최단 경로 갱신문제를 해결하는 분산알고리듬

박 정 호† · 이 경 오†† · 강 규 철†††

요 약

최단경로 등의 문제를 해결하는데 필요한 정보가 네트워크상의 프로세서에 분산되어 있는 상황에서, 그들 정보를 교환하면서 그 문제를 해결하는 알고리즘을 분산알고리즘(Distributed Algorithm)이라고 한다.

최단경로가 이미 구성되어 있는 비동기식 네트워크상에서 네트워크 형상이 변할 경우, 이로인해 구성되어 있던 최단경로를 갱신해야 하는 경우가 발생한다. 본 논문에서는 이러한 경우 최단경로를 효율적으로 갱신하는 메시지 복잡도와 이상시간 복잡도가 모두 $O(p^2 - q + n')$인 분산알고리즘을 제안한다. 여기서, q는 삭제 링크의 수, n'는 네트워크의 토폴로지가 변한후의 네트워크상에 존재하는 노드수를 각각 나타낸다. 그리고, p는 삭제 또는 추가 링크를 가진 이중연결성분에 속하는 전체 노드 수를 나타낸다

An Efficient Distributed Algorithm for the Weighted Shortest-path Updating Problem

Jeong-Ho Park† · Kyung-Oh Lee†† · Kyu-Chul Kang†††

ABSTRACT

We consider the weighted shortest path updating problem, that is, the problem to reconstruct the weighted shortest paths in response to topology change of the network. This paper proposes a distributed algorithm that reconstructs the weighted shortest paths after several processors and links are added and deleted. Its message complexity and ideal-time complexity are $O(p^2 - q + n')$ and $O(p^2 + q + n')$ respectively, where n' is the number of processors in the network after the topology change, q is the number of added links, and p is the total number of processors in the biconnected components (of the network before the topology change) including the deleted links or added links

1. Introduction

We consider distributed algorithms operating on a network of processors connected by communication links. On such a network, it is important to reconstruct the shortest paths since the shortest paths are relevant to communication costs and the other network management. Several distributed algorithms for computing the shortest path have been proposed[1-3, 5, 6, 8]. These previous works consider the problem for computing the shortest paths from scratch. However, in a real network the 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 updating problems, that is, the problem to reconstruct solution(e.g. the shortest path) in response to the topology change. Some distributed algorithms for the

† 종신회원 : 신문대학교 컴퓨터정보학부 교수
†† 정 회 원 : 신문대학교 컴퓨터정보학부 교수
††† 정 회 원 : 성결대학교 경영행정학부 교수
논문접수 : 2000년 2월 23일, 심사완료 : 2000년 5월 2일

updating problem have been proposed[7,9].

We consider the Weighted Shortest-path Updating Problem(WSUP), that is, the problem to reconstruct the weighted shortest paths in response to topology change. It is obvious that the WSUP can be solved by the known distributed algorithms that compute the weighted shortest paths from scratch. However, it is a natural assumption that each processor knows the *old solution*, that is, each processor initially knows the weighted shortest-path 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 WSUP be solved using an auxiliary information? This is a very interesting subject of study.

Up to date, any algorithm solving the WSUP, that is, the algorithm to reconstruct the shortest path in response to topology change by utilizing some auxiliary information has not proposed. However, the WSUP can be solved by the previous algorithm in [8] which does not utilize any auxiliary information and reconstructs the weighted shortest path from scratch. Its message complexity and ideal-time complexity are $O(n'^2)$ and $O(n'^2)$ respectively, where n' is the number of processors in the network after the topology change.

This paper proposes an efficient distributed algorithm for the WSUP after addition and deletion of several processors and links by utilizing some auxiliary information. Our algorithm utilizes two kinds of auxiliary informations : the solution of the shortest paths and the solution of the biconnected components. The solution of the shortest paths is which of the adjacent links are links connected to its father or its sons. And, the solution of the biconnected components 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. Its message complexity and ideal-time complexity are $O(p^2 + q + n')$ and $O(p^2 + q + n')$ respectively, where q is the number of added links, and p is the total number of processors in the biconnected component (of the network before the topology change) including the deleted links or added links. Note that we assume topology change does not occur during the execution of the algorithm.

In general, it holds that $p \leq n'$ and $q \leq n'^2$ Therefore, our algorithm is superior to the algorithm in [8] with respect to the message complexity and ideal-time complexity. That is, it is more efficient to utilize some auxiliary information than not to utilize any information.

2. Preliminaries

2.1 Graphs

An (undirected) weighted 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). With each edge $e \in E$ let there be associated a number $w(e)$, called its *weight* We use standard definition [8] for a neighbor, a path, a cycle, a connected graph, a 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.

We consider a finite non-null sequence $v_0e_1v_1e_2v_2\cdots e_kv_k$ whose terms are alternately vertices and edges, such that, for $1 \leq i \leq k$, the ends of e_i are v_{i-1} and v_i. If each v_i is distinct, we say that the above sequence is a path from v_0 to v_k. In general, there are many path between v_0 and v_k, and the minimum weighted path among the paths between v_0 and v_k will be called the *shortest path* between v_0 and v_k, and it is denoted by (v_0, v_k)-*path*. Also, its distance is denoted by $d(v_0, v_k)$

When a node r is specified as root, we construct the shortest path between r and the other nodes. It is called the *problem constructing the shortest path.*

2.2 Distributed system model

A network N is a pair (P, L) where P is a finite set of processor 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 notation for networks.

Our model is standard one, that is, (A1) through (A4) are assumed. (See [1, 5] for more details).

(A1) All processors execute the same program. The program consists of (a) internal operation 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 communication directly with its neighbors by sending and receiving message along the links.

(A3) Every link is completely fault-free and works.

(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 The Weighted Shortest-path Updating Problem (WSUP)

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

We say that the shortest path of a network N is already determined in a configuration c, if every processor knows which of its adjacent links are links connected to its father or its sons.

The Weighted Shortest-path Updating Problem (WSUP) is the problem to recompute the weighted shortest paths 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 WSUP is the problem to reach the following final configuration from the following initial configuration.

- The initial configuration : The shortest paths 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 shortest paths of the new network N' are determined. As the solution of the WSUP each processor knows which of the adjacent links are links connected to its father or its sons.

In order to solve the WSUP efficiently, we can make use of some auxiliary information. If we use some auxiliary information, the weighted shortest paths must be also updated so as to correspond to the new network N'. The algorithm proposed in this paper makes use of biconnected components of N, and also recomputes biconnected components of N'. Notice that links of biconnected components 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).
- $N' = (P', L')$: the new network(after topology change).
- $B = (P_B, L_B)$: a biconnected component.
- $T(N,r) = (P_{T(N,r)}, L_{T(N,r)})$: the weighted shortest path tree of the old network N.
- $T(N',r) = (P_{T(N',r)}, L_{T(N',r)})$: the weighted shortest path tree of the new network N'.
- $S_{T(N',r)} = (P'_{T(N',r)}, L'_{T(N',r)})$: the subgraph of $T(N',r)$.

In this paper, the WSUP 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.

The Biconnected Component Updating Problem (BCUP) is the problem to recompute the biconnected components after change of the network topology. That is, it 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 determined. Each link is labeled so that the links with the same label form a biconnected component, and as the solution of the BCUP each processor knows the label assigned to each of its incident links.

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 transimitted 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 the assumption is used only for purpose of evaluation of the ideal time complexity.

3. Algorithm solving the WSUP

In this section, we present an algorithm for reconstructing the shortest paths after topology change.

3.1 The idea of our algorithm

To solve efficiently the WSUP, we use two kinds of information as auxiliary information : the solution of the biconnected components and the solution of the shortest paths.

First, we will present the idea of our algorithm by utilizing the property of the biconnected component. Replace a biconnected component in the network N with a processor, and then N becomes a tree $T' = (V_{T'}, E_{T'})$, where $V_{T'} = \{v \in V_T | v$ corresponds to a biconnected component$\}$ and $E_{T'} = \{(u, v) \in E_T | u(\in V_{T'})$ and $v(\in V_{T'})$ are adjacent biconnected components in $N\}$.

It is assumed that the topology of the network N changed. Then, there might be three cases, according as a biconnected component contains the deleted links or added links.

(case 1) when any biconnected component does not contain the deleted links and added links

Though the topology is changed, this is the case restored to the original network N by recoveries and failures. Because in the result any biconnected components are not changed, the topology of the tree T' is not changed and the shortest paths between the root r and the other processors are the same as before. That is, $T(N,r) = T(N',r)$. Thus, additional operations to solve the WSUP are not necessary.

(case 2) when there is a biconnected component B containing the deleted links

In this case, for each processor u in the B, the shortest path (r, u)-*path* and the distance $d(r, u)$ might be changed. Thus, for the B, we must ignore *the old solution and reconstruct the shortest paths* from scratch.

But, for the another component B' which is the descendant of the B in T', a change of the B has not an effect on the topology within the B'.

If the topology of the B' is not changed, for each processor v in the B', its distance $d(r, v)$ only might be updated. Thus, we must update the distance of the processor v. The distance can easily be updated by

utilizing the shortest path tree of the B'. If the topology of the B' is changed, it is sufficient to reconstruct the shortest path tree of the B' without regard to the change of the B.

(case 3) the case of containing the added links

In this case, added links combine a number of the biconnected components, and these components become one component. For each processor u in these combined components, the shortest path (r, u)-*path* and its distance $d(r, u)$ might be changed. Thus, we must ignore the old solution and reconstruct the shortest paths for these combined components from scratch.

But, for the another component B' which is the descendant of these combined components in T', combined components have no effect on the topology within the B'.

If the topology of the B' is not changed, for each processor v in the B', its distance $d(r, v)$ only might be updated. Thus, we must update the distance of the processor v. The distance can easily be updated by utilizing the shortest path tree of the B'. If the topology of the B' is changed, it is sufficient to reconstruct the shortest path tree of the B' without regard to the combination of the components

3.2 The outline of our algorithm

In order to reconstruct the shortest path tree $T(N',r)=(V_{T(N',r)}, E_{T(N',r)})$, the root r iterates the expansion of the subgraph $S_{T(N',r)}=(SV_{T(N',r)}, SE_{T(N',r)})$ of the tree $T(N',r)$, where in the initial configuration $SV_{T(N',r)}=\{r\}$ and $SE_{T(N',r)}=\varnothing$. By solving the problem reconstructing the shortest paths about each biconnected component, the root r solves the problem reconstructing the shortest paths about the entire new network N'. On the process of the expansion, root r divides a biconnected component into two classes - *the uninjured biconnected component* and *the injured biconnected component*. That is, the B is called *the uninjured biconnected component* if a B of N is unchanged in N'. Otherwise, it is called *the injured biconnected component*.

(step 1) the reconstruction of the biconnected components in N'

In this step, root r reconstructs the biconnected components in N' by utilizing our algorithm in [9]. Each biconnected component in N' has a unique label, and each processor has a label for each incident link.

(step 2) the decision of a component's class

If the label of a biconnected component B in N is not changed in N', the B becomes the uninjured biconnected component. Otherwise, the B becomes the injured biconnected component

(step 3) Reconstruction of the shortest path tree

Root r reconstructs the shortest path tree of each biconnected component as follows.

(step 3-1) the case of the uninjured biconnected component

The processor which has the minimum weighted path in the uninjured biconnected component B is called the *representative* for the B. For this B, the shortest paths between root r and the other processor in the B are not changed.

But, if the distance of the representative for the B is not changed in N', additional operations for this B are not necessary. Otherwise, root r updates the distance of the other processors in the B by utilizing the shortest path of the B

(step 3-2) the case of the injured biconnected component

For the injured biconnected component B, the root r reconstructs the shortest paths of the B by utilizing the algorithm [8]. That is, assume that the subgraph $S_{T(N')}=(SV_{T(N')}, SE_{T(N')})$ of the shortest path tree $T(N')$ is constructed. Then, by utilizing the subgraph $S_{T(N')}$, the root r selects the minimum weighted link among the weight of a link (u, v), where $u \in V_{T(N')}$ and $v \notin V_{T(N')}$. The processor v adjacent to the minimum

weighted link (u, v) and the link (u, v) are contained to the subgraph $S_{T(N')}=(SV_{T(N')}, SE_{T(N')})$ and $S_{T(N')}$ is expanded. Then, the subgraph $S_{T(N')}$ is a pair $(SV_{T(N')}, SE_{T(N')})$, where $SV_{T(N')}=SV_{T(N')} \cup \{v | v$ is the processor adjacent to the minimum weighted link$\}$ and $SE_{T(N')}=SE_{T(N')} \cup \{(u, v)|\ (u, v)$ is the minimum weighted link$\}$.

If the number of the processors in the injured biconnected component B is k, the shortest path tree of the B is completed by containing a processor and link k times.

4. Complexities

In step 1 of our algorithm, the biconnected components are reconstructed by utilizing the algorithm in [9]. In step 2, each biconnected component B is divided into two kinds of class. And, the shortest path of each biconnected component in N' is reconstructed in step 3. That is, if a biconnected component B is the uninjured biconnected component, updating the shortest path of the B can be omitted. Otherwise, the shortest path of the B is reconstructed by utilizing the algorithm in [8]. Therefore, it is obvious from the specification of the algorithm that the algorithm presented in this paper solves correctly the WSUP.

In the following theorem, n' is the number of processors in the network after the topology change, q is the number of added links, and p is the total number of processors in the biconnected component (of the network before the topology change) including the deleted links or added links.

[Theorem 1] The algorithm presented in this paper solves the WSUP with the message complexity of $O(p^2+q+n')$ and ideal-time complexity $O(p^2+q+n')$.

(proof) Because in step 1 the biconnected components are reconstructed by utilizing our algorithm in [9], the message complexity and ideal-time complexity in step 1 are $O(p+q+n')$ respectively.

Because in step 2 the class of each biconnected component is divided and it is easily decided by the label, the message complexity and ideal-time complexity in step 2 are $O(1)$ respectively.

For the uninjured biconnected component B, only the distance in the worst case is updated by utilizing the shortest path tree of the B and a tree has at most $n-1$ links, the message complexity and ideal-time complexity in step 3-1 are $O(n)$ respectively.

For the injured biconnected component B in N', it is assumed that from the assumption there are at most p processors. And step 3-2 is repeated p times in order to reconstruct the shortest path of the B by utilizing the subgraph $S_{T(N')}=(SV_{T(N')}, SE_{T(N')})$ of the shortest path tree $T(N')$. Thus, the message complexity and ideal-time complexity in step 3-2 are $O(p^2)$ respectively.

Therefore, the message complexity and ideal-time complexity of our algorithm are $O(p^2+q+n')$ respectively. ■

5. Conclusions

This paper proposed a distributed algorithm that solves the WSUP after several processors and links are added and deleted. Its message complexity and ideal-time complexity are $O(p^2+q+n')$ and $O(p^2+q+n')$, where n' is the number of processors in the network after the topology change, q is the number of added links, and p is the total number of processors in the biconnected component (of the network before the topology change) including the deleted links or added links.

References

[1] B. Awerbuch : "Complexity of network synchronization," *Journal of ACM*, Vol 32, No.4, pp.804-823(Oct. 1985).

[2] B. Awerbuch and R. G. Gallager : "A new distributed algorithm to find breadth first search trees," *IEEE Trans. on Information Theory*, Vol.IT-33, No.3, pp.315-322(1987).

[3] B. Awerbuch : "Distributed shortest paths algorithm," *Proc. of 21st Symposium on Theory of Computing*, pp.490-500(1980).

[4] M. Ahuja and Y. Zhu : "An efficient distributed algorithms for finding articulation points, bridges and biconnected components in asynchronous network," *In Porc. 9th Conference on Foundations of Software Technology and Theoretical Computer Science*(LNCS 405), pp.99-108(1989).

[5] T. H. Cormen, C. E. Lerserson and R. L. Rivest : "Introduction to Algorithms," The MIT Press(1990).

[6] J. Park, T. Masuzawa, K. Hagihara and N. Tokura : "An efficient distributed algorithm for breadth first spanning tree problem," *Journal of IEICE*(D), Vol. J71-D, No.7, pp.1576-1188(1988).

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

[8] T. Kameda and M Yamashita : "Distributed Algorithms," Kindai-Kagaku-sya(1994)

[9] J. Park, C. Lee : "An Algorithm Solving the Biconnected-components Reconstruction Problem," *Journal of KIPS*, Vol.5, No.10. pp.2512-2520(1998)

박 정 호

e-mail : jhpark@omega.sunmoon.ac.kr
1980년 성균관대학교 사범대학 졸업(문학사)
1980년~1982년 성균관대학교 경영대학원 경보처리학과 (경영학석사)
1985년~1987년 日本 오사까대학교 대학원 정보공학전공(공학석사)
1987년~1990년 日本 오사카대학교 대학원 정보공학전공(공학박사)
1996년~현재 한국정보처리학회 총무이사
1991년~현재 선문대학교 컴퓨터정보학부 부교수
1999년~현재 선문대학교 연구처장
관심분야 : 분산알고리즘, 원격교육, XML, 소프트웨어공학

이 경 오

e-mail : leeko@rainbow.sunmoon.ac.kr
1989년 서울대학교 계산통계학과
1994년 서울대학교 전산과학과 그래픽스 전공
1999년 서울대학교 전산과학과 멀티미디어 전공
1999년~현재 선문대학교 컴퓨터정보학부 전임강사
관심분야 : 멀티미디어, 데이타베이스, 인터넷응용

강 규 철

e-mail : kkch@sungkyul.ac.kr
1980년 인하대학교 산업공학과
1982년 명지대학교 경영학과 생산관리 전공
1996년 명지대학교 경영학과 생산관리 전공
1976년~1999년 텔슨전자 경영자문 팀장
1994년~1999년 명지전문대학 총동문회장
1998년~1999년 산업정영시스템학회 이사
1996년~2000년 성결대학교 경영행정학부 교수
관심분야 생산관리, 지식경영, 일용직 근로자의 가치관