■ 論 文 ■

링크표지갱신 다수경로탐색 알고리즘

K-th Path Search Algorithms with the Link Label Correcting

이 미 영
(위스콘신대학교 토목환경공학부 박사)

백 남 철
(한국건설기술연구원 첨단도로시스템연구센터 선임연구원)

최 대 순
(한국건설기술연구원 첨단도로시스템연구센터 수석연구원)

신 성 일
(서울시정개발연구원 도시교통연구부 부연구위원)

목 차

Key Words : 덩굴망, 링크표지확정, 다수경로탐색, 대안경로, 합리적 경로선택

요 약

최적경로 알고리즘에서 링크표지를 활용하면 도시가로망 상에 나타나는 유턴이나 피턴과 같은 주행했던 교차로를 다시 주행하는 통행에 대한 설명이 가능하다. 본 연구에서는 링크표지기법을 표지갱신기반 다수경로탐색 알고리즘으로 확대하는 것이 목적이다. 이를 위해 도시가로망에서 발생하는 운전자의 합리적 통행행태를 링크기반으로 개념화하고, 이들 행태가 반영되도록 링크표지갱신 다경로알고리즘을 제안하였다. 소규모 네트워크 테스트를 통하여 알고리즘의 수행과정과 결과의 적정성을 확인하였다. 대규모 네트워크 컴퓨터 수행을 통해 해의 퇴화현상을 파악하고 소수의 대안경로 정보제공에 활용 가능한 (K-1)차원 알고리즘을 제안하였다. 결론적으로 제안된 다수경로 알고리즘으로 링크표지기법이 제공하는 합리적 통행행태의 고려가 다수의 경로에도 가능해졌을 뿐만 아니라, 효율적인 대안경로 제공의 활용단계를 위해 한단계 전진되었다.

Ⅰ. 서론

교통망에서 기점과 종점을 연결하는 경로가 일련의 노드순서로 표현되는 경우 경로를 구성하는 방법에 따라 Loop와 Tree로 분류된다. Tree는 경로구성에 있어 같은 노드의 반복이 허용되지 않으나 Loop에서는 노드반복이 허용된다. 도시교통망을 주행하는 차량이 교차로 회전금지와 같은 규제 때문에 Loop형 통행처럼 교차로를 반복해서 주행하는 현상이 나타나므로, 경로정보의 제공을 위해서는 loop형 통행의 고려가 필요하다.

그러나 경로를 일련의 링크순서로 표현하는 경우에는 Loop형 통행을 경로정보제공에 활용하는 데는 문제가 존재한다. Loop는 경로를 구성함에 있어 링크반복에 대한 제약이 없다는 가정을 포함하고 있어, 교통망에서 최적통행에 대한 운전자의 특성을 고려할 때, 링크의 반복주행을 보장하는 것은 무리가 있다. 따라서, 교통망에서 나타나는 경로를 합리적으로 표현하기 위해서는 Loop의 개념이 링크의 반복여부로 표현된 경로의 통행현상으로 축소되어야 한다. 본 연구에서 합리적 통행은 "통행경로를 노드와 링크의 순서로 표현할 때 노드의 반복은 가능하나, 링크의 반복은 가능하지 않다"는 것으로 Loop형 통행의 가정이 링크를 기반으로 축소된 것이다.

링크를 근거로 해서 합리적 통행패턴을 고려한 경로를 탐색하는 기존의 알고리즘은 최단경로 탐색알고리즘을 기반으로 개발되어 왔다. 기존의 링크기반 최적경로 알고리즘은 우선 교차로의 지체 또는 회전금지를 네트워크의 확장 없이 고려하기 위한 방안으로 개발되어 왔다(Potts & Oliver, 1972; 노정현 & 남궁성, 1995; 김현명 & 임용택, 1999; 장인성, 2000). 링크기반 최적경로 알고리즘을 이용하여 다수의 경로를 순차적으로 탐색하는 방안이 제시되기는 했으나(이미영 외, 2003), 기존에 개발된 최단경로 알고리즘을 활용한 것이지 다수의 경로를 동시에 탐색하는 알고리즘의 근본적인 연구에 대한 시도는 아니었다.

본 연구는 기존의 노드표지에 근거해서 개발된 다수경로 탐색알고리즘을 링크표지로 전환함으로써 링크표지가 갖는 장점인 교차로의 특성과 합리적 통행을 고려한 다수의 경로를 동시에 탐색하는 알고리즘으로 발전시키는데 목적이 있다. 이를 위한 본 연구의 세부목적은 두 가지로서 첫째는 출발지에서 모든 노드까지 다수의 경로를 동시에 탐색하는 방안의 하나인 노드표지갱신(Node Label Correcting) 다경로 알고리즘에 링크표지를 적용된 알고리즘을 개발하는 것이고, 두 번째는 제안된 알고리즘을 실제 대규모 도시가로망상에 적용함으로써 발생되는 해 퇴화현상(Solution Degeneration)을 완화시키거나 극복하는 알고리즘을 제안하는 것이다.

본 연구는 다음과 같이 진행된다. 2장에서는 기존의 최적경로와 다경로알고리즘을 대상으로 이론적인 배경을 고찰하고, 3장에서는 링크표지에 근거한 합리적 통행원리를 개념화하고 이를 링크표지갱신 알고리즘에 확대 적용하는 방안을 제안하며, 4장에서는 제안된 알고리즘을 실제 예제에 적용하고, 우리나라 네트워크에 적용결과를 바탕으로 알고리즘의 개선방안을 제시하며, 5장에서는 결론과 향후연구과제에 대한 검토를 수행한다.

Ⅱ. 이론적 배경

최단경로알고리즘은 노드와 링크표지기법 모두 적용되고 있으나, 다경로알고리즘에서는 노드표지기법만 이용되고 있다. 본 절에서 표지갱신기법에 근거한 최단경로와 다경로알고리즘에 대한 수행과정을 살표본다.

1. 표식

본 연구에서 사용되는 표식은 다음과 같다.

- K : 복수경로의 수
- p,q,k : 경로의 순서($p,q,k=1,2,\cdots K$)
- r,s,i,j : 노드표지; 특히 r,s는 각각 출발지와 도착지노드
- i_k : 노드 i가 벡터로 표현된 경우 노드 i의 k번째 요소($i=\{i_1,i_2,\cdots,i_k\cdots,i_K\}$)
- a,b : 링크표지
- a_k : 링크 a가 벡터로 표현된 경우 링크 a의 k번째 요소($a=\{a_1,a_2\cdots,a_k,\cdots,a_K\}$)
- π_k^{ri} : 출발지 r에서 노드 i까지 k번째 통행비용
- π^{ri} : 출발지 r에서 노드 i까지 k개의 비용으로 구성된 벡터, $\pi^{ri}=\{\pi_1^{ri},\pi_2^{ri},,,\pi_k^{ri}\}$
- π_k^{ra} : 출발지 r에서 링크 a의 도착지점까지 k번째 통행비용

π^{ra} : 출발지 r에서 링크 a의 도착지점까지의 k개의 비용으로 구성된 벡터
$\pi^{ra}=\{\pi_1^{ra}, \pi_2^{ra}, , , \pi_k^{ra}\}$

$c_{a(i,j)}$: 링크 $a(i,j)$의 통행비용

d_{ab} : 링크 a의 출발노드에서 링크 b의 도착노드로 진행시 발생하는 회전지체

L : 링크집합
N : 노드집합
R : 출발지노드집합
S : 도착지노드집합
A_r : 출발노드가 r인 링크집합
B_s : 도착노드가 s인 링크집합
Q : 탐색링크집합
M : 탐색노드집합

2. 표지갱신 최단경로알고리즘

노드표지갱신(Node Label Correcting) 최적경로알고리즘은 출발지부터 탐색을 시작하여 먼저 탐색된 노드를 우선적인 다음탐색노드로 고려하여(FIFO:First In First Out) 링크의 도착노드까지의 비용을 계산한다. 알고리즘은 노드비용의 갱신이 더 이상 이루어지지 않을 만큼 반복하여 수행된다. 식(1)은 출발지에서 각 노드까지의 최단경로비용이 갱신되어 계산되는 수식을 나타낸다. 이 방법은 Moore(1957)에 의해 제안되었다.

$$\pi_1^{ri} = \min_{i \neq j}\{\pi_1^{ri} + c_{ij}, \pi_1^{rj}\} \quad (1)$$

경로추적과정은 Bellman의 최적원리 (1957)에 근거한 전노드 개념을 적용하여 최적해가 보장된다. 식(1)은 동적프로그래밍(Dynamic Programming)을 적용하여 최적경로를 탐색하는 과정을 보여준다. 크게 Step 1, Step 2, Step 3의 3단계로 구성되어 있다. Step 1에서는 전체 노드표지가 초기화되고 출발지에서 연결된 링크의 도착노드를 우선적으로 탐색한다. Stet 2에서는 다음탐색노드를 결정하며, 이때 탐색노드가 없을 경우 알고리즘은 종료된다. Step 3에서는 탐색노드와 연결된 링크를 대상으로 노드표지 갱신과정이 이루어진다.

Step 1 : 초기화
$\pi_r^{ri}=\infty,\ \forall i \in N,\ r \in R;$
$\pi_1^{ri}=c_{rj},\ \forall (r,j) \in A_r,\ r \in R;$
$M=M \bigcup \{j\};$

Step 2 : 다음탐색노드결정
If : $M=\phi$: 종료.
Else : $i=i_1,\ M=\{i_1, i_2, i_3, , , i_m\};$
$M=M-\{i_1\};$

Step 3 : 다음노드($i \rightarrow j$)로 확장
If : $\pi_1^{ri}+c_{ij} < \pi_1^{rj} : \pi_1^{rj}=\pi_1^{ri}+c_{ij},\ \forall j \in A_i;$
$M=M \bigcup \{j\};$
Go to Step 2;
Else : Go To Step 2;

링크표지갱신(Link Label Correcting) 최적경로알고리즘은 출발노드에서 연결된 링크를 우선 탐색을 시작한다. 이후 우선적으로 탐색된 링크를 다음탐색링크로 선정하고(FIFO:First In First Out), 선정된 링크의 도착노드를 출발노드로 하는 인접링크까지의 링크비용을 갱신한다. 알고리즘은 링크비용의 갱신이 더 이상 이루어지지 않을 만큼 반복하여 수행한다. 식(2)는 출발지 r에서 각 링크까지의 최단경로비용이 갱신되어 계산되는 수식을 나타낸다. 출발지 r에서 링크 b의 출발지점까지의 최단경로비용을 계산하기 위해서는 두 개의 인접된 링크 a와 b로 구성된 표지를 이용한다. 이때 노드표지와의 차이점은 두 인접링크에서 발생하는 회전비용(d_{ab})을 추가적인 네트워크의 확장 없이 고려할 수 있다는 것이다. 최적경로추적과정에서 기존의 최적경로알고리즘에서 채택하고 있는 전노드 개념(Moore, 1957)을 링크로 유추된 개념을 적용하므로 Bellman의 최적원리(1957)에 의한 최적해가 보장된다. 이 방법은 Curby and Potts (1969)에 의해 처음 제안되었으나, 링크비용만을 이용해 최단경로에 이용하고자 하는 시도는 Potts and Oliver(1972)에 의해 처음 개념적으로 제시되었다. 노정현, 남궁성(1995)은 Potts and Oliver와 동일한 최단경로 알고리즘을 회전금지 및 U-turn이 존재하는 가로망에 적용하였다. 이승환 외(1996)는 양방향 링크표지기법을 적용하여 탐색영역을 축소함으로써 알고리즘의 수행속도를 향상시켰다. 김현명 & 임용택

(1999)은 링크기반 최단경로기법을 이용해 복수 수단의 환승을 고려한 최단경로탐색방법을 제시하였다. 장인성(2000)은 서비스시간의 제약이 존재하는 복합교통망에 링크기반 최단경로 알고리즘을 제안하였다. 링크기반 최적경로의 구성식은 식(2)와 같으며 알고리즘에 대한 보다 자세한 사항은 임강원 & 임용택(2003)에 수록되어 있다.

$$\pi_1^{rb} = \min_{a \neq b}\{\pi_1^{ra} + c_b + d_{ab},\ \pi_1^{rb}\} \qquad (2)$$

식(2)의 링크표지에 근거한 표지확정 최단경로알고리즘은 다음과 같다. 노드표지기법에 비해 차이점은 회전지체의 고려와 함께, 링크표지에 근거해서 최단경로비용을 계산한 것이므로 알고리즘의 수행과정이 종료되면 노드표지에 근거한 비용으로 환산 해주어야 한다는 것이다(Step 2의 $Q=\phi$).

Step 1 : 초기화
$\pi_1^{ra} = \infty,\ \forall a \in L,\ r \in R$;
$\pi_1^{ra} = c_a,\ \forall a \in A_r,\ r \in R$;
$Q = Q \cup \{a\}$;

Step 2 : 다음탐색링크결정
If : $Q=\phi$: $\pi_1^{ri} = \min \lfloor \pi_1^{ra} \rfloor,\ a \in B_i$;
종료.
Else : $a = a_1,\ Q = \{a_1, a_2, a_3, , , a_m\}$;
$Q = Q - \{a_1\}$;

Step 3 : 다음링크($a \rightarrow b$)로 확장
If : $\pi_1^{ra} + d_{ab} + c_b < \pi_1^{rb}$: π_1^{rb}
$= \pi_1^{ra} + d_{ab} + c_b,\ \forall b \in D_a$;
$Q = Q \cup \{b\}$;
Go to Step 2;
Else : Go To Step 2;

3. 표지갱신기반 다경로알고리즘

다경로알고리즘은 경로비용의 순차적인 순서를 갖는 복수의 경로를 탐색하는 것이다. 기존에 제안된 알고리즘은 크게 2가지 방법으로 분류된다. 하나는 기존의 표지확정/갱신 방법을 확대 이용하는 것이고, 다른 하나는 경로의 부분삭제를 통한 다른 경로를 발견하는 방법이다. 두 방법의 가장 큰 차이로서, 전자는 K개의 경로를 동시에 탐색하는 방법(Shier, 1979; Pollack, 1961; Bellman & Kalaba, 1968)을 따르며, 후자는 최단경로알고리즘을 활용하여 탐색된 경로(1부터 K-1까지)를 기반으로 K번째 경로를 탐색하는 방법(Martins, 1984; Azevedo, 1963; Yen, 1971)을 따른다는 것이다. 부분삭제방법은 최단경로알고리즘을 그대로 활용하므로 해를 비교적 쉽게 산정할 수 있다는 장점이 있으나, 네트워크가 커지면 부분삭제의 링크개수가 많은 조합으로 나타나므로 컴퓨터 수행시간에 불리하다는 단점이 있다. 반면 표지확정/교정방법은 알고리즘의 해법이 복잡하나 적절한 k수에 대해 대규모 네트워크에 적용이 가능할 정도로 알고리즘 수행시간이 짧다는데 장점이 있다.

노드표지갱신 다경로알고리즘은 다수의 경로를 탐색한다는 것을 제외하고는 알고리즘 특성이 최적경로알고리즘과 거의 유사하다. Shier(1979)에 의해 제안된 노드표지갱신 다경로알고리즘은 다음과 같다.

Step 1 : 초기화
$\pi^{ri} = \{\infty, \infty, , , \infty\},\ \forall i \in N,\ r \in R$;
$\pi^{rj} = \{c_{rj}, \infty, , , \infty\},\ \forall (r,j) \in A,\ r \in R$;
$M = M \cup \{j\}$;

Step 2 : 다음탐색노드결정
if $M=\phi$: 종료.
Else : $i = i_1,\ M = \{i_1, i_2, i_3, , , , i_m\}$;
$M = M - \{i_1\}$;

Step 3 : 다음노드($i \rightarrow j$)로 확장
If : $\pi_p^{ri} + c_{ij} < \pi_q^{rj}$: $\pi_q^{ri} = \pi_p^{ri} + c_{ij}$,
$\forall j \in A_i,\ p, q = 1, 2, , , k$;
$M = M \cup \{j\}$;
Go to Step 2;
Else : Go To Step 2;

Ⅲ. 링크표지확정 다경로알고리즘

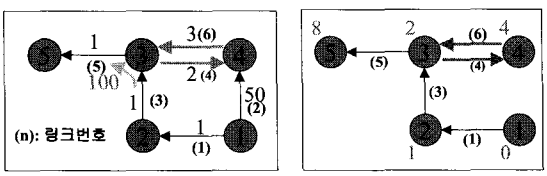
링크표지기반 최적경로가 갖는 장점인 합리적 통행행태를 고려할 수 있고, 다경로 알고리즘은 다수의 대안경로의 제공이 가능하다는데 있다. 이러한 링크

표지를 다경로 알고리즘으로 확대시킨 연구는 이미영 외(2003)에 의해 최초로 시도되었으나, 이는 출발지와 목적지가 정해진 One-to-One의 경우에 기존의 링크기반 최단경로 알고리즘을 활용한 것이지, 단일 출발지에서 모든 노드까지의 다수의 경로를 동시에 탐색하는 알고리즘(One-to-All)에 대해서는 아직 시도된 바가 없다. 본 연구에서 제안하는 알고리즘은 노드기반의 다경로 알고리즘을 링크표지로 전환 함으로서 위의 두 가지 장점을 포함한 합리적 통행행태가 반영된 다수의 경로를 동시에 탐색이 가능하다. 이를 위해 우선적으로 합리적 통행행태를 링크기반표지로 구성하는 원칙에 대한 설명과 이에 근거한 알고리즘을 제안하도록 한다.

1. 합리적 통행원리

합리적 경로란 "경로를 노드로 구성 할 경우 노드의 반복은 허용되나, 링크의 반복은 허용되지 않는다"는 것이다. 이를 그림1의 간단한 도식으로 설명한다. 〈그림 1〉의 (a)는 노드3을 기준으로 2->3->5방향에 100점의 회전점수가 존재하고, 노드3->4->3에서 U-Turn이 허용되기 때문에 노드1을 출발점으로 전노드까지의 최단경로를 탐색할 경우 (b)와 같이 구성될 수 있다. 이때 노드1에서 5까지의 최단경로를 예를 들어, 이 경로를 일련의 노드의 순서로 구성할 경우, 1->2->3->4->3->5와 같이 노드3이 경로를 구성하기 위하여 두 번 나타난다. 이 경로를 다시 링크로 구성하면, (1)->(3)->(4)->(6)->(5)의 순서로 나타나며 링크의 경우 링크의 번호가 경로를 구성하기 위하여 한 번만 나타나게 된다.

〈그림 1〉은 도시가로망에서 표현되는 간단한 통행패턴의 예를 든 경우로서, 만약 인접교차로에 연속하여 회전규제가 존재하는 경우나 교차로의 신호등에 의한 회전지체가 존재하는 특성 때문에 더욱 다양한



(a) 네트워크 (b) 노드1에서 최단경로탐색

〈그림 1〉 네트워크와 출발지에서 모든 노드까지 최단경로탐색

합리적 경로의 표현이 필요하다. 이 모든 경우가 경로를 구성하기 위하여 교차로를 여러 번 통과하는 것은 허용되나, 주행한 링크를 다시 주행하는 경우는 없다는 가정을 만족하고 있으며, 이 원리는 이미 링크를 구성한 표지를 이용하여 자연스럽게 표현이 가능하다.

2. 링크표지갱신 다경로알고리즘

본 연구에서 제안하는 링크표지갱신 다경로알고리즘의 수행과정은 아래에 나타나 있다. 링크기반 알고리즘이 위에서 위의 2절에서 설명한 기존 알고리즘과 다른 점은 링크표지에 근거해서 최단경로비용을 계산한 것이므로 알고리즘의 수행과정이 종료되면 노드표지에 근거한 비용으로 재환산 해주어야 한다는 것이다. 이 과정이 Step 2에 더 이상 탐색링크가 없을 때인 ($Q=\phi$) 알고리즘의 종료와 함께 나타나고 있다. 또한 다음 링크의 선정조건으로서, 선 투입 선 선출(First-In-First-Out)의 일반적으로 표지갱신 최적경로 알고리즘이 채택하고 있는 원칙을 유지하고 있음을 Step 2의 Else문에서 나타내고 있다. 그러나 표지갱신 최적경로 알고리즘과 다른 점은 Q 리스트에 포함되어 있는 링크에 갱신이 이루어진 경우에는 이 링크는 기존의 위치에서 Q의 가장 후미로 이동된다.

Step 1 : 초기화

$\pi^{ra}=\{\infty,\infty,,,\infty\}$, $\forall a\in L$, $r\in R$;

$\pi^{ra}=\{c_a,\infty,,,\infty\}$, $\forall a(r,j)\in A_r$, $r\in R$;

$Q=Q\bigcup\{a\}$, $\forall a(r,j)\in A_r$;

Step 2 : 다음탐색링크결정

if : $Q=\phi$: $\pi_q^{ri}=\min[\{\pi_i^{ra}\}]$,

$a\in B_i$, $\{\pi_i^{ra}\}=\pi^{ra}-\{\pi_1^{ra},\pi_2^{ra},,,\pi_p^{ra}\}$, $p=q-1$;

종료.

Else : $a=a_1$, $Q=\{a_1,a_2,a_3,,,a_m\}$;

$Q=Q-\{a_1\}$;

Step 3 : 다음링크($a->b$)로 확장

If : $\pi_p^{ra}+d_{ab}+c_b<\pi_q^{rb}$: $\pi_q^{rb}=\pi_q^{rb}+d_{ab}+c_b$,

$\forall b\in A_j$, $a=(i,j)$, $p,q=1,2,,,k$;

$Q=Q\bigcup\{b\}$;

Go to Step 2;

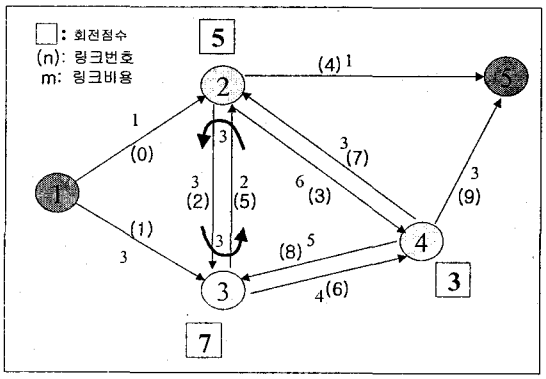
Else : Go To Step 2;

Ⅳ. 사례연구

본 절에서는 3가지의 각각 다른 사례연구를 통해 알고리즘의 수행과정 및 특성을 파악한다. Visual Studio 6.0에서 제공하는 C++언어와 MFC(Microsoft Foundation Class)에서 제공되는 자료구조(Data Structure)를 활용하여 작성되었다.

1. 알고리즘 수행과정

간단하게 표현된 네트워크를 통해 알고리즘의 수행과정을 추적한다. 〈그림 2〉은 적용된 네트워크를 나타내는데 10개의 링크와 5개의 노드로 구성되어 있다. 링크상에는 거리(비용)가 표현되어 있고 노드의 사각형은 링크를 중심으로 전방형 회전점수(turn penalty)를 나타낸다. U-방향회전(U-Turn)은 방향화살표가 존재하는 방향의 2->3->2와 3->2->3 두 지점만 가능하다. 출발지와 도착지는 노드1과 5이며 출발지1에서 도착지까지 5개(K=5)의 경로를 탐색한다.



〈그림 2〉 적용된 네트워크

〈표 1〉은 링크표지로 적용된 다경로알고리즘을 적용하여 출발지1에서 모든 노드까지 5개의 탐색하는 과정을 보여주고 있다. 비용과 함께 표시된 사각형은 노드1에서 노드5까지 k번째 경로가 탐색 되었음을 순차적으로 나타낸 것이다. 이때 현재링크에서 다음 탐색링크로 진행되면서 각 링크까지의 비용이 갱신되는 과정을 나타내고 있다. 표지확정 최단경로 알고리즘과 유사하게 다경로알고리즘에서도 모든 링크까지의 K개의 비용이 더 이상 갱신되지 않을 때까지 알고리즘이 반복 수행된다. 최적경로에서 탐색링크 선정 방법은 우선진입 우선진출(FIFO:First-In-First-Out) 원칙에 의거하나 제안된 다경로 알고리즘에서는 원칙적으로 FIFO이나 다음탐색링크로 선정된 링크가 이미 Q리스트에 존재하는 경우 그 링크를 Q의 가장 뒤에 위치시키게 된다. 이러한 과정이 반복(6과 7)에 나타나 있는데, 반복6에서 링크(4->5)는 Q의 중간에 위치되어 있으나 반복7에서 링크(3->4)의 탐색과정 중 링크(4->5)의 표지갱신이 이루어 졌으므로 다음탐색링크 Q의 후미로 위치이동 하게 된다. 각 링크에서 k번째 경로는 Bellman의 최적원리(1957)에 근거해서 전링크(Previous Link)와 전링크까지의 k번째 경로를 통해서 출발노드까지 역 추적된 경로를 통하여 발견된다.

링크(2->5)에서 5개의 경로비용은 (7, 18, 20, 26, 32)이고 링크(4->5)에서 (18, 20, 26, 29, 31)이므로 노드5까지의 5개 순서의 비용은 링크(2->5)의 상위 3개, 링크(4->5)의 상위 2개 비용으로 구성될 수 있다. 각 비용벡터에 관련된 전링크와 전링크의 k번째 경로를 역추적하면 출발지 노드1까지의 다음의 5개 경로를 구할 수 있게 된다.

〔K=1〕 (비용 : 7.0) : 1->2->5
〔K=2〕 (비용 : 18.0) : 1->3->2->5
〔K=3〕 (비용 : 18.0) : 1->2->4->5
〔K=4〕 (비용 : 20.0) : 1->2->3->2->5
〔K=5〕 (비용 : 20.0) : 1->3->4->5

링크(2->5)의 첫번째 경로를 예를 들면, 지원링크번호가 0이고 0번 링크의 0번째 경로에서 출발한 경로이므로 링크0(1->2)의 비용은 1이고 0번째 경로를 따라 링크(2->5)까지 도달하기 위해서는 회전페널티 5와 (2->5)의 링크비용의 합산이므로 1+5+1=7 된다.

2. 회전금지 및 U-Turn

도시가로망의 교차로에서 회전금지와 U-Turn이 통행의 효율성을 증진시킬 목적으로 규제방법으로 활용된다. 본 사례연구에서는 이러한 통행규제를 포함하는 네트워크에서의 알고리즘 활용성을 점검한다. 〈그림 3〉은 수정된 Sioux Fall Network(26노드, 76링크)를 대상으로 회전금지 15방향에서 존재하며, U-Turn은 한 방향(11->14->11)을 제외하고 전방향에서 금지된다.

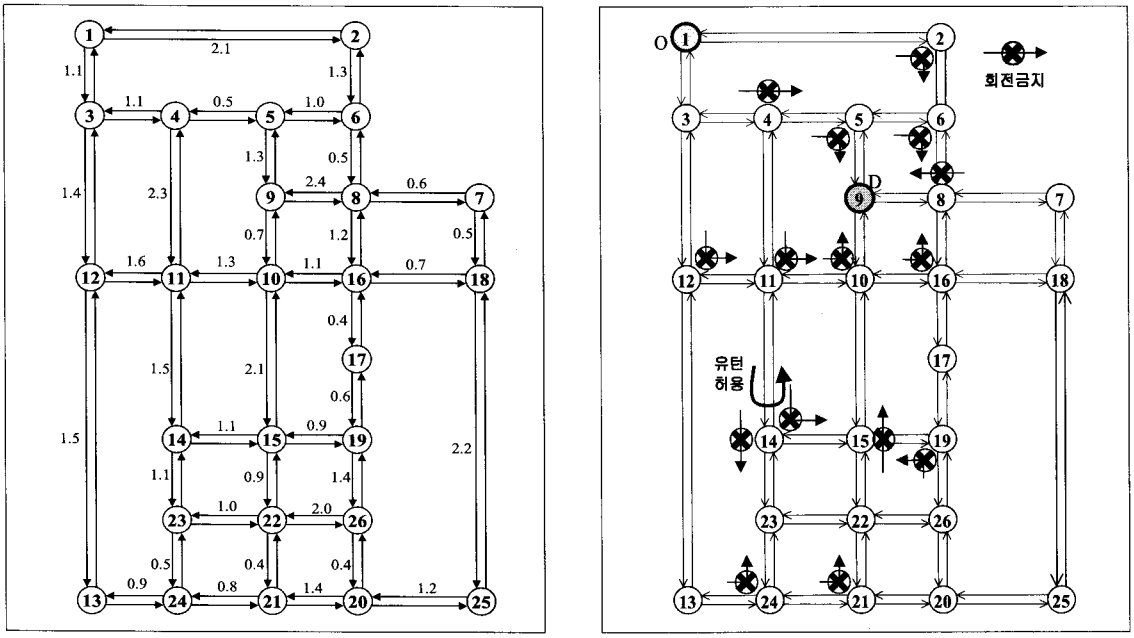
〈표 1〉 출발지에서 전체노드까지 탐색경로발견(K=5)

반복	현재 탐색링크	다음 탐색링크	5개의 비용	전링크(전링크의 k번째경로) 0~9(0~4)
0	초기화	(1->2) (1->3)	(1, ∞, ∞, ∞, ∞) (3, ∞, ∞, ∞, ∞)	(r(0), X(X), X(X), X(X), X(X)) (r(0), X(X), X(X), X(X), X(X))
1	(1->2)	(1->3) (2->3) (2->4) (2->5)	(3, ∞, ∞, ∞, ∞) (9, ∞, ∞, ∞, ∞) (12, ∞, ∞, ∞, ∞) (7, ∞, ∞, ∞, ∞)	(r(0), X(X), X(X), X(X), X(X)) (0(0), X(X), X(X), X(X), X(X)) (0(0), X(X), X(X), X(X), X(X)) (0(0), X(X), X(X), X(X), X(X))
2	(1->3)	(2->3) (2->4) (2->5) (3->2) (3->4)	(9, ∞, ∞, ∞, ∞) (12, ∞, ∞, ∞, ∞) (7, ∞, ∞, ∞, ∞) (12, ∞, ∞, ∞, ∞) (14, ∞, ∞, ∞, ∞)	(0(0), X(X), X(X), X(X), X(X)) (0(0), X(X), X(X), X(X), X(X)) (0(0), X(X), X(X), X(X), X(X)) (1(0), X(X), X(X), X(X), X(X)) (1(0), X(X), X(X), X(X), X(X))
3	(2->3)	(2->4) (2->5) (3->2) (3->4)	(14, ∞, ∞, ∞, ∞) (7, ∞, ∞, ∞, ∞) (12, 14, ∞, ∞, ∞) (14, 20, ∞, ∞, ∞)	(0(0), X(X), X(X), X(X), X(X)) (0(0), X(X), X(X), X(X), X(X)) (1(0), 2(0), X(X), X(X), X(X)) (1(0), 2(0), X(X), X(X), X(X))
4	(2->4)	(2->5) (3->2) (3->4) (4->3) (4->5)	(7, ∞, ∞, ∞, ∞) (12, 14, ∞, ∞, ∞) (14, 20, ∞, ∞, ∞) (20, ∞, ∞, ∞, ∞) (18, ∞, ∞, ∞, ∞)	(0(0), X(X), X(X), X(X), X(X)) (1(0), 2(0), X(X), X(X), X(X)) (1(0), 2(0), X(X), X(X), X(X)) (3(0), X(X), X(X), X(X), X(X)) (3(0), X(X), X(X), X(X), X(X))
5	(2->5)	(3->2) (3->4) (4->3) (4->5)	(12, 14, ∞, ∞, ∞, ∞) (14, 20, ∞, ∞, ∞, ∞) (20, ∞, ∞, ∞, ∞) (18, ∞, ∞, ∞, ∞)	(1(0), 2(0), X(X), X(X), X(X)) (1(0), 2(0), X(X), X(X), X(X)) (3(0), X(X), X(X), X(X), X(X)) (3(0), X(X), X(X), X(X), X(X))
6	(3->2)	(3->4) (4->3) (4->5) (2->3) (2->4) (2->5)	(12, 14, ∞, ∞, ∞) (20, ∞, ∞, ∞, ∞) (18, ∞, ∞, ∞, ∞) (9, 18, ∞, ∞, ∞) (12, 23, 25, ∞, ∞) (7, 18, 20, ∞, ∞)	(1(0), 2(0), X(X) , X(X) , X(X)) (3(0), X(X), X(X) , X(X) , X(X)) (3(0), X(X), X(X) , X(X) , X(X)) (0(0), 5(0), X(X) , X(X) , X(X)) (0(0), 5(0), 5(1) , X(X) , X(X)) (0(0), 5(0), 5(1) , X(X) , X(X))
7	(3->4)	(4->3) (2->3) (2->4) (2->5) (4->2) (4->5)	(20.0, ∞, ∞, ∞, ∞) (9, 18, ∞, ∞, ∞) (12, 23, 25, ∞, ∞) (7, 18, 20, ∞, ∞) (20, 26, ∞, ∞, ∞) (18, 20, 26, ∞, ∞)	(3(0), X(X), X(X), X(X), X(X)) (0(0), 5(0), X(X), X(X), X(X)) (0(0), 5(0), 5(1), X(X), X(X)) (0(0), 5(0), 5(1), X(X), X(X)) (6(0), 6(1), X(X), X(X), X(X)) (3(0), 6(0), 6(1), X(X), X(X))
8	(4->3)	(2->3) (2->4) (2->5) (4->2) (4->5) (3->2)	(9, 18, ∞, ∞, ∞) (12, 23, 25, ∞, ∞) (7, 18, 20, ∞, ∞) (20, 26, ∞, ∞, ∞) (18, 20, 26, ∞, ∞) (12, 14, 29, ∞, ∞)	(0(0), 5(0), X(X), X(X), X(X)) (0(0), 5(0), 5(1), X(X), X(X)) (0(0), 5(0), 5(1), X(X), X(X)) (6(0), 6(1), X(X), X(X), X(X)) (3(0), 6(0), 6(1), X(X), X(X)) (1(0), 2(0), 8(0), X(X), X(X))
9	(2->3)	(2->4) (2->5) (4->2) (4->5) (3->2) (3->4)	(12, 23, 25, ∞, ∞) (7, 18, 20, ∞, ∞) (20, 26, ∞, ∞, ∞) (18, 20, 26, ∞, ∞) (12, 14, 29, ∞, ∞) (14, 20, 29, ∞, ∞)	(0(0), 5(0), 5(1), X(X), X(X)) (0(0), 5(0), 5(1), X(X), X(X)) (6(0), 6(1), X(X), X(X), X(X)) (3(0), 6(0), 6(1), X(X), X(X)) (1(0), 2(0), 8(0), X(X), X(X)) (1(0), 2(0), 2(1), X(X), X(X))

〈표 1〉 출발지에서 전체노드까지 탐색경로발견(K=5)(계속)

반복	현재 탐색링크	다음 탐색링크	5개의 비용	전링크(전링크의 k번째경로) 0~9(0~4)
10	(2->4)	(2->5) (4->2) (3->2) (3->4) (4->3) (4->5)	(7, 18, 20, ∞, ∞) (20, 26, ∞, ∞, ∞) (12, 14, 29, ∞, ∞) (14, 20, 29, ∞, ∞) (20, 31, 33, ∞, ∞) (18, 20, 26, 29, 31)	(0(0), 5(0), 5(1), X(X), X(X)) (6(0), 6(1), X(X), X(X), X(X)) (1(0), 2(0), 8(0), X(X), X(X)) (1(0), 2(0), 2(1), X(X), X(X)) (3(0), 3(1), 3(2), X(X), X(X)) (3(0), 6(0), 6(1), 3(1), 3(2))
11	(2->5)	(4->2) (3->2) (3->4) (4->3) (4->5)	(20, 26, ∞, ∞, ∞) (12, 14, 29, ∞, ∞) (14, 20, 29, ∞, ∞) (20, 31, 33, ∞, ∞) (18, 20, 26, 29, 31)	(6(0), 6(1), X(X), X(X), X(X)) (1(0), 2(0), 8(0), X(X), X(X)) (1(0), 2(0), 2(1), X(X), X(X)) (3(0), 3(1), 3(2), X(X), X(X)) (3(0), 6(0), 6(1), 3(1), 3(2))
12	(4->2)	(3->2) (3->4) (4->3) (4->5) (2->3) (2->5)	(12, 14, 29, ∞, ∞) (14, 20, 29, ∞, ∞) (20, 31, 33, ∞, ∞) (18, 20, 26, 29, 31) (9, 18, 28, ∞, ∞) (7, 18, 20, 26, 32)	(1(0), 2(0), 8(0), X(X), X(X)) (1(0), 2(0), 2(1), X(X), X(X)) (3(0), 3(1), 3(2), X(X), X(X)) (3(0), 6(0), 6(1), 3(1), 3(2)) (0(0), 5(0), 7(0), X(X), X(X)) (0(0), 5(0), 5(1), 7(1), 7(2))
13	(3->2)	(3->4) (4->3) (4->5) (2->5) (2->3)	(14, 20, 29, ∞, ∞) (20, 31, 33, ∞, ∞) (18, 20, 26, 29, 31) (7, 18, 20, 26, 32) (9, 18, 28, 35, ∞)	(1(0), 2(0), 2(1), X(X), X(X)) (3(0), 3(1), 3(2), X(X), X(X)) (3(0), 6(0), 6(1), 3(1), 3(2)) (0(0), 5(0), 5(1), 7(1), 7(2)) (0(0), 5(0), 7(0), 5(2), X(X))
14	(3->4)	(4->3) (4->5) (2->5) (2->3) (4->2)	(20, 31, 33, ∞, ∞) (18, 20, 26, 29, 31) (7, 18, 20, 26, 32) (9, 18, 28, 35, ∞) (20, 26, 35, ∞, ∞)	(3(0), 3(1), 3(2), X(X), X(X)) (3(0), 6(0), 6(1), 3(1), 3(2)) (0(0), 5(0), 5(1), 7(1), 7(2)) (0(0), 5(0), 7(0), 5(2), X(X)) (6(0), 6(1), 6(2), X(X), X(X))
15	(4->3)	(4->5) (2->5) (2->3) (4->2)	(18, 20, 26, 29, 31) (7, 18, 20, 26, 32) (9, 18, 28, 35, ∞) (20, 26, 35, ∞, ∞)	(3(0), 6(0), 6(1), 3(1), 3(2)) (0(0), 5(0), 5(1), 7(1), 7(2)) (0(0), 5(0), 7(0), 5(2), X(X)) (6(0), 6(1), 6(2), X(X), X(X))
16	(4->5)	(2->5) (2->3) (4->2)	(7, 18, 20, 26, 32) (9, 18, 28, 35, ∞) (20, 26, 35, ∞, ∞)	(0(0), 5(0), 5(1), 7(1), 7(2)) (0(0), 5(0), 7(0), 5(2), X(X)) (6(0), 6(1), 6(2), X(X), X(X))
17	(2->5)	(2->3) (4->2)	(9, 18, 28, 35, ∞) (20, 26, 35, ∞, ∞)	(0(0), 5(0), 7(0), 5(2), X(X)) (6(0), 6(1), 6(2), X(X), X(X))
18	(2->3)	(4->2) (3->2) (3->4)	(20, 26, 35, ∞, ∞) (12, 14, 29, 33, ∞) (14, 20, 29, 46, ∞)	(6(0), 6(1), 6(2), X(X), X(X)) (1(0), 2(0), 8(0), 2(2), X(X)) (1(0), 2(0), 2(1), 2(3), X(X))
19	(4->2)	(3->2) (3->4)	(12, 14, 29, 33, ∞) (14, 20, 29, 46, ∞)	(1(0), 2(0), 8(0), 2(2), X(X)) (1(0), 2(0), 2(1), 2(3), X(X))
20	(3->2)	(3->4) (2->4)	(14, 20, 29, 46, ∞) (12, 23, 25, 44, ∞)	(1(0), 2(0), 2(1), 2(3), X(X)) (0(0), 5(0), 5(1), 5(3), X(X))
21	(3->4)	(2->4) (4->2)	(12, 23, 25, 44, ∞) (20, 26, 35, 52, ∞	(0(0), 5(0), 5(1), 5(3), X(X)) (6(0), 6(1), 6(2), 6(3), X(X))
22	(2->4)	(4->2) (4->3)	(20, 26, 35, 52, ∞) (20, 31, 33, 52, ∞)	(6(0), 6(1), 6(2), 6(3), X(X)) (3(0), 3(1), 3(2), 3(3), X(X))
23	(4->2)	(4->3)	(20, 31, 33, 52, ∞)	(3(0), 3(1), 3(2), 3(3), X(X))
24	(4->3)	-	-	-

r : 출발지에 인접



〈그림 3〉 교통망과 통행규제(U-Turn/회전금지)

〈표 2〉는 출발지1에서 도착지 9까지이 통행규제를 고려, 총 362번의 반복수행을 통해 20개의 경로를 탐색한 결과이다. 결과에서 나타나듯이 K개의 경로를 탐색하는 과정에서 순차적으로 비용이 증가되며, 탐색된 경로에서 유턴이나 피턴이나 합리적인 통행현상이 반영되고 있다. 서술했듯이 합리적인 통행현상에는 경로를 구성함에 있어 링크의 반복은 발생하지 않는다.

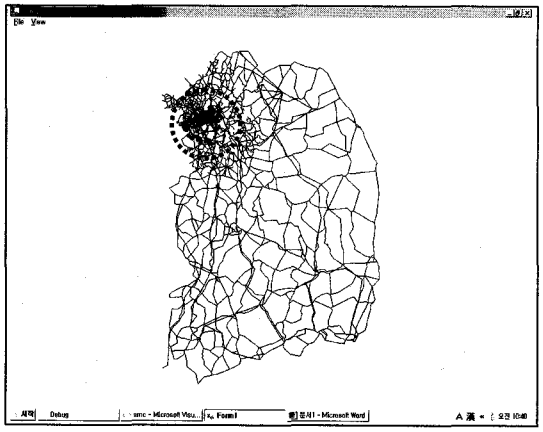
〈표 2〉 통행규제가 고려된 경로탐색(K=20)

순서	비용	노드순서에 의한 경로표시
1	11.70	9〈-10〈-16〈-17〈-19〈-26〈-20〈-21〈-24〈-13〈-12〈-3〈-1
2	13.00	9〈-10〈-16〈-18〈-25〈-20〈-21〈-24〈-13〈-12〈-3〈-1
3	13.50	9〈-8〈-16〈-17〈-19〈-26〈-20〈-21〈-24〈-13〈-12〈-3〈-1
4	13.90	9〈-5〈-6〈-8〈-16〈-17〈-19〈-26〈-20〈-21〈-24〈-13〈-12〈-3〈-1
5	14.10	9〈-10〈-16〈-17〈-19〈-15〈-22〈-26〈-20〈-21〈-24〈-13〈-12〈-3〈-1
6	14.40	9〈-5〈-6〈-8〈-7〈-18〈-25〈-20〈-21〈-24〈-13〈-12〈-3〈-1
7	14.50	9〈-5〈-6〈-8〈-7〈-18〈-16〈-10〈-[11]〈-14〈-[11]〈-4〈-3〈-1 (*)
8	14.50	9〈-5〈-6〈-8〈-7〈-18〈-16〈-17〈-19〈-26〈-20〈-21〈-24〈-13〈-12〈-3〈-1
9	14.60	9〈-[10]〈-15〈-19〈-17〈-16〈-[10]〈-[11]〈-14〈-[11]〈-4〈-3〈-1 (*)
10	14.60	9〈-[10]〈-16〈-17〈-19〈-15〈-[10]〈-[11]〈-14〈-[11]〈-4〈-3〈-1 (*)
11	14.60	9〈-10〈-16〈-8〈-7〈-18〈-25〈-20〈-21〈-24〈-13〈-12〈-3〈-1
12	14.70	9〈-10〈-[16]〈-8〈-7〈-18〈-[16]〈-10〈-[11]〈-14〈-[11]〈-4〈-3〈-1 (*)
13	14.70	9〈-10〈-[16]〈-8〈-7〈-18〈-[16]〈-17〈-19〈-26〈-20〈-21〈-24〈-13〈-12〈-3〈-1 (*)
14	14.70	9〈-10〈-[16]〈-18〈-7〈-8〈-[16]〈-17〈-19〈-26〈-20〈-21〈-24〈-13〈-12〈-3〈-1 (*)
15	14.80	9〈-8〈-16〈-18〈-25〈-20〈-21〈-24〈-13〈-12〈-3〈-1
16	15.20	9〈-5〈-6〈-8〈-16〈-18〈-25〈-20〈-21〈-24〈-13〈-12〈-3〈-1
17	15.30	9〈-10〈-16〈-17〈-19〈-26〈-20〈-21〈-24〈-13〈-12〈-11〈-4〈-3〈-1
18	15.50	9〈-10〈-15〈-14〈-23〈-22〈-26〈-20〈-21〈-24〈-13〈-12〈-3〈-1
19	15.90	9〈-8〈-16〈-17〈-19〈-15〈-22〈-26〈-20〈-21〈-24〈-13〈-12〈-3〈-1
20	15.90	9〈-10〈-15〈-19〈-17〈-16〈-18〈-25〈-20〈-21〈-24〈-13〈-12〈-3〈-1

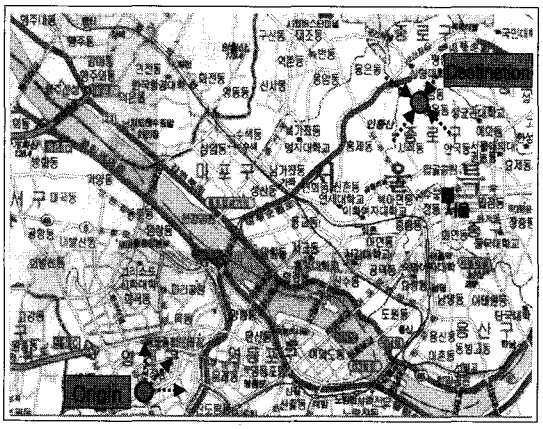
(*) : 경로상 복수노드 □가 나타난 합리적 통행행태

3. 대규모 네트워크 및 알고리즘개선

본 사례연구에서는 대규모 네트워크를 대상으로 알고리즘의 현실적 활용가능성을 모색한다. 실제 네트워크에 적용되기 위해서 알고리즘의 문제점을 파악하고 개선방안을 모색해 본다. 네트워크는 〈그림 4〉와 같이 우리나라의 내륙지역 전체를 대상으로 하며, 37944개의 링크와, 15680개의 노드로 구성되어 있다. 출발지와 도착지는 서울시의 임의의 두 지점으로, 출발지는 서울시의 양천구 신월동의 한 지점과 도착지는 종로구 부암동의 한 지점이다. 네트워크에서 두 지점은 각각 10151과 105737번의 노드번호를 가지고 있다. 출발지에서 목적지에 접근하기 위해서는 서울시를 관통하는 한강상 교량을 통과해야 한다. 〈그림 3〉에서 나타나는 주요통과 예상 교량은 성산대교, 양화대교, 서강대교, 마포대교가 있다.



(a) 적용 네트워크



(b) 출발지와 도착지

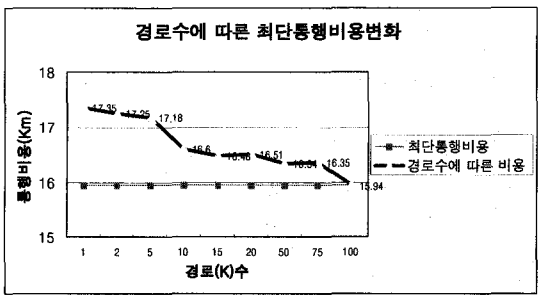
〈그림 4〉 적용된 네트워크와 기종점

1) 대안경로 활용의 문제점

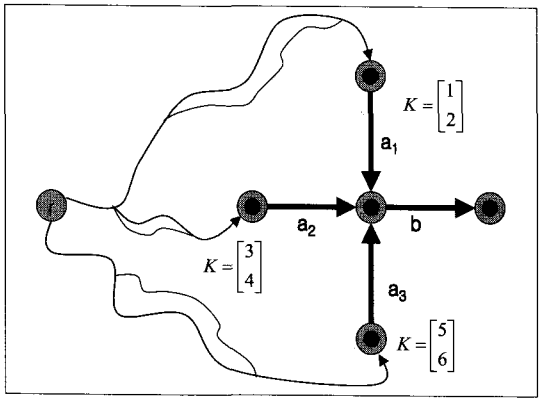
운전자에게 제공되는 경로정보의 대안은 운전자가 비교적 용이하게 경로의 특성을 이해하고 선택하도록 구성되어야 한다. 이를 위해 일반적으로 경로대안이 갖추어야 할 중요한 조건으로서 운전자가 선별 가능하도록 한정된 수개의 대안을 제공하여 경로의 선택이 용이하게 이루어지도록 되어야 한다. 그러나 본 연구에서 제안하는 링크표지갱신 다경로알고리즘을 대규모 네트워크에 적용하는 경우, K의 수가 적을수록 최적해에 근접하기 어렵다는 단점을 가지고 있다. 그림은 〈그림 5〉의 네트워크를 K값에 따라 알고리즘에서 도출된 첫번째 경로의 통행비용을 최적경로 알고리즘(Moore, 1957; Dijkstra, 1959)에서 도출한 값과 비교한 것이다. 보는 바와 같이, K값이 작을수록(1에 근접할수록) 최적해에서 멀어지며, K값이 커질수록(100에서) 최적해와 일치되는 것으로 나타났다. 이는 수개의 대안만을 도출하여 대안경로로서 적용하는 것은 알고리즘이 갖는 약점 때문에 한계가 있으며, 최적해에 근접하기 위해서는 컴퓨터의 K를 충분히 증가시켜야 하고, 이는 검퓨터 수행시간을 매우 길게 가져 손실을 유발함을 의미한다(참고로 본 연구에는 명시하지 않았으나 네트워크의 규모가 작은 〈그림 3〉의 Sioux Fall Network의 경우에는 이러한 문제점이 발생하지 않았다.).

이처럼 대규모 네트워크의 적용에서 문제가 발생되는 이유는 출발지에서 각 링크로 탐색을 확대해 나가는 동안 해의 퇴화(Degenerate)의 발생확률이 네트워크가 커질수록 크게 증가되기 때문이다.

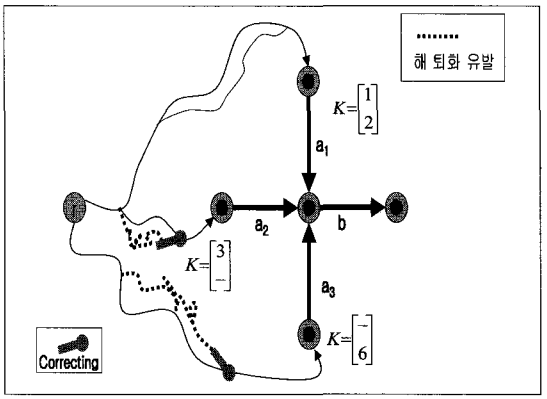
〈그림 6〉은 출발지 r에서 링크(b)까지 6개 (K=6)의 경로를 탐색하기 위한 링크갱신과정을 예로 나타낸 것이다. 6개의 경로를 갱신하기 위해 우선 링크(b)와 인접한 3개의 전링크 a_1, a_2, a_3의 최단비용



〈그림 5〉 경로수에 따른 알고리즘의 최단통행비용



〈그림 6〉 출발지r에서 링크b까지 K개의 비용탐색



〈그림 7〉 K개의 비용탐색과정 해의 퇴화현상

과 전링크에서 링크(b)까지의 회전비용(d_{a_1b}, d_{a_2b}, d_{a_3b}), 그리고 링크(b)의 비용(c_b)을 고려되어 링크(b)비용이 갱신되게 된다. 이때 링크(b)의 비용을 구성하기 위하여 a_1은 첫 번째와 두 번째의 전링크로, 링크 a_2은 세 번째와 네 번째의 전링크, 링크 a_3은 다섯 번째와 여섯 번째로 전링크를 구성하고 있다. 이들 관계를 고려하여 링크(b)까지의 6개 비용의 갱신과정이 식(3)에 나타나 있다.

$$[\pi^{rb}]^T=\begin{Bmatrix}\pi_1^{rb}\\ \pi_2^{rb}\\ \pi_3^{rb}\\ \pi_4^{rb}\\ \pi_5^{rb}\\ \pi_6^{rb}\end{Bmatrix}=\begin{Bmatrix}\pi_1^{ra_1}+d_{a_1b}+c_b\\ \pi_2^{ra_1}+d_{a_1b}+c_b\\ \pi_1^{ra_1}+d_{a_2b}+c_b\\ \pi_2^{ra_2}+d_{a_2b}+c_b\\ \pi_1^{ra_3}+d_{a_3b}+c_b\\ \pi_2^{ra_3}+d_{a_3b}+c_b\end{Bmatrix} \tag{3}$$

해의 퇴화문제는 링크(b)의 6개의 경로를 구성하기 위해 포함되어 있는 링크의 갱신과정이 계속 진행되면서 발생된다. 〈그림 7〉은 4번째와 5번째의 경로에 포함되어 경로의 비용을 구성하고 있는 링크를 대상으로 갱신과정이 다시 진행된 것을 예로 보여준 것이다. 이 갱신과정에서 그 링크까지의 전링크가 바뀌게 되면 출발지에서 이들 링크까지의 경로와 이 링크에서 링크(b)까지의 경로를 연결하면(링크가 경로상에 1번만 나타나는) 합리적 통행가정을 위반하는 loop형 경로가 발생하게 된다. 그러므로 링크 (b)의 4번째와 5번째의 경로는 해의 구성조건인 합리적 통행현상을 만족하지 못하므로 삭제됨이 타당하다. 식(4)는 이미 갱신된 링크(b)의 비용에서 해의 네 번째와 다섯 번째의 경로에서 loop가 포함되었으므로 이를 퇴화현상으로 인지하고, 퇴화된 경로를 삭제하여 상위 4개로 재구성된 링크(b)의 비용을 나타내고 있다. 이러한 과정에서 K값이 적을 수록, 전체적인 해의 질은 떨어질 확률이 높게 된다.

$$[\pi^{rb}]^T=\begin{Bmatrix}\pi_1^{rb}\\ \pi_2^{rb}\\ \pi_3^{rb}\\ \pi_4^{rb}\\ \pi_5^{rb}\\ \pi_6^{rb}\end{Bmatrix}=\begin{Bmatrix}\pi_1^{ra_1}+d_{a_1b}+c_b\\ \pi_2^{ra_1}+d_{a_1b}+c_b\\ \pi_1^{ra_2}+d_{a_2b}+c_b\\ \pi_2^{ra_2}+d_{a_2b}+c_b\\ \pi_1^{ra_3}+d_{a_3b}+c_b\\ \pi_2^{ra_3}+d_{a_3b}+c_b\end{Bmatrix}$$

$$\Rightarrow\begin{Bmatrix}\pi_1^{ra_1}+d_{a_1b}+c_b\\ \pi_2^{ra_1}+d_{a_1b}+c_b\\ \pi_1^{ra_2}+d_{a_2b}+c_b\\ \infty\\ \infty\\ \pi_2^{ra_3}+d_{a_3b}+c_b\end{Bmatrix}\Rightarrow\begin{Bmatrix}\pi_1^{ra_1}+d_{a_1b}+c_b\\ \pi_2^{ra_1}+d_{a_1b}+c_b\\ \pi_1^{ra_2}+d_{a_2b}+c_b\\ \pi_2^{ra_3}+d_{a_3b}+c_b\\ \infty\\ \infty\end{Bmatrix} \tag{4}$$

2) (K-1)차원 경로탐색 알고리즘

해의 퇴화현상을 완화 또는 극복하기 위하여 본 연구에서 (K-1)차원 다수경로 탐색알고리즘을 제안한다. 제안된 알고리즘은 출발지에서 각 링크의 최적경로(1st)는 정해진 값이므로 우선 최적경로는 고정시켜서 해의 퇴화를 최적경로에서 발생시키는 것을 우선적으로 방지하는 것이다. 이는 최적경로 1차원은 고정시켜 네트워크의 문제를 (K)차원에서 (K-1)차원으로 축소하는 효과를 가져오며, 모든 링크까지의 고정된 최적해 영향으로 인접 링크의 첫 번째이후(K>1)의 해 값도 개선되도록 유도하게 된다.

(K-1)차원으로 축소된 링크표지갱신 다경로알고리즘은 다음과 같다. (K)차원 알고리즘과에 비해 (K-1)

차원 알고리즘의 차이점은 크게 두 가지로서 1) Step 1 초기화 과정에서 링크표지기반 최적경로알고리즘을 활용하여 출발지에서 모든 링크까지의 최적경로와 비용을 먼저 탐색해서 고정시키고 ($\pi^{ra}=\{\pi_1^{ra},\infty,,,\infty\}$), 2) Step 3의 링크($b$)의 비용갱신 과정에서 첫번째 경로 (K=1)의 비용은 이미 최적해가 초기화 과정에서 고정되었으므로 경로갱신과정에서 고려하지 않는다는 ($q=2,,,k$) 것이다.

Step 1 : 초기화

SPTV로 π_1^{ra}의 결정, $\forall a\in L$;

$\pi^{ra}=\{\pi_1^{ra},\infty,,,\infty\}$, $\forall a\in L$, $r\in R$;

$Q=Q\bigcup\{a\}$, $\forall a\in A_r$;

Step 2 : 다음탐색링크결정

if : $Q=\phi$: $\pi_q^{ri}=\min[\{\pi^{ra}\}-\{\pi_1^{ra},\pi_2^{ra},,,\pi_p^{ra}\}]$,

$a\in B_i$, $p=q-1$;

종료.

Else : $a=a_1$, $Q=\{a_1,a_2,a_3,,,a_m\}$;

$Q=Q-\{a_1\}$;

Step 3 : 회전지체를 고려한 다음링크($a->b$)로 확장

If : $\pi_p^{ra}+d_{ab}+c_b<,\pi_q^{rb}$: $\pi_q^{rb}=\pi_p^{ra}+d_{ab}+c_b$,

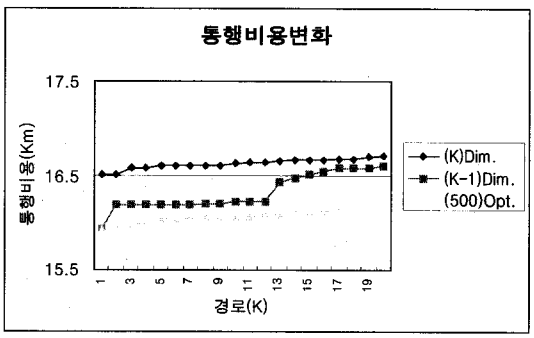
$\forall b\in A_j$, $a=(i,j)$, $p=1,2,,,k$, $q=2,,,k$;

$Q=Q\bigcup\{b\}$;

Go to Step 2;

Else : Go To Step 2;

〈그림 8〉은 20개 경로를 탐색하기 위해 (K=20)과 (K-1=19)차원의 알고리즘으로 도출한 값을 비교한 것을 보여주고 있다. 여기서 최적해(Optimal Solution)는 (K=500)으로 도출한 상위 20개의 경로를 나타낸 것이다. 〈그림 8〉에서 보는 바와 같이 (K-1)차원에서 경로의 수가 증가함에도 불구하고 10개 경로 미만까지는 최적해에 보다 가까운 결과를 유지하는 것으로 파악되었다. 이는 대안경로의 수가 작은 경우에는 (K-1)차원을 고려하는 알고리즘이 수행노력대비 향상된 해를 제공하는 것을 의미한다고 볼 수 있다.



〈그림 8〉 경로수에 따른 최적통행비용의 변화과정

V. 결론

교통망에서 네트워크 이론을 적용하여 다수경로의 합리적 통행을 고려하는 알고리즘을 제안하는 것이 본 연구의 주요 목적이다. 여기서 합리적인 통행이란 "경로를 노드 또는 링크의 일련의 순서로 표현할 때, 경로를 구성하는 노드의 반복은 허용해도 링크의 반복은 허용하지 않는다"는 것이다. 최적경로 탐색과정에서 링크로 구성된 표지를 적용하는 경우 이러한 합리적 통행현상을 알고리즘에 고려 가능함이 최적경로 알고리즘으로 이미 규명되었다. 본 연구는 기존에 노드표지갱신(Node Label Correcting)기반으로 제안되었던 다경로알고리즘을 링크표지로 전환하는 방안을 제시했다. 합리적 통행을 고려한 경로탐색이 이루어 지는가를 파악하기 위하여 회전지체 및 통행금지 등 도시가로망 특성이 반영된 모의 네트워크를 통해 알고리즘의 수행성을 확인하였다. 대안경로제공의 역할을 수행하기 위한 가능성을 타진하기 위해 소수의 대안경로를 제공하는 방안을 중심으로 알고리즘의 한계를 규명하고, (K)차원의 알고리즘을 (K-1)차원으로 축소하는 방안을 적용하여 보다 개선된 기능을 수행하도록 개선하였다. 결론적으로 (K-1)차원 링크표지갱신 알고리즘의 제안으로 최적경로알고리즘위주로 개발되었던 링크표지기반 알고리즘이 갖는 장점을 수용하는 하면서도 대안경로선정을 위한 실용적인 개발방향으로 한 단계 나아갔음이 확인되었다.

추후연구과제로서 표지갱신 다수경로알고리즘이 링크비용이 음의 값을(Negative Cost) 갖는 경우에도 적용이 가능하다는 점을 활용하는 것으로, 비용과 링크편익(Benefit)으로 고려되는 경우의 현실적용에 알고리즘의 장점이 고려될 수 있을 것으로 기대된다. 예를 들면 운전자에게 특정 링크 또는 지점을 통행하는 경우 소요된 비용을 편익으로 환산해주는 경우의 예가 있을 수 있다.

참고문헌

1. 김현명 · 임용택(1999), "유전 알고리듬을 이용한 전역탐색 최단경로 알고리듬개발", 대한교통학회지, 제17권 제2호, 대한교통학회, pp.163~178.
2. 노정현 · 남궁성(1995), "도시가로망에 적합한 최단경로탐색기법의 개발", 국토계획, 제30권 제5호, 대한국토도시계획학회, pp.153~168.
3. 이미영 · 유기윤 · 김정현 · 신성일(2003), "덩굴망 통행패턴을 고려한 One-To-One 다경로알고리즘", 대한교통학회지, 제21권 제6호, 대한교통학회, pp.89~99.
4. 이승환 · 최기주 · 김원길(1996), "도시부 ATIS 효율적 적용을 위한 탐색영역기법 및 양방향 링크탐색 알고리즘의 구현", 대한교통학회지, 제14권 제3호, pp.45~59.
5. 임강원 · 임용택(2003), 교통망분석론, 서울대학교출판부.
6. 장인성(2000), "서비스시간 제약이 존재하는 도시부 복합교통망을 위한 링크기반의 최단경로탐색 알고리즘", 대한교통학회지, 제18권 제6호, 대한교통학회, pp.111~121.
7. Azevedo J. A., Costa M. E. O. S., Madeira J. J. E. R. S., and Martins E. Q. V.(1993), An Algorithm from the Ranking of Shortest Paths, European Journal of Operational Research, Vol.69, pp.97~106.
8. Bellman R.(1957), Dynamic Programming, Princeton University Press, Princeton, New Jersey.
9. Bellman R. and Kalaba R.(1968), On Kth Best Policies. J. SIAM 8, pp.582~588.
10. Dijkstra E. W.(1959) A Note of Two Problems in Connected with Graphs. Numerical Mathematics. I, pp.269~271.
11. Kirby R. F. and Potts R. B.(1969) The Minimum Route Problem for Networks with Turn Penalties and Prohibitions. Transportation Research 3, pp.397~408.
12. Martins E. Q. V.(1984) An Algorithm for Ranking Paths that May Contain Cycles, European Journal of Operational Research, Vol.18, pp.123~130.
13. Moore E. F.(1957) The Shortest Path through A Maze. Proc. Int. Conf. on the Theory of Switching. Harvard Univ., Cambridge, MA.
14. Pollack M.(1961) The Kth Best Route Through A Network, Operations Research, Vol.9, pp.578~580.
15. Potts R. B. and Oliver R. M.(1972) Flows in Transportation Networks. Academic Press.
16. Shier R. D.(1979) On Algorithms from Finding the k Shortest Paths in a Network, Networks, Vol.9, pp.195~214.
17. Yen J. Y.(1971) Finding the K shortest Loopless Paths in a Network, Management Science, Vol.17, pp.711~715.

♧ 주 작 성 자 : 이미영
♧ 논문투고일 : 2004. 1. 27
논문심사일 : 2004. 3. 2 (1차)
2004. 3. 24 (2차)
2004. 3. 31 (3차)
심사판정일 : 2004. 3. 31
♧ 반론접수기한 : 2004. 8. 31