2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

자바를 이용한 그래프 검색 알고리즘의 시각화

°정연진, 전상현, 김은규, 이광모, 최홍식
한림대학교 컴퓨터공학과
e-mail : yjjung@hallym.ac.kr

Visualization of Graph Search Algorithm using Java

Yeon-jin Jung Sang-hyun Cheon, Eun-kyu Kim,
Kwang-mo Lee, Hong-sik Choi

Dept. of Computer Engineering, Hallym University

요약

최단경로문제(Shortest Path Problem)는 네트웍에서 하나 혹은 그 이상의 노드들의 쌍 사이에서 가장 짧은 경로, 가장 저렴한 경로 또는 가장 신뢰할 만한 경로를 찾을 때 고려된다. 컴퓨터나 통신망들은 edge-weighted 그래프로 대치될 수 있으며 그렇게 함으로써 최단 경로를 찾아줄 수 있다. 통신 링크는 실제 실패할 수도 있고, 또한 전송될 데이터의 양에 따라 전달되는 시간이 달라지기도 하므로, 가장 신뢰할만한 경로 중에서 가장 빠른 경로(The Quickest Most Reliable Path) 문제와 가장 빠른 경로 중에서 가장 신뢰할만한 경로(The Most Reliable Quickest Path) 문제는 최단경로문제보다 더 현실적이다[1]. 이 논문에서는 그 중 '가장 신뢰할만한 경로 중에서 가장 빠른 경로' 문제를 자바를 사용하여 시각화함으로써 가변 상황에 따라 다른 경로를 찾아주는 과정을 보여준다.

1. 서론

최단경로문제(Shortest Path Problem)는 네트웍에서 하나 혹은 그 이상의 노드들의 쌍 사이에서 가장 짧은 경로, 가장 저렴한 경로 또는 가장 신뢰할 만한 경로를 찾을 때 고려된다. 컴퓨터나 통신망들은 edge-weighted 그래프로 대치될 수 있으며 그렇게 함으로써 최단 경로를 찾아줄 수 있다. 통신망에서 그것은 중요한 응용이므로 여러 곳에서 많은 관심을 받았다.

통신 링크는 실제 실패할 수도 있고, 또한 전송될 데이터의 양에 따라 전달되는 시간이 달라지기도 하므로, 가장 신뢰할만한 경로 중에서 가장 빠른 경로 문제와 가장 빠른 경로 중에서 가장 신뢰할만한 경로 문제는 최단경로문제보다 더 현실적이다.

이에 대하여, Chen과 Chin은 가장 빠른 경로 문제(*quickest path problem*)라고 불리는 새로운 최단 경로문제를 제시했다. 그것의 기본적인 방법은 하나의 출발 노드에서 목적지 노드로 주어진 일정량의 데이터를 보내기 위한 가장 빠른 경로를 찾는 것이다[2]. 이것을 발전시킨 형태가 G. Xue가 제시한 가장 신뢰할만한 경로 중 가장 빠른 경로와 가장 빠른 경로 중 가장 신뢰할만한 경로인데[1], 본 논문에서는 이것의 알고리즘을 자바를 이용하여 시각화함으로써 선택 사항을 달리 해줌에 따라 달라지는 경로들을 살펴보았다.

컴퓨터나 통신망은 edge-weighted 그래프 N=(V, E, c, l, p)으로 모형화 시킬 수 있다.

여기서, G(V, E)는 다중 에지와 self-loop가 없는 무방향 그래프, V는 vertex 집합, E는 m edge의 집합으로, N에 있는 노드와 에지들은 각각 컴퓨터와

커뮤니케이션으로 해석된다. c(u, v)≥0으로 edge {u, v}의 용량(capacity), l(u, v)≥0으로 edge {u, v}의 지연(delay)으로 통과 시간을 의미하고, p(u, v) ∈[0, 1]로 edge {u, v}의 선택적인 확률로 신뢰도를 뜻한다. 다시 말해, c(u, v)는 노드 u에서부터 v까지 시간 당 전송될 수 있는 데이터의 최대양을 나타내며, l(u, v)은 노드 u에서부터 노드 v까지 데이터를 보내는데 요구되는 유도(lead) 시간을 나타낸다. 달리 규정이 없는 한, c와 l은 정수 함수로 가정하고, 데이터 σ의 양과 전송 시간은 정수 변수로 가정한다.

에지 {u, v}를 통해 노드 u에서 노드 v로 데이터 σ unit을 보내기 위해 걸리는 최소한의 전송 시간은,

$$l(u, v) + \left\lceil \frac{\sigma}{c(u,v)} \right\rceil \quad \cdots\cdots\cdots (1)$$

만큼 요구된다.

경로 π=(v_1, v_2, ..., v_k)일 때, v_1-v_k를 따라 v_1에서부터 v_k로 가는 데이터 σ unit을 전송하기 위한 최소한의 전송 시간은,

$$T(\pi,\ \sigma)= \sum_{i=1}^{k-1} l(v_i,v_{i+1})+ \left\lceil \frac{\sigma}{\min_{i=1}^{k-1}\{c(v_i,v_{i+1})\}} \right\rceil \quad \cdots\cdots\cdots (2)$$

만큼 요구된다.

에지 {u, v}를 통한 전송이 성공할 확률은 p(u, v)이다.

경로 π의 신뢰도는 $R(\pi) = \prod_{i=1}^{k-1} p(v_i,v_{i+1})$ 모든 ρ (v_i, v_{i+1}) 의 곱으로 나타낸다 [3].

그림 1은 간단한 네트웍의 예이다.

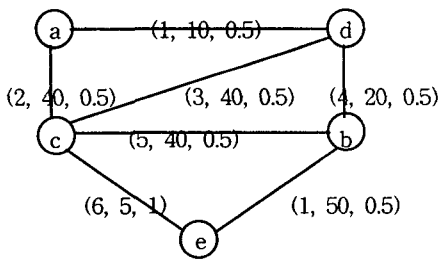

그림 1. 각 에지에 (l, c, p) 값을 갖는 네트웍

각 에지에는 (l, c, p)가 주어지고 이것들을 통해 값을 산출해낸다.

출발점을 a로 보고 도착점을 e로 봤을 때, a에서 e로 갈 수 있는 모든 경로 π를 구하고, 전송될 데이터의 양을 변화시켜가면서 식(1), (2)를 통해 전송 시간 T(π, σ)를 계산하면 다음과 같다.

a에서 e까지 경로 (a, d, b, e)를 통해 데이터를 10unit 전송하려면, 최소 시간은 (2)에 의해 7이 걸린다. 이것은 a에서 e로 10unit을 이동하기 위한 가장 빠른 유일한 것이다. 그러나, 데이터의 양이 40unit일 경우에는 그와 같은 경로로는 10의 시간이 걸린다. 그 때, (a, c, b, e)를 이용하면 전송 시간은 9가 된다. 이렇게 전송할 데이터의 양이 변화함에 따라 다른 결과가 나오는 것을 [표 1]을 통해 알 수 있다.

[표1] a에서 e로 가는 모든 경로들의 전송시간 및 확률

π \ σ	T(π,10)	T(π,20)	T(π,40)	T(π,60)	T(π,100)	R(π)
(a,d,c,b,e)	12	13	15	17	21	0.0625
(a,d,c,e)	13	15	19	23	31	0.25
(a,d,b,c,e)	18	20	24	28	36	0.125
(a,d,b,e)	**7**	**8**	10	12	16	**0.125**
(a,c,d,b,c)	11	11	12	13	15	0.0625
(a,c,b,e)	9	9	**9**	**10**	11	**0.125**

2. 가장 신뢰할만한 경로 중에서 가장 빠른 경로 (The Quickest Most Reliable Path)

T(π, σ)가 가장 신뢰할만한 a-e 경로 중에서 최소값이 되면, a에서 e로 데이터 σ unit을 전송할 수 있는 '가장 신뢰할만한 경로 중에서 가장 빠른 경로'라고 한다.[Xue]

p(u, v) = 0이면, 에지 {u, v}의 비용 cost(u, v) = cost(v, u) =∞이고, 그렇지 않은 경우 $\log \frac{1}{p(u,v)}$ 라고 정의한다. 그렇게 되면 새로 정의된 비용 함수를 써서 가장 믿을만한 경로를 계산하는 것은 가장 짧은 경로를 계산하는 것과 같아진다.

Xue는 [1]에서 SPN(Shortest Path Network)을 구하여 가장 신뢰할만한 경로 중 가장 빠른 경로를 계산하는 알고리즘을 제안하였다. 이 논문에서는 그 알고리즘 부분을 적용하여 자바로 시각화함으로써 단계별 그래프 검색이 가능하도록 하였다. SPN의 arc(u, v)에 대해 {u, v}는 N의 에지이다. 따라서, arc(u, v)의 지연(delay)와 용량(capacity)은 각각

l(u, v), c(u, v)로 자연스레 정의 될 수 있다. SPN은 N의 가장 믿을만한 모든 s-t 경로들의 집합이므로, s-t로 데이터 σ unit을 전송하는 SPN 내의 빠른 s-t 경로는 가장 빠르면서 믿을만한 s-t 경로이다.

다음은 SPN을 구하는 알고리즘이다.

```
for (int pi = 0 ; pi < dist.length; pi++)
{
  p_dist[pi] = graph.getLogProbability(s, pi);
  if (pi != s)
    p_parent[pi] = s;
}
p_check[s] = 1;
p_checked++;
while (p_checked < graph.size)
{
  int x = 0;
  while (p_check[x] != 0)
    x++;
  for (int i = 0; i < graph.size; i++)
  {
    if (p_check[i] == 0 && p_dist[i] <  p_dist[x])
    {
      x = i;
    }
  }
  p_check[x] = 1;
  p_checked++;
  for (int i = 0; i < graph.size; i++)                {
    if (x == i || graph.getProbability (x, i)
       >= graph.nothing || p_check[i] != 0)
    {
      continue;
    }
    double p_d = p_dist[x] + graph.getLogProbability(x, i);
    if (p_dist[x] >= graph.nothing)
      System.out.println ("Disconnected");
    if (p_d < p_dist[i])
    {
      p_dist[i] = p_d;
      p_parent[i] = x;
    }
  }
}
```

그림 2는 이 과정을 시각화한 것이다[4, 5]. 즉 임의의 노드를 찍은 후, 해당하는 (l, c, p)값을 주어 출발지에서 목적지까지 경로를 찾아내도록 한다. 필요에 따라서는 단계별로 변화를 추적해가면서도 볼 수 있다.

이 알고리즘을 구현한 환경은 JDK1.3의 Swing을 사용하였으며, 4개의 패널로 구성되고 그 역할은 다음과 같다.

- Input Box Panel : 사용자의 입력을 처리 Draw패널에 값을 전달.
- Draw Panel : 실제적인 알고리즘의 수행 과정을 표현.
- Menu Panel : 수행을 원하는 버튼들의 집합으로 다양한 출력을 제어.
- Documentation Panel : 수행 중에는 과정에서의 결과를 표시하며, 수행에 필요한 정보를 가지고 있다.

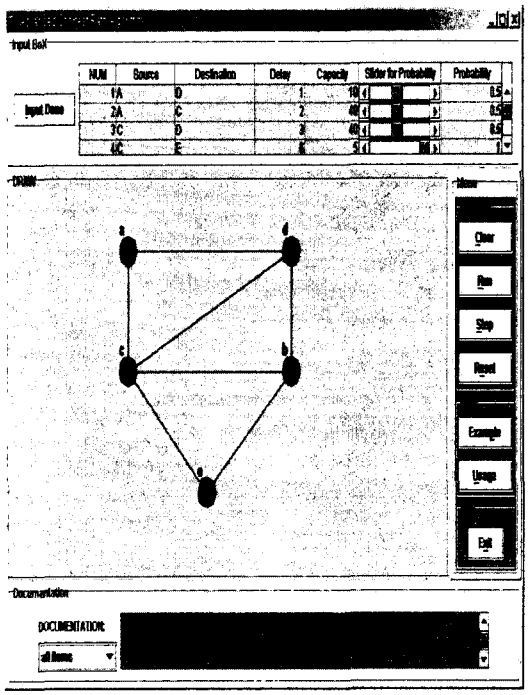

그림 2 가장 신뢰할만한 경로 중 가장 빠른 경로

사용자는 마우스 버튼을 누름으로서 노드의 위치를 정할 수 있다. 노드의 개수는 Table에게 전달되고, Source와 Destination의 아이템은 그 개수만큼만을 표시하도록 하여 사용자의 오류를 막았으며, 사용자가 edge를 제거하기를 원할 경우를 대비하여 NONE을 두었다. NONE이 선택되면, 두 노드의 edge는 제거된다. Delay와 Capacity는 셀의 편집으로 변경할 수 있다. 확률은 현재의 구현에서는 스크롤을 사용하였다. 그러나, 정확한 확률을 편집하기에는 좀 어려움이 따르므로 좀 더 편리하도록 하기 위해서 확률의 값을 다음의 칼럼에 수치로 나타나도록 하였다. 모든 입력이 완료되면 Input Done버튼을 누름으로서 Draw에 정보를 전달한다. 전달된 정보에 의해 노드와 노드사이의 edge가 연결되고 사용자는 Run 버튼을 누름으로서 전체적인 실행을 할 수 있고, 또한 Step버튼을 누르면 각각의 수행이 단계별로 진행되도록 할 수 있다. 두 가지의 실행 상태는 스레드로 구현하여 시간의 흐름을 조절하였다. 수행

단계에서의 결과는 Documentation Panel에 보여진다. 스레드를 이용함으로써 얻을 수 있는 이점은 사용자가 계속적인 수행을 원치 않을 경우에는 수행중에도 멈출 수가 있다는 것이다. 어플리케이션과 애플릿이 함께 작동 되도록 만들면 웹으로도 접근할 수 있으나, 명확한 코드를 보여주지는 못하므로 두 가지 버전을 따로 구현하였다. 이렇게 함으로써 좀 더 명확한 구조를 보여준다.

3. 결론 및 향후 과제

이 논문에서는 가장 신뢰할만한 경로 중 가장 빠른 경로를 구하는 과정을 시각화함으로써 용량이나 지연시간, 확률에 따라 다양한 경로의 변화를 볼 수 있도록 하였다. 이 그래프 문제는 통신망 설계 분야에 응용이 가능한데 그것은 공급되는 회선의 종류가 다르므로 어떤 회선을 선택하느냐에 따라 비용이 달라질 수 있기 때문이다. 이 시각화된 프로그램을 이용함으로써 가장 짧은 경로만이 아닌 신뢰도가 높으면서 빠른 경로를 찾아줄 수 있고 그 과정을 눈으로 확인 할 수 있다. 다음의 구현에서는 에지 위에 확률을 두어 변경을 쉽게 함으로서 수행을 할 수 있는 환경을 만들 것이다. 또한 직렬화를 이용한 저장 기능과 불러오기 기능을 추가를 통해 다른 피어에서도 사용되어질 수 있도록 할 것이다. 앞으로는 가장 빠른 경로 중에서 가장 신뢰할만한 경로 문제를 시각화하고자하고 그래프 이론의 일반적인 부분으로 연구하고자 한다.

참고문헌

[1] Guoliang Xue, "End-to-End Data Paths: Quickest or Most Reliable?", IEEE Communications letters, Vol.2, No.6, June 1998.

[2] Y.L.Chen and Y.H.Chin, "The Quickest Path Porblem", Computers Opns. Res., Vol.17, No.2, pp.153-161, 1990.

[3] J.B. Rosen, S.Z. Sun and G.L. Xue, "Algorithms for The Quickest Path Problem and The Enumeration of Quickest Paths", Computers Ops Res. Vol.18, No.6, pp579-584, 1991.

[4] Gilles Brassard and Paul Bratley, "Algorithmics Theory and Practice", Prentice Hall, pp.87-92, 1988.

[5] Dijkstra's Shortest Path Algorithm Animation in Java, from original code by Carla Laffra, http://www.cs.uwa.edu.au/undergraduate/courses/230.300/readings/graphapplet/graph.html 1996