韓國經營科學會誌
第11券 第1號
1986年 4月

－研究報告－

일반화된 선형/목표계획법의 마이크로컴퓨터용 소프트웨어 개발*

Software Development of Generalized Linear/Goal Programming for Microcomputer

차 동 완 **
고 재 문 ***
이 원 택 **

Abstract

The propose of this study is to present a generalized linear/goal programming software, which has been developed to run on microcomputers with at least 512K bytes of memory. The main characteristics of our algorithm for solving LP/GP problems are outlined as follows: First, it uses the revised simplex algorithm, which is the most efficient computational procedure for computers. Second, it employs the sparse matrix technique to overcome the limited memory of microcomputers. Last, it uses the modified product form of inverse (MPFI) to reduce round-off errors. The test runs with our code written in FORTRAN show that it can be used as an effective tool for solving linear/goal programming problems of considerable size.

I. 서 론

오늘날 조직의 관리자는 상반된 이해나 불완전한 정보 및 제한된 자원이라는 환경 속에서, 여러가지 조직의 형태를 결정하기 위하여 매우 빈번하고 어려운 의사결정에 직면하게 된다. 이러한 때 경영과학(OR) 기법은 실제 상황에 대한 정보분석, 예측, 판단 그리고 의사결정 등에 유용하게 쓰인다.

경영과학 기법 중에서 가장 많이 실제 응용에 쓰이고 있는 것은 시뮬레이션(Simulation), 예측기법(Forecasting), 선형계획법(LP), 목표계획법(GP), 재고이론, 동적계획법, 네트워크기법, 대기이론 등이 있으며, 이밖에도 무수히 많은 경영과학 기법이 다방면에 걸쳐 유용하게 쓰이고 있다.

선형계획법은 실생활에 많이 적용되고 있는 대표적인 경영과학기법으로서 제한된 자원을 효율적으으로 배분하여 목적을 최적화하고자 할 때 이용되는 기법이다. 그 이용도에 따라 대형 컴퓨터용으로 많은 Package가 개발되어 왔으며, 최근에 점차 널리 활용되어 가고 있는 마이크로컴퓨터에서 수행되는 LP Package도 많이 보급되었다[5]. 가장 널리 알려진 마이크로컴퓨터용 LP Package에는 LP88, LP83, Super LINDO/PC, BEST-PLAN, LPMaster, LPP, MPP 등이 있으며 대부분이 IBM－PC/XT(256K)에서 수행된다[10].

그러나 현실의 많은 문제들이 공동의 제약조건하에서 서로 다른 다수의 목적이 상충되고 있음에 반해, 선형계획법은 그러한 복수의 목적을 다룰 수 없

* 이 논문은 아산사회복지사업재단의 1984년도 연구비 지원에 의하여 연구되었음.
** 한국과학기술원 경영과학과
*** 울산대학교 산업공학과

다는 단점이 있다. 이러한 현실을 충분히 반영하기 위해 개발된 것이 목표계획법(Goal Programming)으로 이는 다수의 목적을 동시에 고려하면서 최적의 해를 구하는 경영과학기법이다 [2]. 이 기법은 현실반영의 강점으로 인해 최근에 수많은 현실적인 문제를 해결하는데 자주 이용되고 있다[3,6,7]. 그러나, 현재까지 개발된 대부분의 GP Package가 대형 컴퓨터용이기 때문에 마이크로컴퓨터로서는 다룰 수 없는 실정이어서 그 기법의 확산에 장애가 되고 있다. 따라서, 마이크로컴퓨터를 이용하여 LP와 GP를 동시에 해결할 수 있는 Package가 절실히 요구되고 있다.

이러한 요구는 두 계획법 모두가 의사결정변수에 대하여 목적함수와 제약식이 선형인 공통적 특성을 갖고 있으므로 쉽게 해결될 수 있다. 본 연구는 일반화된 선형/목표계획법의 마이크로컴퓨터용 소프트웨어를 개발하는데 그 목적이 있다.

LP와 GP 문제를 푸는 기법으로는 Revised Simplex Algorithm을 사용하였으며, 그 프로그래밍에 있어서 컴퓨터 언어 중 일반적으로 잘 알려진 FORTRAN을 사용하였다. 본 연구에서는 개발된 소프트웨어 (LGPM : Linear/Goal Programming for Microcomputer)에서는, 마이크로컴퓨터의 적은 기억용량을 극복하고 최적해의 정확성을 보다 높이기 위하여, 기존의 기법들을 조금 개선한 보다 효율적인 기법들 - Sparse Matrix Technique, Modified Product Form of Inverse (MPFI), New Reinversion Scheme, FCFS Pivot Rule 등 - 을 이용하였다.

최근 PC의 기억용량이 512K 또는 640K까지 확장됨에 부응하여, LGPM은 최소기억용량 512K인 IBM-PC/XT 또는 이와 호환성이 있는 컴퓨터용으로 개발하였다. 또한 기존의 Package와 비교하기 위하여 256K용으로도 개발하여 그 성능을 비교한 결과, LGPM이 최적해의 정확성이 뛰어남을 입증했으며, 그 밖의 계산속도나 취급할 수 있는 문제의 크기 등에서는 거의 비슷하였으나, LGPM은 사후분석 (Post-Optimality Analysis)이나 특별한 제약식 - 예를 들어, 변수의 상한 또는 하한, 그리고 제약식의 범위로 표시된 제약식 - 의 처리에 있어서 뒤떨어진다. 그러나, 이러한 단점은 앞으로 LGPM 을 좀더 개선함으로서 충분히 제거될 수 있을 것이다.

Ⅱ. 기본 설계

2.1. 취급할 문제

본 연구에서는 개발된 소프트웨어는 LGPM (Linear/Goal Programming for Microcomputer)이며, 이것은 선형계획법 (LP) 문제뿐만 아니라 목표계획법 (GP) 문제도 효율적으로 해결한다. 해법으로 사용된 Revised Simplex Algorithm은 마이크로 컴퓨터를 이용하여 LP 또는 GP 문제를 해결할 때 가장 효율적인 해법이다[8]. 먼저 LP 문제와 GP 문제를 각각 행렬형태로 나타낸 후에, LP와 GP를 모두 포함하는 문제를 형렬형태로 표시함으로써, 이러한 Revised Simplex Algorithm을 LP/GP 문제에 적용하게 된다.

일반적인 LP 문제를 행렬형태로 나타내면 다음의 (1)과 같다.

$$\begin{aligned} &\text{minimize} && cx \\ &s.t. && Ax = b \qquad x \geq 0 \end{aligned} \tag{1}$$

여기서, c는 (1×n) 행벡터, x와 0는 (n×1) 열벡터, b는 (m×1)열벡터, 그리고 A는 (m×n)행렬이다.

또한 여러개의 목표가 주어져 있고 이들 목표에 대한 우선순위와 중요도가 정해져 있는 가장 일반화된 목표계획법 문제는 다음의 (2)와 같다[2].

$$\begin{aligned} &\text{minimize} && \| P_1(n,p), \cdots, P_q(n,p) \| \\ &\text{s.t.} && Ax + n - p = g \\ & && x, n, p \geq 0 \end{aligned} \tag{2}$$

여기서, g는 m개의 목표를 나타내는 (m×1) 열벡터이며, $P_k(n,p)$는 이들 목표를 달성시키기 위하여 우선순위와 중요도를 고려한 k번째 목표성취함수이다. 그리고, n과 p는 목표달성에 있어서 미달 또는 초과를 나타내는 편차변수이다.

이러한 LP 또는 GP 문제를 모두 포함할 수 있는 가장 일반화된 LP/GP 문제를 형렬형태로 나타내면 다음의 (3)과 같다[3,4].

$$\begin{aligned} &\text{lexico-graphically minimize} \\ &\quad a = (C_1 x, \cdots, C_q x) \\ &s.t. \quad Ax = b \\ &\qquad x \geq 0 \end{aligned} \tag{3}$$

여기서, a는 (1×q) 목표성취 벡터, C_k는 k번째 우선순위에 대한 (1×n) 행벡터, x와 0는 (n×1) 열벡터 b는 (1×m) 열벡터, 그리고 A는 (m×n) 행렬이다. 또한

x는 의사결정 변수뿐만 아니라 편차변수((2)에서의 n과 p)도 포함한다. 따라서, $q=1$일 때, LP문제를 나타낼 수 있게 된다.

2.2. Revised Simpex Algorithm에 의한 해법

(3)과 같은 LP/GP 문제를 행렬체계로 나타내면 다음의 (4)와 같은 형태로 된다.

$$\left|\begin{array}{cc} -C_1 & 0 \\ \cdot & \cdot \\ -C_q & 0 \\ \hline A & b \end{array}\right| \qquad (4)$$

여기서, k번째 단계에서의 Basic Variable과 Nonbasic Variable에 대응하는 A행렬의 열벡터를 모은 행렬을 각각 B와 N이라 하고, 마찬가지로 C도 각각에 대응하는 것을 C_{B_t}와 C_{N_t}라 하면, (4)는 다음의 (5)로 변형이 된다. 즉, $A=[B\ N]$ 그리고 $C=[C_{B_t}\ C_{N_t}]$를 (4)에 대입하면 (5)를 쉽게 얻을 수 있다.

$$\left|\begin{array}{ccc} -C_{B1} & -C_{N1} & 0 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ -C_{Bq} & -C_{Nq} & 0 \\ \hline B & N & b \end{array}\right| \qquad (5)$$

그리고, k 단계의 끝에서는, (5)가 다음의 (6)으로 바뀌게 된다.

$$\left|\begin{array}{ccc} 0 & \overline{C}_1 & a_1^* \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ 0 & \overline{C}_q & a_q^* \\ \hline I & B^{-1}N & B^{-1}b \end{array}\right| \qquad (6)$$

여기서, a_i^*는 그때까지 달성된 목표이며 다음의 (7)로 표시된다.

$$a_i^* = C_{B_i} B^{-1} b,\ i=1,2,\cdots,q \qquad (7)$$

그리고, $\overline{C}_i$는 C_{N_i}가 변화되어진 계수들이며 다음의 (8)로 표시된다.

$$\overline{C}_i = C_{B_i} B^{-1} N - C_{N_i},\ i=1,\ 2,\cdots,q \qquad (8)$$

이러한 (6)의 행렬체계가 k번째 단계의 끝에 도출되었을 때, k번째 단계의 최적조건, 변수제거 그리고 진입변수의 결정 등은 다음과 같다.

(최적조건)

$\overline{C}=[\overline{C}_{ij}]$를 Nonbasic Variable에 대응하는 $\overline{C}$들을 행벡터로 하여 만든 하는 $(q\times(n-m))$ 행렬이고, $\overline{C}_{\cdot j}$를 $\overline{C}$의 j번째 열벡터라 하자. 모든 j에 대하여 $\overline{C}_j$가 lexico-nonnegativity를 만족하면, k번째 단계에서 최적이다.

(변수제거)

$k+1$번째 단계에서 k번째 단계까지의 최적조건을 만족하기 위해 Nonbasic Variable중 $\overline{C}_{kj}>0$ 인 것을 제거하여 다음 단계에서 Basic Variable로 바뀌지 못하게 하여야 한다. 그 이유는, 만약 k번째까지 최적조건을 만족하고 있던 상태에서 $\overline{C}_{kj}>0$ 이었던 Nonbasic Variable을 $k+1$번째 단계에서 Basic Variable로 결정하게 되면, $\overline{C}_{\cdot j}$들 중 어떤 j에 대하여 lexico-nonnegativity를 깨뜨리게 되어 최적조건을 위반하게 되기 때문이다.

(진입변수 결정)

위에서 변수제거를 수행하고 나서, 나머지 Nonbasic Variable 중에서 $\overline{C}_{k+1,j}<0$인 변수를 $k+1$ 단계의 진입변수로 결정한다.

Ⅲ. LGPM에 사용된 기법들

3.1. MPFI (Modified Product Form of Inverse)

Revised Simplex Algorithm에서 모든 단계의 계산을 수행할 때, $B^{-1}y$ 또는 yB^{-1}의 두가지 형태로 나타나기 때문에, B^{-1}를 어떻게 기억시키는가 하는 것이 꼭 필요하다. 컴퓨터에서 B^{-1}를 기억시키는 가장 대표적인 방법은 PFI(Product Form of Inverse) 기법이다[9, 11]. 즉,

$$B^{-1} = E_m E_{m-1} \cdots E_1 \qquad (8)$$

$E_t\ (t=1,\ \cdots,\ m)$들은 하나의 열을 제외한 모든 열이 단위 벡터로 구성된, 다음의 (9)와 같은 정방행렬이다.

$$E_t = \left|\begin{array}{ccccc} 1 & & -b_{1s}/p_t & & \\ & \ddots & \vdots & & \\ & & 1/p_t & & \\ & & \vdots & \ddots & \\ & & -b_{ms}/p_t & & 1 \end{array}\right| \qquad (9)$$

여기서, $B\cdot s=(b_{1s},\ \cdots,\ b_{rs},\ \cdots,\ b_{ms})$는 t번째 Basic Variable에 대응하는 A행렬의 열벡터 $A\cdot s$

에 E_1, E_2, …, E_{t-1}를 차례로 곱하여 얻은 것이며, t번째 pivot 원소인 p_t는 $B\cdot s$의 r번째 원소이다. B^{-1}를 PFI 기법으로 표시함으로써, $B^{-1}y$ 또는 yB^{-1}의 계산은 오른쪽 또는 왼쪽부터 차례로 쉽게 계산되어 진다.

그러나, 컴퓨터가 임의의 실수(real number)를 memory에 기억시킬때 그 수를 정확하게 표현할 수는 없다. 즉, single precision (4 bytes)으로 실수를 기억시킬 때 최대 유효숫자는 십진법으로 6자리이다. 따라서 7자리 이상의 유효숫자를 갖는 임의의 실수는 오차를 가진채로 컴퓨터에 기억되며, 이러한 컴퓨터의 기억 오차를 일반적으로 "round - off error"라 부른다. 그러므로, 컴퓨터로 모든 계산을 수행할 때 발생하는 수들의 정확성을 최대한으로 보장할수록 보다 계산 결과에 대한 정확한 값을 얻을 수 있게 된다.

PFI 기법에서 E_t들을 만드는데 p_t로 나누기를 함으로써, 일반적으로 그 나누어진 수는 원래의 수보다 소숫점 이하로 많은 자릿수를 갖게 됨에 따라 컴퓨터가 기억하는 과정에서 "round-off error" 가 발생하므로, 이것을 제거시키기 위하여 pivot행 에 대하여 p_t로 나누는 것을 일단 보류하고, 그 다음 단계에서 나누기 계산을 수행하면 약간의 "round-off error"를 감소시키게 된다. 그 이유는, 한단계 늦추어서 E_t를 곱하고나서 p_t로 나누어진 계산의 결과에서 발생하는 수는, 원래의 수가 이미 E_t를 곱한 후이므로 그 수는 상당히 작은 "round-off error" 를 갖게 되기 때문이다.

그러므로, "round-off error"를 감소시키려는 방법으로 기존의 PFI 기법을 약간 수정하여 만든 것이 MPFI 기법이다. MPF 는 PFI에서의 E_t를 하나의 대각 행렬 D_t의 역행렬과 E_t와 유사한 행렬인 F_t로 분해하여 (즉, $E_t = D_t^{-1} F_t$) B^{-1}를 다음의 (10)과 같이 나타낸다.

$$B^{-1} = D_m^{-1}\ D_{m-1}^{-1} F_m D_{m-2}^{-1} F_{m-1} \cdots D_2^{-1} F_3 D_1^{-1} F_2 F_1 \tag{10}$$

여기서 D_t는 t번째 행의 대각 원소가 pivot 원소로 된 대각 행렬이며, 즉, 단위행렬(Identity Matrix)에서 t번째 대각원소를 p_t로 대체한 행렬이다. 그리고, F_t은 (9)와 비슷하며 t번째 행의 대각원소가 1인 점이 다른 다음의 (11)과 같은 행렬이다.

이 방법은 PFI에서의 나누기 계산을 한단계 후에 처리하는 것과 같으며 "round-off error"를 감소시키게 된다. 이 MPFI로 B^{-1}를 나타낼 때 기억

$$F_t = \begin{vmatrix} 1 & & & -b_{1s}/p_t & & \\ & \ddots & & \vdots & & \\ & & 1 & & & \\ & & & \ddots & & \\ & & & & \ddots & \\ & & & -b_{ms}/p_t & & 1 \end{vmatrix} \tag{11}$$

용량은 D_t를 기억시키는 양만큼이 더 필요한 것처럼 보이나, 실제로 D_t를 기억시키지 않으며 F_t의 t번째 대각원소를 그대로 p_t로 기억시킴으로써 D_t를 기억시킨 것과 같아지며, F_t의 계산에서는 제외시키게 된다.

3.2. 새로운 Reinversion 방법

일정단계가 지나면 B^{-1}를 다시 계산하는 Reinversion을 수행하게 된다. 그 이유는 그동안 쌓였던 오차를 없애며 B^{-1}에서 불필요한 항을 제거시킴으로써 계산속도를 빠르게 할 수 있기 때문이다. 이러한 Reinversion은 F_t들을 가능한 한 sparse 하게 하기 위하여, Basis Matrix에서 가장 sparse한 column부터 pivoting을 수행하여야 한다. 즉, F_t들이 갖는 비영의 원소(nonzero element)들의 수를 줄임으로써, 계산속도를 빠르게 할 수 있을뿐만 아니라 "round-off error"도 감소시키게 된다.

MPFI 방식으로 B^{-1}를 새로이 기억시키기 위하여는, 먼저 Basic Variable들에 대응하는 A행렬에서의 열벡터를 모아 각 열벡터들을 가장 sparse한 순서로 나열하여 Basis Matrix B를 만들어야 한다. 그런 후에 (11)과 같은 F_t를 만든다. 이때, t번째 F_t를 만드는데 이용된 $B\cdot s$는 B의 s번째 열벡터 $A\cdot s$에 F_1, …, F_{t-1}와 D_1^{-1}, …, D_{t-2}^{-1}를 다음의 (12)와 같이 차례로 곱하여 얻게 된다.

$$\begin{aligned} B\cdot s = {} & D_{t-2}^{-1} F_{t-1} D_{t-3}^{-1} F_{t-2} \\ & \cdots D_2^{-1} F_3 D_1^{-1} F_2 F_1 A\cdot s \end{aligned} \tag{12}$$

그리고, t번째 pivot행을 결정하는데 있어서는 $B\cdot s$의 원소 중 그 절대값이 가장 큰것으로 결정한다.

3.3. Sparse Matrix Technique의 이용

일반적으로, LP 또는 GP 문제에서의 입력자료인 A행렬과 C들은 매우 sparse하며, 이러한 입력자료를 마이크로컴퓨터가 갖고 있는 적은 기억용량을 효과적으로 이용하기 위하여 Sparse Matrix Te-

chnique을 이용하여야 한다. 이 방법은 sparse 행렬의 형태와 그 용도에 따라 여러가지로 분류되며 [9, 11], 이러한 여러가지 방법들을 대표하는 Sparse Matrix Technique은 Sparse Row-wise Representation과 Sparse Column-wise Representation이다. 이 두 방법의 특징은 기억 용량을 가장 적게 쓰며, 동시에 matrix multiplication, addtion, permutation, transpose 등 중요한 행렬 연산에 있어서 매우 간편하게 처리된다는 점이다.

본 연구에서 채택한 방법은 Sparse Column-wise Representation이며, 이것은 Revised Simplex Algorithm에서와 MPFI에 있어서, 모든 계산이 (13)과 같이 하나의 열벡터와 행렬의 곱하기로 이루어지기 때문이다. 이 방법으로 입력자료에서 0이 아닌 수치만 각 열 별로 연결시켜 기억하게 되며, 프로그램 수행 중 필요한 자료를 효과적으로 이용할 수 있게 된다.

또한 프로그램의 작성에 있어서, 이러한 입력자료 및 F_t들이 갖는 nonzero elements를 기억하기 위한 memory allocation은 기억용량의 효과적으로 이용하는데 있어서 필수적인 과정이다. 즉, 0이 아닌 입력자료가 프로그램 내에서 발생시키는 F_t들이 갖게되는 nonzero element를 기억시키는 데에도 Sparse Column-wise Representation을 사용하며, 입력자료와 F_t들을 기억시키는데 필요한 기억용량의 할당에 따라 취급할 수 있는 LP 또는 GP 문제의 크기가 결정된다. 본 연구에서 개발된 LGPM은 256K용과 512K용에서 각각 〈표 1〉과 같은 크기의 문제를 해결할 수 있다.

〈표 1〉 LGPM이 해결할 수 있는 LP 또는 GP 문제의 크기

문제의 크기	256K	512K
최대 변수의 수	900개	1,400개
최대 제약식의 수	400개	700개
최대 0이 아닌 입력자료의 수	8,000개	10,000개
MPFI에서 발생하는 0이 아닌 수치의 수	20,000개	20,000개

3.4. FCFS Pivot Rule

GP 문제는 편차변수들의 합을 가장 최소로 할때, 즉 가능한 한 이들 변수를 nonbasic 변수로하여 0의 값을 갖도록 유도하면 최적해를 빠르게 구할 수 있다. 이러한 GP 문제의 특성은, 일반화된 목표계획법 문제 (2)의 목적함수에서 쉽게 알 수 있다. 즉, 다음의 (13)를 극소화시키기 위하여 n과 p를 될 수 있는 한 0의 값을 갖도록 유도하면, 목적함수의 값이 최소로 된다.

$$a = | p_1(n, p), \cdots, p_q(n, p) | \qquad (13)$$

이러한 특성을 살리기 위하여, Entering 변수를 결정함에 있어서 주요의사결정 변수를 먼저 고려하는 FCFS Pivot Rule을 적용시키는 방법을 사용하였다. 그러나, 자칫 소위 "cycling" 현상이 일어날 가능성이 존재하므로, 그러한 현상을 예방하기 위한 방법도 함께 사용하였다[1]. 또한 LP의 경우에는 종전의 Maximum Pivot Rule을 그대로 이용하였다.

Ⅳ. LGPM과 기존의 Package와의 비교

LP 또는 GP Package들을 비교하는데 있어서 가장 촛점이 되는 것은 그것이 해결할 수 있는 문제의 크기와 해법의 계산속도이다. 그밖에 비교의 기준으로는 가격, 최소기억용량, 정확성의 정도, 입력자료 작성의 간편성, 최적해 분석의 용이성, 사후분석, 그리고 사용방법에 대한 소개 등이 있다[10]. 〈표 2〉는 이러한 비교항목에 따라 기존의 Package들을 분석해 놓은 것에 본 연구에서 개발한 LGPM을 추가로 붙여놓은 것이다.

기존의 Package와 비교하기 위하여 LGPM을 256K용으로 개발하여 비교한 결과, 사후분석(Post-Optimality Analysis)이나 특별한 제약식—예를 들면, 변수의 상한 또는 하한, 그리고 제약식의 범위로 표시된 제약식—의 처리에서 뒤떨어지지만, 이러한 단점은 앞으로 LGPM을 좀더 개선함으로써 충분히 보완될 수 있을 것이다. LGPM이 취급할 수 있는 문제의 크기는 다른 Package 보다 더 크며, 다른 비교항목에서는 거의 비슷하다. 특히 정확성의 정도에 있어서는 같은 single precision 이지만 LGPM이 더 우수하다.

〈표 2〉 기존의 Package와 LGPM과의 비교(자료 : R. Sharda, 1984)

Package	LP88	LP83	Super LINDO / PC	BEST-PLAN	LPMaster	Ours (LGPM)
Price	$99	$795	$795	$1000	$795	-
Maximum Size	255×2555	340×513	120×300	200×500	500×1000	400×900
Precision	double	double	single	single	single	single
Spreadsheet Compatibility	no	yes	yes	yes	no	yes
Special Constraints	yes	no	no	yes	no	no
Sensitivity Analysis	yes	no	yes	yes	no	no
Tutorial	no	yes	yes	no	yes	yes

V. LGPM의 구성

프로그램은 하나의 주프로그램과 16개의 부프로그램으로 구성되어 있다. 주프로그램은 여러가지 프로그램 상수의 초기치를 주고 입력자료를 읽어들이는 부프로그램을 부르며, 초기가능해를 찾는 부프로그램과 최적해를 발견하는 부프로그램을 차례로 수행시킨다. 각 부프로그램의 역할은 다음과 같다.

1. INIT 프로그램 상수에 대한 초기치 부여
2. RED 자료 입력
3. STS 입력자료의 착오 존재여부를 확인하여 출력
4. BFS1 Slack 변수 또는 Surplus 변수 도입
5. BFS2 Basic Feasible Solution을 찾음
6. DUAL Dual Simplex 수행
7. BOS 주 제어 프로그램
8. XCK 현재의 해를 가산
9. VER Reinversion 수행
10. GET Dual Price 계산
11. DEL Reduced Cost 계산
12. FCFS Entering 변수 선택
13. JMY Entering Column Generation
14. ROW Leaving 변수 선택
15. PIV Pivoting 수행
16. WRT 최적해, 잠재가 등 각종 정보의 출력

VI. 사용예

여기서는 실제 LP/GP 문제를 LGPM으로 해결한 결과를 보도록 하자. LP/GP문제, 입력자료, 그리고 결과 출력은 다음과 같다.

6.1. LP/GP 문제

Lexico-graphically minimize

$a = (N1,\ N2)$

$$
\begin{aligned}
s.t.\ & 7X1+3X2+7X3+N1-P1=17 : \text{CONST 1}\\
& 4X1+2X2+2X3+N2-P2=8 : \text{CONST 2}\\
& 6X1+1X2+4X3 \le 7 : \text{CONST 3}\\
& 4X1+2X2+2X3 \le 6 : \text{CONST 4}\\
& 1X1 \le 1 : \text{CONST 5}\\
& 1X2 \le 1 : \text{CONST 6}\\
& 1X3 \le 1 : \text{CONST 7}\\
& X1, X2, X3, N1, N2, P1, P2 \ge 0
\end{aligned}
$$

	X1	X2	X3	N1	N2	P1	P2	RHS
Obj. 1				1				
Obj. 2					1			
CONST 1	7	3	7	1		-1		= 17
CONST 2	4	2	2		1		-1	= 8
CONST 3	6	1	4					≤ 7
CONST 4	4	2	2					≤ 6
CONST 5	1							≤ 1
CONST 6		1						≤ 1
CONST 7			1					≤ 1

6.2. 입력자료

```
BEGN    PROJECT
SIZE        2       7   7
OBJ.    MIN
```

```
*       Obj. 1
*       Obj. 2
END
RHS     R. H. S.
*       CONST 1        0 17.0000
*       CONST 2        0  8.0000
*       CONST 3       -1  7.0000
*       CONST 4       -1  6.0000
*       CONST 5       -1  1.0000
*       CONST 6       -1  1.0000
*       CONST 7       -1  1.0000
END
MATRIX
*       1       2  X1      CONST 1   7.0000
*       1       3  X1      CONST 2   4.0000
*       1       4  X1      CONST 3   6.0000
*       1       5  X1      CONST 4   4.0000
*       1       6  X1      CONST 5   1.0000
*       2       2  X2      CONST 1   3.0000
*       2       3  X2      CONST 2   2.0000
*       2       4  X2      CONST 3   1.0000
*       2       5  X2      CONST 4   2.0000
*       2       7  X2      CONST 6   1.0000
*       3       2  X3      CONST 1   7.0000
*       3       3  X3      CONST 2   2.0000
*       3       4  X3      CONST 3   4.0000
*       3       5  X3      CONST 4   2.0000
*       3       8  X3      CONST 7   1.0000
*       4       0  N1      Obj. 1    1.0000
*       4       2  N1      CONST 1   1.0000
*       5       1  N2      Obj. 2    1.0000
*       5       3  N2      CONST 2   1.0000
*       6       2  P1      CONST 1  -1.0000
*       7       3  P2      CONST 2  -1.0000
END
SOLV
ENDATA
```

6.3. LGPM 수행

```
A〉 LGPM
Linear/Goal Progamming for Microcomputers
Version 1.00 (C) Copyright 1985, 1986
Dept. of Management Science in KAIST
Input Data File [.LPD] : PROJECT. LPD
Output File   [.OUT] : PROJECT. OUT
```

6.4. 결과 출력

```
LGPM EXECUTION
*** OPTIMAL SOLUTION ***
ITERATIONS     =  5
PIVOTINGS      =  5
REINVERSIONS   =  0
OBJECTIVE 1 : Obj. 1   =  4.66667
OBJECTIVE 2 : Obj. 2   =  2.66667
```

CONSTRAINTS

NO.ROW.NAME	ACTIVITY	R. H. S.	DUAL PRICE
1 Obj. 1	4.66667	.00000	.00000
2 Obj. 2	2.66667	.00000	1.00000
3 CONST 1	17.00000	17.00000	.00000
4 CONST 2	8.00000	8.00000	-1.00000
5 CONST 3	7.00000	7.00000	.66667
6 CONST 4	5.33333	6.00000	.00000
7 CONST 5	.33333	1.00000	.00000
8 CONST 6	1.00000	1.00000	1.33333
9 CONST 7	1.00000	1.00000	-.66667

COLUMNS

OBJECTIVE 1 : Obj. 1 = 4.66667

NO. COL. NAME	VALUE	TRUE COST	REDUCED COST
1 X1	.33333	.00000	.00000
2 X2	1.00000	.00000	.00000
3 X3	1.00000	.00000	.00000
4 N1	4.66667	1.00000	.00000
5 N2	2.66667	.00000	.00000
6 P1	.00000	.00000	1.00000
7 P2	.00000	.00000	.00000

OBJECTIVE 2 : Obj. 2 = 2.66667

NO. COL. NAME	VALUE	TRUE COST	REDUCED COST
1 X1	.33333	.00000	.00000
2 X2	1.00000	.00000	.00000
3 X3	1.00000	.00000	.00000
4 N1	4.66667	.00000	.00000
5 N2	2.66667	1.00000	.00000
6 P1	.00000	.00000	.00000
7 P2	.00000	.00000	1.00000

Ⅶ. 결 론

본 연구에서 개발된 LGPM (Linear/Goal Programming for Microcomputer)은 FORTRAN 언어로 프로그래밍하였으며, IBM PC/XT 또는 이와 호환성이 있는 컴퓨터 (최소기억용량 512K)에서 수행된다. LGPM은 LP를 objective가 하나인 경우의 GP로 간주함으로서 처리할 수 있으며, 해결할 수 있는 LP/GP 문제의 크기를 보면, 최대 변수의 수 1,400개, 최대 제약식의 수 700개, 최대 0이 아닌 입력자료의 수 10,000개, 그리고 MPFI에서 발생하는 최대 0이 아닌 수치의 수 20,000개 등이다.

LGPM code 개발에는 본 연구의 새로운 결과인 MPFI, Reinversion 방법, Sparse Matrix Technique, FCFS Pivot Rule 등의 새로운 효율적인 기법들을 사용하였다.

LGPM은 대형 컴퓨터를 위한 GP 또는 LP Packge에 비하여 그 효능이 크게 뒤떨어지지 않으며, 마이크로컴퓨터용이기 때문에 사용자가 편리하게 이용할 수 있을뿐만 아니라 계산결과를 간단명료하게 보여준다.

이러한 마이크로컴퓨터용 소프트웨어가 개발되어 일반에게 쉽게 활용될 수 있으므로, 앞으로 OR/경영과학 기법을 적용한 마이크로컴퓨터용 소프트웨어에 대한 연구 및 개발이 활발해질 것으로 사료된다. 또한 기존의 기법보다 효율적이고 새로운 해법을 개발함으로써 학문적인 기여를 하였다.

주 요 참 고 문 헌

1. R. G. Bland, New Finite Pivoting Rules for the Simplex Algorithm, *Math. of Oper. Res.*, 2, pp. 103 – 107, 1977.
2. J. P. Ignizio, *Goal Programming and Extensions*, Heath and Company, Toronto, 1976.
3. J. P. Ignizio, Generalized Goal Programming, An Overview, *Comput. & Ops. Res.*, Vol. 10, No. 4, pp. 277 – 289, 1983.
4. J. P. Ignizio, A Note on Computational Methods in Lexicographic Linear Goal Programming, *J. Opl. Res. Soc.*, Vol. 34, pp. 539 – 542, 1983.
5. L. P. Jennergren, OR and Micros, *Europ. J. Opl. Res.*, 20, pp. 1 – 9, 1985.
6. J. S. H. Kornbluth, A Survey of Goal Programming, *OMEGA*, 1, pp. 193 – 205, 1973.
7. S. M. Lee, M, Gen and J. P. Shim, Goal Programming for Decision Making : A State-of-the-Art Survey, Working Paper, Dept. of Mgt., Univ. of Nebraska-Lincoln, 1982.
8. D. L. Olson, Comparison of Four Goal Programming Algorithms, *J. Opl. Res. Soc.*, Vol. 35, pp. 347 – 354, 1984.
9. S. Pissanetzky, *Sparse Matrix Technology*, Academic Press Inc., New York, 1984.
10. R. Sharda, Linear Programming on Microcomputers : A Survey, Working Paper 84 – 15 College of Business Administration, Oklahoma State Univ., 1984.
11. R. P. Tewarson, *Sparse Matrices*, Academic Press Inc., New York, 1979.