교통 상황의 변화를 고려한 동적 최단경로 탐색 알고리즘

A Dynamic Shortest Path Algorithm with Change of Traffic Conditions

정 희 석
(서울대학교 지구환경시스템공학부 석사과정)

김 창 호
(美 일리노이대학 도시 및 지역계획학과 석좌교수)

박 창 호
(서울대학교 지구환경시스템공학부 교수)

Key Words : 통행비용, 동적 변화, phase, Dijkstra 알고리즘

목 차

I. 서론

현재 세계적으로 활발히 진행되고 있는 지능형 교통체계(Intelligent Transportation System : ITS) 분야의 연구와 더불어 현실성이 반영된 합리적인 경로를 도출할 수 있는 최단경로 탐색 알고리즘의 개발에 대한 중요성이 날로 높아지고 있다. 최단경로 탐색 알고리즘은 ITS의 범주에 속하는 첨단여행자 정보체계(Advanced Traveler Information System : ATIS)의 운영을 위한 핵심 요소이다.

최근에 많은 사람들이 이용하고 있는 교통정보 안내 서비스는 교통수단 활용의 질을 높이고, 그에 따라 사회적 비용 또한 절감시키는 등 매우 큰 역할을 하고 있다. 특히 도시 내 교통 상황은 변동 속도가 매우 빨라서 한시 앞의 상황도 예측하기 어렵기 때문에 정확한 교통정보의 안내는 더욱 절실하다. 그러나 이러한 교통 상황 변동을 제대로 반영하지 못한다면 정확성이 결여된 질 낮은 정보를 이용자들에게 제공할 수밖에 없게 되며, 이용자들로부터 신뢰받지 못하는 쓸모없는 정보들을 생성하여 오히려 사회적 비용을 낭비하게 되는 역효과를 초래할 수도 있다.

기존의 최단경로 탐색 알고리즘들을 통해 고정된 통행 비용을 바탕으로 최단경로를 탐색하는 것은 시간이 흐름에 따라 비용이 변하는 실제 상황을 제대로 고려하지 못하는 단점이 있다. 이에 따라 본 연구에서는 기존의 Dijkstra 알고리즘을 바탕으로 하여 일정 시간간격에 따라 변하여 입력되는 통행 비용을 반영하여 동적 최단경로를 도출하는 알고리즘을 개발하고자 한다.

II. 기존 알고리즘의 고찰

네트워크의 모든 링크가 비음수(nonnegative)의 통행비용을 가질 때 특정 노드로부터 다른 모든 노드까지의 최단경로를 구하는 가장 효율적인 알고리즘들 중의 하나가 바로 Dijkstra 알고리즘이다. 이 알고리즘은 기점으로부터 다른 모든 노드까지의 총 길이가 최소가 되도록 구성하기 위해 노드를 "기점으로부터의 최소비용이 알려진 노드"와 "그 외의 노드"로 분류한다. 이렇게 분류된 집합을 이용하여 모든 노드들의 거리표지를 무한대(∞)로 갖는 임시표지로 지정하고, 시작노드는 영(0)의 값을 갖는 영구표지로 지정한다. 시작노드에 연결된 노드들의 거리 표지는 초기에 지정된 상한값으로부터 시작노드에서 이들 노드까지의 실제 거리값들로 갱신됨과 동시에 임시표지로 지정된다. 그런 다음 이 임시표지로 지정된 노드들 중에서 시작노드로부터 가장 작은 링크 길이의 합을 갖는 노드를 다음 노드로 선택하고 이 노드를 영구표지로 지정한다. 이어지는 각 반복 단계에서 모든 노드들이 영구표지로 지정될 때까지 이러한 과정들이 반복적으로 수행된다.

다음은 Dijkstra 알고리즘에서 이용되는 변수 및 구체적인 수행 과정이다.

S : 이미 영구표지가 기록되어 기점으로부터의 최소통행비용이 알려진 노드들의 집합

T : 아직 영구표지가 기록되지 않은 노드들의 집합

π_i^* : 노드 i까지의 확정된 최소통행비용(영구표지)

π_j : 노드 j까지의 최소통행비용 추정치(임시표지)

d_{ij} : k번째 phase의 노드 i 와 노드 j 사이의 링크통행비용

p_j : 노드 j의 앞 노드

단계 0 : 초기화

· $S = \{1\}$

· $T = \{j \mid$ 노드 1을 제외한 네트워크를 구성하는 모든 노드$\}$

· $\pi_1^* \leftarrow 0$

· $p_1 \leftarrow 0$

· $\pi_j \leftarrow d_{1j}\ (j=2,\cdots,N)$

· $p_j \leftarrow 1\ (j=2,\cdots,N)$

단계 1 : 영구표지

· $\pi_i^* \leftarrow \min[\pi_j]$

· $S \leftarrow S \cup \{i^*\}$

· $T \leftarrow T - \{j\}$

· 만약 더 이상 임시표지(j)가 없으면, 단계 3으로 간다.

단계 2 : 임시표지 수정

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

· 만약 $\pi_i^* + d_{ij} < \pi_j$이면, $p_j \leftarrow i^*$

· 단계 1로 되돌아간다.

단계 3 : 종료

· 목적지로부터 p_i를 이용하여 최단경로를 역추적한다.

· 만약 $p_i = 0$이면, 역추적을 중단한다.

· 끝낸다.

즉, Dijkstra 알고리즘은 최소의 임시표지를 갖는 노드를 선택하고, 그 노드를 영구표지로 지정하며, 그 노드에 인접하고 있는 노드들의 거리표지를 갱신하기 위해 점점 확장하여 나아가는 방법인 것이다. 이 방법은 또한 탐욕적 알고리즘(greedy algorithm)의 대표적 예로서 잘 알려져 있는데, 일반적으로 탐욕적 알고리즘은 각 단계에서 최선(best)으로 보이는 것을 선택 또는 수행함으로써 문제를 해결해 나가는 방법이다.

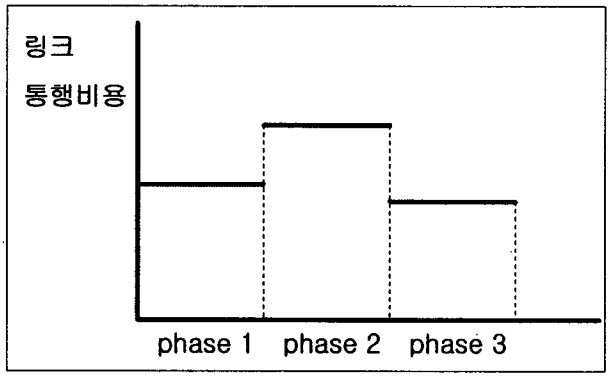
III. 동적 알고리즘의 개발

1. 알고리즘의 기본 개념

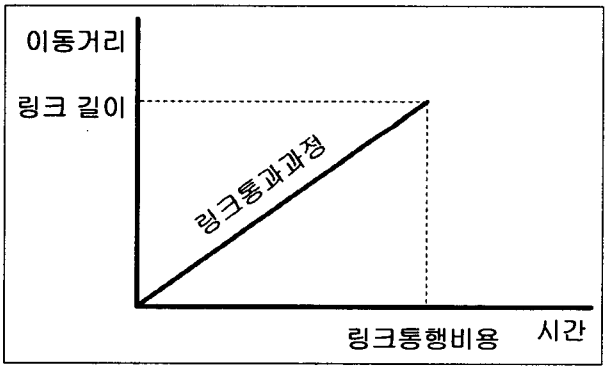
Dijkstra 알고리즘은 링크 통행비용이 고정되어 있어 시간의 경과에 따른 링크 통행비용의 변화를 전혀 반영할 수 없는 단점이 있다. 본 연구에서는 Dijkstra 알고리즘을 바탕으로 통행비용의 동적 변화를 고려하기 위해 통행비용이 갱신되는 "시격(time interval)"을 설정하고, 각각의 시간 간격들을 "phase"라는 단위로 구분하였다. 즉, phase는 일정한 시격만큼 경과하면 다음 phase로 넘어가고, 각 phase마다 링크의 통행비용이 달라지는 것이다.

이 때 링크 상을 진행하고 있는 도중에 phase가 바뀌는 경우에 대한 처리가 필요한데, 이는 다음과 같은 가정을 전제로 한다.

- 같은 phase 동안의 링크 통행시간은 일정하며, phase가 바뀌면 링크 통행시간은 불연속적으로 변화한다. <그림 1>
- 같은 phase 동안 링크를 진행할 때는 일정한 속도로 진행한다. <그림 2>



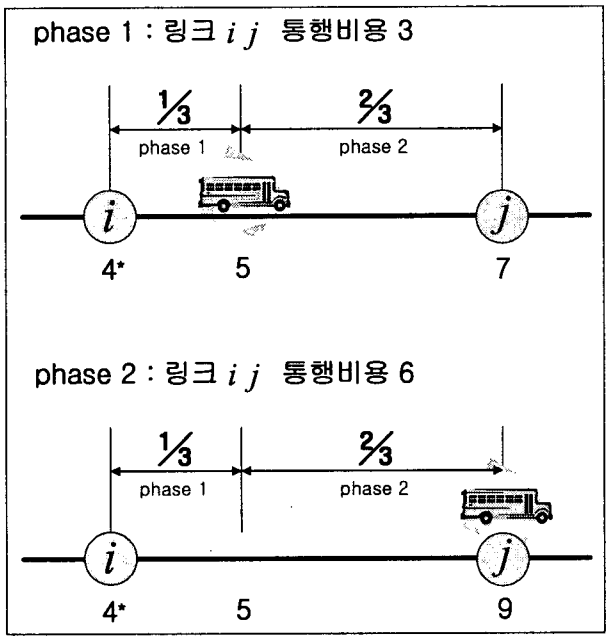
<그림 1> 가정 1 - 링크통행비용 조건



<그림 2> 가정 2 - 링크진행속도 조건

<그림 3>은 매 5분마다 phase가 바뀌는 네트워크 상의 링크를 나타내고 있다. 노드 i의 영구표지가 4이고, 첫 번째 phase에서는 링크 ij의 통행비용이 3이지만 두 번째 phase에서는 6으로 바뀌어서 차량이 링크 ij 상을 통과하는 도중에 phase가 바뀌게 된다. 이 경우, 노드 i를 통과하는 시점으로부터 phase가 바뀌기까지는 1분이 남았고 통행비용이 3이므로 가정에 의해 첫 번째 phase 동안에는 링크 ij 길이의 1/3만큼 이동하게 된다. 두 번째 phase에는 링크 ij의 통행비용이 6으로 바뀌고, 이는 노드 i로부터 노드 j까지의 링크 전 구간을 통과하는데 드는 비용이다. 그런데 이 경우에 차량은 링크의 2/3만 통과하면 되므로 기점으로부터 노드 j에 도달

하는데 드는 통행비용은 $5+(2/3)\times 6=9$가 된다. 이것이 본 연구에서 제시하는 알고리즘의 기본 개념이다.



<그림 3> 동적 알고리즘의 기본 개념

2. 알고리즘의 개발

제안되는 알고리즘은 다음과 같은 변수들을 사용하여 개발되었다.

S : 이미 영구표지가 기록되어 기점으로부터의 최소통행비용이 알려진 노드들의 집합

T : 아직 영구표지가 기록되지 않은 노드들의 집합

π_i^* : 노드 i까지의 확정된 최소통행비용(영구표지)

π_j : 노드 j까지의 최소통행비용 추정치(임시표지)

I : 시격(time interval)

k : phase의 변화를 나타내는 카운터

d_{ij}^k : k번째 phase에서의 노드 i와 노드 j 사이의 링크통행비용

p_j : 노드 j의 앞 노드

f_{ij} : 차량이 노드 i와 노드 j 사이의 링크를 진행하는 도중에 다음 phase로 시간이 경과하는 경우, 다음 phase에서 진행할 남은 링크길이의 비율

$\tilde{\pi}_i$: 노드 i에서 더 이상 진행하지 못하고 다음 phase로 시간이 경과했음을 나타내는 factor

알고리즘을 구성하는 구체적인 단계들은 다음과 같다.

단계 0 : 초기화

- $S=\{1\}$
- $T=\{j \mid$ 노드 1을 제외한 네트워크를 구성하는 모든 노드$\}$
- $\pi_1^* \leftarrow 0$
- $k \leftarrow 1$
- $p_1 \leftarrow 0$
- $f_{ij} \leftarrow 1 \ (\forall\ i, j)$
- $\pi_j \leftarrow d_{1j}^1 \ (j=2,\cdots,N)$
- $p_j \leftarrow 1 \ (j=2,\cdots,N)$
- $\tilde{\pi}_1 \leftarrow 0,\ \tilde{\pi}_j \leftarrow 0 \ (j=2,\cdots,N)$
- I 설정

단계 1 : 조건 검사

- 만약 $\min[\pi_j] \le kI$이면, 단계 2로 간다.
- 만약 $\min[\pi_j]=\infty$이면, 단계 5로 간다.
- 만약 $\min[\pi_j]\neq\infty$이고 $\min[\pi_j]>kI$이면, 단계 4로 간다.
- 만약 더 이상 임시표지(j)가 없으면, 단계 6으로 간다.

단계 2 : 영구표지

- $\pi_i^* \leftarrow \min[\pi_j]$
- $S \leftarrow S \cup \{i^*\}$
- $T \leftarrow T-\{j\}$

단계 3 : 임시표지 수정

- $\pi_j \leftarrow \min[\pi_j,\ \pi_i^*+d_{ij}^k]$
- 만약 $\pi_i^*+d_{ij}^k<\pi_j$이면, $p_j \leftarrow i^*$
- 단계 1로 되돌아간다.

단계 4 : 링크 상 진행 도중 phase 변화시

- 노드 j에 연결된 모든 노드 i에 대해 차례로 다음의 식을 적용하여 갱신한다.

$$\tilde{\pi}_i \leftarrow kI$$

$$f_{ij} \leftarrow f_{ij}-\frac{\min[\tilde{\pi}_i-\pi_i^*,\ I]}{f_{ij}\times d_{ij}^k}$$

- $\pi_j \leftarrow \infty$
- $p_j \leftarrow \infty$
- 단계 1로 되돌아간다.

단계 5 : 통행비용 갱신

- $k \leftarrow k+1$
- 모든 링크의 통행비용을 새로운 d_{ij}^k로 갱신
- $\tilde{\pi}_i=(k-1)I$인 모든 노드 i들을 차례로 비교하여 다음의 식으로 π_j를 갱신한다.

$$\pi_j \leftarrow \min[\pi_j,\ (k-1)I+f_{ij}\times d_{ij}]$$

· 만약 $(k-1)I+f_{ij}\times d_{ij}<\pi_j$이면, $p_j \leftarrow i$

· 단계 1로 되돌아간다.

단계 6 : 종료

· 목적지로부터 p_i를 이용하여 최단경로를 역추적한다.

· 만약 $p_i=0$이면, 역추적을 중단한다.

· 끝낸다.

이 알고리즘의 핵심은 크게 단계 2와 3, 단계 4, 단계 5의 세 가지로 구성되어 있다고 할 수 있다. 단계 2와 3은 Dijkstra 알고리즘에서 이용하고 있는 임시표지와 영구표지 기록 과정과 동일하다. 단계 4는 링크 상을 통과하는 도중에 phase가 바뀌는 경우에 phase가 바뀌기 이전까지의 통행을 기록하기 위한 과정이고, 단계 5는 phase가 바뀐 후에 갱신된 통행 비용을 이용하여 노드들에 임시표지를 다시 기록하는 과정을 나타낸 것이다. 즉, 단계 4와 단계 5는 통행 비용의 동적 변화를 고려하기 위해 Dijkstra 알고리즘에 새로 추가된 단계라고 할 수 있다.

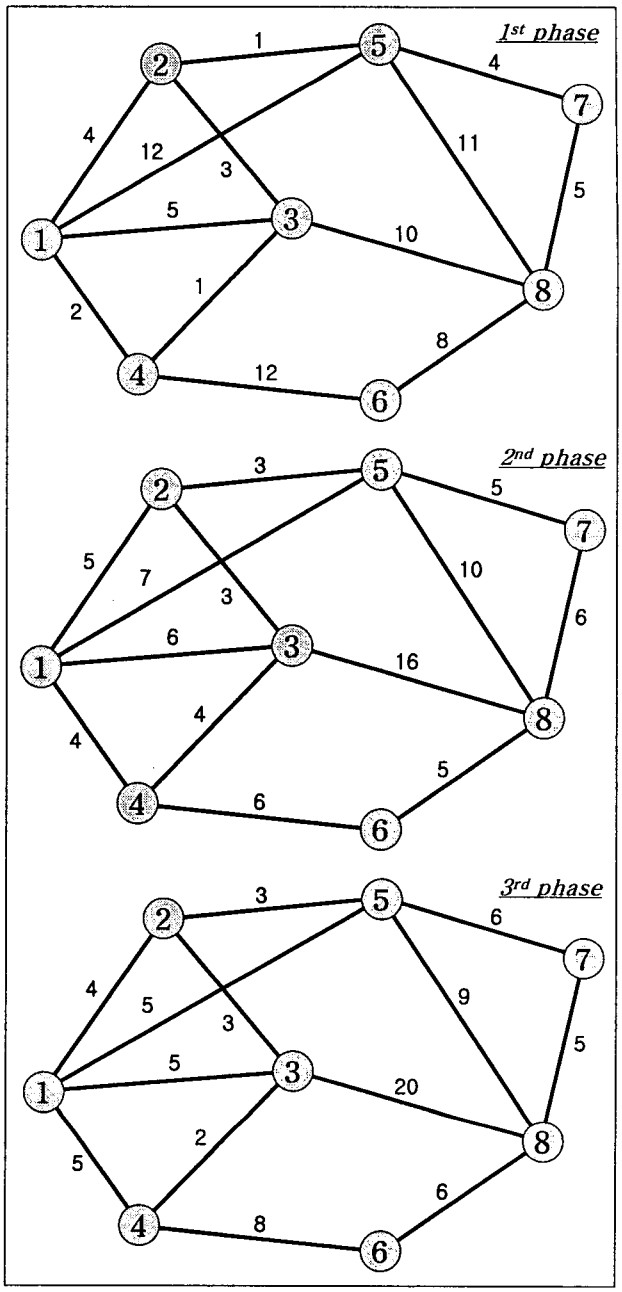
이 외에 단계 0은 Dijkstra 알고리즘의 초기화와 마찬가지로 노드들을 "기점으로부터의 최소비용이 알려진 노드"와 "그 외의 노드"로 분류하고, 각 변수들을 초기화하며, 시격을 설정한다. 단계 1은 경로탐색 상황을 파악하여 단계 2, 단계 4, 단계 5로 분기시키는 역할을 한다. 마지막으로 단계 6은 알고리즘의 종료 단계로써 각 노드에 기록된 전노드 정보를 이용하여 역추적하여 목적지까지의 최단경로를 최종적으로 도출한다.

IV. 적용 예제

1. 동적 알고리즘의 적용

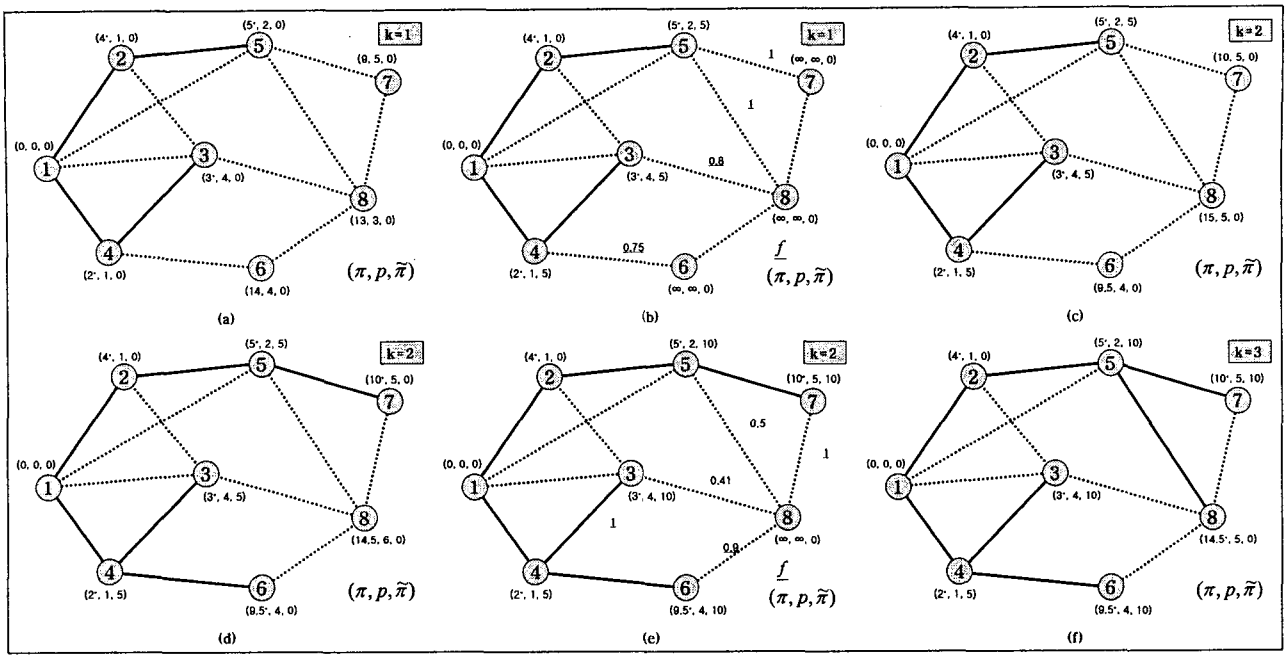
<그림 4>에 예시된 간단한 네트워크를 이용하여 본 연구에서 제안된 알고리즘의 동작 과정을 살펴보고자 한다. 예시된 네트워크는 8개의 노드와 13개의 링크로 구성되어 있고, 각각 5분의 시격으로 3개의 phase 동안의 통행 비용이 나타나 있다.

이 네트워크에 본 연구에서 개발한 알고리즘을 적용하면, 먼저 단계 0과 1을 거쳐 Dijkstra 알고리즘과 동일한 과정인 단계 2, 3을 몇 회 반복하게 되고, <그림 5> (a)에 나타나는 바와 같이 노드 1, 2, 3, 4, 5에는 영구표지가 기록되며, 나머지 노드 6, 7, 8에는 임시표지가 기록된다. 그런데 이 임시표지들은 모두 두 번째 phase로 바뀌는 시간인 5분을 초과하기 때문에 더 이상 단계 2, 3을 반복할 수 없고, 단계 4로 넘어가게 된다. 단계 4에서는 <그림 5> (b)와 같이 집합 S의 원소인 노드 3, 4, 5와 집합 T의 원소인 노드 6, 7, 8을 직접 연결하는 링크들에 대해 phase가 바뀌기 전까지 통행하고 남은 구간의 비율들 f_{ij}를 계산하고, 노드 6, 7, 8의 임시표지와



<그림 4> 적용 예제

전노드 정보를 모두 무한대(∞)로 표시한다. 이 과정을 반복하면 결국 집합 T에 속한 노드 6, 7, 8의 임시표지가 모두 무한대(∞)가 되므로 단계 5로 넘어간다. 단계 5에서는 두 번째 phase로 이동하여 새로 갱신된 통행 비용과 f_{ij}를 이용하여 <그림 5> (c)와 같이 집합 T에 속한 노드 6, 7, 8의 임시표지를 모두 새로 기록한다. 그리고 단계 2, 3을 반복하면 <그림 5> (d)와 같이 노드 6, 7이 집합 T에서 집합 S로 이동하여 영구표지를 갖게 되고, 노드 8만이 집합 T에 남아서 임시표지를 갖는데 이것은 다음 phase로 바뀌는 시간인 10을 초과하게 된다. 따라서 <그림 5> (e)와 같이 단계 4를 통해 노드 3, 6, 7, 8과 노드 8을 직접 연결하는 링크들에 대해 f_{ij}를 계산하고, 노드 8의 임시표지와 전노드 정보를 모두 무한



<그림 5> 동적 알고리즘의 적용 과정

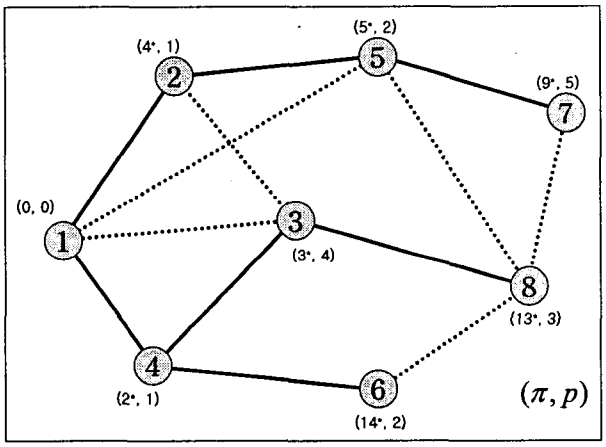
대(∞)로 표시한 후, 앞서의 경우와 마찬가지 방법으로 갱신된 통행비용과 f_{ij}를 이용하여 노드 8의 임시표지를 새로 기록한다. 마지막으로 단계 2, 3을 반복하면 <그림 5> (f)와 같이 노드 8에도 영구표지가 기록되어 네트워크 상에 더 이상의 임시표지가 존재하지 않으므로 단계 6으로 이동, 알고리즘이 종료된다.

알고리즘의 적용 결과, 기점노드 1로부터 종점노드 8까지의 최단경로는 1-2-5-8 순의 경로이며, 이 때의 통행 비용은 14.5이다.

2. 기존의 알고리즘을 적용한 결과와 비교

기존의 Dijkstra 알고리즘은 본 연구에서 제시한 동적 알고리즘의 단계 4와 단계 5를 제외한 것과 거의 일치한다. 그런데 앞서 설명한 바와 같이 단계 4와 단계 5는 네트워크 상의 통행 비용의 동적 변화를 반영하기 위한 단계이다. 다시 말해 Dijkstra 알고리즘은 시간의 경과에 따라 변하는 통행 비용을 고려하는 단계가 없다는 것이다. 따라서 Dijkstra 알고리즘에 <그림 4>에서 예시된 네트워크를 적용하기 위해서는 첫 번째 phase에서 나타난 통행 비용만을 이용할 수밖에 없고, 이를 적용하여 최단 경로를 구하면 결과는 <그림 6>과 같이 1-4-3-8 순으로 나타나며, 이 때의 통행 비용은 13이다.

본 연구에서 제시된 동적 알고리즘을 적용한 결과 <그림 5> (f)와 기존의 Dijkstra 알고리즘을 적용한 결과 <그림 6>의 차이는 쉽게 관찰할 수 있다. 이러한 차이가 나타나는 이유는 노드 5와 노드 8을 잇는 링크의 통행 비용은 서서히 감소하는데 반해, 노드 3과 노드 8을 잇는 링크의 통행 비용이 급격히 증가하는 등의 변화를 Dijkstra 알고리즘은 적절히 반영하지 못하기 때문이라고 할 수 있다.



<그림 6> 기존 알고리즘의 적용 결과

V. 결론

본 연구에서는 Dijkstra 알고리즘을 검토하여 시간의 경과에 따라 변하는 통행 비용을 전혀 반영할 수 없는 문제점을 도출하였다. 그리고 이런 문제점을 해결하기 위하여 Dijkstra 알고리즘을 바탕으로 하는 동적 알고리즘을 제시하였다.

Dijkstra 알고리즘은 최소길이 표지를 갖는 다음 노드를 선택하는 부분에서 엄청난 연산시간을 소모한다. 그동안 수많은 연구들이 Dijkstra 알고리즘의 이러한 문제점을 개선하기 위해 노력을 해왔고, 그 결과로 여러 가지의 수정된 Dijkstra 알고리즘들이 존재하는 것이 사실이다. 그러나 본 연구에서는 시간 경과에 따른 통행 비용의 변화라는 새로운 요소를 도입하는 것을 목표로 하였기 때문에 가장 기본적인 Dijkstra 알고리즘을 바탕으로 연구가 진행되었다. 그로 인해 제시된 동적 알고리즘은 연산시간에 관한 요소는 전혀 고려되지 못했

다는 한계점을 지니고 있다.

따라서 본 연구에서 제시된 동적 최단경로 탐색 알고리즘은 현재 많은 사람들이 이용하고 있는 Car Navigation System에 바로 적용하기에는 무리가 따른다. 그러나 향후에 이 알고리즘을 기반으로 하여 연산속도를 개선하는 방법론에 대한 연구와 더불어 통행시간 예측 분야의 연구가 동시에 활발히 이루어진다면 더욱 정확하고 신속한 양질의 최단경로 정보를 제공할 수 있을 것이다.

참고문헌

1. 강맹규(1991), 네트워크와 알고리듬, 박영사
2. 오승(2004), 위치기반서비스를 위한 휴리스틱 경로탐색 모형의 개발, 서울대학교 박사학위논문
3. 교통개발연구원(2003), Mobile 위치추적기반 교통정보안내 사업 제1차년도 최종보고서