經營科學
第19卷 第1號
2002年 5月

최단경로문제의 사전처리 해법에 관한 연구*

명 영 수**

An algorithm for the preprocessing shortest path problem*

Young-Soo Myung**

Abstract

Given a directed network, a designated arc, and lower and upper bounds for the distance of each arc, the preprocessing shortest path problem is a decision problem that decides whether there is some choice of distance vector such that the distance of each arc honors the given lower and upper bound restriction, and such that the designated arc is on some shortest path from a source node to a destination node with respect to the chosen distance vector. The preprocessing shortest path problem has many real world applications such as communication and transportation network management and the problem is known to be NP-complete. In this paper, we develop an algorithm that solves the problem using the structural properties of shortest paths.

Keyword : Preprocessing Shortest Path Problem, Branch and Bound Algorithm.

1. 서 론

최단경로문제(shortest path problem)는 잘 알려진 문제로서 망(network)과 망의 아크(arc)에 거리가 주어져 있을 때, 시작노드(source node)에서 도착노드(destination node)까지의 최단거리의 경로를 구하는 문제이다. 본 논문에서 고려하는 망은 망에 포함된 아크에 방향성이 있는 망(directed network)을 가정한다. 최단경로문제는 현실문제에의 광범위한 응용성 때문에 그 동안 많은 연구가 이루어져 왔으며, 이제까지 개발된 최단경로문제의 해법에 대해서는 Ahuja 등[1], Cherka-

논문접수일 : 2001년 7월 18일 논문게재확정일 : 2002년 2월 6일
* 이 논문은 2000년도 한국학술진흥재단의 연구비에 의하여 연구되었음(KRF-2000-041-C00295).
** 단국대학교 경상학부

ssky 등 [2], Deo와 Pang [3] 등에 잘 정리되어 있다. 최단경로문제에서는 아크의 거리에 대한 정보가 사전에 주어져 있다는 것을 전제로 하고 있다. 그러나 최단경로문제의 많은 응용 예에서는 거리가 단순한 물리적인 거리가 아니라 문제의 특성에 따라 주어진 가중치를 표현하는 경우가 많다. 그리고 가중치의 특성에 따라서는 거리로서 주어진 가중치의 값이 시시각각 변화하는 경우도 있다. 당연히 아크의 거리가 변한다면 최단경로도 따라서 변하게 될 것이다. 본 논문의 관심은 각 아크의 거리가 일정한 범위 내에서 변화할 때 변화하는 거리에 상관없이 어떠한 최단경로에도 포함되지 않는 아크를 발견하는 것이다.

시간에 따라 거리가 변화하고 거리의 변화에 따라 최단경로에 대한 반복적인 계산이 필요한 경우를 생각해 보자. 또한 사전에 발생 가능한 거리를 일정한 범위의 값으로 예측할 수 있다고 가정하자. 사전에 예측한 범위 안에서 아크의 거리가 실현되는 경우, 최단경로에 포함되지 않는 망의 구성요소를 알 수 있다면 이러한 요소를 미리 망에서 제거함으로써 망과 관련된 데이터를 보관하는 데 필요한 메모리는 물론 최단경로의 반복적인 계산에 필요한 시간을 절약할 수 있을 것이다. 최단경로문제의 사전처리(the preprocessing shortest path problem)는 이처럼 각 아크의 거리에 대해 실현 가능한 값의 하한(lower bound)과 상한(upper bound)이 주어졌을 때, 각 아크의 거리가 주어진 범위 내의 값으로 어떻게 실현되더라도 최단경로에는 포함되지 않는, 다시 말해서 최단경로의 계산에는 영향을 주지 않는 망의 구성요소를 찾아내는 문제이다.

최단경로문제의 사전처리는 통신망과 수송망 등에서 많은 응용 예를 발견할 수 있다. 통신망에서 트래픽(traffic)을 보내야 하는 시작노드로부터 트래픽을 받아야 하는 도착노드까지 트래픽을 전송할 경로를 결정할 때, 즉 루팅(routing)방법을 결정할 때를 고려해 보자. 많은 경우에 루팅방법은 시작노드에서 도착노드까지의 최단경로문제를 풀어서 결정하게 된다. 이때 아크의 거리는 물리적인 거리 대신에 트래픽이 아크의 양쪽 노드사이를 통과하는 데 소요되는 지연시간(delay)을 사용한다. 지연시간은 시시각각 끊임없이 변동하게 되므로 통신망의 노드에 해당하는 라우터(router)에서는 지연시간에 대한 정보를 서로 교환하고, 이에 따라 수정된 거리의 정보에 맞춰 반복적으로 최단경로를 계산하게 된다. 특히 최근에 사용되고 있는 OSPF(Open Shortest Path First) 프로토콜처럼 링크-상태 경로선정방법(Link-State Routing)을 사용하는 인터넷 루팅의 경우는, 분산방식으로 운영되는 거리-벡터 경로선정방법(Distance-Vector Routing)을 사용하는 루팅과는 달리, 라우터가 자신이 속한 망의 정보를 모두 가지고 자신으로부터 다른 라우터까지의 최단경로를 직접 계산하게 된다[6]. 따라서 지연시간을 일정 범위 내에서 예측할 수 있다면 최단경로문제의 사전처리 방법을 통하여 최단경로계산에 필요한 망의 크기를 줄일 수 있다. 따라서 각 라우터에서 최단경로를 계산할 때 소요되는 시간은 물론 지연시간의 변화를 알리기 위해서 라우터간에 교환해야 되는 정보의 양도 줄게 되어 통신망의 트래픽은 현저히 감소될 수 있다.

유사한 응용 예는 교통망에서도 발견할 수 있다. 교통망에서는 아크의 거리는 아크에 해당하는 도로를 통과하는데 걸리는 소요시간이며 이는 시간대에 따라 변화한다. 차량에 장착하는 자동운항장치는 시시각각 변화하는 도로의 통과시간을 제공받아서 운전자의 목적지까지 이르는 최단경로를 계산하게 된다. 이 경우에도 우리가 각 도로의 통과시간에 대한 최소시간과 최대시간에 대한 예측을 할 수 있다면, 최단경로문제의 사전처리를 통하여 최단경로 계산에 영향을 주지 않는 망의 구성요소를 제거함으로써 반복적으로 이루어지는 최단경로의 계산시간을 절약할 수 있다.

그 동안 최단경로문제의 민감도분석과 같은 전통적인 사후분석에 대한 연구는 있었으나, 본 논문에서 다루는 사전처리처럼 거리가 불확실성을

갖고 발생할 때 최단경로계산에 영향을 주지 않는 망의 구성요소를 고려하는 연구는 이루어진 적이 없었다. 다만 최근에 Lai와 Orlin [5]에 의해서 이 문제가 NP-complete에 속하는 어려운 문제임이 규명되었을 뿐이다. 특히 Lai와 Orlin [5]은 주어진 망이 사이클이 없는 그래프(acyclic graph)인 경우도 최단경로의 사전처리문제는 NP-complete 임을 밝혔다. 이러한 문제의 난이도에 대한 결과는 주어진 망이 방향성이 있는 경우이며, 주어진 망이 방향성이 없는 경우에 대해서는 아직 난이도가 밝혀져 있지 않다. 물론 방향성이 없는 망에서 사이클이 없는 경우에는 트리(tree)가 되므로 사전처리 문제는 단순하게 해결된다. 이처럼 최단경로문제의 사전처리는 난이도의 규명 외에는 아직 이 문제를 풀기 위한 해법은 제시되지 못하고 있다. 따라서 이 문제의 응용성을 고려할 때 효율적인 해법을 개발하는 것은 꼭 필요한 과제이다. 또한 사전처리문제는 모든 최적화문제에 적용될 수 있으므로 최단경로문제에 대한 사전처리 방법의 연구는 다른 최적화문제에 대한 사전처리를 위한 연구의 시발점이 될 수 있다. 이러한 점에서 본 연구는 최적화분야의 이론적 발전을 위해서도 바람직한 시도라고 생각된다.

본 논문에서는 최단경로의 구조적 특성을 이용해서 각 아크의 거리가 일정한 범위 내의 값을 갖는 경우에 특정 아크가 최단경로에 포함되는지 여부를 판정할 수 있는 검사방법을 개발하고 이를 이용하여 최단경로문제의 사전처리를 할 수 있는 분단탐색법(branch and bound method)을 제시하고자 한다. 최단경로문제의 사전처리는 기본적으로 어떤 한 아크가 최단경로에 포함되는지 여부를 판정하는 것으로 정의되어 있다. 그러나 실제 통신망이나 교통망에서 응용될 때에는 특정 아크가 아니라 최단경로에 포함되지 않는 모든 아크를 발견해야 될 경우도 있다. 이 경우에 특정 아크가 최단경로에 포함되는지 여부를 판정할 수 있는 검사방법을 모든 아크에 동시에 적용하게 되는데, 이때 계산의 중복 없이 가장 효율적으로 적용할 수 있는 방법도 아울러 제시하기로 한다. 그리고 개발된 해법의 실제문제에서의 계산상 효율성을 평가하기 위해서 컴퓨터 프로그램을 작성하여 다양한 크기의 망에서 사전처리 해법의 계산실험을 하기로 한다.

2. 용어의 정의 및 문제의 특성

방향성이 있는 망을 $G=(V,A)$로 표시하고, V는 노드의 집합을 A는 아크의 집합을 표시하기로 한다. 노드 i에서 j로 향한 아크 $a\in A$는 필요에 따라서는 a대신 (i,j)로 표시되기도 한다. 각 아크 $a=(i,j)\in A$의 거리를 d_a 또는 d_{ij} 로 표현하기로 한다. 각 아크 $a\in A$의 실현 가능한 거리의 상한과 하한을 u_a(또는 u_{ij})와 l_a(또는 l_{ij})로 각각 나타내기로 한다. 아크의 거리에 대한 상한과 하한은 양(positive)의 값을 갖는다고 가정하자. 망에는 시작노드 s와 도착노드 t가 사전에 정해져 있다. 아크의 거리가 갖을 수 있는 모든 실현 가능한 값을 표시하기 위하여 실현 가능한 거리-벡터로 구성되어 있는 집합을 $D=\{d\in R^{|A|} | l_a\le d_a\le u_a,\ \forall a\in A\}$로 정의한다. 각 아크의 거리가 특정 실현 가능한 거리-벡터 $d\in D$로 주어졌을 때, $d(i,j)$는 노드 i로부터 노드 j까지의 최단경로의 거리를 표시한다. 같은 방법으로 $l(i,j)$는 각 아크의 거리가 하한에 고정되었을 때 노드 i로부터 노드 j까지의 최단경로의 거리를 표시하고, $u(i,j)$는 각 아크의 거리가 상한에 고정되었을 때 노드 i로부터 노드 j까지의 최단경로의 거리를 표시한다. 노드 s로부터 노드 t까지의 경로를 $s-t$ 경로, 최단경로를 $s-t$ 최단경로라고 부르기로 한다.

앞서 언급한대로 최단경로문제의 사전처리(the preprocessing shortest path problem : PSPP)는 방향성이 있는 망 $G=(V,A)$, 각 아크 $a\in A$의 거리에 대한 상한과 하한 u_a와 l_a, 시작노드 s

와 도착노드 t가 주어졌을 때, 어떤 특정 아크 (i,j)가 $s-t$ 최단경로에 포함되도록 하여 주는 실현 가능한 거리-벡터 $d \in D$가 존재하는 지를 판정하는 문제이다. 만약에 그러한 실현 가능한 거리-벡터가 존재하지 않는다면 아크 (i,j)는 망에서 사전에 제거되어도 사후적으로 아크들의 거리가 어떻게 실현될지라도 $s-t$ 최단경로를 구하는 것에는 영향을 주지 않을 것이다. 이러한 아크를 *열등아크*(dominated arc)라고 부르기로 한다. 따라서 PSPP는 어떤 특정 아크가 열등아크인지를 판정하는 문제로도 정의할 수 있다.

PSPP를 푸는 단순한 방법으로는 모든 실현 가능한 거리-벡터에 대해서 최단경로문제를 풀어서 특정 아크의 포함 여부를 검사하는 것이다. 그러나 이 방법은 실현 가능한 벡터의 수가 무한하게 많으므로 현실적으로는 불가능한 접근방법이다. 다음의 정리는 PSPP를 풀기 위해서 실현 가능한 거리-벡터 모두를 고려할 필요는 없다는 사실을 알려 준다.

정리 1 어떤 특정 아크 (i,j)가 열등아크가 아니라고 가정하자. 그러면, 각 아크의 거리가 하한 또는 상한으로 주어진 거리-벡터 중 적어도 하나의 거리-벡터에 대해서 구해진 $s-t$ 최단경로에 (i,j)가 포함된다.

증명

아크 (i,j)가 열등아크가 아니라면 어떤 실현 가능한 거리-벡터 d에 대한 $s-t$ 최단경로 P에 (i,j)가 포함될 것이다. 이제 P에 속한 아크에 대해서는 거리가 하한으로 나머지 아크에 대해서는 거리가 상한으로 주어지는 또 다른 실현 가능한 거리-벡터 d'을 고려해 보자. 그러면 P는 d'에 대해서도 $s-t$ 최단경로가 됨은 자명하며 따라서 정리가 성립된다.

정리 1을 이용하면 유한한 개수의 거리-벡터에 대해서만 최단경로문제를 풀어도 PSPP를 풀 수 있음을 알 수 있다. 하지만 여전히 고려해야 될 벡터의 수는 노드의 수에 대해서 지수함수로 표시되는 엄청난 많은 수이므로 고려 대상 벡터에 대해서 일일이 최단경로문제를 푸는 접근방법은 여전히 비현실적인 접근방법이다. 물론 PSPP가 NP-complete임을 감안하면 PSPP를 풀기 위해서 고려해야 될 거리-벡터의 수가 노드의 수에 대해서 다항함수(polynomial function)로 표시될 수는 없을 것이다. 왜냐하면 이것이 가능하면 PSPP를 다항식시간(polynomial time)에 풀 수 있게 되므로 PSPP가 NP-*complete*이라는 사실에 모순되기 때문이다.

앞서 지적한대로 고려해야 될 벡터에 대해서 일일이 최단경로문제를 푸는 접근방법은 좋은 방법은 아니나, PSPP의 정확한 해, 즉 어떤 아크가 열등아크인지를 정확히 판정하기 위해서는 최악의 경우는 모든 고려 대상이 되는 벡터를 점검해야 된다. 그러나 고려 대상이 되는 벡터를 점검할 때 좀 더 현명한 방법을 사용한다면, 항상은 아니더라도 많은 경우에 가능한 적은 수의 최단경로문제를 풀고서도 정확한 판정을 할 수 있을 것이다. 이를 위해서 다음 장에서는 최단경로의 구조적 특성을 이용해서 특정 아크가 열등아크인지 검사하는 방법을 제시하고, 이를 이용하여 PSPP의 정확한 해를 찾는 분단탐색법(branch and bound method)을 개발하기로 한다.

3. 분단탐색법을 이용한 최단경로문제의 사전처리를 위한 해법

2장의 정리 1에 의하면, 어떤 특정 아크 (i,j)가 열등아크인지 여부를 판정하기 위해서는 실현 가능한 거리-벡터 중 각 아크의 거리가 하한 또는 상한을 갖는 거리-벡터들에 대해서 구해진 $s-t$ 최단경로만을 고려하면 된다. 즉 아크 (i,j)가 그러한 $s-t$ 최단경로에 포함되는지를 검색함으로써 (i,j)가 열등아크인지 여부를 판정할 수 있다. 여기서는 이러한 검색을 좀 더 효율적으로 수행하기 위해서 분단탐색법의 방법을 이용하기

로 한다.

3.1 분단탐색나무(branch and bound tree)의 구조

PSPP의 경우는 의사결정문제(decision problem)의 형태를 갖고 있으므로 분단탐색과정은 최적화문제의 경우와 비교해서 약간의 차이가 있다. PSPP에 대한 분단탐색법에서의 부분문제(partial problem)는 일부 아크의 거리를 하한 또는 상한에 고정한 PSPP가 된다. 즉 여기서는 실현 가능한 거리-벡터들을 분할해 가면서, 분할된 벡터의 집합에 대해서 고려 대상의 아크가 열등아크인지 여부를 판정하는 것이다. 따라서 상위문제에서 분단에 사용할 아크를 정한 뒤, 이 아크를 하한에 고정한 부분문제와 상한에 고정한 부분문제 두 개로 분할하게 된다. 설명을 위해서 다음과 같은 기호를 추가로 정의하자. (SP_0)는 원 문제를, (SP_k)는 분단탐색법에서 생성된 k번째 부분문제를 표시한다. (SP_k)는 원 문제와 비교할 때 일부의 아크가 상한 또는 하한에 고정되어 있는 PSPP이다. A_k^l는 거리가 하한에 고정된 아크의 집합을, A_k^u는 거리가 상한에 고정된 아크의 집합을, A_k^f는 고정되지 않은 아크의 집합을 나타내는 것으로 정의한다. 즉 $A_k^f = A \backslash (A_k^l \cup A_k^u)$이다. 그리고 (SP_k)에서 고려하는 실현 가능한 거리-벡터들의 부분집합 $D_k \subseteq D$를 다음과 같이 정의한다 :

$$D_k = \{d \in R^{|A|} \mid d_a = l_a \ \forall a \in A_k^l,\ d_a = u_a \ \forall a \in A_k^u,\ l_a \le d_a \le u_a \ \forall a \in A_k^f\}$$

또한 최적화문제와는 달리 부분문제의 최적해에 대한 상한과 하한은 의사결정문제의 성격상 존재하지 않는다. 그러나 우리는 부분문제와 원 문제 사이에 다음과 같은 관계가 성립한다는 사실을 이용하여 분단탐색법의 효율성을 높일 수 있다.

관찰 1 어느 아크 (i, j)가 (SP_k)에서 열등아크가 아니면 당연히 (i, j)는 원 문제에 대해서도 열등아크가 아니다. 반면에, 아크 (i, j)가 (SP_k)의 열등아크인 경우에는, (i, j)가 원 문제에 대해서 열등아크인지는 알 수 없고, 다만 모든 거리-벡터 $d \in D_k$에 대해서 구해진 $s-t$ 최단경로에 아크 (i, j)는 포함되지 않는다는 사실은 알 수 있다. 따라서 전자의 경우에는 PSPP의 최종해를 얻게 되며, 후자의 경우에는 D_k에 속한 거리-벡터에 대해서는 더 이상 검색할 필요가 없게 되므로 분단탐색나무에서 D_k에 해당하는 가지(branch)는 탐색이 종료된다.

위에 기술한 관찰의 내용을 적절히 이용하여 분단탐색법의 효율성을 높이기 위해서 우리는 다음 절에서 두 개의 검사 방법을 개발한다. 이 두 검사는 현재의 부분문제에 대해서 고려 대상 아크가 열등아크인지 아닌지를 항상은 아니지만 상당히 많은 경우에 판정해 주는 방법이다.

3.2 열등아크 판정을 위한 두 가지 검사

어떤 아크가 열등아크임을 보이기 위해서는 모든 거리-벡터 $d \in D$에 대해서 구해진 $s-t$ 최단경로에 해당 아크가 포함되지 않는다는 것을 보여야 한다. 한편 어떤 아크가 열등아크가 아닌지를 보이기 위해서는 그 아크가 $s-t$ 최단경로에 포함되도록 하는 거리-벡터를 제시함으로써 가능하다. 여기에서 우리는 두 가지 검사를 개발한다. 첫째 검사는 특정 아크가 열등아크임을, 둘째 검사는 열등아크가 아님을 확인하기 위한 것이다. 첫째 검사는 다음과 같은 사실에 근거하여 만들어졌다.

정리 2 어떤 아크 (i, j)에 대해서 다음 중 하나가 성립하면 아크 (i, j)는 열등아크이다.

(i) $l_{ij} > u(i, j)$,

(ii) $l(s, i) + l_{ij} > u(s, j)$,

(iii) $l_{ij}+l(j,t) > u(i,t)$,

(iv) $l(s,i)+l_{ij}+l(j,t) > u(s,t)$.

증명

모든 경우가 성립됨은 쉽게 알 수 있으므로 지면의 절약을 위해서 (i)의 경우만 간단히 설명하기로 하자. $l_{ij} > u(i,j)$라는 사실로부터 $u(i,j)$값에 해당하는 i에서 j까지의 최단경로(이 경로를 P_{ij}^{u}로 표시한다)에는 아크 (i,j)가 포함되지 않았음이 틀림없다. 임의의 실현 가능한 거리-벡터 d를 고려해 보자. 그리고 이 벡터에 대해서 아크 (i,j)를 지나가는 s에서 t까지의 경로 중에 거리가 최소가 되는 경로를 P라고 하자. 이해를 돕기 위해서 <그림 1>에 경로들의 관계를 나타내었다. 그러면 P의 아크 (i,j)를 i에서 j까지의 경로 P_{ij}^{u}로 대체한 경로는 P보다 거리가 짧게 된다. 따라서 P는 벡터 d에 대해서 s에서 t까지의 최단경로가 될 수 없다.

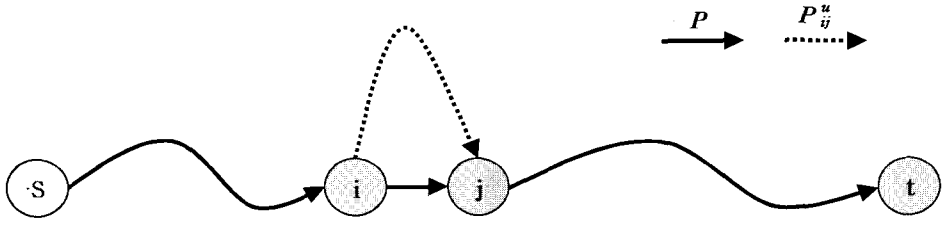
특정 아크가 열등아크인지를 판정하는 첫째 검사는 다음과 같이 정리 2의 네 가지 조건을 점검하는 것이다.

검사 1 : 정리 2의 네 가지 조건을 점검하여 이 중 어느 하나라도 성립하면 아크 (i,j)를 열등아크로 판정한다.

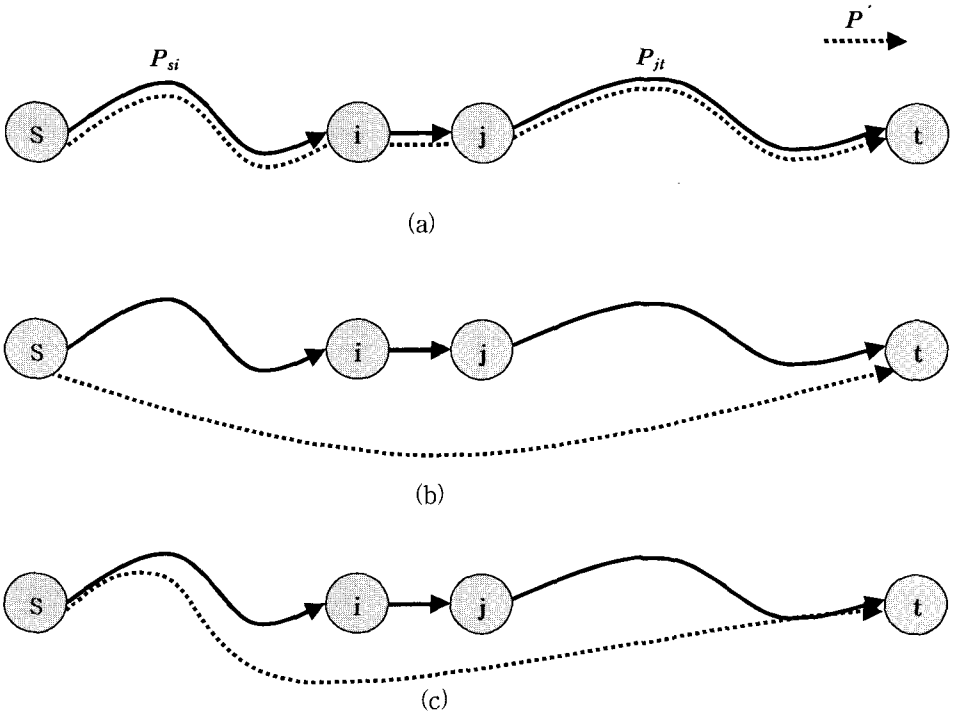
앞서 언급한대로 어떤 아크 (i,j)가 열등아크가 아닌지를 보이기 위해서는 그 아크가 $s-t$ 최단경로에 포함되도록 하여 주는 거리-벡터를 제시하여야 한다. 물론 이러한 벡터가 존재하더라도 이 벡터를 정확하게 찾아내는 것은 원 문제를 푸는 것과 같은 난이도를 갖게 될 것이므로 간단한 방법으로 원하는 벡터를 찾는 방법은 불가능할 것이다. 우리가 개발한 검사 2는 우리가 찾는 벡터의 조건에 가장 근접할 수 있는 벡터를 간단한 절차를 통하여 구성하고 이 벡터에 의해서 구해진 $s-t$ 최단경로에 아크 (i,j)가 포함되는지를 알아보는 것이다.

검사 2 : 일단 모든 아크의 거리를 하한(l_a)에 놓고 s에서 i까지의 최단경로 P_{si} 와 j에서 t까지의 최단경로 P_{jt}를 구한다. 구해진 P_{si}와 P_{jt}를 이용하여 다음과 같이 거리-벡터 d를 구성한다 : P_{si}와 P_{jt}에 포함된 아크 및 아크 (i,j)에 대한 거리는 하한으로, 나머지 아크에 대한 거리는 상한으로 할당한다. 이와 같이 구성한 벡터 d에 대한 $s-t$ 최단경로에 아크 (i,j)가 포함되면 아크 (i,j)는 열등아크가 아니라고 판정한다.

검사 1과 2는 분단탐색나무의 첫째 노드의 문제 즉 원 문제 (SP_0)를 상정하여 기술되어 있다. 그러나 분단탐색나무의 중간단계에서 발생하는 부분문제 (SP_k)와 원 문제 (SP_0)의 관계를 고려하면, 부분문제 (SP_k)에 대해서 검사 1과 2를 실행하는 방법도 원 문제에 실행하는 방법과 차이가 없음을 알 수 있다. 즉 (SP_k)는 원 문제에서 A_k^l에 포함된 아크의 거리가 하한에 A_k^u에 포함된 아크의 거리는 상한에 고정되어 있는 경우이므로, (SP_k)에서 A_k^l에 포함된 아크의 거리의 상한과 하한을 하한에 일치시키고, A_k^u에 포함된 아



〈그림 1〉 열등아크의 충분조건 (i)의 경우



〈그림 2〉 발생 가능한 P_{si}, P_{jt} 및 P'의 관계

크에 대해서는 거리의 상한과 하한을 상한에 일치시키면, (SP_k)는 원 문제와 비교해서 일부 아크의 상한과 하한만 변화된 PSPP가 된다. 따라서 부분문제 (SP_k)에 대해서 검사 1과 2를 실행하는 것은 변환된 PSPP의 원 문제에 검사 1과 2를 실행하는 것과 같다.

3.3 분단에 사용될 아크의 선정

앞 절에서 소개한 두 검사를 수행하고도 열등아크 여부를 판정하지 못하는 경우에는 현재의 부분문제에서 상한 또는 하한에 고정되어 있지 않은 아크를 선택하여 현재의 문제를 두 개의 부분문제로 분할하게 된다. 이 때 어떤 아크를 선택할 것인지는 분단탐색법의 계산결과에 영향을 주는 중요한 요소이다. 우리의 해법에서는 검사 2의 결과를 이용하여 분단에 사용될 아크를 결정하게 된다. 검사 2에서 P_{si}와 P_{jt}에 포함된 아크와 아크 (i, j)의 거리는 하한에, 나머지 아크의 거리는 상한에 고정한 거리-벡터에 대해서 구한 $s-t$ 최단경로를 P'이라고 하자. 그러면 발생 가능한 P_{si}, P_{jt} 및 P'의 관계는 <그림 2>와 같다. (a)의 경우는 P'이 P_{si}와 아크 (i, j), 그리고 P_{jt}와 일치하는 경우이며 이 경우는 검사 2에 의해서 아크 (i, j)가 열등아크가 아닌 것으로 판정되는 경우이다. 한편 (b)의 경우는 실제로 발생하지 않는다. 왜냐하면 검사 2에서 구성한 거리-벡터의 성격상 P'의 모든 아크의 거리는 상한에 고정되어 있으므로 정리 2의 조건 (iv)를 만족하게 되는데, 이런 경우라면 이미 검사 1에서 열등아크로 판정되었을 것이기 때문이다.

따라서 검사 1과 2를 마친 후에도 아크 (i, j)의 열등아크 여부를 판정 못하였다면 검사 2에서

의 P_{si}, P_{jt} 및 P' 의 관계는 <그림 2>의 (c)와 같이 나타날 것이다. 즉 P_{si}와 P_{jt}에 포함된 아크와 아크 (i, j)를 포함하는 아크의 부분집합을 A'로 표기하면, $P' \cap A' \neq \emptyset$가 성립되게 된다. 우리의 분단탐색법에서의 분단을 위한 아크의 선택은 $P' \cap A'$에 속하는 아크 중에서 상한과 하한에 고정되어 있지 않은 아크, 다시 말해서 $P' \cap A' \cap A_k^f$에 속한 아크를 분단아크(branching arc)로 선택하는 것이다. 항상 $P' \cap A' \cap A_k^f \neq \emptyset$이 성립된다. 왜냐하면 이런 경우는 앞서 (b)에 대한 설명과 마찬가지 이유로 (SP_k)에 대해서 정리 2의 조건 (iv)를 만족하게 되어서 이미 검사 1에서 열등아크로 판정되었을 것이기 때문이다.

우리의 분단탐색해법에서는 부분문제를 분할할 때 분단아크를 하한에 고정시키는 문제를 먼저 생성시킨다. 또한 컴퓨터의 메모리를 적게 사용하기 위하여 분단탐색나무의 부분문제를 찾는 순서를 후입선출(Last-in Fitst-out)의 형태로 하는 깊이-우선-처리(Depth-First Search) 방식을 사용하였다.

4. 다수의 열등아크를 동시에 찾는 방법

서론에서 언급한 대로, 최단경로문제의 사전처리는 통신망과 수송망 등에서 거리의 변화에 따라 최단경로에 대한 반복적인 계산이 필요하고, 사전에 실현 가능한 거리를 일정한 범위의 값으로 예측할 수 있을 때 활용된다. 이 때 사전처리의 목적은 예측한 범위 안에서 아크의 거리가 실현되는 경우 최단경로에 영향을 주지 않는 망의 구성요소를 사전에 파악하는 것이다. 이러한 요소를 미리 망에서 제거함으로써 망과 관련한 데이터의 보관에 필요한 메모리는 물론 최단경로의 반복적인 계산에 필요한 시간을 절약할 수 있기 때문이다. 이처럼 실제 응용 예에서는 특정 아크가 아니라 최단경로에 포함되지 않는 모든 아크를 발견해야 될 경우도 있다.

이제까지 3장에서 소개한 해법은 어떤 특정 아크가 열등아크인지 아닌지를 판정하는 문제를 전제로 하였다. 그러나 특정한 아크가 아니라 주어진 망에서 필요 없는 모든 아크를 알아내는 것이 목적일 때는, 3장의 해법을 각각의 아크에 일일이 실행할 수도 있지만 이보다 좀 더 효율적인 방법이 있다면 유용할 것이다. 물론 열등아크를 모두 정확히 가려내고자 한다면 각각의 아크에 대해서 분단탐색법을 실행하여야 할 것이다. 여기에서 우리가 고려하려는 것은 3장에서 열등아크인지를 검증하는 데 사용한 검사 1을 모든 아크에 동시에 적용하여 열등아크로 판정되는 아크를 우선적으로 가려내는 방법에 대해서이다. 우리의 관심은 검사 1을 모든 아크에 실행할 때, 계산의 중복 없이 가장 효율적으로 적용할 수 있는 방법을 모색하는 것이다. 물론 검사 1을 통하여 열등아크로 판정되지 않은 아크들에 대해서는 분단탐색법을 적용함으로써 가능할 것이다.

우선 정리 2의 조건 (ii), (iii), (iv)를 모든 아크에 적용하기 위해서는 모든 노드 $v \in V$에 대해서 $l(s, v)$와 $u(s, v)$ 그리고 $l(v, t)$와 $u(v, t)$를 계산해야 한다. 이를 위해서 전진적 다익스트라 해법(forward Dijkstra's algorithm)을 거리의 하한과 상한을 각각 적용하여 s에서 각 노드 $v \in V \setminus \{s\}$까지 두 번 수행하고, 또한 t에서 각 노드 $v \in V \setminus \{t\}$까지 후진적 다익스트라 해법(backward Dijkstra's algorithm)을 두 번 수행하면 된다. 정리 2의 조건 (i)을 적용하기 위해서는 모든 아크 $(i, j) \in A$에 대해서 $u(i, j)$를 구해야 되는데 이를 위해서는 최단경로해법을 $|V|$번 수행하든지, $O(|V|^3)$의 계산시간이 필요한 Floyd-Warshall의 해법을 사용함으로써 가능하다[1]. 다음 장의 계산결과에서 보이겠지만 이러한 확장된 검사 1은 적은 계산시간에 상대적으로 많은 열등아크를 효율적으로 발견한다. 물론 열등아크를 모두 정확히 가려내고자 한다면 위에서 제시한 확장

된 검사 1에서 열등아크로 판정되지 않은 아크들에 대해서 3장에서 소개한 검사 2와 함께 분단탐색법을 적용함으로써 가능할 것이다.

5. 계산실험 및 결과 분석

5장에서는 본 연구에서 제시된 최단경로문제의 사전처리를 위한 해법들이 실제크기의 문제를 풀 때에 얼마나 효율적인지를 평가하기로 한다. 이를 위해서 3장과 4장에서 개발된 해법을 C 언어를 통하여 프로그램으로 구현하였다. 계산시험에 사용할 대상문제를 만들기 위한 표준적인 네트워크 생성 프로그램도 다음과 같이 구현하였다. 먼저 (100×100) 사각형 모양의 격자에 필요한 수만큼의 노드를 임의로 위치시키고, 임의의 노드간에 사전에 정한 개수의 아크를 위치시켰다. 계산실험을 위하여 두 종류의 문제 그룹을 만들었는데 첫째 그룹은 특별한 구조적 특성이 없는 일반적인 그래프이며 둘째 그룹은 사이클이 없는 그래프(acyclic graph)이다. 또한 노드의 수와 아크의 수의 비율, 즉 그래프의 밀도(density)가 서로 다른 다양한 망을 생성하였다. 그리고 모든 실험은 200Mhz CPU를 가진 펜티엄급 PC에서 실행되어 졌다.

우선 3장에서 제시한 분단탐색법을 실험하기 위해서는 하나의 아크에 대해서 분단탐색법을 적용하게 되는데 우리는 생성된 망별로 모든 아크에 대해서 일일이 분단탐색법을 적용하였다. 즉 10000개의 아크를 갖는 네트워크에 대해서는 10000개의 문제를 풀게 된다. <표 1>~<표 3>에 이러한 계산결과가 나타나 있다. <표 1>에는 노드의 수가 50개인 일반적인 그래프에 대해서 그래프의 밀도와 아크의 거리에 주어진 상한과 하한의 크기에 따른 계산결과가 나타나 있다. 각 네트워크에서 아크의 거리의 하한은 100에서 1000사이의 값으로 임의로 생성되었고 둘째 열에 표시된 상한-하한의 값을 문제별로 모든 아크의 하한에 더해서 상한을 설정하였다. 표에는 아크별로 사전처리 해법을 적용한 결과 별도의 분단과정 없이 원 문제에서 검사 1 또는 2로 열등아크 여부를 판정한 경우와 분단과정이 필요한 경우를 나누어서 표시하였으며 분단과정이 필요했던 경우도 계산시간에 따라 3가지로 분류하여 표시하였다. <표 1>에서 나타난 것처럼 상한과 하한의 차이가 커지면 각 아크가 열등아크가 될 확률은 낮아진다. 그리고 아크의 수가 늘어나면 계산시간이 늘어나는 것은 당연한 결과이나 상한과 하한의 차이에 따른 계산시간의 변화는 일정한 방향으로 증감되지 않고 그래프의 밀도에 따라 다양하게 나타날 수 있

〈표 1〉 상한과 하한의 변화에 따른 계산결과

문제크기		상한 - 하한	열등아크		분단탐색 불필요		분단탐색 필요					
							< 1 초		1 ~ 10초		> 10 초	
노드 수	아크 수		개수	(%)	개수	(%)	개수	(%)	개수	(%)	개수	(%)
50	100	200	89	89	30	30	70	70	0	0	0	0
		1000	59	59	39	39	14	14	36	36	11	11
		3000	58	58	42	42	5	5	11	11	42	42
	200	200	148	74	132	66	66	33	2	1	0	0
		1000	22	11	135	67.5	43	21.5	3	1.5	19	9.5
		3000	16	8	136	68	46	23	5	2.5	13	6.5
	500	200	380	76	384	76.8	89	17.8	5	1	22	4.4
		1000	39	7.8	343	68.6	120	24	6	1.2	21	4.2
		3000	17	3.4	360	72	135	27	3	0.6	2	0.4
	750	200	554	73.9	598	79.7	131	17.5	21	2.8	17	2.3
		1000	63	8.4	576	76.8	144	19.2	3	0.4	27	3.6
		3000	30	4	603	80.4	143	19.1	1	0.1	3	0.4

〈표 2〉 대규모의 구조적 특성이 없는 일반그래프에 대한 계산결과

문제크기		분단탐색 불필요		분단탐색 필요					
				< 1 초		1 - 10초		> 10 초	
노드 수	아크 수	개수	(%)	개수	(%)	개수	(%)	개수	(%)
100	400	251	62.8	146	36.5	3	0.8	0	0.0
	2500	2113	84.5	339	13.6	27	1.1	21	0.8
200	800	400	50.0	94	11.8	58	7.3	248	31.0
	10000	8951	89.5	715	7.2	253	2.5	81	0.8

〈표 3〉 대규모의 사이클이 없는 그래프에 대한 계산결과

문제크기		분단탐색 불필요		분단탐색 필요					
				<1초		1 - 10초		>10초	
노드 수	아크 수	개수	(%)	개수	(%)	개수	(%)	개수	(%)
100	400	397	99.3	3	0.8	0	0.0	0	0.0
	2500	2476	99.0	24	1.0	0	0.0	0	0.0
200	800	760	95.0	39	4.9	1	0.1	0	0.0
	10000	9741	97.4	243	2.4	14	0.1	2	0.0

음을 알 수 있다.

<표 2>와 <표 3>에는 큰 규모의 일반적인 그래프와 사이클이 없는 그래프에 대한 계산시간의 비교 결과가 나타나 있다. 여기서는 각 아크의 상한과 하한도 난수를 이용하여 임의로 발생시켰다. <표 2>와 <표 3>에서 나타난 것처럼 사이클이 없는 그래프의 경우가 계산이 수월함을 알 수 있다.

마지막으로 <표 4>에는 열등아크를 가능한 많이 한 번에 골라내는 확장된 검사 1의 적용결과가 나타나 있다. (i)에서 (iv)로 표시된 열은 정리 2의 조건 (i)에서 (iv)에 해당하는 검사에서 발견된 열등아크의 수를 표시한다. 아크에 따라서는 둘 이상의 조건에서 모두 발견되는 경우도 있다. 종합이라고 표시된 열에서는 아크 중에서 중복되어 계산된 아크를 제거했을 때 검사 1에서 열등아크로 판명된 아크의 수를 나타낸다. <표 4>의 수치는 동일한 규모의 네트워크에 대한 5개의 서로 다른 문제의 결과를 평균한 결과이다.

〈표 4〉 열등아크를 동시에 찾아내는 해법의 계산실험결과(5문제의 평균치)

문제크기		조건별로 판정한 열등아크의 개수				종 합		계산시간 (초)
노드 수	아크 수	(i)	(ii)	(iii)	(v)	개수	(%)	
100	400	65.4	108.8	101.2	154.2	176	44.0	0.25
	2500	1436.2	1789.6	1789.2	2027	2092.6	83.7	0.44
200	800	95.6	184.8	164.8	192.2	258	32.3	1.85
	10000	6502	7575.8	7290.4	7854.6	8262.8	82.6	5.19
300	1200	110.6	227	188.8	334.2	387.8	32.3	6.02
	22500	15462.4	16484.2	17212.8	18471.8	19491.6	86.6	24.32
400	1600	135.8	179.6	296.6	338.2	455	28.4	14.34
	40000	28447.4	31176.2	31729.8	35247.2	35588.4	89.0	72.61
500	2000	139.8	315.6	215	225	465.4	23.3	27.91
	62500	45717	50654.6	50297.8	56788.4	57355.4	91.8	170.02

계산결과에 나타난 것처럼 본 논문에서 제시한 해법은 적절한 시간 내에 대다수의 최단경로문제의 사전처리를 수행하였다. 하지만 서론에서 밝힌 것처럼 이론적으로 최단경로의 사전처리문제는 NP-complete이며 실제 계산실험에서 10초 이상 경과된 문제 중에는 100초 이상이 경과되는 등 쉽게 판정되지 않는 문제도 다수 발견되었다. 그리고 계산결과에서도 나타났지만 아크의 거리-벡터의 구조에 따라서는 많은 수의 열등아크가 존재할 수 있다는 점이다. 따라서 이러한 경우에 최단경로문제의 사전처리는 우리가 기대한대로 상당한 효과가 있을 것임을 짐작할 수 있다.

6. 결 론

최단경로문제의 사전처리는 망에서 아크의 거리가 확정적인 값으로 주어지는 대신에 실현 가능한 값에 대한 범위로 주어지는 경우에 각 아크의 거리가 주어진 범위 내의 어떠한 값으로 실현되더라도 최단경로에 포함되지 않는 열등아크를 찾아내는 것이다. 본 연구에서는 이러한 최단경로문제의 사전처리 방법을 제시하였다. 최단경로문제의 사전처리는 시간에 따라 거리가 변화하고 거리의 변화에 따라 최단경로에 대한 반복적인 계산이 필요한 통신망과 교통망의 운영에서 많은 응용 예를 발견할 수 있으나 최근에야 Lai와 Orlin에 의해서 이 문제가 NP-complete에 속하는 어려운 문제임이 규명되었을 뿐 이 문제를 풀기 위한 해법은 제시되지 못하였었다. 본 논문에서는 최단경로의 구조적 특성을 이용해서 각 아크의 거리가 사전에 주어진 범위 내의 값을 갖는 경우에 특정 아크가 최단경로계산에 필요한지 여부를 판정할 수 있는 방법을 개발하고 이를 이용하여 최단경로문제의 사전처리를 위한 분단탐색법(branch and bound method)을 제시하였다. 다양한 망에 대한 계산실험을 통해서 제시된 해법의 효율성을 확인하였으며 망에 따라서는 많은 수의 열등아크가 존재함도 확인하였다. 앞서 언급한대로 이 문제의 응용성을 고려할 때 좀 더 효율적인 해법을 개발하는 것은 앞으로도 필요한 연구이며 다른 최적화문제에 대한 사전처리도 최적화분야의 이론적 발전을 위해서 시도해 볼 만한 연구과제로 생각된다.

참 고 문 헌

[1] Ahuja, R.K., T.L. Magnanti and J.B. Orlin, Network Flows : theory, algorithms, and applications, Prentice-Hall, New Jersey, 1993.

[2] Cherkassky, B.V., A.V. Golberg and T. Radzik, "Shortest paths algorithms : Theory and experimental evaluation," Mathematical Programming 73, (1996), pp.129-174.

[3] Deo, N. and C.-y. Pang, "Shortest-path algorithms : Taxanomy and annotation," Networks 14, (1984), pp.275-323.

[4] Garey M.R. and D.S. Johnson, Computers and intractability : a guide to the theory of NP-completeness, W.H. Freeman and Co., New York, 1979.

[5] Lai T.-C. and J.B. Orlin, "The Complexity of Preprocessing," Unpublished manuscript (1999).

[6] Streenstrup, M.E., Routing in communications networks, Prentice-Hall, New Jersey, 1995.