アルゴリズム　１７－７
（1990.　9.28）

２連結グラフに対する最適な耐故障性路線割当

和田 幸一[†]　羅 予頻[‡]　川口 喜三男[†]

[†]名古屋工業大学　[‡]御幸毛織

あらまし：　通信網や計算機網における情報伝達の信頼性をいかに経済的に効率よく高めるかは重要な問題である．通常，通信網（計算機網）は局（計算機）を点，通信回線を辺とした無向グラフで表される．通信網における任意の二点間の通信は無向グラフの通路を用いて行われる．通信網の二点間の通信経路を決定することを路線割当（routing）と呼ぶが，路線割当が前もって計算されている通信網の信頼性の尺度として，通信網を表す有向グラフとその路線割当及び，故障点と故障辺によって定義されるＳＲ－グラフ（surviving route graph）と呼ばれる有向グラフの直径を評価しようという提案がなされている．ＳＲ－グラフは，通信網を表した無向グラフの故障していない点からなり，二点間の通信経路が故障していないとき，その二点間に有向辺をもつ有向グラフとして定義される．本稿では，すべての２連結グラフに対して，ＳＲ－グラフの直径が２となる路線割当が定義できることを示す．通信網において故障数が０でない限りＳＲ－グラフの直径は２以上になるので，本稿で示した路線割当は最適になる．また，その路線割当が$O(n^2)$で計算できることを示す。ここで，nは２連結グラフの点数である．

Optimal Fault-tolerant Routing for 2-connected Graphs

Koichi Wada[†], Yupin Luo[‡] and Kimio Kawaguchi[†]

[†]Nagoya Institute of Technology
Gokiso-cho, Syowa-ku, Nagoya 466, JAPAN
[‡]Miyuki Keori Co. Ltd.
Nagoya 451, JAPAN

Abstract: The diameter of the surviving route graph $R(G,\rho)/F$ for a graph G, a routing ρ and a set of faults F (denoted by $D(R(G,\rho)/F)$), where two nonfaulty nodes x and y are connected by a directed edge iff there are no faults on the route from x to y, could be one of the fault-tolerance measures for the graph G and the routing ρ.

In this paper, we show that we construct a routing ρ for any n-node 2-connected graph G such that $D(R(G,\rho)/\{f\}) \leq 2$ for any fault f in time $O(n^2)$.

1 Introduction

Consider a communication network or an undirected graph G in which a limited number of link and/or node faults F might occur. A *routing* ρ for a graph defines at most one path called *route* for each ordered pair of nodes. We assume that it must be chosen without knowing which components might be faulty.

Given a graph G, a routing ρ and a set of faults F, the *surviving route graph* $R(G,\rho)/F$ is defined to be a directed graph consisting of all nonfaulty nodes in G, with a directed edge from a node x to a node y iff the route from x to y is intact. The diameter of the surviving route graph could be one of the fault-tolerance measures for the graph G and the routing ρ [1, 2]. Many results have been obtained for the diameter of the surviving route graph [3, 5, 6, 11, 13].

If the diameter of the surviving route graph is two, the routing is said to be *optimal.* Because as long as faults are assumed to occur in a network, the diameter of the surviving route graph is more than one. It is shown that an optimal routing can be constructed for any n-node k-connected graph such that $n \geq 7k^3\lceil log_2 n\rceil$ [6]. However, it is an open question whether or not we can construct an optimal routing for all k-connected graphs for fixed k.

In this paper, we show that an optimal routing can be constructed for all 2-connected graphs. Imase *et al.* have shown an optimal routing for all 2-connected graphs [5]. However, Our routing is completely different from theirs, and their routing is optimal for only node faults. Our routing shown here is not only optimal for node faults and edge faults but also simpler than their routing. We also show that our routing can be constructed in $O(n^2)$ time, where n is the number of nodes in the graph.

2 Preliminary

In this section, we give definitions and terminology. We refer readers to [4] for basic graph terminology.

Unless otherwise stated, we deal with an undirected graph $G = (V, E)$ that corresponds to a network. For a node v of G, $N_G(v) = \{u|(v, u) \in E\}$. A graph G is *k-connected* if there exist k node-disjoint paths between every pair of distinct nodes in G. For a node v of G and a node set $V' \subseteq V - \{v\}$ of G, *(v, V')-fan* is a set of $|V'|$ disjoint paths from v to all nodes of V'. The *distance* between nodes x and y in G is the length of the shortest path between x and y and is denoted by $dis_G(x, y)$. The *diameter* of G is the maximum of $dis_G(x, y)$ over all pairs of nodes in G.

A *tree* T is an undirected graph that is connected and acyclic. A *rooted tree* is a tree T with a distinguished node r, called the *root.* Let v be a node in a rooted tree T which is not the root, and let $(v_0 = r, v_1, \ldots, v_p = v)$ be the simple path from r to v in T. Then, v_i is the *parent* of v_{i+1} and v_{i+1} is a *child* of $v_i (0 \leq i \leq p-1)$. A node with no children is a *leaf.*

Let $G = (V, E)$ be a graph and let x and y be nodes of G. Define $P_G(x, y)$ to be the set of all simple paths from the node x to the node y in G, and $P(G)$ to be the set of all simple paths in G. A *routing* is a partial function $\rho : V \times V \to P(G)$ such that $\rho(x, y) \in P_G(x, y)(x \neq y)$. The path specified to be $\rho(x, y)$ is called the *route from x to y.*

For a graph $G = (V, E)$, let $F \subseteq V \cup E$ be a set of nodes and edges called a set of *faults.* We call $F \cap V (= F_V)$ and $F \cap E (= F_E)$ the set of *node faults* and the set of *edge faults*, respectively. If an object such as route or path does not contain any element of F, the object is said to be *fault free.*

For a graph $G = (V, E)$, a routing ρ on G and a set of faults $F (= F_V \cup F_E)$, the *surviving route graph*, $R(G, \rho)/F$, is a directed graph with node set $V - F_V$ and edge set $E(G, \rho, F) = \{< x, y > |\rho(x, y) is\ defined\ and\ fault\ free\}$.

A routing ρ is a *minimal* routing if $|\rho(x, y)| = dis_G(x, y)$ for any ordered pair (x, y) in the domain of ρ. A routing ρ is a *bidirectional* routing if $\rho(x, y) = \rho(y, x)$ for any node pair (x, y) in the domain of ρ. Note that if the routing ρ is bidirectional, the surviving route graph $R(G, \rho)/F$ can be represented as an undirected graph and $dis_{R(G,\rho)/F}(x, y) = dis_{R(G,\rho)/F}(y, x)$ for any pair of distinct nodes x and y.

Let $\rho(x, y)\rho(y, z)$ denote the route $\rho(x, y)$ from x to y followed by the route $\rho(y, z)$ from y to z. We call $\rho(x_0, x_1)\rho(x_1, x_2)\ldots\rho(x_{p-1}, x_p)$ a *route sequence of length p from x_0 to x_p.* Note that every route sequence is a path and not necessarily a simple one.

Let G be a graph and ρ be a routing on G. For an ordered pair (x, y) of distinct nodes x and y in G, *(x, y) is $(d, f)-$tolerant with respect to ρ* if for every set of F of f faults in G, $dis_{R(G,\rho)/F}(x, y) \leq d$. If the routing ρ is apparent in the context, we simply say (x, y) is $(d, f)-$tolerant. A *routing ρ on a graph G is (d,f)-tolerant* if (x, y) is $(d, f)-$tolerant for any ordered pair of distinct nodes x and y. A *graph G is said to be $(d, f)-$tolerant* if there exists a $(d, f)-$tolerant routing ρ on G.

Lemma 1 *Let ρ be a routing on a graph G and let (x, y) be any ordered pair of distinct nodes in G. If there exist $f + 1$ node-disjoint route sequences of length less than or equal to d for x and y, then (x, y) is $(d, f)-$tolerant.*

3 Optimal Routing for 2-connected graphs

A 2-connected graph with the fewest edges is a cycle graph $C_n = (\{0, 1, \ldots, n-1\}, \{(i, (i+1) \bmod n)|0 \leq i \leq n-1\})$. It is easily shown that if n is even, then we can not construct any minimal and $(2, 1)$-tolerant routing on C_n.

Theorem 1 [8] *There exists a 2-connected graph for which minimal and $(2, 1)$-tolerant routing can not be defined.*

Therefore, we must find a non-minimal and $(2, 1)$-tolerant routing to define for all 2-connected graphs.

Let $G = (V, E)$ be a 2-connected graph. Let $\prec$ denote a totally ordered relation on $V \times V$. Let define $H(v) = \{u|v \prec u \text{ and } (v, u) \in E\}$ and $L(v) = \{u|u \prec v \text{ and } (v, u) \in E\}$. For $U \subseteq V$, $min_\prec U$ and $max_\prec U$ denote the minimum and maximum element in U with respect to $\prec$.

If a totally ordered relation $\prec$ on $V \times V$ satisfies the following conditions, it is said to be *cyclic*.

1. There exists an edge $(v_H, v_L) \in E$ such that $H(v_H) = \phi$ and $L(v_L) = \phi$, that is there is an edge between $max_{\prec}V$ and $min_{\prec}V$.
2. For any node v except v_H, $H(v) \neq \phi$ and for any node v except v_L, $L(v) \neq \phi$.

Example 1 Let $C_n = (V, E)$ be a cycle graph. We can define a cyclic totally ordered relation on $V \times V$ as follows. Let $(v_1, v_2, \ldots, v_{n-1}, v_n, v_1)$ be the cycle. The specific nodes v_L and v_H in the cyclic relation are defined to be v_1 and v_n. A relation $\prec$ is defined to be $\{v_{i-1} \prec v_i | 2 \leq i \leq n\}$. The transitive closure of the relation $\prec$ is a totally ordered relation and satisfies the conditions of the cyclic relation. If a 2-connected graph G contains a Hamiltonian cycle, we can similarly define a cyclic relation for G. Although every 2-connected graph does not contain a Hamiltonian cycle, we will show in the next section that a cyclic relation can be defined for every 2-connected graph.

For a node v in G and a cyclic relation $\prec$, we define two paths $P_H[v, v_H] = (v_0(= v), v_1, \ldots, v_k(= v_H))$, where $v \neq v_H$ and $v_i = max_{\prec}N_G(v_{i-1})(1 \leq i \leq k)$ and $P_L[v, v_L] = (v_0(= v), v_1, \ldots, v_k(= v_L))$, where $v \neq v_L$ and $v_i = min_{\prec}N_G(v_{i-1})(1 \leq i \leq k)$. From the definition of the cyclic relation, two paths $P_H[v, v_H]$ and $P_L[v, v_L]$ are well defined and have the following property, which can be derived from the definition.

Lemma 2 *Let $G = (V, E)$ be a 2-connected graph. For any node $x(\neq v_H, v_L)$, $P_H[x, v_H]$ and $P_L[x, v_L]$ are node-disjoint.*

We construct an optimal routing ρ for a 2-connected graph on which a cyclic relation is defined.
routing ρ

1. $\rho(v_H, v_L) = \rho(v_L, v_H) = (v_H, v_L)$.
2. For $x(\neq v_H, v_L)$,
   $\rho(x, v_H) = \rho(v_H, x) = P_H[x, v_H]$ and
   $\rho(x, v_L) = \rho(v_L, x) = P_L[x, v_L]$.
3. For x and y such that $x \neq v_H, v_L$, $y \neq v_H, v_L$ and $x \prec y$,
   $\rho(x, y) = \rho(y, x) = P_L[x, v_L](v_L, v_H)P_H[y, v_H]$.

The routing ρ is a bidirectional but non-minimal routing. Figure 1 shows the routing ρ for the cycle graph C_n with the cyclic relation defined in Example 1.

Lemma 3 *The routing ρ is $(2, 1)$-tolerant.*

Proof Let f be any fault in G and $R = R(G, \rho)/\{f\}$. Let x and y be any pair of distinct nonfaulty nodes in G.

(1) Suppose that $x = v_L$ and $y = v_H$. If $f \neq (v_L, v_H)$ then $dis_R(x, y) = 1$. Otherwise for any node $z(\neq x, y)$, the route sequence $\rho(x, z)\rho(z, y) = P_L[z, x]P_H[z, y]$ can not contain f. Thus, $dis_R(x, y) \leq 2$.

(2) Suppose that $x \neq v_L$ and $y = v_H$. There are two node-disjoint route sequences $\rho(x, v_H) = P_H[x, v_H]$

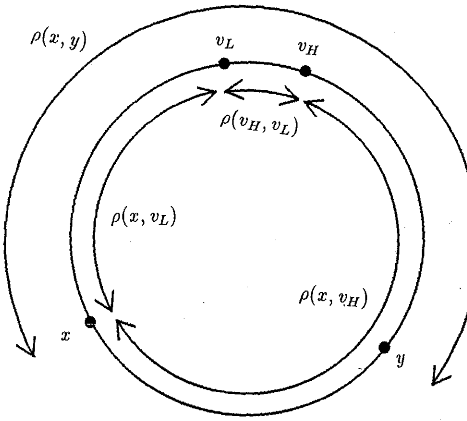


Figure 1: The routing ρ for the cycle graph C_n.

and $\rho(x, v_L)\rho(v_L, v_H) = P_L[x, v_L](v_L, v_H)$, because of Lemma 2. Thus, $dis_R(x, y) \leq 2$ from Lemma 1.

(3) For the case that $x = v_L$ and $y \neq v_H$, it can be proved similar to the case (2).

(4) Suppose that $x \neq v_L$ and $y \neq v_H$. We can assume that $x \prec y$. If $f = (v_L, v_H)$ or ($f \in V$ and $f \prec x$) or ($f = (f_1, f_2) \in E$ and $min_{\prec}\{f_1, f_2\} \prec x$), then the route sequence $\rho(x, v_H)\rho(v_H, y)$ is fault free. If ($f \in V$ and $y \prec f$) or ($f = (f_1, f_2) \in E$ and $max_{\prec}\{f_1, f_2\} \prec y$), then the route sequence $\rho(x, v_L)\rho(v_L, y)$ is fault free. Otherwise, the route $\rho(x, y) = P_L[x, v_L](v_L, v_H)P_H[y, v_H]$ is obviously fault free. Thus $dis_R(x, y) \leq 2$.

Therefore, the routing ρ is $(2, 1)$-tolerant. □

4 Algorithms for the Cyclic Relation

We show two algorithms to compute a cyclic relation for every 2-connected graph. One is constructed by using node-disjoint paths in G and is called NDP, and the other utilizes the property of a block-cutnode graph [4] and is called BCG.

Given any 2-connected graph $G = (V, E)$, let $n = |V|$ and $m = |E|$. Since a linear time algorithms have been shown to find a spanning 2-connected subgraph $G' = (V, E')$ for G such that $|E'| \leq 2n - 6$ [10, 12], we can assume that $m \leq 2n - 6(m = O(n))$.

4.1 Algorithm NDP

The algorithm NDP is shown in Figure 2. In the algorithm NDP, we define specific nodes v_H and v_L. In the first execution of the while loop, $(x, \{v_H, v_L\})$-fan and the edge (v_H, v_L) form a cycle and a cyclic relation is defined on nodes included in the $(x, \{v_H, v_L\})$-fan (Figure 3(a)). In the later execution of the while loop, the relation is augmented for nodes in the paths $(x = a_0, a_1, \ldots, a_{i_a})$ and $(x = b_0, b_1, \ldots, b_{i_b})$ to reserve the conditions of the cyclic

Algorithm NDP

```
Input a 2-connected graph $G = (V, E)$.
Outout a cyclic relation on $V \times V$.
Algorithm
(1)Let $(u, v)$ be any edge of $G$;
(2)$v_H \leftarrow v$; $v_L \leftarrow u$;
(3)Set $v_L \prec v_H$;
(4)$V_0 \leftarrow \{v, u\}$;
(5)while $V - V_0 \neq \phi$ do
(6)    begin
(7)       Let $x$ be any node in $V - V_0$;
(8-1)     $(x, \{v_H, v_L\})$-fan is denoted by
(8-2)     $(x = a_0, a_1, \ldots, a_{p-1}, a_p = x_H)$ and
(8-3)     $(x = b_0, b_1, \ldots, b_{q-1}, b_q = x_L)$.
(9)       Let $i_a$ be the smallest index $i$ such that $a_i \in V_0$.
(10)      Similarly $i_b$ is also defined.
(11-1)    Without loss of generality, $b_{i_b} \prec^+ a_{i_a}$, where
(11-2)    $\prec^+$ denotes the transitive closure of $\prec$.
(12-1)    Set $x' \prec b_{i_b-1} \prec \ldots$
                      $\prec b_1 \prec x \prec a_1 \prec \ldots \prec a_{i_a}$,
(12-2)    where $x'$ denotes the largest node which is
             smaller than $a_{i_a}$ in $V_0$ with respect to $\prec^+$.
(13)      $V_0 \leftarrow V_0 \cup \{x, a_1, \ldots, a_{i_a-1}, b_1, \ldots, b_{i_b-1}\}$
(14)    end
(15)Output the transitive closure of $\prec$.
```

Figure 2: Algorithm NDP

relation(Figure 3(b)).

We show an example of execution of NDP for a 2-connected graph in Figure 4.1.

The next lemma is easily proved by induction on the number of iterations of the while loop.

Lemma 4 *Before the execution of line (5) of the algorithm NDP, the transitive closure of the relation $\prec$ on $V_0 \times V_0$ satisfies the conditions of the cyclic relation.*

Lemma 5 *Let $G = (V, E)$ be any 2-connected graph. The algorithm NDP computes a cyclic relation on $V \times V$ in time $O(n^{2.5})$.*

Proof The correctness of NDP is immediate from Lemma 4.

In the while loop of NDP(lines (5)-(15)), each line except (8) can be computed in time $O(n)$. Since $(x, \{v_H, v_L\})$-fan is computed in time $O(n^{0.5}m)$ [9] and we can assume that $m = O(n)$, line (8) can be computed in time $O(n^{1.5})$. The iteration of the while loop is at most n, lines (1)-(4) take $O(1)$ time and line (15) can be computed in time $O(n)$. Thus the running time of the algorithm NDP is $O(n^{2.5})$. □

4.2 Algorithm BCG

A *cutnode* of a graph is one whose removal increases the number of components and a *bridge* is such an edge. A *block* of a graph is a bridge or a maximal 2-connected subgraph.

For a connected graph G with blocks $\{B_i\}$ and cutnodes $\{c_j\}$, the *block-cutnode graph* of G,denoted by $bc(G)$, is defined as the graph having node set $V_b \cup V_c$ and edge set E_{bc}:

$$\begin{aligned} V_b &= \{B_i | B_i \text{ is a block but not a bridge }\}, \\ V_c &= \{c_j | c_j \text{ is a cutnode or } deg(C_j) = 1\} \text{ and} \\ E_{bc} &= \{(B_i, c_j) | c_j \in V(B_i), c_j \in V_c\} \\ &\quad \cup \{(c_j, c_{j'}) | (c_j, c_{j'}) \text{ is a bridge }\}, \end{aligned}$$

where $V(B_i)$ denotes the node set of the block B_i.

A node in V_b is called a *block node*. For a block node in $bc(G)$, $B(v)$ denotes the block in G. For a node v of G and a node v' of $bc(G)$, if one of the following conditions holds *v is said to correspond to v'*.

(1) $v = v'(v' \in V_c)$

(2) $v \in V(v') - V_c(v' \in V_b)$

Figure 5 shows an example of a graph G and its block-cutnode graph.

Every block-cutnode graph is a tree [4]. If G is a 2-connected graph, the following property holds.

Lemma 6 *Let $G = (V, E)$ be a 2-connected graph and let z be any node in V. Let v be any leaf in the block-cutnode graph $bc(G-z) = (V_b \cup V_c, E_{bc})$, If $v \in V_c$, then $v \in N_G(z)$ and if $v \in V_b$, then there is a node u in the block $B(v)$ such that $u \in N_G(z)$ and u is not a cutnode of G.*

The algorithm BCG is shown in Figure 6. The algorithm BCG is based on a topological ordering [7] on the directed acyclic graph constructed by regarding the edges of a rooted tree $bc(G - z)$ as directed from parent to child and adding directed edges $\{(v', z) | v \in N_G(z)$, v corresponds to $v'\}$ to the directed rooted tree $bc(G - z)$.

We show an example of execution of BCG for a 2-connected graph in Figure 7. If $bc(G - v_L)$ has no block nodes ($V_b = \phi$), it is obvious that BCG computes a cyclic relation correctly by using Lemma 6. It can be shown by induction on the number of nodes in G for the general case.

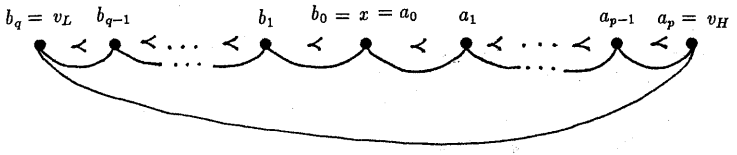
Lemma 7 *Let $G = (V, E)$ be any 2-connected graph. The algorithm BCG computes a cyclic relation on $V \times V$ in time $O(n^2)$.*

Proof In the algorithm BCG, the procedure orderblock is called at most n. Each orderblock$(H = (V', E'), r, z)$ can be computed in time $O(|E'|)$, because the block-cutnode graph is constructed in time $O(|E'|)$[9] and preorder of the nodes in T is computed in time $O(k)$. Since $m = O(n)$, BCG can be computed in time $O(n^2)$. □

The main theorem follows from Lemma 3,5 and 7.

Theorem 2 *Any 2-connected graph is $(2, 1)$-tolerant.*

Acknowledgement This research was partly supported by a Scientific Research Grant-In-Aid from the Ministry of Education, Science and Culture, Japan, and by the Okawa Institute of Information and Telecommunication.



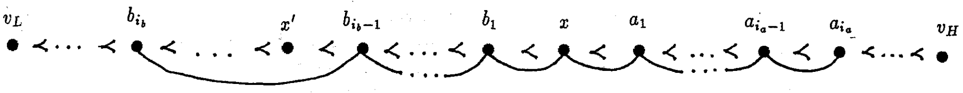
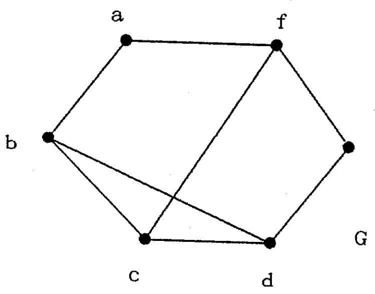


Figure 3: An Construction of a cyclic relation.



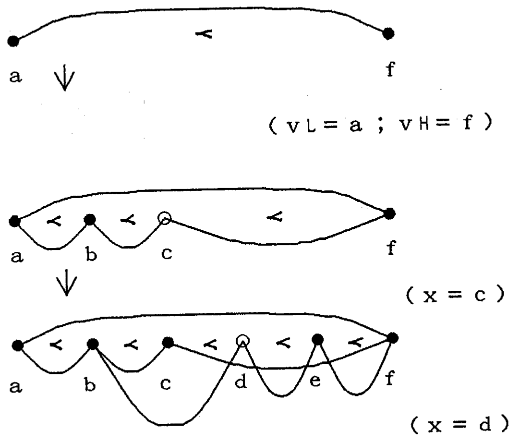


Figure 4: An example of execution of NDP.

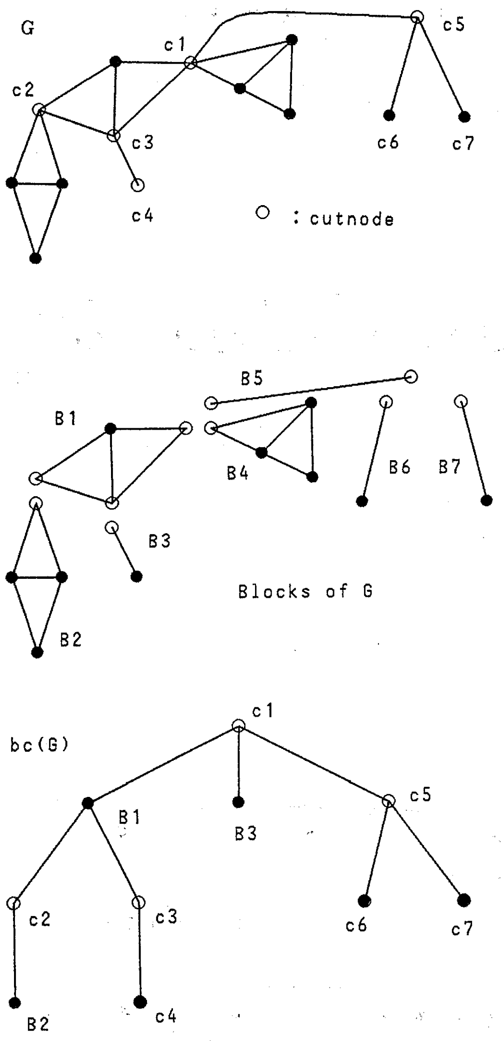
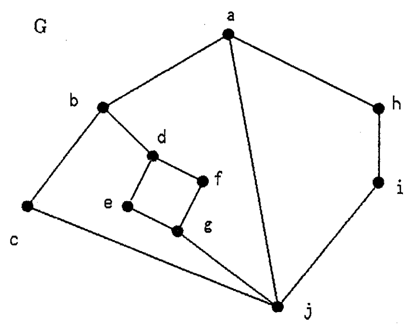


Figure 5: An example of a block-cutnode graph

Algorithm BCG

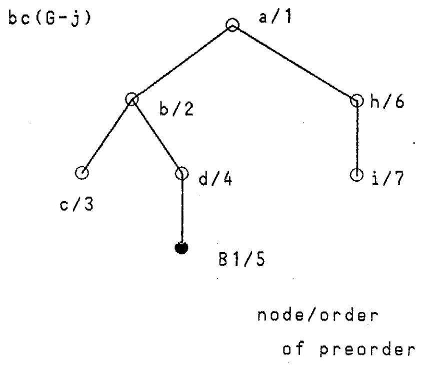
```
Input a 2-connected graph $G = (V, E)$.
Output a cyclic relation $\prec$ on $V \times V$.
Algorithm
(1)for each node $v$ in $V$ do $\ell(v) := undefined$;
(2)Let $(z_1, z_2)$ be any edge in $E$;
(3)$v_H \leftarrow z_1$; $v_L \leftarrow z_2$;
(4)$i \leftarrow 1$;
(5)orderblock($G, v_H, v_L$);
(6)Set the relation $\prec$ to the totally ordered relation
   determined by the labels $\ell(v)(v \in V)$.


(7)procedure orderblock($H, r, z$);
(8)begin
(9)    $T \leftarrow bc(H - z) = (V_b \cup V_c, E_{bc})$;
(10)   Let $T$ be a rooted tree with the node to which
          $r$ corresponds as a root;
(11)   Let $v_1, \ldots, v_k$ be nodes in $T$ ordered by preorder;
(12)   for $j := 1$ to $k$ do
(13)       if $v_j \in V_b$ then
               orderblock($B(v_j), par(v_j), ch(v_j)$)
(14)           { where par($v_j$) is the parent of $v_j$, if $v_j$ is
               not the root otherwise par($v_j$)=r and ch($v_j$)
               is a child of $v_j$, if $v_j$ is not a leaf otherwise
               ch($v_j$) $\in N_H(z) \cup V(v_j)$ }.
(15)       else { $v_j \in V_c$ }
(16)           if $\ell(v_j) = undefined$ then setorder($v_j, i$)
(17)   setorder(z,i);
(18)end;

(19)procedure setorder(v, var i);
(20)begin
(21)   $\ell(v) \leftarrow i$;
(22)   $i \leftarrow i + 1$
(23)end;
```

Figure 6: Algorithm BCG.



VH= a ; VL= j {lines (2)-(3)}

orderblock(G,a,j) {line (5)}

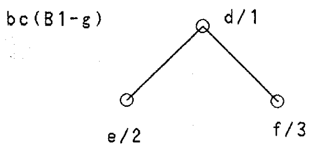


{lines (9)-(11)}

$\mathcal{L}(a) = 1$; $\mathcal{L}(b) = 2$;

$\mathcal{L}(c) = 3$; $\mathcal{L}(d) = 4$ {line (16)}

orderblock(B1,d,g) {line (13)}



{lines (9)-(11)}

$\mathcal{L}(e) = 5$; $\mathcal{L}(f) = 6$ {line (16)}

$\mathcal{L}(g) = 7$ {line (17)}

$\mathcal{L}(h) = 8$; $\mathcal{L}(i) = 9$ {line (16)}

$\mathcal{L}(j) = 10$ {line (17)}

Figure 7: An example of execution of BCG.

References

[1] A.Broder, D.Dolev, M.Fischer and B.Simons: "Efficient fault tolerant routing in network," *Information and Computation*75,52–64(1987).

[2] D.Dolev, J.Halpern , B.Simons and H.Strong: "A new look at fault tolerant routing," *Information and Computation*72,180–196(1987).

[3] P.Feldman: "Fault tolerance of minimal path routing in a network,*in Proceedings, 17th ACM STOC*,pp.327–334(1985).

[4] F.Harary, Graph theory, *Addison-Wesley*, Reading, MA(1969).

[5] M.Imase and Y.Manabe: "Fault tolerant routings in a k-connected network," *Paper of Technical Group*, COMP86-70,IECE(1987)(in Japanese).

[6] K.Kawaguchi and K.Wada: "New results in graph routing," to appear in *Information and Computation*(1990).

[7] D.E.Knuth: The art of computer programming, Vol. 1: Fundamental algorithms, 2nd ed., *Addison-Wesley*, Reading, MA(1973).

[8] Y.Luo, K.Wada and K.Kawaguchi: "A construction of networks with highly fault-tolerant routings," *Trans. IEICE*, J70-A,11,1620–1631(1987)(in Japanese).

[9] K.Mehlhorn: Data structures and algorithm 2: Graph algorithms and NP-completeness, *Springer*(1984).

[10] H.Nagamochi and T.Ibaraki: "k-edge-connected and k-node-connected spanning subgraphs," *Paper of Technical Group*, AL-10,IPSJ(1989).

[11] D.Peleg and B.Simons: "On fault tolerant routing in general graph," *Information and Computation*74,33–49(1987).

[12] H.Suzuki, N.Takahashi, T.Nishizeki, H.Miyano and S.Ueno: "An algorithm for tripartitioning 3-connected graphs," *J. IPSJ*31,5,584–592(1990)(in Japanese).

[13] K.Wada and K.Kawaguchi: Efficient fault-tolerant fixed routings on $(k+1)$-connected digraphs," to appear in *Discrete Applied Math.*(1990).