論 文
34~6~3

# 병렬구조 전산기를 이용한 최단 경로 계산

## Shortest Path Calculation Using Parallel Processor System

徐 昌 鎭* · 李 章 揆**
(Chang-Jin Suh · Jang-Gyu Lee)

**Abstract**

Shortest path calculations for a large-scale network have to be performed using a decomposition technique, since the calculations require large memory size which increases by the square of the number of vertices in the network. Also, the calculation time increases by the cube of the number of vertices in the network. In the decomposition technique, the network is broken into a number of smaller size subnetworks for each of which shortest paths are computed. A union of the solutions provides the solution of the original network. In all of the decomposition algorithms developed up to now, boundary vertices which divide all the subnetworks have to be included in computing shortest paths for each subnetwork. In this paper, an improved algorithm is developed to reduce the number of boundary vertices to be engaged. In the algorithm, only those boundary vertices that are directly connected to the subnetwork are engaged. The algorithm is suitable for an application to real time computation using a parallel processor system which consists of a number of micro-computers or prcessors. The algorithm has been applied to a 39-vertex network and a 232-vertex network. The results show that it is efficient and has better performance than any other algorithms.

A parallel processor system has been built employing an MZ-80 micro-computer and two Z-80 microprocessor kits. The former is used as a master processor and the latter as slave processors. The algorithm is embedded into the system and proven effective for real-time shortest path computations.

## 1. 서 론

회로망은 점과 같이 위치만을 표시하는 정점(Vertex)과 정점들을 이어주는 가지(Edge)로 이루어진다. 만일 두 정점 사이에 직접 연결되어 있는 가지가 존재하지 않을 때에는 중간에 여러 정점을 거치면서 연결되어 있을 것이다. 이와같이 연속적으로 연결된 여러개의 가지를 경로(Path)라 한다. 최단경로 계산을 할 경우에는 각각의 가지들에 실수의 가지값(Weight)이 주어지게 된다. 이때 한 정점에서 다른 정점까지의 가능한 모든 경로 중 각각의 가지값을 합한 경로값이 가장 적어지는 경로를 최단경로라 한다. 최단경로 계산이란 모든 정점에서 모든 정점까지의 최단거리를 알려주는 거리행렬과 최단경로를 나타내는 경로행렬을 구하는 문제이다.

최단경로 계산은 운송시스템, 정보통신시스템, 전력수송시스템 등에서 효율적인 시스템 운영을 위하여 필요한 하나의 OR (Operations Research) 문제이며 실시간(Real Time)으로 계산되어야 할 필요성 때문에 빠른 계산 속도가 요구된다. 지금까지 가장 많이 사용되는 방법들을 살펴보면 Floyd 법[1], Dantzig 법[2] 은 최초로 개발된 알고리즘들로서 똑같은 계산량이 요구되며 단지 그 계산 순서가 서로 다를 뿐이다. Rosenthal 방법[3]은 분류방법 (sorting)을 도입함으로써 계산량을 감소하였다. 이러한 방법들은 계산량이 정점수의 세제곱에 비례하고 사용 메모리 또한 제곱의 두배 혹은 그 이상이 필요하게 되어 메모리의 상한이 주어지는 실제 전산시스템에 적용할 경우 회로망이 커짐에 따라 사용의 제한이 따르게 된다. 가상 메모리(Virtual Memory)의 사용이

*正 會 員 : 新榮電機(株)第一그룹研究員
**正 會 員 : 서울大 工大 制御計測工學科 助教授 · 工博
接受日字 : 1985年 3月 8日

가능할지라도 메모리 참조가 빈번한 알고리즘의 특성 때문에 수행시간이 무척 길어지게 되며, 많은 양의 가상 메모리를 사용했을 때는 더욱 더 많은 시간이 소요된다.

이러한 문제점을 해결하기 위해서 전체의 회로망을 몇개의 소회로망으로 나누어 소회로망에서 구한 정보들을 토대로 전체 회로망의 최단경로 계산을 하는 분할방법[4]이 등장하여 대형 회로망에서도 메모리를 적게 사용하면서 빠른 수행시간으로 계산을 마치게 되었다. 그러나 기존의 분할방법은 소회로망의 수가 증가할수록 중복된 계산을 하게 되어 이러한 단점을 소경계 정점집합 개념을 사용하여 분할방법을 좀 더 개선할 수 있다. 분할방법은 상술한 정점 이외에 병렬적으로 계산이 수행될 수 있다는 특성이 있다. 이로 인하여 분할방법은 병렬구조를 가진 전산기의 도입으로 매우 빠른 수행시간으로 최단경로 계산을 마칠 수 있다. 여기에서 병렬구조를 가진 전산기란 여러개의 프로쎄서가 독립성을 가지고 각기 다른 업무(Job) 수행을 동시에 해 나가는 다중 마이크로프로쎄서 시스템을 말한다. 이러한 시스템은 VLSI 기술의 발전과 마이크로프로쎄서 칩 가격의 하락으로 다른 형태의 전산기와 경쟁할 수 있는 새로운 형태의 전산시스템으로 대두되고 있다.[6]~[9]

본 논문에서는 다중 마이크로 프로쎄서 시스템을 사용하여 분할법에 의한 알고리즘을 수행함으로써 알고리즘이 새로운 전산시스템에 적합함을 보이고자 한다.

논문의 구성은 다음과 같다. 2장에서는 최단경로 계산을 정의하고 분할하지 않은 상태에서 이 문제를 푸는 알고리즘을 설명한다. 3장에서는 분할방법에 의한 최단경로 계산을 소개하고 새로운 알고리즘을 제시하며 4장에서는 이 개선된 알고리즘을 분석한다. 5장에서는 알고리즘의 병렬화를 보이고 이를 1대의 MZ-80 8비트 마이크로 전산기에 2대의 Z-80 마이크로 프로쎄서를 연결한 병렬구조 시스템을 구성하여 분할 방법에 의한 최단경로 계산을 수행한 예를 설명한다.

## 2. 최단경로 계산

회로망은 정점(Vertex)과 가지(Edge)로 구성된다. 정점은 1, 2, 3……N의 순서로 고유번호를 하나씩 부여하여 구분하며 이 정의에 의거하여 가지(Edge)는 출발정점과 도착정점을 명시함으로써 정의된다. 이러한 정의로 얻어진 정점집합 V와 가지집합 A로부터 회로망 G는 정의되고 이에 따라 회로망의 행렬화가 가능하다. 행렬의 행과 열은 출발정점의 고유번호와 도착정점의 고유번호를 각각 뜻한다. 행렬원소의 의미에 따라 비용행렬 C, 거리행렬 D, 경로행렬 S로 나눈다. 비용행렬은 주어진 회로망을 행렬화하여 얻어진 것으로 최단경로 계산의 입력이다. 행렬원소 c(i, j)는 출발정점에서 도착정점까지 한개의 가지로 연결되어 있을 경우 그 가지의 가지값을, 가지가 존재하지 않을 때에는 '∞'를, 출발정점과 도착정점이 일치할 경우는 '0'을 각각 가진다. 거리행렬원소 d(i, j)는 출발정점에서 도착정점까지의 최단거리를 나타낸다. 이 값은 최단경로의 가지값들의 합만을 알려주고 있으므로 d(i, j)의 값이 나오게 된 최단경로를 경로행렬 S에 간직시켜야 한다. 경로행렬원소 s(i, j)는 정점 i에서 정점 j로 가장 빠르게 도달하기 위해서 정점에서 경유하여야 할 다음 정점번호를 간직하고 있다. 정점 i에서 정점 j까지의 최단경로를 알기 위해서는 s(i, j)를 찾아보고 이 값이 $i_1$이라면 s($i_1$, j)를 그 다음에 찾아본다. 만일 그 값이 $i_2$라면 s($i_2$, j)를 찾아본다. 이러한 절차를 s($i_K$, j) = j가 나올 때까지 반복한다. 이때 i에서 j까지의 **최단경로**는 $i \rightarrow i_1 \rightarrow \cdots \rightarrow i_K \rightarrow j$가 된다. 주어진 회로망에 대하여 거리행렬과 경로행렬을 알게되면 회로망내의 한 정점에서 다른 정점까지의 최단 경로와 그 값을 얻을 수 있게되고 따라서 최단경로 계산이란 주어진 비용행렬로부터 거리행렬과 경로행렬을 구함을 뜻한다. 최단경로 계산은 여러 방법에 의해서 풀 수 있지만 기본적으로 다음과 같은 식을 기초로 계산된다.

$$d(i,k) = \min(d(i,k), d(i,j)+d(j,k)), \quad i \neq j, \quad j \neq k \qquad (1)$$

정점 i에서 정점 k로 갈수 있는 방법은 직접 하나의 가지로 가장 빨리 갈수도 있으나 여러개의 중간정점을 지나면서도 더 적은 경로 값을 가질 수도 있으므로 이러한 상황을 식(1)에서 검토하는 것이다. 모든 경우를 다 고려하여 이러한 비교를 하면 d(i,k)는 최종적인 최단거리를 얻게 된다. 단지 이 비교를 어떠한 순서에 의해서 하느냐에 따라서 방법들이 분류가 되며 몇번의 비교를 행했는가에 따라서 그 알고리즘의 우수성이 결정된다. 잘 알려진 최단경로 계산 알고리즘으로는 Floyd 방법[1], Dantzig 방법[2], 그리고 Rosenthal 방법[3]이 있다. Floyd 방법과 Dantzig 방법은 모두 식(1)과 같은 오퍼레이션을 회로망내 모든 정점에 대하여 반복적으로 실시하는 것이고 Rosenthal 방법은 식(1)의 오퍼레이션에 분류방법(Sor-

ting) 을 도입하여 오퍼레이션 수를 감소하는 것이다. 일반적인 회로망에 대하여 Rosenthal 방법은 다른 두 가지 방법과 비교하여 3배 내지 4배 빠른 계산속도를 갖는다.[4]

## 3. 분할방법에 의한 최단경로 계산

분할방법은 회로망을 분할함으로써 가능하다. 주어진 회로망을 어떻게 분할할 것인가 하는 것이 선결되어야 하나 그 자체가 하나의 어려운 문제로 이 논문에서는 다루지 않았으며 이는 참고문헌[5]에 기술되어 있다. 여기서는 회로망이 이미 분할된 상태에서 최단경로 계산을 시작한다.

우선 회로망이 분할됨에 따라 정의되는 용어를 설명한다. 경계정점 (Boundary Vertex) 및 경계정점집합(Boundary Vertex Set)은 경계정점의 모든 원소와 이것에 직접 연결된 가지를 전체회로망에서 제거할 경우 원하는 수 만큼의 소회로망으로 분리하게 하는 정점 및 이 정점들로 이루어진 집합이다.

절단집합 ( cutset)은 경계정점집합과 이 정점들에 직접 연결된 가지를 합한 집합이다. 소회로망(Subnetwork)은 전체회로망에서 절단집합을 제거함으로써 나누어진 몇개의 회로망중의 한 회로망을 말한다. 소경계정점집합(Sub−boundary Vertex Set) i는 소회로망 i의 임의의 한 정점으로부터 하나의 가지에 의해 연결된 경계정점으로 이루어진 집합이다. 소절단집합(Sub − cutset) i는 소경계정점집합 i와 이 정점에 직접 연결된 가지를 합한 집합이다. 분할

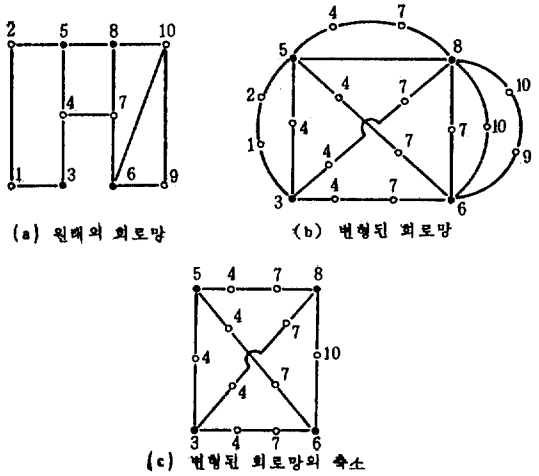



그림 1. 회로망 변형의 예
**Fig. 1.** An example of network transformation.

방법에 의한 최단경로 계산의 핵심은 경계정점사이의 최단경로를 알 수 있으면 소회로망 밖의 모든 경우는 경계정점사이의 최단정보를 모두 포함시킬 수 있어 불필요한 계산을 줄이게 된다는 것이다. 우선 경계정점사이의 최단경로 계산을 해 보자. 이 방법은 본래의 회로망을 동일하지만 다른 형태의 회로망으로 변형시킨 후에 이를 간략하게 만들고 간략해진 회로망으로부터 최단경로 계산을 하여 최단경로 행렬 및 최단거리 행렬을 얻는다. 그림 1에 회로망이 변하는 과정을 보여주고 있다. 그림 1(b)는 원래의 회로망 그림 1(a)를 경계정점만을 중심으로 변형시킨 모습이다. 즉 경계정점을 제외한 나머지 정점은 마치 가지의 일부라고 간주하여 중간에 다른 경계정점을 경유하지 않고 도달할 수 있는 경계정점사이의 모든 가능한 경로를 도시하였다. 각 경계정점사이의 경로에는 동일한 소회로망에 속해 있는 정점들만으로 이루어져 있다. 그런데 그림 1(b)에서 보듯이 변형된 회로망에는 동일한 두 정점사이의 가지가 2개 이상도 생기게 된다. 최단경로계산에는 2개 이상의 가지가 있는 경우 가지값이 큰 가지는 그 보다 가지값이 작은 가지 때문에 한번도 사용되지 못하므로 경로값이 가장 작은 경로를 남겨놓고 나머지는 회로망에서 제거하여도 무관하다. 그 결과가 그림 1(c)에 나와 있는데 이 변형된 회로망은 경계정점외의 정점들을 정점이 아닌 것으로 간주한다면 이 회로망은 정점의 수가 $n_b$ ( $n_b$ 는 경계정점수) 인 회로망이라 할 수 있다. 그러므로 이 회로망에 기존의 최단경로 계산을 적용한다면 경계정점사이의 최단경로를 계산할 수 있다.

보조정리 1 : 경계정점을 경유하지 않고서는 다른 소회로망에 속한 정점들은 서로 연결될 수 없다.

(증명) 정의에 의하여 경계정점과 그와 관련된 가지를 합한 절단집합을 전체 회로망에서 제거시키면 몇 개의 소회로망으로 나누어지게 되므로 다른 소회로망에 속한 정점들을 연결하려면 경계정점을 반드시 경유해야 한다. 이 말에 대우를 취하면 보조정리 1이 된다.

보조정리 2 : 한 경계정점에서 다른 경계정점을 경유하지 않고 주위의 소회로망 i의 정점을 하나 이상 경유하면서 경계정점에 도달 할 수 있다면 이 두 경계정점은 같은 소경계정점집합 의 원소이다.

(증명) 소경계정점에서 소회로망으로 유입되었다면 보조정리 1에 의해 다른 소회로망으로 유출될 수 없다. 이런 조건에서 도달할 수 있는 경계정점은 소경계정점의 정의에 따라 소경계정점집

합 i에 속한 경계정점 뿐이다.

정리 1 : 다음과 같은 절차를 거쳐 경계정점끼리의 최단경로 계산을 할 수 있다.

단계 1 : 소회로망 i와 소절단집합 i ( i = 1, ···, n : n은 소회로망의 수) 를 합한 n개의 회로망으로부터 각각 최단경로 계산을 행한다.

단계 2 : 동일한 경계정점사이에 2개 이상의 경로가 존재할 경우 제일 작은 경로값을 가진 경로만을 남긴다.

단계 3 : 단계 2로부터 만들어진 회로망으로 다시 한번 최단경로 계산을 한다.

(증명) 우선 앞에서 언급한 해설과 위의 절차가 동일함을 밝히고 모든 경우에서 최단경로값이 계산될 수 있음을 밝힌다. 보조정리 1, 2에 의해 그림 1(b)의 회로망에서 가지로 표현된 경계정점사이의 모든 경로들은 단계 1과 같은 방법으로 n개로 나누었어도 그중 한 회로망내에서 존재하고 있다. 그러므로 이 n개의 회로망에서 최단경로 계산을 하여도 최단경로는 계속 그림 1(b)에서 남아있게 된다. 경계정점에서 경계정점으로의 경로값은 어떤 소회로망으로 향하느냐에 따라 달라지게 되므로 각 소회로망에서 구한 값을 비교하여 제일 적은 경로값만을 고르면 그림 1(c)와 같은 형태의 회로망을 얻게 된다. 이 상태에서 단계 3을 수행하면 보조정리에서 가정했던 중간에 다른 경계정점을 경유하지 않는다는 제한을 없앤 것이므로 최종적인 경계정점사이의 최단거리 정보를 얻게 된다.

이제 모든 경우에 있어서 최단경로가 얻어지는지의 여부를 알아보자.

(1) 최단경로가 중간에 경계정점을 경유하지 않는 경우 즉 하나의 소회로망만을 지나면서 최종 경계정점에 도달하는 경우 : 단계 2에서 얻어진다.

(2) 최단경로가 중간에 경계정점을 경유하는 경우 즉 여러개의 소회로망을 거쳐서 최종경계정점에 도달하는 경우 : 단계 2를 토대로 단계 3에서 최단값을 얻을 수 있다. Q.E.D.

위의 절차에 의해서 경계정점 사이의 최단경로 계산을 마친후 이 값들을 기초로 하여 나머지 경우를 구하여 본다. 그림 2와 같이 두 개로 나누어진 회로망을 생각해 보자 소회로망 내에서의 최단경로 및 최단거리는 소회로망과 절단집합을 합한 회로망에서 최단경로 계산을 함으로써 구할 수 있다. $G_s$내의 두 정점인 $V_1$에서 $V_2$까지의 최단경로는 두가지 경우를

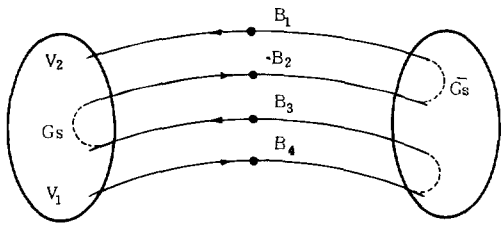



그림 2. 소회로망내의 최단거리 계산

**Fig. 2** Shortest path calculations in a subnetwork.

생각할 수 있다. 우선 최단경로가 $G_s$ 밖의 정점들을 경유하지 않는다면 당연히 최단값들을 얻을 것이다. 만일 그림 2에서처럼 최단경로가 중간에 몇번 $\bar{G}_s$ 내의 정점을 경유하는 경우에도 역시 최단값을 얻을 수 있게 된다. 이것은 소회로망 밖으로 나가기 위해서는 반드시 경계정점을 경유해야 하는데 $B_4$에서 $B_3$까지와, $B_3$에서 $B_2$까지, $B_2$에서 $B_1$까지의 최단경로가 구해져 있으며 이것이 $B_4$에서 $B_1$까지의 최단경로로 이미 얻어진 상태이다. 이 상태에서 $V_1$에서 $B_4$까지와 $B_1$에서 $V_2$까지의 최단거리를 $G_s$와 절단집합을 합한 회로망으로부터 구할 수 있으므로 가능하다. 서로 다른 소회로망끼리의 최단거리는 최단정보를 이용하여 계산량을 줄일 수 있다. 경계정점이 $B_1, B_2, \cdots, B_n$일때 한 소회로망 내의 정점 $V_1$에서 다른 소회로망내의 정점 $V_2$까지의 최단경로는 아래 식(2)에서 얻어진다.

$$d(V_1, V_2) = \min_{1 \le K \le n}\{(d(V_1, B_K) + d(B_K, V_2)\} \quad (2)$$

여기서 n은 전체 경계점의 갯수이다. 만일 소회로망의 갯수가 3개 이상이 되며 경계정점을 사용할 경우 낭비가 생기게 된다. 이 불필요한 부분을 경계정점집합 대신 소경계정점집합을 이용함으로써 제거할 수 있다. 이 때에는 소회로망내의 정점과 경계정점 사이의 최단경로를 얻기 위해서는 소회로망 i와 소경계정점집합 i에 속한 경계정점사이의 최대경로를 얻은 후 이를 토대로 소회로망 i내의 정점과 경계정점 사이의 최단경로를 구하는 두 단계의 계산을 하게 된다. 그림 3에 있는 상황을 생각해 보자. $V_1$과 $V_2$사이의 최단경로가 여러 소회로망을 지나는 이러한 경우에도 중간과정은 $B_2$에서 $B_1$까지의 최단경로로 집약될 수 있으므로 소절단집합 i와 소회로망 i를 합한 회로망에서의 최단경로 계산으로 소회로망내의 모든 정점 사이의 최단경로를 계산할 수 있다. 소회로망 i의 한 정점과 경계정점 사이의 최단경로를 알기 위해서는 소회로망 i의 한 정점과 소경계정점집합 i에 속하지 않은 경계정점 사이의 최단경로를 구하여야 하는데 이것은 식(3)에서 얻어진다.

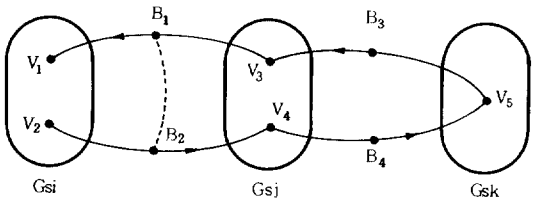



그림 3. 소경계정점집합을 이용한 소회로망내에서의 최단경로 계산

**Fig. 3** Shortest path calculations in a subnetwork utilizing sub-boundary vertex set.

$$d(V,\bar{b}_i) = \min_{1\le j\le n_i}\{d(V,b_j) + d(b_j,\bar{b}_i)\}$$
$$d(\bar{b}_i,V) = \min_{1\le j\le n_i}\{d(\bar{b}_i,b_j) + d(b_j,V)\} \quad (3)$$

여기에서 V는 소회로망 i내의 한 정점을, $b_j$는 소경계정점집합 i에 속해 있는 경계정점을, $\bar{b}_i$는 소경계정점집합에 속하지 않는 경계정점을, $n_i$는 소경계정점집합 i의 원소 갯수를 각각 나타낸다. 서로 다른 회로망끼리의 최단경로 계산도 소경계정점집합을 이용하여 계산수를 줄일 수 있다. 그림 4의 상황을 생각하여 $V_1$에서 $V_2$까지의 최단경로 계산을 중간의 모든 경계정점을 고려한 식(2)를 이용하면 중복적인 계산임을 알 수 있다. $d(V_1, V_2)$는 동일한 경로에 대하여 $d(V_1, B_1) + d(B_1, V_2)$ 및 $d(V_1, B_2) + d(B_2, V_2)$로 중복하여 계산되어진다. 그러므로 출발소회로망의 소경계정점집합만을 비교한다던지 도착소회로망의 소경계정점들만을 비교하여야 한다. 즉, 식(4)로서도 최단경로 값을 얻을 수 있다.

$$d(V_1,V_2) = \min_{1\le i\le n_b}\{d(V_1,B_i) + d(B_i,V_2)\} \quad (4)$$

여기에서 $n_b$는 출발정점이 속한 소회로망에 관련된 소경계정점집합의 소경계정점 갯수를 $B_i$는 그 소경계정점을 나타낸다.

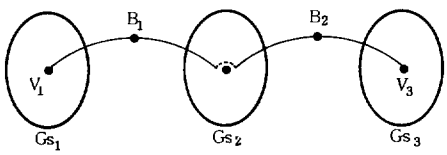



그림 4. 소회로 망사이의 최단경로 계산

**Fig. 4** Shortest path calculations between subnetworks.

## 4. 소경계정점을 이용한 분할방법에 의한 최단경로 계산방법의 비교

앞에서 제시한 소경계정점을 이용한 분할방법의 효율성을 알아보기 위해서 정점수가 N개인 회로망을 여러 방법을 사용하여 최단경로 계산을 할때 수행하여야할 산술 계산의 횟수를 서로 비교한다. 정점수가 N개인 회로망을 B개의 경계정점을 이용하여 $\ell$개의 소회로망으로 나누고 소회로망 j의 경계정점수를 $N_i$개 소경계정점집합 i의 원소 갯수를 $B_i$라 하면 다음과 같은 결과가 성립한다. 우선 전체 정점의 수는 모든 소회로망의 정점수와 경계 정점수를 더한 것이므로 식(5)가 성립하며 경계정점은 대부분 두 개의 소회로망사이에 존재하므로 대략적으로 식(6)의 관계가 성립한다.

$$\sum_{i=1,\ell}(N_i+B) = N \quad (5)$$
$$2B = \sum_{i=1,\ell} B_i \quad (6)$$

산술연산은 덧셈과 비교 만으로 이루어지는데 이 두 가지의 연산횟수는 비슷하므로 여기에서는 덧셈의 횟수만 비교 하기로 한다. 분할을 하지 않은 상태의 최단경로 계산은 주어진 정점의 세제곱 만큼의 덧셈을 수행한다고 하자. 경계정점으로 회로망을 분할한 경우 최단경로 계산을 하기 위해서는 경계정점사이의 최단경로 계산, 소회로망과 절단집합을 합한 회로망에서의 최단경로 계산, 소회로망사이의 최단경로 계산을 행하여야 하므로 이 모든 횟수를 순서대로 적어보면 식(7)과 같다.

$$\left(\sum_{i=1,\ell}(B+N_i)^3+B^3\right)+\sum_{i=1,\ell}(B+N_i)^3 + \sum_{i=1,\ell}\sum_{\substack{i=1,\ell\\ i\neq j}}(N_i.B.N_j) \quad (7)$$

소경계정점집합을 이용한 분할방법의 덧셈횟수를 같은 순서로 적으면 식(8)이 된다. 이때 마지막 항의 $\sum_{i=1,\ell} N_i B_i (B-B_i)$는 소회로망 i의 정점과 소경계정점집합 i 에속하지 않는 경계정점사이의 최단경로 계산에 필요한 덧셈의 횟수이다.

$$\left(\sum_{i=1,\ell}(B_i+N_i)^3+B^3\right)+\left(\sum_{i=1,\ell}(B_i+N_i)^3+\sum_{i=1,\ell}N_iB_i(B-B_i)\right)+\sum_{j=1,\ell}\sum_{\substack{i=1,\ell\\ i\neq j}}(N_i.B_i.N_j) \quad (8)$$

소경계정점집합 개념을 사용한 최단경로 계산 알고리즘과 소 경계정점집합 개념을 사용하지 않은 분할방법 그리고 분할방법을 사용하지 않은 알고리즘들

사이의 결과를 검토하고 또 계산횟수를 서로 비교하기 위하여 정점이 39개인 회로망과 232개인 회로망을 선택하여 적용 시켜 보았다. 그 결과는 물론 일치하였으며 각 방법들 사이의 계산횟수를 표1에, 계산시간을 표2에 표시하였다. 사용된 계산기는 PDP 11/44이며 계산시간에는 데이타 입력시간도 포함된 것이다. 여러가지 경우를 고려 하기 위하여 회로망을 몇가지 다른 소회로망으로 분할하여 조사하였다.

표1을 살펴보면 분할방법은 분할하지 않은 경우보다 더 적은 횟수의 산술연산을 하며 분할방법중 소경계정점을 사용하여 계산한 경우는 경계정점을 사용하여 계산한 경우보다 적은 횟수의 산술연산을 하고 있음을 알 수 있다. 특히 회로망이 커감에 따라 또 소회로망의 수가 증가할 수록 그 차이는 커진다. 정점수가 232개인 회로망의 계산량을 살펴보면 경계정점을 이용한 분할방법은 최적 분할회로망 수가 4—10 사이에 있으며 소경계정점을 이용한 분

表1. 산술 연산횟수의 비교

**Table 1.** Comparison of number of computations.

| 정점수 (N) | 소회로망수(ℓ) | 소회로망의 정점수 (Ni) | 경계정점수(B) | 소경계정점수(Bi) | 분할하지않은 경우 ($N^3$) | 경계정점을이용한계산 | 소경계정점을 이용한계산 |
|---|---|---|---|---|---|---|---|
| 39 | 4 | 8 | 7 | 3.5 | $5.93 \times 10^4$ | $2.91 \times 10^4$ | $1.4 \times 10^4$ |
| 39 | 7 | 4.14 | 10 | 2.86 | $5.93 \times 10^4$ | $4.21 \times 10^4$ | $8.1 \times 10^3$ |
| 232 | 4 | 55.25 | 11 | 5.5 | $1.25 \times 10^7$ | $2.46 \times 10^6$ | $1.87 \times 10^6$ |
| 232 | 10 | 21 | 22 | 4.4 | $1.25 \times 10^7$ | $1.70 \times 10^6$ | $4.38 \times 10^5$ |
| 232 | 18 | 11.2 | 30 | 3.3 | $1.25 \times 10^7$ | $2.62 \times 10^6$ | $3.05 \times 10^5$ |

할방법은 10보다 크다. 이처럼 소경계정점을 이용한 최단경로 계산은 다른 방법을 이용한 경우보다 연산횟수를 적게하고도 최단값을 얻을 수 있으며 특히 기존의 경계정점을 이용한 분할방법에 비해서 동일한 분할된 회로망에서의 경우 연산횟수가 줄어듬은 문론 최적 소회로망수가 더 크기 때문에 더 많은 소회로망으로 분할함으로써 더 적은 연산횟수로 계산을 마치게 된다.

표2에 보여진 것과같이 두 방법 사이의 계산시간 비례는 표1에서와 거의 유사하다. 표1에 비해표2의 수행시간은 회로망이 커짐에 따라 길어지게 되는 것은 가상 메모리(Virtual Memory)를 사용 하였기 때문에 메모리를 많이 사용함에 따라 메모리를 참조하는데 소요되는 시간이 길어지기 때문이다.

표 2. 수행시간의 비교

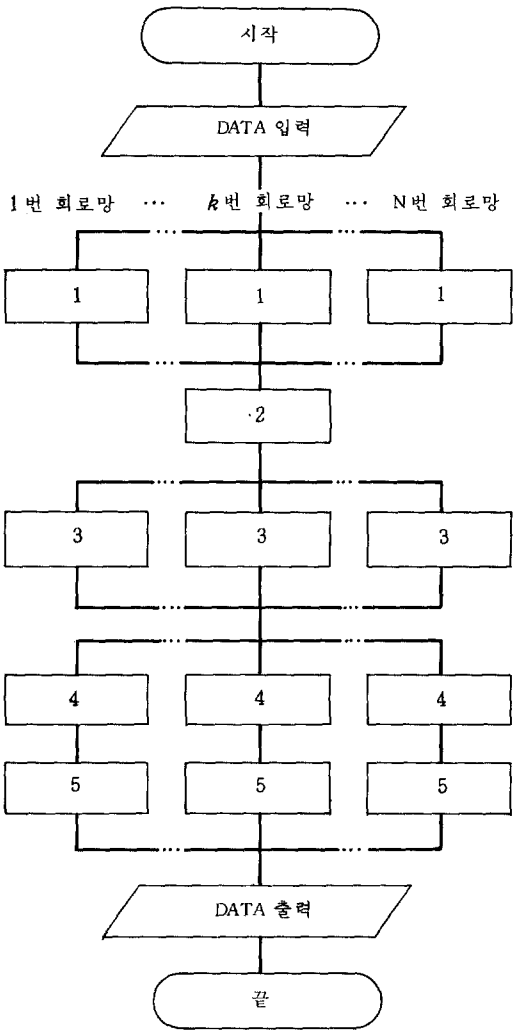
**Table 2.** Comparison of computation time.

| 전체정점수 | 소회로망수 | 경계정점수 | 경계정점을이용한계산 | 소경계정점을이용한계산 |
|---|---|---|---|---|
| 39 | 4 | 7 | 17.4 | 16.9 |
| 232 | 4 | 11 | 3032.0 | 2447.2 |
| 232 | 10 | 22 | 2897.0 | 930.1 |

## 5. 최단경로 계산의 병렬화 및 실현

분할방법에 의한 최단경로 계산은 구조적으로 병렬적인 수행이 가능하다. 그림5는 병렬성을 나타내는 곳을 펼쳐 노시한 것이다. 그림 5에서보듯이 병렬계산이 가능한 과정은 세곳이 있다. 병렬과정1과 병렬과정2는 그 기능과 입출력 변수가 동일한 까닭에 하나로 묶어서 생각할 수 있으며 병렬과정 1과 2에서는 $D_k, S_k$만이 입력으로 사용되나 병렬단계 3에서는 N개 $D_i$, $S_i$ $\leq i \leq N$)행렬을 요구하고 있다. 전산기의 배열은 주종(master — slave)구조로 하였는데 그 이유는 병렬화가 가능하거나 불 가능한 과정들이 확연히 정의되어서, 동시에 행할 수 없는 부분은 주전산기가 처리하고 병렬화가 가능한 과정들은 종전산기를 사용하는 이 논문의 경우에는 종전산기의 수가 변하게 될경우 가장 용이한 방법으로 가감이 가능하기 때문이다. 주전산기에서는 초기행렬의 계산, 회로망의 정보입력, 최단정보들의 출력 및 병렬처리가 불 가능한 부분을 모두 처리하며 종전산기에게 수행의 시작을 지시한다. 종전산기에서는 병렬화가 가능한 부분들을 수행하고 주전산기에 결과를 보낸다.

정보 교환은 PIO (Z—80 Peripheral Input Output Controller)를 이용한 폴링방식을 사용하였다. 그림 5의 병렬과정 1, 2의 경우 시간이득을 계산하여 보면 소경계정점의 수와 소회로 망의 정점수의 합이 n 이라 하고 소회로 망의 수를 ℓ이라할 때 한번 Floyd 방식을 완전히 수행할 때 소비되는 기계어로된 명령어의 갯수는 $85 \times n^3$ 이며 한 종전산기에 입력정보를 모두 전송할 때 필요한명령어는 $75 \times n^2$이다. 이때 병렬수행에 따른 이득은 식(9)와 같다. 첫항은 병렬수행을 하지 않을때 소비되는 명령어의 갯수이며 뒷항은 병렬수행에 소비되는 명령어의 갯수이다.




1, 3 $k$번째 소회로망과 소절단집합 $k$를 합한 회로망에서의 최단경로 계산

2 경계정점사이의 최단경로 계산

4 소경계정점집합 $k$에 속하지 않은 경계정점과 소회로망 I ( I ≠ $k$ ) 사이의 최단경로 계산

5 소회로망 I ( I ≠ $k$ ) 에서부터 소회로망 K까지 최단경로 계산

그림 5. 병렬화된 최단경로 계산의 흐름 선도

**Fig. 5.** Flow chart of parallelized shortest path calculations.

$$\ell \times 85 \times n^3 - \{ ( \ell + 1 ) \times 75 \times n^2 + 85 \times n^3 \} = n^2 \times \{ 85 \times ( \ell - 1 ) \times n - ( \ell + 1 ) \times 75 \} \quad (9)$$

실제 시스템의 구성은 MZ-80을 주전산기로 하고 2대의 CRC-800A 실험용 키트를 종전산기로 사용하였으며 프로그램 언어는 주전산기에는 Hu-Basic을 사용하였고 입출력관계는 어셈블리어를 사용하였으며 종전산기는 모두 어셈블리어를 사용하였다. 동작은 주전산기에서 비용행렬을 입력하여 이들을 그림 5의 순서대로 진행하여 병렬이 가능한 부분에서는 종전산기에 배분하고 다시 이 결과를 모아 모든 정점과 정점사이의 최단경로 계산 결과를 모니터에 출력한다. 실제 39개의 정점을 4개의 경계정점으로 분할된 회로망을 이용하여 최단경로 계산을 하여 보았다. MZ-80 만을 이용한 경우 51분 소비하던 계산과정이 15분으로 단축되었다. 이차이는 Hu-Basic 어보다 어셈블리어가 빠른 것을 감안하더라도 병렬수행으로 얻어지는 시간이득이 큰것을 말해주는 것이다.

## 6. 결 론

정점수가 많은 대형회로망의 최단거리 계산을 정점수의 제곱에 해당하는 메모리 요구와 세제곱에 비례하는 연산수 때문에 분할방법의 사용을 불가피하게 하였다. 기존의 분할법은 각 소회로망의 최단거리계산을 수행할 때 모든 경계정점( Boundary Vertices )을 포함하여 계산하여야 하므로 회로망이 커지고 소회로망수가 증가함에 따라 자연 경계정점의 수도 증가하여 여러대의 소형 컴퓨타를 병렬로 사용하는 병렬수행( Parallel Processing )을 하고자 할때 불편한 점이 많다. 본 논문에서는 소회로망의 최단거리 계산에서 모든 경계정점을 고려하지 않고 계산하고자 하는 소회로망에 직접 연결된 소경계정점집합만을 사용하여 최단거리를 얻어내는 알고리즘을 개발하여 그것이 병렬수행에 효과적으로 사용될 수 있음을 제시하였다. 개발된 알고리즘은 수식이 간편하여 소형 콤퓨타에 이용하기 편리하도록 되어 있고 기존 분할법에 비교하여 계산시간을 단축시킬 수 있다. 실례로 39개 정점 회로망과 232개 정점 회로망을 사용하여 비교 검토한 결과 새로운 알고리즘의 계산속도가 2배 이상 개선됨을 보였다.

새로운 알고리즘을 사용한 병렬수행(Parallel Processing) 실시를 조사하기 위해 한대의 MZ-80 소형 콤퓨타를 주전산기로 두 대의 Z-80 마이크로프로쎄서 시스템을 종전산기로 하여 병렬수행 프로그램을 작성하여 39정점 회로망에 대하여 계산하였다. 그로부터 만족스런 결과를 얻었으며 이는 새로운 알고리즘이 병렬시스템에 사용하기 적합한 것을 입증하는 것이다. 최근 고성능 프로쎄서들이 개발됨에 따라 병렬시스템 또는 다중 프로쎄서 시스템(Mul-

ulti - Processor System)이 대두되고 있으며 본 논문에서는 잘 개발된 알고리즘이 그러한 시스템에 적용되어 계산시간을 개선시키고 값싼 시스템을 이용하여 실시간 계산을 가능하게 함을 보여주었다.

## 참고문헌


1) R.W. Floyd, "Algorithm 97: Shortest Path," Comm. Ass. Comput. Mach. Vol. 5, p.345, 1962.
2) G.B. Dantzig, "On the Shortest Route through a Network," management Sci., Vol. 8, pp.187-190, 1960.
3) A. Rosenthal, "On Finding Shortest Distances in a Graph," Discrete Math. Vol. 10, pp.159-162, 1974.
4) Jang G. Lee, William G. Vogt, and Marlin H. Mickle, "Calculation of the Shortest Paths by Optimal Decomposition," IEEE Tr. on S.M.C. Vol. 12, No.3, pp.1410-1415, 1983.
5) Jang G. Lee, William G. Vogt, and Marlin H. Mickle "Optimal Decomposition of Large-scale Networks," IEEE Tr. on S.M.C. Vol. 9, No. 7, pp.369-375, 1979.
6) Barry R. Borgerson, "The Viability of Multimicroprocessor System," IEEE Trans. Com., pp.26-30, 1976.
7) Panl M. Russo, "Interprocessor Communication for Muti-microcomputer System," IEEE Trans. Com., pp.67-76, 1977.
8) Anita A. Johns, "Experience Using Multiprocessor System," Computer Surveys, Vol.12, pp.121-165, 1980.