단체법 프로그램 LPAKO 개발에 관한 연구*

박순달** · 김우제*** · 박찬규** · 임성묵**

Development of LPAKO : Software of Simplex Method for Liner Programming*

Soondal Park** · Yu Je Kim*** · Chan Kyu Park · Seong Mook Lim**

ABSTRACT

The purpose of this paper is to develope a large-scale simplex method program LPAKO. Various up-to-date techniques are argued and implemented.

In LPAKO, basis matrices are stored in a LU factorized form, and Reid's method is used to update LU maintaining high sparsity and numerical stability, and further Markowitz's ordering is used in factorizing a basis matrix into a sparse LU form.

As the data structures of basis matrix, Gustavson's data structure and row-column linked list structure are considered. The various criteria for reinversion are also discussed.

The dynamic steepest-edge simplex algorithm is used for selection of an entering variable, and a new variation of the MINOS' perturbation technique is suggested for the resolution of degeneracy. Many preprocessing and scaling techniques are implemented. In addition, a new, effective initial basis construction method are suggested, and the criteria for optimality and infeasibility are suggested respectively.

Finally, LPAKO is compared with MINOS by test results.

1. 서 론

다음과 같은 상하한 제약이 있는 선형계획문제가 있다고 하자.

$$\begin{aligned} \text{Max } & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t } & \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b} \\ & \boldsymbol{l}\le\boldsymbol{x}\le\boldsymbol{u} \end{aligned}$$

선형계획문제의 해법 중에 하나인 단체법의

* 본 연구는 한국과학재단의 목적기초연구과제(과제번호 95-0200-39-01-2)에 의해 지원되었음.
** 서울대학교 산업공학과
*** 대진대학교 산업공학과

구현에 관한 연구는 단체법이 개발된 이래로 많은 연구가 수행되어 단체법 프로그램은 수행속도나 다룰 수 있는 문제의 크기, 수치적 안정성과 해의 정확도 면에서 크게 개선되었다. 그러나 대부분의 단체법 프로그램은 상용화되어 개발에 사용된 기법들은 잘 알려져 있지 않다.

본 연구는 단체법의 구현에 관련된 기존의 연구 결과를 구현할 때 발생되는 여러 가지 문제점과 기존의 연구에서 다루지 않았던 구현시의 문제점들을 고찰하고 이러한 문제점들을 단체법 프로그램 LPAKO에서 해결하는 방법을 제시한다. LPAKO는 공개된 프로그램으로 단체법의 연구나 단체법을 사용하는 응용 최적화 프로그램에 많이 활용될 수 있을 것으로 기대된다.

단체법의 효율적인 구현에 있어서 가장 중요한 문제 중의 하나가 기저역행렬의 계산과 보관·수정 방법이다([1][2]). 기저역행렬의 경우 초기에는 명시형 또는 적산형으로 B^{-1} 전체를 보관하였으나 현재는 많은 단체법 프로그램들에서 수치안정성과 행렬의 희소성을 고려하여 상하분해형으로 기저역행렬을 보관한다([1][11]).

상하분해형으로 기저역행렬을 사용하기 위해서는 크게 기저행렬을 상하삼각행렬로 분해하는 상하분해(LU factorization)방법과 기저에서 한 열이 바뀌었을 때 기저역행렬을 수정하는 기저수정(Updating)방법이 필요하게 된다. 이 과정에서 순서화와 선회연산 전략에 따라 기저역행렬의 수치오차 및 희소도가 달라지게 되어, 계산 시간과 결과의 정확성에도 큰 영향을 미치게 된다. 따라서, 상하분해와 기저수정 방법에 대한 많은 연구가 있어 왔으며, 이들 중 상하분해시에 Markowitz 순서화, 수정방법에 Bartels-Golub 방법과 Reid 방법이 효율적으로 알려져 널리 이용되고 이들의 효율적인 구현에 대해서도 많은 연구가 이루어졌다([9][12][18] [20][21]).

한편, 프로그램이 안정적이고 빠른 시간내에 해를 구하기 위해서는 수치적 안정성이나 기저역행렬의 희소도 등에 대한 고려가 필요하다. 만약 수치적 안정성이나 기저역행렬의 희소도가 나빠지면 재역산(Reinversion)을 수행하는데 효율적인 재역산의 시점 선정은 중요한 문제가 된다([3][6][20]).

대형선형계획문제에서는 진입변수 선정 방법에 따라 반복회수에 큰 차이가 발생하며 퇴화가 심한 문제에서는 단체법의 수행시간이 증가하게 되므로 퇴화 방지나 해의 순환을 방지하는 기법들이 필요하다([2][5][14][16]).

단체법을 적용하기 이전에 주어진 문제를 분석하여 변수나 제약식을 제거하는 사전처리나 규모화(Scaling) 기법도 단체법 프로그램의 수행속도에 많은 영향을 미치므로 이를 효과적으로 구현해야 한다([8]). 또한 수치적 안정성과 높은 희소도를 갖는 초기기저를 선정하는 것은 단체법의 전체 반복회수와 수행시간을 줄이는 효과를 가져온다([7][10]).

본 연구에서는 LPAKO에 구현된 기저의 상하분해, 재역산 기준, 평가전략과 퇴화 방지, 사전처리와 규모화, 초기기저 구성 방법 등을 설명하고 구현된 방법들과 자료구조가 효율적임을 실험을 통해 제시하고자 한다. 끝으로 LPAKO에서 사용된 최적판정 방법과 비가능 판정 방법도 함께 제시한다.

2. 기저의 상하분해

상하분해 기저역행렬

상하분해 방법은 다음과 같이 기저행렬 B를 상하삼각행렬로 분해하여 사용한다.

$B=LU$, L:상삼각행렬, U: 하삼각행렬

상하분해 방법은 명시형이나 적산형에 비해 행렬의 희소성을 잘 이용할 수 있어 계산량을 줄일 수 있고 수치 안정성을 이루기 쉽다는 장점을 가진다([1]). 상하분해형으로 기저역행렬을 사용하기 위해서는 크게 기저행렬을 상하삼각행렬로 분해하는 상하분해 방법과 기저에서 한 열이 바뀌었을 때 기저역행렬을 수정하는 기저수정 방법이 필요하게 된다.

상하분해 형태로 보관된 기저의 수정 방법에 관한 연구는 Bartels, Golub, Forrest, Tomlin, Reid, Saunders 등에 의해 이루어 졌다([9][18] [20][21]). Bartels-Golub 방법은 수치적 안정성이 우수하여 이의 변형들이 실용적으로 많이 사용되고 있다([1][9][20][21]). 그러나 Bartels- Golub방법은 U에 새로 추가되는 비영요소가 많아 희소성을 충분히 이용하지 못한다는 단점이 있는데 이를 보완한 방법이 Reid에 의해 제안되었다([20]). Reid의 기저수정 방법은 행·열 치환을 통하여 상삼각행렬에 발생하는 대각 아래의 비영인 열(Spike)의 크기를 줄인 후 선회연산을 행한다. 이렇게 함으로써 희소도를 유지시키고, 선회연산의 횟수가 줄어들게 되므로 수치오차의 누적을 막을 수 있는 장점이 있다([20]). Forrest-Tomlin 방법은 U에 새로 추가되는 비영요소가 매우 작다는 장점이 있으나 Bartels-Golub 방법에 비해 수치적 안정성면에서 좋지 않다([1][18]).

일반적으로 상삼각행렬은 명시적으로 보관하되 비영요소만을 보관한다. 하삼각행렬은 역행렬을 취하여 가우스 소거 과정의 기본행연산(Elementary row operation)를 나타내는 에타파일(Eta-file)를 순차적으로 보관한다. 즉, L^{-1}와 U를 보관한다. 기저행렬의 한 열이 바뀌는 경우에 U를 수정해야 하는데 필요에 따라 U의 행과 열을 바꾸거나 비영요소가 추가로 삽입되거나 삭제되므로 이를 효율적으로 수행할 수 있는 U의 자료구조가 요구된다([4][6]).

상삼각행렬 U를 보관하는 자료구조로 Gusta-vson 구조와 Reid 등에 의해 사용된 연결리스트 구조가 알려져 있다([19]). Gustavson 구조는 같은 행(또는)열의 비영요소는 인접하도록 보관하는 구조로 페이징 시에 유리한 반면에 여유공간이 없을 때 보다 큰 기억공간을 잡아 비영요소를 옮겨 주어야 하고 기억공간의 압축 과정도 필요하게 된다. 반면 연결리스트 구조는 사용당시 정수로 링크를 나타낼 경우 비용요소의 갯수에 제한이 있고 페이징 환경에서는 좋지 않는 것으로 나타났다([19]). 그러나 최근의 컴퓨터들에서는 4 바이트 정수를 사용하고 페이징 기법에서도 많은 개선이 이루어져서 연결리스트의 효율성에 대한 재검토가 요구된다.

기저행렬의 상하분해시에는 선회요소를 선택하는 방법에 따라 분해된 상·하삼각행렬의 희소도와 수치적 안정성이 좌우된다. 상·하삼각행렬의 희소도를 높이려는 방법으로 마코위츠(Markowitz) 순서화 방법이 많이 이용되고 아울러 수치적 안정성을 유지하게 위해 역치선회연산(Threshold pivoting) 방법을 함께 사용한다([12]). 가우스소거에서 남아 있는 i번째 행과 j번째 열의 비영요소수를 각각 r_i, c_j라 할 때 마코위츠 개수(Markowitz count)는 $(r_i-1)(c_j-1)$로 정의된다([12]). 마코위츠 순서화는 선회연산시의 곱셈·덧셈연산의 횟수와 선회연산으로 새로 발생할 비영요소수를 작게 하기 위해 마코위츠 개수가 최소가 되는 요소를 선회요소로 선택하는 방법이다. 역치선회연산 방법은 비영요소의 절대값이 비영요소가 속한

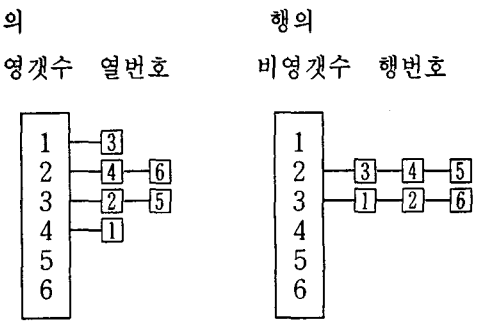
열(또는 행)의 비영요소의 절대값의 최대값의 u($0<u\leq 1$)배 이상인 요소만을 선회요소로 선택할 수 있게 제한하는 방법이다.

마코위츠 순서화의 구현

마코위츠 순서화는 마코위츠 개수를 계산하기 위해 부분행렬의 각 행과 열의 비영요소수를 알아야 한다. 부분행렬의 모든 요소에 대해 마코위츠 개수를 계산하여 최소의 마코위츠 개수를 갖는 비영요소를 검색하는 데는 많은 시간이 소요되므로 검색시간을 줄이는 방법이 요구된다. 본 연구에서는 마코위츠 순서화에서 검색시간을 줄이기 위해 행과 열의 비영요소 수를 두 개의 양방향 연결 리스트 자료구조를 이용하여 보관한다. 예를 들어 설명하면 다음 그림과 같다.

	4	3	1	2	3	2
3	X	·	·	·	X	X
3	·	X	·	X	X	·
2	X	·	X	·	·	·
2	X	·	·	X	·	·
2	·	X	·	·	X	·
3	X	X	·	·	·	X

(숫자는 각 행·열의 비영요소 갯수)



마코위츠 순서화 구현을 위한 자료구조

위의 연결 리스트에는 부분행렬의 각 행·열이 비영요소 갯수에 의해 정렬된 순서로 나타나게 된다. 선회요소의 탐색을 위해서 비영요소가 작은 행과 열 순서대로 마코위츠 개수를 계산한다. 이 때 이후에 발견될 수 있는 마코위츠 개수의 하한을 예측할 수 있는데 탐색과정에서 이 하한값보다 작거나 같은 마코위츠 개수를 갖는 비영요소를 찾으면 탐색을 멈춘다.

하한을 이용한 방법에서는 비영요소가 작은 순서대로 열과 행을 교대로 탐색하는데 비영요소가 k개의 열을 탐색할 때 이후에 발생될 수 있는 마코위츠 개수의 하한은 $(k-1)(k-1)$ 이다. 또한 비영요소가 k개인 행을 탐색할 때 이후에 발생될 수 있는 마코위츠 개수의 하한은 $k(k-1)$ 이다. 하한 설정 방법은 Duff에 의해 제안된 방법으로 평균적으로 몇 개의 행·열만 조사하면 선회요소를 찾을 수 있다고 한다([12]). 검색과정에서는 아직 검색하지 않은 비영요소들의 마코위츠 개수의 하한값을 예측하여, 선회연산 후보자의 마코위츠 개수가 이 하한값이 되는 순간 검색이 끝내게 된다([12]).

선회연산이 일어날 때마다 이를 수정함으로써 항상 정렬된 순서를 유지할 수 있는데 연결리스트 수정에는 상수 시간이 소요된다.

기저보관 자료구조

① 연결리스트 구조

연결 리스트 구조는 비영요소들을 링크로 연결해 놓은 자료구조인데 링크는 행과 열 두 가지가 있다. 연결리스트를 구현하기 위해서는 우선 비영요소를 보관하는 배열과 각 비영요소의 행 지수와 열 지수를 보관하는 배열, 각 비영요소의 같은 행 또는 열의 다음 비영요소의 위치를 가르키는 배열이 있어야 한다. 다음으로 행이나 열을 따라 가면서 비영요소를 접근할 수 있도록 각 행과 열의 비영요소의 시작 위치를 가르키는 배열이 요구된다.

상삼각행렬 U를 수정할 때 행과 열을 바꾸어 주는 경우도 있으므로 행치환과 열치환을 보관하는 배열도 필요하다. U를 연결리스트 구조로 구현할 때 필요한 배열은 다음과 같다.

BIN[] : U의 비영요소 값

BROWI[], BCOLI[] : 각 행과 열의 BIN[ ]에서의 시작 주소

BRN[], BCN[] : U의 비영요소의 행지수, 열지수

BRNEX[],BCNEX[] : 각 비영요소의 같은 행·열의 다음 비영요소의 BIN[]에서의 위치

P[] : 행 치환의 결과를 보관

Q[] : 열 치환의 결과를 보관

AVPTR : 여유공간의 시작 위치

여기서 주의할 것은 사용되지 않는 여유기억공간들은 BRNEX[]를 이용하여 연결리스트 형태로 관리된다는 점이다.

하삼각행렬은 L^{-1} 형태로 보관하는데 L^{-1}의 각 요소 값을 명시적으로 보관하지 않고 에타파일(eta-file)을 순차적으로 보관한다. 한 번의 기본 행 연산을 기억하기 위해서는 두 행의 지수와 한 행에 곱해지는 승수(Multiplier)가 필요하다. 즉, 에타파일의 하나의 요소는 (행지수1, 행지수2, 승수)로 구성된다. 사용된 배열은 다음과 같다.

BIN[] : 승수를 보관

BRN[] : 행지수1을 보관

BCN[] : 행지수2를 보관

위의 U를 보관하는데 사용되는 배열들이 L^{-1}를 보관하는데도 사용되는데 이는 기억공간을 줄이기 위해 아래 그림과 같이 각 배열의 앞부분은 U를 보관하는데 사용되고 뒤부분은 L^{-1}를 보관하는데 사용된다.

0 1 2 … … MAXBI

U	여유 공간	L^{-1}

② Gustavson 구조

Gustavson 구조는 U의 비영요소를 행별로 보관하는데, 비영요소를 보관하는 배열과 이 비영요소의 열 지수를 보관하는 배열이 있고, 각 행의 시작 주소와 비영요소 개수를 보관하는 배열이 있다. 또한, 각 열의 비영요소의 행 지수와 시작 주소, 비영요소 개수를 보관하는 배열이 있다.

Gustavson 구조는 행단위로 그 행의 비영요소를 연속적으로 저장하기 때문에 U의 행을 따라 가면서 행하는 연산에서는 유리하나 U의 열을 따라 가면서 행하는 연산에서는 연결 리스트에 비해서 불리하다. 한편, 행에 새로운 비영요소가 생길 경우에 그 행에 여유공간이 없을 경우에는 그 행 전체를 새로운 위치로 옮기는 연산이 필요하므로, 해법의 수행 도중에 자료 내부에 사용하지 않는 공간이 발생한다. 그리고 더 이상 옮길 공간이 없을 경우에는 부가적으로 비어있는 공간을 제거하는 압축 과정이 필요하다. 다음과 같은 배열을 사용하여 U를 Gustavson 구조로 구현할 수 있다.

BIN[] : 비영요소의 값을 보관.

COLN[], ROWN[] : 비영요소의 열 지수, 행지수

BCOLI[], BROWII[] : U의 각 열과 행의 BIN[]에서의 시작 주소

LENROW[], LENCOL[] : 각 행과 열의 비영요소 갯수

P[] : 행 치환의 결과를 보관
Q[] : 열 치환의 결과를 보관

L^{-1}의 보관은 연결리스트에서의 보관방법과 동일하다. 또한 연결리스트와 동일하게 U와 L^{-1}를 하나의 배열의 앞부분과 뒷부분에 나누어 저장한다.

3. 재역산

재역산의 기준

재역산은 다음과 같은 2 가지 목적을 위해 수행된다.

· 상하삼각행렬의 누적된 수치 오차 수정과 수치 안정성 확보
· 상하삼각행렬의 희소도 유지

상하분해 형태의 기저역행렬을 사용하여 단체법을 수행해 나가면서 기저역행렬이 여러 회 수정되면 컴퓨터의 계산 특성상 수치오차가 누적되어 부정확한 기저역행렬을 유지할 가능성이 커진다. 이러한 경우에는 기저행렬에 대한 상하분해를 새로이 수행하는 재역산 과정이 요구된다. 또한 상하삼각행렬이 수치적으로 불안정할 때 이후의 계산이 부정확해질 수 있으므로 미리 재역산을 통해 상하삼각행렬을 수치적으로 안정되게 유지하고자 하는 목적으로 재역산은 수행될 수 있다.

한편, 재역산이 계산의 오차를 줄이고자 하는 목적을 가지고도 있지만 계산의 오차와는 상관없이 상하분해행렬의 희소도를 가능한 한 높게 유지하고자 하는 목적으로 재역산을 수행하기도 한다. 기저수정이 반복됨에 따라, 점차로 상하삼각행렬의 밀집도(Density)가 높아지면 단체법의 수행 속도가 저하되므로, 적절한 시간에 재역산을 실시하여 상하삼각행렬을 다시 희소하게 만들어 주는 것이 전체 수행시간면에서 유리하다.

이상의 논의에서 알 수 있듯이 재역산 시점을 결정하는데 사용되는 기준도 수치적 안정성을 위한 기준과 희소도의 유지를 위한 2 가지 기준이 필요하다. 이 때 수치적 안정성와 희소도 유지간의 관련성도 중시하여 복합적으로 재역산 시점을 선정하는 방법이 바람직하다.

① 수치오차 수정과 수치안정성 확보를 위한 재역산 기준

현재의 상하분해행렬에 누적된 계산 오차를 측적하는 방법으로 $\| \boldsymbol{b} - B\boldsymbol{x} \|$ 또는 $\| \boldsymbol{\pi}B - \boldsymbol{c}_B \|$ (단, $\boldsymbol{\pi}$는 단체승수, $\boldsymbol{c}_B$는 기저변수의 목적함수 계수 벡터)이 있으나 계산량이 많고 실용적으로 정확하게 오차를 측정해 주지 못하는 문제점이 있다.

② 희소도에 의한 재역산 기준

상하분해행렬의 희소도 유지를 위한 재역산 기준으로 다음과 같은 2 가지 기준을 고려할 수 있다.

· 변동 희소도 기준
· 입력자료 희소도 기준

변동희소도 기준은 재역산 직후의 밀집도(density)를 측정하여 이를 기준으로 일정 배 이상으로 밀집도가 증가하면 재역산을 수행하는 방법이고 두번째 방법은 입력자료행렬(A)의 밀집도를 측정하여 이를 기준으로 상하삼각행렬의 밀집도가 기준 이상으로 증가하면 재역산을 수행하는 방법이다.

수치오차와 희소도를 동시에 고려한 재역산의 구현

수치적 안정성과 희소도를 높이는 것은 서로 상충되는 경우가 많다. 희소도를 높이기 위해서

는 마코위츠 순서화 방법을 사용하여 마코위츠 개수가 가장 작은 비영요소를 선회요소로 선택하는 것이 좋은데, 수치적 안정성을 위해 역치선회연산 방법을 사용하므로 마코위츠 개수가 최소이지만 수치적 안정성 때문에 선회요소로 선택되지 않는 경우가 발생한다. 따라서 수치적 안정성을 지나치게 강조하면 희소도가 나빠지게 되고 반대로 희소도를 높이다 보면 수치적 안정성이 깨질 수 있다.

수치오차와 희소도를 동시에 고려한 재역산은 수치적 안정성을 보아 가면서 역치선회연산의 u값을 동적으로 수정해 주는 방법이다([20]). 즉, 수치적으로 안정한 경우에는 u값을 감소시켜 희소도를 최대한 살릴 수 있도록 해 주고 수치적으로 불안정한 경우에는 u값을 증가시켜 수치적 안정성을 이룰 수 있도록 하는 방법이다. 이 때 상하분해에서 역치선회연산에 사용되는 u 값과 기저수정시에 사용되는 u 값을 각각 u_r, u_p로 따로 보관한다. LPAKO에서는 보통 $u_r=u_p=0.1$로 두고, 수치적으로 안정한 경우에는 $u_r=u_p=0.07$, 수치적으로 불안정한 경우에는 $u_r=0.2$, $u_p=0.3$으로 둔다.

4. 평가전략과 퇴화 해결을 위한 탈락변수 선정

평가전략

LPAKO에서는 진입변수 선정 방법으로 최대경사능선법(Steepest-edge Rule)의 한 변형인 동적 최대경사능선법(Dynamic Steepest-edge rule)([17])을 사용한다. 최대경사능선법은 할인가를 실제 개선방향의 크기로 정규화한 값을 진입변수 선택 기준으로 삼는다.

최대경사능선법에서 $\gamma_j=\|\eta_j\|$ (단, $\eta_j=[-B^{-1}A_{.j},0,..,1,..0]^T$)의 값을 유지하는 방법은 다음과 같다. 기저가 B이고 $\|\eta_j\|$ 값을 알고 있다고 하자. 진입변수가 x_p, 탈락행이 r, 즉 x_r이 탈락된다고 하고 수정된 기저가 $\overline{B}$, $\overline{\eta_j}=[-\overline{B^{-1}A_{.j}},0,..,1,..0]^T$일 때 $\overline{\gamma_j}=\|\overline{\eta_j}\|$을 계산해야 한다. 다음 식을 통해 매 회 $\overline{\gamma_j}$를 갱신해 갈 수 있다([15][17]).

$$\overline{\eta_r}=-\frac{1}{\alpha_p}\eta_p$$

$$\overline{\eta_j}=\eta_j-\overline{\alpha_j}\eta_p,\ j>m,\ j\neq p$$

단, $\alpha_{p=}(B^{-1}A_{.p})_r=\overline{a_{rp}}$, $\overline{\alpha_j}=e_r^TB^{-1}A_{.j}$

이고,

$$\overline{\gamma_r}=\frac{\gamma_p}{\alpha_p^2},$$

$$\overline{\gamma_j}=\gamma_j-2\overline{\alpha_j}v^TA_{.j}+\overline{\alpha_j^2}\gamma_p,\ j>m,\ j\neq p$$

단, $v=B^{-T}\overline{A_{.p}}$

이다.

최대경사능선법을 사용할 때의 장점은 반복횟수를 줄일 수 있다는 점이고, 단점은 계산량이 많다는 점이다. 위의 계산과정을 보면 일반적인 단체법과 비교해서 최대경사능선법에서는 추가적으로 $v=B^{-T}\overline{A_{.p}}$와 $\sigma=e_r^TB^{-1}$를 계산한다.

최대경사능선법의 한 변형인 동적 최대경사능선법(Dynamic steepest-edge rule)에서는 γ_j를 계산하는데 η_j의 모든 요소를 다 고려하는 것이 아니라 특정한 요소만 고려한다([17]). 즉, 참조프레임(Reference framework)에 있는 변수에 해당하는 요소만 γ_j를 계산하는데 고려한다. 처음에 비기저변수만을 참조프레임 R에 넣고 시작한 후 기저변수가 비기저변수로 빠져 나갈

때 빠져 나가는 기저변수가 R에 속하지 않으면 이 변수를 새로이 R에 추가한다. 따라서 R에 속하는 변수의 개수는 점점 많아지게 되고 많은 반복이 진행된 후에는 최대경사능선법과 동일하게 된다. 처음에 R에는 비기저변수만 있으므로 γ_j=1로 설정된다.

최대경사능선법은 그 자체로는 많은 양의 계산을 필요로 하는데 최대경사능선법에 요구되는 계산을 할인가와 단체승수 수정에 사용하면 전체 계산시간을 줄일 수 있다([15]). 할인가의 수정으로 비기저 변수 전체의 할인가를 이용할 수 있어 반복회수를 줄일 수 있다. 현재 비기저변수의 할인가를 $\overline{c_j}$라 하고 다음 회에서의 할인가를 $\tilde{c_j}$라 하자. 또한 현재의 단체승수를 π라 하고 다음 회에서의 단체승수를 $\tilde{\pi}$라 하자. 그러면 다음의 관계식이 성립한다([15]).

$$\tilde{c_j} = \overline{c_j} - \overline{c_q}(\overline{a_{pj}} / \overline{a_{pq}}),$$

$$\tilde{\pi} = \pi - (\overline{c_q} / \overline{a_{pq}}) B^{-T} e_p$$

또한 동적 최대경사능선법을 구현하면서 여러 가지 계산의 효율화를 이룰 수가 있는데, 우선 γ_j 값 및 단체승수값의 수정시 필요한 부분만 수정함으로써 계산량을 줄일 수 있다. 그리고 σ, v 각각의 계산에서 필요한 기저역행렬 연산을 따로 하지 않고 한꺼번에 행할 수도 있다. 즉, 한번의 기저역행렬 자료구조의 접근으로 두 가지 계산을 동시에 수행할 수 있다. 그리고 γ_j을 계속해서 수정해 가다 보면 오차가 누적될 수 있는데 그것을 보정하기 위해 일정 수행 횟수마다 새로이 계산해야 한다.

퇴화 해결를 위한 탈락변수 선정 방법

대형문제에서 브랑법(Bland's Rule)은 실제 퇴화문제를 해결하는데 효과가 없어 $l-\delta e \le x \le u+\delta e$ (단, $\delta > 0$와) 같이 상·하한을 섭동시켜서 탈락변수를 선정하는 방법을 많이 사용하고 있다. 위의 방법의 대표적인 예가 Devex 방법([16])과 MINOS의 EXPAND 프로시저([14])이다. LPAKO는 EXPAND 프로시저를 변형한 방법을 사용하고 있다.

EXPAND 프로시저의 기본적인 개념은 개선폭이 항상 양수가 되도록 변수의 상하한을 섭동시키고 매회 섭동시키는 양(δ)을 늘려가다가 일정 시점마다 다시 초기값으로 재설정하는 과정을 반복하는 것이다.

EXPAND 프로시저를 의사코드(Pseudo-code)로 표현하면 다음과 같다.

```
procedure EXPAND
  (α1, γ1) ← 최소비율검정 ( x,p,
  l−δe,u+ δe, r)
    γ←0;  p_max ←0;
    for j=1 until n do
      α_j ← 비율(x_j, p_j, l_j, u_j);
      if α_j≤α1 and |p_j| > p_max then
         γ← j;  α2 ← α_j; p_max ← |p_j|;
      end if
    end for
    α_min ← τ / p_max ;
    α ← max {α2, α_min}
```

위의 프로시저를 적용할 때에는 매회 δ을 일정값(τ)만큼 증가시키고, 그 값이 최대가능허용오차를 벗어나거나 일정회수(K) 수행 이후에는 다시 δ의 값을 초기가능허용오차값으로 재설정한다. EXPAND 프로시저에서는 기저변수가 탈락되면서 비기저변수가 될 때, 그 값이 원래의 상·하한을 벗어나므로 그 값을 보관해야 할 필요가 있다.

LPAKO에서 사용하고 있는 퇴화 방지 방법

은 EXPAND 프로시저의 변형 형태로 EXPAND와는 다음 세가지 점에서 차이가 있다.

첫째, 항상 EXPAND 프로시저를 사용하는 것은 아니다. 즉, 퇴화가 심하게 일어날 것이라고 예상되는 상황에서만 EXPAND 프로시저를 사용한다. EXPAND를 매회 사용하게 되면 비가능성이 누적되어 오히려 수행회수가 늘어나는 경우가 발생할 수 있다. 그러므로 퇴화가 아주 많이 발생할 때에만 EXPAND 프로시저를 사용하는 것이 오히려 수행도면에서는 좋을 수 있다. LPAKO에서는 두가지 상황에서 EXPAND 프로시저를 사용하는데, 상한 또는 하한값에 0.001 이하로 근접한 기저 변수가 전체 기저변수의 90% 이상일 경우와 최소비율검정에서 0이 나오는 반복횟수가 연속해서 100번 이상 일어날 경우에 EXPAND 프로시저를 사용되게 된다.

둘째, 여러 가지 매개변수의 선정이 다르다. LPAKO에서는 효율적인 매개변수의 값을 다음과 같이 실험적으로 구하였다.

초기가능허용오차 = 10^{-8}, 최대가능허용오차 = 10^{-6}, $\tau = 10^{-10}$, $K = 300$

셋째, 비기저변수의 값은 보관하지 않는다. EXPAND에서는 기저가 탈락되면 그 값이 원래의 상한이나 하한에 있지 않고 δ 만큼 벗어나 있게 되므로 그 값을 보관하게 하고 있지만 LPAKO에서는 그렇지 않다. 즉, 기저변수의 값만 유지하고 비기저의 변수는 그 상태(상한 또는 하한)만을 표시하고 있다. 비기저 변수값까지 보관하게 되면 저장공간, 추가되는 계산량이 너무 크기 때문이다.

5. 사전처리와 규모화

사전처리

LPAKO의 사전처리는 크게 문제확인, 치환에 의한 행제거, 행·열에 대한 처리 등의 3단계로 구성된다. 문제확인 단계에서는 비영요소가 없는 행에 대한 처리를 수행하고 각 변수의 상한과 하한을 확인하여 상한이 하한보다 작은 변수가 있으면 입력된 문제를 비가능으로 판정하고 상한과 하한이 같은 변수는 값을 고정시켜 제거한다.

치환에 의한 행처리에서는 2개의 변수만으로 구성된 '='형태의 제약식을 이용하여 다른 한 변수를 치환하여 문제에서 제거하는 과정이다.

행에 대한 처리에서는 각 행이 가질수 있는 최대값과 최소값을 구하여 이를 우변상수와 비교하여 비가능인가를 판정하고 최대값 또는 최소값이 우변상수값과 같으면 그 행에 속한 변수들의 값을 고정함으로써 문제에서 제거할 수 있다.

열에 대한 처리에서는 변수의 값을 상한 또는 하한으로 고정시키더라도 다른 변수들의 가능해 집합에 영향을 주지 않는 경우 목적함수 계수에 따라 해당 변수를 상한 또는 하한으로 고정시킨다.

LPAKO에서는 사전처리된 문제를 풀어 구한 해로부터 사전처리하기 전의 원문제의 해를 복원하여 출력한다.

규모화

규모화가 나쁜 문제(Poorly scaled problems)들은 역행렬이 수치적으로 불안정하게 되므로, 정확한 해를 구하지 못하는 경우가 생기게 된다([8]). 따라서 이러한 행렬의 경우 값의 범위를 고르게 만들어 수치오차를 줄이는 규모화(Scaling) 과정이 필요하게 된다. LPAKO에는 다음과 같이 4 가지의 규모화 방법이 구현되어 있다.

· 산술평균에 의한 방법

· 최대값에 의한 방법
· 기하평균에 의한 방법
· 기하평균값 규모화 이후 최대값 규모화 방법

첫번째 방법은 행과 열의 산술평균값을 구한 다음 그 값으로 행과 열을 나누어 규모화를 수행한다. 두번째와 세번째 방법은 각각 행과 열의 최대값과 기하평균값으로 나눈다. 마지막 방법은 먼저 기하평균에 의한 규모화를 수행하고 그 결과의 행렬에 대해 다시 최대값에 의한 규모화를 수행한다.

6. 초기기저

구조변수 사용 순서

초기기저는 단체법의 전체 반복회수에 큰 영향을 주고 초기기저의 밀집도가 커지면 전체 수행시간에도 영향을 미치게 된다. 따라서 초기기저는 희소도가 높고 최적기저에 근접할수록 좋다([7][10]).

LPAKO의 초기기저 구성 방법은 Bixby가 제안한 초기기저 구성 방법에서 구조변수들을 초기기저에 포함시키는 우선 순위와 구조변수들간의 일차독립성을 판단하는 부분을 수정·보완한 방법을 사용하고 있다([10]).

구조변수 사용 순서는 초기기저를 희소하게 유지하여 수행 시간을 줄이기 위해 구조변수를 희소도 순서로 사용하는 방법이다. 즉, 단일요소열(Singleton column)을 최우선으로 사용하고 이후 비영요소가 2개, 3개...인 구조변수열을 비영요소 개수의 오름차순으로 사용한다. 이 때 비영요소수가 같으면 상·하한의 폭이 넓은 변수를 우선적으로 사용하고 상하한 폭이 같으면 목적함수의 계수를 참조하여, 최대화 문제의 경우 목적함수 계수의 값이 큰 변수를 우선적으로 사용한다. 이 방법은 각 구조변수 x_j에 대해 다음과 같은 기준함수 $p_j(x_j)$을 정의함으로써 명료하게 서술할 수 있다([7]).

$$p(x_j) = \tau_j M^2 + \frac{1}{u_j - l_j} M + S_z S_b c_j$$

τ_j : j 열의 비영요소 갯수
u_j : x_j의 상한
l_j : x_j의 하한
c_j : j 열의 목적함수 계수
S_z : 목적함수의 형태에 따라,

최대화 문제이면 -1, 최소화 문제이면 1

$$S_b : \begin{cases} sign(u_j) & , \ |u_j| \geq |l_j| \\ sign(l_j) & , \ |u_j| < |l_j| \end{cases}$$

M : 큰 수

여기서 $p(x_j)$를 구하여 이 값의 오름차순으로 구조변수를 정렬한 순서를 구조변수 선택의 순서로 사용한다. $p(x_j)$의 첫번째 항은 희소도에 대한 고려, 두번째 항은 상·하한의 폭에 대한 고려이고, 마지막 항은 목적함수 계수의 값에 대한 고려이다.

구조변수간의 일차독립성

가우스 소거를 이용한 일차독립성 방법은 많은 연산량을 필요로 하여 큰 문제의 경우 초기기저의 구성에 많은 시간을 소모한다. 반면, 가우스 소거 연산을 사용하지 않는 Bixby의 발견적 기법은 일차독립이면서도 초기기저로 사용하지 못하게 되는 구조변수가 많다. 두 방법의 이러한 이러한 단점을 개선하기 위해 LPAKO에서는 선회연산을 통하지 않고 비영요소의 위치로만 일차독립성을 확인하는 발견적 기법인 비중복 비영요소법을 사용한다([6]).

여기서는 다음과 같은 성질을 이용한다. $a_{rs}=0$, $a_{rt}\neq 0$인 r이 존재하면 $A_{\cdot s}$, $A_{\cdot t}$는 일차독립이다. 이는 $A_{\cdot s}$, $A_{\cdot t}$의 일차결합(Linear combination)으로 r번째 행에 0을 만들어내기

위해서는 계수가 모두 0일 수 밖에 없다는 사실로부터 자명하다. 이 성질을 k개의 열에 대해 확장할 수 있다. k>1일 때 k-1개의 열 $A_{\cdot j_1}, A_{\cdot j_2}, \cdots, A_{\cdot j_{k-1}}$이 일차독립이라고 하자. 여기서 $a_{ri_1} = a_{ri_2} = \cdots = a_{ri_{k-1}} = 0$이고 $a_{ri_k} \neq 0$인 r이 존재하면 k개의 열 $A_{\cdot j_1}, A_{\cdot j_2}, \cdots, A_{\cdot j_{k-1}}, A_{\cdot j_k}$는 일차독립이다. 이 방법은 이전에 초기기저로 선택된 구조변수열과 비교하여 비영요소의 행의 위치가 겹치지 않는 비영요소의 존재 여부로 일차독립성을 판별하므로 비중복 비영요소법으로 일컫기로 한다. 이 방법은 비영요소의 위치로만 일차독립성을 판별하므로, 선회연산이 없어 수행시간이 짧은 장점이 있다. 그러나, 위의 성질의 역은 성립하지 않아 일차독립이지만 초기기저로 선택되지 못하는 열이 있을 수 있으나, 실험적으로 Bixby가 제안한 발견적 기법보다 더 많은 구조변수를 초기기저에 포함시킬 수 있다. 또, 처음에 선택한 열의 밀집도가 높을수록 조건을 만족하기 어려워지므로, 비중복 비영요소법의 경우 구조변수의 사용 순서를 희소도 순서로 하였을 때 인공변수를 더욱 적게 사용하게 된다.

7. 최적판정과 비가능판정

최적판정

단체법을 사용하여 상하한 제약이 있는 일반한계 선형계획문제를 풀 때 최적조건은 다음과 같다. 단, $\overline{c_j}$는 j번째 변수의 할인가를 의미한다.

모든 변수 x_j가

$x_j = l_j$이면, $\overline{c_j} \geq 0$

$x_j = u_j$이면, $\overline{c_j} \leq 0$

이다.

실제 단체법을 구현할 때는 위의 이론적인 최적조건을 정확하게 만족하지 않는 경우가 많다. 이는 컴퓨터의 계산오차 때문인데 이러한 경우를 위해서 최적조건을 약간 완화하여 최적성을 판단하게 된다.

LPAKO에서는 최적조건을 완화하여 다음과 같은 조건을 만족하면 현재의 기저해를 최적해로 간주하고 프로그램 수행을 종료하게 된다. 이 때, LPAKO에서는 $\varepsilon = 10^{-6}$으로 설정하고 있다.

모든 변수 x_j가

$x_j = l_j$이면, $\overline{c_j} \geq -\varepsilon$

$x_j = u_j$이면, $\overline{c_j} \leq \varepsilon$

이다.

비가능성 판정

2국면법을 사용하는 단체법에서 비가능성(Infeasibility) 판정은 국면 1의 최적해의 목적함수값이 0이 아니면 비가능으로 판정하게 된다. 그러나 최적판정의 경우와 마찬가지로 비가능 문제라 하더라도 계산오차에 인해 국면 1의 최적 목적함수 값이 정확히 0이 되지 않는 경우가 자주 발생한다.

LPAKO에서의 비가능성 판정는 국면 1이 최적인가를 판단하여 최적인 경우 국면 1의 목적함수 값을 보고 결정하게 된다. 국면 1에서 최적해에 도달했는가는 앞 절의 최적판정에 설명된 방법을 사용하고 국면 1의 목적함수값이 ξ보다 크면 주어진 문제를 비가능으로 판정한다. 이 때 ξ는 10^{-6}으로 설정되어 있다.

8. 실험결과 및 결론

단체법 프로그램인 LPAKO ver 3.3f에는 앞에서 언급한 여러 가지 기법들이 구현되었다. LPAKO는 C언어를 사용하고 UNIX와 DOS 환경 모두에서 수행 가능하다. 기저행렬은 마코위츠 순서화를 사용하여 상하분해하였고 상삼각행렬은 행·열연결리스트 형태로 비영요소만 보관하였다. 그리고 기저역행렬의 수정은 Reid법을 사용하였다. 재역산은 수치오차 소도를 동시에 고려한 방법을 사용하였고, 진입변수선정방법은 동적 최대경사능선법을 채택하였으며, 퇴화방지를 위해 EXPAND 프로시저와 Devex 방법을 절충한 방법을 사용하였다. 사전처리와 기하평균 규모화를 사용하였으며 희소도에 비중을 두는 초기기저 선정 방법을 채택하였다.

<표 1> LPAKO와 MINOS, CPLEX의 비교 실험결과

문제이름	문제 크기			LPAKO ver 3.3f			MINOS ver 5.3			CPLEX ver 4.0.7		
	행수	열수	Nonz	시간	IT	목적함수 값	시간	IT	목적함수 값	시간	IT	목적함수 값
BNL1	644	1175	6129	3.88	1170	1.97762956IE+03	5.2	1300	1.97762943IE+03	1.69	1838	1.97762956IE+3
SCFXM2	661	914	5229	1.62	580	3.666026156E+04	3.1	774	3.666026156E+04	0.53	615	3.666026156E+4
SHIP08S	779	2387	9501	0.69	250	1.920098210E+06	2.5	256	1.92009821IE+06	0.37	304	1.920098210E+6
SHIP08L	779	4283	17085	2.34	429	1.909055211E+06	4.6	473	1.909055211E+06	0.80	470	1.909055211E+6
25FV47	822	1571	11127	15.07	2001	5.501845888E+03	30.4	5997	5.501845888E+03	6.83	2558	5.501845888E-3
SCFXM3	991	1371	7846	3.58	884	5.490125454E+04	5.6	1120	5.490125455E+04	0.98	870	5.490125455E-4
TRUSS	1001	8806	36642	70.98	6223	4.588158471E+05	*	*		18.57	9287	4.588158471E+5
SCTAP2	1091	1880	8124	1.53	431	1.724807142E+03	3.7	604	1.724807143E+03	0.73	702	1.724807142E-3
WOODW	1099	8405	37478	6.12	916	1.304476333E+00	*	*		2.15	1310	1.304476333E+0
SHIP12L	1152	5427	21597	4.20	705	1.470187919E+06	8.5	961	1.470187919E+06	1.70	661	1.470187919E+6
SHIP12S	1152	2763	10941	1.13	367	1.489236134E+06	4.0	437	1.489236134E+06	0.49	358	1.489236134E+6
SCTAP3	1481	2480	10734	2.67	557	1.424000000E+03	6.8	1027	1.424000000E+03	1.08	841	1.424000000E+3
DEGEN3	1504	1818	26230	42.01	2352	-9.872940000E+02	*	*		19.49	4376	-9.872940000E-2
STOCFOR2	2158	2031	9492	5.78	710	-3.902440853E+04	25.9	3041	-3.902440854E+04	2.62	1456	-3.902440853E+4
D2Q06C	2172	5167	35674	158.69	7168	1.227842108E+05	*	*		64.08	8824	1.227842108E+5
BNL2	2325	3489	16124	29.78	3034	1.811236540E+03	*	*		8.47	4671	1.811236540E+3
STOCFOR3	16676	15695	74004	706.10	7646	-3.997678394E+04	*	*		129.45	12007	-3.997678394E-4
FIT1D	25	1026	14430	2.30	1033	-9.146378092E+03	4.4	2606	-9.146378092E+03	0.66	1137	-9.146378092E+3
KB2	44	41	291	0.04	36	-1.749900129E+03	0.1	50	-1.749900130E+03	0.01	26	-1.749900129E-3
RECIPE	92	180	752	0.03	19	-2.666160000E+02	0.1	27	-2.666160000E+02	0.02	38	-2.666160000E-2
GROW7	141	301	2633	0.23	141	-4.778781181E+07	0.6	175	-4.778781181E+07	0.26	225	-4.778781181E-7
BOEING2	167	143	1339	0.08	120	-3.150187280E+02	0.3	159	-3.150187280E+02	0.09	172	-3.150187280E-2
VTPBASE	199	203	914	0.04	70	1.298314624E+05	0.3	151	1.298314625E+05	0.05	24	1.298314624E-5
BORE3D	234	315	1525	0.06	42	1.373080394E+03	0.4	118	1.373080394E+03	0.08	41	1.373080394E-3
CAPRI	272	353	1786	0.28	242	2.690012913E+03	0.6	252	2.690012914E+03	0.16	308	2.690012913E-3
GROW15	301	645	5665	1.63	394	-1.068709413E+08	1.9	426	-1.068709413E+08	1.14	612	-1.068709412E-8
BOEING1	351	384	3865	0.75	513	-3.352135675E+02	1.4	597	-3.352135675E+02	0.41	566	-3.352135675E-2
STAIR	357	467	3857	1.90	398	-2.512669511E+02	2.2	578	-2.512669512E+02	0.63	318	-2.512669511E-2
STANDATA	360	1075	3038	0.16	83	1.257699500E+03	0.7	131	1.257699500E+03	0.15	51	1.257699500e-3
ETAMACRO	401	688	2489	0.83	543	-7.557152333E+02	1.8	697	-7.557152183E+02	0.32	615	-7.557152333E-2

* : 풀지 못함

Netlib 문제([13])에 대한 LPAKO ver 3.3f와 MINOS ver 5.3, CPLEX ver 4.0.7의 비교 실험 결과는 다음 [표 4.2]와 같다. 실험기종은 SUN Ultra 170으로 주기억 공간 64MB이고 LPAKO는 GCC ver 2.0의 컴파일 옵션 -O2를 주어 컴파일 하였고 MINOS는 최적화 옵션 '-xO5'을 주어 컴파일 하였다.

<표 1>에서 Nonz는 행렬 A의 비영요소의 개수를 의미하고 IT는 국면 1과 2를 포함된 전체 반복회수이다. 목적함수 값은 각 프로그램의 최적해의 목적함수 값을 의미하고 시간은 CPU 사용시간을 초(Second) 단위로 측정한 것이다.

LPAKO는 수행속도면에서는 MINOS에 비해 평균적으로 2~3배정도 빠르고 반복회수도 MINOS에 비해 작았다. 반면, CPLEX에 비해서는 수행시간면에서는 평균 2~3배 정도 느리지만 반복횟수 면에서는 작은 것으로 나타났다. 목적함수 값을 비교해 보면 거의 대부분의 문제에서 LPAKO와 CPLEX는 유효자리수 10자리까지 같았으나 MINOS의 경우는 유효자리 7자리 이하에서 목적함수 값이 다른 경우도 있었다.

실험에서 나타난 바와 같이 단체법 프로그램 LPAKO에서 사용한 기저행렬 보관 방법, 진입변수 선정 방법, 퇴화 방지와 재역산 전략, 사전처리와 규모화, 초기기저 구성 방법 등은 상당히 효율적이며 LPAKO는 대형문제에서도 안정적으로 최적해를 구해 준다. 그러나, 수행시간면에서 상용 프로그램과의 속도 차이을 줄이기 위한 추후 연구가 요구된다.

참 고 문 헌

[1] 박순달, 선형계획법(3정판), 민영사, 1992

[2] 김우제, 강완모, 김민정, 박순달, "일반한계 선형계획법에서 비영요소만 보관하는 자료구조와 평가전략의 효율성에 관한 연구," 전산활용연구, 제6권 1호(1994), pp. 55-66.

[3] 김우제, 안재근, 박순달, "단체법 프로그램의 효율화와 통합," 경영과학, 제11권 3호(1994), pp. 11-26.

[4] 김우제, 안재근, 서용원, 성명기, 박순달, "단체법에서 기저행렬과 입력자료의 보관방법과 자료구조," 한국경영과학회/대한산업공학회 '95 춘계 공동학술대회 논문집, 1995.

[5] 박찬규, 임성묵, 박순달, "동적 steepest-edge 방법과 퇴화방지기법의 구현," 한국경영과학회/대한산업공학회 '97 춘계 공동학술대회 논문집, 1997

[6] 서용원, 김우제, 박순달, "선형계획법프로그램의 수치오차보정과 행렬희소도 유지," 한국경영과학회 '95 추계학술대회 논문집, 1995

[7] 서용원, 김우제, 박순달, "단체법에서의 초기기저 구성에 관한 연구," 경영과학, 제13권 3호(1996), pp. 105-113.

[8] 안재근, 김우제, 박순달, "단체법에서의 규모화와 허용오차," 전산활용연구, 제6권 1호(1994), pp. 29-39.

[9] Bartels. R. H., G. H. Golub., "The Simplex method of linear programming using LU decomposition," *Communication of ACM* 12(1969), pp. 266-268.

[10] Bixby, R. E., "Implementing the Simplex Method : The Initial Basis," *ORSA J. on Computing*, Vol. 4, No. 2, 1992, 267-284.

[11] Bixby, R. E., "Progress in Linear Programming," *ORSA J. on Computing*, Vol. 6, No. 1, 1994, pp. 15-22.

[12] Duff. I. S., A. M. Erisman, J. K. Reid,

Direct Methods for Sparse Matrices, Clarendon Press, Oxford 1986.

[13] Gay, D.M., "Electronic mail distribution fo linear programming test problems," *Mathematical Programming Society COAL Newsletter*(1985).

[14] Gill, Philip E., W. Murray , M. A. Saunders, M. H. Wright, "A practical anti-cycling procesure for linearly constrained optimization," *Mathematical Programming* 45(1989), pp. 437-474.

[15] Goldfarb, D., "Using the steepest-edge simplex algorithm to solve sparse linear programs," in : J. Bunch and D. Rose, eds., *Sparse Matrix Computations*, Academic Press, New York(1976), pp. 227-240.

[16] Harris, P. M. J.,, "Pivot selection methods for the Devex LP code," *Mathematical Programming* 5(1973), pp. 1-28.

[17] Forrest, J. J., Donald Goldfarb, "Steepest-edge simplex algorithms for linear programming," *Mathematical Programming* 57(1992), pp. 341-374.

[18] Forrest. J. J. H., J. A. Tomlin, "Updating triangular factors of the basis to maintain sparsity in the product-form simplex method," *Mathematical Programming* 2(1972), pp. 263-278.

[19] Reid. J. K., "Fortran Subroutines for Handling Sparse Linear Programming Bases," *Computer Science and Systems Devision*, A.E.R.E., Harwell R.8269, 1976, pp. 1-23.

[20] Reid J. K., "A Sparsity-Exploiting Variant of the Bartels-Golub Decomposition for Linear Programming Bases," *Computer Science and Systems Devision*, A.E.R.E., Harwell, CSS20, 1975, pp. 1-23.

[21] Saunders, M. A., "A fast, stable implementation of the Simplex method using Bartels-Golub updating," in : J. Bunch and D. Rose, eds., *Sparse Matrix Computations*, Academic Press, New York (1976), pp. 213-226.