■ 論 文 ■

회전제한이 있는 도로망을 위한 고속 최적경로 알고리즘

A fast shortest path algorithm for road networks having turn prohibitions

성 태 경 (충남대학교 전기공학과 교수)
명 선 영 ((주)네비콤 선임연구원)
홍 원 철 ((주)네비콤 선임연구원)

목 차

Ⅰ. 서론
Ⅱ. 기존의 회전제한을 고려한 경로계산 알고리즘들의 비교
Ⅲ. 회전제한이 있는 도로망을 위한 제안된 도로망 모델과 알고리즘
1. 회전제한이 있는 도로망을 위한 제안된 도로망 모델과 알고리즘
2. 제안된 도로망 모델과 알고리즘에 의한 최적경로계산 예
Ⅳ. 실험결과
Ⅴ. 결론
참고문헌

요 약

대도시에는 회전제한을 갖는 교차로가 다수 존재하며, 도로망에 있어서 최적경로를 계산할 때 회전제한을 고려해야 한다. 본 논문에서는 회전제한을 갖는 도로망에서 새로운 경로계획 방법을 제안하였다. 각 회전제한 노드에 대하여 U턴 혹은 P턴을 이용하여 대체경로를 미리 계산하고 이를 도로망 데이터베이스에 포함하는 도로망 모델을 제안하였다. 제안된 모델은 기존의 도로망 모델에서 회전제한을 표현하기 위하여 사용하던 가상 노드를 포함하지 않기 때문에 효율적이다. 제안된 모델을 이용하여 최적경로를 계산하기 위하여 새로운 최적경로 알고리즘을 제안하였다. 회전제한이 있는 노드에 대하여 마디를 정의하였으며, 이를 이용하여 대체경로의 비용과 타 경로와의 비용을 비교할 수 있도록 하였다. 제안된 경로계획 방법은 회전제한이 존재하는 도로망에 대하여 최적경로를 고속으로 계산할 수 있는 효율적인 방법이다.

Ⅰ. 서론

최적 경로 문제는 망(network)을 이용하는 교통, 통신 시스템 등에서 제기되는 문제로서 지난 30년 동안 최적화 대상 및 응용 분야에 따라 연구가 진행되어 왔다. 이 문제는 적절히 묘사된 망 모델과 최적 경로 알고리즘을 이용하여 해결할 수 있으며, 각종 응용 분야에서 제기되는 그 분야만의 독특한 문제를 해결할 수 있도록 설계된 다양한 최적 경로 알고리즘들이 발표되었다.(Cherkassky, 1993; Gallo, 1988; Deo, 1984; Sheffi, 1985) 교통 분야에서도 도로망의 특성들을 고려한 알고리즘에 대한 연구가 활발히 진행되어 현재는 최적 경로 알고리즘 분류의 한 분야를 이루고 있다. 최근에는 차세대 교통 체계로서 세계적으로 주목을 받고 있는 지능 교통 시스템(intelligent transportation systems)의 첨단 여행자 정보 시스템(advanced traveler information system)이나 주행 안내 시스템(route guidance system) 등을 위한 최적 경로 알고리즘의 개발이 활발하게 진행되고 있다.(Zhao, 1997; Walker, 1990; 이승환, 1996; 김익기, 1998) 지능 교통 시스템에서 차량의 정보단말로 이용되는 차량 항법 시스템은 경로 안내 서비스를 위해 CD-ROM 등의 저장매체에 탑재된 경로 계산용 도로망을 이용하여 목적지까지의 최적경로를 출발점 혹은 주행 도중 필요한 시기에 신속히 계산해야 한다. 차량 항법 시스템의 제한된 메모리를 이용하여 경로계산을 할 때 도로망 데이터의 크기와 알고리즘의 성능은 계산시간의 중요한 요소가 된다.

도로망을 모델링할 때에는 도로의 연결성, 방향성, 링크(link)를 통과하기 위한 비용, 그리고 교차로에서의 회전 비용이나 회전 제한 등을 고려해야 한다. 링크 통과 비용은 최적 경로의 의미에 따라 주행 거리나 여행 시간 등을 고려하여 구한다. 도로망이 갖는 독특한 특성으로서 교차로에서의 회전 제한을 들 수 있다. 특히 대도시의 복잡한 도로망에서는 회전을 제한하는 교차로가 흔히 발견되며, 최적 경로를 계산함에 있어서 회전 제한을 고려하는 것이 중요한 요소가 된다. 도로망의 회전 제한을 고려하기 위하여 기존에는 도로를 노드로 도로간의 회전을 링크로 표현하거나 교차로의 노드를 8개로 표현하여 교차로에서의 회전을 표현하는 확장 도로망 모델을 사용하였다.(Kirby, 1969; Wattleworth, 1963; Caldwell, 1961) 확장된 도로망 모델은 기존의 최적경로 알고리즘을 그대로 이용할 수 있으나 망의 정점 및 간선 수가 증가하고 그 결과 최적 경로를 찾기 위한 계산량이 증가되는 문제점을 갖는다.

도로망의 회전제한을 고려하기 위하여 비확장 도로망 모델을 이용하여 회전제한 문제를 알고리즘에서 해결하는 최적경로 탐색 방법도 함께 진행되어 왔다.(Thomas, 1991; 이승환, 1996; 김익기, 1998) 이 방법은 계산시간에 큰 영향을 주는 노드와 링크의 수가 감소한다는 장점에도 불구하고 알고리즘에서 회전제한을 고려하는데 상당한 계산시간을 갖는다는 문제점을 갖는다.

본 논문에서는 회전 제한이 있는 도로망을 효율적으로 표현할 수 있는 도로망 모델을 제안하였다. 즉, 각 회전 제한 방향에 대하여 U턴이나 P턴을 이용하는 대체 경로를 미리 계산하여 이를 도로망 데이터에 포함시킨 비확장 구조의 도로망 모델을 제안하였다. 제안된 도로망 모델을 이용하여 최적 경로를 계산하기 위해서는 링크 통과 비용 뿐만 아니라 대체 경로를 이용하는 회전 비용을 고려해야 하기 때문에 기존의 Dijkstra 알고리즘(Dijkstra, 1959) 등을 그대로 적용할 수 없다. 이러한 문제점을 극복하기 위하여 본 논문에서는 제안된 도로망 모델에 적합한 새로운 최적경로 알고리즘을 제안하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 기존의 회전제한을 고려한 경로계산 방법들을 비교하고 Ⅲ장에서는 새로 제안하는 도로망 모델과 이를 이용하여 최적 경로를 계산하는 경로 계산 알고리즘을 설명한다. Ⅳ장에서는 실제 도로망 데이터를 이용하여 제안된 경로계산 방법의 성능을 기존의 방법과 비교한 후 마지막으로 결론을 제시한다.

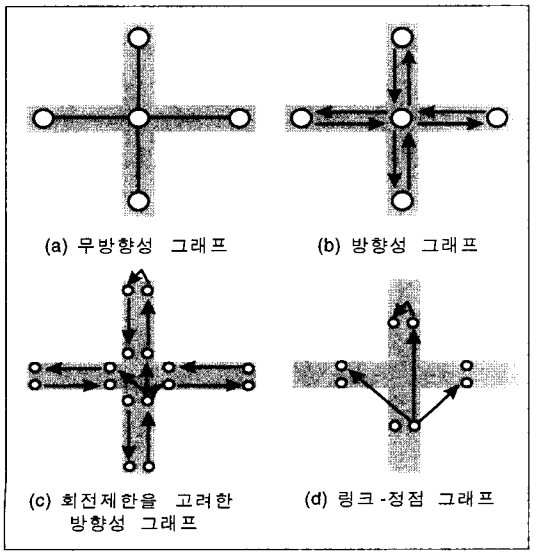
Ⅱ. 기존의 회전제한을 고려한 경로계산 알고리즘들의 비교

좌회전 금지, U턴, P턴 등 회전제한을 포함한 도로망에서의 최적경로 탐색은 크게 두 가지 방법에 의해서 해결될 수 있다. 첫 번째 방법은 도로망의 특성인 회전제한을 도로망 모델에 포함시켜 기존의 최적경로 알고리즘을 적용하는 것이고 두 번째 방법은 도로망을 기존의 도로망 형태대로 모델링 한 후

회전제한을 해결하는 알고리즘을 적용하는 것이다. 이번 절에서는 지금까지 연구되어 왔던 도로망 모델과 알고리즘의 종류들을 나열하고 그 장단점을 비교한다.

일반적으로 그래프(Graph)는 정점(vertex)의 집합 V(G)와 간선(edge)의 집합 E(G)로 표현되며 간선에 부과되는 가중치(weight) w를 포함시킴으로써 망(network)을 표현할 수 있다. 최적 경로 계산을 위한 망을 표시하기 위해서는 먼저 최적 경로의 정의에 따른 망의 특성을 모델링 해야 하며, 이에 따라 적절한 형태의 그래프를 선택하여 나타내고자 하는 망의 특성을 표현해야 한다. 도로망을 표시할 경우 다음과 같은 도로망의 특성들 중에서 필요한 사항을 고려해야 한다(Sheffi, 1985).

- 도로의 연결 상태 : 도로간의 교차로나 램프(ramp) 등에 의한 도로의 연결, 도로 종료점이나 진입 금지 지점 등에 의한 도로의 단절
- 도로의 방향성 : 양방향 도로, 일방통행 도로 등
- 도로의 통과비용 : 도로의 차선수, 교차로간 건널목수, 도로길이 등의 정적인 요소에 의한 통과비용, 차량진행속도, 차량통행량 등의 동적인 요소에 의한 통과비용
- 교차로 등 노드에서의 특성 : 좌회전 금지에 의한 진행 방향 제한, U턴에 의한 진행 방향 변경 허용, 교차로에서의 신호 대기 시간 등



〈그림 1〉 도로망 특성의 표현에 따른 각종 그래프의 예

〈그림 1〉은 이러한 도로망의 여러 가지 특성을 표시한 그래프이다. 그림에서 (a)~(c)는 교차로, U턴 지점, 도로 종료점 등의 노드(node)를 정점으로 표시하고 그 사이의 도로인 링크(link)를 간선으로 표시한 노드-정점(node-vertex) 그래프이다. 이들 중에서 (a)는 단순히 노드와 링크의 연결 상태만을 표시하기 위한 방식이며, (b)는 한 노드에서 인접한 다른 노드로의 진행 상태를 방향성을 가지고 표현한 것이다. (c)는 교차로 통과비용을 표현할 수 있도록 교차로 노드를 8개의 정점으로 확장한 것으로서 교차로에서 한 간선에서 다른 간선으로의 진행 방향을 각각 표시할 수 있다.(Wattleworth, 1963) (d)의 그래프는 링크를 정점으로 하고 노드를 간선으로 표현한 링크-정점(link-vertex) 그래프이며, 이러한 형태로 도로망을 표시할 경우 (c)에 비하여 데이터베이스의 크기를 줄일 수 있다.(Caldwell, 1961) 〈표 1〉은 〈그림 1〉의 (b), (c), (d) 방식으로 $k \times k$ 노드 수를 갖는 격자형 도로망을 표시할 때 필요한 정점 및 간선 수를 비교한 것이다. 표에서 모든 링크는 양방향이고 U턴은 없다고 가정하였으며, 회전 제한에 따른 간선 수의 감소는 고려하지 않았다.

〈표 1〉에서 보인 바와 같이 도로망을 〈그림 1〉의 (d)와 같은 링크-정점 그래프로 표시한 확장 모델은 (b)와 같은 비확장 모델과 비교하여 정점 및 간선의 수가 많다. 그러나 기존의 최적경로 알고리즘을 이용하여 회전제한을 갖는 망에서의 최적경로를 계산하기 위하여 〈그림 1〉의 (c)나 (d)의 그래프를 이용해야 한다.

일반적으로 경로 계산 알고리즘들은 정점에 부여되는 라벨의 관리 방법에 따라 라벨 고정 방식(label setting method)과 라벨 수정 방식(label correcting method)으로 구분할 수 있다. 라벨 고정 방식에서 모든 정점들은 일시적으로 라벨된 상태나 영구적으로 라벨된 상태를 갖는다. 경로계산 알고리즘에서 반복되는 루틴을

〈표 1〉 노드-정점 그래프와 링크-정점 그래프의 데이터 크기 비교

	노드-정점 그래프		링크-정점 그래프
	그림 1 (b)	그림 1 (c)	그림 1 (d)
정점 수	k^2	$8k(k-1)$	$4k(k-1)$
간선 수	$4k(k-1)$	$4(4k^2-7k+2)$	$4(3k^2-6k+2)$

행할 때마다 하나의 정점이 영구적으로 라벨되며 그것은 시작정점부터 그 정점까지 최적의 경로를 갖는다는 것을 의미한다. 본 논문에서는 반복되는 루틴을 행할 때 영구적으로 선택되는 정점을 스캔되는 정점이라 정의하고 선택된 정점이 인접정점들의 값들을 갱신하는 과정을 가지치기로 정의한다. 또한 일시적으로 라벨된 정점들 중 한번이라도 다른 정점에 의해 값이 갱신된 정점들의 집합을 라벨집합으로 정의한다. 한번 영구적으로 라벨된 정점은 경로계산이 끝날 때까지 값의 수정이 일어나지 않는다. Dijkstra 알고리즘이 라벨 고정 방식의 대표적인 예이다. 라벨 수정 방식은 경로계산이 끝날 때까지 어떠한 정점도 영구적으로 라벨 되지 않는다. 인접한 어떤 정점에 의해 더 좋은 비용으로 수정된 정점은 라벨 집합의 큐(queue)에 들어가 다시 경로계산 반복 루틴에 사용될 정점으로 선택될 수 있다. 한번 경로계산 반복 루틴에 사용된 정점이라도 다른 정점에 의해 라벨이 수정되어 또 다시 경로계산 반복 루틴에 이용될 수 있다. Moore 알고리즘, D'Esopo 알고리즘 등이 이에 속한다(Dial, 1979; Pape, 1974).

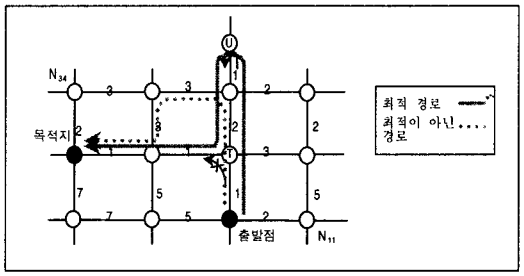
라벨 고정 방식인 Dijkstra 알고리즘은 정점 수를 n, 간선 수를 m이라 할 때 계산시간은 $O(m+n\log n)$에 따라 증가하는 것으로 알려져 있다.(Ahuja, 1990) 그러나 Moore 알고리즘과 같은 라벨 수정 방식은 정점의 라벨을 계속 수정시키면서 최적경로를 탐색하므로 가장 좋은 탐색 시간을 가질 경우와 가장 나쁜 탐색 시간을 가질 경우의 시간 차이가 매우 클 수 있어 일정한 계산시간을 갖지 않는다. 자동차 항법 시스템에서의 최적경로 탐색과 같은 기능은 어느 때나 일정한 시간 내에 사용자에게 최적경로를 제공해야 한다. 따라서 이러한 응용에는 Moore 알고리즘과 같은 라벨 수정방식 보다는 Dijkstra 알고리즘과 같은 라벨 고정 방식을 적용하는 것이 바람직하므로 본 논문에서는 Dijkstra 방식에 대하여 고려한다.

회전제한이 존재하는 도로망에서 Dijkstra 알고리즘 같은 기존의 알고리즘을 이용한 최적 경로 탐색은 알고리즘이 회전제한을 해결하는 U턴이나 P턴을 지원하지 않으므로 최적의 경로가 아닌 다른 경로가 얻어질 경우가 발생할 수 있다. 기존의 알고리즘을 이용하기 위해서는 도로망을 회전제한 특성이 고려된 〈그림 1〉의 (c)나 (d)와 같은 모델로 변환하여야 한다. 이러한 경우 앞에서 언급한 바와 같이 많은 수의 정점 및 간선을 필요로 하기 때문에 이를 저장하기 위한 데이터베이스의 크기가 증가한다. 또한 라벨집합의 원소 수가 증가하기 때문에 스캔될 정점을 탐색하는 시간 및 스캔되는 정점의 수가 증가하여 최적경로를 계산하는 소요 시간이 증가하게 된다.

회전제한을 표현하기 위한 도로망 모델에 관한 연구뿐 아니라 회전제한을 경로계산 중에 고려하는 방법에 관한 연구도 진행되어 왔다. 회전제한을 고려한 알고리즘은 보통 〈그림 1〉의 (b)와 같은 모델을 그대로 이용하고 경로를 계산하는 과정 중에서 회전제한 문제를 해결한다. 덩굴망 최적경로 탐색 알고리즘이 이를 시도한 알고리즘이지만 연속된 좌회전 금지가 있는 도로망에서 불합리한 경로를 추적할 가능성이 있다는 사실을 최기주(1995), 노정현(1995)이 지적하였다. 이를 해결하기 위하여 김익기(1998)는 수정형 덩굴망 최단경로 탐색 알고리즘을 소개하였다. 이 알고리즘은 덩굴망 최단경로 탐색 알고리즘의 계산법을 기초로 하였으나 각 정점마다 서로 다른 방향으로부터 도달될 때 다시 스캔될 수 있도록 함으로써 도로망 모델은 〈그림 1〉의 (b)의 형태를 이용하였으나 계산과정의 형태가 〈그림 1〉의 (d)와 매우 유사함을 알 수 있다.

회전제한을 위한 경로계산 방법을 제시한 이승환(1996)의 링크탐색 알고리즘은 〈그림 1〉의 (b)와 같은 모델을 그대로 이용하면서 U턴이 이루는 정점수를 감소시킴으로써 데이터베이스의 크기를 줄인 반면 회전제한 정점에서 U턴을 고려할 때 인접한 간선에 있는 U턴만을 고려하므로 〈그림 2〉에서와 같이 그 이상의 영역에 있는 U턴을 이용하여 최적경로를 얻어야 할 경우에는 잘못된 최적경로가 찾아질 수 있다.

이상에서 볼 때 회전제한을 포함한 도로망에서 최적의



〈그림 2〉 링크탐색 알고리즘을 적용하여 구한 최적이 아닌 경로의 예

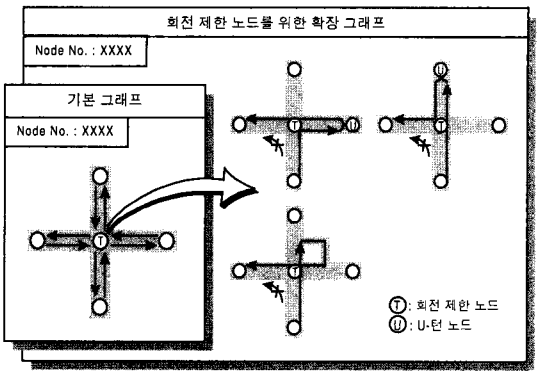
경로를 빠르게 찾을 수 있는 기존의 알고리즘들 중 〈그림 1〉의 (d)와 같은 모델에 Dijkstra 알고리즘을 적용한 것이 성능 면에서 가장 우수함을 알 수 있다. 본 논문에서 제안하는 알고리즘은 〈그림 1〉의 (b)의 모델을 기본으로 하고 라벨 고정 방식의 변형인 알고리즘으로 최적경로를 구하는 방법으로 지금까지의 방법보다 우수한 성능을 나타낸다. 즉 제안한 알고리즘은 회전제한을 고려하기 위하여 모든 노드들에 동일하게 다시 스캔하는 방법을 적용하지 않고 회전제한 고려를 위해 필요한 노드들만을 다시 스캔함으로써 필요없는 오버헤드를 줄이면서 회전제한이 고려된 최적의 경로를 얻을 수 있도록 하였다.

Ⅲ. 회전제한이 있는 도로망을 위한 제안된 도로망 모델과 알고리즘

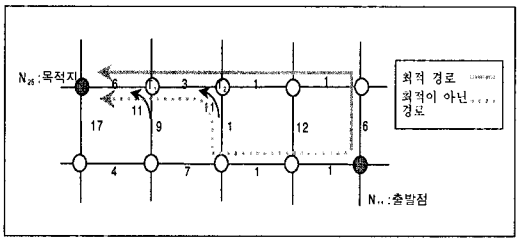
1. 회전제한이 있는 도로망을 위한 제안된 도로망 모델과 알고리즘

본 논문에서는 회전 제한이 있는 도로망을 효율적으로 표현하기 위하여 〈그림 1〉 (b)와 같은 비확장 구조를 갖는 도로망 모델을 제안한다. 제안된 모델의 기본 개념은 회전 제한이 있는 방향으로 진행할 수 있도록 해주는 대체 경로를 미리 계산하여 이를 데이터베이스로 갖고 있다면 모든 교차로는 회전 제한이 없는 것으로 생각할 수 있다는 것이다. 즉 회전 제한이 없는 교차로의 경우에는 〈그림 1〉 (b)와 같은 형태의 그래프를 이용하고 회전 제한이 있는 교차로에 대해서는 〈그림 3〉과 같은 형태의 확장 그래프의 대체경로를 이용하여 회전 제한 방향으로 진행이 가능하도록 한다. 확장 그래프의 대체경로는 〈그림 3〉의 세 가지 가능한 대체 경로 중에서 최소 비용의 경로 구조를 가지게 되며, 회전 제한 정점으로의 진입 정점, 경유 정점 및 진출 정점을 모두 표시하고 있어야 한다. 제안된 모델에서 회전 제한이 있는 방향으로 진행할 경우에는 간선 통과비용 외에 대체경로의 비용에 의한 회전 비용이 추가되며, 대체경로를 만들 때 U턴을 이미 고려하였으므로 제안된 모델을 이용하여 경로계산을 할 때 U턴 방향은 고려하지 않는다.

제안된 도로망 모델은 회전 제한이 있는 특정 방향



〈그림 3〉 회전 제한이 있는 정점에 대한 확장 그래프



〈그림 4〉 제안된 모델에 Dijkstra 알고리즘을 적용하여 구한 최적이 아닌 경로의 예

에 대해 정점에서의 회전비용이 추가되어 있기 때문에 기존의 Dijkstra 알고리즘을 그대로 적용하면 계산된 경로가 최적이 아닐 수 있다. 예를 들어 〈그림 4〉와 같은 도로망에 대하여 최적경로를 계산하는 경우를 생각해 보자.

〈그림 4〉의 도로망은 N_{23}의 정점에 회전 제한이 존재하여 N_{13}으로부터 N_{24}로 진행하는 방향에 회전 비용이 부가되기 때문에 출발점(N_{11})으로부터 목적지(N_{25})까지의 최적경로는 실선(N_{11}-N_{21}-N_{22}-N_{23}-N_{24}-N_{25})이 되며 비용은 17이 된다. 그러나 Dijkstra 알고리즘을 적용하여 계산된 최적경로는 점선(N_{11}-N_{12}-N_{13}-N_{23}-N_{24}-N_{25})이 되고 이때의 비용은 23이다. 이러한 결과가 발생하는 이유는 Dijkstra 알고리즘을 이용할 때 N_{22}보다 N_{23}이 먼저 스캔되어 사실상 최적경로를 이루는 N_{11}-N_{21}-N_{22} 경로로부터 N_{23}으로 진행되는 경로를 고려할 수 없기 때문이다.

제안한 도로망 모델상에서 최적 경로를 계산하기 위해서는 가지치기시 회전제한 방향의 진출 정점(〈그림 4〉의 경우 회전제한 N_{23}에 대한 N_{24}와 회전제한 N_{24}에 대한 N_{25})에 대하여 회전제한 방향으로의 진행 비용과 다른 경로를 통한 진행 비용을 비교할 수 있어

야 한다. 본 논문에서는 정점을 회전 제한이 있는 정점과 회전제한이 없는 정점으로 구분하여 회전제한이 있는 정점에 대해서는 진행 방향에 따른 비용을 비교할 수 있는 알고리즘을 제안한다. 즉, 회전 제한이 있는 정점이 회전 제한 방향으로 스캔된 경우 추후 회전 제한 방향이 아닌 다른 방향으로부터 도달할 수 있도록 함으로써 회전 제한 진출 정점에 대하여 진행 방향에 따른 비용을 비교할 수 있도록 한다. 이를 위하여 본 논문에서는 회전 제한이 있는 정점에 대하여 마디(knot)를 정의한다.

마디는 특정 정점들이 모여서 구성되며, 회전제한 정점이 회전제한 방향으로 스캔된 경우에 회전제한 방향이 아닌 다른 방향으로부터 회전제한 정점에 도달할 수 있는 경로를 만드는데 사용한다. 그러므로 회전제한 정점은 경로계산을 위해 가지치기를 하는 방향과 반대방향의 마디를 생성하여 다시 스캔될 수 있도록 관리하기 위해 마디의 근원을 갖는다. 마디의 근원(root)은 회전제한 정점이 회전 제한 방향으로 스캔되는 시점에 만들어지며, 마디의 원소가 되기 위해 초기화 작업을 한다. 초기화는 이미 스캔된 정점을 다시 스캔되도록 하기 위한 절차로서 마디의 원소로 소속될 정점의 비용 및 이전 노드(back node) 정보를 별도의 변수에 저장한 후 해당 정점의 비용과 이전 노드를 초기값으로 만든다. 해당 정점의 이전 노드를 제외한 인접 정점 중에서 이미 스캔되었거나 마디에 포함된 정점이 있다면 이들로부터 도달되는 비용 중 최소값과 해당하는 인접 정점 값을 이용하여 비용과 이전 노드의 초기값을 정한다. 이때 인접 정점이 마디에 포함된 정점인 경우에는 마디에 포함되기 이전 값을 이용한다. 또한 해당정점은 다시 스캔가능한 상태로 만든 후 초기화된 비용과 이전 노드를 가지고 마디에 포함되며, 이러한 과정을 마디 원소의 초기화로 정의한다. 마디 원소의 초기화는 하나의 정점에 대하여 한번 이상 발생할 수 있다. 이러한 경우에는 비용 및 이전 노드 초기값을 설정할 때 인접 정점들 중에서 이전에 초기화에 사용했던 정점들을 제외한 나머지 정점들로부터 도달되는 비용 중 최소값을 이용한다. 회전제한 정점이 마디 원소의 초기화를 수행한 후 근원으로 들어가는 과정을 마디의 생성으로 정의한다.

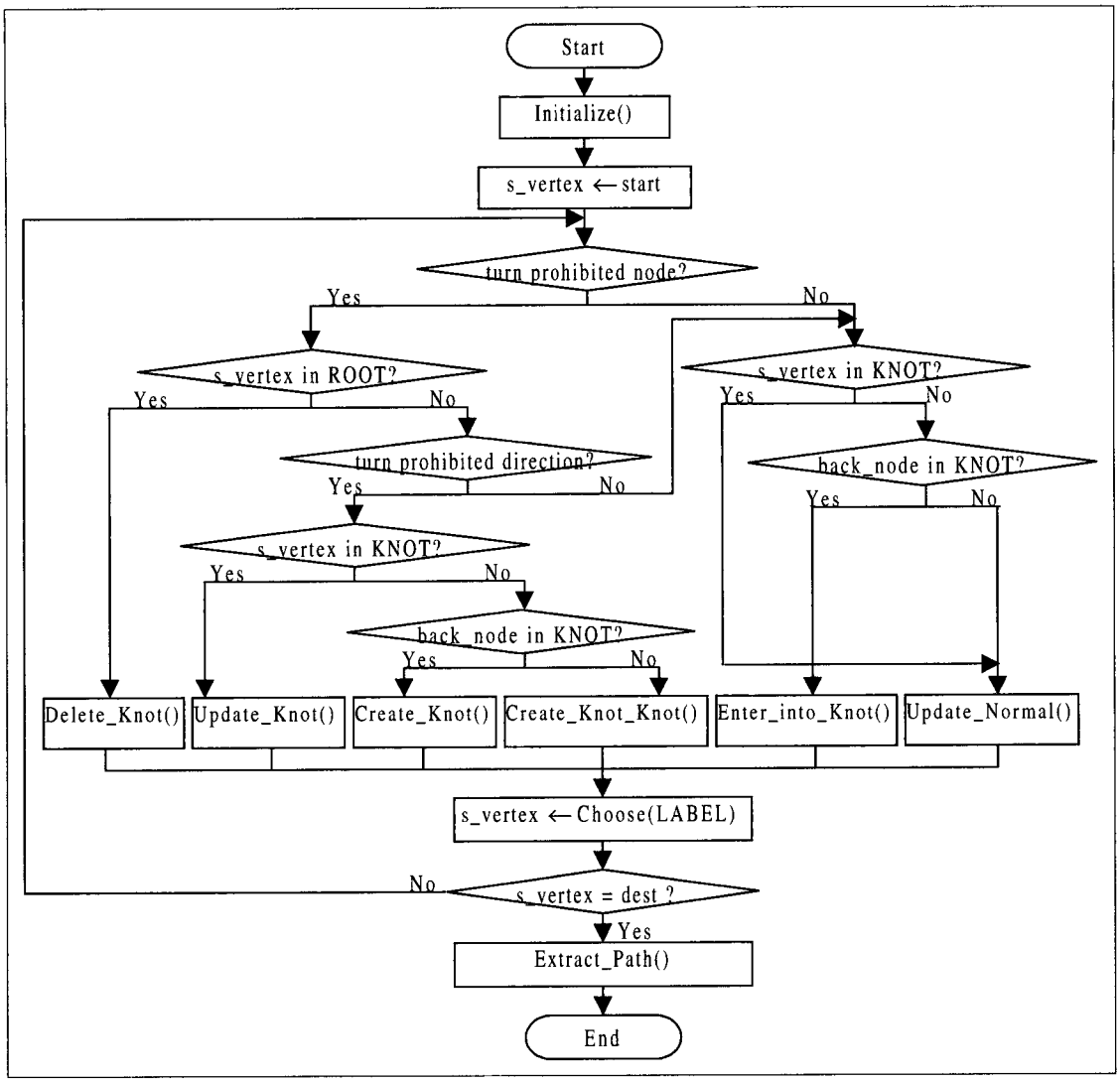
근원으로부터 가지치기된 정점들이 스캔될 때 추후 역 방향으로 다시 가지치기가 진행되어 근원에 도달할 수 있도록 마디에 포함하여야 한다. 즉, 마디 내에 있는 정점을 이전 노드로 가지는 정점이 스캔될 때 해당 점점은 마디 원소의 초기화를 거친 후 이전 노드의 자손으로서 마디의 원소로 소속된다. 현재 스캔되는 정점이 회전제한 정점인 근원과 n개의 정점을 통하여 연결되었다면 이 정점은 근원의 n+1대 자손(child)이 되며, 이러한 과정을 마디의 성장으로 정의한다.

근원인 정점이 다시 스캔되어 가지치기를 할 때에는 마디의 정보를 이용하여 인접 정점의 비용과 이전 노드 값을 결정한다. 이러한 경우 이전 노드는 마디 내의 정점의 열(array)로 구성된다. 정점의 열은 근원으로부터 마디 원소 정점의 이전 노드 값을 추적하여 구성하다가 마디에 존재하지 않는 이전 노드를 발견하면 이를 열의 최종 정점으로 한다. 근원인 정점에서 가지치기가 끝나면 마디를 삭제하며, 이때 마디의 원소인 정점들에 대한 비용과 이전 노드 값은 마디 원소의 초기화 이전 값으로 환원시킨다. 만약 정점에 두개 이상의 회전 제한이 존재하는 경우에는 근원의 1대 자손 중 근원을 다시 스캔하도록 만든 자손 및 해당 자손의 자손들을 삭제하고 그들의 비용과 이전 노드 값을 마디 원소의 초기화 이전 값으로 환원시킨다. 또한 근원 정점에 대해서는 마디 원소의 초기화를 수행하고 해당 근원에 대한 마디는 계속 존속시킨다. 마디의 삭제는 해당 근원의 모든 회전제한 방향들에 대하여 다른 경로들과 비교가 끝나면 수행한다. 이상과 같은 과정을 마디의 삭제로 정의한다.

마디를 이용하여 최적경로를 계산하는 제안된 알고리즘의 구조는 〈그림 5〉와 같다. 그림에서 start는 출발점, dest는 목적지를 의미하며, 기타의 변수에 대한 정의는 다음과 같다.

- KNOT : 앞에서 정의한 마디들의 집합. 마디에 속하는 정점은 한번 이상 스캔된 적이 있는 정점들로 구성됨.
- ROOT : 마디의 근원이 되는 정점들의 집합.
- LABEL : 가지치기에 의하여 라벨(label)된 정점들의 집합. 여기에 속하는 정점은 현재까지의 비용과 이전 노드 값을 가짐.
- s_vertex : LABEL 내에서 가장 작은 값을 가진 정점을 할당하는 변수. 즉, 현재 스캔되는 정점임.

제안된 알고리즘에서 각 정점에 대하여 관리하는 정보는 다음과 같다.



〈그림 5〉 제안된 경로계산 알고리즘의 흐름도

- scan : 정점의 스캔 여부를 나타내는 변수.
- back_node : 출발점으로부터 최소 비용으로 현재 정점에 도달한 경로에서의 이전 노드 번호. back_node는 하나의 정점 또는 정점의 열로 표현된다.
- cost : 출발노드로부터 현재까지의 도달한 경로에 대한 비용.
- old_back : 정점이 마디 원소가 되기 위한 초기화를 하기 전에 back_node의 값을 저장하는 변수.
- old_cost : 정점이 마디 원소가 되기 위한 초기화를 하기 전에 cost의 값을 저장하는 변수.
- turn_flag : 스캔되는 정점이 회전제한을 포함하는지를 나타내는 변수.
- turn_x_node : 회전제한 정점에 대하여 관리되는 정보로서 해당 정점의 인접 정점들에 대한 cost와 back_node 값을 저장하는 변수. 회전제한 정점이 연속적으로 존재하는 특수한 경우에 사용되며, 하나의 회전제한 정점 1에 대하여 이미 마디의 삭제까지 이루어진 상황에서 인접한 다른 회전제한 정점 2가 근원이 되기 위한 마디 원소의 초기화 과정을 수행할 때 이 정보를 이용한다. 즉 마디의 삭제까지 이루어진 정점 1은 이미 스캔된 정점이므로 정점 2의 마디 원소의 초기화 과정 중에서 스캔된 인접 정점으로부터의 비용을 비교하는 과정에 고려되어야 하며, 정점 1의 turn_x_node

정보 중 정점 2에 대한 값을 이용하여 다른 정점으로부터 오는 비용과 비교한다.

제안된 알고리즘은 9개의 절차로 구성되며, 특히 이들 중 마지막 6개는 스캔되는 정점의 상태에 따라 각각 처리하는 절차이다.

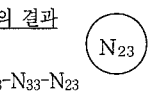
① Initialize() : 제안된 알고리즘에서 사용하는 변수들을 초기화하는 절차.
② Choose(LABEL) : LABEL 집합에서 가장 작은 비용을 갖는 정점을 찾는 절차.
③ Extract_Path() : 최적경로를 추출하는 절차. 목적지 정점이 스캔되는 정점으로 선택되면 이전 노드 정보를 이용하여 최적경로를 구성한다.
④ Update_Normal() : 일반적인 Dijkstra 알고리즘에서와 같이 스캔되는 정점에서 가지치기하여 인접 정점에 도달한 비용이 인접 정점이 이미 갖고 있는 비용보다 작을 때 새로운 비용과 이전 노드로 갱신한다.
⑤ Create_Knot() : 스캔되는 정점이 회전제한 정점이고 회전제한 방향으로 스캔되는 경우에 사용하는 절차. 마디의 생성을 수행하며, 스캔되는 정점의 인접 정점 중 회전제한 진출정점에 대해서는 대체경로를 이용하여 가지치기를 하고 나머지 정점에 대해서는 Update_Normal()에서와 같은 방법으로 가지치기를 한다. 해당 정점은 마디 원소의 초기화를 거쳐 마디의 근원으로 소속된다.
⑥ Enter_into_Knot() : 스캔되는 정점의 이전 노드가 마디에 포함되어 있는 경우에 사용하는 절차. 마디의 성장을 수행하며, 스캔되는 정점은 인접 정점에 대해 Update_Normal()에서와 같은 방법으로 가지치기를 한다. 해당 정점은 마디 원소의 초기화를 거친 후 마디 내의 이전 노드의 자손으로서 소속된다.
⑦ Create_Knot_Knot() : 스캔되는 정점이 회전제한 정점이고 회전제한 방향으로 스캔되며 이전 노드가 마디에 포함되어 있는 경우에 사용하는 절차. Create_Knot()과 동일한 과정을 통하여 해당 정점은 새로운 마디의 근원으로 소속된다. 또한 이전 노드의 자손으로서 이전 노드가 소속된 마디에 소속된다. 이러한 경우 어떤 마디 내의 원소가 다른 마디의 근원으로서 존재한다.
⑧ Update_Knot() : 스캔되는 정점이 회전제한 정점이고 회전제한 방향으로 스캔되며 해당 정점이 이미 마디에 포함되어 있는 경우에 사용하는 절차. 마디의 성장 중에서 특별한 경우이며, 스캔되는 정점의 자손들을 제외한 인접 정점 중 회전제한 진출정점은 대체경로를 가지고 가지치기하고 나머지 정점들은 Update_Normal()에서와 같은 방법으로 가지치기한다. 이미 마디에 소속되어 있으므로 새로 마디에 포함 시키지는 않으나 마디 원소의 초기화를 하여 추후 회전제한 진출 정점이 다른 경로로부터 도달된 비용과 비교할 수 있도록 한다.
⑨ Delete_Knot() : 스캔되는 정점이 ROOT에 포함되어 있을 경우에 사용하는 절차. 마디의 삭제 과정을 수행하며, 이를 통하여 구한 정점의 열을 이용하여 인접 정점으로 가지치기를 한다.

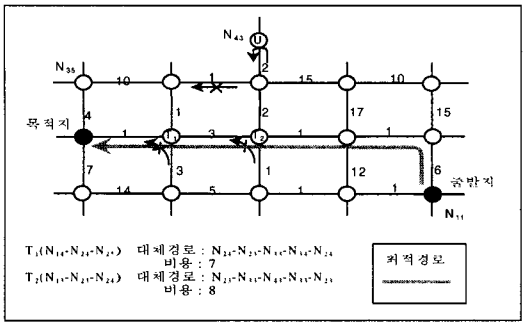
2. 제안된 도로망 모델과 알고리즘에 의한 최적 경로계산 예

〈그림 6〉의 도로망으로 예를 들어 제안된 알고리즘의 동작을 살펴보자. 그림에서 회전제한인 정점들에 대해 그 대체 경로와 비용을 표기하였다. 알고리즘의 계산이 끝난 후 목적지 정점부터 출발점 정점까지 추적해서 최적의 경로를 얻는 처리과정을 위해 대체경로도 그와 같은 순서로 저장하였다.

반복 4 : 회전제한 정점 N_{23}은 회전제한으로 스캔되었기 때문에 Create_Knot() 절차를 수행한다. 즉 가지치기를 한 후 근원만 존재하는 마디를 만들고 다시 스캔될 수 있도록 마디 원소의 초기화를 한다. N_{24}로 가지치기 할 때 회전제한 방향으로 스캔되므로 대체경로와 그 비용을 이용하여 N_{24}의 이전노드 및 비용을 갱신한다. 이전노드는 보통 단일 정점이 되지만 다음에서 보여지는 것과 같이 대체경로로부터 얻어지거나 혹은 마디로부터 경로가 얻어질 경우 정점들의 열로 구성될 수 있다. 다음은 절차수행의 결과와 N_{23}이 근원을 이루는 마디를 보여준다.

마디 원소(N_{23})의 초기화	가지치기 후 N_{24}의 결과
old_back : N_{13}	back_node:
old_cost : 3	N_{23}-N_{33}-N_{43}-N_{33}-N_{23}
back_node : 0	cost : 14
cost : $+\infty$	

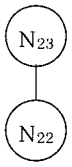




〈그림 6〉 예제를 위한 도로망

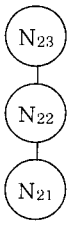
반복 5 : N_{22}의 이전노드가 마디에 포함되어 있는 N_{23}이기 때문에 Enter_into_Knot() 절차를 수행한다. 즉 가지치기를 하고 마디 원소의 초기화를 한 후 정점 N_{23}의 자손으로 마디에 포함된다. 마디 원소의 초기화 과정에서 인접한 정점 N_{12}가 이미 스캔된 노드이고 N_{22}는 N_{12}로부터 도달될 수 있으므로 N_{12}를 이전노드로 할당하고 N_{12}를 통해 N_{22}에 도달되는 값 13을 비용으로 할당한다. 다음은 마디 원소의 초기화 과정을 거친 N_{22}의 값과 수정된 마디를 보여준다.

마디 원소(N_{22})의 초기화
old_back : N_{23}
old_cost : 4
back_node : N_{12}
cost : 13



반복 6 : N_{21}의 이전노드가 마디에 포함되어 있는 N_{22}이기 때문에 Enter_into_Knot() 절차를 수행한다. 즉 가지치기를 하고 마디 원소의 초기화를 한 후 정점 N_{22}의 자손으로 마디에 포함된다. 마디 원소의 초기화 과정에서 인접한 정점 N_{11}이 이미 스캔된 노드이고 N_{21}은 N_{11}로부터 도달될 수 있으므로 N_{11}을 이전노드로 할당하고 N_{11}를 통해 N_{21}에 도달되는 값 6을 비용으로 할당한다. 다음은 마디 원소의 초기화 과정을 거친 N_{21}의 값과 수정된 마디를 보여준다.

마디 원소(N_{21})의초기화
old_back : N_{22}
old_cost : 5
back_node : N_{11}
cost : 6



반복 8 : 스캔될 노드로 선택된 N_{21}은 마디에 포함되지만 어떠한 회전제한도 없으므로 Update_ Normal() 절차를 수행한다. 즉 인접 정점들로 가지치기를 한다. 가지치기 이후에는 스캔된 상태로 되며 마디에서는 어떠한 변화도 갖지 않는다. 마디의 형태는 반복 6과 같다.

반복 11 : 반복 8과 같이 Update_Normal() 절차를 수행한다. 마디의 형태는 반복 6과 같다.

반복 12 : 선택된 N_{23}은 마디의 근원이므로 Delete_ Knot() 절차를 수행한다. 인접 정점들로 가지치기 할 때 N_{23}의 인접 정점인 N_{24}는 회전제한 정점이고 이미 N_{14}로부터 도달되어 10의 비용을 가지고 있다. N_{23}은 또 다른 방향으로부터 N_{24}에 도달되는 정점이므로 N_{23}으로부터 가지치기되어 갱신되는 이전노드와 비용은 앞에서 언급한 turn_ x_node에 저장된다. 즉 현재 상태의 정점 N_{23}과 N_{23}으로부터 도달되는 비용 11을 정점 N_{23}의 turn_x_node 정보에 N_{24}의 이전노드와 비용으로 저장한다. 또한 N_{23}을 근원으로 하는 마디의 근원으로부터 이전 노드를 추적하여 얻은 정점들의 열에 마지막 노드의 이전노드를 덧붙인 경로 N_{23}-N_{22}-N_{21}-N_{11}을 정점 N_{23}으로부터 도달될 때의 이전노드로 turn_ x_node에 저장한다. N_{24}에 도달되는 또 다른 경로를 얻기 위해 N_{23}을 근원으로 생성했던 마디는 N_{23}에 더 이상 다른 회전제한이 존재하지 않으므로 삭제되고 마디 내에 있던 N_{23}, N_{22}, N_{21}의 비용 및 이전노드는 각 정점에서 관리하는 old_ back과 old_cost을 이용하여 마디 원소의 초기화 이전의 값으로 환원한다. N_{23}을 근원으로 하는 마디의 삭제 후 마디는 더 이상 존재하지 않는다.

반복 13 : 회전제한 정점 N_{24}는 회전제한방향으로 스캔되었기 때문에 Create_Knot() 절차를 수행한다. 즉 가지치기를 한 후 근원만 존재하는 마디를 만들고 다시 스캔될 수 있도록 마디 원소의 초기화를 한다. N_{25}로 가지치기 할 때 회전제한방향으로 스캔되므로 대체경로를 N_{25}의 이전 노드로 갱신하고 그 대체경로를 통과하여 도착되는 비용 18을 N_{25}의 비용으로 갱신한다. 마디 원소의 초기화 과정에서 N_{24}의 현재 이전노드와 회전제한진출방향의 정점을 제외한 인접정점들 중 스캔된 정점 N_{23}으로부터 N_{24}에 도달될 수 있으므로 N_{23}의 turn_x_node

정보 중 N_{24}로 도달할 때의 정보를 이용하여 N_{24}을 통과하여 도달할 때의 비용과 이전노드를 N_{24}의 비용과 이전 노드 값으로 갱신한다. 갱신 후 N_{24}의 결과 값과 생성된 마디의 형태는 다음과 같다.

마디 원소(N_{24})의 초기화
old_back : N_{14}
old_cost : 10
back_node :
N_{23}-N_{22}-N_{21}-N_{11}
cost : 11

가지치기 후 N_{25}의 결과
back_node
N_{24}-N_{23}-N_{33}-N_{34}-N_{24}
cost : 18

반복 15 : 마디의 근원을 이루는 정점 N_{24}가 스캔될 노드이므로 Delete_Knot() 절차를 수행한다. 인접정점들에 가지치기할 때 N_{25}는 현재의 비용 18보다 N_{24}를 통과한 비용 12가 더 작으므로 비용을 12로 갱신한다. 또한 N_{24}가 근원을 이루는 마디의 근원부터 이전노드를 찾아 마지막 노드(여기서는 근원인 N_{24}만 존재하는 마디이므로 마지막 노드는 N_{24})의 이전노드(N_{24}의 이전노드N_{23}-N_{22}-N_{21}-N_{11})를 덧붙인 경로 N_{24}-N_{23}-N_{22}-N_{21}-N_{11}을 N_{25}의 이전 노드로 갱신한다. 정점 N_{24}에 더 이상의 다른 회전제한이 존재하지 않으므로 N_{24}를 근원으로 하는 마디는 삭제되며 마디 내에 존재하던 N_{24}의 비용과 이전노드는 old_cost와 old_back을 이용하여 마디 원소의 초기화 이전의 비용과 이전 노드 값으로 환원한다. N_{24}를 근원으로 하는 마디의 삭제 후 더 이상의 마디는 존재하지 않는다.

〈표 2〉 〈그림 6〉에 대한 제안된 알고리즘의 경로계산 과정

반복	선택 정점	가지치기 결과				
		정점	비용	이전노드	라벨 집합	스캔된 정점
1	N_{11}	N_{12}	1	N_{11}	N_{12},N_{21}	N_{11}
		N_{21}	6	N_{11}		
2	N_{12}	N_{13}	2	N_{12}	N_{13},N_{21},N_{22}	N_{11},N_{12}
		N_{22}	13	N_{12}		
3	N_{13}	N_{14}	7	N_{13}	N_{14},N_{21},N_{22},N_{23}	N_{11},N_{12},N_{13}
		N_{23}	3	N_{13}		
4	N_{23}	N_{24}	14	N_{23}-N_{33}-N_{43}-N_{33}-N_{23}	N_{14},N_{21},N_{22},N_{24},N_{33}	N_{11},N_{12},N_{13}
		N_{22}	4	N_{23}		
		N_{33}	5	N_{23}		
5	N_{22}	N_{21}	5	N_{22}	N_{14},N_{21},N_{22},N_{24},N_{32},N_{33}	N_{11},N_{12},N_{13}
		N_{32}	21	N_{22}		
6	N_{21}	N_{31}	20	N_{21}	N_{14},N_{21},N_{22},N_{24},N_{31},N_{32},N_{33}	N_{11},N_{12},N_{13}
7	N_{33}	N_{32}	20	N_{33}	N_{14},N_{21},N_{22},N_{24},N_{31},N_{32},N_{43}	N_{11},N_{12},N_{13},N_{33}
		N_{43}	7	N_{33}		
8	N_{21}	N_{22}	7	N_{21}	N_{14},N_{22},N_{24},N_{31},N_{32},N_{43}	N_{11},N_{12},N_{13},N_{21},N_{33}
9	N_{14}	N_{15}	21	N_{14}	N_{15},N_{22},N_{24},N_{31},N_{32},N_{43}	N_{11},N_{12},N_{13},N_{14},N_{21},N_{33}
		N_{24}	10	N_{14}		
10	N_{43}	-	-		N_{15},N_{22},N_{24},N_{31},N_{32}	N_{11},N_{12},N_{13},N_{14},N_{21},N_{33},N_{43}
11	N_{22}	N_{23}	8	N_{22}	N_{15},N_{23},N_{24},N_{31},N_{32}	N_{11},N_{12},N_{13},N_{14},N_{21},N_{22},N_{33},N_{43}
12	N_{23}	(N_{24})	(11)	(N_{23}-N_{22}-N_{21}-N_{11})	N_{15},N_{24},N_{31},N_{32}	N_{11},N_{12},N_{13},N_{14},N_{21},N_{22},N_{23},N_{33},N_{43}
13	N_{24}	N_{25}	18	N_{24}-N_{23}-N_{33}-N_{34}-N_{24}	N_{15},N_{24},N_{31},N_{32},N_{34}	N_{11},N_{12},N_{13},N_{14},N_{21},N_{22},N_{23},N_{33},N_{43}
		N_{34}	11	N_{24}		
14	N_{34}	N_{35}	21	N_{34}	N_{15},N_{24},N_{31},N_{32},N_{35}	N_{11},N_{12},N_{13},N_{14},N_{21},N_{22},N_{23},N_{33},N_{34},N_{43}
15	N_{24}	N_{25}	12	N_{24}-N_{23}-N_{22}-N_{21}-N_{11}	N_{15},N_{25},N_{31},N_{32},N_{35}	N_{11},N_{12},N_{13},N_{14},N_{21},N_{22},N_{23},N_{24},N_{33},N_{34},N_{43}
16	N_{25}					

(괄호로 표시된 부분은 turn_x_node내에 저장되는 정보임.)

반복 16 : 가장 작은 비용을 갖는 정점이 목적지 정점이므로 경로계산의 반복 루틴을 끝낸다.

최적경로를 얻기 위하여 목적지 정점부터 시작하여 출발지 정점에 도달할 때까지 이전노드를 추적한다. 〈그림 6〉의 예에서는 목적지 정점의 이전노드 자체가 출발지 정점인 N_{11}까지의 경로를 포함하므로 얻어지는 최적경로는 N_{25}-N_{24}-N_{23}-N_{22}-N_{21}-N_{11} 이다.

제안된 알고리즘은 라벨 고정 방식의 변형으로서 마디에 포함되는 정점의 경우에는 라벨 수정이 가능하도록 하는 방식이다. 라벨 수정을 통하여 회전제한 정점에 대해서는 회전비용을 포함하는 경로의 비용과 다른 경로의 비용을 비교할 수 있다. 그러므로 제안된 알고리즘은 제안된 도로망 모델을 이용하여 최적 경로를 구할 수 있다. 제안된 알고리즘은 기존의 Dijkstra 알고리즘보다 복잡하고 하나의 정점에 대한 처리시간이 다소 길다. 그러나 Dijkstra 알고리즘의 경우 회전제한을 고려하기 위해서는 확장모델을 사용해야 하기 때문에 많은 수의 정점 및 간선에 대하여 계산을 하여야 한다. 반면 제안된 알고리즘은 2장에서 제안한 비확장 구조의 도로망 모델을 사용하기 때문에 동일한 도로망에 대하여 확장 모델의 1/3~1/2가량의 정점과 간선을 이용하여 최적경로를 계산하며, 그 결과 계산시간을 대폭 줄일 수 있다.

Ⅳ. 실험결과

회전 제한이 있는 도로망에 대하여 〈그림 1〉의 (d)와 같은 링크-정점 모델에 Dijkstra 알고리즘을 적용하는 기존의 경로계산 방법(Method A)과 제안된 도로망 모델과 최적경로 알고리즘을 이용한 방법(Method B)의 성능을 비교하였다. 실험에 사용된 도로망은 서울지역에 대한 1/25,000 도로지도 4개 도엽을 합하여 구성한 데이터를 이용하였다. 이는 20 Km×30 Km에 해당하는 넓이이며, 도로망의 노드 분포는 〈그림 7〉과 같다. 기타 실험에 사용된 도로망의 노드, 링크에 대한 자세한 사항은 다음과 같다.

노드 수(교차로, U턴 지점 포함) : 5379
링크 수(양방향, 일방통행 포함) : 8074
U턴 지점 수 : 80
회전제한 수 : 2117

〈그림 7〉 서울지역 도로망의 노드 분포

서울지역 도로망에 대하여 링크-정점 모델과 제안된 모델로 경로 계산용 도로망을 구성하였을 때 소요된 데이터베이스의 크기는 〈표 2〉와 같다. 실험에 사용한 도로망에 대하여 제안된 모델은 확장모델의 약 1/2의 정점 및 간선으로 표현되며, 약 1/2의 데이터베이스 크기를 갖는다.

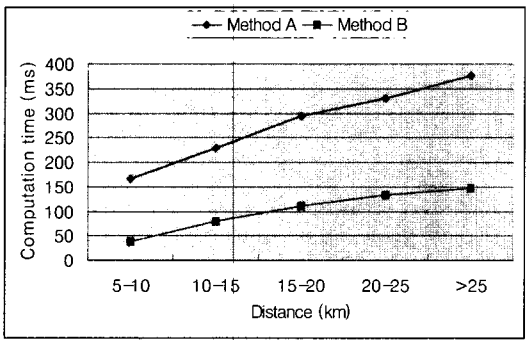
주어진 도로망에 대하여 두 가지 경로 계산 방법을 이용하여 PC상에서 경로계산을 수행하였다. 사용된 PC는 CPU가 Intel Pentium MMX 200MHz이고 주 메모리가 32MB이며, Windows 95 환경이다. 〈표 4〉는 두 가지 방법을 이용하여 구한 최적경로 계산시간, 스캔된 정점수, 그리고 비용을 출발점-목적지간 거리에 따라 비교한 것이다. 특히 제안된 방법에 대해서는 생성된 마디의 수 및 마디에 포함된 정점의 수를 표시하였다. 표에서 구한 값들은 각 거리 구간마다 해당 거리를 갖는 출발점-목적지를 200쌍 무작위로 선택하여 최적경로를 구한 결과의 평균값이다. 비교 항목 중에서 거리구간에 따른 계산시간의 변화 추이를 그래프로 표시하면 〈그림 8〉과 같다.

〈표 3〉 제안된 도로망 모델과 확장 도로망 모델의 데이터베이스 크기 비교

	정점 수	간선 수	데이터베이스 크기
제안된 모델	5379	13137	244,264bytes
링크-정점 모델	13137	25352	457,452bytes

〈표 4〉 경로 계산 알고리즘의 성능 비교

	Method B					Method A		
거리 구간	계산시간 (ms)	스캔된 정점 수	생성된 마디 수	마디 내 정점 수	비 용	계산시간 (ms)	스캔된 정점 수	비 용
5-10km	38.5	1262.8	82.3	248.1	7632.9	165.7	2151.1	7632.9
10-15km	80.1	2654.9	164.0	496.6	12427.1	229.3	4829.3	12427.1
15-20km	111.4	3887.7	228.2	699.4	17243.9	294.4	7438.5	17243.9
20-25km	132.7	4687.8	269.5	833.9	22304.5	329.8	9294.3	22304.5
25km이상	146.7	5129.6	289.5	923.5	24608.4	374.8	10023.1	24608.4



〈그림 8〉 거리구간에 따른 계산시간 비교

실험에서 선택된 1,000가지의 서로 다른 출발점-목적지에 대하여 제안된 방법은 매 경우마다 기존의 방법과 동일한 최적경로를 계산하였다. 그러나 계산 시간에 있어서는 제안된 방법이 기존의 방법보다 2배 이상의 우수한 성능을 나타내었다. 이는 스캔된 정점 수에서 알 수 있듯이 확장 모델을 이용하는 기존 방법의 경우에는 2배 이상의 정점을 관리하고 처리해야 하기 때문이다.

V. 결론

본 논문에서는 회전 제한이 있는 도로망에서의 고속 경로 계산을 위한 새로운 도로망 모델 및 경로계산 알고리즘을 제안하였다. 제안된 도로망 모델은 회전제한이 있는 도로망을 표현함에 있어서 기존의 확장모델과 비교하여 1/2 이하의 데이터베이스 용량으로 동일한 도로망을 표현하였다. 제안된 도로망을 이용하는 새로운 경로계산 알고리즘은 확장모델을 이용해야 하는 기존의 알고리즘과 비교하여 계산속도가 2배 이상 향상된 성능을 보였다. 제안된 경로계산 방법은 우리나라와 같이 회전 제한이 다수 존재하는 도로망 상에서 최적 경로를 고속으로 계산할 수 있는 효율적인 방법이다. 특히 차량 항법 시스템이나 휴대용 항법 단말 등과 같이 도로망 데이터베이스의 소형화와 고속 경로계산을 필요로 하는 시스템에 제안된 방법이 유용하게 적용될 수 있으리라 예상한다.

참고문헌

1. 김익기, ATIS를 위한 수정형 덩굴망 최단경로 탐색 알고리즘의 개발, 대한교통학회지, 제16권, 제2호, 1998, pp.157~167.
2. 노정현, 남궁선, 도시가로망에 적합한 최단경로탐색 기법의 개발, 대한국토 도시계획학회지, 제30권, 제5호, 1995, pp.153~168.
3. 이승환, 최기주, 김원길, 도시부 ATIS 효율적 적용을 위한 탐색영역기법 및 양방향 링크탐색 알고리즘의 구현, 대한교통학회지, 제14권, 제3호, 1996, pp.45~59.
4. 최기주, U-TURN을 포함한 가로망의 표현 및 최단경로의 구현, 대한교통학회지, 제13권, 제3호, 1995, pp.35~51.
5. Ahuja, R. K., K. Mehlhorn, R. B. Orlin and R. E. Tarjan, Faster algorithms for the shortest path problem, J. ACM, Vol. 37, 1990, pp.213~223.
6. Caldwell, T., On finding minimum routes in a network with turn penalties, Comm. ACM, Vol. 4, 1961, pp.107~108.
7. Cherkassky, B. V., A. V. Goldberg and T. Radzik, Shortest paths algorithms : theory and experimental evaluation, Tech. Rep. STAN-CS-93-1480, Stanford Univ., 1993.
8. Deo, N. and C. Pang, Shortest-path algori-

thms : taxonomy and annotation, Networks, Vol. 14, 1984, pp.275~323.
9. Dial, R.B., F. Glover, D. Karney and D. Klingman, A computational analysis of alternative algorithms and labeling techniques for finding shortest path trees, Networks, Vol. 9, 1979, pp.215~248.
10. Dijkstra, E. W., A note on two problems in connection with graphs, Numer. Math., Vol.1, 1959, pp.269~271.
11. Gallo, G and S. Pallottino, Shortest path algorithms, Annals of Oper. Res. Vol. 13, 1988, pp.3~79.
12. Kirby, R. R. and R. B. Potts The minimum route problem for networks with turn penalties and prohibitions, Transportation Res., Vol. 3, 1969, pp.397~408.
13. Pape, U., Implementation and efficiency of Moore-algorithms for the shortest route problem, Math. Programming, Vol. 7, 1974, pp.212~222.
14. Sheffi, Y., Urban transportation networks, Prentice-Hall, 1985.
15. Thomas. R., Traffic Assignment Techniques, Avebury Technical, 1991.
16. Walker, J. Mobile information systems, Artech House, 1990.
17. Wattleworth, J. A. and P. W. Shuldiner, Analytical methods in transportation; left-turn penalties in traffic assignment models, J. Engng Mech. Div. Am. Soc. civ. Engrs, Vol. 89, 1963, pp.97~126.
18. Zhao, Y., Vehicle location and navigation systems, Artech House, 1997.