이중연결성분 재구성 분산알고리즘

이 창 석† · 박 정 호†† · 구 연 설†††

요 약

본 논문에서는 2연결성분 재구성문제를 해결하는 분산알고리즘을 제안한다. 2연결성분 재구성문제란 2연결성분이 이미 구성되어 있는 네트워크상에서 네트워크의 토폴로지 변화가 생겼을 때 토폴로지 변화에 따라 2연결성분을 재구성하는 문제이다. 본 논문에서는 복수개의 프로세서와 링크의 추가와 삭제에 대해 2연결성분 재구성문제를 해결하는 메시지복잡도 $O(n' + a + b)$), 이상시간복잡도 $O(n')$의 분산 알고리즘을 제안한다. 여기서 n'는 토폴로지 변화후의 네트워크의 프로세서수, a는 추가 링크수를 나타낸다. 또 b는 삭제 링크를 포함하는 (토폴로지 변화전의) 2연결성분에 포함된 링크수의 합계를 나타낸다.

An Algorithm Solving the Biconnected-components Reconstruction Problem

Chang-Suk Lee† · Jung-Ho Park†† · Yeon-Seol Koo†††

ABSTRACT

This paper considers the Biconnected-components Reconstruction Problem(BRP), that is, the problem to reconstruct the biconnected-components in response to topology change of the network. This paper proposes a distributed algorithm that solves the BRP after several processors and links are added and deleted. Its message complexity and its ideal-time complexity are $O(n'+a+b)$ and $O(n')$ respectively, where n' is the number of processors in the network after the topology change, a is the number of added links, and b is the total number of links in the biconnected components (of the network before the topology change) including the deleted links.

1. 서 론

This paper considers distributed algorithms operating on a network of processors connected by communication links. On such a network, it is important to compute the biconnected components since the biconnected components are relevant to reliability of the network. Several distributed algorithms for computing the biconnected components have been proposed[5, 6, 7, 9]. These previous works consider the problem for computing the biconnected components from scratch. In a real network, however, topology of a network often changes because of addition(e.g. recoveries) and deletion(e.g. failures) of processors and links. This makes it important to study distributed algorithms for Reconstruction problems, that is the problems to reconstruct solutions(e.g. the bicon-

※ 본 연구는 정보통신부에서 시행하는 대학기초연구 지원사업에서 일부를 지원 받았음.
† 종신회원 : 농협전문대학 전산과 교수
†† 정 회 원 : 선문대학교 정보과학부 교수
††† 정 회 원 : 충북대학교 컴퓨터과학과 교수
논문접수 : 1997년 2월 20일, 심사완료 : 1997년 5월 26일

nected components) in response to the topology change. Some distributed algorithms for the reconstruction problem have been proposed[8].

This paper considers the Biconnected-components Reconstruction Problem(BRP), that is, the problem to reconstruct the biconnected components in response to topology change. It is obvious that the BRP can be solved by the known distributed algorithms that compute the biconnected components from scratch. However, it is a natural assumption that each processor knows the old solution, that is, each processor initially knows the biconnected components of the old network(i.e. the network before the topology change). The information available at each processor is not necessarily restricted to the old solution. More generally, we can assume that each processor has some auxiliary information about the old network. This raises a question : How efficiently can the BRP be solved using such an auxiliary information ? This is an interesting subject of study.

This paper proposes a distributed algorithm for the BRP after addition and deletion of several processors and links. The solution of the BRP is the labels assigned to all links : each link is labeled so that the links with the same label form a biconnected component, and each processor has to know the label assigned to each of its incident links. Since the algorithm uses a spanning tree of the old network as auxiliary information, it also reconstructs the spanning tree in response to the topology change. Its message complexity and its ideal time complexity are $O(n' + a + b)$ and $O(n')$ respectively, where n' is the number of processors in the new network(i.e. the network after the topology change), a is the number of added links, and b is the total number of links in the biconnected components(of the old network) including the deleted links. Since the length of every message is $O(\log n')$, its bit complexity is $O((n' + a + b)\log n')$. The (total) space complexity is $O(e\log n + e'\log n')$, where n(resp. e) is the number of processors(resp. links) in the old network, and e' is the number of links in the new network. Note that we assume topology change does not occur during the execution of the algorithm.

Swaminathan and Goldman[8] present a distributed algorithm for the BRP in a different environment. They consider a completely connected network, that is, each processor can directly communicate with any other processor by sending and receiving messages. They consider a logical structure of the processors on the completely connected network. The logical structure changes due to addition and deletion of logical links between processors, and the BRP they considered is a problem to maintain the biconnected components of the logical structure. Since the network is completely connected, their algorithm allows the processors to communicate with any other processors directly regardless of the topology of the logical structure. This assumption makes it possible to deal with topology change during the execution of the algorithm. Its message complexity and ideal time complexity are respectively $O(c + m')$ and $O(m')$ for deletion of one link, and respectively $O(c' + m')$ and $O(m')$ for addition of one link, where c(resp. c') is the number of the processors in the biconnected component containing the deleted link(resp. the added link), and m' is the number of the biconnected components in the new network. In the case of addition and deletion of k links, the message complexity and the ideal time complexity become k times as large as those for one link. In their definition, the solution of the BRP is the sets of processor IDs : each of the sets consists of the IDs of the processors contained in the same biconnected component, and each processor computes the ID sets for all biconnected components containing it. This requires the high bit complexity and the high space complexity.

2. Preliminaries

2.1 Graphs

An (undirected) graph G is a pair (V, E), where V is a finite set of vertices and E is a set of

edges(i.e. unordered pairs of distinct vertices in V). We use standard definitions [3] for a neighbor, a path, a cycle, a connected graph, a spanning tree, etc. This paper considers only connected graphs.

A biconnected component is a maximal set of edges such that any two edges in the set lie on a common cycle. Notice that each edge is contained in exactly one biconnected component.

2.2 Distributed system model

A network N is a pair (P, L), where P is a finite set of processors and L is a set of (communication) links (i.e. unordered pairs of distinct processors in P). From the definition, we can consider a network as a graph, and thus we use graph terminologies and notations for networks.

Our model is standard one, that is, (A1) through (A4) are assumed. (See [5, 11] for more details).

(A1) All processors execute the same program. The program consists of (a) internal operations within a processor, (b) send operations to send messages to its neighbors, and (c) receive operations to receive messages from its neighbors.

(A2) Every link is a bidirectional link and the processors can communicate directly with its neighbors by sending and receiving messages along the links.

(A3) Every link is completely fault-free and works as a FIFO queue.

(A4) The network is completely asynchronous, that is, there is no bound on message delay, clock drift, or the time necessary to execute a step. Thus, there is no timing assumption.

2.3 Biconnected-component Reconstruction Problem (BRP)

A network configuration is a global state of the entire network. We simply call it a configuration.

We say that the biconnected components of a network N are already determined in a configuration c, if every biconnected component is uniquely labeled and every processor knows, for each incident link, the label of the biconnected component containing the link. Recall that every link is contained in exactly one biconnected component.

The Biconnected-component Reconstruction Problem (BRP) is the problem to recompute the biconnected-components after change of the network topology. We consider topology change due to addition and deletion of several processors and links. Throughout this paper, the network before the topology change (resp. after the topology change) is called an old network(resp. a new network) and denoted by N=(P, L) (resp. N′=(P′, L′)).

More precisely, the BRP is the problem to reach the following final configuration from the following initial configuration.

- The initial configuration : the biconnected components of the old network N are already determined, and processors incident to the added or deleted links know which incident links are added or deleted.

- The final configuration : the biconnected components of the new network N′ are already determined.

In order to solve the BRP efficiently, we can make use of some auxiliary information. If we use some auxiliary information, the auxiliary information must be also updated so as to correspond to the new network N′. The algorithm proposed in this paper makes use of a spanning tree of N, and also recomputes a spanning tree of N′. Notice that links of the spanning tree of N may be deleted in the topology change.

Throughout this paper, we use the following notations.

- $N=(P, L)$: the old network(before topology change).
- $B_1, B_2, \cdots, B_m$: the biconnected components of N. m is the number of the biconnected components in N.
- T : a spanning tree of N available at the initial configuration. For convenience, we consider a spanning tree as a set of links.
- $T_1, T_2, \cdots, T_m$: $T_i = T \cap B_i$ for each $i(1 \leq i \leq m)$. Notice that T_i is a spanning tree of $N_i = (P_i, B_i)$ where $P_i = \{u \in P \mid (u,v) \in B_i\}$.
- $N' = (P', L')$: the new network (after topology change).
- $B_1', B_2', \cdots, B_m'$: the biconnected components of N'. m' is the number of the biconnected components in N'.
- P_a(resp. P_d) : the set of the added processors(resp. deleted processors). It holds that $P_a \cap P_d = \emptyset$ and $P' = P \cup P_a - P_d$.
- L_a(resp. L_d) : the set of the added links(resp. deleted links). It holds that $L_a \cap L_d = \emptyset$ and $L' = L \cup L_a - L_d$.

In this paper, the BRP is considered under the following assumptions.

(A5) Both the old and the new networks are connected networks.

(A6) The topology change does not occur during execution of a distributed algorithm.

(A7) There exists exactly one initiator (i.e. the processor spontaneously) starts execution of a distributed algorithm). Each of other processors starts the algorithm on receipt of a message.

2.4 Measures of efficiency

In this paper, we use the following efficiency measures of a distributed algorithm.

- Message complexity : The (worst case) message complexity is the maximum total number of messages transmitted during any execution of the algorithm.

- Bit complexity : The (worst case) bit complexity is the maximum total number of bits transmitted during any execution of the algorithm.

- Ideal time complexity : The (worst case) ideal time complexity is the maximum number of time units from start to the completion of the algorithm. In estimation of the ideal time complexity, we assume that the propagation delay of every link is at most one time unit. Notice that this assumption is used only for purpose of evaluation of the ideal time complexity.

- Space complexity : The (worst case) space complexity is the maximum total amount of storage of all processors in the whole network.

3. Algorithm for BRP

In this section, we present an algorithm for updating the biconnected components after topology change.

3.1 Properties of the biconnected components

Park et al. [7] proposed a distributed algorithm for computing the biconnected-components. By applying the algorithm to the new network N', we can solve the BRP with the message complexity $O(e')$ and the ideal time complexity $O(n')$, where n'(resp. e') is the number of processors(resp. links) in N'. The algorithm is based on a distributed depth first search algorithm in [6]. The depth first search algorithm checks all links in the network, and this requires the message complexity to be $O(e')$.

In the BRP, however, the following lemma obviously holds. The lemma implies that it is is not necessary to check all links in the network, and that there is possibility of reducing the message comple-

xity of the BRP.

Lemma 1 If there is no deleted link in a biconnected component B of the old network, B is included in a biconnected component of the new network. ■

In what follows, for a biconnected component B of the old network, we call it an injured biconnected component if it contains a deleted link(i.e. $B \cap L_d \neq \phi$), and call it an uninjured biconnected component if it contains no deleted link(i.e. $B \cap L_d = \phi$).

If follows from Lemma 1 that it is not necessary to check the biconnectivity of the links in an uninjured biconnected component. However, the links amy be included in a larger biconnected component of the new network, and we have to check the uninjured biconnected component to some extent. The following lemma implies that we have only to check the links of a spanning tree of the uninjured biconnected component.

lemma 2 Let $N^* = (P', L^*)$ be a network defined as follows :

$L^* = \bigcup_{1 \le i \le m} A_i \cup L_a - L_d$, where $A_i = T_i$(if $B_i \cap L_d = \phi$), and $A_i = B_i$(if $B_i \cap L_d \neq \phi$).

Let $u(\in P')$ be an arbitrary processor, and $L'(u)$ be the set of all links incident to u in N'(i.e. $L'(u) = \{(u,v) \in L' \mid v \in P'\}$). Let R_u be the minimum equivalence relation on the link set $L'(u)$ that satifies the following three conditions :

(a) If both f and g(where $\{f, g\} \subseteq L'(u) \cap L$) are links of the same uninjured biconnected component of the old network N, then fR_ug holds.

(b) If both f and g(where $\{f, g\} \subseteq L'(u) \cap L^*$) are links of the same biconnected component of N^*, then fR_ug holds.

(c) If both of fR_ug and gR_uh hold, then fR_uh holds.

Then, for any links of f and g in $L'(u)$, f and g are links of the same biconnected component of N' if and only if fR_ug holds.

(proof) ◇ *If part* : If links f and g are those of the same uninjured biconnected component, it follows from Lemma 1 that f and g are links of the same biconnected component of N'. It is also clear that f and g are links of the same biconnected component of N', if f and g are links of the same biconnected component of N^*. Therefore, f and g are links of the same biconnected component of N' if fR_ug holds.

◇ *Only if part* : We prove the only if part by induction. For the induction, we define a network $N^{*j} = (P', L^{*j})$ for each $j(0 \le j \le m)$ as follows :

$L^{*j} = \bigcup_{1 \le i \le m} A_i^j \cup L_a - L_d$,

where $A_i^j = T_i$ (if $B_i \cap L_d = \phi$ and $i \le j$), and $A_i^j = B_i$ (if $B_i \cap L_d \neq \phi$ or $i > j$).

From the definition, $N^{*,0} = N'$ holds. We can construct N^{*j} from N^{*j-1} by replacing B_j with T_j if B_j is an uninjured biconnected component. It holds $N^{*j} = N^{*j-1}$ if B_j is an injured biconnected component. It is clear that $N^{*,m} = N^*$.

We also define an equivalence relation R_u^j on $L'(u)$ by the similar way to the definition of R_u. The only difference is the condition (b) in the definition : we consider the biconnected components of N^{*j} instead of those of N^*.

By the induction on $j(0 \le j \le m)$, we can prove that fR_u^jg holds if f and g are links of the same biconnected component of N'.

○ *Inductive basis* : For j = 0, it follows from $N^{*,0} = N'$ that fR_u^0g holds if f and g are links of the same biconnected component of N'.

○ *Inductive assumption* : Assume that fR_u^kg holds if f and g are links of the same biconnected component of N'.

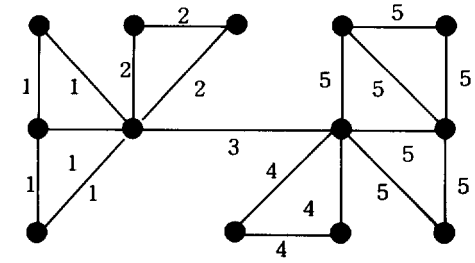
○ *Inductive step* : It is sufficient to show the

claim1 : $f R_u^{k+1} g$ holds if $f R_u^{k} g$ holds. If $N^{*,k+1} = N^{*,k}$, it is clear that the claim holds. In the followings, we prove the claim for the case of $N^{*,k+1} \neq N^{*,k}$ by contradiction.

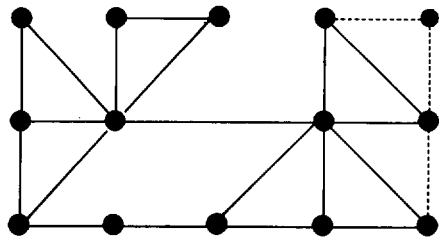
If the claim does not hold, we can show that there exist links f = (u,v) and g = (u,w) in $N^{*,k+1}$ such that f and g are links of a common cycle of $N^{*,k}$ but $f R_u^{k+1} g$ does not hold. Consider the simple v-w path p obtained from the cycle by removing f and g. Since $N^{*,k+1}$ is obtained from $N^{*,k}$ by replacing B_{k+1} with T_{k+1} and the path p does not exist in $N^{*,k+1}$, p contains a link in $B_{k+1} - T_{k+1}$. By replacing the links in $B_{k+1} - T_{k+1}$ with the paths in T_{k+1} and removing cycles if created, we can obtain a simple v-w path p' in $N^{*,k+1}$. If the path p' does not contain u, f and g are links of a common cycle in $N^{*,k+1}$ and $f R_u^{k+1} g$ holds. It is a contradiction. If the path p' contains u, we can show that there exist links f' = (u,v') and g' = (u,w') in T_{k+1} such that f = f' or f and f' are links of a common cycle in $N^{*,k+1}$ and such that g = g' or g and g' are links of a common cycle in $N^{*,k+1}$. In any case, $f R_u^{k+1} f'$, $f' R_u^{k+1} g'$, and $g' R_u^{k+1} g$ hold, and thus $f R_u^{k+1} g$ holds. It is a contradiction.■

Lemma 2 considers the biconnectivity relation only on the links incident to a common processor u. However, since the links of the same biconnected component form a connected subnetwork, the above local biconnectivity relation uniquely defines the biconnected components of the entire network.

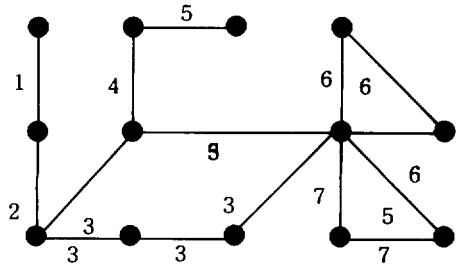
Example Figure 1(a) shows an old network N and its biconnected components. The links with the same number form a biconnected component of N. Figure 1(b) shows a new network N' after the topology change : one processor and three links are added, and one processor and three links are added, and one processor and three links are deleted. Figure 1(c) shows the network N^* defined in Lemma2, and its biconnected components. Figure 1(d) shows the biconnected components of N'.



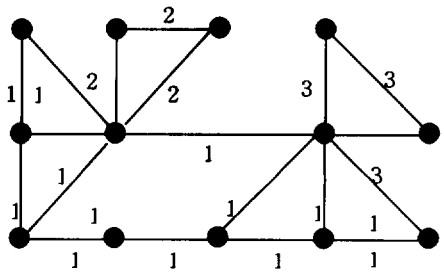
(a) an old network N



(b) a new network N



(c) the network N^* defined in Lemma 2



(d) the biconnected components of N'

(Fig. 1) Network N^*

Remark that we can obtain the biconnected components of N′ from those of N and N^* by following Lemma2. ■

3.2 Outline of the algorithm

The algorithm proposed in this paper utilizes Lemma 2 to solve the BRP. Thus, the algorithm consists of the following three phases.

P1 Construction of the network N^* as defined in Lemma 2 : every processor in the new network N′ determines which incident links are those of N^*.

P2 Computation of the biconnected components of N^* : every processor assigns a label to each of its incident links in N^* so that the links with the same label form a biconnected component of N^*.

P3 Computation of the biconnected components of the new network N′ : Based on Lemma 2, every processor assigns a label to each of its incident links in N′ so that the links with the same label form a biconnected component of N′.

3.3 Description of the algorithm

In this subsection, we present the algorithm for the BRP.

3.3.1 The first phase

In the first phase, the network N^* defined in Lemma 2 is constructed, that is, every processor in the new network N′ determines which incident links are those of N^*.

To construct N^* efficiently, the algorithm performs the depth-first search of N^* during the construction of N^*. When the initiator starts execution of the algorithm or a processor is first visited by the depth-first search, the processor (say u) determines the incident links of N^* as follows. After determining the incident links of N^*, u makes the depth-first search proceed(i.e. u sends the depth-first search token to one of its neighbors).

- For each added link (u,v), u determines that (u,v) is a link of N^*.
- For each link (u,v) of the old network N, u determines that (u,v) is a link of N^*, if (u,v) is a link an injured biconnected component or (u,v) is a tree link of the spanning tree T. If (u,v) is a link of an uninjured biconnected component and (u,v) is a non-tree link of T, u determines that (u,v) is not a link of N^*.

For each link (u,v) of the old network, a processor u has to determine whether the biconnected component (say B_i), containing (u,v) is injured or not. The processor u can determine it using the spanning tree T_i of B_i. (Recall $T_i = T \cap B_i$). If there exists a deleted link in B_i, T_i may be partitioned into some fragments. In any case, however, there exists a processor w in the fragment containing u such that w is incident to a deleted link in B_i. Thus, by the broadcast-and-convergecast on the fragment of T_i, u can efficiently determine whether B_i contains a deleted link or not. To avoid the duplicate checks within the fragment, u broadcasts the result to all processors in the fragment.

3.3.2 The second phase

In the second phase, the biconnected components of N^* are computed by applying our algorithm in [7] to N^*. The algorithm in [7] is based on the depth-first search of N^*, and it also constructs a rooted spanning tree of N^*. We utilize the spanning tree in the third phase.

3.3.3 The three phase

In the third phase, the biconnected components of the new network N′ are computed. From Lemma 2, each processor can locally determine which incident links are contained in the same biconnected component. In the BRP, however, we have to assign a common label to *all* links of the same biconnected component.

To determine the label assigned to each link, we use the rooted spanning tree T' of N^* constructed in the second phase. Notice that the processors in the same biconnected component of N' appear consecutively in T', that is, $T' \cap B_i'$ is a spanning tree of B_i' for each biconnected component $B_i'(1 \le i \le m')$ of the new network N'. Thus, for each $B_i'(1 \le i \le m')$, we can uniquely determine a processor u_i that is nearest to the root among processors incident to a link of B_i'. We call the processor u_i a *representative* of B_i'. The representative u_i determines the unique label of B_i' as follows.

In the third phase, the depth-first traverse of T' is executed. During the depth-first traverse, we maintain the number t of labeled biconnected components of N'. When a representative u_i first choose a link of B_i to make the depth-first traverse proceed, u_i increments t by one and determines the label of B_i to be the updated t. The processors incident to a link of B_i is informed of the label in progress of the depth-first traverse.

3.4 Complexities

In the following theorem, n(resp. e) is the number of processors(resp. links) in the old network, n'(resp. e') is the number of processors (resp. links) in the new network, a is the number of added links, and b is the total number of links of N' in the injured biconnected components.

Theorem 1 The algorithm presented in this section solves the BRP with the message complexity of $O(n' + a + b)$, the bit complexity $O((n' + a + b)\log n')$, the ideal time complexity $O(n')$, and the space complexity $O(e\log n + e'\log n')$.

(proof) The first and second phases can be executed with the message complexity $O(n' + a + b)$ and the ideal time complexity $O(n')$ using the depth-first search algorithm in [6] and the biconnected component algorithm in [7]. It is also clear that both of the message complexity and the ideal time complexity of the third phase are $O(n')$. Thus, the message complexity and the ideal time complexity of the whole algorithm is $O(n' + a + b)$ and $O(n')$ respectively.

Notice that the label assigned to a link does not exceed the number of processors. Thus, the length of each message is $O(\log n')$, and the bit complexity is $O(n' + a + b)\log n')$. Since each processor maintains the old label and the new label for each of its incident links, the space complexity is $O(e\log n + e'\log n')$. ■

4. Conclusions

This paper proposed a distributed algorithm that solves the BRP after several processors and links are added and deleted. Its message complexity and its ideal-time complexity are $O(n'+a+b)$ and $O(n')$ respectively, where n' is the number of processors in the network after the topology change, a is the number of added links, and b is the total number of links in the biconnected components (of the network before the topology change) including the deleted links.

References

[1] M. Ahuja and Y. Zhu. An efficient distributed algorithm for finding articulation points, bridges and biconnected components in asynchronous networks. In Proc. 9th Conference on Foundations of Software Technology and Theoretical Computer Science(LNCS 405), pp.99-108, 1989.

[2] E.J.H.Chang. Echo algorithms : depth parallel operations on general graphs. IEEE Trans. Software Engineering, 8(4) : pp.391-401, 1982.

[3] T.H.Cormen, C.E.Leiserson, and R.L.Rivest. Introduction to Algorithms, The MIT Press, 1990.

[4] W. Hohberg. How to find biconnected components in distributed networks. Journal of Parallel and Distributed Computing, 9(4) : pp.374-386, 1990.

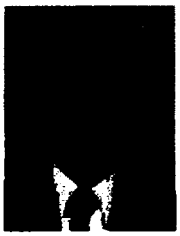
[5] T. Kameda and M.Yamashita. Distributed Algorithms.(in Japanese), Kindai-Kagaku-Sya, 1994.

[6] K.B.Lakshmanan, N.Meenakshi, and K.Thulasiraman. A time optimal message efficient distributed algorithm for depth first search. Information Processing Letters, 25 : pp.103-109, 1987.

[7] J.Park, T.Masuzawa, K.Hagihara, and N.Tokura. Efficient distributed algorithms solving problems about the connectivity of network.(in Japanese), Journal of IEICE(DI), J72-D-I(5) : pp.343-356, 1989.

[8] B. Swaminathan and K.J.Goldman. An incremental distributed algorithm for computing biconnected components. In Proc. 8th International Workshop on Distributed Algorithms(LNCS 857), pp.238-252, 1994.

[9] G.Tel. Introduction to Distributed Algorithms, Cambridge University Press, 1994.



이 창 석

1978년 성균관대학교 사범대학 수학교육과 졸업

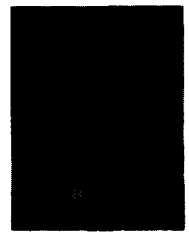
1981년 성균관대학교 경영대학원 정보처리학과 졸업

1995년 충북대학교 대학원 전자계산학과 박사과정 수료

1981년~1991년 대전실업전문대학 전자계산과 부교수

1992년~현재 농협대학 전자계산과 부교수로 재직 중

관심분야 : 분산알고리즘, 소프트웨어공학



박 정 호

1980년 성균관대학교 사범대학 졸업(문학사)

1980년~1982년 성균관대학교 경영대학원 정보처리학과(경영학석사)

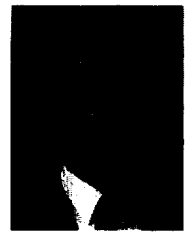
1985년~1987년 日本 오사까대학교 대학원 정보공학전공(공학석사)

1987년~1990년 日本 오사까대학교 대학원 정보공학전공(공학박사)

1996년~현재 한국정보처리학회 총무이사

1991년~현재 선문대학교 정보과학부 부교수

관심분야 : 분산알고리즘, 소프트웨어공학



구 연 설

1964년 청주대 상학과

1975년 성균관대 대학원 전자자료처리학과

1981년 동국대 대학원 통계학과(이학석사)

1988년 광운대학교 대학원 전자계산학과(이학박사)

1979년 이후 충북대전자계산소장, 한국정보과학회 부회장 역임

현재 충북대 컴퓨터과학과 교수

관심분야 : 소프트웨어공학, 정보통신 등