Journal of the Korean Institute
of Industrial Engineers
Vol. 19, No.2, June., 1993.

K-最大容量經路 計算法에 관한 硏究

A Study on Algorithms for Calculating the k-Maximum Capacity Paths in a Network

金炳秀* · 金忠英**

Byung-Su Kim* · Chung-Young Kim**

Abstract

Methods for calculating k shortest paths in a network system, are based on a analogy which exists between the solution of a network problem and traditional techniques for solving linear equations.

This paper modifies an algebraic structure of the K shortest path method and develops k maximum flow methods. On the basis of both theoretical and algebraic structure, three iteration methods are developed and the effective procedure of each method are provided. Finally, computational complexity is discussed for those methods.

1. 序 論

네트웍分野에서 決定的네트웍(deterministic network)은 네트웍內에서 모든 값이 固定的으로 주어져 있는 네트웍으로서 이의 主要問題중에는 最短經路(shortest path)問題, 最大흐름(maximum flow)問題, 最小費用흐름(minimum cost flow)問題 等이 있다[2].

最短經路問題에 대하여 1969年度에 Dreyfus[5]가 그때까지 發表된 最短經路技法들에 대하여 綜合的으로 評價하여 發表한 일이 있었고, 그후 1971年度에 Carré[3]는 네트웍模型에 代數的構造(algebraic structure)를 導入시켜 最短經路問題를 포함한 一般的인 네트웍經路問題를 解決하였으며, 1976年度에 Shier[8, 9]는 Carré의 方法을 擴張하여 複數個의 最短經路를 구하는 問題인 k-最短經路(k-shortest paths)問題에도 適用할 수 있음을 보였다.

한편 最大흐름問題에 대하여서는 1962年度에 Ford 및 Fulkerson[6]에 의하여 標識方法(labeling method)이 發表된 이래 많은 技法들이 開發되었으며 1986年度에 Tarjan[10]이 그때까지 發表된 여러가지 最大흐름技法들을 槪括的으로 紹介하였으며, 현재까지 새로운 技法들이 계속 發表되고 있는 實情이다.

本 硏究에서는 Shier의 k-最短經路技法에 導入된 代數的構造를 最大容量經路問題에 適用될 수 있도록 變形시켜서 네트웍에서 k-最大容量經路(複數個의 最大容量經路)를 算出하는

* 安企部

** 國防大學院

計算法(algorithm)을 開發한다.

本 硏究에 대한 接近方法은 네트웍에 代數的 構造(algebraic structure)와 一般的演算(generalized operations)을 導入한 다음, 一次聯立方程式 問題의 古典的技法인 Jacobi方法, Gauss-Seidel方法 및 double-sweep方法 等의 反復法을 變形시켜 一般的 Jacobi方法, 一般的 Gauss-Seidel方法 및 一般的 double-sweep方法 等으로 擴張한다. 이의 基本槪念은 k-最大容量經路의 最初推定벡터를 이용하여 다음 推定벡터를 算出하며 다시 이 推定벡터로서 그 다음 推定벡터를 算出하는 過程을 最適이 될 때까지 反復하는 것이다.

本 硏究에서 다루는 네트웍은 弧上에 흐름(flow)이 있는 네트웍으로서 陰의 값을 가진 흐름이 없고 自體環(loop)이 없는 네트웍에 限定하여 硏究한다.

2. k-最大容量經路에 관한 理論的 考察

2.1 理論的 背景

네트웍(network)은 마디(node)의 集合과 마디를 연결하는 弧(arc)의 集合으로 이루어진다. 마디의 集合을 N이라 하고 弧의 集合을 A라 하면 네트웍 G는 G=(N, A)로 表示되며 마디 i에서 마디 j로 향하는 弧(i, j)에는 最大限度로 흐를 수 있는 容量(capacity)이 정해져 있다고 假定한다. 네트웍에서 i_0, i_1, …, i_m을 마디라 할 때(i_0, i_1), (i_1, i_2), …, (i_{m-1}, i_m)가 順序的으로 弧로 연결되어 있다면, [(i_0, i_1), (i_1, i_2), …, (i_{m-1}, i_m)]을 源마디(source)i_0에서 着마디(sink)i_m까지 이르는 經路(path)라 하고 經路上에 있는 弧의 갯수를 그 經路의 크기(size)라 한다. 따라서 i_0에서 i_m까지 이르는 經路(path)는 크기가 m이다. 그리고 經路上에 나타난 모든 마디가 相異한 經路를 基本經路(elementary path)라 한다.

源마디와 着마디가 같은 經路를 回路(circuit)라 하고 그중에서 源마디와 着마디가 같은 것을 제외하고는 모든 마디가 相異한 回路를 基本回路(elementary circuit)라 하며 經路의 크기가 1인 回路를 自體環(loop)이라 한다. 經路容量(path capacity)은 經路上의 모든 弧들의 容量중에서 最小값으로 定義한다. 이는 그 經路上에 흐름이 흐를 수 있는 最大容量을 의미한다.

本 硏究에서는 네트웍내의 特定 두마디 사이에 存在하는 모든 經路들 중에서 經路容量이 가장 큰 順序로 k개의 經路들, 즉, k-最大容量經路(k-maximum capacity paths)를 찾는 問題를 다루며 本 硏究에서의 네트웍은 다음 2가지 制約條件을 만족하는 것으로 假定한다.

(1) 네트웍上의 모든 弧의 容量은 非陰이다.

(2) 네트웍內의 自體環은 包含되어 있지 않다.

本 硏究에서 導出하고자 하는 k-最大容量 計算法은 代數的構造와 演算을 기초로 하기 때문에 이에 관한 基本槪念을 여기에서 먼저 설명한다. 한 네트웍의 마디 s에서 마디 t까지의 4-最大容量을 구하고자 한다면 마디 s에서 마디 t까지의 모든 經路를 구한 다음 經路容量을 크기순으로 羅列하여 제일 큰 4개를 택하면 된다. 선택된 4개의 經路容量을 크기순으로 a_1, a_2, a_3, a_4라고 한다면 4-最大容量은 (a_1, a_2, a_3, a_4)와 같이 벡터로 表示될 수 있으며 大小關係 $a_1 > a_2 > a_3 > a_4$가 成立한다. 만약 값이 서로 다른 經路容量이 2개 밖에 없으면 a_1과 a_2의 값만 存在하므로 여기에서 $a_3=a_4=0$으로 두고 0>0이라고 약속하면 4-最大容量벡터는 (a_1, a_2, a_3, a_4)로 表示되고 $a_1 > a_2 > a_3 > a_4$도 成立한다. 그러므로 一般的으로 k-最大容量의 경우에 있어서도 k-最大容量벡터 (a_1, a_2, …, a_k)에서 항상 大小關係 $a_1>a_2>\cdots>a_k$가 成立됨을 알 수 있다.

마디 s에서 마디 i를 經由하여 마디 t까지 가는 經路의 4-最大容量을 예를 들어 (10, 6, 3, 0)이라 하고 마디 s에서 마디 j를 經由하여 마디 t에 이르는 4-最大容量을 (9, 6, 4, 2)이라 하면 마디 s에서 마디 i나 또는 마디 j를 經由하여 마디 t까지 가는 4-最大容量은 (10, 9,

6, 4)가 된다. 이 벡터는 벡터 (10, 6, 3, 0)과 (9, 6, 4, 2)를 하나로 합친 集合 {10, 6, 3, 0, 9, 6, 4, 2}에서 크기순으로 제일 큰 4개를 선택한 것이다.

이와 같은 演算을 一般的最大化(generalized maximization)라고 부르기로 하며 ⊕로 表示한다. 그러면 앞 演算은 (10, 6, 3, 0)⊕(9, 6, 4, 2)=(10, 9, 6, 4)로 나타낼 수 있다. 또한 마디 s에서 마디 i에 이르는 4-最大容量을 (7, 5, 2, 1)이라 하고 마디 i를 經由하여 마디 t까지 가는 經路는 經路上의 모든 弧의 容量중 最小容量으로만 흐를 수 있으므로 4-最大容量은 (6, 5, 3, 2)가 된다. 이 벡터는 表 2-1에서와 같이 (7, 5, 2, 1)과 (6, 5, 3, 0)을 서로 交叉해 가며 比較하여 작은 값을 택한 集合에서 크기순으로 제일 큰 4개를 선택한 것과 같다.

表 2-1. 一般的最少化 演算

min	6	5	3	0
7	6	5	3	0
5	5	5	3	0
2	2	2	2	0
1	1	1	1	0

이 演算을 一般的最少化(generalized minimization)라고 부르기로 하며 ⊗로 表示한다. 그러면 위 演算은 (7, 5, 2, 1)⊗(6, 5, 3, 0)=(6, 5, 3, 2)로 나타낼 수 있다.

2.2 k-最大容量에 관한 代數的 構造

k-最大容量을 구하기 위한 代數的構造와 一般的演算을 導出하기 위하여 사용되는 記號는 다음과 같이 定義한다.

- R　實數의 集合
- R*　R∪{∞} ; 實數의 集合과 無限大(∞)와의 合集合
- n　네트웍內의 總 마디數
- k　經路容量의 要望數
- $\max_j$　주어진 集合의 元素 중에서 j번째로 큰 元素를 구하는 演算
- C^k　$\{\mathbf{a}=(a_1, a_2, \cdots, a_k) \mid a_i \in R^*, a_1 > a_2 > \cdots > a_k\}$: k개의 元素가 크기순으로 配列된 벡터의 集合(단 0>0으로 約定함).
- M^k　C^k의 元素인 $\mathbf{a}=(a_1, a_2, \cdots, a_k)$形態의 벡터가 n^2개 모여서 이루어진 n×n 行列의 集合

$\mathbf{a}, \mathbf{b}, \mathbf{c} \in C^k$이며 다음과 같다고 한다.

$$\mathbf{a}=(a_1, a_2, \cdots, a_k)$$
$$\mathbf{b}=(b_1, b_2, \cdots, b_k)$$
$$\mathbf{c}=(c_1, c_2, \cdots, c_k)$$

一般的最大化(generalized maximization)

一般的最大化는 ⊕로 表示하고

$$\mathbf{a} \oplus \mathbf{b} = \mathbf{c} \Leftrightarrow \mathbf{c}_j = \max_j\{a_1, \cdots, a_k, b_1, \cdots, b_k\}, \quad j=1, 2, \cdots, k \qquad (2-1)$$

와 같이 定義한다.

一般的最小化(generalized minimization)

一般的最小化는 ⊗로 表示하고

$$\mathbf{a} \otimes \mathbf{b} = \mathbf{c} \Leftrightarrow \mathbf{c}_j = \max_j\{\min(a_i, b_m) \mid i, m=1, \cdots, k\}, \quad j=1, 2, \cdots, k \qquad (2-2)$$

와 같이 定義한다. a와 b에서 各各 하나의 元素를 택하여 그중에서 작은 값을 선택하면 모두 k^2개의 값이 생기는데 여기에서 j번째로 큰 값이 c_j이다.

零벡터를 $\mathbf{v}=(0, 0, \cdots, 0)$ 그리고 單位벡터를 $\mathbf{e}=(\infty, 0, \cdots, 0)$으로 定義하면 一般的最大化와 一般的最少化에는 다음과 같은 性質이 成立함을 알 수 있다. 이러한 代數的構造 (C^k, ⊕, ⊗)를 다오이드(dioid)라고 한다[7].

$$\mathbf{a} \oplus \mathbf{b} = \mathbf{b} \oplus \mathbf{a}, \ \mathbf{a} \otimes \mathbf{b} = \mathbf{b} \otimes \mathbf{a}$$
$$\mathbf{a} \oplus (\mathbf{b} \oplus \mathbf{c}) = (\mathbf{a} \oplus \mathbf{b}) \oplus \mathbf{c},$$
$$\mathbf{a} \otimes (\mathbf{b} \otimes \mathbf{c}) = (\mathbf{a} \otimes \mathbf{b}) \otimes \mathbf{c}$$
$$\mathbf{a} \oplus \mathbf{v} = \mathbf{a}, \ \mathbf{a} \otimes \mathbf{e} = \mathbf{a} \qquad (2-3)$$
$$\mathbf{a} \otimes (\mathbf{b} \oplus \mathbf{c}) = (\mathbf{a} \otimes \mathbf{b}) \oplus (\mathbf{a} \otimes \mathbf{c})$$

$\mathbf{a}\oplus\mathbf{a}=\mathbf{a},\ \mathbf{a}\otimes\mathbf{v}=\mathbf{v}$

이 代數的構造에 다음과 같은 順序關係(order relation)≤를 定義할 수 있다.

$$\mathbf{a}\le\mathbf{b}\Leftrightarrow\mathbf{s}\oplus\mathbf{b}=\mathbf{b} \qquad (2-4)$$

C^k에서 比較計算이 定義되면 $(C^k, \le)$로 表示할 수 있으며 $(C^k, \le)$는 다음과 같이 部分順序集合(partially ordered set)을 이룬다.

$$\begin{aligned} &\mathbf{a}\le\mathbf{a} && \text{(反射率)} \\ &\mathbf{a}\le\mathbf{b},\ \mathbf{b}\le\mathbf{a}\Rightarrow\mathbf{a}=\mathbf{b} && \text{(反對稱律)} \quad (2-5) \\ &\mathbf{a}\le\mathbf{b},\ \mathbf{b}\le\mathbf{c}\Rightarrow\mathbf{a}\le\mathbf{c} && \text{(推移律)} \end{aligned}$$

그리고 다음과 같은 性質도 成立한다.

$$\begin{aligned} &\mathbf{v}\le\mathbf{a} \\ &\mathbf{a}\le\mathbf{a}\oplus\mathbf{b} \\ &\mathbf{a}\le\mathbf{b},\ \mathbf{c}\le\mathbf{d}\Rightarrow\mathbf{a}\oplus\mathbf{c}\le\mathbf{b}\oplus\mathbf{d} \qquad (2-6) \\ &\mathbf{a}\le\mathbf{b},\ \mathbf{c}\le\mathbf{d}\Rightarrow\mathbf{a}\otimes\mathbf{c}\le\mathbf{b}\otimes\mathbf{d} \end{aligned}$$

C^k에서의 두 演算 ⊕와 ⊗를 行列의 加算과 乘算으로 확장시켜 代數的構造$(M^k, \oplus, \otimes)$를 定義할 수 있다. 行列의 모든 元素가 **v**인 零行列을 V로 두고, 行列의 主對角元素가 **e**이고 나머지는 v인 單位行列을 E로 두면 零行列 V와 單位行列 E는 다음과 같이 表示된다.

$$V=\begin{bmatrix} \mathbf{v} & \mathbf{v} & \mathbf{v} & \cdots & \mathbf{v} \\ \mathbf{v} & \mathbf{v} & \mathbf{v} & \cdots & \mathbf{v} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \mathbf{v} & \mathbf{v} & \mathbf{v} & \cdots & \mathbf{v} \end{bmatrix}, \quad E=\begin{bmatrix} \mathbf{e} & \mathbf{v} & \mathbf{v} & \cdots & \mathbf{v} \\ \mathbf{v} & \mathbf{e} & \mathbf{v} & \cdots & \mathbf{v} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \mathbf{v} & \mathbf{v} & \mathbf{v} & \cdots & \mathbf{e} \end{bmatrix}$$

확장된 代數的構造 $(M^k, \oplus, \otimes)$에서도 性質(2-3)과 類似한 性質이 成立함을 알 수 있다.

$A, B, C\in M^k$라고 할 때

$$\begin{aligned} &A\oplus B=B\oplus A \\ &A\oplus(B\oplus C)=(A\oplus B)\oplus C, \\ &A\otimes(B\otimes C)=(A\otimes B)\otimes C \\ &A\oplus V=A,\ A\otimes E=E\otimes A=A \qquad (2-7) \\ &A\otimes(B\oplus C)=(A\otimes B)\oplus(A\otimes C), \\ &(B\oplus C)\otimes A=(B\otimes A)\oplus(C\otimes A) \\ &A\oplus A=A,\ A\otimes V=V\otimes A=V \end{aligned}$$

同一한 方法으로 順序關係도 $(M^k, \le)$에 확장할 수 있다.

$$A\le B \Leftrightarrow A\oplus B=B$$

그러면 性質(2-5) 및 (2-6)이 M^k에서도 成立한다.

$$\begin{aligned} &A\le A \\ &A\le B,\ B\le A\Rightarrow A=B \\ &A\le B,\ B\le C\Rightarrow A\le C \\ &V\le A \qquad (2-8) \\ &A\le A\oplus B \\ &A\le B,\ C\le D\Rightarrow A\oplus C\le B\oplus D \\ &A\le B,\ C\le D\Rightarrow A\otimes C\le B\otimes D \end{aligned}$$

마디가 n개인 네트웍에서 f_{ij}를 마디i에서 마디 j로 향하는 弧의 容量이라고 하면 $a_{ij}=(f_{ij}, 0, \cdots, 0)$로 두어 $n\times n$ 弧容量行列(arc capacity matrix) $A=(a_{ij})\in M^k$를 定義할 수 있다. 또한 $A^0=E$로 定義하고 $j\ge1$일 때 A^j는 A를 j번 反復하여 一般的最少化한 行列이라고 하면

$$A^j=A\otimes A\otimes\cdots\otimes A \qquad \text{(j번 反復)}$$

이 되고, $m\ge0$이라고 하면 $A^{<m>}$과 A^*를 다음과 같이 定義할 수 있다.

$$\begin{aligned} A^{<m>}&=E\oplus A\oplus A^2\oplus\cdots\oplus A^m=\sum_{j=0}^{m} A_j \\ A^* &=E\oplus A\oplus A^2\oplus A^3\oplus\cdots=\sum_{j=0}^{\infty} A_s \end{aligned} \qquad (2-9)$$

行列 A^m의 i行 j列의 元素는 네트웍의 마디 i에서 마디 j까지 이르는 經路중 經路의 크기가 m인 k-最大容量을 갖는 經路임을 의미하고, 行列 $A^{<m>}$의 i行 j列의 元素는 마디 i에서 마디 j로 가는 크기 m 以下인 k-最大容量을 갖는 經路임을 의미한다. 그리고 行列 A^*의 i行 j列의 元素는 經路의 크기와 관계 없이 마디 i에서 마디 j로 가는 모든 經路에 대한 k-最大容量을 의미한다. 여기에서 A^*는 다음과 같은 特性을 지닌다[9].

$$\lim_{m\to\infty} A^{<m>} = A^*$$
$$(E \oplus A)^m = A^{<m>}$$
$$E \oplus A \oplus A^* = A^* \qquad (2-10)$$
$$h \geq 1 \Rightarrow (A^*)^h = A^*$$
$$A \leq B \Rightarrow A^* \leq B^*$$

行列 A^*는 네트웍의 임의의 두마디간에 흐를 수 있는 k-最大容量을 나타낸다. 따라서 네트웍에서 k-最大容量을 구하는 問題는 주어진 弧容量行列 A에 對應하는 行列 A^*를 計算하는 問題로 볼 수 있다.

補助整理1 : 마디가 n개 있는 네트웍의 弧容量行列을 A라고 하면 다음 조건을 만족하는 自然數 w가 반드시 存在한다. 즉, $h \geq w$인 모든 자연수 h에 대해서 $A^* = A^{<h>}$이 성립한다.

(證明)

行列 A^*에서 임의의 s行 t列 元素를 $a_{st} = (c_1, \cdots, c_k)$라고 하고 0보다 큰 經路容量 중에서 임의로 택하여 c_m $(1 \leq m \leq k)$이라고 하자. 그러면 c_m은 네트웍內의 마디 s에서 마디 t에 이르는 m번째로 큰 容量이며 그때의 한 經路를 P이라고 하면 $c_m = \min\{$經路 P上의 各弧의 容量$\}$이므로 經路 P상의 弧 중에서 容量이 c_m인 弧가 存在한다. 그 弧를 (i, j)라고 두면 經路 P는 그림 2-1과 같이 다음 3가지 經路의 合으로 表示된다.

[(s, s′), ⋯, (i′, i)]	經路 P_1,
[(i, j)]	弧 (i, j),
[(j, j′), ⋯, (t′, t)]	經路 P_2

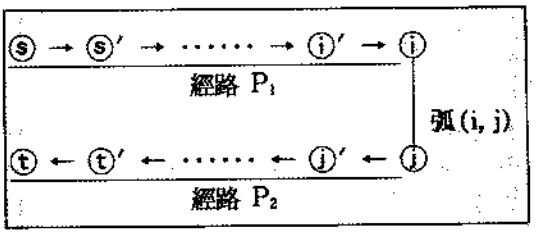

그림 2-1. 經路 P의 分解

經路 P_1과 P_2에서 모든 回路를 除去하면 各各 基本經路 P_3와 P_4를 얻을 수 있다. 여기에서 $P_3[(i, j)]P_4$로 形成되는 새로운 經路를 P′으로 놓으면 P′의 經路容量도 c_m이다.

네트웍에 마디가 n개 있으면 基本經路의 크기(size)는 n 以下이므로 經路 P′의 經路容量도 c_m이다.

네트웍에 마디가 n개 있으면 基本經路의 크기(size)는 以下이므로 經路 P′의 크기(size)는 $(n+1+n) = 2n+1$을 초과할 수 없다. 이는 k-最大容量行列 A^*에서의 임의의 經路容量에 對應하는 經路중 그 크기(size)가 $2n+1$을 초과하지 않는 經路가 반드시 존재함을 의미한다. 그러므로 $w = 2n+1$로 두면 補助定理1은 成立한다.

補助定理1에 의해 A^*의 計算은 $A^{<2n+1>}$을 計算하면 充分하다는 것을 알 수 있으며, 特性 (2-10)에 의하여 $A^{<2n+1>} = (E \oplus A)^{2n+1}$이므로 $A^* = (E \oplus A)^{2^q}$로 表示할 수 있다. 여기에서 $q = \lceil \log_2(2n+1) \rceil$이고 $\lceil a \rceil$는 a보다 같거나 큰 最小整數를 의미한다. 그러므로 行列 A^*는 1회의 行列 덧셈(⊕)과 q회의 行列 곱셈(⊗)으로 구할 수 있다. 이것을 計算法으로 表現하면 다음과 같다.

計算法1 : A^*를 計算한다.

(1) $B = E \oplus A$를 計算한다.
　$q = \lceil \log_2(2n+1) \rceil$로 둔다.

(2) ①-② 過程을 순차적 q회 反復한다.
　① $C = B \otimes B$를 計算한다.
　② $B = C$로 둔다.

(3) $A^* = B$로 두고 끝낸다.

補助定理2 : 回路가 없는 마디 n개인 네트웍의 弧容量行列을 A라 하고 $\mathbf{x} = (x_1, \cdots, x_n)$, $\mathbf{b} = (b_1, \cdots, b_n)$을 M^k에 속하는 行列의 行이라 두면 다음 두 方程式은 같은 解를 가진다[9].

(1) $\mathbf{x} = \mathbf{x} \otimes A \oplus \mathbf{b}$

(2) $\mathbf{x} = \mathbf{b} \otimes A^*$

補助整理3 : M^k에 속하는 임의의 A와 B에 대하여 $(A \oplus B)^* = A^* \otimes (B \otimes A^*)^*$가 성립한

다[9].

3. k-最大容量經路 計算法

3.1 反復에 의한 k-最大容量 計算技法

지금까지 討議한 代數的構造에 관한 理論을 適用하여 네트웍내에 임의로 주어진 마디 s에서 다른 모든 마디에 이르는 k-最大容量을 구하는 세가지 方法에 대하여 알아본다. 이것은 네트웍의 弧容量行列 A에 대한 行列 A^*의 s번째 行을 구하는 작업이며, 單位行列 E의 s번째 行을 $\mathbf{e}_s$라고 定義하면 行列 A^*의 s번째 行은 $\mathbf{e}_s \oplus A^*$로 表示되므로 $\mathbf{e}_s \otimes A^*$를 구하는 작업이라고 할 수 있다.

여기에 서술된 모든 方法들은 一次聯立方程式의 解를 구하는 方法의 일종인 反復技法[1, 4]을 第2章에서 설명한 네트웍의 代數的構造에 適用시킨 것이다. 편의상 ⊕ 演算記號는 혼동을 일으키지 않는 한 이후 생략한다.

3.1.1 Jacobi方法

네트웍의 마디s에서 다른 모든 마디에 이르는 k-最大容量의 最初推定벡터를 $\mathbf{x}^{(0)}$이라 하고 $r(\geq 0)$회째 推定벡터를 $\mathbf{x}^{(r)}$이라 하며 $\mathbf{e}_s$는 單位行列 E의 s번째 行벡터라고 한다. 그러면 구하고자 하는 해 $\mathbf{x}=\mathbf{e}_sA^*$는 보조정리 2에 의거 $\mathbf{x}=\mathbf{x}\otimes A \oplus \mathbf{e}_s$가 된다. 이때 이를 구하는 절차를 一般的 Jacobi方法(Generalized Jacobi Method)이라 하며, 이는 다음 반복적인 式

$$\mathbf{x}^{(0)}=\mathbf{e}_s$$
$$\mathbf{x}^{(r+1)}=\mathbf{x}^{(r)}A \oplus \mathbf{e}_s (r\geq 0) \qquad (3-1)$$

에 의하여 r회째 推定벡터 $\mathbf{x}^{(r)}$을 구한다. 여기에서 $\mathbf{x}^{(r+1)}$의 모든 性分은 $\mathbf{x}^{(r)}$의 n개 成分을 기초로 計算된다. 式(3−1)의 反復法에 따라서 推定벡터를 계속 나열하여 보면

$$\mathbf{x}^{(1)}=\mathbf{x}^{(0)}A \oplus \mathbf{e}_s=\mathbf{e}_s(E \oplus A)=\mathbf{e}_sA^{<1>}$$
$$\mathbf{x}^{(2)}=\mathbf{x}^{(1)}A \oplus \mathbf{e}_s=\mathbf{e}_sA^{<1>}A \oplus \mathbf{e}_s=\mathbf{e}_sA^{<2>}$$
$$\cdots$$
$$\mathbf{x}^{(r)}=\mathbf{e}_sA^{<r>}$$

와 같으며 補助定理1로부터 $r\geq w$인 모든 r에 대해 $\mathbf{x}^{(r)}=\mathbf{e}_sA^*$가 成立함을 알 수 있다.

定理1 : Jacobi方法에 의한 推定벡터 $\mathbf{x}^{(r)}$은 最大$(2n+1)$회 反復이내로 $\mathbf{e}_sA^*$에 수렴한다[3].

定理1에서 $\mathbf{x}^{(r+1)}=\mathbf{x}^{(r)}$ $(r\geq 0)$이면 $\mathbf{e}_sA^*$으로의 수렴이 보장된다. 이는

$$\mathbf{x}^{(r+1)}=\mathbf{x}^{(r)} \Rightarrow \mathbf{x}^{(r+2)}=\mathbf{x}^{(r+1)}A \oplus \mathbf{e}_s=\mathbf{x}^{(r)}A \oplus \mathbf{e}_s=\mathbf{x}^{(r+1)}$$
$$\Rightarrow \mathbf{x}^{(r)}=\mathbf{x}^{(r+1)}=\mathbf{x}^{(r+2)}=\cdots=\mathbf{x}^{(w)}=\mathbf{e}_sA^*$$

이기 때문이다. 이상의 Jocobi方法을 計算法으로 表現하면 다음과 같다.

計算法2 : 一般的 Jacobi方法으로 $\mathbf{e}_sA^*$를 計算한다.

(1) $\mathbf{x}^{(0)}=\mathbf{e}_s$ 및 $r=0$으로 둔다.
(2) $\mathbf{x}^{(r+1)}=\mathbf{x}^{(r)}A \oplus \mathbf{e}_s$를 計算한다.
(3) 만약 $\mathbf{x}^{(r)}=\mathbf{x}^{(r+1)}$이면 단계 (4)로 간다.
　아니면 $r=r+1$로두고 단계 (2)로 간다.
(4) $\mathbf{x}^{(r)}=\mathbf{e}_sA^*$이므로 끝낸다.

$\mathbf{e}_sA^*$의 j번째 成分은 C^k의 元素로서 마디 s에서 마디 j로 가는 모든 경로에 대한 k-最大容量을 의미한다.

3.1.2 Gauss-Seidel方法

임의의 正方行列 B는 對角線行列(diagonal matrix) D와 上位三角行列(upper triangular matrix) U 및 下位三角行列(lower triangular matrix) L의 합으로

$$B=U+D+L$$

와 같이 分解할 수 있다. 여기서 취급하는 네트웍 G의 弧容量行列 A도 이와같이 分解될 수 있으며 가정에서 네트웍上에는 自體環이 없으므로 對角線行列 D는 零行列 V와 같다. 따라서 $A=U \oplus L$로 表示할 수 있다. 여기에서 弧容量行列이 각각 U와 L인 네트웍 G_U와 G_L은 回路가 없는 네트웍임을 알 수 있다.

Jocobi方法에서와 같이 最初推定벡터를 $\mathbf{x}^{(0)}$

이라 하고 $r(\geq 0)$회째 推定벡터를 $\mathbf{x}^{(r)}$이라 하며 $\mathbf{e}_s$는 單位行列 E의 s번째 行벡터라 한다. 그러면 구하고자 하는 해 $\mathbf{x}=\mathbf{e}_sA^*$는 補助定理2로부터 $\mathbf{x}=\mathbf{x}(U\oplus L)+\mathbf{e}_s=\mathbf{x}U\oplus\mathbf{x}L+\mathbf{e}_s$가 된다. 이때 이를 푸는 절차를 一般的 Gauss-Seidel 方法 (Generalized Gauss - Seidel Method)이라하며, 이는 다음 반복적인 式

$$\mathbf{x}^{(0)}=\mathbf{e}_s$$
$$\mathbf{x}^{(r+1)}=\mathbf{x}^{(r+1)}U\oplus\mathbf{x}^{(r)}L\oplus\mathbf{e}_s(r\geq 0) \qquad (3-2)$$

에 의하여 r회째 推定벡터 $\mathbf{x}^{(r)}$을 구한다. 여기에서 $\mathbf{x}^{(r+1)}$의 j번째 成分 $x_j^{(r+1)}$은 $x_1^{(r+1)}, \cdots, x_{j-1}^{(r+1)}$ 및 $x_{j+1}^{(r)}, \cdots, x_n^{(r)}$을 기초로 計算되며 計算順序는 $j=1, 2, \cdots, n$의 순으로 시행된다.

U는 回路가 없는 네트웍의 弧容量行列이므로 補助定理2에 의하여 式(3-2)와 同値인 方程式을 만들면 다음과 같다.

$$\mathbf{x}^{(r+1)}=\mathbf{x}^{(r)}LU^*\oplus\mathbf{e}_sU^* \qquad (3-3)$$

여기에서 弧容量行列이 LU^*인 네트웍H를 생각하여 보면 네트웍H의 각 弧는 네트웍 G_L의 한 弧와 네트웍 G_U에서의 經路를 합친 것에 대응되며 즉 네트웍 G의 經路를 나타낸다. 이 사실에 의하여 式(3-3)을 살펴보면 Gauss-Seidel方法은 Jacobi方法의 變形에 불과하다는 것을 알 수 있다. 式(3-3)의 反復法에 따라 推定벡터를 계속 구하면 다음과 같다.

$$\mathbf{x}^{(1)}=\mathbf{x}^{(0)}LU^*\oplus\mathbf{e}_sU^*$$
$$\begin{aligned}\mathbf{x}^{(2)}&=\mathbf{x}^{(1)}LU^*\oplus\mathbf{e}_sU^*\\&=(\mathbf{x}^{(0)}LU^*\oplus\mathbf{e}_sU^*)LU^*\oplus\mathbf{e}_sU^*\\&=\mathbf{x}^{(0)}(LU^*)^2\oplus\mathbf{e}_sU^*(E\oplus LU^*)\\&=\mathbf{x}^{(0)}(LU^*)^2\oplus\mathbf{e}_sU^*(LU^*)^{<1>}\end{aligned}$$
$$\cdots$$
$$\mathbf{x}^{(r)}=\mathbf{x}^{(0)}(LU^*)^r\oplus\mathbf{e}_sU^*(LU^*)^{<r-1>} \qquad (3-4)$$

行列 LU^*에 대해서도 補助定理1이 適用되므로 w가 存在하며 $r>w$에 대하여 $\mathbf{x}^{(r)}=\mathbf{x}^{(0)}(LU^*)^r\oplus\mathbf{e}_sU^*(LU^*)$가 成立한다. 그리고 $A=U\oplus L$이므로 補助定理3에 의하여 $A^*=(U\oplus L)^*=U^*(LU^*)^*$가 成立한다. 그러므로 $r>w$일 때 性質(2-8)에 의하여

$$\mathbf{x}^{(r)}=\mathbf{x}^{(0)}(LU^*)^r\oplus\mathbf{e}_sA^*\geq\mathbf{e}_sA^*$$

가 成立한다. 한편 $\mathbf{x}^{(0)}=\mathbf{e}_s=\mathbf{e}_sE\leq\mathbf{e}_sA^*$이고 $L\leq A\leq A^*$이며 $U\leq A$에서 $U^*\leq A^*$이므로 特性(2-10)에 의하여

$$\mathbf{x}^{(0)}(LU^*)^r\leq\mathbf{e}_sA(A^*A^*)^r=\mathbf{e}_sA^*$$

가 成立한다. 따라서 $\mathbf{x}^{(r)}\leq\mathbf{e}_sA^*\oplus\mathbf{e}_sA^*=\mathbf{e}_sA^*$이다. 여기에서 性質(2-8)의 反對稱律에 의하여 $r>w$인 모든 r에 대하여 $\mathbf{x}^{(r)}=\mathbf{e}_sA^*$가 成立함을 알 수 있다.

定理2 : Gauss-Seidel方法에 의한 推定벡터 $\mathbf{x}^{(r)}$은 有限回 反復으로 $\mathbf{e}_sA^*$에 수렴한다[3].

Jacobi方法에서와 같이 $\mathbf{x}^{(r+1)}=\mathbf{x}^{(r)}(r\geq 0)$이면 $\mathbf{e}_sA^*$으로의 수렴이 보장되며 이상의 Gauss-Seidel方法을 計算法으로 表現하면 다음과 같다.

計算法3 : 一般的 Gauss-Seidel方法으로 $\mathbf{e}_sA^*$를 計算한다.

(1) $A=U\oplus L$로 分解하고 $\mathbf{x}^{(0)}=\mathbf{e}_s$ 및 $r=0$으로 둔다.
(2) $\mathbf{x}^{(r+1)}=\mathbf{x}^{(r+1)}U\oplus\mathbf{x}^{(r)}L\oplus\mathbf{e}_s$를 計算한다.
(3) 만약 $\mathbf{x}^{(r)}=\mathbf{x}^{(r+1)}$이면 단계 (4)로 간다. 아니면 $r=r+1$로 두고 단계 (2)로 간다.
(4) $\mathbf{x}^{(r)}=\mathbf{e}_sA^*$이므로 끝낸다.

Jacobi方法보다 Gauss-Seidel方法이 現反復段階에서 最適解에 더 가까운 推定벡터를 사용하므로 推定벡터가 더 빨리 最適解에 도달할 것이라 추측할 수 있다.

3.1.3 double-sweep方法

Gauss-Seidel方法은 r회째 推定벡터 $\mathbf{x}^{(r)}$의 成分 $x_j^{(r)}$을 計算할 때 항상 $j=1, 2, \cdots, n$의 順序로 시행한다. 여기에서 $\mathbf{x}^{(r)}$을 計算할 때 r이 홀수와 짝수일 때에 따라서 $j=n, n-1, \cdots, 1$의 順序와 $j=1, 2, \cdots, n$의 順序를 交代로 사용하면 最適解에 더욱 빠르게 수렴할 것이라는 생각이 double-sweep方法을 開發하게 된 動機라고 볼 수 있다.

Gauss-Seidel方法에서와 같이 弧容量行列

A를 U⊕L로 分解하면 一般的 double-sweep 方法(Generalized Double-Sweep Method)은

$$\mathbf{x}^{(0)}=\mathbf{e}_s$$
$$\mathbf{x}^{(2r-1)}=\mathbf{x}^{(2r-1)}L\oplus\mathbf{x}^{(2r-2)}$$
$$\mathbf{x}^{(2r)}=\mathbf{x}^{(2r)}U\oplus\mathbf{x}^{(2r-1)}\ (r\geq 1) \qquad (3-5)$$

에 의하여 계속적인 推定벡터를 구하는 方法이다. 여기에서 $\mathbf{x}^{(2r-1)}$의 成分 $x_j^{(2r-1)}$은 j=n, n−1, …, 1의 順序로 計算하는 後向檢査(backward sweep)를 사용하고, $\mathbf{x}^{(2r)}$의 成分 $x_j^{(2r)}$은 j=1, 2, …, n의 順序로 計算하는 前向檢査(forward sweep)를 사용한다.

L과 U는 回路가 없는 네트웍의 弧容量行列이므로 補助定理2에 의해 式(3−5)와 同値인 방定食을 만들면 다음과 같으며,

$$\mathbf{x}^{(2r-1)}=\mathbf{x}^{(2r-2)}L^*$$
$$\mathbf{x}^{(2r)}=\mathbf{x}^{(2r-1)}U^* \qquad (3-6)$$

式(3−6)을 反復하여 사용하면 다음과 같이 된다.

$$\mathbf{x}^{(2r-1)}=\mathbf{x}^{(0)}(L^*U^*)^{r-1}L^*$$
$$\mathbf{x}^{(2r)}=\mathbf{x}^{(0)}(L^*U^*)^{r} \qquad (3-7)$$

$L^*\geq E\oplus L$ 및 $U^*\geq E\oplus U$ 이므로 $L^*U^*\geq(E\oplus L)(E\oplus U)\geq E\oplus L\oplus U=E\oplus A$이며 $(L^*U^*)^m\leq(E\oplus A)^m=A^{<m>}$이다. 따라서 式(3−7)과 補助定理1에 의하여 r>w이면

$$\mathbf{x}^{(2r-1)}\geq\mathbf{x}^{(0)}A^{<r-1>}(E\oplus L)\geq\mathbf{x}^{(0)}A^*$$
$$\mathbf{x}^{(2r)}\geq\mathbf{x}^{(0)}A^{<r>}=\mathbf{x}^{(0)}A^*$$

가 成立한다. 한편 $L\leq A$와 $U\leq A$에서 $L^*\leq A^*$ 및 $U^*\leq A^*$이므로 $L^*U^*\leq A^*A^*=A^*$이다. 그러므로 r≥1이면 特性(2−10)에 의하여 式(3−7)에서

$$\mathbf{x}^{(2r-1)}\geq\mathbf{x}^{(0)}A^{<r-1>}(E\oplus L)\geq\mathbf{x}^{(0)}A^*$$
$$\mathbf{x}^{(2r)}\geq\mathbf{x}^{(0)}A^{<r>}=\mathbf{x}^{(0)}A^*$$

가 成立한다. 한편 $L\leq A$와 $U\leq A$에서 $L^*\leq A^*$이므로 $L^*U^*\leq A^*A^*=A^*$이다. 그러므로 r≥1이면 特性(2−10)에 의하여 式(3−7)에서

$$\mathbf{x}^{(2r-1)}\leq\mathbf{x}^{(0)}(A^*)^{r-1}A^*=\mathbf{x}^{(0)}A^*$$
$$\mathbf{x}^{(2r)}\leq\mathbf{x}^{(0)}A^{r}=\mathbf{x}^{(0)}A^*$$

가 成立한다. 이상을 綜合하면 충분히 큰 t에 대하여 $\mathbf{x}^{(t)}=\mathbf{x}^{(0)}A^*=\mathbf{e}_sA^*$가 成立함을 알 수 있다.

定理3: double-sweep方法에 의한 推定벡터 $\mathbf{x}^{(r)}$은 有限回 反復으로 $\mathbf{e}_sA^*$에 수렴한다[9].

double-sweep方法의 수렴조건에 대하여 알아본다. 우선 $\mathbf{x}^{(2r)}=\mathbf{x}^{(2r-1)}$이라고 가정하면

$$\mathbf{x}^{(2r+1)}=\mathbf{x}^{(2r)}L^*=\mathbf{x}^{(2r-1)}L^*=\mathbf{x}^{(2r-2)}L^*L^*$$
$$=\mathbf{x}^{(2r-2)}L^*=\mathbf{x}^{(2r-1)}$$
$$\mathbf{x}^{(2r+2)}=\mathbf{x}^{(2r+1)}U^*=\mathbf{x}^{(2r-1)}U^*=\mathbf{x}^{(2r)}$$

이므로 $\mathbf{x}^{(2r-1)}=\mathbf{x}^{(2r)}=\mathbf{x}^{(2r+1)}=\mathbf{x}^{(2r+2)}=\mathbf{e}_sA^*$이다. $\mathbf{x}^{(2r+1)}=\mathbf{x}^{(2r)}$일때도 마찬가지로 $\mathbf{x}^{(2r)}=\mathbf{e}_sA^*$가 成立한다. 그러므로 t≥1일 때 $\mathbf{x}^{(t+1)}=\mathbf{x}^{(t)}$이면 $\mathbf{e}_sA^*$으로의 수렴이 보장된다. 이상을 計算法으로 表示하면 다음과 같다.

計算法4: 一般的 double-sweep方法으로 $\mathbf{e}_sA^*$를 計算한다.

(1) $A=U\oplus L$로 分解하고 $\mathbf{x}^{(0)}=\mathbf{e}_s$ 및 r=0으로 둔다.

(2) 만약 r이 홀수이면 단계 (3)으로 간다.
아니면 後向으로 $\mathbf{x}^{(r+1)}=\mathbf{x}^{(r+1)}L\oplus\mathbf{x}^{(r)}$을 計算하고 단계 (4)로 간다.

(3) 前向으로 $\mathbf{x}^{(r+1)}=\mathbf{x}^{(r+1)}U\oplus\mathbf{x}^{(r)}$을 計算한다.

(4) 만약 r≥1이고 $\mathbf{x}^{(r)}=\mathbf{x}^{(r+1)}$이면 단계 (5)로 간다.
아니면 r=r+1로 두고 단계 (2)로 간다.

(5) $\mathbf{x}^{(r)}=\mathbf{e}_sA^*$이므로 끝낸다.

3.2 例題

간단한 例題를 통하여 3-最大容量을 구하기 위한 弧容量行列 A를 構成하고 지금까지 討議한 3가지 方法으로써 마디 1에서 모든 마디에 이르는 3-最大容量을 구하여 본다.

그림 3−1의 네트웍에서 弧上의 數는 弧容量이라고 假定한다. 그러면 3가지 方法에서 사

용되는 弧容量行列 A와 最初推定벡터 $\mathbf{x}^{(0)}$은 다음과 같다.

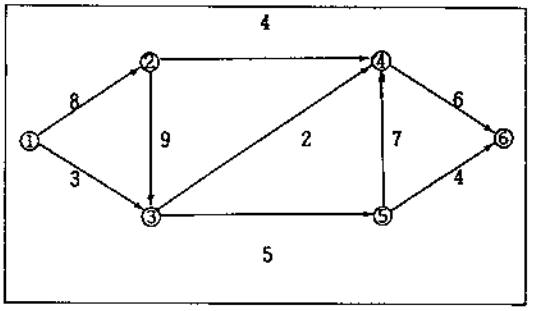

그림 3-1. 3-最大容量 例題 네트웍

$$A=\begin{bmatrix}(0,0,0) & (8,0,0) & (3,0,0) & (0,0,0) & (0,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (9,0,0) & (4,0,0) & (0,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (2,0,0) & (5,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (6,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (7,0,0) & (0,0,0) & (4,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0)\end{bmatrix}$$

$$\begin{aligned}\mathbf{x}^{(0)}&=\mathbf{e}_1\\ &=[(\infty,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)\\ &\quad (0,0,0)\ (0,0,0)\ (0,0,0)].\end{aligned}$$

3.2.1 Jacobi方法

Jacobi方法은 式(3-1)에 의하여 r회째 推定벡터 $\mathbf{x}^{(r)}$을 구하며 $\mathbf{x}^{(r+1)}$의 모든 成分은 $\mathbf{x}^{(r)}$의 6개 成分을 기초로 計算된다.

$$\begin{aligned}\mathbf{x}^{(1)}&=\mathbf{x}^{(0)}\oplus A\oplus\mathbf{e}_1\\ &=[(\infty,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)]\\ &\otimes\begin{bmatrix}(0,0,0) & (8,0,0) & (3,0,0) & (0,0,0) & (0,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (9,0,0) & (4,0,0) & (0,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (2,0,0) & (5,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (6,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (7,0,0) & (0,0,0) & (4,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0)\end{bmatrix}\\ &\oplus[(\infty,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)]\\ &=[\ (0,0,0)\ (8,0,0)\ (3,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)]\\ &\oplus[(\infty,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)]\\ &=[(\infty,0,0)\ (8,0,0)\ (3,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)]\end{aligned}$$

이 되며 마찬가지로 $\mathbf{x}^{(2)}$를 計算하면

$$\begin{aligned}\mathbf{x}^{(2)}&=\mathbf{x}^{(1)}\oplus A\oplus\mathbf{e}_1\\ &=[(\infty,0,0)\ (8,0,0)\ (3,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)]\\ &\otimes\begin{bmatrix}(0,0,0) & (8,0,0) & (3,0,0) & (0,0,0) & (0,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (9,0,0) & (4,0,0) & (0,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (2,0,0) & (5,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (6,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (7,0,0) & (0,0,0) & (4,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0)\end{bmatrix}\\ &\oplus[(\infty,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)]\\ &=[\ (0,0,0)\ (8,0,0)\ (8,3,0)\ (4,2,0)\ (3,0,0)\ (0,0,0)]\\ &\oplus[(\infty,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)]\\ &=[(\infty,0,0)\ (8,0,0)\ (8,3,0)\ (4,2,0)\ (3,0,0)\ (0,0,0)]\end{aligned}$$

이 된다. 이와같은 方法으로써 계속하여 $\mathbf{x}^{(3)}$, $\mathbf{x}^{(4)}$, … 等을 計算하면

$$\begin{aligned}\mathbf{x}^{(3)}&=\mathbf{x}^{(2)}\otimes A\oplus\mathbf{e}_1\\ &=[(\infty,0,0)\ (8,0,0)\ (8,3,0)\ (4,3,2)\ (5,3,0)\ (4,3,2)]\\ \mathbf{x}^{(4)}&=\mathbf{x}^{(3)}\otimes A\oplus\mathbf{e}_1\\ &=[(\infty,0,0)\ (8,0,0)\ (8,3,0)\ (5,4,3)\ (5,3,0)\ (4,3,2)]\\ \mathbf{x}^{(5)}&=\mathbf{x}^{(4)}\otimes A\oplus\mathbf{e}_1\\ &=[(\infty,0,0)\ (8,0,0)\ (8,3,0)\ (5,4,3)\ (5,3,0)\ (5,4,3)]\\ \mathbf{x}^{(6)}&=\mathbf{x}^{(5)}\otimes A\oplus\mathbf{e}_1\\ &=[(\infty,0,0)\ (8,0,0)\ (8,3,0)\ (5,4,3)\ (5,3,0)\ (5,4,3)]\end{aligned}$$

과 같이 되며 여기에서 $\mathbf{x}^{(6)}=\mathbf{x}^{(5)}$이므로 $\mathbf{x}^{(5)}$가 最終解이다.

3.2.2 Gauss-Seidel方法

弧容量行列 A를 上位三角行列 U와 下位三角行列 L로 分解하여 $A=U\oplus L$이 되도록 하면 U와 L은 다음과 같다.

$$U=\begin{bmatrix}(0,0,0) & (8,0,0) & (3,0,0) & (0,0,0) & (0,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (9,0,0) & (4,0,0) & (0,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (2,0,0) & (5,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (6,0,0)\\ (0,0,0) & (0,0,0) & (7,0,0) & (0,0,0) & (0,0,0) & (4,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0)\end{bmatrix}$$

$$L=\begin{bmatrix}(0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (7,0,0) & (0,0,0) & (0,0,0)\\ (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0) & (0,0,0)\end{bmatrix}$$

Gauss-Seidel方法은 式(3−2)에 의하여 r 회째 推定벡터 $\mathbf{x}^{(r)}$을 구하는데 $\mathbf{x}^{(r+1)}$의 j번째 成分 $x_j^{(r+1)}$은 $x_1^{(r+1)}$, …, $x_{j-1}^{(r+1)}$ 및 $x_{j+1}^{(r)}$, …, $x_6^{(r)}$을 기초로 計算되며 計算順序는 j=1, 2, …, 6의 순으로 시행된다. 먼저

$$\mathbf{x}^{(0)}\otimes \mathrm{L}\oplus \mathbf{e}_1=[(\infty,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)]$$

를 계산한 다음에 $\mathbf{x}^{(1)}$의 1번째 成分 $x_1^{(1)}$은 $\mathbf{x}^{(1)}$(始作時 $\mathbf{x}^{(0)}$과 같음.)과 行列 U의 1列과의 ⊕ 演算을 시행한 結果와 $(\mathbf{x}^{(0)}\oplus \mathrm{L}\oplus \mathbf{e}_1)$의 1번째 成分 (∞,0,0)을 ⊕ 演算하면

$$x_1^{(1)}=\mathbf{x}^{(1)}\oplus(\text{行列 U의 1列})\oplus(\infty,0,0)=(\infty,0,0)$$

을 얻는다. 그리고 $\mathbf{x}^{(1)}$의 2번째 成分 $x_2^{(1)}$은 $\mathbf{x}^{(1)}=[x_1^{(1)}x_2^{(0)}\cdots x_6^{(0)}]$과 行列 U의 2烈과의 ⊗演算을 시행한 結果와 $(\mathbf{x}^{(0)}\otimes \mathrm{L}\oplus \mathbf{e}_1)$의 2번째 成分(0,0,0)을 ⊕ 演算하면 된다. 같은 방법으로 $x_6^{(1)}$까지 계산하면 다음 결과를 얻는다.

$$\mathbf{x}^{(1)}=[(\infty,0,0)\ (8,0,0)\ (8,3,0)\ (4,2,0)\ (5,3,0)\ (4,3,2)]$$

같은 방법으로 $\mathbf{x}^{(1)}$을 사용하여 $\mathbf{x}^{(2)}$를, $\mathbf{x}^{(2)}$를 사용하여 $\mathbf{x}^{(3)}$를 계산하면 다음 결과를 얻는다.

$$\mathbf{x}^{(2)}=[(\infty,0,0)\ (8,0,0)\ (8,3,0)\ (5,4,3)\ (5,3,0)\ (5,4,3)]$$

$$\mathbf{x}^{(3)}=[(\infty,0,0)\ (8,0,0)\ (8,3,0)\ (5,4,3)\ (5,3,0)\ (5,4,3)]$$

여기서 $\mathbf{x}^{(3)}=\mathbf{x}^{(2)}$이므로 $\mathbf{x}^{(2)}$가 最終解이다.

3.2.3 double-sweep 方法

double-sweep 方法은 Gauss-Swidel方法과 같이 弧容量行列 A를 A=U⊕L로 分解한 다음에 式(3−5)에 의하여 推定벡터를 計算하되 홀수번째 推定벡터인 $\mathbf{x}^{(2r-1)}$의 成分 $x_j^{(2r-1)}$은 j=n, n−1, …, 1의 順序로 計算하는 後向檢査를 사용하고, 짝수번째 推定벡터인 $\mathbf{x}^{(2r)}$의 成分 $x_j^{(2r)}$은 j=1, 2, …n의 順序로 計算하는 前向檢査를 사용한다.

$$\mathbf{x}^{(1)}=[(\infty,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)\ (0,0,0)]$$

$$\mathbf{x}^{(2)}=[(\infty,0,0)\ (8,0,0)\ (8,3,0)\ (4,2,0)\ (5,3,0)\ (4,3,2)]$$

$$\mathbf{x}^{(3)}=[(\infty,0,0)\ (8,0,0)\ (8,3,0)\ (5,4,3)\ (5,3,0)\ (4,3,2)]$$

$$\mathbf{x}^{(4)}=[(\infty,0,0)\ (8,0,0)\ (8,3,0)\ (5,4,3)\ (5,3,0)\ (5,4,3)]$$

$$\mathbf{x}^{(5)}=[(\infty,0,0)\ (8,0,0)\ (8,3,0)\ (5,4,3)\ (5,3,0)\ (5,4,3)]$$

여기서 $\mathbf{x}^{(5)}=\mathbf{x}^{(4)}$가 最終解이다.

여기에서 最終解로 수렴하는데 필요한 Jacobi方法 및 Gauss-Seidel方法의 推定벡터 計算回數와 double-sweep方法의 前後向檢査回數는 各各 6회, 3회 및 2.5회임을 알 수 있다.

3.3 k-最終容量經路 識別節次

세가지 方法에 의거 特定 源마디에서 모든 마디에 이르는 k-最大容量을 일단 구하고 나면 追跡節次(tracing procedure)를 따라서 經路容量에 對應하는 該當 經路를 識別하게 된다. 실제로 源마디 s에서 容量經路를 구하는데 필요한 情報는 $\mathbf{e}_s\mathbf{A}^*$에 모두 包含되어 있다.

이 追跡節次를 설명하기 위하여 源마디 s에서 마디 i까지의 k-最大容量 가운데 m번째 (1≤m≤k)로 큰 容量 $(a^*si)_m$에 對應하는 經路를 알아보도록 하며 편의상 $C(i, m)=(a^*si)_m$으로 定義하고 $0<C(i, m)<\infty$으로 가정한다. 그러면 다음 條件을 만족하는 마디 i와 인접한 마디 j를 選定할 수 있다.

$$\min\{C(j, \ell), f_{ji}\}=C(i, m) \qquad (3-8)$$

단, f_{ji}는 弧(j, i)의 容量이고,

$C(j, \ell)$은 $1\le \ell \le k$로 마디 s에서 마디 j까지의 ℓ번째로 큰 容量이다.

式(3−8)을 만족하는 마디 j와 번호 ℓ은 적어도 하나 이상 存在한다. 왜냐하면 源마디 s에서 마디 i까지의 m번째로 큰 容量에 對應하는 經路중 그 經路上의 中間마디를 택하여 生成되는 마디 s에서 中間마디까지의 部分經路를

생각하면 그 部分經路의 容量은 中間마디의 k-最大容量중 하나에 該當되기 때문이다.

여기서 마디 j는 要求經路의 中間마디에 該當하며 일단 j가 發見되고나면 다시 $i \Leftarrow j$, $m \Leftarrow \ell$로 두고 j=s가 될 때까지 同一한 節次를 反復한다. 그리고 中間過程에서 式(3-8)을 만족하는 마디 j가 여러개 있을 수 있는데 그때에는 要求經路가 複數個 存在하게 된다. 이상을 計算法으로 나타내면 다음과 같다.

計算法5 : 源마디 s에서 마디 i에 이르는 m번째($1 \le m \le k$)로 큰 容量 C(i, m)에 對應하는 經路를 追跡함 (단, $0 < C(i,m) < \infty$).

(1) 經路上의 마디番號를 記憶할 크기 2n의 LIST를 定義하며,

ptr=1, LIST(ptr)=i로 둔다.

(2) 마디 i와 인접하며 다음 條件을 만족하는 마디 j를 選定한다.

$\min\{C(j, \ell), f_{ji}\} = C(i, m)$

단, f_{ji}는 弧(j, i)의 容量이고,

$C(j, \ell)$은 $1 \le \ell \le k$로 마디 s에서 마디 j까지의 ℓ번째로 큰 容量이다.

(3) ptr=ptr+1, LIST(ptr)=j로 둔다.

(4) 만약 j=s이면 단계 (5)로 간다.

아니면 $i=j$, $m=\ell$로 두고 단계 (2)로 간다.

(5) 要求經路상의 마디인 LIST(ptr), LIST(ptr-1), LIST(1)을 찾았으므로 끝낸다.

만약 回路가 없는 經路를 發見하기 원한다면 지금까지의 追跡節次에서 인접마디를 選定할 때 그 마디가 마디選定 전까지의 經路上에 있던 마디로서 이미 고려되었는지 아닌지를 檢査하는 過程을 計算法 5에 追加하게 된다. 그러므로 回路가 없는 容量經路를 追跡할 경우는 計算法5의 단계(2)와 단계(3) 사이에 다음의 단계를 追加하면 된다.

(추가단계) 만약 j가 LIST(p)(p=1, …, ptr)와 同一하면 단계 (2)로 간다.

4. 세 技法의 比較

Jacobi方法, Gauss-Seidel方法 및 double-sweep方法 等의 計算法을 比較하기 위하여 각 計算法의 計算上의 複雜性(computational complexity)을 分析하여 본다. 세가지 方法에 있어서 計算複雜性은 一般的演算인 一般的最大化와 一般的最少化의 函數라고 할 수 있으며 一般的最大化와 一般的最少化는 거의 同一한 計算時間이 소요된다고 볼 수 있다. 이는 一般的最大化는 k회 比較演算이 要求되며 一般的最小化는 거의 (k+2)회 정도의 比較演算이 필요하기 때문이다,

마디가 n개인 네트웍에서 우선 각 方法別 推定벡터 1회 計算時 要求되는 일반적 演算의 總數를 알아보면 Jacobi方法은 $\mathbf{x}^{(r+1)} = \mathbf{x}^{(r)}A \oplus \mathbf{e}_s$ 計算過程에서 $\mathbf{x}^{(r)}A$의 計算에 필요한 一般的最少化와 一般的最大化의 數는 各各 n(n-1)이며 그 結果와 $\mathbf{e}_s$의 ⊕ 計算에는 n회의 一般的最大化가 필요하다. 그러므로 Jacobi方法에서 推定벡터의 1회 計算時 要求되는 一般的演算의 總數는 $2n(n-1)+n=n(2n-1)$이 된다.

한편 Gauss-Seidel方法은 $\mathbf{x}^{(r+1)} = \mathbf{x}^{(4+1)}U \oplus \mathbf{x}^{(r)}L \oplus \mathbf{e}_s$ 計算過程에서 마디는 1, 2, …, n과 같이 順序的으로 다루어지며 이 過程에서 필요한 一般的最小化 ⊗와 一般的最大化 ⊕의 數는 表 4-1과 같다. Gauss-Seidel方法에서 推定 벡터의 1회 計算時 필요한 一般的演算의 總數는 表 4-1에 의하여 $2n^2$이 된다.

表 4-1. Gauss-Seidel方法時 一般的演算의 數

마디	一般的最小化의 數	一般的最大化의 數
1	0+(n-1)	n+1
2	1+(n-2)	n+1
3	2+(n-3)	n+1
…	…	…
n	(n-1)+0	n+1
計	n(n-1)	n(n+1)

마찬가지 方法으로 double-sweep方法에서도 前後向檢査時에 필요한 一般的 演算의 總數는 $2n^2$이 되므로 Jacobi方法 및 Gauss-Seidel方法의 推定벡터 1회 計算時와 double-sweep方法의 前後向檢査 1회 計算時에 필요한 一般的 演算의 總數는 모두 약 $2n^2$이 됨을 알 수 있다.

한편 定理1에 의하여 jacobi方法의 反復回數는 最惡의 경우에 $(2n+1)$이 요구되므로 計算을 終了하기 위한 一般的演算의 最大計算量은 다음과 같다.

$$(2n+1)n(2n-1)=n(4n^2-1)\fallingdotseq 4n^3$$

이상에서 세가지 方法의 計算複雜性이 비슷한 것 같지만 다음 定理에 의하여 double-sweep方法이 가장 우수하고, Gauss-Seidel方法이 다음으로 우수한 것을 알 수 있다.

定理4 : Jacobi方法, Gauss-Seidel方法 및 double-sweep方法에 의한 推定벡터 列을 각각 $\mathbf{x}^{(r)}$, $\mathbf{y}^{(r)}$, $\mathbf{z}^{(r)}$로 두면 모든 $r\geq 0$에 대하여 다음 不等式이 成立한다[9].

$$\mathbf{x}^{(r)}\leq \mathbf{y}^{(r)}\leq \mathbf{z}^{(2r)} \qquad (4-1)$$

5. 結 論

네트웍問題를 다룰 때 단일의 最適經路가 追加制約條件으로 사용될 수 없을 경우는 次善의 最適解를 구해야 할 필요성에 直面하게 된다. 本 硏究에서는 이런 問題를 解決하기 위하여 容量네트웍에서 複數個의 最適解를 구하기 위한 代數的構造를 開發하고 이 代數的構造에서 k-最大容量經路를 決定하는 方法을 提示하였다.

먼저 네트웍에 弧容量의 相互關係로 定義되는 2가지 一般的演算(一般的最大化 및 一般的最少化)을 導入하여 代數的構造를 設定하고 順序關係를 定義한 다음에 k-最大容量經路問題는 네트웍의 弧容量行列 A에 對應하는 k-最大容量行列 A^*를 구하는 問題로 歸着된다는 사실을 보였고 補助定理1에 의하여 陰의 容量 및 自體環이 없는 네트웍에서 有限回 計算으로 A^*를 구할 수 있음을 誘道하였으며 計算法1에 그 計算過程을 提示하였다. 計算法1은 1회 實行으로 네트웍의 임의의 마디에서 임의의 마디에 이르는 k-最大容量을 구하는 長點이 있으나 計算이 복잡하고 時間이 많이 걸리는 短點을 가지고 있다.

그 다음으로 線型方程式의 解를 구하는 古典的技法인 Jacobi方法, Gauss-Seidel方法 및 double-sweep方法 等의 反復技法을 變形시킨 一般的 Jacobi方法, 一般的 Gauss-Seidel方法 및 一般的 double-sweep方法 等으로 k-最大容量經路問題를 解決하는 理論과 節次를 定理1-定理3 및 計算法2-計算法4에서 提示하였다. 이 方法들의 基本槪念은 k-最大容量의 最初推定벡터를 이용하여 다음 推定벡터를 算出하며 다시 이 推定벡터로서 그 다음 推定벡터를 算出하는 過程을 最適解로 수렴될 때까지 反復하는 것이다.

最適解의 計算에 필요한 一般的演算의 最大計算量은 마디수가 n일 때 3가지 方法 모두 $4n^3$이지만 double-sweep方法이 가장 우수하고, Gauss-Seidel方法이 다음으로 우수하다는 것이 定理4에 의거 判明되었다. 反復技法은 計算法1보다는 計算이 간편하고 時間도 적게 소요되지만 源마디가 變更되면 다시 計算하여야 하는 短點이 있음을 밝혀 둔다.

네트웍의 特定 源마디에서 모든 마디에 이르는 k-最大容量을 일단 구하고 나면 計算法5의 追跡節次에 의거 該當되는 k-最大容量經路를 구할 수 있으며 回路가 없는 經路를 구하는 경우도 고려하였다.

k-最大容量은 육로, 해로, 항공로를 포함하는 수송망에 대해 疏通量을 통제하는 데 적용 가능하며 통신망, 전산망 등에 사용량을 통제하는 데 필요한 정보를 제공할 수 있을 것이다.

參 考 文 獻

1. Anton, H., *Elementary Linear Algebra*, New York : John Wiley & Sons, Inc., 1981.

2. Bradley, G.H., "Survey of Deterministic Networks", *AIIE Transactions*, Vol.7., 1975, pp.222-234.
3. Carré, B.A., "An Algebra for Network Routing Problems", *Journal of Inst. Math. Appl.*, Vol.7, 1971, pp.273-294.
4. Conte, S.D. and C. de Boor, *Elementary Numerical Analysis*, New York : McGraw-Hill, Inc., 1965.
5. Dreyfus, S.E., "An Appraisal of Some Shortest-Path Algorithms", *Operations Research*, Vol. 17, 1969, pp.395-412.
6. Ford, L.R. and D.R. Fulkerson, *Flows in Networks*, Princeton, N.J. : Princeton Univerity Press, 1962.
7. Gondran, M. and M. Minoux, *Graphs and Algorithms*, New York : Wiley Interscience, 1984.
8. Shier, D.R., "Computational Experience With an Algorithm for Finding the k Shortest Paths in a Network," *Journal of Res. Nat. Bur. Stand.*, Vol. 78B, 1974, pp. 139-165.
9. Shier, D.R., "Iterative Methods for Determining the k Shortest Paths in a Network", *Networks*, Vol. 6, 1976, pp.205-229.
10. Tarjan, R.E., "Algorithms for Maximum Network Flow", *Mathematical Programming Study*, Vol.26, 1986, pp.1-11.