□ 論 文 □

時間 展開形 네트워크를 利用한 線路의 最大흐름 스케줄링

A Maximal-Flow Scheduling Using Time Expaneded Network in a track

李 達 相 (東義大學校 產業工學科)

金 滿 植 (漢陽大學校 產業工學科)

目 次

ABSTRACT

This paper treats the problem to schedule for trains with low transit priority so as to maximizing the number that can be sent during given time periods without interfering with the fixed schedule for train with high transit priority in a track.

We transform the this problem into Time Expanded Network without traverse time through application of Ford and Fulkerson Model and construct the Enumeration Algorithm for solutions using TENET Generator (TENETGEN).

Finally, we compare our algolithm with Dinic's Maximal-Flow Algorithm and examine the availability of our procedures in personal computer.

Ⅰ. 序 言

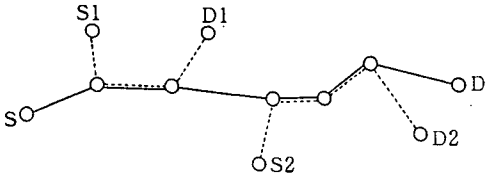
鐵道로 連結되어 있는 出發地 S와 目的地 D 사이의 物動量이 크게 增大되면, 列車 時刻表의 調整 혹은 線路의 增設에 따른 檢討가 必要

하게 된다. 특히 막대한 費用이 드는 線路의 增設을 檢討하게 될때는 필수적으로 現在 旣存 線路의 最大 흐름容量을 檢討하게 된다. 그러나 通過 優先順位가 다른 列車의 種類가 많고 中間의 線路의 線路가 다른 선과 겹치게 되는 境遇, 列車 時刻表의 作成은 勿論 旣存 線路의 最大흐름 容量을 把握하는 것도 쉽지 않다.

旣存線路의 最大 흐름 容量을 決定하는 重要한 因子로서는 通過 優先 順位가 다른 列車의 數, 다른 선과 겹치는 구간들의 다른 선 列車의 通行量, 各 驛과 驛사이의 運行時間, 安全을 위해 필수적인 各 列車사이의 最小限의 時間 間隔 등이 있다.

時間 展開形 네트워크를 利用한 線路의 最大 흐름 스케쥴링이란 다음과 같다. 出發地 S와 目的地 D 中間에 s개의 驛 혹은 簡易驛이 設置되어 있어, 通過 優先順位가 높은 列車가 通過하게 될 때는 通過 優先 順位가 낮은 列車가 비켜 줄수 있도록 대피할 수 있는 場所가 마련되어 있다. 또한 線路의 一部分은 다른선의 一部로 使用될수 있으며 〈그림 1〉, 이 境遇 그 구간을 通過하는 다른 列車 時刻은 정해져 있으며 通過 順位가 원 線路보다 높은 것으로 看做한다. 이 때 出發地 S와 目的地 D까지, 現在 通過 優先 順位가 높은 列車의 列車時刻表를 變更시키지 않으면서 日程 期間동안 보낼 수 있는 通過 優先 順位가 낮은 列車의 最大 흐름과 그때의 日程들을 구하는 問題이다. 이 問題는 動的 네트워크(Dynamic Network)의 最大 動的 流量 問題로 풀 수 있다.

本 硏究에서는 위 問題를 Ford-Fulkerson 模型을 應用하여 移動時間이 除去된 時間 展開形 네트워크(Time-Expanded Network : TENET)로 變換하고 TENET Generator를 開發, 使用하여 TENET의 data를 구한다. 또한 變換된 TENET의 構造的 單純性과 方向性을 利用하여 解를 찾는 簡單한 列擧 解法을 提示하고 이 列擧 解法과 Dinic의 最大 흐름 解法의 計算量(complexity)을 比較함으로 旣存의 解法보다 優數함을 보인다.



〈그림 1〉 다른 線의 一部로 使用되는 線路

Ⅱ. 最大 動的 流量 問題와 時間 展開形 네트워크(Maximal-Dynamic-Flow Problem : MDFP and Time-Expanded NETwork : TENET)

動的네트워크(dynamic network : DNET)란 호의 用量 및 費用 파라메타 이외에 호를 通過하기 위해 所要되는 時間, 즉 移動時間(travel time)이 있는 network를 말한다. 動的 network에서는 流量을 動的流量(dynamic flow)라고 하는데 最大 動的 流量 問題는 指定된 時間 T동안의 最大 動的 流量을 구하는 問題이다.

一般的으로 動的 流量 問題는 直接 動的 network(G)에서 最適化하는 方法[1]과 動的 network G를 移動時間의 파라메타가 除去된 時間展開形 network(TENET), G_T로 變換하여 이 變換된 network 에 정적 network의 解法을 適用하여 最適解를 구하는 方法[3,4]이 있다.

이 중 TENET는 network 構造(교점과 호 및 각 파라메타)가 時間的으로 變하는 複雜한 動的 network도 쉽게 表現될 수 있으며 특히 각 列車 間의 時間 間隔을 一定한 時間 이상 維持해야 하는 鐵道의 列車 scheduling 問題는 動的 network 보다는 時間 展開形 network이 다루기 쉽다.

DNET에서 TENET로 展開하는 過程은 다음과 같다.

G=(N, A)를 node 集合 N과 arc 集合 A를 가진 有方向 network라 하고 |N|= n 그리고 |A|=m이라 하자.

각호 $(x, y) \in A$에 대해 $C(x, y)$는 비음정수인

用量(capacity)를 나타내고 $a(x, y)$는 移動時間(arc—travel time)을 뜻한다.

또한 各各 $|S| = |T| = q \geq 1$를 갖는 the sources와 the sinks인 集合 $S \subset N$과 $T \subset N$가 주어지고 $S = (s_1, s_2, \cdots, s_q)$와 $T = (t_1, t_2, \cdots, t_q)$라 하면, $i = 1, 2, \cdots, q$에 대해 S_i에서 T_i로 들어가는 흐름 만이 고려의 對象이 된다.

時間 $t = 0, 1, 2, \cdots, p$에서 the network G상에 흐름이 發生한다면 最大 動的 흐름 問題는 p時間 동안에 the sinks $t_1, t_2, \cdots, t_p$에 到達하는 最大 흐름量과 이 흐름이 거치는 經路를 구하는 것이다.

이때 호의 用量과 移動 時間이 解에 影響을 주고 이것은 TENET로 가장 잘 나타내어 진다. TENET, $G_T = (N_T, A_T)$는 다음과 같은 方法으로 構築된다.

각 node $x \in N$에 대해 $t = 0, 1, 2, \cdots, p$에서 node $x(t)$를 定義하고 各 arc $(x, y) \in A$에 대해 $t = 0, 1, 2, \cdots, p-a(x, y)$에서 用量이 $c(x, y)$인 arc $(x(t), y[t + a(x, y)])$를 구한다. 마지막으로 $x \in S$ T인 모든 x에 대해, $t = 0, 1, 2, \cdots, p-1$에서 arc $(x(t), x(t+1))$을 부가하고 거기에 무한대의 用量을 할당한다.

그러면 network G_T는

$$|N_T| = (P+1) \times n \text{ 이고,}$$

$$|A_T| = \sum_{(x,y) \in A \text{ and } p-a(x,y)} [p+1-a(x, y)] + p(|S| + |T|)$$

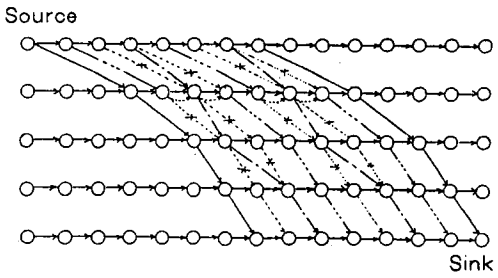
이다. 이때 P가 충분히 크면, 모든 (x, y) A에 대해 $P - a(x, y) \geq 0$ 이고, 따라서

$$|A| = m(p+1) - \sum_{(x,y) \in A} a(x, y) + 2pq$$

이 境遇 最大 動的흐름 問題는 G에서 $i = 1, 2, \cdots, q$에 대해 $S_i(0)$에서 $t_i(P)$로 가는 最大 흐름量을 구하는 것이다.

이에 대한 硏究로서 Ford and Fulkerson[8]은 주어진 P에 대해 G를 G_T로 變換하여 transhipment Problem을 풀어 解를 구하였고 Minieta[4]는 Ford and Fulkerson 앨고리즘을 修正하여 a latest-departure earliest-arrival 最大 動的흐름 模型을 構築하여 解를 구하였다.

이들 硏究의 移動 時間이 除去된 時間 展開形 네트워크에서는 遲滯를 피하는 最大 動的 流量이 항상 存在하기 때문에 이 遲滯弧에 부과된 容量은 重要하지 않았다. 그러나 통과 優先 順位가 높은 列車에 의해 旣存의 時間 展開形 네트워크가 修正되면 遲滯弧를 使用해야 하는 最大 動的 流量이 存在하게 되며, 이 예가 〈그림 2〉에 나타나 있다. 여기서 遲滯弧를 使用하지 않는 經路는 2개, 遲滯弧를 使用하는 經路가 2개로 總 4개의 經路를 갖는다.



——→: 遲滯弧를 使用하지 않는 經路
—·····→: 遲滯弧를 使用하는 經路
—·—·→: 통과 優先 順位가 높은 列車가 지나는 經路
--X--→: 통과 優先順位가 통과함으로 지워지는 旣存 列車의 經路

〈그림 2〉 遲滯弧를 使用하는 最大 動的 流量

Ⅲ. 線路의 最大 흐름 스케줄링을 위한 TENET 模型

1. 假 定

1) 出發地 S와 目的地 D는 한개의 線路로 되어 있다. 즉, S에서 D로 가는 經路는 하나 뿐이다.
2) 各 列車의 驛에서 出發과 도착時間 間隔은 安全을 위해 d 時間 이상이어야 한다.

3) 列車 k의 指点 s와 指点 s+1까지 移動時間 T는 d의 정수배이다.

4) 各 列車가 出發地 S를 떠나는 時間 亦是 d의 정수배이다.

5) 통과 優先 順位가 높은 列車가 한 指点을 통과할 때는 그 指点을 통과하는 優先順位가 낮은 列車는 대피하여 그 列車가 통과할 때까지 기다려야 한다.

2. 記號 說明

T : 指定된 時間

d : 各 列車가 維持해야 하는 最小한의 時間 間隔

s : 出發地와 도착지 사이의 驛의 數

n_i : 驛 i가 대피시킬 수 있는 통과 優先順位가 낮은 列車의 數 (i=1, 2, 3, ···, s+2)

$t_i^{(k)}$: 列車 의 驛 와 驛 사이 移動時間

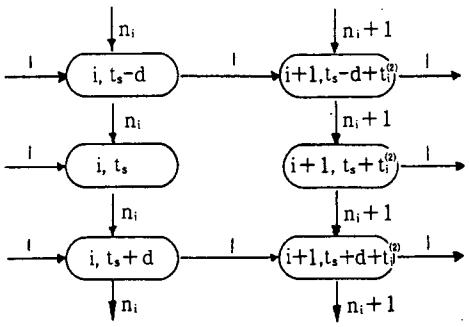
k=1 : 통과 優先 順位가 높은 列車

k=2 : 통과 優先 順位가 낮은 列車

出發地는 驛 1로 도착地를 驛 s+2로 하면 node (i, j)는 時刻 j에서의 驛 i(i=1, 2, ···, s, s+1, s+2 and j=0, d, 2d, ···, T/d)를 나타낸다. 특히 node (1, 0)은 the source, node (s+2, T/d)는 the sink를 意味한다.

오직 한 列車 (K=2)가 時刻 j에서 驛 i를 떠나서 驛 i+1에 도착하기전 K=1인 列車에 의해 추월되지 않을 境遇에만 node (i, j)와 node $(i+1, j+t_i^{(2)})$에 用量이 1인 arc $((i, j), (i+1, j+t_i^{(2)})$가 存在하게 된다. 또한 j=0, d, 2d, ···, T/d에 대해 node (i, j)와 node (i, j+d) 사이에 用量이 n_i인 arc ((i, j), (i, j+d)) 이 存在하며, 이외의 다른 arc는 存在하지 않는다. 예를 들어 K=1인 列車가 驛i를 t_s에 出發하여 驛 i+1에 닿기 전에 移動 時間 t_i를 가진 K=2인 列車를 추월할 수 있다면 〈그림 3〉은 TENET의 一部를 表現하게 된다.

이러한 構造를 修理 模型으로 作成하면 目的函數는 node (1, 0)에서 node (s+2, T/d)까지의 最大 흐름量을 구하는 것이 된다.



〈그림 3〉 TENET의 一部 構造

〈修理模型〉

Maximing

Subject to

$$\sum_{\langle u, v\rangle} X_{\langle l, m\rangle \langle u, v\rangle} - \sum_{\langle w, z\rangle} X_{\langle l, m\rangle} = \begin{cases} V & \langle l, m\rangle = S \\ 0 & \langle l, m\rangle = S, D \\ -V & \langle l, m\rangle = D \end{cases}$$

$$0 < -X_{\langle l, m\rangle \langle u, v\rangle} < -C_{\langle l, m\rangle \langle u, v\rangle}$$

여기서,

$\langle l, m\rangle$, $\langle u, v\rangle$, $\langle w, z\rangle$은 node를 表示

$l, u, v \in I = \{I : I = 1, 2, \cdots, s+2\}$

$m, v, z \in J = \{j : J = 1, 2, \cdots, T/d\}$

V : 구하고자 하는 總 流量

X_{ij} : arc (i, j)의 흐름量

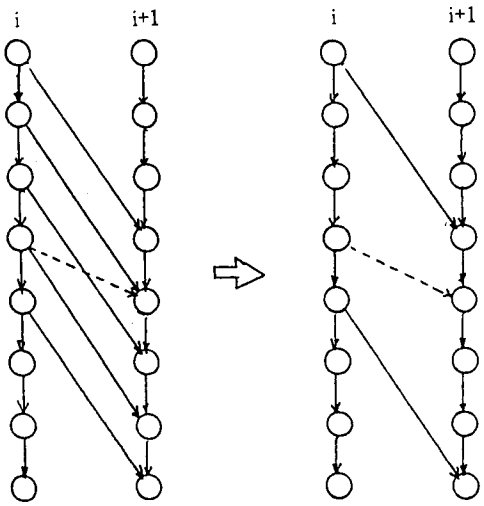
C_{ij} : arc (i, j)의 用量

Ⅳ. 時間 展開形 네트워크 Generator (TENET)

DNET G를 TENET G_T로 變換 할 때 指定된 時間 T가 커지면 變換 network G_T는 構造는 複雜하지 않으나 크기가 커지는 短点이 있다. 예를들어 出發地와 도착地를 包含하여 통과 指点이 20개이고 指定期間 T가 24시간, 時間間隔 d=0. 1일 境遇 總 node 수가 4, 800개 arc 수의 상한이 9, 597개인 network로 된다. 이처럼 큰 network를 手作業으로 入力시킨다는 것은 매우 불편한 일이며, 따라서 데이타를 簡單히 入力시킬 수 있는 方法을 구하게 된다. 이에 TENET

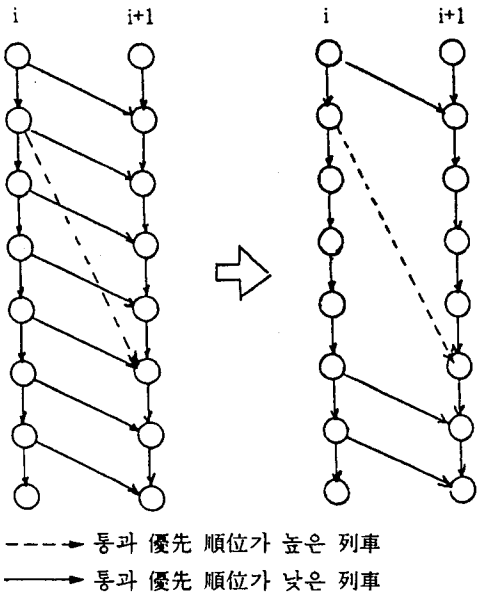
의 arc方向의 規則性을 利用하여 TENETGEN이라고 명명한 TENET Generator를 開發한다.

TENET Gernerator를 開發하기 위해 통과 優先 順位가 있는 線路의 最大흐름 問題에 있어 TENET의 構造의 規則性의 一部를 簡略히 그림으로 表示하면 〈그림 4〉, 〈그림 5〉, 〈그림 6〉, 〈그림 7〉과 같다. 〈그림 4〉는 驛 i와 驛 i+1를 통과하는데 걸리는 時間이 통과 優先 順位가 높은 列車가 통과 優先 順位가 낮은 列車보다 더 느릴 境遇, TENETGEN에서 일어나는 弧의 削除를 나타낸다. 이들 境遇 통과 優先 順位가 높은 列車에 대한 弧의 始点에서 나가는 弧와 終点으로 들어오는 弧들 중 遲滯弧를 除外한 모든 弧가 削除된다.

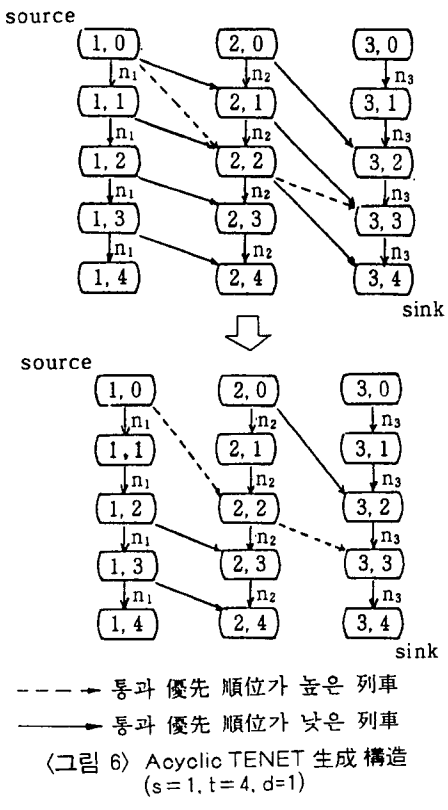


〈그림 4〉 $t_i^{(1)} < t_i^{(2)}$인 境遇의 호의 削除

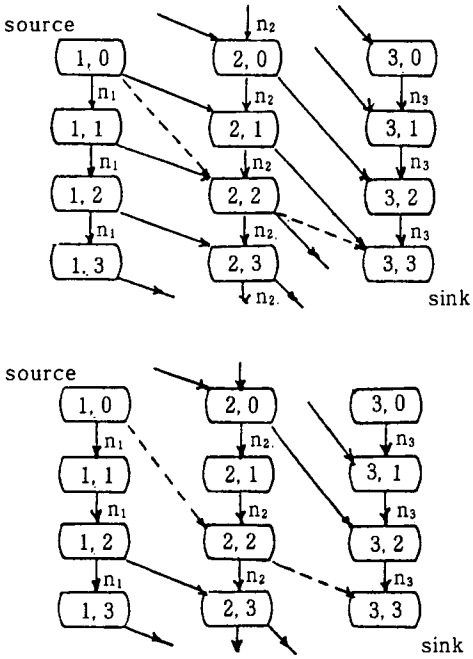
〈그림 6〉는 주어진 期間 T가 단속적으로 일어난 境遇에 生成되는 TENET의 形態이다. 이 境遇 source (1, 0)에서 sink (3, 4)까지의 經路 數는 0개이다. 즉, 4시간 동안 出發地 1에서 目的地 3까지 갈 수 있는 통과 우선 순위가 낮은 列車의 수는 0이다.



〈그림 5〉 $t_i^{(1)} > t_i^{(2)}$인 境遇의 호의 削除



〈그림 6〉 Acyclic TENET 生成 構造
(s = 1, t = 4, d=1)



〈그림 7〉 Cyclic TENET 生成構造
(s=1, t=4, d=1)

〈그림 7〉은 T가 繼續 連續 된다고 할때 生成되는 TENET의 形態를 나타낸다. 이 境遇 source (1, 0)에서 sink (3, 3)까지 生成되는 經路의 數는 2개이고 그 經路는 (1, 0)→(2, 1)→(3, 3), (1, 3)→(2, 0)→(3, 2) 이다.

즉 통과 優先 順位가 높은 列車의 時刻表를 變更시키지 않으면서 T=4 時間동안 통과 우선순위가 낮은 열차를 보낼 수 있는 최대의 수는 2대이고, 그때 출발지의 出發時刻은 0時, 3時이다.

TENETGEN의 基本的인 槪念은 각 node사이에 통과 優先 順位가 낮은 列車의 進行 arc를 發生시켜 놓고, 통과 優先 順位가 높은 列車의 進行 arc가 發生될때 TENET構造의 規則性을 따라 이미 生成된 통과 優先 順位가 낮은 列車의 進行 arc를 소거해가는 것이다. 여기에서 生成된 時間 展開形 네트워크(NENET)는 各 교점(node)에서 나오고 들어가는 호는 各各 最大 2개 이고 종점(sink)으로 향하는 단方向의 特性을 갖고 있다.

Ⅴ. 時間 展開形 네트워크를 利用한 線路의 最大 흐름 스케줄링을 구하는 節次

1) TENETGEN을 利用한 TENET의 發生
   段階 1 : 中間驛의 數(s), 計劃期間(T), 時間間隔(d), TENET의 種類 (1 : Acyclic Net, 2 : Cyclic Net)를 入力
   段階 2 : 各 驛에서의 用量 (n_i)를 入力 (i=1, 2, ⋯, s+2)
   段階 3 : 통과 優先 順位가 높은 列車와 통과 優先 順位가 낮은 列車의 各 驛間 運行時間 $T_i^{(1)}$, $T_i^{(2)}$를 入力 (i=1, 2, ⋯, s+1)
   段階 4 : 통과 優先順位가 높은 列車의 出發地 에서의 發車時間(PST)을 入力
   段階 5 : TENETGEN을 實行
2) 最大 흐름 스케줄링 決定
   1) 의 段階 5에서 發生된 DATA로 列擧 解法을 使用하여 解를 구함

Ⅵ. 列擧 解法 節次

[1] 段階 1
   (1.1) 出發地에서 優先順位가 낮은 列車의 出發 時刻에 대한 調査되지 않은 첫 교점에서 다음 目的地로 나가는 호가 있는지 調査한다.
   (1.2) 있으면 이 교점에 표지하고 이 호의 目的地를 교점 (i, j)라 하면 이 호의 容量을 1만큼 減小시키고 段階 2로 가고, 없으면 (1, 1)로 간다.
   (1.3) 出發地에서 調査되지 않은 교점이 있으면 끝낸다.

[2] 段階 2

(2.1) 교점(i, j)가 最終 目的地이면 段階 3으로 간다.

(2.2) 교점(i, j)가 最終 目的地가 아니면 교점 (i, j)에서 교점$(i+1, j+t_i^{(2)})$로 가는 호가 있는지를 調査한다.

(2.2.1) 만약 있으면 교점(i, j)에 표지하고 교점$(i+1, j+t_i^{(2)})$로 가는 호의 容量을 1만큼 減小시키고 $i=i+1$, $j=j+t_i^{(2)}$로 놓고 (2, 1)로 간다.

(2.2.2) 없으면 교점 (i, j)에 표지하고 교점(i, j+d)로 가는 호가 있는지 調査한다.

(2.2.2.1) 있으면 교점 (i, j+d)로 가는 호의 容量을 1만큼 減小시키고 j=j+d로 놓고 (2.1)로 간다.

(2.2.2.2) 없으면 지금까지 표지된 經路의 容量을 1만큼 增加시킴과 아울러 표지된 것을 모두 지우고 段階 1로 간다.

[3] 段階 3

(3.1) 經路의 數를 1만큼 增加시키고 지금까지의 표지를 모두 지우고 段階 1로 간다.

Ⅶ. 열거 해법과 Dinic 最大 流量 解法 比較

1. 計算量(complexity)의 比較

生成된 時間 展開形 네트워크(TENET)에서 構造的 單純性과 단方向性을 利用하여 經路들을 살펴 보면 source에서 sink로 가는 最大 經路의 數는 T/d 개이다. 이것은 各 驛間 cut의 最大 크기를 意味한다.

또한 各 經路는 最小 s+1개, 最大(s+1+T/d)개의 호로 構成된다.

따라서 列擧 解法의 최악의 計算量은(T/d)×(s+1+T/d)이며 O(N)로 表示할 수 있다. 이것은 Dinic 解法의 complexity, $O(N^2M)$보다 작고, Karzanov 解法의 complexity, $O(N^3)$, MPM 解法의 complexity, $O(N^3)$보다 작다.

이러한 理由는 위의 精巧한 解法들이 一般的인 복잡한 네트워크에 대해 效率的 解를 구하기 위해 追加로 計算을 더하기 때문이다.

2. 實驗의 施行과 結果

本 硏究에서 計算 實驗을 위하여 使用한 컴퓨터는 IBM PC/AT 호환기종을 使用하였다. (Qunix AT, 주기억 用量 : 1M, clock 速度 : 12.5 MHz, 80287 coprocessor 非採用)

한편 프로그램의 컴파일 및 링크過程에는 Microsoft FORTRAN77 Ver. 3.3 Compiler와 Microsoft 8086 Object Linker V. 3.04를 使用하였으며 DOS V. 3.2 하에서 施行하였다.

여기서 計算時間은 Macro-Assembler로 부프로그램을 作成하여 system時間을 測定하였으며, 이 때 入出力에 所要되는 時間은 배재하였다.

實驗을 施行한 結果가 〈表1〉에 要約되어 있다.

위 實驗에서는 本 硏究에서 開發한 列擧 解法과 Dinic의 最大 流量 解法을 比較, 檢討하기 위하여 PC에서 實行할 때 다룰 수 있는 最大 크기의 Network을 주로 다루었으며, 각 驛에서 列車를 대피시킬 수 있는 場所의 크기는 列車의 3~5배 사이에서, 各 列車의 各 驛間 移動時間은 d의 3~4배 내에서, 優先 順位가 높은 列車의 發生 數는 T/d의 4.5%에서 임의로 주었으며, 이 때 使用된 난수는 IMSL의 난수 發生 SUBROUTINE GGUBS를 利用하여 發生시켰다.

또한 實驗은 같은 데이타를 가지고 Acyclic net와 Cyclic net 두 境遇 모두를 행하였는 바 Cyclic Net의 境遇 生成된 path의 수가 많았고, 各 驛間의 移動時間이 작을 境遇 더 많은 path가 發生하였는데 이것은 當然한 結果이다.

表 1에서 보듯이 列擧 解法이 Dinic 解法보다 20~30배 精度 解를 구하는 速度가 빨랐다. 위 實驗 結果 중 最大 實行 時間은 Dinic 解法에서

43.50초인 反面 列擧 解法은 2.13초로 20배 精度 작고, 時間의 絶對값이 작으므로 本 列擧 解法의 PC에서 利用 可能性을 보여준 것이라 하겠다.

〈表 1〉 實驗의 結果

No	S	單位 : 時間		Network의 크기		優先順位 列車의 發生數	Network 의 種類	生成된 path 의 數	實行時間(초)	
		T	d	N	M				Dinic解法	列擧解法
1	1	5.0	1.0	18	20	1	A	0	0.00	0.00
				15	20		C	3	0.05	0.00
2	3	12.0	0.1	605	1017	6	A	89	4.56	0.22
				600	1018		C	96	5.88	0.22
3	5	12.0	0.1	847	1468	6	A	87	4.84	0.27
				840	1468		C	96	6.54	0.33
4	5	24.0	0.1	1687	2956	11	A	190	16.97	0.61
				1680	2953		C	196	19.98	0.72
5	10	12.0	0.1	1452	2633	6	A	87	5.93	0.33
				1440	2620		C	102	8.29	0.33
6	10	24.0	0.1	2892	5290	11	A	192	19.00	0.59
				2880	5265		C	207	23.84	0.60
7	15	12.0	0.1	2057	3764	6	A	69	6.54	0.44
				2040	3718		C	96	10.48	0.49
8	15	24.0	0.1	4097	7540	11	A	169	20.38	0.98
				4080	7478		C	196	28.45	1.04
9	21	24.0	0.1	5543	10357	11	A	154	19.83	0.88
				5520	10303		C	196	30.31	0.88
10	25	24.0	0.1	6507	12221	11	A	147	21.80	1.21
				6480	12113		C	196	34.32	1.21
11	31	24.0	0.1	7953	14939	11	A	130	24.83	1.81
				7920	14751		C	196	43.50	2.13

S : 中間驛의 數 T : Time periods d : 列車사이의 時間 間隔 N : node 數 M : arc 數

Ⅷ. 結 論

本 硏究에서는 線路의 最大 흐름 日程을 結定하는데 時間 展開形 network(TENET)模型과 TENET Generator(TENETGEN)을 開發, 使用하고 여기에서 얻어진 network의 構造的 單純性과 方向性을 利用하여 列擧 解法을 開發하였다. 이 列擧 解法과 Dinic의 最大 흐름 解法의 計算量(complexity)을 比較 檢討하고 PC의 利用 可能性에 대한 檢討를 위해 實驗을 해 본 結果, 列擧 解法이 Dinic의 解法보다 20~30배 精度 빠른 結果치를 얻었다.

이것은 Dinic解法이 network의 構造와는 무관하게 最大 流量을 구하기 위해 層(layer) network를 構成하는 바, 이에 消費되는 時間이 많았기 때문이라 사료된다. 또한 본 列擧 解法에서 차지하는 기억용량은 Dinic解法보다 50% 이상 작아서 훨씬 더 큰 規模의 問題를 다룰 수 있다.

앞으로 PC의 性能이 더욱 向上됨에 따라 TENET의 利用 可能性은 더욱 커질것이고 아울러 본 列擧 解法의 效用도 크리라 期待된다.

〈參 考 文 獻〉

1. 강 맹규, 네트워크와 앨고리즘, 한양대, 1988.
2. 이 달상, 김 만식, "통과 優先 順位가 있는 線路의 最大 흐름 問題," 工業經營學會誌, 제 13권 21집, pp. 111—118, 1990.
3. Ford, L. R., Jr. and D. R. Fulkerson, "Constructing Maximal Dyamic Flows from Static Flows," Opns. Res., Vol. 6, pp. 419—433, 1958.
4. Minieka, E., "Maximal Lexicographic, and Dynamic Network Flows," Opns. Res., Vol. No. 2, pp. 517—527, 1973.
5. Hillier, F. S and G. J. Lieberman, Introduction to Operations Research, Fourth Edition, Holden—Day, Inc., Oakland, 1986.
6. Goldfarb, D. and M. D. Grigoriadis, "A computational Comparison of the Dinic and Network Simplex Methods for Maximum Flow," Annals of Opns. Res., Vol. 13, pp. 83—123, 1988.
7. Bellmore, M. and R. R. Vemuganti, " On Multi—Commodity Maximal Dynamic Flows," Opns. Res., Vol. 21, No. 1, pp. 10—21, 1973.
8. Ford, L. R., Jr. and Fulkerson, Flows in Networks, Princeton University Press, Princeton, N. J., 1962.
9. Wilkinson, W. L., "An Algorithm for Universal Maximal Dynamic Flows in a Network," Opns. Res., Vol. 19, pp. 1602—1612, 1971.