마이크로 컴퓨터를 利用한 最小費用 넽웍문제의 解法*

Design and Implementation of Minimum Cost Network Programs on a Microcomputer

梁 光 敏**

《目 次》

Ⅰ. 序 論

現在 사용되고 있는 컴퓨터는 中·大型컴퓨터가 大宗을 이루고 있으나 技術革新에 따라 급격히 컴퓨터가 小型化, 에너지節約型化하고 특히 低價格化함에 따라 그 普及이 普遍化하고 있다. 그러나 情報處理에 所要되는 비용中 主된 부문은 하드웨어 보다는 소프트웨어의 購入一開發에 소요되는 비용이며 이 比率은 時間이 經過함에 따라 더욱 加重될 것으로 보여진다. 外國의 경우 旣存 소프트웨어는 中·大型컴퓨터를 對象으로 開發된 것이나 최근에 와서는 小型個人用 컴퓨터를 위한 소프트·웨어의 開發이 급속히 진척되고 있는 實情이다. 이러한 小型컴퓨터를 대상으로한 소프트웨어의 開發은 대규모기업이나 硏究所에 의해 組織的이고 體系的으로 이루어지기 보다는 個人이나 小規模 소프트웨어 회사에 의해 商業的 목적으로 이루어지는 것이 일반적이다.

본 硏究에서는 問題의 屬性으로 因해 大型컴퓨터를 이용해야만 했던 數理計劃(mathematical programming) 문제를 小型個人用컴퓨터에서 그 解를 구해보고자 하는 試圖이고 이는 현재 部分的으로 硏究가 진행되고 있다.

* 本 硏究는 1982年度 峨山社會福祉事業財團의 硏究費 支援에 의한 것임.
** 中央大學校 經營大學 敎授

技術向上에 의해 個人用컴퓨터가 容量·性能面에서 하루가 다르게 발전하므로 數理計劃 등과 같은 計量經營技法도 必然的으로 個人用컴퓨터에 의해 해결될 것으로 보여진다. 이러한 經營意思決定技法을 中·大型이 아닌 個人用컴퓨터에서 푸는 데는 여러가지 문제가 고려되어야 한다. 解法上의 문제(algorithm)는 어퍼레이션스·리서치(operations research)의 문제이며 컴퓨터에의 實際履行(implementation)의 문제는 컴퓨터科學의 문제이다. 資料構造(data structure)가 전체의 效率을 좌우하는 例는 많이 볼 수 있다. 따라서 이와 같은 計量經營意思決定문제를 個人用컴퓨터에 의해 效率的으로 풀기 위해서는 經營科學과 컴퓨터科學의 學際的인 接近에 의해서 만이 가능하다고 본다.

本 연구는 적은 容量의 個人用컴퓨터를 사용하여 解를 구할 수 있는 計量經營문제의 選定과 이에 대한 效率的인 解法의 開發-實際履行을 통해 中·大型컴퓨터와 그 性能을 비교하는 것이 課題이다. 硏究에 사용될 컴퓨터의 選定과 適用 對象문제의 선정은 相互 聯關되는 문제이나 본 연구에서는 현재 우리나라에 널리 보급되어 쉽게 接近할 수 있고 性能面에서 적합한 機種을 우선 選定하고 이 시스템을 기준하여 적합한 적용대상문제를 선정하여 大型컴퓨터를 사용했을 경우와 성능을 비교분석하고자 한다.

사용 컴퓨터의 選定은 현재 우리나라에서 組立·生產되는 個人用 또는 小型컴퓨터의 機種이 제한된 관계로 비교적 용이한 과제이다. 우선 적용문제의 성격상 문제크기(input size)와 성능비교에 따른 문제發生裝置(problem generator)의 사용이 不可避한 관계로 빠른 副記憶裝置(secondary storage)를 具備해야 함이 필수적이고 동시에 휴대성(portability)을 고려하여 널리 보급된 運營시스템(operating system), 즉 현재 소형컴퓨터의 사실상의 標準運營시스템인 CP/M을 탑재한 機種으로 제한했다.

적용대상문제의 선정은 여러가지 計量經營문제中 數理計劃문제가 그 解法이 反復的이라는 점에서 컴퓨터의 特性을 충분히 발휘할 수 있는 적합한 문제라고 판단된다. 數理計劃문제에서는 많은 量의 計算時間과 더불어 문제의 크기가 증가함에 따라 다루어야 할 자료量의 증가를 감당하는 것이 어려운 점이다. 특히 현재 보급되어 있는 個人用 내지 小型컴퓨터는 小數處理장치(floating-point processor)가 구비되어 있지 않은 것이 대부분이므로 小數를 多量으로 처리해야 하는 문제에는 性能비교面에서 적합치 않다고 판단된다. 따라서 적용문제의 選定은 數理計劃문제 中 극히 제한된 小數演算만으로 解를 얻을 수 있고 또한 그 응용例가 광범위한 最小費用넷웍(minimum cost flow network) 문제를 선정하여 본 연구의 대상으로 했다.

문제의 解法面에서는 컴퓨터科學의 發展과 더불어 1970년代에 原單體法(primal simplex method)을 이용한 解法에 많은 진전이 있었다. Glover, Karney와 Klingman[12]은 輸送問題에 三分類(triple label) 표현을 이용하였고, Langley는 上限流量을 갖는 수송문제에 原單體法 三分類法을 적용하였다.[12] 1977年 Bradley, Brown과 Graves[5]는 이 방법에 의

한 基底表現을 이용한 原單體法을 연구하였다. 本 硏究는 Bradley의 연구결과를 土臺로, 마이크로컴퓨터에서 履行할 수 있는 코오드의 設計 및 大型컴퓨터와 性能을 비교하려는 試圖이다.

II. 模型과 解法

1. 模 型

最小費用 模型은 輸送費用을 最小化시키기 위하여 어느 弧線(arc)을 通하여 얼마나 많은 量을 수송하여야 하는가를 결정하는 문제이다. 즉 流量의 上限과 下限을 만족시키고 또한 각 交點(node)의 外部流量에 관한 制約을 만족시키면서 총수송비용을 최소로 하는 流量(x_k)을 결정하는 문제인 것이다.

이는 다음과 같이 數理計劃模型으로 표현할 수 있다.

$$\text{Minimize} \quad \sum_{k\in A} c_k x_k$$

$$\text{subject to} \sum_{k\in A \text{ with tail}_i} x_k - \sum_{k\in A \text{ with head}_i} x_k = b_i, \quad i\in N, \quad l_k \le x_k \le u_k, \quad k\in A,$$

여기서 b_i가 陰數이면 需要交點이고, 陽數이면 供給交點이며 0인 경우는 단지 流量을 仲介만하는 交點을 뜻한다. N은 交點의 集合을, A는 弧線의 集合을 表示하며, c_k는 弧線 k의 單位當 수송비용을, l_k 및 u_k는 弧線 k가 許容할 수 있는 流量의 上·下限을 각각 나타낸다.

위의 數理模型은 넷웍에 나타나있는 弧線에 대한 資料만을 저장·기억시키고 演算도 이들 弧線에 관한 자료의 利用만으로 隨行될 수 있음을 강조한 模型이다. 위와 같은 模型을 사용하는 이유는 大規模문제에서 한개의 弧線에 연결된 모든 交點雙들을 表現한다는 것이 不適當하고 특히 實際的인 문제에서 흔히 볼 수 있는 한雙의 交點에 多數의 弧線이 존재할 때는 더욱 부적당하기 때문이다.

이 模型은 多樣한 문제에 적용될 수 있고 특히 많은 變數를 가진 문제에는 餘他 最適化 技法보다도 效率的인 컴퓨터 코오드를 만들 수 있기 때문에 최근에 이 모형에 대한 관심이 集中되고 있다.

2. 解 法

넷웍模型은 각 弧線이 變數이고 각 交點이 制約式으로 구성된 線型計劃문제로 보아 解를 구할 수도 있으나 大規模問題에 商用 線型計劃코오드를 사용했을 경우 넷웍문제 解法보다 資料 저장과 計算時間面에서 보다 많은 空間을 필요로 한다. 따라서 넷웍문제를 線型計劃

의 特別한 경우로 다루기 보다는 線型計劃의 表現과 數學的 이론의 전개에 있어 다른 연구 분야로 다루는 것이 요즘의 경향이다. 이는 넷웍문제에 대한 접근方法이 圖表理論(graph theory)에 기반을 두는 이유 때문이다.

먼저 일반적인 線型計劃문제의 解法에 대하여 說明하고, 이를 資料構造가 特異한 넷웍模型에 적용시켜 설명코자 한다.

우선 限界變數單體法의 수학적인 사항을 약간 언급하도록 한다. 일반 單體法의 演算量은 制約式의 數에 의하여 결정된다. 반면에 限界變數 單體法은 限界流量制約式을 一般制約式에 포함시키지 않고 最適解를 導出할 수 있도록 單體法을 修正하여 演算量과 연산시간을 節減시킨 방법이다.

限界變數模型의 代數的 표현은 다음과 같다.

$$\begin{aligned} &\text{Minimize} \quad \boldsymbol{cx} \\ &\text{Subject to} \quad \boldsymbol{Ax}=\boldsymbol{b} \\ &\qquad\qquad \boldsymbol{l}\leq\boldsymbol{x}\leq\boldsymbol{u} \end{aligned}$$

여기서 $\boldsymbol{c}$는 費用, $\boldsymbol{x}$는 流量, $\boldsymbol{A}$는 技術係數, $\boldsymbol{b}$는 外部流量, $\boldsymbol{l}$과 $\boldsymbol{u}$는 각 각 流量의 上下限을 나타낸다. 變數(流量)의 下限은 變數變換에 의해 쉽게 除去될 수 있고 上限도 基底變數가 上限에 도달했을 때 變數變換과 右邊의 變換에 의해 모두 默示的으로 表現할 수 있다.

즉

$$\begin{aligned} &\text{Minimize} \quad \boldsymbol{cx} \\ &\text{Subject to} \quad \boldsymbol{Ax}=\boldsymbol{b} \\ &\qquad\qquad \boldsymbol{x}\geq\boldsymbol{0} \end{aligned}$$

로 표현할 수 있다.

위의 行列 $\boldsymbol{A}$를 $\boldsymbol{A}=[\boldsymbol{B},\ \boldsymbol{N}]$로 基底 $\boldsymbol{B}$와 非基底 $\boldsymbol{N}$로 分割한다. 實行可能基底가 주어지면 唯一한 非基底가 존재한다. 현재의 基底變數가 $\boldsymbol{x}_B$, 非基底변수를 $\boldsymbol{x}_N$라 하면 제약식은

$$\boldsymbol{Ax}=[\boldsymbol{B},\ \boldsymbol{N}]\begin{bmatrix}\boldsymbol{x}_B\\ \boldsymbol{x}_N\end{bmatrix}=\boldsymbol{Bx}_B+\boldsymbol{Nx}_N=\boldsymbol{b} \tag{1}$$

로 표현된다. $\boldsymbol{x}_N=0$이므로 $\boldsymbol{Bx}_B=\boldsymbol{b}$가 된다. 基底變數 $\boldsymbol{x}_B$의 값을 $\hat{\boldsymbol{x}}$라고 하면

$\hat{\boldsymbol{x}}_1=\boldsymbol{B}^{-1}\boldsymbol{b}$가 된다. 또한 $\boldsymbol{B}$가 基底이기 때문에 唯一한 變數變換行列 $\boldsymbol{Z}$가 존재하는데 이는 다음 식을 만족시켜야 한다.

$$\boldsymbol{BZ}=\boldsymbol{N}$$

윗式에 x_N을 곱한 후 $B\hat{x}=b$에서 빼면

$$B(x_B-Zx_N)+Nx_N=b \tag{1}$$

가 된다.

윗式을 (1)에서 빼면

$B(x_B-[\hat{x}-Zx_N])=0$이 되고 B는(基底이므로) 線型獨立이기 때문에

$$x_B=\hat{x}-Zx_N$$

이 되고, 一般解는

$$x=\begin{bmatrix}x_B\\x_N\end{bmatrix}=\begin{bmatrix}\hat{x}-Zx_N\\x_N\end{bmatrix}$$

로 쓸 수 있다.

A와 같은 방법으로 C도 分割하면 目的함수는

$$cx=(c_B,\ c_N)\begin{bmatrix}x_B\\x_N\end{bmatrix}=c_Bx_B+c_Nx_N=c_Bx_B$$

이다. 따라서 일반적인 目的함수의 값은

$$\begin{aligned}cx=c_Bx_B+c_Nx_N&=c_B(\hat{x}-Zx_N)+c_Nx_N\\&=c_B\hat{x}+(c_N-c_BZ)x_N\\&=c_B\hat{x}+(c_N-uN)x_N\end{aligned}$$

위에서 u는 單體乘數 또는 雙對解로

$$uB=c_B$$
$$u=c_BB^{-1}$$

의 관계가 있다.

다음은 넬웡문제에 一般 原單體法을 적용할 때 特殊한 點에 관해 논의하고자 한다.

일반 線型計劃문제와 달리 仲介地經由수송문제와 같은 넬웡문제에 效率的인 解法을 設計할 수 있음은 基底가 置換에 인한 三角分割(triangulation)에 의하여 三角行列(triangular matrix)로 表現될 수 있다는 사실 때문이다. 이러한 三角基底表現의 長點은 樞軸反復演算(pivot)의 각 단계에서 改善된 解를 計算함에 있어 새로운 基底를 간편하고 효율적인 再三角分割(retriangulation)에 의해 구할 수 있다는 데 연유한다.

基底 B가 右上 三角行列(upper triangular matrix)일 때 B는 行의 數와 階數(rank)가 동일하고 各列은 한개의 1 또는 -1로 구성되거나 1과 -1로 구성된다. 넬웡문제의 제약식行列 A로 부터의 基底 B는 항상 行과 列의 再配列에 의해 三角行列로 變形될 수 있음은 잘 알려진 사실이다.

基底는 다음의 그래프(graph)로 정의될 수 있다. 三角基底 B의 行의 配列을 $I=\{i_1, i_2, \cdots, i_m\}$으로 표시하자. 각 交點 i_k와 관련하여 $p(i_k)$는 三角基底의 k번째列에서 非對角線元素의 行번호라고 하자. 만일 對角線을 제외한 곳이 모두 零이면 $p(i_k)$는 $m+1$이 된다. 따라서 그래프는 交點 1, 2, $\cdots$, $m+1$과 $(i_k, p(i_k))$, $k=1, 2, \cdots, m$ 즉 i_k로부터 $p(i_k)$로 가는 弧線으로 구성된다. 各 交點 i_k는 $h<k$일 때 교점 $p(i_k)=i_h$와 연결되거나 교점 $m+1$에 연결된다. 따라서 각 交點으로부터 交點 $m+1$까지 方向이 있는 經路가 존재하게 된다. 이 그래프를 交點 $m+1$이 뿌리인 뿌리있는 樹木形圖(rooted arborescence graph)라고 부른다.

交點 i에 대해 $p(i)$는 i의 前置點(predecessor)이라 부르고 뿌리있는 樹木形圖를 前置圖라 하며 이는 解의 計算을 위한 資料構造로 利用된다. 前置函數 $p()$는 樹木形을 간략하게 표현하는 도구이며 이는 한 交點으로 부터 뿌리까지의 唯一한 經路를 발견하기 위하여 反復計算하는데 이용된다. 이 經路를 後退經路(back path)라고 부르며 이들 經路上의 交點을 除外한 모든 交點을 後置點(successor)이라고 한다. 三角分割과 前置圖와의 관계는 두개의 非零元素에 대하여 各行은 交點을, 各列은 弧線을 각각 表示하는 것이 된다. 따라서 넬웡線型문제에서 各 交點은 等式制約으로 表現되고, 交點 i에서 j로 연결되는 弧線 k는 i行에 1, j行에 -1, 그 이외는 0으로 구성되는 列과 대응되는 變數로 표현된다.

單體法과 넬웡解法의 演算절차를 서로 관련시켜 설명하기 위해 基底 B는 右上 三角行列이고, i번째 行은 交點 i와 대응되며 對角線元素들은 모두 1이라고 가정한다. 만약 대각선元素가 -1인 경우는 變數變換을 통해 이미 1로 變換시켰다고 간주한다.

위의 가정下에 넬웡연산 절차를 다음에 서술한다.

단계 1(進入變數의 決定) : 單體乘數 u가 주어지면 非基底弧線에 대한 進入變數는 $c_k - u_i + u_j$로 計算된다.

단계 2(比率計算) : 基底에서 나가는 弧線을 定하기 위해 $BZ^k=N^k$를 Z^k에 관하여 푼다. B가 三角基底이므로 $(B^{-1})^j$는 간단히 구할 수 있으며 $(B^{-1})^j$의 $j+1, j+2, \cdots, m-1, m$번째의 元素는 0이 되고 j번째 元素는 1이 된다. 前置圖위에서 交點 j에서 뿌리까지의 後退經路上에 있는 모든 交點에 대응하는 行들은 $(B^{-1})^j$에 원소 1을 갖고 그 이외는 모두 0이다. 이 $(B^{-1})^j$는 前置函數 $p()$로 부터 바로 구해진다. $(B^{-1})^i$와 $(B^{-1})^j$를 구한 후 Z^k는 $(B^{-1})^i-(B^{-1})^j$이므로 쉽게 구해진다. 즉 $(B^{-1})^i$만이 1일 경우는 1, $(B^{-1})^j$만이 1일 경우는 -1, $(B^{-1})^i$와 $(B^{-1})^j$가 모두 1이거나 0일 경우는 0이다. Z^k의 演算은 더하

기나 빼기만으로 계산이 되며 더우기 交點 i와 j에서 뿌리까지의 후퇴경로上의 공통交點을 배제함으로써 더욱 계산 量을 줄일 수 있다.

단계 3(樞軸反復演算) : Z^k를 이용하여 解를 變換한다. 이를 위해 單體乘數 $\boldsymbol{u}$를 다음과 같이 구한다.

$$\tilde{\boldsymbol{u}}=\boldsymbol{u}+\lambda(\boldsymbol{B}^{-1})_i$$

새로운 基底列을 $\boldsymbol{N}^k$라고 하면

$$\begin{aligned}\tilde{\boldsymbol{u}}\boldsymbol{N}^k&=\boldsymbol{u}\boldsymbol{N}^k+\lambda(\boldsymbol{B}^{-1})_i\boldsymbol{N}^k=\boldsymbol{u}\boldsymbol{N}^k+\lambda(\boldsymbol{B}^{-1})_i\boldsymbol{B}\boldsymbol{Z}^k\\&=\boldsymbol{u}\boldsymbol{N}^k+\lambda Z_i^k=\boldsymbol{c}_k\\\lambda&=(\boldsymbol{c}_k-\boldsymbol{u}\boldsymbol{N}^k)/Z_i^k\end{aligned}$$

따라서 單體乘數 $\boldsymbol{u}$는 각 단계마다 計算하는 것 보다는 變換하는 것이 쉽다. $\boldsymbol{B}$의 l번째 列이 進出弧線이라 하면 $(\boldsymbol{B}^{-1})_i$를 구성하는 원소들은 l번째 원소의 1과 進出弧線이 交點 i에서 뿌리까지 後退經路上에 있을 때 원소 j의 1을 除外하고는 모두 0이다. 前置圖上에서 $(\boldsymbol{B}^{-1})_l$의 元素들은 交點 l과 그 後置點에 대하여는 1이고 나머지는 0이다. 따라서 $(\boldsymbol{B}^{-1})_i$이 0과 1로 이루어지므로 $\boldsymbol{u}$의 變換은 u_i과 l의 後置點의 $\boldsymbol{u}$에 더하면 된다.

위에서 설명한 바와 같이 基底의 變換이 行列의 數値演算에 의해서가 아닌 基底의 副樹木形으로 剪定(pruning)과 再配列(rehanging)을 통하여 효율적으로 수행되므로 문제의 크기가 대규모인 넬웍문제도 거의 實時間에 풀릴 수 있는 연유가 되는 것이다.

III. 實際履行

위의 解法을 컴퓨터 코오드로 만드는데는 다음의 사항이 고려되어 設計되었다.

1) 사용된 컴퓨터 言語는 異機種間의 휴대성을 갖도록 하기 위해 高級言語로 작성하고 實行속도를 고려하여 interpreter 言語보다 compiler 言語를 사용한다.

2) 資料構造의 選擇은 필요 記憶裝置量보다 實行速度面에 比重을 두어 選定한다.

3) 일반적인 넬웍문제를 풀 수 있도록 設計하고 문제에 대한 特殊性은 利用者가 入力하지 않고 코오드가 自體 認識토록 한다.

첫번째 고려사항과 관련하여 사용언어로는 Microsoft 社의 FORTRAN-80을 선정하였으며 두번째의 記憶장치 對 時間(space vs. time)의 문제는 앞으로의 個人用컴퓨터의 기억장치용량이 급속히 증가하는 추세에 따라 實行速度面에 置重하는 資料構造를 택했다. 넬웍코오드가 일반線型문제와 달리 빨리 풀릴 수 있는 理由는 基底의 表現이 樹木形으로 될수 있고 樹木形을 變換(update)하는 效率的인 알고리즘을 설계할 수 있기 때문이다. 基底樹木形

表現에는 linked list 構造가 사용됐다.

문제의 入力은 Klingman[18]의 NETGEN에서와 마찬가지로 SHARE format을 따르도록 했다. SHARE format은 두 種類의 카드로 되어 있다.

card 1 交點數

card 2 弧線데이터

(弧線名), (始點), (終點), (費用), (上限), (下限), (初期流量)

여기서 弧線名 및 初期流量은 임의 資料이다. 弧線의 個數는 end of file을 만날 때까지 레코드 數를 세므로서 認知하도록 되어있다. 읽혀지는 弧線의 順序는 無關하며 負의 費用도 許容하도록 되어 있다. 이는 최소비용 넬웍으로 轉換될 수 있는 여러가지 문제(例로 最大流量, 最長거리 등)도 취급할 수 있도록 하기 위함이다.

현재 가장 널리 보급되고 있는 個人用 또는 小型컴퓨터는 8-bit 컴퓨터가 大宗이라고 보아 機械는 8-bit 機種에 64 KB RAM을 사용하는 것으로 보고 코오드를 考案했다. 데이터 領域은 |A|를 弧線數, |N|을 交點數로 표시할 경우 4|A|+8|N|이 필요하여 64 KB 중 運營시스템(56 K CP/M)이 占有하는 부분, 所要 I/O buffer 영역을 감안한 후 풀 수 있는 문제의 크기는 交點이 430개, 弧線이 2010개의 문제로 限定되어 있다. 여기서 사용된 單語(word)는 모두 integer*2 type으로서 사용. 메모리가 許容되는 限 交點 또는 弧線의 數는 32,767까지 가능하다. overflow를 고려하여 目的函數(費用)만이 double precision(8 byte)으로 表示된 唯一한 實變數이다.

코오드는 네개의 모듈로 구성되어 있다. 卽 driver, input, output 및 solver이다. 따라서 大規模문제에 넬웍문제가 副문제로 되어있는 경우 서브루틴으로 이용할 수 있도록 考案했다. input module에는 弧線를 順序대로 sort하는 과정이 포함되어 있다. 計算實驗에서의 隨行시간은 위의 入·出力을 除外한 solver module에서의 所要시간만을 測定한 것이다.

기억장치 容量의 制限과 built-in function을 사용하기 위해 2-byte 整數를 사용하였다. 使用된 MS-FORTRAN 80는 負의 整數表現에 two's complement를 사용하므로 數式評價(expression evaluation)에서 underflow 또는 overflow가 생길 경우 이를 체크할 수 있도록 되어 있지 않아 豫期치 않은 결과가 나온다는 것을 아는데 오랜 時間을 허비했음을 밝혀둔다.

IV. 計算實驗

國産 APPLE-Ⅱ 互換機種, 運營시스템으로 CP/M v. 2.2, 프로그래밍言語로 Microsoft 社의 FORTRAN-80를 사용하였으며 實行速度를 위한 下級言語는 일체 사용치 않았다.

作成된 코오드의 性能시험을 하기 위해 문제發生器(problem generator)를 사용하였다.

이 發生器는 Klingman[18]의 NETGEN을 약간 修正한 것으로 여기서 사용된 亂數發生器는 FORTRAN-80의 built-in function인 RAN(X)가 사용되었다. 따라서 문제發生器의 異機種間의 휴대성은 결여되나 APPLE 및 FORTRAN-80를 사용한 경우는 항상 同一한 문제를 生成시킬 수 있어 다른 解法과의 性能비교에 이용될 수 있도록 했다. 휴대성이 保障되는 Schrage[24]의 亂數發生器의 사용도 고려해 보았으나 8-bit 機器의 경우 휴대성을 보장하기 위해서는 double precision을 사용하여야 하며 이 경우 亂數 1000개 生成에 소요되는 時間만도 實驗結果 6分이상이 되어 휴대용 亂數발생기는 時間소요 面에서 적당치 않다고 판단되어 사용치 못했다.

이 문제發生器는 記憶용량의 제약으로 交點數는 1500까지, 弧線數는 使用 diskette의 容量이 126 KB로 限定되어 최대 2800개 까지의 문제를 生成할 수 있게 設計되어 있다. unformatted I/O를 사용하여 弧線數를 약간 늘리는 것은 가능하다.

性能시험을 위하여 다루어진 문제는 Klingman[18]의 연구에서 쓰여진 문제와 同一規模의 문제들을 選定하여 計算實驗에 사용했다. 係數까지 同一한 문제를 생성하지 않는 것은 사용 亂數發生器의 非效率性과 2-byte 整數의 사용으로 總供給流量 등을 表現할 수 없었기 때문이다(2-byte 整數를 사용할 경우 表現가능한 最大整數는 $2^{15}=32767$로 制限된다). 물론 4-byte 整數의 사용이 許容안되는 것은 아니나 이는 제한된 기억용량과 固有 function이 2-byte argument를 취하는 것에 연유한다).

프로그램 隨行時間의 측정은 사용된 컴퓨터에 時計(real-time clock)[6]가 裝着되어 있지 않은 관계로 프로그램中에 鍾(beeper)[20]을 삽입하여 鍾이 울릴때 마다 스톱워치를 눌러 經過時間을 測定하는 手動式을 채택하였다. 誤差의 범위는 일秒미만이므로 性能비교의 目的으로는 타당하다고 본다.

大型컴퓨터와의 實行속도面의 비교는 이미 출판된 자료가 현재의 문제發生器가 生成한 문제와는 同一하지 않아서 엄밀한 比較는 아니나 〈表-1〉에서 보는 바와 같이 解에 요하는 시간이 大型컴퓨터와 비교될 수 있을 정도이며 또한 문제의 크기에 따라 거의 線型으로 增加하는 것이 特記할 만한다. 交點數가 200개 미만의 문제는 거의 實時間(10秒미만)에 풀 수 있어 실제 문제에 응용범위가 넓다고 보겠다.

V. 結　　論

現在까지 經營意思決定의 한 도구인 數理計劃문제가 모두 大型컴퓨터에서만 다루어져 왔으나 본 연구에서는 技術革新에 의해 컴퓨터의 價格이 하락하고 個人用컴퓨터가 보편화하는 추세에 따라 이와같은 문제를 小型個人用 컴퓨터를 사용하여 解를 구하도록 하는 것이 연구의 목적이었다.

〈表 1〉 計算實驗結果

Prob*	Nodes*	Source*	Sink*	Arcs*	Total Supply	Transship		Percent		Upper		Sol'n Time*	Boeing**
						Source	Sink	Cost	Capac	Min	Max		
1	40	20	20	260	20,000	0	0	0	0	0	0	2.7	NA
2				300								3.3	
3				400								3.0	
4	60	30	30	630	30,000							6.3	
5				900								7.7	
6	80	40	40	300	40							5.6	
7				450								6.3	
8				600								7.5	
9				750								8.8	
10				900								8.6	
11		8	30	262	30,000			30	20	3,200	6,000	5.4	
12				488								7.3	
13				262						4,000	24,000	5.5	
14				488								7.3	
15				283		5	25		40	3,200	6,000	6.2	
16				567								8.3	
17				283						4,000	24,000	5.5	
18				567								7.9	
19		4	6	276		0	0		80	3,200	6,000	4.3	
20				535								5.3	
21				276						4,000	24,000	3.7	
22				535								5.3	
23	200	10	10	580				0	0	0	0	13.1	
24				680								16.4	
25				880								17.9	
26				960								16.6	
27		100	100	1,300	10,000							26.9	30.25
28				1,501								31.5	21.59
29				2,000								39.7	31.47
30				2,008								36.6	NA
31	300	15	15	868	30,000							23.0	
32				877								24.4	
33				1,021								27.9	
34				1,146								24.8	
35	400	200	200	1,500	200							1.03.4	30.39
36		8	60	1,306	20,000			30	20	800	1,500	46.4	42.13
37										1,000	6,000	43.6	20.52
38				1,416		5	50		40	800	1,500	45.8	20.21
39										1,000	6,000	43.2	21.36
40		4	12	1,382		0	0		80	800	1,500	44.2	20.81
41										1,000	6,000	36.9	12.60
42	435	20	30	2,000	30,000	10	20	0	0	0	0	49.5	NA

註) 모든 문제의 弧線비용은 1에서부터 100 사이의 一樣亂數로 주어짐.
문제 발생기의 亂數種子는 2223임
* I/O를 제외한 시간으로 測定단위는 秒임.
** CDC 6600에서 FORTRAN RUN 컴파일러로 계산실험한 결과임[18].

小型컴퓨터의 記憶容量의 制限性, 實行速度面의 제약 등을 감안하여 여러 數理計劃中 넬웡문제를 選定하여 硏究對象으로 했다. 이는 넬웡문제가 다른 數理計劃문제와 달리 整數演算만으로도 그 解를 구할 수 있다는 점과 간편한 基底의 表現, 또 그 變換의 容易性 등의 利點에 연유한다.

考案된 코오드의 效率性을 立證하기 위해 無作爲로 生成된 문제를 國產 個人用컴퓨터를 사용하여 푼 결과 400개의 交點과 2000개의 弧線을 갖는 문제의 解를 60秒미만에 구할수 있었다. 일반적으로 같은 크기의 實際문제에 비해 無作爲로 生成된 문제의 解를 구하는데 더 많은 시간을 要한다는 계산경험에 비추어 볼 때 매우 고무적인 결과이었다.

이와같은 계산결과의 成果는 數理計劃문제中에서 最小費用넬웡문제는 小型컴퓨터를 사용하여 實際 문제에 적용할 수 있음을 보여준다.

본 연구에서 개발한 코오드는 最小費用넬웡문제는 물론이고 볼록費用(convex cost)을 갖는 넬웡문제, 最大流量문제(maximum flow problem), 最短거리문제(shortest path problem) 및 最長거리문제(longest path problem)에도 적용될 수 있도록 했다.

참 고 문 헌

E. Balas, P.L. Hammer, "On the Transportation Problem-Part I," *Cahiers du Centre d'Etudes de Recherche Ophrationelle*, Vol.4, No.2(1962), p.98.

R.S. Barr, F. Glover, D. Klingman, "An Improved Version of the Out-of Kilter Method and a Comparative Study of Computer Codes," *Mathematical Programming*, Vol.7, No.1(August, 1974), pp.60-86.

G.C. Berresford, A.M. Rockett and J.C. Stevenson, "Khachiyan's Algorithm, Part I: A New Solution to Linear Programming Problems," *BYTE*, Vol.5, No.8(August, 1980), pp.198-208.

___ "Khachiyan's Algorithm, Part II: Problems with the Algortihm," *BYTE*, Vol.5, No.9(September, 1980), pp.242-

G.H. Bradley, G.G. Brown and G.W. Graves, "Design and Implementation of Large Scale Primal Transshipment Algorithm," *Management Science*, Vol.24, No.1 (September, 1977), pp.1-34.

Steven A. Ciarcia, "Everyone Can Know the Real Time," *BYTE*, Vol.7, No.5(May, 1982), pp.34-58.

R.J. Clasen, "The Numerical Solution of Network Problems Using the Out-of-Kilter Algorithm," *RAND Memorandum RM-5456-PR*, Santa Monica, 1968.

Harlan Crowder, Ron S. Dembo and John M. Mulvey, "On Computational Experiments with Mathematical Software," *ACM Transactions on Mathematical Software*, Vol. 5, No.2(June, 1979), pp.193-203.

L.R. Ford and D.R. Fulkerson, "A Primal-Dual Algorithm for the Capacitated Hitchcock Problem," *Naval Research Logistics Quarterly*, Vol.4, No.1(March, 1957), p.47.

D.R. Fulkerson, "An Out-of-Kilter Method for Minimal Cost Flow Problem," *SIAM Journal of Applied Mathematics*, Vol.9, No.1(March, 1961), p.18.

Arthur M. Geoffrion, "Use of Microcomputers in MS/OR," *Interfaces*, Vol.13, No. 1 (February, 1983).

F. Glover, D. Karney and D. Klingman, "Implementation and Computational Comparisons of Primal, Dual and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems," *Networks*, Vol.4, No.3(1974), p.191.

Terry P. Harrison, "Personal Computer and MS/OR," *OR/MS Today*, Vol.10, No.4 (July-August, 1983), p.10.

F.L. Hitchcock, "The Distribution of a Product from Several Sources to Numerous Localitise," *Journal of Mathematics and Physics*, Vol. 20, No.2, p.224.

Thom Hogan, *Osborne CP/M User Guides*, Osborne, 1982.

P. Issacson "Personal Computing Position Paper," *SigPC Notes*, Vol.1, No.2(1978), pp.5-9.

Paul A. Jenson, J. Wesley Barnes, *Network Flow Programming*, John Wiley & Sons, 1980.

D. Klingman, A. Napier, and J. Stutz, "NETGEN—A Program for Generating Large Scale Capacitated Assignments, Transportation, and Minimum Cost Flow Network Problems", *Management Science*, Vol.20, No.5 (January, 1974), pp.814-821.

T.C. Koopmans, "Optimum Utilization of the Transportation System," *Scientific Paper of T.C. Koopmans*, Springer-Verlag, 1970.

Microsoft, *FORTRAN-80 Reference Manual: Microsoft FORTRAN-80 for the Apple II Computer*, 1980.

———, *FORTRAN-80 User's Guide: Microsoft FORTRAN-80 for the Apple II Computer*, 1980.

Doan T. Modianos, Robert C. Scott and Larry W. Cornwell, "Random Number Generation on Microcomputers," *interfaces*, Vol.14, No.4(July-August, 1984), pp.81-87.

J.M. Nilles, "The Personal Computer and Society: A Technological Assessment,"

SigPC Notes, Vol.1, No.3(1978), pp.23-31.

Linus Schrage, "A More Portable Fortran Random Number Generator," *ACM Transactions on Mathematical Software*, Vol.5, No.2(June, 1979), pp.132-138.

R.E.D. Woolsey and H.S. Swanson, *Operations Research for Immediate Application: A Quick and Dirty Manual*, Harper and Row, 1975.

Rodnay Zaks, *The CP/M Handbook with MP/M*, Sybex, 1980.