韓國經營科學會誌
第23卷 第3號
1998年 9月

교차로 제약과 지연이 있는 네트워크에서 최단경로탐색*

박찬규** · 박순달** · 진희채***

A Fast Algorithm for Shortest Path Problem for Network with Turn Penalities and Prohibitions*

Chan-Kyoo Park** · Soondal Park** · Heu-Chae Jin***

Abstract

Shortest path problem in road network with turn penalties and prohibitions frequently arises from various transportation optimization models. In this paper, we propose a new algorithm for the shortest path problem with turn prohibitions and delays. The proposed algorithm maintains distance labels of arcs, which is similar to labels of nodes of Dijkstra's algorithm. Fibonacci heap implementation of the proposed algorithm solves the problem in $O(mn+m\log m)$. We provide a new insight in transforming network with turn penalties and prohibitions into another network in which turn penalties and prohibitions are implicitly considered. The proposed algorithm is implemented using new data structure and compared with Ziliaskopoulos' algorithm. Computational results show that the proposed algorithm is very efficient.

1. 서 론

도로네트워크에서 최단경로를 구하는 문제는 교통최적화에서 빈번히 발생하는 문제로서 경로 선정뿐만 아니라 교통량 할당이나 기타 교통 흐름의 분석 등의 부문제가 된다. 최근 들어 GIS(Geographic Information System)를 활용한 CNS(Car Navigation System)를 비롯한 다양한 응용 분야에서도 도로네트워크의 최단경로

* 본 연구는 정보통신부 '97년도 대학기초연구지원사업(97-G-0683)의 지원을 받음.
** 서울대학교 산업공학과
*** 한국 전산원

를 보다 빠른 시간에 찾을 필요가 있다.

도로네트워크에서는 현실적 상황을 고려한 최단경로를 찾기 위해 혼잡(congestion)과 교차로 등의 제약을 고려한 모델이 개발되었다. 혼잡은 도로의 통과속도나 통과시간이 교통량에 따라 달라진다는 사실을 반영하고자 개발된 개념으로 이명석([2]), Cooke([5]), Hall([9]), Kaufman([10]), Orda([12]), Ziliaskopoulos([14]) 등의 연구가 있었다. 또한 실제 도로 네트워크에는 교차로가 있고 교차로는 어느 방향으로 진입하느냐에 따라 나갈 수 있는 도로나 교차로 대기시간들이 달라지므로 교차로가 없는 경우와는 구별된다. 본 연구는 이러한 교차로 제약이 주어진 도로네트워크에서 최단경로를 찾는 문제를 다룬다.

도로네트워크에서 교차로 제약이 주어지면 기존의 Dijkstra 알고리즘이나 꼬리표수정(label-correcting)방법을 적용하여 최단경로를 구할 수 없다. 또한 가장 빠른 경로에 점의 반복이 발생할 수도 있다. 따라서 이를 해결하기 위한 교차로 제약과 교차로 회전시의 지연 등을 고려한 최단경로 알고리즘들이 개발되어 왔다.

경로선정에서 교차로 제약을 고려한 기존의 연구는 크게 2 가지 접근 방법에 의해 분류될 수 있다. 첫 번째 방법으로 네트워크를 적절히 변형하여 교차로 제약이 없는 네트워크로 만들어 기존의 최단경로 알고리즘을 적용하는 방법이다. 물론 네트워크 변형과정에서 교차로 제약이 변형된 네트워크에 암묵적으로 표현될 수 있어야 한다. 이러한 접근 방법으로 Yagar([13])의 연구와 Easa([7])의 연구 등이 있었다. Ziliaskopoulos([15])에 따르면 이러한 접근 방법은 네트워크의 점과 호의 증가로 많은 계산 시간과 저장공간이 요구되고 구현이 어려우며 네트워크 위상(topology)의 변경에 유연하지 못하다는 단점이 있다.

두 번째 접근 방법은 기존의 최단경로 알고리즘을 수정하여 교차로 제약을 고려할 수 있도록 수정하는 것이다. Kirby & Putts([11])는 Caldwell([4])의 연구를 기반으로 교차로 제약이 있는 최단경로 문제에서 최단경로가 만족해야 하는 필요충분조건을 제시하였다. Easa([6])는 위의 제시된 필요충분조건을 만족하는 꼬리표수정 알고리즘을 제안하였고 Ziliaskopoulos([15])는 교차로 회전시의 지연(turn penalties)을 추가로 고려한 꼬리표수정 알고리즘을 제안하고 아울러 이를 구현하기 위한 자료구조도 제시하였다. 또한 Florian & Nguyen([8])은 Dijkstra 알고리즘을 수정하여 금지된 호를 지나지 않는 알고리즘을 제안하였으나 최단경로를 구하지 못하는 예가 Easa([6])에 의해 제시되었다.

본 연구에서는 교차로 제약과 교차로 회전시의 지연을 고려할 수 있도록 기존의 꼬리표설정 알고리즘(label setting algorithm)을 수정한 새로운 꼬리표설정 알고리즘을 제시하고 아울러 제안된 알고리즘을 새로운 자료구조를 사용하여 구현하고 그 수행결과를 Ziliaskopoulos([15])의 알고리즘과 비교한다.

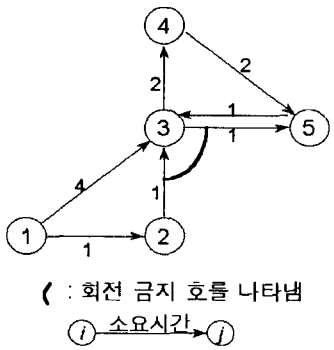
2. 교차로 제약이 있는 네트워크 변환에 관한 새로운 관찰

유방향 네트워크 $G=(N,A)$에서 점과 호의 나열 $i(=i_1)-a_1-i_2-a_2-\cdots-i_{r-1}-a_{r-1}-j(=i_r)$가 $1\le k\le r-1$인 모든 k에 대해 $a_k=$

$(i_k, i_{k+1}) \in A$이면 경로(path)라 한다. 경로 중에서 동일한 점이 반복되어 있지 않은 경로를 단순 경로(simple path)라 한다. 교차로 제약이 없는 네트워크에서 최단경로는 단순경로이지만 교차로 제약이 있는 네트워크에서 최단경로는 단순경로가 아닐 수 있다. 교차로 제약이 없는 네트워크에서의 최단경로문제 해법은 [1]을 참조하기 바란다.

교차로 제약이 있는 네트워크에서의 최단경로 선정 알고리즘을 제안하기에 앞서 다음과 같은 교차로 제약이 있는 네트워크를 교차로 제약이 암묵적으로 고려된 네트워크로 변형하는 새로운 방법을 보자.

다음과 같은 교차로 제약과 지연이 있는 네트워크를 고려해 보자.



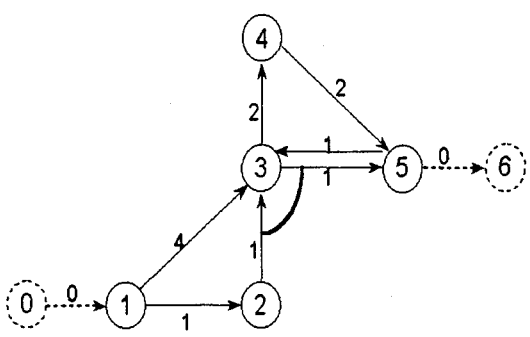
[그림 1] 교차로가 있는 네트워크([6])

교차로	지연 시간
①→②→③	0
①→③→④	2
①→③→⑤	2
②→③→④	1
③→④→⑤	1
④→⑤→③	1
⑤→③→④	0
③→⑤→③	0
⑤→③→⑤	0

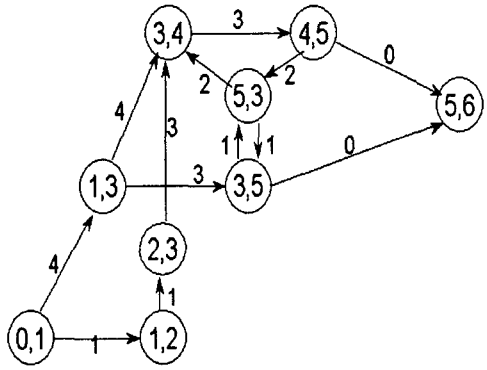
[그림 2] 교차로 지연시간

[그림 1]에서 점 ②에서 점 ③으로 진입하여 점 ⑤로는 갈 수 없다. 즉 점②→점③→점⑤는 갈 수 없는 경로이다. 또한 [그림 2]로부터 점 ①→점③→점④로 갈 때 교차로인 점 ③에서의 지연시간은 2임을 알 수 있다.

[그림 3]과 같이 점 ⓪과 점 ⑥을 첨가한다. 점 ⓪에서 출발점 ①로 가는 호를 추가하고 도착점 ⑤에서 점 ⑥으로 가는 호를 첨가한다. 물론 호 (⓪,①)은 회전 지연 없이 점 ①에서 나가는 모든 호를 통해 빠져나갈 수 있으며 점 ⑤로 들어오는 모든 호는 회전 지연 없이 호 (⑤,⑥)을 통해 점 ⑥으로 갈 수 있다. [그림 4]는 교차로 제약이 암묵적으로 고려된 변형된 네트워크로 [그림 3]의 각 호가 [그림 4]의 한 점에 해당된다. 예를 들어, [그림 3]의 호(①,③)이 [그림 4]의 점〈1,3〉이 된다. 그리고 [그림 4]의 각 점의 이웃점은 [그림 3]의 해당호의 이웃한 호들에 대응한다. 여기서 어떤 호의 이웃한 호란 그 호의 도착점을 출발점으로 하는 호들 중에 교차로 금지가 없는 호를 말한다. [그림 3]에서 호(①,③)의 이웃한 호의 집합은 {(③,④), (③,⑤)}이다. [그림 4]의 점〈1,3〉의 이웃한 점은 점〈3,4〉와 점〈3,5〉이 된다. [그림 4]의 호(〈a,b〉, 〈b,c〉)의 길이는 [그림 3]의 호(b,c)의 길이와 [그림 3]의 a→b→c의 교차로 지연시간을 합한 값이다. 예를 들어 [그림 4]에서 호(〈1,3〉,〈3,4〉)의 길이는 4가 되는데 이는 [그림 3]의 호(③,④)의 길이 2와 점①→점③→점④의 교차로 지연 시간 2을 더한 값임을 알 수 있다. 또한 [그림 4]에서 출발점은 점〈0,1〉이고 도착점은 점〈5,6〉이 된다.



[그림 3] 점과 호를 추가한 후의 네트워크



[그림 4] 변환된 네트워크

변형된 네트워크에서는 기존의 Dijkstra 방법을 사용하여 최단경로를 구할 수 있고 이 때 점〈0,1〉에서 점〈5,6〉까지 최단경로는 점〈0,1〉→점〈1,3〉→점〈3,5〉→점〈5,6〉이고 이는 원래의 네트워크에서 ①→③→⑤에 해당한다.

위에서 제시된 교차로가 주어진 도로네트워크를 교차로가 암묵적으로 고려된 네트워크로의 변형방법은 새로운 알고리즘의 핵심 개념이다. 제안될 알고리즘에서는 각 호별로 거리 꼬리표(distance label)를 유지하고 각 호의 이웃한 호를 탐색해 나가게 된다.

3. 새로운 알고리즘

점 i에서 점 j로 가는 최단경로를 $SR(i \rightarrow j)$로 표시하자. Dijkstra 알고리즘은 각 점의 거리 꼬리표를 붙여 출발점에서 도착점까지 최단경로를 구한다. 그러나 교차로 금지 제약이 추가되면 주어진 점을 어떤 호를 따라 진입하느냐에 따라 나갈 수 있는 호들이 결정된다. 따라서 각 점의 들어오는 각 호마다 하나씩 거리 꼬리표를 유지할 필요가 있다. 이는 교차로 제약이 추가되면 최단경로에 점의 반복이 나타날 수는 있으나 호의 반복은 나타나지 않는다는 사실과도 관련이 있다. 제안된 알고리즘에서는 네트워크의 모든 호에 대해 하나의 꼬리표를 유지하는데 이 때 각 호의 꼬리표는 해당 호를 따라 그 호의 도착점(head node)에 진입하는 경로의 길이를 의미한다.

알고리즘을 제시하기에 앞서 필요한 기호를 먼저 정의한다. 주어진 유방향 도로네트워크를 $G=(N, A)$이라 하고 $|N|=n$, $|A|=m$이라 하자. s를 출발점, t를 도착점이라 하고 호 e의 출발점과 도착점을 각각 $TAIL(e)$, $HEAD(e)$라 하고 소요시간을 $c(e)$ (≥ 0)라 하자. 또한 s에서 t로 가는 경로는 반드시 하나 이상 존재한다고 가정한다. 호 e를 통해 점 $HEAD(e)$에 들어와 호 g를 따라 $HEAD(g)$로 갈 수 있으면 $[e, g]$를 가능고리(admissible hook)라 하자. 또한 호 e와 호 g간의 교차로 지연시간을 $p(e, g)$라 하자. 즉, $TAIL(e) \rightarrow HEAD(e)$ $(= TAIL(g)) \rightarrow HEAD(g)$에 걸리는 교차로 지연시간이 $p(e, g)$이다. 호 e의 이웃한 호의 집합 $ArcAdj(e)=$

$\{g \mid [e,g]$는 가능고리$\}$로 정의하고 $ArcAdj^{-1}(e) = \{g \mid e \in ArcAdj(g)\}$로 정의한다.

알고리즘

단계 0. 초기화

모든 호 g에 대해,

$$d(g) = \begin{cases} c(g), & \text{if } (s = TAIL(g), HEAD(g)) \in A \\ \infty, & \text{otherwise} \end{cases}$$

$pred(g) = 0$, if $(s = TAIL(g), HEAD(g)) \in A$

로 둔다.

$S \leftarrow \emptyset$, $\overline{S} \leftarrow \{1,2,\cdots,m\}$.

단계 1. 호 선택

$h = \arg\min_{e \in \overline{S}} d(e)$인 호 h를 구한다.

단계 2. 거리 꼬리표 수정

$S \leftarrow S \cup \{h\}$, $\overline{S} \leftarrow \overline{S} - \{h\}$.

만약 $HEAD(h) = t$이면, 종료하고,

아니면

$e \in ArcAdj(h)$인 모든 호 e에 대해,

만약 $d(e) > d(h) + c(e) + p(h,e)$이면,

$d(e) = d(h) + c(e) + p(h,e)$,

$pred(e) = h$

로 둔다.

단계 1로 간다.

출발점 s에서 호 h를 거쳐 점 k까지 가는 최단경로를 $SR_h(s \rightarrow k)$로 표시하자. $d(e)$은 교차로 제약을 만족하며 점 s에서 호 e를 거쳐 $HEAD(e)$로 가는 발견된 경로 중에서 최단경로의 길이를 가지게 된다. 또한, $e \in S$이면 이미 $SR_e(s \rightarrow HEAD(e))$가 발견된 것을 의미하고 $e \in \overline{S}$이면 아직까지 호 e를 거쳐 $HEAD(e)$까지 가는 최단경로가 결정되지 않은 호들의 집합이다. 여기서 $pred(e)$는 경로에서 호 e의 바로 앞 호를 가리킨다. $pred(e)$를 이용하면 최단경로를 쉽게 구성할 수 있다.

정리 1. 제안한 알고리즘의 정당성

제안된 알고리즘은 많아야 m회의 호 선택단계 수행 안에 s에서 t로 가는 최단경로를 찾는다.

PROOF) Ahuja([3])의 Dijkstra 알고리즘의 증명을 약간 수정하여 제안된 알고리즘의 정당성(correctness)을 증명할 수 있다. 수학적 귀납법을 사용하여 임의의 $r = |S|$에 대해 다음의 2가지 사실이 성립함을 증명하면 된다.

i) $a \in S$인 모든 호 a는 $SR_a(s \rightarrow HEAD(a))$가 구해져 있고 최단경로의 길이는 $d(a)$이다.

ii) $a \in \overline{S}$인 모든 호 a의 $d(a)$는 S에 속하는 호와 호 a를 지나 $HEAD(a)$로 가는 최단경로의 길이이다.

$r = 1$일 때는 호의 거리가 비음이므로 i)은 자명하게 성립하고 거리 꼬리표 수정 단계에서 선택된 호에서 갈 수 있는 호들의 거리 꼬리표를 수정하므로 ii)도 명백히 성립한다. 임의의 r에서 먼저 i)이 성립함을 보인다. 호 h가 호 선택 단계에서 선택되었다고 하자. i)를 증명하기 위해 $d(h)$가 $SR_h(s \rightarrow HEAD(h))$의 길이임을 보인다. 귀납적 가정에 의해 $d(h)$은 S에 속하는 호들과 호 h만을 지나 $HEAD(h)$로 가는 최단경로의 길이이다. $\overline{S}$에 속하는 호를 한 개라도 거쳐 호 h를 지나 $HEAD(h)$로 가는 경로 P의 길이가 $d(h)$보다 작지 않음을 보이자. P를 두 개의 부분 경로 P_1과 P_2로 나

누되 P_1는 최소한 하나 이상의 S에 속하는 호들만으로 구성되고 P_1의 마지막 호 l가 $l \in \overline{S}$를 만족하게 하자. 귀납적 가정에 의해 $d(l) \geq d(h)$이고 호의 길이는 비음이므로 P_2의 길이는 0 이상이다. 따라서 P의 길이는 $d(h)$보다 크거나 같게 된다. 즉, $d(h)$는 $SR_h(s \rightarrow HEAD(h))$의 길이이다.

다음으로 ii)가 성립함을 보이자. 호 h가 선택되어 단계 2에서 S에 추가되면 $\overline{S}-\{h\}$에 있는 호 중에서 호 h와 가능고리를 이루는 호들의 거리 꼬리표를 수정해 주므로 $\overline{S}-\{h\}$에 있는 호들의 거리 꼬리표는 귀납적 가정 ii)를 만족시킨다. ■

Fibonacci 힙(heap)을 사용하여 제안된 알고리즘을 구현하는 것을 생각해 보자. Fibonacci 힙(heap)에 대한 자세한 설명은 [3]을 참조하기 바란다. 먼저 각 호에 대해 거리 꼬리표가 하나씩 존재하므로 총 m개의 거리 꼬리표를 유지해야 한다. 또한 호 e의 거리 꼬리표 $d(e)$은 최악의 경우 $|ArcAdj^{-1}(e)|$회만큼 수행된다. 따라서 아래와 같이 거리 꼬리표의 총수정회수는 mn보다 작다.

$$\begin{aligned}\sum_{e \in A} ArcAdj^{-1}(e) &= \sum_{e \in A} ArcAdj(e) \\ &\leq \sum_{e \in A} n \\ &= mn\end{aligned}$$

이다.

정리 2. 제시한 알고리즘의 복잡도

제안한 알고리즘의 Fibonacci 힙 구현의 복잡도는 $O(mn + m\log m)$이다.

PROOF) Ahuja et al.([3])의 Dijkstra 방법의 Fibonacci 힙 구현의 복잡도 분석을 이용하자. Fibonacci 힙에서 최소의 거리 꼬리표를 제거하는 데에 요구되는 시간은 $O(\log m)$이다. 따라서 최소의 거리 꼬리표를 가진 호를 제거하는데 걸리는 총시간은 $O(m\log m)$이다. 또한 거리 꼬리표를 한번 수정하는데 걸리는 시간은 $O(1)$이고 거리 꼬리표의 총수정회수는 $O(mn)$이므로 거리 꼬리표 수정에 소요되는 전체 연산 시간은 $O(mn)$이다. 따라서 제안된 알고리즘의 복잡도는 이를 모두 더한 $O(mn + m\log m)$이 된다. ■

교차로 제약이 없는 경우 Fibonacci 힙을 이용한 Dijkstra 방법은 $O(m + n\log n)$으로서 최소의 거리 꼬리표를 제거하는데 총 $O(n\log n)$연산이 걸리고 거리 꼬리표를 수정하는데 총 $O(m)$연산이 소요된다. 이는 교차로 제약이 추가됨으로써 복잡도가 $O(n)$배정도 증가함을 의미한다.

교차로 제약이 없는 보통 네트워크에서 꼬리표 수정방법 중에서 가장 우수한 것으로 알려진 FIFO 꼬리표 수정 방법(FIFO label correcting algorithm)의 복잡도는 $O(mn)$이다([3]). 이는 최단경로의 길이가 많아야 $(n-1)$보다 작거나 같다는 사실을 이용한다. 그러나 교차로가 주어지면 최단경로의 길이는 최악의 경우 $(m-1)$가 되므로 Ziliaskopoulos([15])가 제안한 알고리즘의 복잡도는 최악의 경우 $O(m^2)$이 된다. 이는 제안된 알고리즘의 복잡도 $O(mn + m\log m)$보다 큼을 알 수 있다.

제시된 알고리즘으로 [그림 1]의 점 ①에서 출발하여 점 ⑤로 가는 최단경로를 구해 보자.

각 호의 번호는 [그림 5]와 같고 [그림 2]의 교차로 지연시간을 고쳐 쓰면 [그림 6]과 같다.

호 번호	출발점	도착점
1	①	②
2	①	③
3	②	③
4	③	④
5	③	⑤
6	④	⑤
7	⑤	⑥

[그림 5] 각 호의 번호

교차로	지연 시간
①→②→③	p(1,3)=0
①→③→④	p(2,4)=2
①→③→⑤	p(2,5)=2
②→③→④	p(3,4)=1
③→④→⑤	p(4,6)=1
④→⑤→③	p(6,7)=1
⑤→③→④	p(7,4)=0
③→⑤→③	p(5,7)=0
⑤→③→⑤	p(7,5)=0

[그림 6] 교차로 지연시간

단계 0. 초기화

$d(1)=1,\ d(2)=4,\ d(3)=d(4)=d(5)=d(6)=d(7)=\infty$.

$pred(1)=0,\ pred(2)=0,\ pred(3)=pred(4)=0,$

$pred(5)=pred(6)=pred(7)=0$.

$S \leftarrow \varnothing,\ \overline{S} \leftarrow \{1,2,3,4,5,6,7\}$.

단계 1. 호 선택

$1=argmin\{d(1),d(2),d(3),d(4),d(5),d(6),d(7)\}$.

단계 2. 거리 꼬리표 수정

$S \leftarrow \{1\},\ \overline{S} \leftarrow \{2,3,4,5,6,7\}$.

$d(3) \leftarrow d(1)+c(1)+p(1,3)=2$.

$pred(3) \leftarrow 1$.

단계 1. 호 선택

$3=argmin\{d(2),\ d(3),\ d(4),\ d(5),\ d(6),\ d(7)\}$.

단계 2. 거리 꼬리표 수정

$S \leftarrow \{1,3\},\ \overline{S} \leftarrow \{2,4,5,6,7\}$.

$d(4) \leftarrow d(2)+c(4)+p(2,4)=5$.

$pred(4) \leftarrow 2$.

단계 1. 호 선택

$2 = argmin\{d(2),\ d(4),\ d(5),\ d(6),\ d(7)\}$.

단계 2. 거리 꼬리표 수정

$S \leftarrow \{1,2,3\},\ \overline{S} \leftarrow \{4,5,6,7\}$.

$d(5) \leftarrow d(2)+c(5)+p(2,5)=7$.

$pred(5) \leftarrow 2$

단계 1. 호 선택

$4 = argmin\{d(4),\ d(5),\ d(6),\ d(7)\}$.

단계 2. 거리 꼬리표 수정

$S \leftarrow \{1,2,3,4\},\ \overline{S} \leftarrow \{5,6,7\}$

$d(6) \leftarrow d(4)+c(6)+p(4,6)=8$.

$pred(6) \leftarrow 4$.

단계 1. 호 선택

$5 = argmin\{d(5),\ d(6),\ d(7)\}$.

단계 2. 거리 꼬리표 수정

$S \leftarrow \{1,2,3,4,5\},\ \overline{S} \leftarrow \{6,7\}$.

종료.

따라서 점 ①에서 점 ⑤로 가는 최단경로는 ①→③→⑤이고 교차로 회전 지연시간을 합쳐 최단경로의 길이는 7이다.

4. 구현과 실험 결과

Ziliaskopoulos([15])은 EFSS(Extended Forward Star Structure)를 이용하여 교차로의 지연 시간을 저장하였고 교차로 회전 금지는 매우 큰 지연시간을 주어 표현하였다. 본 연구에서는 다음과 같은 LFSF(Link Forward Star Form)을 제안한다. 이 방법은 교차로 회전 금지를 매우 큰 지연시간을 주는 방법을 사용하여 명시적으로 표현하지 않아도 된다.

Arc No	Tail	Head	Travel Time	Adjacent Arc	Turn Penalties
1	①	②	1	3	0
2	①	③	4	4	2
3	②	③	1	5	2
4	③	④	2	4	1
5	③	⑤	1	6	1
6	④	⑤	2	7	0
7	⑤	③	1	7	1
				4	0
				5	0

[그림 7] LFSF의 예

LFSF에서는 각 호별로 호의 도착점과 출발점을 저장하고 호의 소요시간과 이웃호들이 저장된 배열의 시작주소를 가리키는 배열이 사용된다. 물론 Tail[]과 Head[]를 저장하지 않아도 가능하지만 이해를 위해 같이 적어 두었다.

예를 들어 [그림 1]과 같은 교차로 제약이 있는 도로네트워크를 LFSF를 사용하여 표현하면 [그림 7]과 같다.

제안한 알고리즘에서는 각 호의 이웃한 호를 찾아야 하는데 LFSF는 이웃한 호가 연속적으로 배열되어 있으므로 쉽게 이웃한 호집합을 찾을 수 있으나 EFSS에서는 이웃한 호를 찾기 위해서 여러 번의 부가적인 연산이 요구된다.

본 연구에서 제안한 알고리즘을 구현하여

문제 이름	점수	호수	교차로 제약이 없는 경우			교차로 제약이 있는 경우		
			최단경로의 길이	FIFO 수행시간	Dijkstra 수행시간	최단경로의 길이	Ziliaskopoulos 알고리즘	제안한 알고리즘
SPGRID3	661	1980	1266	1	0	1384	1	0
SPGRID5	1036	3105	1730	1	0	1730	1	0
SPGRID10	2001	6000	1877	1	0	1966	3	0
SPRAND9	1789	3837	20544	1	0	20544	2	0
SPRAND10	2000	8999	15664	1	1	39188	2	1
SPRAND11	2789	10000	29965	1	1	4229	4	1
SPRAND15	10007	40007	31174	12	7	64928	24	9
SPRAND17	15708	60987	28313	24	10	76400	48	37
SPRAND20	20000	90000	25887	36	20	178475	48	43
SPRAND21	20001	80004	26156	32	13	45669	61	20
SPRAND25	100000	400000	36222	249	181	76927	359	64
SPRAND26	120000	480000	31265	287	141	85087	447	345
SPRAND27	139998	500284	39667	306	221	69515	539	254
SPRAND28	173883	754843	33302	495	354	80946	544	147
SPRAND29	190000	902744	18907	671	42	127075	426	121
SPRAND30	195000	499000	66835	280	353	69489	661	377
Changwon	4075	8977	1885.58	1	1	2706.76	2	1

[그림 8] 실험 결과

Ziliaskopoulos가 제안한 알고리즘([15])과 그 수행도를 비교하였다([그림 8]). 본 연구에서 사용한 자료구조는 LFSF를 사용하였으며 Ziliaskopoulos가 제안한 알고리즘은 EFSS를 사용하였다. 교차로 제약이 없는 경우 FIFO 꼬리표 수정 알고리즘(label correcting algorithm)과 라딕스 힙(radix heap)를 이용한 Dijkstra 방법을 사용하였다. 교차로 제약이 있는 경우도 마찬가지로 FIFO 꼬리표수정 알고리즘(label correcting algorithm)을 수정하여 Ziliaskopoulos가 제안한 방법을 구현하였고 본 연구에서 제안한 방법은 라딕스 힙(radix heap)를 사용하여 구현하였다. 실험 기종은 SUN Ultra 170이고 시간은 단위 1/100초 단위로 측정한 것이다.

[그림 8]에서 알 수 있듯이 본 연구에서 제안한 방법이 Ziliaskopoulos가 구현한 방법에 비해 약 2~3배 정도 빠름을 알 수 있다.

5. 결 론

본 연구에서는 도로네트워크에서 보다 현실적인 경로를 선정하기 위한 모형으로 교차로 제약과 교차로 회전 시간이 주어지는 경로 선정 문제에서 최단경로를 찾는 방법을 제안하였다. 본 연구에서 제안한 방법은 꼬리표 설정 방법으로 기존의 교차로를 고려한 꼬리표 수정 방법보다 이론적인 복잡도뿐만 아니라 구현을 통한 비교실험에서도 우수한 것으로 나타났다.

추후 과제로는 도로의 혼잡과 교차로 제약을 동시에 고려하는 보다 현실적인 문제로 본 연구의 결과를 확장하는 것이다.

참 고 문 헌

[1] 박순달, OR(경영과학), 민영사, 1996.

[2] 이명석, 박순달, "다수개의 여행시간이 주어진 경우의 지정된 마디간의 최단경로문제", 「경영과학」, 제7권 2호(1990.12), pp.51-57.

[3] Ahuja, R. K., T. L. Magnanti and J. B. Orlin, *Network flows*, Prentice-Hall Inc., 1993.

[4] Caldwell, T., "On finding minimal routes in a network with turning penalties", *Comm. ACM* 4(1961), pp.107-108.

[5] Cooke, K. L. and E. Halsey, "The shortest route through a network with time-dependent inter-nodal transit times", *J. of Math. Anal. Appl.* 14(1996), pp.493-498.

[6] Easa, S. M., "Shortest route algorithm with movement prohibitions", *Transportation Research* 19B (1985), pp.197-208.

[7] Easa, S. M., "Traffic assignment in Pratice : Overview and guidelines for users", *Transportation Engng* 117 (1991), pp.231-243.

[8] Florian, M. and S. Nguyen, "Recent experiments with equilibrium methods for the study of a congested urban area", Int. Symp. Traffic Equilibrium Methods, Montreal, Canada, 1975.

[9] Hall, R. W., "The fastest path through a network with random time-dependent travel times", *Transportation Science* 20 (1986), pp.182-188.

[10] Kaufman, D. E. and R. R. Smith, "Fastest paths in time-dependent networks for IVHS application", *IVHS Journal* 1(1993), pp.1-12.

[11] Kirby, R. F. and R. B. Potts, "The minimum route problem for networks with turn penalties and prohibitions", *Transportation Res.* 3(1969), pp.397-408.

[12] Orda, A. and R. Rom, "Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length", *J. of the ACM* 37(1990), pp.607-625.

[13] Yagar, S., "Dynamic traffic assignment by individual path minimiztion and queuing", *Transportation Res.* 5(1971), pp.179-196.

[14] Ziliaskopoulos, A. K. and H. S. Mahmassani, "Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications", *Transportation Res. Rec.* 1408(1993), pp.94-100.

[15] Ziliaskopoulos, A. K. and H. S. Mahmassani, "A note on least time path computation considering delay and prohibitions for intersection movements", *Transportation Res.* 30B(1996), pp.359-367.