도시부 ATIS 적용을 위한 동적 다중 표지 덩굴망 알고리즘의 개발

A Development of Dynamic Multi-labelling Vine Algorithm for Urban ATIS Application

박 상 준 (한양대학교 교통공학과 석사과정)

김 익 기 (한양대학교 교통공학과 부교수)

목 차

요 약

기존에 건설된 교통시설의 운영효율을 극대화하기 위한 지능형 교통체계의 한 분야인 ATIS는 도로를 이용하는 통행자에게 편리성을 제공하는 동시에 도로를 효율적으로 운영할 수 있는 정보체계이다. ATIS 체계하에서 통행자에게 신뢰성 있는 정보를 제공하기 위해서는 교차로서의 회전에 의한 지체를 정확하게 반영함은 물론이고 실시간으로 변화하는 교통상황을 반영할 수 있는 동적인 최단경로 탐색 알고리즘이 요구된다. 하지만 기존에 발표된 동적인 최단경로 탐색 알고리즘은 회전에 대한 정보를 반영하지 못하며 정적인 최단경로 탐색알고리즘 조차 회전에 대한 정보를 정확히 반영하지 못한다.

본 연구에서는 이러한 이유에서 알고리즘 내부에서 회전을 반영하기 위해 수정형 덩굴망 알고리즘의 표지기법을 이용하여 동적인 최단경로 탐색알고리즘을 개발하였다. 본 연구에서 개발한 동적인 최단경로 탐색 알고리즘은 정적인 상태의 수정형 덩굴망 알고리즘에 시간에 따라 변화하는 교통상황을 반영하기 위해 시간에 대한 변수를 추가하였다. 이렇게 해서 알고리즘은 시간대별로 변화하는 통행시간을 고려하여 최단 경로를 탐색하게 되며 출발시점을 기준으로 표지를 설정하여 모든 앞선 시간에 대해 경로를 고려하도록 하였다. 매 단계에서 전 노드를 추적하여 회전에 관한 정보를 반영하도록 하였다.

따라서 본 연구에서 개발한 최단경로 탐색 알고리즘은 교차로에서의 회전에 대한 정보와 통행금지 등을 정확히 반영하며 실시간으로 변화하는 통행시간을 반영함으로써 신뢰성 있는 노선정보를 ATIS를 이용하는 통행자들에게 제공하는데 활용될 수 있는 기법이다.

Ⅰ. 서 론

도심 내부의 교통혼잡을 완화시키기 위해 기존 교통시설의 운영효율을 최대화하기 위한 방안중의 하나가 지능형 교통체계(Intelligent Transportation System : ITS)로 이는 기존의 도로교통체계에 전기·전자, 통신등 최신의 첨단기술을 접목시킨 종합교통체계라 할 수 있다. 이러한 ITS의 한 분야인 첨단 여행자 정보 체계(Advanced Traveler Information System : ATIS)의 동적 최적경로 안내 시스템(Dynamic Route Guidance System : DRGS)은 통행자에게 도로상황에 대한 질 높은 정보를 제공하여 통행자들이 그들의 운행에 있어서 최적의 경로를 이용하게 함으로써 개인적 손실을 감소시킬 수 있으며 교통 통제의 개념이 도입되면 전체 교통망의 혼잡의 감소로 시설운영의 효율성을 증대시킬 수 있는 방안이다.

DRGS에서 통행자에 대한 노선정보의 안내 및 유도를 위해 기본적으로 최단경로탐색 알고리즘이 사용되고 있다. 따라서 최단경로 탐색 알고리즘이 도시부의 실시간으로 변화하는 통행패턴과 U-turn, P-turn등 여러 형태의 회전 및 회전에 대한 지체를 반영할 수 있어야만 DRGS가 현실적인 정보를 통행자에게 제공할 수 있다. 하지만 아직까지 실시간으로 변화하는 통행패턴과 여러 형태의 회전 및 회전에 대한 지체를 반영할 수 있는 알고리즘에 대한 연구가 없었다.

따라서 본 연구에서는 도시부의 여러 형태의 회전 및 지체를 반영하여 실시간으로 운전자에게 목적지까지의 최단 경로를 제공하기 위해 출발시간과 각 링크에서의 통행시간을 고려하여 최단경로를 찾는 동적인 최단경로 탐색 알고리즘을 개발하려 한다.

이때 도시부의 여러 형태의 회전 및 지체를 고려하기 위해 덩굴망 알고리즘을 이용한다. 그러나 기존의 덩굴망 알고리즘은 회전의 여러 형태를 적용하는데 한계를 나타내고 있기 때문에 이러한 한계를 극복한 수정형 덩굴망 알고리즘을 사용하기로 한다.

Ⅱ. 기존 연구의 검토

1. 최단경로 탐색 문제 와 알고리즘

최단경로를 탐색하는 문제는 출발노드와 도착노드의 경로상에서 발생하는 비용 의 합을 최소화하는 경로를 찾는 문제이다. 이때 경로상의 비용은 링크에서만 발생하고 링크와 링크의 연결지점으로써 회전의 축이 되는 노드에서의 비용이[1] 링크 비용에 포함되어 있는 것이라면 링크에서 다음링크로 연결되는 방향에 상관없이 그 비용이 동일하다는 기본적인 가정을 하고 있다.[2] 이러한 가정은 특히 교통분야에 이 문제를 응용할 경우 중대한 오류를 야기할 수 있다.

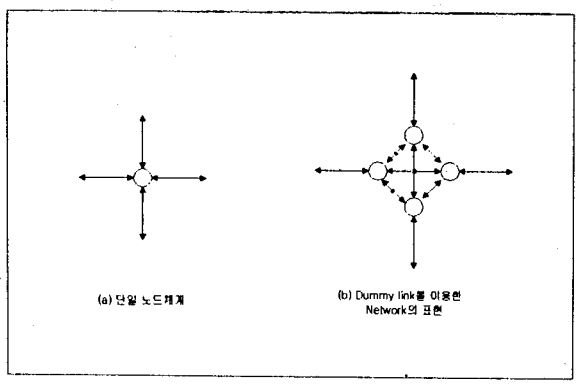
최단경로 알고리즘에서 회전을 반영하기 위한 방법으로는 크게 회전이 일어나는 교차로를 하나의 노드로써 표현하는 단일 노드체계로 구성된 네트워크를 수정하여 기존의 알고리즘을 사용하는 경우와 알고리즘 내부에서 회전을 반영하도록 하는 경우 2가지로 나누어 볼 수 있다.

네트워크를 수정하는 경우는 회전이 발생하는 교차로에 한해서 회전을 고려할 수 있도록 교차로를 나타내는 노드에 접근하는 방향별로 가상의 링크(Dummy link)를 사용하여 이 가상의 링크를 이용하여 회전의 정보를 넣어주는 노드체계를 설정하는 방법과 네트워크 전체에 대해 링크의 중간에 가상의 노드(Dummy Node)를 설정하여 네트워크를 이 가상의 노드로 재구성하여 사용하는 Mid-Block 노드체계를 설정하는 방법이 있다.

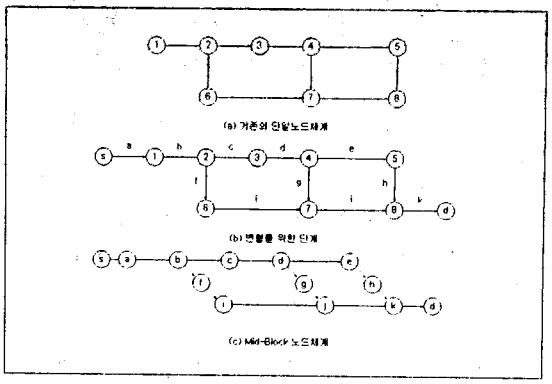
---

1) 이를 회전부담금 이라 하고 링크 외에서 발생하는 비용이라 하여 External Cost라 하기도 한다.

2) 회전부담금은 회전 방향 즉 우회전, 직진, 혹은 좌회전의 경우 각각 다르게 반영되어야 함이 옳지만 이를 링크의 비용에 포함시키게 되면 회전의 방향과는 상관없이 평균값만이 사용될 수밖에 없다.



<그림 1> 접근로별 노드체계



<그림 2> Mid-Block 노드체계

알고리즘 내부에서 회전을 반영할 수 있는 알고리즘에는 덩굴망 알고리즘과 링크 표지 알고리즘들이 있다. 노정현, 남궁성(1995)과 이승환, 최기주와 김원길(1996)에 의해 발표된 링크표지알고리즘(링크표지기법)은 기존의 알고리즘이 노드에 표지를 하는 것과는 다르게 단일 노드체계로 구성된 네트워크에 대해 링크에 표지를 하여 두 개의 링크를 이용해 회전에 대한 방향성을 갖도록 한 것이다. 이러한 링크표지알고리즘은 Mid-Block 노드체계와 개념상에서 크게 다르지 않으며 단지 네트워크의 변형을 알고리즘 내부에서 수행하는 것이다. FHWA에서 제시한 덩굴망 최단경로탐색 알고리즘은 단일 노드체계로 구성된 네트워크에서 노드를 탐색해 나감에 있어 자신의 노드와 자신의 전노드, 자신의 전전노드를 동시에 고려할 수 있어 이를 갖고 회전에 대한 방향성을 고려하게 된다. 하지만 이러한 덩굴망 알고리즘은 두 개 이상의 연속된 좌회전 금지 교차로와 U-turn 및 P-turn 허용 교차로 등에서 비현실적 최단경로를 제공한다고 이미 여러 논문에서 발표되었다. 덩굴망 알고리즘의 이러한 오류는 Bellman의 최적 원리에 의해 한 node에 대해서 하나의 최적 경로만을 찾기 때문에 회전이 제약을 받는 경우 차기 대안을 고려할 수 없다는 것이다.

이와 같은 단점을 개선하기 위해서 통행비용 계산과정과 노선 역추적 과정을 수정하여 수정형 덩굴망 알고리즘이 개발되었다(김익기 1998). 노드에 표지를 설정함에 있어서 노드에 접근하는 방향을 고려하여 경로의 탐색이 종료된 후 경로를 추적할 때 이 방향에 따라 경로를 재구성하는 것이 수정형 덩굴망 알고리즘의 개념이다. 이러한 개념에 의해 Thomas (1991)의 기존 덩굴망 알고리즘과 다르게 김익기(1998)가 제시한 수정형 덩굴망 알고리즘은 하나의 노드에 필요에 의해서 접근로 수만큼의 표지를 갖는다.

이러한 수정형 덩굴망이 그 개념은 충분히 논리적이며 실제 기존의 덩굴망 알고리즘의 한계점을 해결하였지만 실제 알고리즘을 실행한 결과 특수한 경우에 있어서 해를 찾지 못하는 경우가 발생하였다. 그 이유는 전산 속도를 빠르게 하기 위해 노드 i에서 앞으로 탐색할 노드 j의 집합을 설정하는 단계에서 i와 같은 해당노드의 최소노선비용의 전노드를 j의 집합에서 제외시키기 위해 j_g를 $j_g = p_{ri}^h$, ($C_{ri}^h = \min\{ C_{ri}^1, C_{ri}^2, \cdots C_{ri}^n \}$) [3]인 경우는 j의 집합에 포함시키지 않는다는 조건 때문이었다. 하지만 이 조건은 C_{ri}^2 에 의해 개선 될 수 있는 노선의 추적을 막는 경우를 야기할 수 있었다.

따라서 본 연구에서는 김익기(1998)에 의해 제시된 수정형 덩굴망 알고리즘에서 고려되지 못하였던 최단경로를 탐색할 수 있도록 i 노드에 연결된 j 노드의 집합을 구하는 단계를 다음과 같이 수정하였다. 이를 편의를 위해 본 연구에서 MILVA(Multi-Labeling Vine Algorithm)로 부르기로 한다.

3) 사용된 기호는 김익기(1998)의 논문에서 사용한 기호를 따랐다.

집합 N에서 순서에 따라 노드 i를 선택하고 노드 i 를 집합 N의 목록에서 제거한다.

$J_i = \{j_1, j_2, \ldots\}$ 의 집합을 구한다.

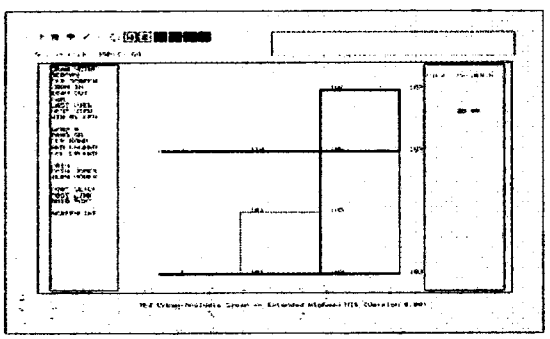
하지만 $C_{ri}^1 = \min\{ C_{ri}^1, C_{ri}^2, \cdots, C_{ri}^n \}$ 인 경우, 앞서 C_{ri}^n 가 수정이 되었다면 j_g를 집합에 포함시키고, 그렇지 않은 경우는 $j_g = p_{ri}^1$ 이면 j_g를 집합에 포함시키지 않는다. 단 j_g-i-j_g 의 U-turn 이 허용될 경우에는 j_g를 집합에 포함시킨다.

만일 집합 N에서 선택할 노드가 없으면 (단계 6)으로 간다.

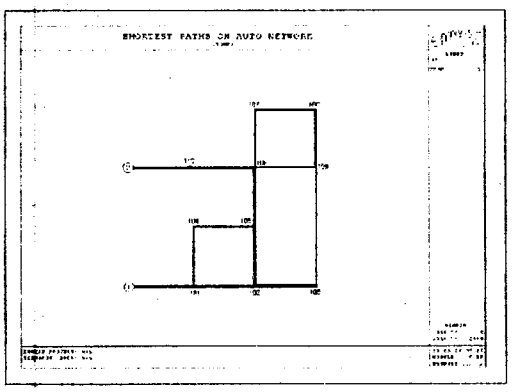
2. 상용프로그램의 최단경로 탐색 알고리즘

현재 국내에서 교통수요 분석에 이용이 되는 상용 프로그램으로는 Tranplan, Emme/2, TransCAD, 사통팔달등이 있는데 이러한 프로그램들은 전통적인 교통수요분석 모형인 4단계 모형에 근간을 두고 있다. 여기서 4단계 모형의 노선배정단계는 Wardrop의 제 1법칙을 만족시키는 사용자 균형상태의 교통패턴을 찾기위해 일반적으로 Frank-Wolf 알고리즘을 사용하는데 이 알고리즘은 All-or-Nothing assignment을 반복하여 수행하게 된다. 이때 All-or-Nothing assignment는 절대적으로 최단경로 탐색알고리즘에 의존하게 되기 때문에 프로그램에 사용된 최단경로 탐색알고리즘에 따라 서로 다른 특성을 나타나게 된다. 프로그램 내에서 최단경로를 찾기 위해 Tranplan의 경우는 덩굴망 알고리즘을 사용하고 있으며 Emme/2는 링크표지 알고리즘을 사용한다. 하지만 불행히도 TransCAD, 사통팔달은 사용된 알고리즘이나 Source Code를 공개하지 않았기 때문에 사용된 최단경로 탐색 알고리즘을 확인할 수는 없다.

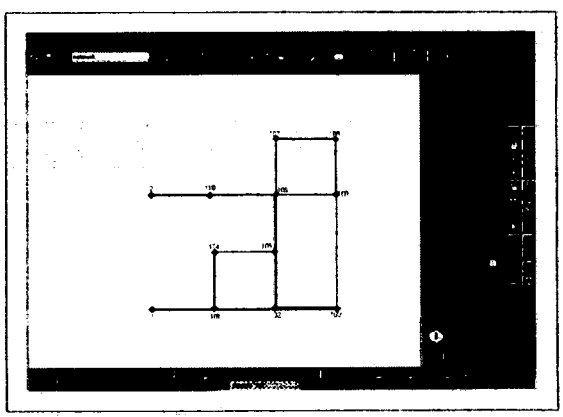
따라서 김익기(1998)가 사용한 U-turn, P-turn등을 포함한 단순 가로망을 이용하여 상용프로그램이 실제 U-turn, P-turn등을 반영한 경로를 찾을 수 있는지 확인하여 보았다. 그 결과 다음의 그림과 같이 네 가지 프로그램 모두가 합리적인 경로를 찾았다.



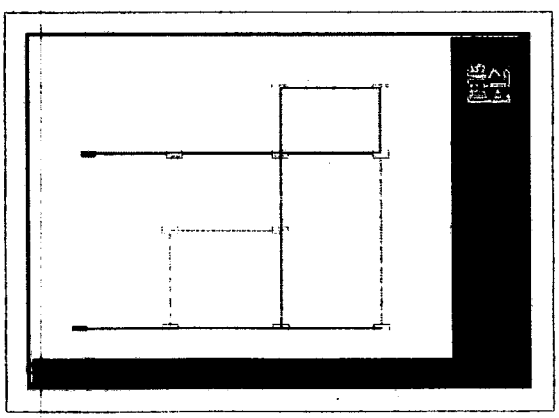
< 그림 3 > Tranplan



< 그림 4 > Emme/2



< 그림 5 > TransCAD



< 그림 6 > 사통팔달

3. 동적 최단경로 탐색 알고리즘

앞선 알고리즘들이 정적인 상황의 통행비용을 고려한 반면 동적 최단경로 탐색 알고리즘은 시간의 변화에 따른 통행비용을 고려하는 것이라 할 수 있다. 즉 앞선 알고리즘을 정적 최단경로 탐색 알고리즘으로 부른다면 이는 통행이 시작될 때의 교통망내의 통행패턴이 통행이 완료될 때까지 유지된다는 가정을 하고 있는 것이다. 그와 반대로 동적 최단경로 탐색 알고리즘은 교통망내의 통행패턴은 시간의 흐름에 따라 변화할 수 있다는 것이다. 동적 최단경로 탐색 알고리즘에서 최단경로 비용은 이전 시간의 최단경로 비용에 영향을 받게 된다. 그 이유는 현재 링크의 통행시간은 현재 링크의 출발노드에 도착하는데 걸린 통행시간에 의해 정의가 되기 때문이다. 따라서 현재 링크의 출발노드에 도착한 시간에 따라 최단경로는 결정이 된다. 이러한 의미에서 본 알고리즘을 Time-dependent shortest path algorithm 이라 부르기도 한다.

동적 최단경로 탐색 알고리즘은 찾고자 하는 경로에 따라 One-To-All 과 All-To-One의 경우가 있다(Ismail Chabini, 1997). One-To-All과 All-To-One의 차이점은 전자의 경우는 출발지점을 정하고 각각의 도착지점에 도착하는 시간을 찾는 것이고, 후자의 경우는 도착지점을 정하고 각각의 출발지점에서 도착지점에 도착하기 위한 출발시간을 찾는다고 볼 수 있다. 즉 One-To-All의 경우는 특정 시간에 출발하여 목적 node에 도착한 시간이 각기 다를 수 있고 , All-To-One의 경우는 출발한 시간이 각기 다르고 동일시간에 목적 node에 도착하게 된다. 이와 같은 특성에 의해 One-To-All의 경우는 사용자의 측면의 운전자 정보 시스템에서 요구하듯이 출발지점에서의 출발시각을 이용해 도착지점까지의 최단 노선을 찾는데 유용하게 사용될 수 있으며 이 경우 최단경로의 통행시간은 다음과 같이 계산될 수 있다.

$$\lambda_j(I_0)^* = \min \{\lambda_i(I_0) + d_{ij}[\lambda_i(I_0)]\}$$
$$= \min \{I_i + d_{ij}(I_i)\}$$

여기서 $\lambda_j(I_0)^*$는 최초의 출발시간 I_0에 출발노드 h에서 j까지의 최단시간을 나타내고, $d_{ij}(I)$는 시간대 I에 노드 i에서 j까지의 통행시간을 나타내며, I_i 는 노드 i에서의 출발시간을 뜻한다. One-To-All의 최단경로 탐색 문제의 경우에는 FIFO조건이 만족이 된다면 Dijkstra's 알고리즘이 Time-dependent fastest path problem을 푸는데 사용이 될 수 있다.
Dreyfus(1969)가 이것에 대해 처음 언급을 했으며 후에 Kaufman and Smith(1993)이 수식으로써 이를 증명하였다.

All-To-One의 경우는 운영자 측면에서 도로의 혼잡을 완화시키기 위한 정책을 결정할 때 사용될 수 있다. 예를 들자면 도로의 혼잡을 완화시킬 목적으로 이용자의 출발시간을 분산시키는 정책을 시행한다고 할 때 All-To-One 알고리즘을 사용하여 혼잡이 가장 심한 시간대에 혼잡이 가장 심한 장소에 도착하기 위한 각각의 출발장소에서의 출발시간대를 찾아 낼 수 있다. 이러한 경우 시간 I에 노드 i에서 노드 q 까지의 최단시간 $\pi_i(I)^*$는 다음과 같이 계산된다.

$$\pi_i(I)^* = \min \{\pi_j[I + d_{ij}(I)] + d_{ij}(I)\}$$

이산시간을 고려한 동적 최단경로 탐색에서는 연속적인 시간을 고려한 경우와는 다르게 도착한 시점과 정보가 제공되는 시점을 일치하지 않는 문제가 발생한다. 따라서 Mahmassani(1993)는 후 방향 표지 탐색 알고리즘(Backward-star labelling algorithm)을 이용하여 All-To-One의 경로를 탐색하는데 있어서 정보가 제공되는 시점과 노드에 도착한 시점을 맞추기 위해 $I_0+k\delta < \tau < I_0+(k+1)\delta$ 인 모든 τ에 대해 다음과 같은 가정을 하고 있다.

$$d_{ij}(\tau) = d_{ij}(I_0+k\delta)$$

위 식은 시간간격 $I_0+k\delta$ 와 $I_0+(k+1)\delta$ 사이에 도착한 차량에 대해 $I_0+k\delta$ 때의 통행시간을 사용한다는 것을 의미한다.
전 방향 표지 탐색 알고리즘(Forward-star labelling algorithm)을 이용하여 One-To-All의 경로를

탐색하는데 있어서도 후 방향 표지 탐색 알고리즘에서와 같은 가정이 필요로 한다.

이와 같은 이산적 시간을 고려한 동적 최단 경로 탐색 알고리즘에서는 각 시간대별로 논리적 모순이 없는지를 확인해야 한다. 다시 말하면 임의의 한 link에서 연속한 두 개의 시간대에서 앞선 시간대의 통행시간 보다 뒤따르는 시간대의 통행시간이 빠를 경우에는 뒤늦게 출발한 차량이 앞차를 추월하는 경우가 발생하므로 이를 주의 깊게 검증해야 한다. 즉 link에 먼저 진입한 차량이 link를 먼저 빠져나가야 한다는 FIFO (First-In-First-Out) 조건을 검증해야 한다. 따라서 같은 link를 이용하는 연속한 임의의 시간 t와 t+Δt에 대해 아래의 식이 성립되어야 한다. (Bin Ran and David Boyce, 1996)

$$t+d_{ij}(t) \leq (t+\Delta t)+d_{ij}(t+\Delta t)$$

위 식은 정리하여 다음과 같이 표현할 수 있다.

$$1+\frac{d_{ij}(t+\Delta t)-d_{ij}(t)}{\Delta t} \geq 0$$

위 식을 다시 정리하여 $\Delta t \rightarrow 0$이면

$$1+d_{ij}'(t) \geq 0$$

즉,

$$d_{ij}'(t) \geq -1$$

이것은 시간의 변화에 대한 통행시간의 감소가 1 보다 적다면 FIFO 조건을 만족시킨다는 것을 의미한다.

III. 이산 시간간격의 다수의 표지를 갖는 덩굴망 알고리즘

DRGS에서 통행자에 대해 노선정보 안내 및 유도를 위해 사용될 최단경로 탐색 알고리즘으로 다음과 같은 이산시간간격을 고려한 최단경로 탐색 알고리즘을 제안한다.

이 알고리즘의 기본가정은 최초의 출발노드와 출발시간은 주어졌으며 link의 통행시간에 대한 정보는 최초의 출발시간에 예측되어진 값을 사용하며 link의 통행시간은 진입한 순간의 예측통행시간이 적용이 되어 link를 빠져나갈 때까지 유지된다고 가정한다.

예측된 link의 통행량은 최대 link 교통량 제약조건과 최대 유출교통량 제약조건을 만족하며 예측된 link의 통행량으로 얻어진 통행시간은 FIFO조건을 만족한다고 가정한다. link의 통행시간에 관한 정보는 일정한 시간간격을 두고 얻을 수 있다고 가정한다.

또한 $I+k\delta < \tau < I+(k+1)\delta$ 인 모든 τ에 대해 다음과 같은 가정을 한다.

$$d_{ij}(\tau)=d_{ij}(I+k\delta)$$

알고리즘은 시간대별 변화를 반영한 One-To-All의 경로를 탐색하기 위해 정적인 상태의 전 방향 표지 탐색 알고리즘(Forward-star labelling algorithm)에 시간 변수를 추가하여 사용하였고 여기에 도시부의 다양한 회전형태를 고려하기 위해 MILVA의 표지기법을 이용하였다.

다음의 기호를 통해 알고리즘을 살펴보면,

R : 출발노드.

D : 도착노드

N : 임의의 노드.

T_0 : 최초의 출발시각

T : 임의의 출발시각.

T*: $T^* = \{T_0, T_0 \leq T_1 \leq T_0+\delta, T_0+\delta \leq T_2 \leq T_0+2\delta, \cdots\cdots, T_0+(M-1)\delta \leq T_M \leq T_0+M\delta\}$

C : 임의의 노드 N에 도달하는 방향을 나타내기 위한 코드로써 노드 N에 하나의 링크로 연결된 노드의 전노드 코드.

Label(N, T_0, C) : T_0시간에 출발노드 R에서 전노드 코드인 C인 전노드를 거쳐 노드 N까지 이르는 최소통

행비용.

P(N, T_0, C) : 출발노드 R에서 노드 N에 이르는 경로로써 전노드 코드 C에 해당하는 전노드의 번호. (Predecessor 혹은 Back node)

PP(N, T_0, C) : 출발노드 R에서 노드 N에 이르는데 전노드 코드 C에 해당하는 전노드의 전노드, 즉 노드 N의 전노드 코드 C에 해당되는 전전노드의 번호. (Back back node)

Time(T, N, M) : T시각의 N노드에서 M노드까지의 통행시간.

Pen(T, L, N, M): T시각의 L노드에서 N노드를 거쳐 M노드까지의 회전 지체 시간.

MTime(N, T_0) : T_0시각의 노드 N까지의 최소통행시간.

(단계 1) 모든 노드에 대해 비용, 전노드, 전전노드의 표지를 초기화한다.

(1) Label(N, T_0, C)=∞ , P(N, T_0, C)=0 , PP(N, T_0, C)=0 ($\forall$ N, C)

(2) Label(R, T_0, 1)=0

(단계 2) R과 하나의 링크로 연결된 노드에 대해 비용, 전노드의 표지를 갱신한다.

(1) 비용, 전노드의 표지를 갱신한다.

Label(Nextnode, T_0, 1) = Time(T_0, R, Nextnode) + T_0

P(Nextnode, T_0, 1)=R , PP(Nextnode, T_0, 1)=0 ,

(2) R을 집합 N의 목록에 기록하여 둔다. 그리고 Cnode=R로 놓는다.

(단계 3) 집합 N에서 순서에 따라 노드 Cnode를 선택하고 노드 Cnode 를 집합 N의 목록에서 제거한다.

(1) 다음의 조건을 만족하는 J_i = {$Nextnode_1$, $Nextnode_2$, . . } 의 집합을 구한다.

(a) Cnode에서 Nextnode로의 통행이 제한되지 않은 경우.

(b) Label(Cnode, T_0, 1)=min{Label(Cnode, T_0,1), Label(Cnode, T_0,2)}일때 앞서 Label(Cnode,T_0,2)가 수정이 된 경우.

(c) (b)의 조건을 만족시키지 못한 경우 Nextnode≠P(Cnode,T_0,1)인경우. 단, Nextnode-Cnode-Nextnode에 U-turn이 허용된 경우는 제외.

(2) 집합 N에서 선택할 노드가 없으면 (단계 8) 로 간다.

(단계 4) 집합 J_i에서 순서에 따라 노드 Nextnode를 선택하고 집합 J_i의 목록에서 Nextnode를 제거한다.

K_j = { N_Next_1 ,N_Next_2 ..... } N_Next ≠Cnode 인 집합을 구한다.

단, Cnode-Nextnode-Cnode에 U-turn이 허용이 될 경우는 N_Next를 K_j에 포함시킨다.

하지만 U-turn 이 허용되지 않거나 N_Next_g = P(Nextnode, T_0 ,1),

(Label(Nextnode,T_0,1)=min{Label(Nextnode,T_0,1), Label(Nextnode,T_0,2)})인 경우에는

N_Next_g를 K_j에 포함시키지 않는다. 만일 선택할 N_Next가 없으면 새로운 Nextnode를 선택한다. 선택할 Nextnode가 없으면 (단계 3)으로 간다.

(단계 5) Nextnode까지의 최소비용을 계산한다.

(1) Nextnode의 전노드가 Cnode인 C를 찾아 Nextnode까지의 최소비용을 계산한다.

MTime(Nextnode, T_0)=Label(Nextnode,T_0,C)

(2) Nextnode의 전노드가 Cnode인 C가 없다면 Cnode의 Label중 작은쪽을 선택해 아래와 같이 MTime을 계산한다.

MTime(Nextnode,T_0)=Label(Cnode,T_0,C)+Time(T_1,Cnode,Nextnode)+Pen(T_1,P1,Cnode,Nextnode)

(P1=P(Cnode,T_0,C))

(T_1≤Label(Cnode,T_0,C), T_1 =max {T^*})

(단계 6) N_Next까지의 최소통행비용을 계산한다.

MTime(N_Next,T_0)= MTime(Nextnode,T_0)+Time(T_2,Nextnode,N_Next)+Pen(T_2,Cnode,Nextnode,N_Next)

(T_2≤MTime(Nextnode, T_0), T_2 =max {T^*})

(단계 7) N_Next의 표지를 갱신한다.

(1) N_Next의 Label중 전노드를 Nextnode로 하는 C를 찾는다.

만약 전노드를 Nextnode로 하는 Code가 없으면 C=1로 한다.

(2) MTime(N_Next,T_0)이 Label(N_Next, T_0, C)보다 작으면

Label(N_Next, T_0, C) = MTime(N_Next,T_0), P(N_Next, T_0, C)=Cnode ,

PP(N_Next, T_0, C)=Nextnode 로 갱신한다.

(3) MTime(N_Next,T_0)이 Label(N_Next, T_0, C)보다 크다면 (단계 4)로 간다.

(단계 8) 노드표지를 이용하여 최단경로를 역으로 추적한다.

(1) Label(D, T_0, C)=min{Label(D, T_0, 1),Label(D, T_0, 2)}인 전노드 코드 C를 찾아

이때의 D노드의 전노드와 전전노드를 기억한다.

K=P(D, T_0, C), J=PP(D, T_0, C)

(2) J, K, D를 순서대로 노선노드 집합 R의 목록에 추가한다.

(3) 노드K의 전노드가 J인 C를 찾아 K의 전전노드 I=PP(K,T_0,C)를 추적한다.

(4) I노드를 노선노드집합 R의 목록의 첫 번째 위치에 추가한다.

(5) 만일 I=R 이면 최단경로 추적을 종료한다.

만일 아니면 K=J, J=I로 놓고 (3)으로 가서 반복한다.

알고리즘은 (단계 1)과 (단계 2)에서 표지를 초기화하고 전전 노드를 고려하기 위해 출발노드에 하나의 링크로 연결된 노드에 대해 표지를 설정하게 된다.

(단계 3)에서 (단계 6)까지는 전전 노드를 고려하여 노드의 표지를 갱신하는 과정이다. (단계 3)과 (단계 4)에서는 현재의 노드(Cnode)와 연결된 통행 가능한 노드(Nextnode)와 Nextnode와 연결된 통행 가능한 노드(N_next)를 찾는다. (단계 5)와 (단계 6)를 통해 Cnode를 전노드로 갖는 Nextnode의 C를 이용해 N_next까지의 최소통행비용(MTime(N_Next,T_0))을 계산한다. (단계 7)에서는 (단계 6)에서 구한 MTime(N_Next,T_0)와 N_Next의 표지(Label(N_Next,T_0,C))와 비교해 MTime(N_Next,T_0)이 Label(N_Next,T_0,C)보다 적을 경우 N_Next의 표지를 갱신하고 그렇지 않을 경우 (단계 4)로 돌아간다.

위의 알고리즘이 정적인 MILVA와 다른 점은 시간 T를 고려하여 최단경로를 탐색한다는 것이다. 이때 출발노드에서 다음 노드로의 통행시간을 선택함에 있어서 출발노드에서의 출발시간을 고려하여 통행시간이 적용될 시간간격을 찾는다. 즉, Cnode - Nextnode의 통행시간을 선택할 때 Cnode에서 출발하는 시각인 Label(Cnode,T_0,C)를 이용하여 Cnode - Nextnode의 통행시간이 적용될 시간간격(T_1)을 찾고 Nextnode - N_Next의 통행시간을 선택할 때 Nextnode에서 출발하는 시각인 Mtime(Nextnode,T_0)를 이용하여 Nextnode-N_Next의 통행시간이 적용될 시간간격(T_2)을 찾는다. (단계 8)의 최단경로를 역 추적하는 과정은 정적인 MILVA와 동일하다.

Ⅳ. 단순 교통망에 의한 알고리즘의 적용

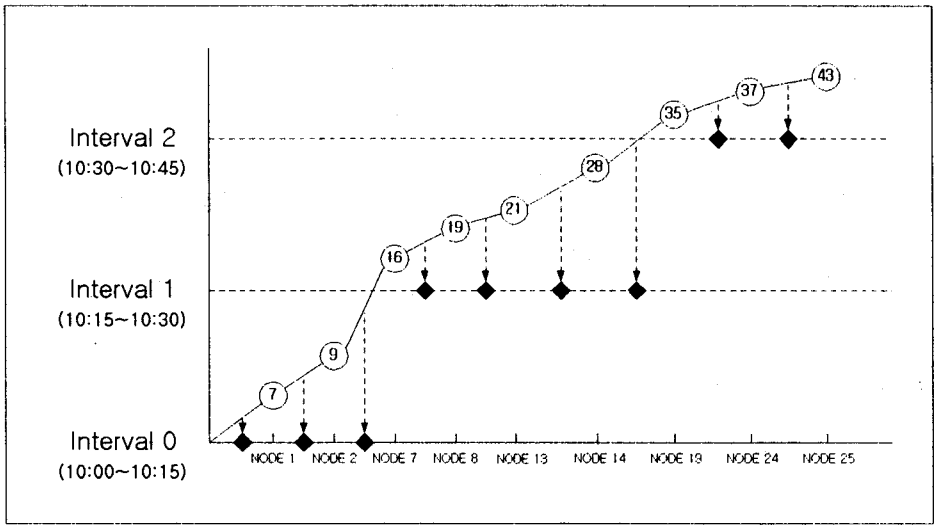
다수의 표지를 갖는 동적인 덩굴망 알고리즘이 최단경로를 어떻게 찾아가는지 그 과정을 살펴보기 위해 단순화된 가로망에 알고리즘을 적용하여 특정 두 노드사이의 최단경로를 탐색해보기로 한다. 알고리즘을 적용하기 위한 단순 교통망은 25개의 노드를 갖는 정방형 격자형 가로망으로 모든 링크는 양방향 통행이 가능하다.(25개의 노드, 200개의 링크) 통행시간은 무작위로 발생시켰으며 시간간격은 15분으로 하고, 경로4-5-10과 9-10-15는 통행이 금지 되어있다(<그림 8> 참조). 그 결과는 다음의 <표 1>와 같다. <그림 7>은 알고리즘이 경로를 찾아가는 과정을 나타내는데 여기서 ◆는 노드의 검색이 일어나게 되는 시점을 나타내며 ⑦,⑨등은 해당 노드까지의 통행시간을 나타낸다. 이때 노드 7번까지는 10:00~10:15분의 시간간격에서 노드의 검색이 일어나며, 노드 19번까지는 10:15~10:30분의 시간간격에서 노드 25번까지는 10:30~10:45분의 시간간격에서 노드의 검색이 일어나게 된다.

최종적으로 이산시간간격을 고려한 동적 최단경로는 경로비용이 43분인 1-2-7-8-13-14-19-24-25를 찾았다. 이러한 경로는 적용하는 시간간격에 따라 변화할 수 있다. 따라서 좀더 세부적인 변화

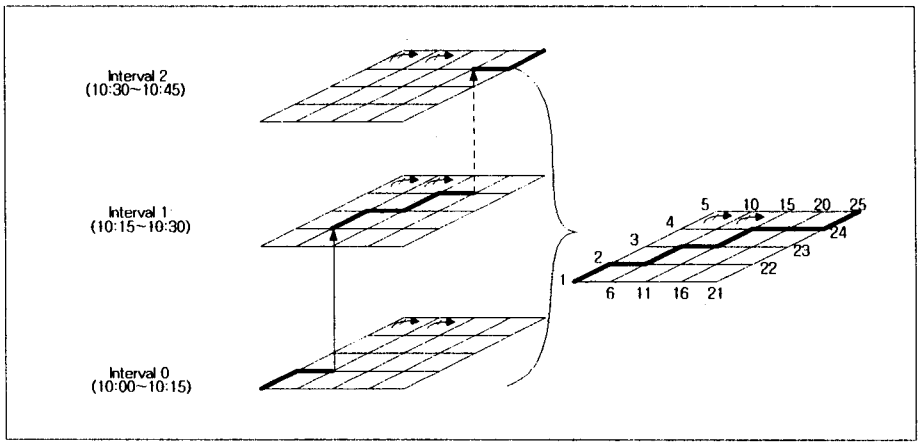
를 반영하기를 원한다면 시간간격을 조절함으로써 해결할 수 있다.

<표 1> 알고리즘의 탐색 결과

```
 <TMILVA> 15 MINUTE INTERVAL CASE
YOU WANT TO SHORTEST PATH FROM 1 TO 25
RESULT AS FOLLOWING
   1    2    7    8   13   14   19   24   25
AND REQUIRED TRAVEL TIME IS    43
```



<그림 7> TMILVA에서 통행시간을 적용하는 시점



<그림 8> TMILVA의 경로탐색과정 과 결과

V. 결론 및 추후 연구방향

기존에 건설된 교통시설의 운영효율을 극대화하기 위한 지능형 교통체계의 한 분야인 ATIS는 도로를 이용하는 통행자에게 편리성을 제공하는 동시에 도로를 효율적으로 운영할 수 있는 정보체계이다. ATIS 체계 하에서 통행자에게 신뢰성 있는 정보를 제공하기 위해서는 교차로에서의 회전에 의한 지체를 정확하게 반영함은 물론이고 실시간으로 변화하는 교통상황을 반영할 수 있는 동적인 최단경로 탐색 알고리즘이 요구된다. 하지만 기존에 발표된 동적인 최단경로 탐색 알고리즘은 회전에 대한 정보를 반영하지 못하며 정적인 최단경로 탐색알고리즘 조차 회전에 대한 정보를 정확히 반영하지 못한다. 링크 탐색 알고리즘이 회전에 대한 정보를 정확히 반영하기는

하지만 이 알고리즘은 네트워크를 변형시켜 기존의 수형망 알고리즘을 사용하는 것과 크게 다르지 않기 때문에 네트워크를 일단 변형시킨 후 수형망 알고리즘을 사용하는 경우와 경로를 탐색할 때마다 알고리즘 내부에서 네트워크를 변형시키는 과정을 되풀이하는 링크탐색 알고리즘을 사용하는 경우에 대해 어느 것이 더 효율적인지 판단을 필요로 한다.

본 연구에서는 이러한 이유에서 알고리즘 내부에서 회전을 반영하기 위해 수정형 덩굴망 알고리즘의 표지기법을 이용하여 동적인 최단경로 탐색알고리즘을 개발하였다. 본 연구에서 개발한 동적인 최단경로 탐색 알고리즘은 정적인 상태의 수정형 덩굴망 알고리즘에 시간에 따라 변화하는 교통상황을 반영하기 위해 시간에 대한 변수를 추가하였다. 알고리즘은 시간대별로 변화하는 통행시간을 고려하여 최단 경로를 탐색하게 되며 이렇게 해서 탐색한 경로에 대한 표지를 출발시점을 기준으로 설정하도록 하여 모든 앞선 시간에 대해 경로를 고려하도록 하였다. 매 단계에서 전 노드를 추적하여 회전에 관한 정보를 반영하도록 하였다.

따라서 본 연구에서 개발한 최단경로 탐색 알고리즘은 교차로에서의 회전에 대한 정보와 통행금지 등을 정확히 반영하며 실시간으로 변화하는 통행시간을 반영함으로써 신뢰성 있는 노선정보를 ATIS를 이용하는 통행자들에게 제공하는데 활용될 수 있는 기법이다.

추후로 수정형 덩굴망 알고리즘의 표지기법을 이용하여 차기 최단경로를 탐색할수 있는 알고리즘을 개발하여 차별화 된 정보를 ATIS 이용자에게 제공하는 연구가 계속 될 수 있으리라 기대한다. 이러한 차기 최단 경로에 대한 정보의 제공이 요구되는 것은 모든 이용자에게 획일적인 최단경로만을 제공할 때 발생할 수 있는 교통망의 혼잡을 완화시킬 수 있기 때문이다. 수정형 덩굴망 알고리즘에서 표지를 설정함에 있어서 표지의 크기 순서로 설정을 한다면 사지 교차로에서 하나의 노드로 들어오는 최대 4개의 경로를 구할 수 있으리라 생각된다.

Ⅵ. 참고 문헌

1. 강맹규, 네트워크와 알고리즘, 박영사, 1991
2. 김익기, "ATIS를 위한 수정형 덩굴망 최단경로 탐색알고리즘의 개발", 대한교통학회 제 16권 제 2호, 1998.
3. 최기주, "U-Turn을 포함한 가로망 표현 및 최단경로의 구현", 대한교통학회 제 13권 제 3호, 1995
4. Ahuja, R. K., Magnanti, T. L., and Orlin, J. B., Network Flows : Theory, Algorithm, and Applications, Prentice-Hall, Inc., 1993
5. FHWA, Traffic Assignment : Methods Applications, Products, U.S. Department of Transportation, August, 1973.
6. Sheffi, Y., Urban Transportation Networks : Equilibrium Analysis with Mathematical Programming Methods, Prentice-Hall, Inc., 1985
7. Thomas, R., Traffic Assignment Techniques, Avebury Technical, 1991
8. S. E. Dreyfus. An Appraisal of Some Shortest-Path Algorithms, Operation Research, Vol.17, 1969
9. A. K. Ziliaskopoulos and H. S. Mahmassani, Time-Dependent, Shortest-Path Algorithm for Real-Time Intelligent Vehicle Highway System Applications, TRR 1408, 1993
10. Ismail Chabini, A New Algorithm for Shortest Paths In Discrete Dynamic Networks, IFAC Transportation Systems, 1997
11. Bin Ran and David Boyce, Modeling Dynamic Transportaions Network - An Intelligent Transportation System Oriented Approach, Springer, 1996
12. Gallo G., S. Pallotino, Shortest Path Methods in Transportation Models, Transportation Planning Models, 1984