〈硏究論文〉

Journal of the Korean Institute of Industrial Engineers
Vol. 1 No. 1 June 1975

最短經路 Algorithm의 計算能率 比較

(Efficiency of Shortest-route Algorithms)

鄭 秀 一*

Abstract

This paper studies the efficiency of four algorithms in determining the shortest-route length between two specified nodes of a network in which every pair of nodes is connected by two nonnegative length arcs. The efficiency is measured in terms of number of additions and comparisons in computation of the shortest-route length. Also, each algorithm is programmed on the IBM 1130 for solving for example problems, and the computing time is measured for further efficiency comparisons.

I. 緖 論

Network 문제 중 最長經路問題는 建築·土木工事, 大單位裝備의 維持, 新製品의 硏究開發 등의 복잡한 作業活動에 대한 計劃, 日程의 作成 및 管理에 PERT 또는 CPM 등의 技法으로 활발히 應用되고 있다. 그러나 이와는 正反對의 槪念을 가진 最短經路問題는 裝備의 交替, 投資計劃, 旅程 또는 set-up問題에 있어서의 經費 및 時間의 최소화 등의 廣範한 應用可能性이 있음에도 불구하고 그 活用이 매우 지지부진하여 앞으로 이에 대한 많은 應用開發이 期待된다.

最短經路問題는 그 應用目的에 따라 다음과 같이 5가지 形態로 分類할 수가 있다.

1) 주어진 network에 있어서의 特定된 2 nodes 사이의 最短經路 決定
2) 可能한 모든 雙의 nodes 사이의 最短經路 決定
3) 1) 및 2)에 있어서의 2번째, 3번째, …… 등의 最短經路 決定
4) 주어진 network을 通한 出發時間에 따르는 最短旅行時間의 決定
5) 特定된 中間의 nodes를 通過하는 2 end-points 사이의 最短經路 決定

以上 5가지 形態의 問題에 대해 各各의 最短經路를 決定하기 위한 많은 algorithm이 發表되어 있고, 또한 個個의 algorithm은 特別한 構造의 network 問題에 가장 適切하게 利用될 수 있도록 開發된 것이다. 그러나 本稿에서는 모든 可能한 雙의 2 nodes 사이가 반드시 nonnegative distance를 갖는 한 雙의 arc (때로는 ∞의 距離를 갖는)로 連結되어 있는 network에 대해 特定된 2 nodes 사이의 最短經路를 決定하는 問題에 局限하여 4個의 algorithm에 대한 計算의 能率性을 檢討·比較하고자 한다.

* 仁荷大學校 工科大學 產業工學科

Ⅱ. Algorithms

n 個의 nodes 로 構成된 network 에 있어서의 特定된 2 nodes 사이의 最短經路를 決定하는 問題는 特定된 2 nodes 중의 出發點을 node 1, 終着點을 node n 이라 하고 餘他의 nodes 에 2 에서 n-1 까지의 番號를 아무렇게나 붙였을 때, node 1 에서 node n 에 이르는 最短經路를 찾는 것과 同一한 問題가 된다. node i 에서 node j 로 連結되는 方向을 가진 arc 의 길이를 d_{ij} 라 하고, $d_{ij} \geq 0$ 및 $d_{ii}=0$ 를 假定하면 $d_{ij}=d_{ji}$ 및 $d_{ij}<d_{ik}+d_{kj}$ 는 一般的으로 成立되지 않는다. 萬一 node i 에서 node j 로 直接 連結되는 arc 가 없는 境遇에는 $d_{ij}=\infty$ 로 한다.

$d_{ij} \geq 0$ 의 條件下에서 node 1 에서 node n, 혹은 node 1 에서 node $j(j=2, 3, \cdots\cdots, n)$에 이르는 最短經路를 찾는 問題에 대해서는 Dijkstra 를 비롯하여 Minty-Ford-Fulkerson, Whiting-Hillier-Danzig, Berge-Ghouila-Houri, Danzig, Nicholson-Murchland, Ford-Moore-Bellman, Yen 및 Danzig-Blattner-Rao 등의 algorithm 이 發表되어 있고, Dreyfus[1]는 이들 algorithm 의 計算能率을 最短經路의 距離를 計算하는 데에 所要되는 最大限度의 加減算 및 數値의 比較(additions and comparisons) 回數를 基準으로 잡아 이를 檢討하여 比較的 能率이 좋은 5 個의 algorithm 에 대해 表 1 과 같은 結果를 얻었다.

그러나 좀더 仔細히 考察하면 表 1 의 數値는 加減될 수 있고 Minty-Ford-Fulkerson 및 Yen algorithm 등의 境遇에는 加減算 및 數値

Table 1 The efficiency of the algorithms

n : number of nodes

Algorithms	Number of additions	Number of comparisions
Dijkatra	$n^2/2$	$2n^2$
M-F-F*	$n^3/6$	$n^3/6$
W-H-D*	$n^2/2$	$n^2/2+n^2\log_2 n+3n$
F-M-B*	n^3	n^3
Yen	$n^3/4$	$n^3/4$

M-F-F*: Minty-Ford-Fulkerson
W-H-D*: Whiting-Hillier-Danzig
F-M-B*: Ford-Moore-Bellman

의 比較 回數가 network 의 構造特性에 따라 크게 左右될 것으로 생각된다. 그래서 現在까지 確認할 수 있었던 Whiting-Hillier-Danzig 를 除外한 4 個의 algorithm 에 대해 筆算時의 加減算 및 數値의 比較 回數에 依한 計算의 能率性을 比較하고 그 回數가 network 의 特性에 影響을 받는 境遇에는 컴퓨터에 依한 例題의 計算時間으로 그 能率性을 檢討해 보고자 한다.

4 個의 algorithm 을 要約하면 各各 다음과 같다.

1. Dijkstra algorithm[1], [2]

먼저 node 1 에 確定値 0 을, 그리고 다른 모든 nodes 에 臨時値 ∞를 附與한다. 다음, node 1 을 除外한 n-1 個의 nodes 하나하나에 대해 node 1 로부터 直接 個個의 node 에 이르는 距離와 이미 各 node 에 附與된 臨時値를 比較하여 그 중 적은 값을 各該當 node 에 대한 새로운 臨時値로 한다. 이렇게 求한 n-1個의 nodes 에 대한 臨時値 중의 최소치를 그 該當 node 의 確定値로 한다.

node k 가 確定値를 갖게 되었다고 假定하자. 다음에는 아직 臨時値를 가진 n-2 個의 nodes 에 대해 위에서 求한 node k 의 確定値와 node k 에서 直接 各 node 에 이르는 距離의 合計와, 各 node 의 臨時値를 比較하여 그 중 적은 값을 各 該當 node 의 새로운 臨時値로 한다. 이렇게 求한 n-2 個의 nodes 에 대한 臨時値 중의 최소치를 該當 node 의 確定値로 定하고, 이 確定値 및 該當 node 를 다음의 確定値 및 그 確定値를 갖게 될 node 를 定하는 데에 利用한다.

이러한 過程을 反復하여 node n 이 確定値를 갖게 되면 그 確定値가 곧 node 1 에서 node n 에 이르는 最短經路의 距離가 된다. 따라서 node 1 에서 node n 에 이르는 最短距離만이 必要한 境遇에는 위의 過程은 많아서 n-1番이 必要할 것이고, 萬一 node 1 에서 다른 모든 nodes 에 이르는 最短距離가 各各 必要한 境遇에는 꼭 n-1 番이 必要하게 될 것이다.

2. Minty-Ford-Fulkerson algorithm[3], [4], [5]

Step 1. v_j 를 node 1 에서 node j 에 이르는

最短經路의 距離라 하고, 또한 $i=j$일 때 $w_i=v_j$로 놓으면, $v_1=w_1=0$이 되고, 이로부터

$v_j=\min_{i<j}[d_{ij}+w_i]$ ……………(1)

의 式을 利用하여 v_j 및 $w_i(i,j=2,3,\cdots\cdots,n)$의 臨時値를 求한다.

Step 2. $i=1$로 하여

a) $j=1,2,\cdots\cdots,n$에 대해 v_j-w_i를 計算하여 b) 및 c)를 檢討한다.

b) 萬一 모든 j에 대해 $d_{ij}\geq v_j-w_i$이고, 또한 $i=n$이면 d)로 간다. $i<n$이면 i를 1 增加시켜 a)로 돌아간다.

c) 萬一 $d_{ij}<v_j-w_i$인 j가 있으면 이에 該當하는 v_j의 새로운 값 v'_j를

$v'_j=d_{ij}+w_i$

의 式으로 求하고 $v_j=w_j=v'_j$로 變更한다. $i=n$이면 d)로 가고, $i<n$이면 i를 1 增加시켜 a)로 돌아간다.

d) c)에서 v_j 및 w_i의 값이 變更되지 않았으면 이 때의 v_j는 node 1에서 node j에 이르는 最短經路의 距離가 된다. 萬一 v_j 및 w_i의 값이 c)에서 變更되었으면 $i=1$로 하여 Step 2를 反復한다.

3. Ford-Moore-Bellman algorithm〔1〕,〔6〕

$f_i^{(0)}=d_{1i}$

및

$f_i^{(k+1)}=\min_j[d_{ji}+f_j^{(k)}],\quad k=0,1,2,\cdots$ …………………………(2)

로부터 $f_i^{(k+1)}$의 값을 求하면 $f_i^{(k+1)}\leq f_i^{(k)}$이 成立된다. $f_i^{(k+1)}=f_i^{(k)}(i=1,2\cdots\cdots,n)$이면 $f_i^{(k)}$는 node 1에서 node i에 이르는 最短經路의 距離가 된다. (2)式으로부터 $f_1^{(k)}=0$은 모든 k값에 대해서 成立한다.

4. Yen algorithm〔1〕

Ford-Moore-Bellman algorithm에서의 (2)式을 다음의 (3) 및 (4)式으로 代替한 것이다. 즉 $k=1,2,\cdots\cdots$에 대해서

$f_i^{(2k-1)}=\min\ [\min_{j<i}(d_{ji}+f_j^{(2k-1)}),$
$f_i^{(2k-2)}],\quad (i=1,2,\cdots\cdots,n)$ …(3)

및

$f_i^{(2k)}=\min[\min_{j>i}(d_{ji}+f_j^{(2k)}),\ f_i^{(2k-1)}],$
$(i=n,n-1,\cdots\cdots,1)$ ………(4)

node 1에서 node $j(j=2,3,\cdots,n)$에 이르는 最短經路는 各 algorithm을 利用하여 求한 最短經路의 距離로부터 쉽게 찾을 수 있을 것이다.

Ⅲ. 各 algorithm의 計算能率

前述한 各 algorithm을 利用하여 node 1에서 node $j(j=2,3,\cdots\cdots,n)$에 이르는 最短經路의 距離를 筆算하는 데에 所要되는 加減算 및 數値의 比較 回數를 檢討하여 各 algorithm의 計算能率을 考察하여 보면 다음과 같다.

1. Dijkstra algorithm의 境遇

j-1 個의 nodes가 確定値를 갖게 되었다고 假定했을 때, j번째의 確定値를 갖는 node를 決定하기 위해서는

1) $n-j+1$ 個의 nodes에 대한 臨時値의 變更與否를 決定하기 위해서 $n-j+1$ 回의 덧셈$(j=3,4,\cdots\cdots,n)$ 및 數値의 比較$(j=2,3,\cdots\cdots,n)$가,

2) $n-j+1$ 個의 臨時値 중의 최소치를 찾기 위해서 $n-j$ 回의 數値의 比較$(j=2,3,\cdots\cdots,n-1)$가

各各 必要하다.

따라서 node 1에서 다른 모든 nodes에 이르는 最短經路의 距離를 計算하기 위해서는

$$\textstyle\sum_{j=3}^{n}(n-j+1)=\sum_{i=1}^{n-2}i=(n^2-3n+2)/2$$

回의 덧셈과,

$$\textstyle\sum_{j=2}^{n}(n-j+1)+\sum_{j=2}^{n-1}(n-j)$$
$$\textstyle=\sum_{i=1}^{n-1}i+\sum_{i=1}^{n-2}i$$
$$=n^2-2n+1$$

回의 數値의 比較가 必要하다.

前述한 algorithms의 內容에서 알 수 있는 바와 같이 node 1에서 node n에 이르는 最短經路의 距離를 求하는 境遇에 Minty-Ford-Fulkerson, Ford-Moore-Bellman 및 Yen algorithm에서는 node 1에서 다른 모든 nodes에 이르는 最短經路의 距離를 求해야 하는 反面, 이 Dijkstra algorithm에서는 特殊한 境遇를 除外하고는 그럴 必要가 없다. 그러므로 node 1에서 node n에 이르는 最短經路의 距離만이 必要한 境遇에는 앞에서 求해 본 덧셈 및 數値의 比較 回數는 훨씬 減少될 수도 있을 것

이다. 그러나 同一한 條件下에서 各 algorithm 의 計算能率을 比較하기 위해서 앞에서 求한 回數를 그대로 利用하고자 한다.

2. Minty-Ford-Fulkerson algorithm 의 境遇

初期條件으로서의 $v_1=w_1=0$ 은 不變値가 되므로 (1)式에서 特定의 j 에 대한 v_j 의 臨時値를 求하기 위해서는 $j-2(j=3,4,\cdots\cdots,n)$回의 덧셈 및 數値의 比較가 必要하다(이 때 $v_2=d_{12}$ 임). 그러므로 Step 1 에서는

$$\textstyle\sum_{j=3}^{n}(j-2)=\sum_{i=1}^{n-2}i=(n-2)(n-1)/2$$

回의 덧셈 및 數値의 比較가 各各 必要하다.

Step 2 에서는 特定의 i 에 대해

a) 에서 $n-1$ 回의 뺄셈,

b) 및 c)에서 $n-1$ 回의 數値의 比較,

c)에서 若干의 덧셈

이 必要하다. 따라서 c)에서의 덧셈을 除外하면 Step 2 를 한번 履行하는 데에는 大略 $n(n-1)$回의 뺄셈 및 數値의 比較가 必要하다.

그런데 어떠한 構造의 network 에 있어서도 Step 2 는 반드시 한번 以上 履行되어야 하므로, Minty-Ford-Fulkerson algorithm 은 Dijkstra algorithm 보다 그 計算能率이 뒤진다.

3. Ford-Moore-Bellman algorithm 의 境遇

$f_i^{(0)}=d_{1i}$ 및 $f_1^{(k)}=0$ 이 주어지면 特定의 k 및 i 에 대해 (2)式으로부터 $f_i^{(k)}$의 값을 求하기 위해서는 $n-2$ 回의 덧셈과 $n-1$ 回의 數値의 比較가 必要하다. 따라서 特定의 $k(i=2,3,\cdots\cdots n)$에 대한 $f_i^{(k)}$의 값을 얻기 위해서는 $(n-1)(n-2)$回의 덧셈과 $(n-1)^2$回의 數値의 比較가 必要하다. 그런데 $f_i^{(k)}$의 값은 node 1 에서 node i 까지의, $k+1$ 또는 그 以下의 arcs 로 構成된 經路의, 臨時最短距離에 該當하는 값이다. 그러므로 k 의 可能한 最大限度의 값은 $k=n-2$ 또는 $k+1=n-1$ 이다. 따라서 最大限度의 덧셈 및 數値의 比較 回數는 各各 $(n-2)(n-1)^2$ 및 $(n-1)^3$이 된다.

또한 $f_i^{(k+1)}=f_i^{(k)}(i=2,3,\cdots\cdots,n)$의 成立 與否를 確認하기 위해서 特定의 k 에 대해 $n-1$ 回의 數値의 比較가 必要하므로 最大限度$(n-1)^2$回의 數値의 比較가 履行되어야 한다.

그러므로 Ford-Moore-Bellman algorithm 에 있어서의 最大限度의 덧셈 및 數値의 比較 回數는 各各 n^3-4n^2+5n-2 및 n^3-2n^2+n이 된다.

위에서 檢討해 본 바에 依하면 (2)式을 利用하여 $f_i^{(1)}$을 求하고 $f_i^{(1)}=f_i^{(0)}$의 成立與否를 確認하기 위해서만도 n^2-3n+2 및 n^2-n回의 덧셈 및 數値의 比較가 必要하다. 따라서 Ford-Moore-Bellman algorithm 의 計算能率도 Dijkstra algorithm 의 그것보다 못하다.

4. Yen algorithm 의 境遇

이 algorithm 에서도 恒常 $f_1^{(k)}=0$ 이 成立하므로 (3)式에서 特定의 k 와 i 에 대해 $i-2(i=3,4,\cdots\cdots,n)$回의 덧셈 및 $i-1(i=2,3,\cdots\cdots,n)$回의 數値의 比較가 所要되므로 特定의 k에 대해서는

$$\textstyle\sum_{i=3}^{n}(i-2)=\sum_{j=1}^{n-2}j=(n-2)(n-1)/2$$

回의 덧셈과

$$\textstyle\sum_{i=2}^{n}(i-1)=\sum_{j=1}^{n-1}j=n(n-1)/2$$

回의 數値의 比較가 必要하다.

(4)式에서는 特定의 k 및 i 에 대해 $n-i(i=n-1,n-2,\cdots\cdots,2)$回의 덧셈 및 數値의 比較가 必要하므로

$$\textstyle\sum_{i=2}^{n-1}(n-i)=\sum_{j=1}^{n-2}j=(n-2)(n-1)/2$$

回의 덧셈 및 數値의 比較가 各各 必要하다.

또한 $f_i^{(2k+1)}=f_i^{(2k)}$의 成立與否를 確認하기 위해서 $n-1$ 回의 數値의 比較가 必要하므로 特定의 k 에 대한 Yen algorithm 에 있어서의 덧셈 및 數値의 比較 回數는 各各 n^2-3n+2 및 n^2-n 이 된다.

이는 同一條件下에서의 Ford-Moore-Bellman algorithm 에 있어서의 덧셈 및 數値의 比較 回數와 같지만 Yen algorithm 의 境遇에는 $f_i^{(2k+1)}=f_i^{(2k)}$ 代身 $f_i^{(l)}=f_i^{(l-1)}(l=1,2,\cdots\cdots)$의 成立與否를 (3) 및 (4)式의 計算 直後에 各各 確認할 수도 있고, 또한 最短經路의 距離의 確定値를 얻게 되는 k 의 값(iteration 의 數)이 前者보다 작을 것이다[1].

따라서 Yen algorithm 이 Ford-Moore-Bellman algorithm 보다 能率的이다. 그러나 Yen algorithm 亦是 Dijkstra algorithm 보다는 非能率的이다.

以上에서 Dijkstra algorithm 의 計算能率이

가장 좋고, Yen algorithm 이 Ford-Moore-Bellman algorithm 보다 能率的이라는 것은 알 수 있지만 餘他의 比較에 대해서는 一般的인 結論을 얻기가 어려우므로, 4個의 例題를 컴퓨터로 計算하여 그 計算時間의 長短을 測定하여 이를 檢討해 보고자 한다.

Ⅳ. 컴퓨터에 依한 各 algorithm의 計算能率 比較

各各 適當한 數의 arcs로 連結되어 있는 10, 25, 50 및 75個의 nodes를 가진 network에 대해 亂數表를 利用하여 整數의 d_{ij}를 個個의 arc에 附與하여 4個의 例題를 만들어 前述한 各 algorithm을 programming하여 IBM 1130型 컴퓨터로 run시켰다.

各 program은 데이터를 읽어 들이는 部分, node 1에서 node j에 이르는 最短經路의 距離를 計算하는 部分 및 各各의 最短經路의 距離 및 經路를 찾아 프린트하는 部分의 3部分으로 構成되고, 첫째 및 셋째 部分은 各 program에서 同一하게 되도록 만들었다. Running time은 CPU에 附着되어 있는 時計를 利用하여 0.0005hr 單位까지, 컴퓨터의 狀態變動을 우려하여 같은 날 午後에 測定하였다. 이 때 各 algorithm의 計算能率에 對應하는 둘째 部分은 그 時間 測定의 精度를 높이기 위해 同一한 計算을 5~100回 反復시키고, 이 部分의 前後에 PAUSE statement를 써서 時間을 測定한 後 그 平均値를 取하였다. 이것을 各各 秒單位로 換算한 것이 表 2이다. 表에서 괄호 속에 든 數値는 各 program 全體에 대한 running time을 參考로 나타낸 것이다.

Table 2 Computing time of the shortest-route length from node 1 to node j, by IBM 1130

(time unit: sec.)

Algorithms	10 nodes	25 nodes	50 nodes	75 nodes
Dijkstra	0.13(63.13)*	0.82(72.82)	3.38(91.58)	7.56(119.16)
M-F-F*	0.29(65.08)	2.38(74.38)	14.76(104.76)	26.64(140.04)
F-M-B*	0.59(61.79)	7.78(76.18)	50.40(138.60)	128.70(240.30)
Yen	0.23(72.23)	1.87(82.87)	9.54(110.34)	15.84(140.04)

M-F-F* : Minty-Ford-Fulkerson
F-M-B* : Ford-Moore-Bellman
(   )* : Running time for the whole program

Ⅴ. 結 論

表 1, 表 2 및 Ⅲ에서의 結果를 綜合하면 $d_{ij} \geq 0$인 network에 있어서의 特定된 2 nodes 사이의 最短經路의 距離를 計算하는 데 있어서는 Dijkstra algorithm이 가장 能率的이었고 다음이 Yen, Minty-Ford-Fulkerson 및 Ford-Moore-Bellman algorithm의 順으로 나타났다. 그러나 緖論에서의 나머지 4가지 形態의 最短經路의 決定 問題에 대해서는 이들의 順序가 아직 未知數이다. 이에 대한 考察은 다음 機會로 미루면서 보다 積極的인 最短經路 問題에 대한 應用開發을 期待해 본다.

參 考 文 獻

〔1〕 Dreyfus, S.E., "An Appraisal of Some Shortest-path Algorithms", *Opns. Res.*, Vol.17, No.3, pp.395-412, May-June, 1969

〔2〕 Hu, T.C., *Integer Programming and Network Flows*, Ch. 10, Addison-Wesley Publishing Co., Reading, Massachusetts, 1969

〔3〕 Gass, S.I., *Linear Programming, Methods and Applications*, 3rd ed., Ch. 10, McGraw-Hill Book Co., New York, 1969

〔4〕 Taha, H.A., *Operations Research, An Introduction*, Ch.5, The Macmillan Co., New York, 1971

〔5〕 Wagner, H.M., *Principles of Operations Research with Applications to Managerial Decisions*, Ch.7, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1969

〔6〕 Nemhauser, G.L., *Introduction to Dynamic Programming*, Ch.4, John Wiley and Sons, Inc., New York, 1966.