산업기술연구(강원대학교 산업기술연구소 논문집), 제21권 A호, 2001.
Journal of Industrial Technology, Kangwon Nat'l Univ., Korea, No. 21 A, 2001.

교차로에서의 좌회전 금지, U-turn, P-turn을 고려한 개선된 Dijkstra Algorithm에 관한 연구

A Study on Dijkstra Algorithm in Crossroad Including Left-turn Restriction, U-turn, and P-turn

김 성 수* 전 영 주** 차 영 민***
Kim, Sung-Soo Jun, Young-Joo Cha, Young-Min

Abstract

U-turn and P-turn as well as left-turn restriction exist in real traffic network. the optimal route should be selected for considering these using shortest path algorithms. But, the traditional algorithms have some limitations to use for considering there.

The objective of this paper is to modify Dijkstra algorithm in order to find the optimal path in real traffic network. The continuous three nodes are used to check turn-restrictions and exclude these from and optimal path. A virtual connection is used to consider U-turn and P-turn

키워드 : *U-턴, P-턴, 교통네트워크, Dijkstra 알고리즘*
Keywords : *U-turn, P-turn, traffic network, Dijkstra Algorithm*

1. 연구의 목적 및 배경

최근 네트워크 관련 논문들이 하루가 다르게 발표되고 있다. 그 중에서 네트워크 이론을 이용한 교통망 최적화 관련 논문을 보면, 최적경로의 선택 문제가 큰 비중을 차지하고 있다. 그리고, 이를 현실적으로 적용하기 위해서 회전금지와 U-turn 및 P-turn을 적용한 문제를 해결하기 위한 노력이 계속되고 있다.

그러나, 이러한 문제들을 해결하기 위한 대부분의 논문에서 제시된 최적경로 알고리즘들은 많은 계산량과 시간이 필요하다는 것을 알 수 있다.

본 논문에서는 실제 교통 네트워크의 상황하에서 최적경로 문제를 최소의 계산량과 시간을 필요로 하는 간단한 구조의 알고리즘을 연구하고, 실제 사용자들이 사용 가능한 최적 경로 안내 시스템의 프로토타입을 설계, 구축하였다.

또한, 기존에 제시된 알고리즘에 대하여 고찰해 보고[1][7][8], Dijkstra 알고리즘에 대한 이해와 회전금지, U-turn 및 P-turn을 고려한 개선된 알고리즘을 설명하였다. 개선된 알고리즘을 Visual C++을 사용하여 최적경로 안내 시스템을 구현하고, 가상 교통 네트워크 문제에 적용하여, 알고리즘의 적합성을 검증하였다.

* 강원대학교 산업공학과 조교수, 공학박사
** 주식회사 파세코, 경영혁신 추진실
*** 강원대학교 대학원 산업공학과 석사과정

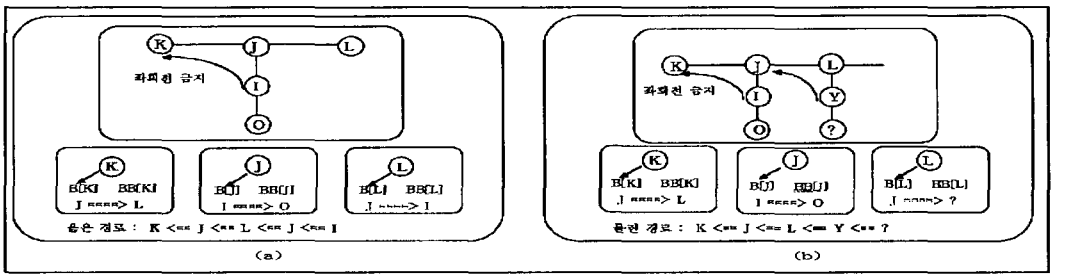


그림1 각 노드의 전번 노드와 전전번 노드

2. 기존에 제시된 알고리즘에 대한 고찰

본 장에서는 교통 네트워크 관련 논문 중, 회전 금지, U-turn 및 P-turn을 고려한 알고리즘의 연구를 간략히 소개한다.

첫 번째, U-turn을 포함한 가로망 표현 및 최단경로의 구현[5]이다. 회전금지를 표현하기 위해, <그림 1>(a) 와 같이 각 노드의 전번 노드와 전전번 노드를 기억해야만 하고, 이에 따르는 저장 공간을 노드 수 만큼 할당하여 알고리즘의 자료로 사용하는 방법을 제시하였다.

이 방법의 단점은 <그림 1> (b) 와 같이 2개의 회전금지 제약이 연속해서 있는 경우에는 전노드, 전전노드 뿐만아니라 전전전노드도 기억하고 있어야만 회전금지를 고려한 최적경로를 찾아낼 수 있다. 이와같이 회전금지가 연속해서 발생할 경우 기존의 알고리즘에 한계가 발생한다는 것이다. 즉, J노드 및 L 노드에서 회전금지 제약이 존재하므로 J노드로부터 K노드에 도달하는데 최적경로의 틀린 정보를 가질 수 있다. 이 문제의 해결을 위해서는 전번 노드의 개념을 확대하여 해결할 수는 있으나 알고리즘의 구성이 어렵고 회전 금지 구간의 갯수가 많아짐에 따라 필요한 기억용량과 계산량이 큰 장애 요소로 작용하게 된다. 또한, U-turn을 나타낼 수 있는 두 가지 방법을 제안하여 비교 분석하였다.

두 번째, 교차로 제약과 지연이 있는 네트워크에서 최단경로탐색[4]이다.

교통네트워크에서 교차로 제약이 주어지면 기존의 Dijkstra 알고리즘 등을 적용하여 최단경로를 구할 수 없다. 또한 가장 빠른 경로에 노드의 반복이 발생할 수도 있다. 따라서, 본래의 교통 네트워크를 변형하여 회전금지 제약을 기존의 Dijkstra 알고리즘에 적용시킬수 있도록 하는 방법을 사용하였다. 이것은 <그림 2>에 나타나 있듯이 본래의 네트워크를 Mid-Block 기법을 이용하여 네트워크를 재구성함으로써 교차로 제약이 암묵적으로 고려된 변형된 네트워크로, <그림 2>(a)의 각호가 <그림 2>(b)의 하나의 노드에 해당된다. <그림 2>(a)에서 <그림 2>(b)로 변형하면서 교차로 지연시간 또는 회전금지(지연시간= ∞)요소를 고려해 줄 수 있다.

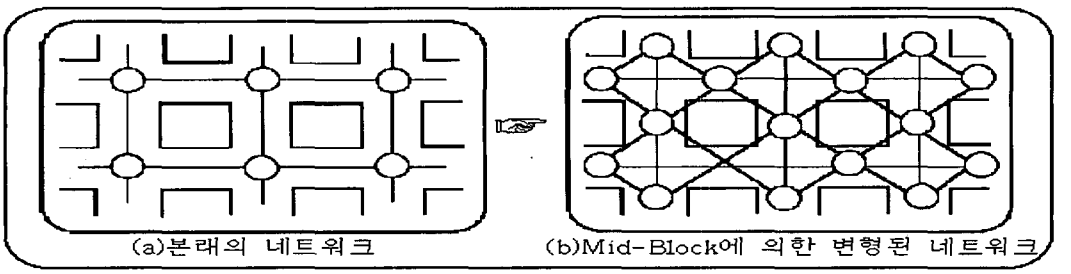


그림2 네트워크의 변형

이 방법은 보다 정밀한 네트워크의 방향성을 나타낼 수 있는 장점이 있는 반면, 노드와 호의 수가 불필요하게 증가하게 되어 계산량의 급증을 야기시키게 된다. 또한, 알고리즘을 실행시킨 후, 기존의 네트워크 형식에서의 노드(즉, 경로의 흐름)로의 변환의 작업이 필요하게 되는 불편함이 있다.

세 번째, 첨단 물류교통 정보 시스템 구현을 위한 최적경로 알고리즘에 관한 연구[3]이다. 이 논문은 다른 여타의 최적경로에 대한 논문과는 달리 그 기초가 되는 알고리즘으로 Floyd-Warshall 알고리즘을 사용하였다.

이 논문에서는 교차로에서의 회전금지 제약과 U-turn과 P-turn에 관한 제약을 가상의 노드를 추가함으로써 해결하였는데, 이에 대한 논문의 요지는 아래와 같다.

<그림 3>에서 [K => J => L]이 회전금지 경로이다. 이와 같이 좌회전금지를 고려해 주기 위해, 좌회전금지의 K → J 호가 없는 경우와 좌회전금지의 J → L 호가 없는 경우로 설정하여, 각각 Floyd-Warshall 알고리즘을 시행한다. 그 결과 더 작은 값을 갖는 경우의 경로를 최적경로로 선택하게 된다. 또한, 회전금지 교차로 내에서의 U-turn과 P-turn의 적용은 <그림 3>에서와 같이 J노드에 대해서 J, J'라는 두 개의 노드로 변형시키면 간단히 Floyd-Warshall 알고리즘에 적용이 가능해진다. 즉, 변형된 네트워크의 좌회전금지 제약 하에서 U-turn과 P-turn을 적용하기 위해서, 회전금지 교차로 노드(J)에 가상의 노드(J')하나를 추가하였다. J'노드는 회전금지 경로의 출발노드인 K노드와 연결된 호가 없을 뿐 J노드와 같은 형태의 노드이다.

그러나 회전금지 제약의 수가 증가함에 따라 2^n배(n: 네트워크의 회전 금지 개수)로 그 계산량이 증가하게 된다. 또, U-turn과 P-turn을 교차로에 적용함에 있어서, 적용의 가부는 네트워크를 더 복잡하고 알고리즘의 설계에 어려움을 준다는 단점이 있다.

3. 실제 교통 네트워크 적용을 위한 Dijkstra 알고리즘의 개선 방안

지금까지의 회전금지 제약과 U-turn 및 P-turn의 적용 알고리즘은 그 계산량과 기억용량면에 있어서 효율성이 많이 떨어지는 단점이 있어 활용의 단계에는 많은 어려움이 있다. 따라서, 본 논문에서는 지금까지의 단점을 개선하기 위한 방법에 중점을 두어 개발되었다.

개선된 알고리즘의 설계에 앞서서 우선, Dijkstra 알고리즘에 대하여 개괄적으로 알아보고자 한다. Dijkstra 알고리즘은 호의 길이가 양수일 때만 알고리즘의 실행이 가능하고, 기본적으로 최적경로의 부분경로 또한 최적경로라는 최적원리를 따른다. (즉, a→b→c→d→e→f 경로가 최적경로라면, 그 부분경로인 c→d→e 경로 또한 최적경로라는 개념.)

이 알고리즘을 간단히 설명하면, 출발노드인 노드1에서 각 노드까지의 거리를 그 노드의 임시표지로하고, 모든 임시 표지 중에서 가장 작은 값을 갖는 임시표지 노드i를 구하여 영구표지화 한다. 노드i에 영구표지가 기록되면, 노드i의 인접노드 j의 임시표지를 검사하여, 영구표지된 값과 영구표지된 노드i로부터 노드j까지의 거리를 합산하여 현재의 임시표지와 비교한다. 만약, 비교한 값이 작다면 현재의 임시표지를 작은 값으로 대치한다. 반대로, 작지 않으면 임시표지를 그대로 둔다.

모든 임시표지 중에서 가장 작은 값을 영구표지

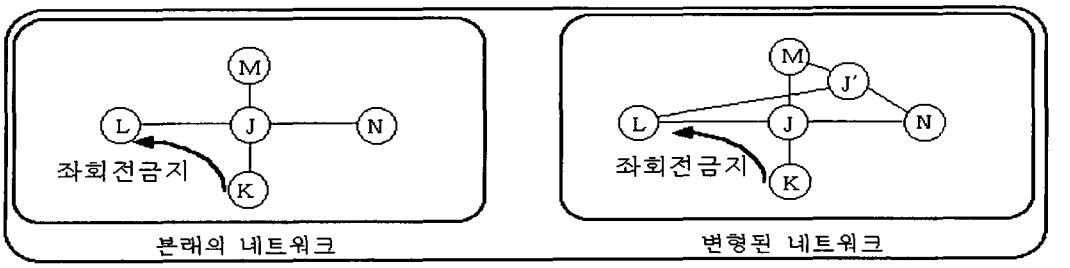


그림3 새로운 Floyd-Warshall 알고리즘에 적용시키기 위한 네트워크 변형

로 선택하고 임시표지의 개선과 영구표지의 선택을 반복한다.

다음은 회전금지를 고려하기 위한 기존 Dijkstra 알고리즘의 주요 개선 내용이다.

a. 각 노드의 이전노드(변수"b"로 표시함)를 하나의 자료로 처리한다.

b. 각 노드의 임시표지를 개선할 때 마다 이전 노드를 개선하는 방법을 적용한다.

c. 비교연산에 의한 차후의 노드가 회전금지 경로인 경우를 최적화 연산의 수행대상에서 제외한다.

d. 회전금지 제약의 수가 많아짐에 따라 상당한 계산량을 요구하게 되고, 알고리즘의 구현이 복잡해지는 것이 보통이지만, 본 논문에서 제시하는 개선된 Dijkstra 알고리즘은 계산량이 상대적으로 적고, 시스템 구현이 다른 알고리즘과 비교하여 상대적으로 용이하다.

본 논문에서 제시하고자 하는 알고리즘은 회전금지, U-turn, P-turn에 만족하는 실제 교통 네트워크에 적용하고자 하는 목적을 가지므로, 알고리즘의 구현을 실제 교통상황을 고려하여 실시하였다.

개선된 Dijkstra 알고리즘을 설명하기 위해, <그림 4> 교통 네트워크를 사용한다.

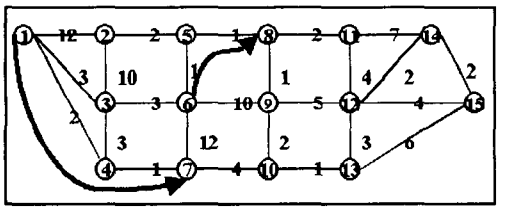


그림4 교통 네트워크사례. (굵은 실선은 회전금지 경로 : 6-5-8, 1-4-7)

3.1. 회전금지 제약의 적용

앞에서 설명을 했듯이, 변수 "b"로 각 노드의 전번 노드값을 기억하여, 노드 3개로 연결되는 정보(전번노드→현재노드→차후노드)를 이미 주어진 좌회전 금지 경로 정보와 비교하여 일치하면, 최적경로를 찾는 과정에서 배제 시킨다.

쉬운 예로 <그림 4>에서 노드1을 출발해서 노드11에 도착하기 위한 최적경로를 찾는다고 할 때, 회전 제약이 없다면 최적경로는 [1→3→6→5→8→11]이 될 것이고, 최적거리는 10이 된다. 그러나 6→5→8 경로가 회전금지 제약이 있다면, 그 결과는 앞의 값과 달라지게 된다.

그 결과, 최적경로는 [1→4→7→10→9→8→11]이고, 최적거리는 12가 된다. 만약 6→5→8 경로와 1→4→7 경로가 동시에 회전 금지 제약이 된다면, 최적경로는 [1→3→4→7→10→9→8→11]이 되고, 최적거리는 16이 됨을 알 수 있다.

이와 같이 개선된 알고리즘의 장점은 회전 금지 제약의 추가에 따라 비교 연산(<표 1>의 단계2 에서, b(전번노드)→i(현재노드)→j(차후노드) 경로 = 회전금지경로)만을 추가시키면 되므로 시스템에 적용이 용이하게 된다. 또한, 영구표지된 노드의 전번노드(변수 "b")를 이용함으로써 쉽게 경로를 구할 수 있다.

3.2. U-turn의 적용

교통 네트워크에서 U-turn문제를 고려하는 것은 그리 어려운 문제가 되지 못한다. 이를 나타내기 위한 방법은 여러가지가 있겠지만, 본 논문에서는 하나의 U-turn을 나타내기 위하여 하나의 가상의 호를 추가해 주는 방법을 사용하였다. 즉, 네트워크의 형태를 변형하는 방법인데, 이것은 Mid-Block 법과는 달리, 엄청난 양의 수정이 필요한 것도 아니고, 메모리나 연산량에 있어서도 부담이 되지 않는 방법이다.

<그림 5> 의 네트워크에서 우회전 금지 경로 6→5→8 이 포함된 교차로에서 노드6 에서 노드8로 가기 위해서, 6→5→2→5→8 경로에서의 U-turn을 감안한다고 가정해 보자.

6→5→2→5→8 경로의 U-turn을 노드5의 중복을 피하면서 네트워크 상에 표현하는 방법은 <그림 5>에서 나타난 것과 같이 기존의 네트워크에서 노드2 로부터 노드8 로의 아크를 단 방향으로 한 개를 추가하고 그 거리를 " $d_{2,8}= d_{2,5}+ d_{5,8}$ =2+1=3 " 로 표현하게 된다.

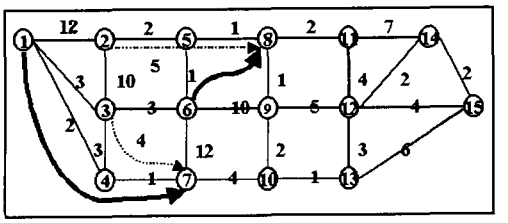


그림 5 U-turn을 적용한 네트워크,(점선:추가된 호, 굵은 실선 : 회전금지)

만약, 노드2에서 노드8의 호가 실제네트워크에 이미 존재한다면, 다음을 고려해야 한다. 만약, 가상의 호 2→8의 거리(3)가 실제의 호 2→8보다 작다면, 가상의 호값을 적용한다. 그러나 만약, 가상의 호 2→8의 거리(3)가 실제의 호 2→8보다 크다면, 가상의 호 값을 적용하지 않는다.

이 방법을 사용할 때, 최적경로의 출력에 있어서, 노드2에서 노드8로의 가상의 호 값이 사용된 최적경로의 부분경로가 주어졌다면, 경로의 해석은 2→5→8 경로를 지나는 것으로 해석하여야 한다는 것이다. (이 부분에 대한 알고리즘의 구현은 다음의 P-turn의 적용에서 설명하기로 한다.)

3.3. P-turn의 적용

실제의 교통 상황을 보면, 교통 사고, 도로 공사에 의한 도로의 봉쇄, 교통량의 급증으로인한 지연 등을 제외하고는 P-turn을 허용하지 않는 도로는 없다. 그리고, 회전 금지 교차로 이외에서의 P-turn은 최적경로의 도출과는 무관하므로 회전 금지 교차로에서의 P-turn만을 고려하기로 한다. 회전 금지 교차로에서 P-turn과 U-turn이 동시에 가능하다면 앞의 U-turn의 적용에서와 같은 방법으로 P-turn과 U-turn이 동시에 해결된다.

그러나 U-turn은 불가능하고 P-turn만이 가능한 교차로라면 상황은 달라지게 된다. 이 부분에 대한 해결방법은 아주 복잡한 알고리즘을 요하게 되므로, 본 논문에서는 알고리즘의 구현을 생략하고, P-turn과 U-turn이 동시에 가능한 경우 만을 고려하였다.

위의 <그림6>(a)에서 H→E→D 경로가 회전 금지 경로이다. 이 경우, H→E→F→E→D 경로를 U-turn경로로 처리하였다. 그렇다면, 교차로 노드 E를 경유하는 가능한 P-turn경로는 <그림6>(b)에서 보는 바와 같이, H→E→B→C→F→E→D 경로가 존재하게 된다.

여기에서 앞에서 설명한, U-turn의 적용법을 사용해서 가상의 호 F→D경로(단방향으로 링크시킴)를 <그림6>(b)와 같이 추가시키면, 이미 경유한 노드(노드 E)를 두 번 지나지 않는다는 Dijkstra 알고리즘의 기본 개념에 위배되지 않으므로 U-turn과 P-turn이 모두 가능하게 된다.

단, 최적경로의 출력에 있어서 U-turn의 경우에 H→E→F→D, P-turn의 경우에 H→E→B→C→F→D로 출력하는 것 대신에, 실제 상황에 맞는 결과인 H→E→F→E→D, H→E→B→C→F→E→D로 출력시켜야 한다. 따라서 본 알고리즘을 실제 프로그램으로 구현하여, 사용자에게 최적경로를 제공할 때에 이 점을 고려해야 한다.

이 문제의 해결은 알고리즘을 시행한 후, 종착노드까지의 최적경로를 출력할 때, 종착노드로부터 그 전번노드를 탐색하는 방법을 사용한 본 알고리즘의 특성을 사용하여, U-turn과 P-turn을 위해 첨가시킨 호, 즉 F→D링크의 후행 노드인 노드D의 전번 노드가 선행 노드인 노드 F가 되면 경로에 있어서 노드 E를 경유하는 것으로 출력 되도록 실제 프로그램에서 비교 연산을 추가시킨다. 이러한 방법은 알고리즘의 설계상, 하나의 가상의 호를

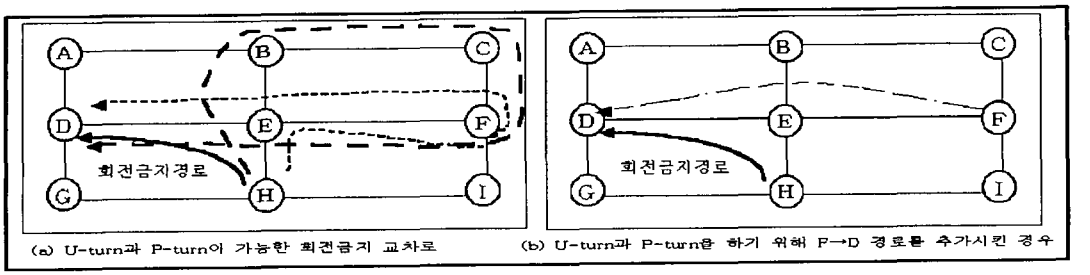


그림 6 P-turn과 U-turn의 적용을 위한 교통 네트워크

추가하고, U-turn 요소에 하나의 비교연산만을 추가시킴으로써 U-turn을 실제적으로 구현할 수 있고, 시스템 구축에 효율을 줄 수 있다.

이 방법은 다음과 같은 경우를 고려하여야 한다. 가상의 호 F→D를 추가시킴에 있어, 실제로 F→D 경로가 본래의 네트워크에서 이미 존재할 때이다. 이 경우에는 노드F 에서 U-turn이 가능한 사전 정보를 기본으로 하여, 가상의 호 F→D의 값과 실제의 호 F→D값을 비교하여 가상의 호가 더 작으면, 본 논문에서 제시한 개선된 Dijkstra 알고리즘에 U-turn / P-turn 요소로 추가하여 가상의 호 F→D값(실제로는 F→E값과 E→D값의 합)을 실제 호 F→D값 대신 사용하여 Dijkstra 알고리즘을 실행하면 될 것이다. 그러나, 실제 존재하는 호 F→D 값이 더 작다면, 가상의 호에 의한 U-turn 이나 P-turn을 경유하는 경로가 최적의 경로로 산출될 수 없기 때문에 노드F에서 노드E를 거쳐 노드D로 가는 U-turn / P-turn 요소에 가상의 호 F→D를 추가시키지 않으면 된다.

4. 최적경로 안내 시스템 설계 및 실험 결과

본 논문에서 제시한 좌회전 금지, U-turn 및 P-turn을 고려한 개선된 Dijkstra 알고리즘은 다음과 같이 <표 1>로 정리 될 수 있다. 즉, 적용하고자하는 네트워크에 회전금지 교차로 내에서, U-turn 및 P-turn을 고려해 주기 위한 호를 <그림 6>(b)와 같이 가상의 호 F→D를 추가시킨 상태에서 <표 1>의 알고리즘을 실행하면, 모든 고려사항에 만족하는 결과가 출력된다.

<표 1>에 나타나 있는 좌회전금지, U-turn 및 P-turn을 고려한 개선된 Dijkstra 알고리즘을 Visual한 환경을 지원하는 Visual C++를 사용하여, <그림 4>의 네트워크를 기반으로 아래와 같이 사용자가 직접 사용할 수 있는 최적경로 안내 시스템 프로토타입을 구축하였다.

우선, 최적경로 안내 시스템을 사용하는 방법을 설명하고, 모든 제약이 없는 경우와 회전금지 제약과 U-turn 및 P-turn을 적용한 예를 설명하겠다.

다음은 최적경로 안내 시스템을 적용하는 방법이다.

첫째, 노드의 개수를 입력하고 입력버튼을 누른다.
(<그림 7>의 경우, 노드수 15 입력)

둘째, 거리입력표에서 기본이 되는 네트워크의 출발과 종착노드에 따른 값을 입력한다.
(호가 없는 경우에는 default 값으로 무한대를 표시하는 99999를 입력)

셋째, 회전금지경로의 개수를 입력하고 회전금지입력버튼을 누른다.
(<그림7>(b)의 경우, 회전금지경로수 2 입력)

넷째, 회전금지경로를 입력한다.
(<그림7>(b)의 경우, 노드 1→4→7, 노드 6→5→8이 회전금지)

다섯째, U-turn 요소의 개수를 입력하고 U-turn입력버튼을 누른다.
(<그림7>(b)의 경우, U-turn개수 2를 입력)

여섯째, U-turn하는 경로를 입력하고, U-turn에 의한 경로버튼을 눌러 거리입력표를 개선한다.
(<그림7>(b)의 경우, U-turn이 가능한 노드 4→3→4→7경로를 가상의 호 3→7를 첨가하여 4→3→7로 입력하고, 노드 5→2→5→8경로를 가상의 호 2→8을 첨가하여 5→2→8로 입력한다. 그리고 경로변경버튼을 누르면, 경로입력표 내의 값들(호 3→7의 값이 만약 기존의 호 3→7이 존재하고 그 호의 값이 4보다 크면 4로, 호 2→8의 값이 만약 기존의 호 2→8이 존재하고 그 호의 값이 3보다 크면 3으로 바뀜)이 자동으로 개선된다.)

일곱째, 종착노드를 입력하고, 출력 버튼을 누른다.
(<그림7>의 경우, 종착노드로 15를 입력)

여덟째, 최단거리와 최적경로를 확인한다. (결과출력)

<표 1> 좌회전 금지, U-turn 및 P-turn을 고려한 Diikstra 알고리즘의 정리

d_{ij} : 노드 i 로부터 노드 j 까지의 거리.

π_j^b : 출발노드(노드1)로부터 노드 j까지 임시표지된 값.

b (위 첨자) : 임시표지된 노드 j의 전번노드의 번호.

π_j^* : 출발노드로부터 노드 j까지 최종 결정된 최적값.(영구표지된 값)

【단계 0】. [초기화]

U-turn이 가능한 노드의 네트워크 정보에 따라 가상의 호(d_{ij} 값 추가)를 연결.

(만약, 연결하고자 하는 노드간에 실제 호가 존재한다면, 노드의 중복을 피하기 위한 가상의 호가 실제 존재하는 호보다 작을 경우만 가상의 호 값을 적용한다.)

모든 노드j에 d_{1j}를 임시표지 π_j^b 로 기록한다. (j=2,...,N).

노드 1과 j를 연결하는 호가 없으면 d_{1j}=∞라고 간주함.

$\pi_1^* \leftarrow 0$

$\pi_j^b \leftarrow d_{1j}$ (j= 2, ...,N)(ALL b= 1)

【단계 1】. 모든 임시표지 π_j^b 중에서 최소치를 선택하여 이를 영구표지함.

영구표지된 노드를 i라고 함.

$\pi_i^b \leftarrow \min_j [\pi_j^b]$

π_i^b 를 π_i^* 로 수정한다.

더 이상 임시표지가 없으면 단계 3으로 간다.

【단계 2】. 단계 1에서 영구표지된 노드 i의 모든 인접노드 j 의 임시표지를 수정함.

영구표지 π_i^* 에 호의 길이 d_{ij} 를 합한 값과 기존의 임시표지 π_j^b 를 비교하여 작은 값을 선택함.

만약, (b-i-j 경로 = 회전금지 경로)이면, j 노드로의 연산은 실행하지 않음. (즉, b-i-j 경로는 최적경로 상에서 배제 함.)

그렇지 않으면(b-i-j 경로 ≠ 회전금지 경로),

$\pi_j^b \leftarrow \min[\pi_j^b, \pi_i^* + d_{ij}]$

그리고, ($\pi_j^b > \pi_i^* + d_{ij}$)이면, π_j^b에서 b = i 로 변환.

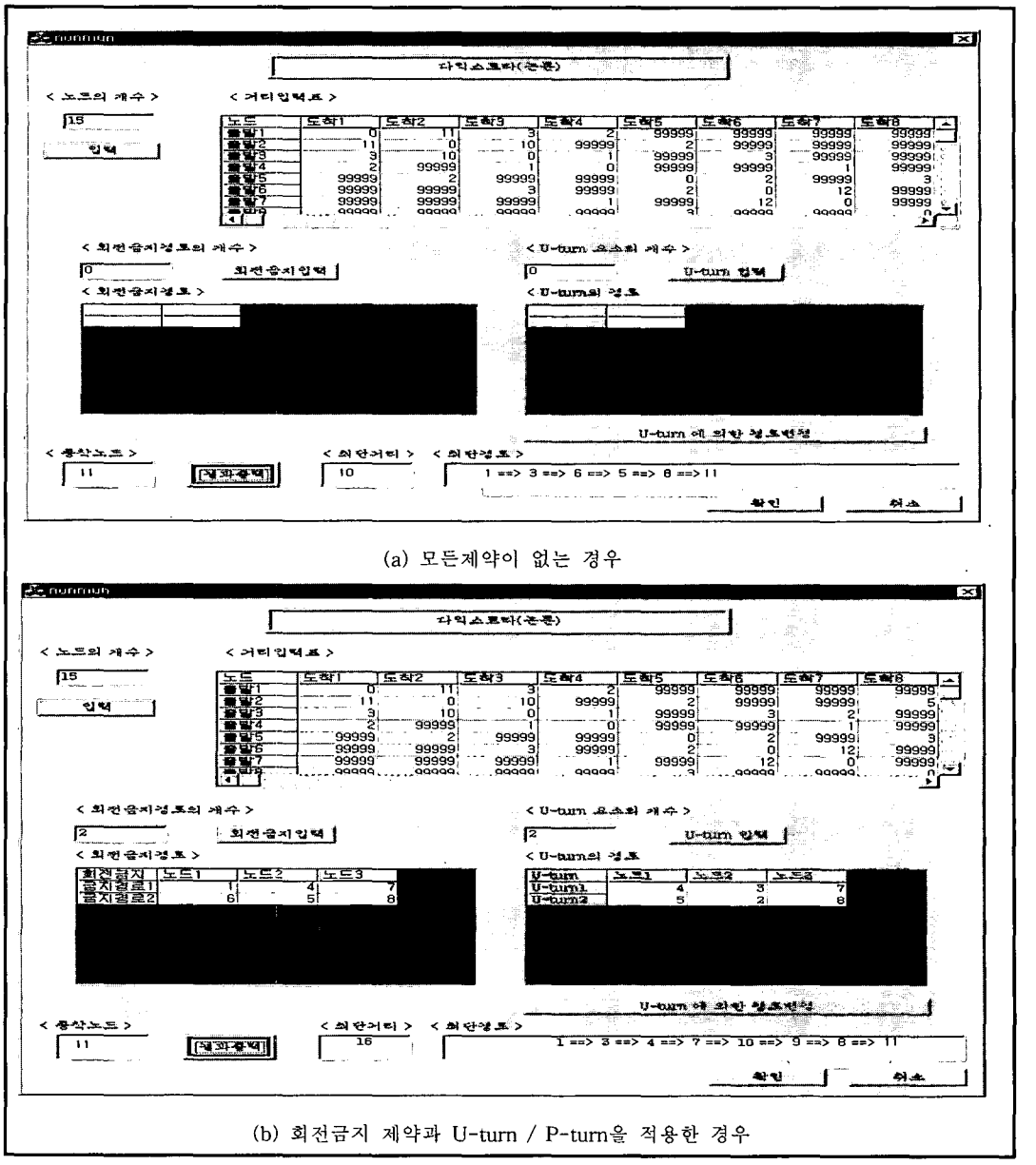
단계 1로 되돌아간다.

【단계 3】. [최적경로의 제시 / 종료]

최적경로를 종착노드에서 시작하여, 전번노드 b를 추적함으로써 최적경로를 제시함.

(이때, U-turn/P-turn을 고려하기 위하여 추가 했던 가상의 호를 없애고, 실제 경로를 제시함.)

알고리즘을 종료한다.



(a) 모든제약이 없는 경우

(b) 회전금지 제약과 U-turn / P-turn을 적용한 경우

그림 7 그림 4 의 네트워크를 적용한 최적경로 안내 시스템 사용예

모든 제약이 없는 <그림7>(a)에서 보는 바와 같이, 출발노드 1에서 종착노드 11까지의 최단 경로가 회전 금지 경로인 6→5→8 경로를 경유하고 있으나, 현재 설정에서는 회전금지를 고려하지 않고 있으므로, 최단거리가 10이고, 최적경로는 1→3→6→5→8→11가 된다.

회전금지 제약과 U-turn / P-turn을 적용한 <그림7>(b)에서 보는 바와 같이, 출발노드 1에서 종착노드 11까지의 최단경로가 제약이 없을 때는 회전 금지 경로인 6→5→8 경로를 경유하였으나, 제약을 적용하였을 때, 회전 금지 경로를 피하여, 새로운 경로를 탐색하게 된다.

표 2 각 알고리즘의 저장공간과 연산량 비교.(N: 노드의 개수, n: 회전금지의 개수)

구분	Dijkstra	Mid-Block 법	전노드, 전전노드고려	본논문의 알고리즘
기억용량	N^2 (거리데이터) N (π값)	유한링크의 수에따라 기하급수적으로 늘어남	N^2(거리 데이터) N (π 값) $2N$ (전노드 값)	N^2(거리 데이터) N (π값) N (전노드 값)
계산량	N^2 (덧셈연산) N^2 (비교연산)	유한링크의 수에따라 기하급수적으로 늘어남	N^2 (덧셈연산) N^2 (비교연산) nN^2 (비교연산)	N^2 (덧셈연산) N^2 (비교연산) nN^2 (비교연산)

이때, U-turn(노드 4→3→4→7, 노드 5→2→5→8)을 고려하여 최적경로를 산출하기 위해, 노드3→7 경로값이 무한대 값에서 "호(3-4) + 호(4-7)"의 값 "4"로, 노드 2→8 경로값이 무한대 값에서 "호(2-5) + 호(5-8)"의 값 "3"으로 계산에 사용된다.

결과적으로, 좌회전금지, U-turn 및 P-turn을 고려하여, 최단거리는 [16]이고, 최적 경로는 [1→3→4→7→10→9→8→11]이 결정되었다.

5. 개선된 알고리즘의 적합성 검증

개선된 알고리즘의 다른 알고리즘과 비교우위에 대한 적합성은 기억용량과 계산량의 기준에서 평가하기로 한다. 더불어 알고리즘의 시스템 구성상의 용이성도 참조하겠다.

고전적인 방법의 Dijkstra 알고리즘과 회전 금지 경로와 U-turn 및 P-turn을 적용시킨 지금까지의 논문에서 제시한 여러 알고리즘에 대해서 기억용량과 계산량을 비교해보면 <표 2>와 같다.

<표 2>에서 보는 바와 같이 회전 금지 경로와 U-turn 및 P-turn를 적용시킨 알고리즘들에서 본 논문의 알고리즘이 우수한 것을 알 수 있다. 그러나 전노드, 전전노드를 고려한 알고리즘과 거의 유사한 적합성을 갖는다.

그러나, 실제 현실에서 최적경로 안내 서비스를 제공하기 위한 컴퓨터 시스템에 적용하기 위한 알고리즘으로는 본 논문의 알고리즘이 용이하다.

6. 결론 및 알고리즘의 차후 적용 방안

본 논문에서는 실제 교통네트워크에서 최적경로를 찾을 때, 회전금지를 고려하기 위해 회전이 일어나는 3개의 노드를 이미 주어진 회전금지 정보 3개의 노드와 비교하여 일치하면, Dijkstra 알고리즘에 의한 최적경로 탐색 구간에서 배제시켰다. 또한 U-turn 및 P-turn을 고려하기 위해 이미 주어진 U-turn 및 P-turn 구간에서 가상의 호를 첨가하여, 이 요소를 고려하는데 노드의 중복사용에 의한 알고리즘한계를 극복하고자 했다.

본 논문에서 제시한 개선된 알고리즘을 바탕으로 사용자들이 사용할 수 있는 최적경로 안내 시스템 프로토타입을 Visual C++로 구현하였다. 개선된 알고리즘은 과거의 알고리즘에 비해 상대적으로 회전금지, U-turn 및 P-turn을 고려하는데 기억용량 및 계산량의 효율이 높고 시스템 구현이 용이하였다. 본 논문의 차후 적용 방안으로, 최근에 관심을 끌고 있는 첨단 교통정보시스템(Advanced Traveller Information System, ATIS)의 최적경로안내 서비스를 하기 위해서, 본 논문에서 새롭게 제시한 알고리즘을 적용할 수 있다.[2][9] 즉, 교통정보센터의 실시간 데이터를 사용하여, 본 논문에서 제시한 개선된 알고리즘에 의한 최적의 경로를 무선 송수신장치를 장착한 차량에 통신 시스템의 중계를 통해 지원할 수 있다.

참고문헌

[1] 강맹규, "네트 워크와 알고리듬" pp 93 ~ 105, 1995년

[2] 김성수, "지능형교통시스템을 위한 정보통신 매체와 동적경로 유도체계 확립에 관한 연구", IE Interfaces 제 11권 제 1호, pp 33-40, 1998년

[3] 김성수, 차영민 "첨단교통정보시스템의 최적경로 알고리즘 개발", working paper, 2000, 강원대학교 산업공학과, 최적화 실험실

[4] 박찬규, 박순달, 진희채, "교차로 제약과 지연이 있는 네트워크에서 최단경로탐색"

[5] 최기주, " U-TURN을 포함한 가로망 표현 및 최단경로의 구현 ", 대한교통학회지 제13권 제3호, 1995년

[6] Ahuja, R. K, Magnamti, T. L., and Orlin, J. B., " Network Flows ", Prentice-Hall, 1993

[7] Hillier, F. S. and Lieberman, G. J.," Introduction to Operations Research ", 6th ed, 1995

[8] Kim, Sung-Soo, Lee, Jong-Hyun, and Kim, Hyung-Wook, "Development of Real Time Mobile Management System and Traveler Information System using Integrated Logistics Information Network", 5th World Congress on Intelligent Transport Systems, Seoul, Korea, October 1998,

[9] Murty, Katta G., "Network Programming", Prentice-Hall, 1992