Two-Queue를 이용한 One-to-One 최단경로 알고리즘

심충섭, 김진석
서울시립대학교 컴퓨터 · 통계학과
{cssim95, jskim}@venus.uos.ac.kr

An One-to-One Shortest Path Algorithm using Two-Queue

Chungsup Sim and Jin Suk Kim
Dept. of Computer Science & Statistics, University of Seoul

요 약

최단경로 탐색에 있어서 출발지와 목적지 사이의 최단경로를 계산하는데 있어서 Label-Setting 알고리즘이 Label-Correcting 알고리즘보다 낫다고 믿어왔다. 하지만 특수한 경우에는 Label-Correcting 알고리즘이 GIS기반의 도로에서 더 좋은 결과를 보인다고 Benjamin의 논문에서 밝혔다[1]. 본 논문에서는 Label-Correcting 알고리즘인 Pallottino의 Graph Growth 알고리즘을 수정하여 One-to-One 최단경로탐색에 적합한 알고리즘을 제안한다.

1. 서 론

최단경로 탐색 알고리즘(Shortest Path Finding Algorithm)은 교통 및 네트워크 분석관련 문제에서 매우 중요하며 알고리즘을 계산하는데는 많은 시간이 소요된다. 이와 같은 최단경로 탐색 문제는 1950년대 Ford, Bellman, Moore, Dijkstra에 의해 최단경로 탐색이 개발되어졌다. 그 이외에도 많은 알고리즘들이 제안되어 왔다[2].

최단경로 알고리즘은 Label-Setting 알고리즘과 Label-Correcting 알고리즘으로 이루어져 있으며 두 알고리즘은 One-to-All 최단경로를 계산하는데 Labeling 방법을 사용한다[3]. Label-Setting 알고리즘과 Label-Correcting 알고리즘의 차이점은 다음과 같다. Label-Setting 알고리즘의 최단경로는 출발지에서 도착지에 이르기까지 도착지가 찾아지고 영구적인 Labeling이 되었을 때 결정된다. 반면 Label-Correcting 알고리즘은 출발지에서 도착지에 이르는 마지막 단계에 임시로 표지된 모든 노드들을 평가하여 최단경로를 결정한다. 이러한 특징으로 인해 Label-Setting 방식이 One-to-One 최단경로탐색에 더 적합하다고 알려져 있었다[2,3,4,5].

Benjamin의 논문에서는 Label-Setting 알고리즘 중에서 가장 빠른 approximate bucket을 이용한 Dijkstra(DIKBA)알고리즘과 Label-Correcting 알고리즘 중에서 가장 빠른 알고리즘인 Pallottino의 Graph Growth(Two-Queue)를 비교하여 큰 도로 네트워크에서는 Two-Queue 알고리즘이 빠르다는 것을 실험으로 보였다[1].

본 논문에서는 One-to-One 최단경로 탐색시 사용할 수 있는 수정된 Two-Q 알고리즘을 제안한다.

2. Graph Growth(Two-Queue) 알고리즘

이 장에서는 Pallottino(1984)에 의해 소개된 두 개의 Queue를 사용한 Graph Growth Algorithm에 대해서 살펴보고자 한다. 알고리즘은 <그림 1>과 같다[4].

Two-Queue알고리즘은 Queue에서 하나의 노드를 가져온 후 그 노드에서 갈 수 있는 모든 노드로의 길을 찾는다. 찾아진 노드로의 경로 값이 기존의 최단거리 값보다 더 좋은 결과 값이 나오면 기존의 최단거리 값을 수정하고 수정된 노드를 Queue에 삽입한다. Queue안에 노드가 없을 때까지 반복한다. Two-Queue알고리즘은 시작노드에서 갈 수 있는 노드로의 길을 모두 찾기 때문에 One-to-All 최단경로 탐색시 사용되는 알

고리즘이다.

```
Procedure ShortestPathTreeConstruction(s)
begin
  Queue_Initialization(Q);
  for i=1 to n do
    d(i) = infinite;
  d(s) = 0;
  while (Q ≠ Null) do
  begin
    Queue_Removal(Q, i);
    for each successor node j of node i do
      if d(j) > d(i) + l(i,j) then
      begin
        d(j) = d(i) + l(i,j);
        Queue_Insertion(Q,j);
      end
  end
end
```

그림 1. Two-Q Algorithm

위 알고리즘은 두 개의 Queue를 사용하는데 Queue의 구조는 <그림 2>와 같다[4].

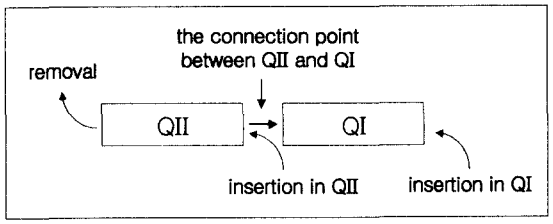

그림 2. Two-Queue의 구조

Queue를 삽입할 때는 다음과 같은 방법으로 QI나 QII에 삽입한다.

- 만일 Queue에 한번도 들어갔었던 적이 없었던 노드는 QI에 삽입한다.
- 만일 Queue에 들어갔었던 적이 있었던 노드는 QII 삽입한다.

3. Two-Q에서 개선된 알고리즘

Label-Setting 방식 중 가장 빠른 approximate bucket을 이용한 Dijkstra(DIKBA)알고리즘과 Label-Correcting 방식 중 가장 빠른 Two-Queue를 이용한 Graph Growth 알고리즘을 비교하였다[1]. 실험을 통해 큰 도로 네트워크에서는 Two-Queue가 DIKBA알고리즘 보다 더 빠른 실행시간을 보여주었고 Two-Queue가 Label-Correcting방식이기 때문에 One-to-One 최단경로 탐색시 One-to-All 최단경로탐색 시간과 같다고 하였다[1]. 하지만, 본 논문에서는 목적지 노드를 찾고서 목적지 노드까지 걸린 시간을 Threshold값으로 주고 Threshold값까지만 노드를 확장하는 방법을 채택하여 전체 모든 노드를 검색하지 않고서도 최단거리를 검색할 수 있도록 하였다. 본 논문에서는 <그림 1>의 알고리즘을 수정하여 One-to-One에 적용하였으며 수정된 Two-Q 알고리즘은 <그림 3>과 같다. <그림 3>에서 음영부분이 수정된 부분이다.

```
Procedure ShortestPathTreeConstruction(s)
begin
  Queue_Initialization(Q);
  for i=1 to n do
    d(i) = infinite;
  d(s) = 0;
  while (Q ≠ Null) do
  begin
    Queue_Removal(Q, i);
    for each successor node j of node i do
      if d(j) > d(i) + l(i,j) then
      begin
        d(j) = d(i) + l(i,j);
        if d(j) < Threshold then
          if d(j) == 목적지노드 then
            Threshold = d(j);
          else
            Queue_Insertion(Q,j);
      end
  end
end
```

그림 3. 수정된 Two-Q 알고리즘

Two-Q 알고리즘의 특징은 시작노드를 기준으로 확장해 나가는 것이다. 그러므로 Two-Q 알고리즘에서 잘못된 곳으로의 확장을 막으면 실행 시간을 줄일 수 있을 것이다. 또한 GIS 도로는 실제 지도상에 위치해 있기 때문에 특수한 경우를 제외하고 대부분의 최단경로탐색은 시작노드에서 목적노드 방향으로 확장해 나간다. 본 논문에서는 시작노드에서 목적노드까지의 절대경로값을 읽어와서 일정한 값 이상으로의 확장을 막아 최단경로 탐색시간을 줄일 수 있는 그래프의 확장을 막는 알고리즘을 제안한다. 그래프의 확장을 막는 알고리즘은 <그림 4>와 같다. <그림 3>에서 음영부분이 수정된 부분이다.

```
Procedure ShortestPathTreeConstruction(s)
begin
  .....
        if d(j) < Threshold then
          if d(j) == 목적지노드 then
            Threshold = d(j);
          else
            if 유효한 노드 then
              Queue_Insertion(Q,j);
  .....
end
```

그림 4. 그래프의 확장을 막는 알고리즘

4. 실험 및 평가

수정된 Two-Q 알고리즘 및 그래프의 확장을 줄여주는 알고리즘을 테스트 하기 위해 본 논문에서는 경주지도를 사용하였다. 사용된 경주지도의 노드 수는 7506개이며, 노드가 연결된 링크의 수는 18936개이다. 수정된 Two-Q알고리즘은 C언어로 구성되어있다[5]. 실험은 PIII-600Mhz processor와 128MB RAM, RedHat Linux 6.2를 사용하였다. 실험은 7506개의 노드 중 임의로 100개의 노드를 선택하고 선택된 노드에서 각각 100개의 노드를 임의로 선택하여 10000번의 최단경로 탐색 시간의 평균을 구하였다. Two_Queue 알고리즘을 사용한 것과 본 논문에서 제안한 수정된 Two_Queue알고리즘의 실행시간을 비교한 표가 <표 1>에 나타나 있다.

표 1. Two_q와 수정된 Two_q 실행시간

알고리즘	Two-Q	수정된 Two-Q
시간(㎲)	7101.11	4418.88

<표 1>의 결과에서도 알 수 있듯이 One-to-One 최단경로탐색인 경우에는 약 62%의 수행시간을 절약하였다.

다음으로 그래프의 확산을 막기위해 본 논문에서 제안된 수정된 Two-Q 알고리즘에 절대거리의 비율을 계산하여 계산 값 이상으로 그래프가 확산되는 것을 막는 알고리즘의 실행결과는 <표 2>에 나타나 있다.

표 2. 그래프 확산 방지 알고리즘 실행시간

절대거리 비율	실행시간(㎲)	error 발생률(%)
1 / 1	5704.38	0.0017
1 / 2	4183.26	0.0051
1 / 3	3981.99	0.0156
1 / 4	3763.10	0.0347
1 / 5	3664.08	0.0528
1 / 6	3576.04	0.0695
1 / 7	3460.82	0.0885
1 / 8	3372.76	0.1028
1 / 9	3330.69	0.1170
1 / 10	3246.14	0.1306

<표 3>에 있는 절대거리비율은 시작노드와 목적노드사이의 절대거리를 구해서 그 값의 비율을 말한다. 절대거리 비율은 그래프 확산 방지 지역을 설정해서 그 설정 이외로 확장하는 그래프를 막아준다. 실행시간의 단위는 ㎲(10^{-6}초)이다. error 발생률은 최단거리를 찾지 못하는 비율을 말한다.

<표 2>의 결과 값을 보면 절대거리의 비율이 작을수록 실행시간을 줄어들지만, error의 발생률이 급격히 증가함을 알 수 있을 것이다.

5. 결 론

본 논문에서 제안된 수정된 Two-Q 알고리즘은 큰 GIS 도로 데이터의 One-to-One 최단경로탐색에서 효율적으로 사용할 수 있다. 그리고 완벽한 최단경로탐색이 아닌 빠른 경로탐색알고리즘을 원하는 경우에는 그래프 확산 방지 알고리즘을 사용하면 수행시간을 줄일 수 있다.

참 고 문 헌

[1] F. B. Zhan and C. E. Noon, "A Comparison Between Label-Setting and Label-Correcting Algorithms for Computing One-to-One Shortest Paths," Journal of GIDA, 2000.

[2] 노정현, 남궁성, "도시가로망에 적합한 최단경로탐색 기법의 개발," Journal of KPA, Vol. 30, No. 5, 1995.

[3] 최기주, "U-Turn을 포함한 가로망 표현 및 최단경로의 구현," Journal of KTRS, Vol. 13, No3, 1995.

[4] F. B. Zhan, "Three Fastest Shortest Path Algorithms on Real Road Networks: Data Structures and Procedures," Journal of GIDA, 1997.

[5] B. V. Cherkassky, A. V. Goldberg and T. Radzik, "Shortest Paths algorithms: Theory and experimental evaluation," Mathematical Programming, 73, 1996.