration is not a two-terminal tree, then add an edge with the end-vertices v_0 and v_ℓ, which will be a cotree-edge, to the path.

(C2) By carrying out (C1), we have now obtained a graph consisting of (closed) paths, each corresponding to a PQ-tree in G, and arcs (or pointers) connecting distinct (closed) paths. Open all the edges which correspond to the distinguished edges between heads and short all the arcs (or pointers) connecting distinct (closed) paths.



(I) (II)

Fig. 5. (I): a graph obtained by applying operation (C1) to the PQ-graph G in Figure 2; and (II): a graph $\tilde{G}$ obtained by operation (C2).

The procedure of (C1) and (C2) applied to the PQ-graph shown in Figure 2 is illustrated in Figure 5. The (C1) and (C2) are carried out when it is required to construct graphs representing a given graphic (0,1)-matrix.

Every PQ-graph, appearing in the course of carrying out the algorithm presented in the next section, determines a class of graphs 2-isomorphic with one another.

4. Algorithm for the Graph-Realization Problem

In this section, we shall propose an efficient algorithm for solving the graph-realization problem.

4.1 An Outline of the Algorithm

For the sequentially connected sequence $(r_0, r_1, \cdots, r_{k-1})$ of fundamental circuits of N, we can easily see that

(i) if N is graphic, then for each $i = 0, 1, \cdots, k-1$ there exists a graph whose fundamental system of circuits are $(r_0, r_1, \cdots, r_i)$,

(ii) if there exists a graph $\tilde{G}_i$ whose fundamental system of circuits are $(r_0, r_1, \cdots, r_{i-1})$, then there exists a graph $\tilde{G}_{i+1}$ whose fundamental system of circuits are $(r_0, r_1, \cdots, r_i)$ if and only if there is a graph $\tilde{G}_i{}^*$ such that $\tilde{G}_i{}^*$ is 2-isomorphic with $\tilde{G}_i$ and the edges of $\pi(r_i)$ defined by (2.2) form an elementary path in $\tilde{G}_i{}^*$.

Therefore, we can consider a (not efficient) method for solving the graph-realization problem as follows.

1° Construct a graph $\tilde{G}_1$ composed of the circuit r_0, where the order of the edges of the circuit r_0 is arbitrary. Set $i \leftarrow 1$.

2° If $i = k$, then the algorithm terminates and N is graphic.

3° Find a graph $\tilde{G}_i^*$ which is 2-isomorphic with $\tilde{G}_i$ and in which the edges of $\pi(r_i)$ form an elementary path. If such a graph $\tilde{G}_i^*$ does not exist, then the algorithm terminates and N is not graphic.

4° Connect the end-vertices of the path, in $\tilde{G}_i^*$, formed by $\pi(r_i)$ to each other by an arbitrary elementary path formed by the edges of $\sigma(r_i)$ defined by (2.3) and the cotree-edge corresponding to the circuit r_i. Denote the resultant graph by $\tilde{G}_{i+1}$.

5° Put $i \leftarrow i+1$ and go back to Step 2°.

The correctness of the above method is clear but it seems difficult to carry out Step 3° efficiently without any sophisticated data structure.

In order to carry out Step 3° efficiently, we shall use PQ-graphs. A PQ-graph G_i expresses a class of graphs which are 2-isomorphic with the graph $\tilde{G}_i$ of Step 3°. The next PQ-graph G_{i+1} is efficiently constructed and expresses a class of graphs which are 2-isomorphic with the graph $\tilde{G}_{i+1}$ of Step 4°.

An efficient algorithm is given as follows.

Algorithm (An Outline)

1° Construct a PQ-tree labeled r_0 with a root which is a P-vertex and has the elements of $S^*(r_0)$, defined by (2.1), as its children.

Put

$\ell(r_0) \leftarrow 0$,

$G_1 \leftarrow$ a PQ-graph consisting of the PQ-tree r_0 alone,

$i \leftarrow 1$.

2° If $i = k$, then the algorithm terminates and N is graphic.

3°-1 (Finding a Fundamental Path)

In the shrunk graph $\hat{G}_i$ of the PQ-graph G_i, find a minimal fundamental path P_i traversing all the vertices which correspond to the PQ-trees containing elements of $S^*(r_i)$, where use is made of the labels ℓ and d^+ on PQ-trees. If such a fundamental path does not exist, then the algorithm terminates and N is not graphic; or else, determine the "type" of P_i. (There are possible eight types. See Figure 6.)

3°-2 (Reducing PQ-trees)

Reduce each PQ-tree corresponding to a vertex in P_i to obtain a PQ-graph G^* such that the graphs generated by G^* are exactly those generated by G_i in which the elements of $\pi(r_i)$ form the edge set of elementary paths. If such a reduction is impossible, then the algorithm terminates and N is not graphic; or else,

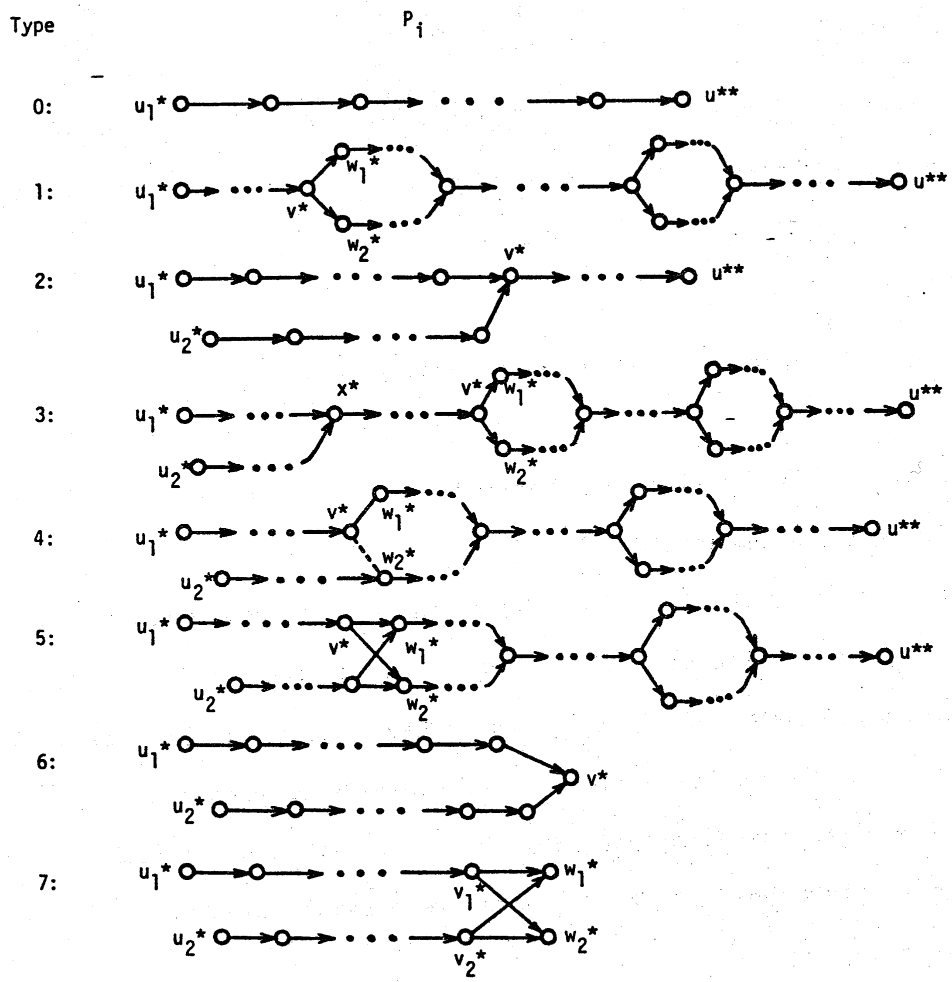


Fig. 6. Types of fundamental paths P_i determined in Step 3°-1.

insert branching vertices b_{i1}^* and b_{i2}^* and, if necessary, a neutral Q-vertex such that b_{i1}^* and b_{i2}^* become end-vertices of a path with the edge set $\pi(r_i)$ in every graph generated by G^*. (A new two-terminal tree may be introduced except for the case of types 0, 1 and 4.) Denote the resultant PQ-graph by G^* again.

4° Construct a PQ-tree, labeled r_i, such that its root is a P-vertex, the children of the root are the elements of $\sigma(r_i)$ and a P-vertex p, and the children of the P-vertex p are the heads h_{i1}^* and h_{i2}^*, where, if $\sigma(r_i)$ is empty, let the P-vertex p be the root of the PQ-tree r_i. Put

$G_{i+1} \leftarrow$ a PQ-graph consisting of the PQ-graph G^*, the PQ-tree r_i and the arcs (h_{ij}^*, b_{ij}^*) $(j=1,2)$,

$\ell(r_i) \leftarrow \max\{\ell(r_{i(j)}) + 1 \mid j=1,2\}$,

where, for each $j = 1, 2$, $r_{i(j)}$ is the PQ-tree which contains the branching vertex b_{ij}^*. Also, put

$$d^+(r_i) \leftarrow \begin{cases} 1 & \text{if } r_{i(1)} = r_{i(2)}, \\ -2 & \text{if } \underline{type} = 0 \text{ or } 1 \text{ and } r_{i(1)} \neq r_{i(2)}, \\ 2 & \text{otherwise.} \end{cases}$$

5° Set $i \leftarrow i+1$ and go back to Step 2°.

Steps 3°-1 and 3°-2 are the most involved part of the algorithm and the details are given in subsections 4.2 and 4.3, respectively. It may be helpful for reading subsections 4.2 and 4.3 if readers refer to examples in Section 6.

4.2 Finding a Fundamental Path

It should be noted that, for the label ℓ determined in Steps 1° and 4°, $\ell(r_i)$ is equal to the maximum number of arcs in directed paths from the vertex r_i to the vertex r_0 in $\hat{G}_i$, where arcs entering into two-terminal trees are not counted.

Let us define $d_i^+(u)$ for each PQ-tree u in $\hat{G}_i$ by

$$d_i^+(u) = \begin{cases} 1 & \text{if the vertex } u \text{ is adjacent to one and only one vertex in } \hat{G}_i, \\ -2 & \text{if } d^+(u) = -2 \text{ or if } d^+(u) = 1 \text{ and the vertex } u \text{ is adjacent to distinct vertices in } \hat{G}_i, \\ 2 & \text{otherwise.} \end{cases}$$

Note that $d_i^+(u) \neq d^+(u)$ only if $d^+(u) = 1$ and the rule (B2) described in Section 3 is applied. Also, note that we do not actually prepare the label d_i^+ since for each u the value of $d_i^+(u)$ is found in a constant running time if necessary. When $d_i^+(u) = 1$, $\Gamma^+(u)$ denotes the vertex, in $\hat{G}_i$, adjacent from u and, when $d_i^+(u) = -2$ or 2, $\Gamma_j^+(u)$ (j=1,2) denote the vertices, in $\hat{G}_i$, adjacent from u such that $\ell(\Gamma_1^+(u)) \geqq \ell(\Gamma_2^+(u))$.

Now, let K be the set of the PQ-trees in G_i containing the elements of $S^*(r_i)$. Note that one of the end-vertices of the fundamental path P_i should have the maximum value of ℓ among K.

We find, in Step 3°-1, the required fundamental path P_i in $\hat{G}_i$ by extending a directed path, starting from a degenerate path composed of an end-vertex of P_i alone, according to the following rules.

Let u be the terminal vertex of a temporarily constructed directed path P_i in $\hat{G}_i$ and ℓ^* be the minimum value of $\ell(v)$ among v's in K.

Rule 1: If $d_i^+(u) = 1$, then extend P_i from u to the vertex $\Gamma^+(u)$.

Rule 2: If $d_i^+(u) = -2$, then let $u_j = \Gamma_j^+(u)$ $(j=1,2)$ and let b_1 be the branching vertex, in the PQ-tree u_1, adjacent from a head of the PQ-tree u.

(2-1) If $\ell(u_1) = \ell^*$, then extend P_i from u to the vertex u_1,

(2-2) else, if the elements of a subset of $S^*(r_i)$ exists consecutively between a head of u_1 and the branching vertex b_1, then extend P_i from u to u_1,

(2-3) else, extend P_i from u to u_2.

Rule 3: If $d_i^+(u) = 2$, then for each $j = 1, 2$ let $u_j = \Gamma_j^+(u)$ and let b_j be the branching vertex, in the PQ-tree u_j, adjacent from a head of u.

(3-1) If neither u_1 nor u_2 is in K, then

if $\ell(u_2) \geqq \ell^*$,

then stop (the algorithm terminates and N is not graphic),

else, extend P_i from u to u_2,

(3-2) else, if either u_1 or u_2 is in K, then extend P_i from u to either u_1 or u_2 which contains an element of $S^*(r_i)$,

(3-3) else (both u_1 and u_2 are in K), then there are the following three possible cases (I)-(III), i.e., letting A_j $(j=1,2)$ be two propositions defined by

A_j = "the elements of $U(r_i, u_j)$ exist exactly between the branching vertex b_j and a head of u_j" $(j=1,2)$,

(I) neither A_1 nor A_2 is true,

(II) either A_1 or A_2 is true, and

(III) both A_1 and A_2 are true.

In case of

(I): stop (the algorithm terminates and N is not graphic),

(II): extend P_i from u to u_j, where j corresponds to the true proposition A_j,

(III): if $\pi(u) \neq \bigcup\{U(r_i,r) \mid r\varepsilon K, \ell(r) < \ell(u)\}$, then stop (the algorithm terminates and N is not graphic), else, extend P_i from u to both u_1 and u_2, connect u_1 with u_2 by the fundamental path P_p, where $u = r_p$ (but this process is not actually performed and the attachment of P_p is hypothetical only for P_i to form a complete fundamental path), and set $v^* \leftarrow u$ and $w_j^* \leftarrow u_j$ $(j=1,2)$.

4.3 Reducing PQ-Trees

In carrying out Step 3°-2, the points are the following (D1)-(D3).

(D1) We proceed from the PQ-tree, corresponding to the turning vertex of P_i, to those PQ-trees which have greater values of ℓ.

(D2) Suppose that, for the PQ-graph G_i, subsets V and W of the ground set U are, respectively, contained in PQ-trees v and w and that the elements of $V \cup W$ form a path in every graph generated by G_i. Then, there are essentially the following three cases (see Figure 7):

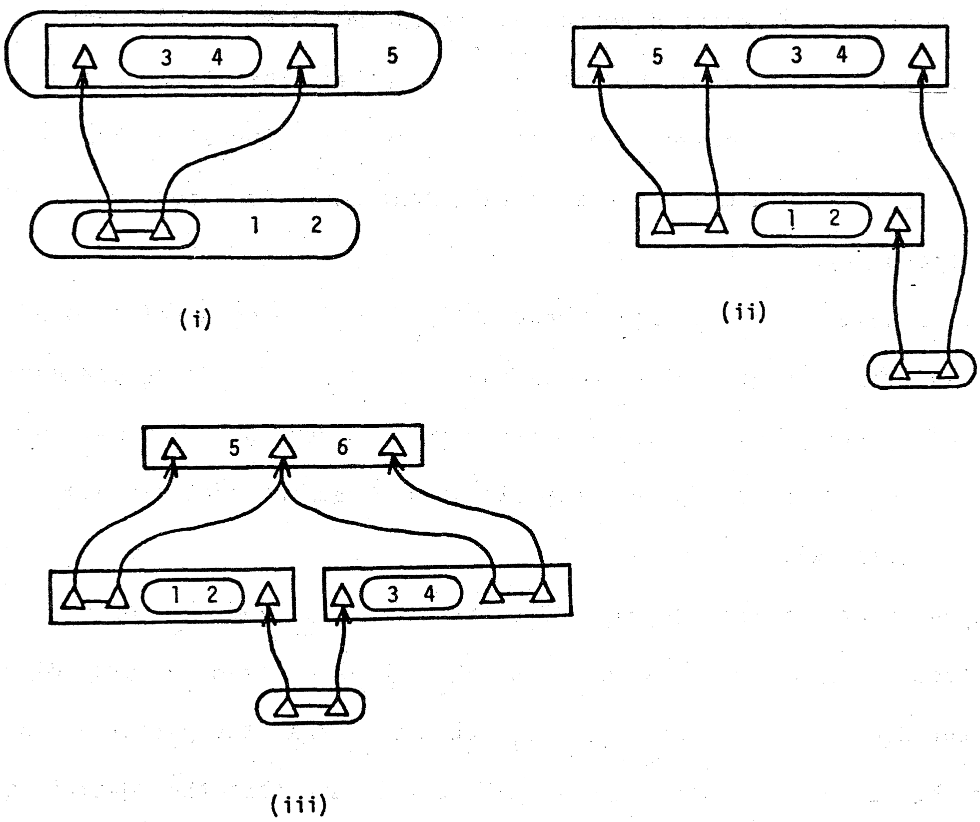


Fig. 7. Examples of cases (i), (ii) and (iii) of (D2), where $V = \{1,2\}$ and $W = \{3,4\}$.

(i) there are two branching vertices b_1 and b_2, in the PQ-tree w, adjacent from the heads of v such that b_1, b_2 and the elements of W are consecutive with b_1 and b_2 being at the both ends of W in every linear arrangement admissible for w and the elements of V and the pair of heads of v are consecutive in every linear arrangement admissible for v,

(ii) there is a branching vertex b, in the PQ-tree w, adjacent from a head h of v such that b (resp. h) and the elements of W (resp. V) are consecutive with b (resp. h) being at an end of W (resp. V) in every linear arrangement admissible for w (resp. v),

(iii) a head h of v and a head h* of w are adjacent to one and the same branching vertex and h (resp. h*) and the elements of V (resp. W) are consecutive with h (resp. h*) being at an end of V (resp. W) in every linear arrangement admissible for v (resp. w).

(D3) Suppose that, in the PQ-graph G_i, the heads h_1 and h_2 of a PQ-tree v and the heads h_1* and h_2* of a PQ-tree w are adjacent to the same branching vertices b_1 and b_2, that the parent of h_1 and h_2 (resp. h_1* and h_2*) is a P-vertex, and that the PQ-tree v (resp. w) contains elements of $S^*(r_i)$. Then, if the PQ-graph G_{i+1} is constructed, it includes a new two-terminal tree t such that the heads of t are adjacent to b_1 and b_2 and that h_1 and h_2 (resp. h_1* and h_2*) are adjacent to the branching vertices of t. (Cf. Example 3 in Section 6. This corresponds to the fact that two two-terminal subgraphs with a common set of attachment vertices merge into a single two-terminal subgraph by an addition of an edge connecting the two two-terminal subgraphs.)

It should, however, be noted that by (B3) of Section 3 we exclude such a PQ-tree that the parent of a neutral Q-vertex is also a neutral

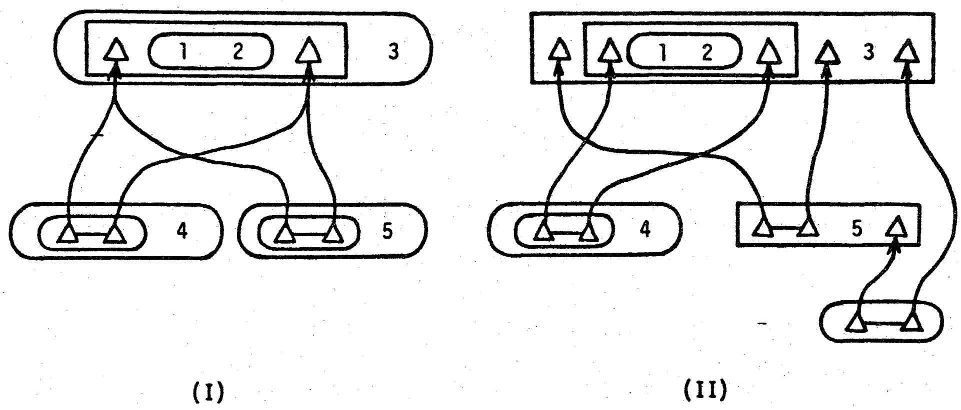


Fig. 8. A reduction (II) of a PQ-graph (I) such that the set of elements 3 and 4 forms a path.

Q-vertex. If there are not less than two PQ-trees, each of which has heads adjacent to the common two branching vertices of a neutral Q-vertex q of a PQ-tree r and if the neutral Q-vertex q collapses, or is absorbed into the other Q-vertex, by a reduction of the PQ-tree r such that one of the branching vertices of q should be at an end of some specified leaf subset of r, then before the reduction we presuppose that there is one more neutral Q-vertex q' which is the parent or a child of q and to which the heads of a PQ-tree adjacent to r and relevant to the present reduction of the PQ-graph is made adjacent and that q remains neutral after the reduction. (See Figure 8.)

5. Validity of the Algorithm and the Computational Complexity

We shall give two theorems, one for the validity of the algorithm and the other for the computational complexity of the algorithm. Their proofs will also be given. (However, they should be considered as sketches of the proofs.)

First, we show the following.

Theorem 5.1: Suppose that the PQ-graph G_i generates exactly all the graphs which have the fundamental circuit matrix N_i of rows r_0, r_1, $\cdots$, r_{i-1} $(1 \leq i < k)$ and that the matrix N_{i+1} of rows r_0, r_1, $\cdots$, r_i is graphic. Then the PQ-graph G_{i+1} is constructed from G_i by Steps 3° and 4° and generates exactly all the graphs which have N_{i+1} as the fundamental circuit matrix.

(Proof) First, we can easily show that all the graphs generated by the PQ-graph G_{i+1}, if it is constructed, have N_{i+1} as their fundamental circuit matrix.

Next, let $\tilde{G}_{i+1}$ be an arbitrary graph which has N_{i+1} as its fundamental circuit matrix. Here, because of the assumption, at least one such graph exists. Then, delete from $\tilde{G}_{i+1}$ the edges of $\sigma(r_i)$ and the cotree-edge r_i (and isolated vertices, if any), and denote the resultant graph by $\tilde{G}_i$*. Since the graph $\tilde{G}_i$* has N_i as its fundamental circuit matrix, there exists a set of linear arrangements admissible for the PQ-trees in G_i which correspondingly yields the graph $\tilde{G}_i$* by the procedure given by (C1) and (C2) in Section 3. Because of the existence of such admissible linear arrangements for PQ-trees in G_i, we can easily see that the PQ-graph G_{i+1} is constructed by Steps 3° and 4°. Furthermore, since the path of the edge set, given by the union of $\sigma(r_i)$ and the cotree-edge r_i, in $\tilde{G}_{i+1}$ can be generated by the PQ-tree r_i and since the heads of the PQ-tree r_i are adjacent to the branching vertices which correspond to the end-vertices of the path of the edge set $\pi(r_i)$ in $\tilde{G}_{i+1}$, the graph $\tilde{G}_{i+1}$ thus can be generated by the PQ-graph G_{i+1}. This completes the proof. Q.E.D.

Note that the PQ-graph G_1 constructed in Step 1° generates all the graphs which have the fundamental circuit matrix of the row r_0 alone. Hence, the correctness of the algorithm follows from Theorem 5.1.

Next, we show the computational complexity.

Theorem 5.2: A running time required for the algorithm is at most $O(\nu\alpha(\nu,k))$, where ν is the number of nonzero elements of the given (0,1)-matrix N, k is the number of rows of N and $\alpha(\cdot,\cdot)$ is a function defined by R. E. Tarjan [10] in terms of Akermann's function.

(Proof) Note that changing, according to (B1) and (B2), the incidence relation of the arcs which connect distinct PQ-trees in the PQ-graph G_i is equivalent to the operation of the union of disjoint subsets of the set of heads in G_i. We call such an operation a UNION. Furthermore, finding a branching vertex adjacent from a head is equivalent to finding the label of the subset which contains the head. We call such an operation a FIND. In Step 3°-1, if a vertex u (not a turning vertex) of P_i corresponds to a PQ-tree which contains no element of $S^*(r_i)$, then a UNION is applied to a head of u in Step 3°-2. Hence, the number of FINDs performed till the end of the algorithm is at most $O(\nu)$. Therefore, by employing the UNION-FIND algorithm in [10], the total running time required for the UNIONs and the FINDs is at most $O(\nu\alpha(\nu,k))$.

On the other hand, a total running time required for reducing PQ-trees is $O(\nu)$ (cf. [3]), where it should be noted that the number of occurrences of PQ-trees which are vertices of P_i's without containing any elements of $S^*(r_i)$ $(i=0,1,\cdots,k-1)$ is bounded by $3k$. This completes the proof. Q.E.D.

Since, for practical values of ν and k, $\alpha(\nu,k)$ is not more than 3 [10], we can solve the graph realization problem in a running time almost proportional to ν, the number of nonzero elements of the given (0,1)-matrix N.

6. Examples

In the following examples, given (0,1)-matrices are connected and the rows are ordered such that they are sequentially connected. Furthermore, all the given (0,1)-matrices are graphic and, for each example, PQ-graphs G_i's constructed in the course of carrying out the algorithm are shown together with a graph $\tilde{G}_k$ having the fundamental circuit matrix as specified. In Figures 9 - 11, an integer beside each PQ-tree denotes a label of the corresponding cotree-edge, and an integer with an asterisk a tree-edge contained in the next fundamental circuit.

Example 1 [6] For a (0,1)-matrix N given by

$$N = \begin{array}{c} \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array} ,$$

see Figure 9.

Example 2 For a (0,1)-matrix N given by

$$N = \begin{array}{c} \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array} ,$$

see Figure 10.

Example 3 For a (0,1)-matrix N given by

$$N = \begin{array}{c} \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} ,$$

see Figure 11. Note that a two-terminal tree appears in the PQ-graph G_5.

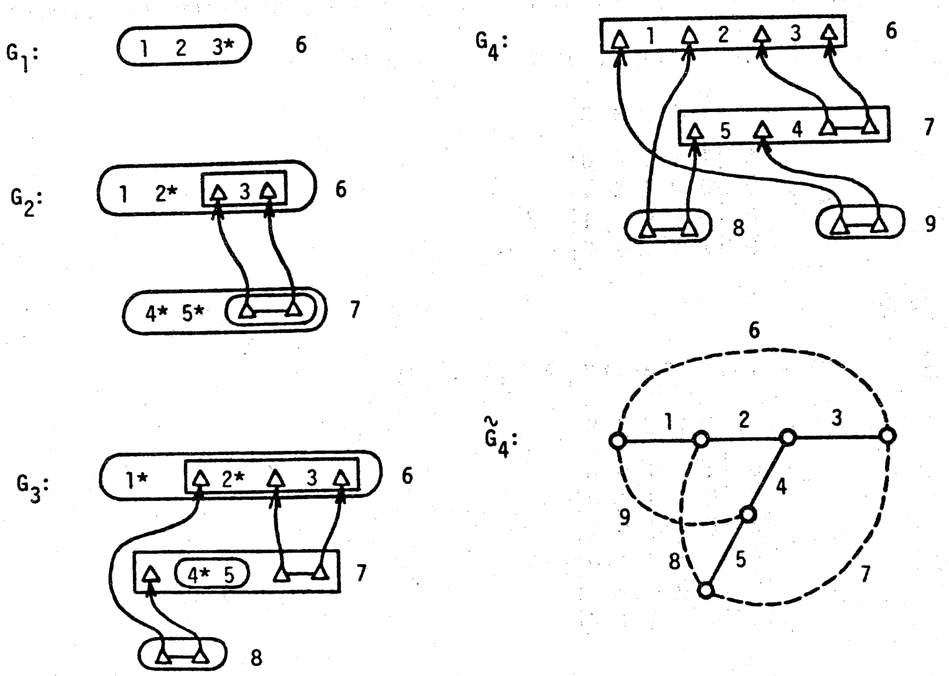


Fig. 9. Example 1.

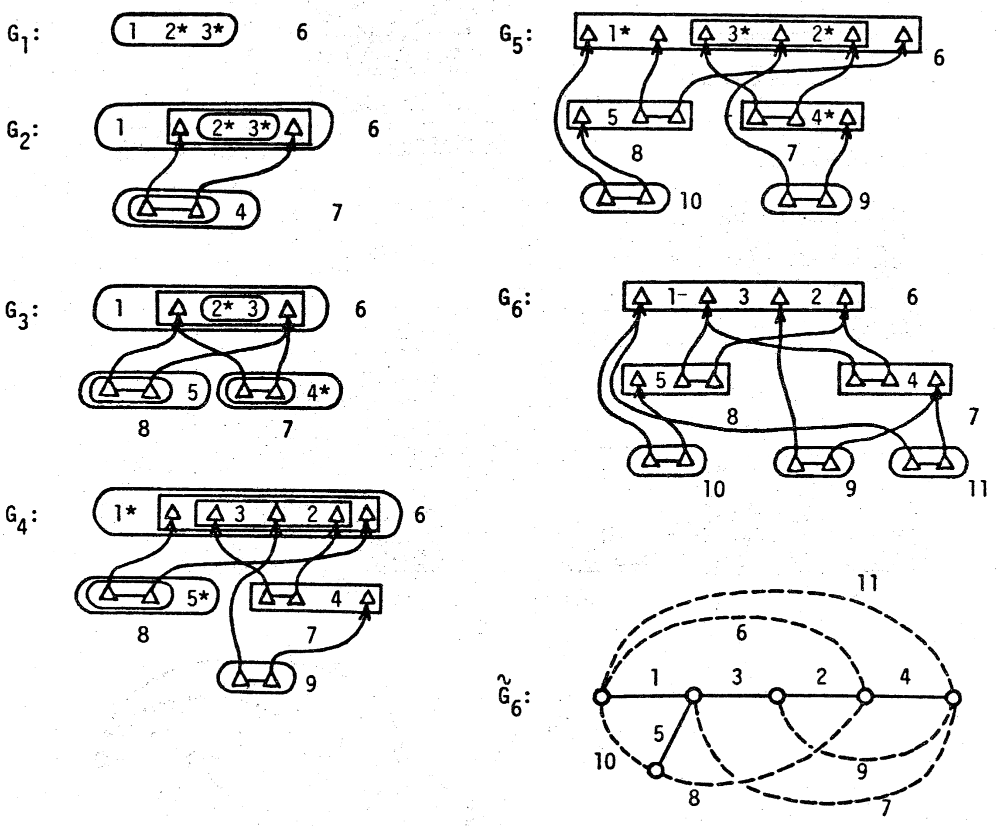


Fig. 10. Example 2.

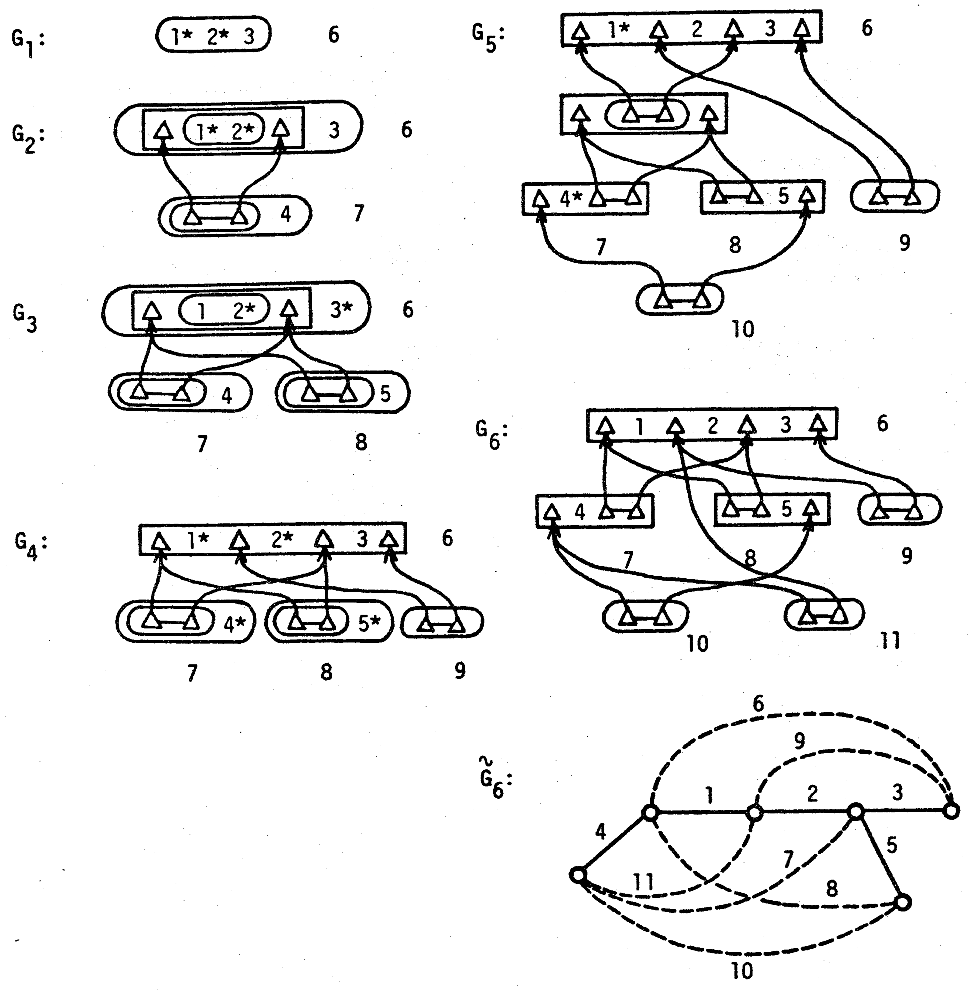


Fig. 11. Example 3.

7. Concluding Remarks

We proposed an efficient algorithm for solving the graph-realization problem by means of PQ-graphs. The algorithm requires a nunning time almost proportional to the number of nonzero elements of a given (0,1)-matrix. The problem of determining whether or not there exists a linear-time solution algorithm is left open.

Finally, it should be noted that, when a (0,1)-matrix N is graphic, the finally obtained PQ-graph G_k expresses the structure of the set of the two-terminal subgraphs [14] of graphs which have N as the fundamental circuit matrix and that we can easily determine such a structure from the PQ-graph G_k.

Acknowledgments: The author is deeply indebted to Professor Masao Iri of the University of Tokyo who made useful discussions on the present paper and directed the author's attention to PQ-trees the use of which significantly simplified the original version of the algorithm.

References

1. A. V. Aho, J. E. Hopcroft and J. D. Ullman, "The Design and Analysis of Computer Algorithms," Addison-Wesley, Reading, Massachusetts, 1974.

2. R. E. Bixby and W. H. Cunningham, Converting linear programs to network problems, to appear in Mathematics of Operations Research.

3. K. S. Booth and G. S. Lueker, Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms, Journal of Computer and System Sciences, 13 (1976), 335-379.

4. S. Fujishige, An efficient algorithm for solving the graph-realization problem by means of PQ-trees, Proceedings of the 1979 International Symposium on Circuits and Systems, 1012-1015, Tokyo, July 1979.

5. R. Gould, Graphs and vector spaces, Journal of Mathematics and Physics 37 (1958), 193-214.

6. M. Iri, A necessary and sufficient condition for a matrix to be the loop or cut-set matrix of a graph and a practical method for the topological synthesis of networks, RAAG Research Notes, Third Series, No. 50 (1962); On the synthesis of loop and cutset matrices and the related problems, RAAG Memoirs 5-A (1868), 4-38.

7. M. Iri, "Network Flow, Transportation and Scheduling: Theory and Algorithm," Academic Press, New York, N. Y. 1969.

8. W. Mayeda, "Graph Theory," John Wiley & Sons, New York, N. Y., 1972.

9. S. Seshu and M. B. Reed, "Linear Graphs and Electrical Networks," Addison-Wesley, Reading, Massachusetts, 1961.

10. R. E. Tarjan, Efficiency of a good but not linear set union algorithm, Journal of the Association for Computing Machinery 22 (1975), 215-225.

11. N. Tomizawa, An $O(m^3)$ algorithm for solving the realization problem of graphs — on combinatorial characterizations of graphic (0,1)-matrices, Papers of the Technical Group on Circuit and System Theory, The Institute of Electronics and Communication Engineers of Japan, CST75-106 (1976) (in Japanese).

12. W. T. Tutte, An algorithm for determining whether a given binary matroid is graphic, Proceedings of the American Mathematical Society 11 (1960), 905-917.

13. W. T. Tutte, From matrices to graphs, Canadian Journal of Mathematics 16 (1964), 108-128.

14. W. T. Tutte, "Connectivity in Graphs," University of Toronto Press, London, 1966.