선형계획법 프로그램의 수치오차보정과 행렬희소도 유지

서용원*, 김우제**, 박순달*

* 서울대학교 산업공학과, ** 대진대학교 산업공학과

본 연구에서는 선형계획법 프로그램의 수치오차보정과 행렬희소도 유지를 통하여 수행 속도를 향상시키는 방안에 대해 다룬다. 수치오차를 줄이기 위해 규모화를 도입하였으며, 계산 과정에서의 상하삼각행렬의 수치오차를 근사적으로 측정하는 방법을 고려하였다. 기저행렬의 상하분해에 널리 사용되는 Markowitz 순서화의 효율적인 구현에 대해 연구하였으며, Reid의 기저수정방법의 효율성에 대해 실험적으로 검토하였다. 또, 행렬의 희소도에 의한 재역산 기준을 수립하여 수행 속도를 개선하였다.

1. 서론

선형계획법은, 이론 자체의 발전 뿐 아니라, 구현된 프로그램에도 많은 발전이 있어왔다. 기저역행렬의 경우 초기에는 명시형 또는 적산형으로 B^{-1} 전체를 보관하고 사용하였으나, 현대적인 프로그램들에서는 대개 수치안정성과 행렬의 희소성을 고려하여 상하분해형으로 기저역행렬을 보관하여 사용한다.

상하분해형으로 기저역행렬을 사용하기 위해서는 크게 기저행렬을 상하삼각행렬로 분해하는 상하분해(Factorizing)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 있어왔으며, 이들 중 상하분해시에 Markowitz 순서화, 수정방법에 Reid의 방법이 효율적으로 알려져 널리 이용되고 있다. 또한, 이들의 효율적인 구현에 대해서도 많은 연구가 이루어졌다.[8][9][15][18][19]

한편, 프로그램이 안정적으로 해를 구해내기 위해서는 수치오차에 대한 각종 고려가 필요하다. 예컨대 규모화나 수치오차의 측정방법 등을 들 수 있다. 이들의 설계에 따라 프로그램의 정확도와 수행 시간은 많은 차이를 가져오게 된다.

본 연구에서는 상하분해, 기저수정, 규모화, 수치오차 측정및 보정방법을 구현, 이들의 성능을 비교·분석해 보고, 안정적인 상하분해 단체법과의 효율화에 대해 고려하고자 한다.

2. 수치오차보정

2.1 규모화

문제에 따라서는 입력 행렬내의 값들의 편차가 커서 역행렬을 구할 때 오차가 크게 발생하게 되는 경우가 있다. 이와 같이 규모화가 나쁜 문제

(Poorly scaled problems)들은 역행렬이 수치적으로 불안정하게 되므로, 정확한 해를 구하지 못하는 경우가 생기게 된다. 따라서 이러한 행렬의 경우 값의 범위를 고르게 만들어 수치오차를 줄이는 규모화(Scaling) 과정이 필요하게 된다. 규모화에는 여러가지 방법이 있으며, 여기서는 그들 중 몇가지를 고려한다.

① 산술평균에 의한 방법

행과 열에 대해서 산술평균값을 계산한 다음 그 값으로 나누어 규모화를 수행한다.

② 최대값에 의한 방법

행과 열에 대해서 최대값을 계산한 다음 그 값으로 나누어 규모화를 수행한다.

③ 기하평균에 의한 방법

행과 열에 대해서 기하평균값을 계산한 다음 그 값으로 나누어 규모화를 수행한다.

④ 기하평균값 규모화 이후 최대값 규모화 방법

먼저 기하평균에 의한 규모화를 수행하고, 그 결과의 행렬에 대해 다시 최대값에 의한 규모화를 수행한다.

이상의 방법들을 수치오차의 측면에서 비교하기 위해 매 Iteration마다 Max $|b_i - \sum_j B_{ij}X_j|$ (b는 우변상수, B는 기저행렬, X는 계산된 해)를 구하여 전체 Iteration에 대해 평균하여 비교하였다. 실험 결과를 [표 1]에 나타내었다.

[표 1]에서, 규모화를 하지 않는 경우에 비하여 산술평균 규모화, 최대값 규모화, 기하평균 규모화, 기하평균 이후 최대값 규모화가 각각 13%, 8.0%, 33%, 2.5%의 오차를 갖는 것으로 나타나, 기하평균 이후 최대값 규모화가 수치오차를 줄이는데 가장 효과가 큰 것으로 나타났다. 수행 시간에서는 규모화 방법간에 유의한 차이를 발견할 수 없었다.

2.2 수치오차의 측정과 재역산

현재의 상하삼각행렬이 수치적으로 불안정한 경우, 이후의 계산 과정에서 누적된 수치오차로 인해 부정확한 해를 구하게 되는 문제가 발생한다. 따라서, 계산 과정에서 수치오차를 측정할 수 있는 방법이 필요하게 된다. 기존에는 다음과 같은 오차측정방법이 사용되었다.

· 기저역행렬과 관련한 오차측정: $\| I_{ii} - B_{ik}^{-1}B_{ki} \|$
· 원문제의 해와 관련한 오차측정: $\| b - BX \|$

실험기종 : SUN SPARC 10

방법 / 문제명	규모화없음		산술평균		최대값		기하평균		기하평균+최대값	
	[1]	[2]	[1]	[2]	[1]	[2]	[1]	[2]	[1]	[2]
scagr7	1.23e-12	1.2	2.58e-12	1.6	7.27e-13	1.4	1.74e-12	1.3	9.09e-13	1.3
scsd6	2.37e-09	13.6	2.19e-09	23.4	2.37e-09	14.0	3.42e-09	20.9	1.13e-09	20.6
israel	6.04e-10	4.9	8.38e-12	6.6	1.16e-11	5.3	2.10e-09	5.0	2.36e-11	6.1
e226	3.86e-11	16.0	2.96e-13	9.3	2.00e-13	11.6	7.90e-12	11.3	1.23e-14	9.5
boeing1	4.66e-10	25.8	2.10e-11	19.0	4.14e-12	17.1	7.58e-12	16.4	3.69e-12	18.6
stair	8.83e-12	41.1	1.97e-11	41.2	8.33e-12	43.2	4.36e-10	43.1	1.81e-12	40.9
pilot4	9.45e-05	123.3	1.05e-09	130.4	1.62e-08	119.3	2.46e-08	119.7	1.85e-10	120.6
agg3	1.56e-09	7.6	6.89e-09	7.2	1.21e-08	6.8	1.59e-09	7.5	3.08e-10	7.6
ship12s	3.72e-14	20.8	1.01e-13	20.1	3.70e-14	20.1	3.91e-14	19.6	2.88e-14	20.2
sierra	2.16e-11	52.5	1.04e-11	52.0	2.69e-12	49.5	1.15e-11	54.3	2.95e-12	51.9

[1] Max $|b_i - \sum_j B_{ij}X_j|$ [2] 수행시간(초)

〔표 1〕 규모화 방법들의 비교

· 쌍대해와 관련한 오차측정: $\| \pi B - C_B \|$

이들 중 기저역행렬과 관련한 오차측정기준 $\| I_{ii} - B_{ik}^{-1} B_{ki} \|$ 의 경우 상하분해형으로 기저역행렬을 보관하는 경우 매우 많은 계산량을 요구하므로 비실용적이다. 나머지 두 방법의 경우 계산량은 그리 많지 않으나 실용적으로 정확하게 오차를 측정해 주지 못하는 문제점이 있었다. 따라서, 본 연구에서는 다음과 같은 두가지 방법을 고려하였다.

① 대각요소 최소·최대 절대값의 비율에 의한 방법

상삼각행렬 대각요소의 최소·최대 절대값을 구하여 이들의 비율에 의해 안정성을 측정하는 방법이다. 비율이 어느 수준을 넘으면 재역산을 실시한다.

② 대각요소 최소 절대값에 의한 방법

대각요소의 최대 절대값에는 큰 의미가 없다고 보고, 최소 절대값이 0에 가까운 정도로서 안정성을 측정한다. 최소 절대값이 특정한 값 이하이면 재역산을 실시한다.

두가지 방법을 비교 실험한 결과가 [표 2]에 나타나 있다.

실험기종 : SUN SPARC 10

방법 / 문제명	오차측정 없음		최대최소 절대값비율		최소절대값 기준	
	[1]	[2]	[1]	[2]	[1]	[2]
scagr7	9.09e-13	1.1	9.09e-13	1.4	9.09e-13	1.3
scsd6	1.13e-09	20.1	1.13e-09	21.6	1.13e-09	20.6
israel	2.36e-11	5.7	2.36e-11	6.3	2.36e-11	6.1
e226	1.23e-14	8.9	1.23e-14	9.9	1.23e-14	9.5
boeing1	3.69e-12	17.4	3.69e-12	19.5	3.69e-12	18.6
stair	3.68e-12	42.1	3.68e-12	46.2	1.81e-10	40.9
pilot4	3.65e-10	115.5	3.65e-10	125.7	1.85e-10	120.6
agg3	3.08e-10	7.0	3.08e-10	7.7	3.08e-10	7.6
ship12s	2.88e-14	19.5	2.88e-14	20.6	2.88e-14	20.2
sierra	2.95e-12	49.8	2.95e-12	53.0	2.95e-12	51.9
pilotnov	1.21e-07	436.9	1.21e-07	444.6	3.89e-09	381.1

[1] Max $| b_i - \sum_j B_{ij} X_j |$ [2] 수행시간(초)

〔표 2〕 수치오차측정방법 비교

[표 2]에서, 실험 대상이 된 문제들 중 대부분에서는 수치오차 측정방법에 의한 오차의 증감을 관찰할 수 없었다. 다만, 크고 수치적으로 불안정한 문제인 pilotnov에서는 최소 절대값에 의한 방법이 오차와 수행 시간을 단축시키는 것으로 관찰되어, 더 크고 불안정한 문제들에서 좋은 성능을 보일 것으로 예상된다.

3. 행렬희소도 유지

3.1 Markowitz 순서화의 구현

기저행렬의 상하분해시에는 순서화가 중요한 문제가 된다. 순서화 방법에 따라 분해된 상하삼각행렬의 희소도가 크게 달라지게 되어 계산 시간의 많은 차이를 가져오게 된다. 상하분해시의 순서화로는 Markowitz 순서화가 많이 이용된다. Markowitz 순서화는 선회연산시의 곱셈·덧셈연산의 횟수와, 선회연산으로 발생할 것으로 예상되는 Fill-in을 적게 하는 요소를 선회요소로 선택하려는 의도를 가지고 있다. 선회연산의 매 Stage마다 나타나는 부분행렬(Active Submatrix)에서

r_i : i 행의 비영요소의 갯수,

c_j : j 열의 비영요소의 갯수

라 할 때, $(r_i - 1)(c_j - 1)$ 값을 Markowitz Count라하며, 이 값을 최소로 하는 비영요소를 선회요소로 선택한다.

Markowitz 순서화는 희소한 상하삼각행렬을 얻는 좋은 Local pivoting strategy이지만,[8][9] 실제로 어떻게 구현하느냐에 따라 순서화에 걸리는 시간에는 큰 차이가 나게 된다.[9]

① Naive한 구현

가장 쉽게 생각해 볼 수 있는 Naive한 구현으로, 부분행렬의 모든 비영요소에 대해 Markowitz Count를 계산하여 이들 각각을 비교·검색하여 최소값을 찾도록 하는 방법을 생각해 볼 수 있다. 이 때, 각 비영요소의 Markowitz Count를 바로 계산할 수 있도록 하기 위해, 부분행렬의 각 행·

열의 비영요소 갯수를 [그림 1]과 같이 일차원배열에 미리 저장해 두었다가 선회연산 과정에서 이를 수정, 유지한다.

	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)

열번호	열의 비영갯수
1	4
2	3
3	1
4	2
5	3
6	2

행번호	행의 비영갯수
1	1
2	2
3	3
4	4
5	5
6	6

〔그림 1〕 Naive한 방법에서 사용된 일차원배열

Naive한 구현은 개념과 구현이 간단하지만 검색에 많은 시간이 소요되는 단점이 있다.[9] 즉, 부분행렬내의 비영요소의 갯수를 τ라고 할 때, Naive한 구현에서는 $O(\tau)$번의 많은 검색이 있게 된다.

② 하한설정 방법

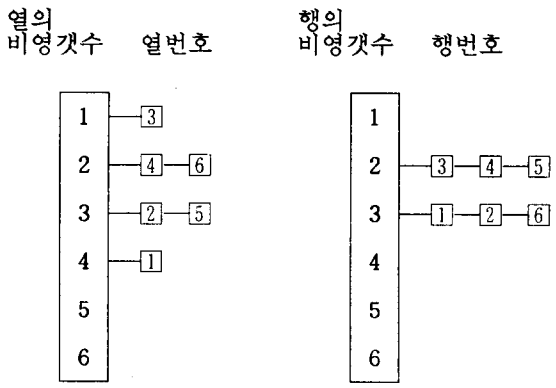
검색을 줄여서 순서화 소요시간을 줄이려는 목적의 구현이다.

Naive한 구현에서 부분행렬 각 행·열의 비영요소 갯수를 저장·수정하기 위해 일차원배열을 사용했던 것이 비해, 하한설정 방법에서는 양방향 연결 리스트 자료구조를 이용한다.

[그림 2]에 하한설정 방법에서 사용된 자료구조를 예를 들어 나타내었다.

	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)



〔그림 2〕 하한설정 방법에서 사용된 자료구조

[그림 2]의 연결 리스트상에는 부분행렬의 각 행·열의 비영요소 갯수에 의해 정렬된 순서가 나타나게 된다. 선회연산이 일어날 때마다 이를 수정함으로서 항상 정렬된 순서를 유지할 수 있다. 처음에 연결리스트를 만드는 데 $O(n)$ (n은 Matrix의 크기)의 시간이 소요되고, 양방향 Linked list이므로 이의 수정·유지에는 상수 시간이면 충분하다.[9]

실제 검색은 행·열의 비영요소 갯수에 의한 정렬 순서를 이용하여, 비영요소가 적은 열과 행으로부터 검색을 시작한다. 이 때, 현재까지 발견된 요소 중 최소의 Markowitz Count를 갖는 비영요소를 기억해 두는데, 이를 선회연산 후보자(potential pivot element)라고 한다.

검색과정에서는 아직 검색하지 않은 비영요소들의 Markowitz Count의 하한값을 예측하여, 현재의 선회연산 후보자의 Markowitz Count가 이 하한값 이하가 되는 순간 검색이 끝내게 된다.[9]

평균적으로는 수개의 행·열만 조사하면 검색이 끝난다고 한다.[9] 그러나, 예제에서 미루어 알

수 있듯이, 최악의 경우에는 Naive한 구현과 마찬가지로 $O(\tau)$의 긴 검색이 발생할 수 있다.

③ 제한된 하한설정 방법

하한설정 방법은 최악의 경우에 $O(\tau)$의 긴 검색이 발생하는 단점이 있다. 이러한 경우를 방지하기 위해, 열(또는 행)단위로만 검색하되 최대 k개의 열(또는 행)에 대해서만 검색을 제한하는 방법이 제한된 하한설정 방법이다.[9][19]

이상의 방법에 의해 Markowitz 순서화에 걸리는 시간을 실험 비교한 결과가 [표 3]에 나타나 있다. [표 3]의 실험은 Markowitz 순서화의 효과만을 분명히 하기 위하여 필요보다 약간 많은 재역산을 실시하도록 하였다.

실험기종 : SUN SPARC 10

방법 / 문제명	Naive		하한설정		제한된 하한설정	
	[1]	[2]	[1]	[2]	[1]	[2]
scagr7	1.6	263	1.4	265	1.4	314
scsd6	16.2	498	13.5	506	17.9	697
israel	5.2	991	4.0	930	5.2	1102
e226	8.0	864	7.9	870	7.5	1032
boeing1	17.6	1183	14.1	1157	14.8	1331
stair	61.3	3673	35.8	3582	40.4	4235
pilot4	239.1	3643	91.3	3380	109.8	4219
agg3	6.9	1466	6.8	1507	6.5	1472
ship12s	16.4	975	17.5	976	17.4	1029
sierra	59.1	2258	45.4	2102	49.4	2454

[1] 수행시간(초) [2] 상하분해후 평균 비영요소 갯수

〔표 3〕 Markowitz순서화의 검색방법 비교

[표 3]에서, 상하분해 후의 비영요소의 갯수는 Naive한 방법과 하한설정 방법이 비슷한 반면 제한된 하한설정 방법에서는 앞의 두 방법에 비해 약 16% 증가된 것을 확인할 수 있다. 또, 수행시간의 경우 Naive한 방법과 제한된 하한설정 방법이 하한설정 방법에 비해 각각 29%, 9%정도 증가되는 것으로 나타났고, 비교적 밀집도가 높은 문제인 stair, pilot4문제에서 더 큰 차이를 보였다. 제한된 하한설정 방법이 적은 검색량에도 불구하고 오히려 하한설정 방법보다 수행시간이 증가하는 것은 순서화 자체에서 감소된 시간보다 비영요소의 증가로 인한 영향이 더 크기 때문인 것으로 해석된다.

3.2 기저 수정 방법의 선택

상하분해시와 마찬가지로, 기저수정에도 순서화 방법에 따라 행렬의 희소도와 계산 시간에 차이가 발생하며, 수치 안정성에 대한 고려도 필요하게 된다. 또, 단체법의 매 회에 일어나는 작업이므로, 수행 시간을 빠르게 하는 것이 상하분해시보다 더욱 중요해진다. 기저수정방법에 대해서도 많은 연구가 있었으나,[4][10][11][15][16][18] 수치 안정성이 우수한 Bartels-Golub방법[4]과 이의 변형들이 실용적으로 많이 사용되고 있다.[18] 특히, Bartels-Golub방법의 단점인 희소성 문제를 보완한 Reid가 제안한 방법[15][16]이 효율적인 것으로 알려져 있다.[14][18]

① Bartels-Golub의 기저수정방법

기저의 한 열이 교체되었을 때 상삼각행렬에는 Spike가 발생한다. Bartels-Golub방법에서는 먼저 열 치환을 통해 Spike열을 상삼각행렬의 맨 마지막 열에 위치시킨다.

다음으로 대각요소 아래에 있는 비영요소를 0으로 만들어 주는 선회연산을 수행한다. 이 때 선회연산을 행하기 전 수치안정성을 고려하여 대각요소가 그 아래에 있는 원소보다 더 큰 값이 되도록 행을 치환한다.[4]

② Reid의 기저수정방법

Reid의 기저수정방법은 행·열 치환을 통하여 상삼각행렬에 들어온 Spike의 크기를 줄인 후 선회연산을 행한다. 이렇게 함으로서 희소도를 유지시키고, 선회연산의 횟수가 줄어들게 되므로 수치오차의 누적을 막을 수 있는 장점이 있다.

Spike의 크기를 줄이기 위해서는 Spike로 인해 형성된 Bump내의 행·열 Singleton들을 각각 Bump의 오른쪽 아래·왼쪽 위로 대칭치환(Symmetric Permutation)시키는 방법을 사용한다. 대각이 아닌 곳에 위치한 Singleton에 대해서

는 비대칭치환을 행하여야 한다.[15]

두 방법을 비교 실험하여 [표 4]에 결과를 나타내었다.

실험기종 : SUN SPARC 10

방법 / 문제명	Bartels-Golub		Reid	
	[1]	[2]	[1]	[2]
scagr7	2.0	4.61	1.2	1.08
scsd6	32.2	4.28	16.8	1.23
israel	8.4	1.55	5.1	1.10
e226	13.8	1.73	7.9	0.78
boeing1	43.3	1.43	15.3	0.38
stair	59.3	2.15	35.9	0.67
pilot4	154.8	1.85	94.0	0.54
agg3	12.0	4.30	5.8	0.77
ship12s	63.9	3.10	14.0	0.18
sierra	224.1	3.54	38.4	0.14

[1] 수행시간(초) [2] 기저수정시 평균 비영요소 증가율(%)

〔표 4〕 기저수정방법의 비교

[표 4]에서, Reid방법을 이용했을 때의 평균 비영요소 증가율이 Bartels-Golub방법에 비해 약 78%정도 낮아지는 것을 관찰할 수 있다. 아울러 수행 시간도 약 56%정도 감소하는 것으로 나타나, Reid방법이 Bartels-Golub방법에 비해 매우 효율적인 것을 알 수 있다.

3.3 행렬희소도 유지를 위한 재역산

기저수정이 반복됨에 따라, 점차로 상하삼각행렬의 밀집도(Density)가 높아지게 된다. 이러한 상태에서는 수행 속도가 크게 저하되므로, 적절한 시간에 재역산을 실시하여 상하삼각행렬을 다시 희소하게 만들어 주는 것이 필요하다. 이 때 재역산 시기를 결정하는 것이 문제가 되는데, 본 연구에서는 다음과 같은 3가지 방법을 고려하였다.

① 고정 횟수 기준

기저수정의 횟수가 정해진 횟수에 이르면 재역산을 하는 방법이다.

② 변동 희소도 기준

재역산 직후의 밀집도를 측정, 이를 기준으로 하여 일정 배 이상이면 재역산을 행한다

③ 입력자료 희소도 기준

입력자료행렬(A)의 밀집도를 측정, 이를 기준으로 상하삼각행렬의 밀집도가 일정 배 이상이면 재역산을 행하는 방법이다.

[표 5]에 이들 3가지 방법에 대한 실험 결과가 나타나 있다.

실험기종 : SUN SPARC 10 표 안은 수행시간(초)

방법 / 문제명	고정횟수 기준	변동희소도 기준	입력자료 희소도기준
scagr7	1.3	1.2	1.2
scsd6	15.4	13.3	16.8
israel	4.2	4.2	5.1
e226	9.1	7.3	7.9
boeing1	14.7	17.3	15.3
stair	32.4	35.7	35.9
pilot4	107.6	100.5	94.0
agg3	6.7	5.9	5.8
ship12s	16.6	14.3	14.0
sierra	40.6	38.3	38.4

〔표 5〕 기저수정방법의 비교

[표 5]에서, 3가지 방법 사이의 유의한 차이는 찾아볼 수 없으므로, 어느 방법이 우수하다고 단정짓기는 어려우나, 큰 문제일수록 입력자료 희소도 기준이 수행시간을 단축시키는 경향이 있음을 관찰할 수 있다.

4. 결론

이상에서, 선형계획법 프로그램의 수치오차를 보정하는 방법과 행렬희소도를 유지하여 수행 속도를 개선시키는 방안에 대해 언급하였다.

수치오차를 보정하는 방법에서는 규모화와 수치오차 측정 방법이 고려되었으며, 규모화는 기하평균 후 최대값 규모화를 하는 방법이 가장 수치안정성을 높이는 것으로 나타났다. 수치오차를 측정하는 방법들 사이에서는 본 연구에서 행해진 실험에서는 유의한 차이를 찾아볼 수 없었으나,

더 크고 수치적으로 불안정한 문제들에 대한 실험이 요구된다.

한편, 행렬희소도를 유지하기 위해 상하분해시의 순서화, 기저수정방법, 행렬의 희소도에 대한 재역산 방법이 고려되었다. 상하분해시 수행되는 Markowitz 순서화의 수행 시간을 줄이기 위해 고려된 여러 검색 방법들 중 하한설정 방법이 가장 효율적인 것으로 나타났다. 또, 기저수정방법으로는 Reid의 방법이 Bartels-Golub방법에 비해 수행시간을 크게 단축시키는 결과를 보여주었다. 행렬의 희소도에 의한 재역산 기준들 사이에서는 유의한 차이는 찾아볼 수 없었으나, 큰 문제일수록 입력자료희소도기준이 수행시간을 단축시키는 경향이 있음을 관찰하였다.

추후에 수치오차를 빠르고 정확하게 찾아낼 수 있는 방법과 행렬희소도에 대한 재역산 기준, Markowitz 순서화의 보다 효율적인 검색방법과 자료구조, 기저수정방법의 개선 등에 대한 보다 많은 연구가 필요하다.

참고문헌

[1] 박순달, 「선형계획법(3정판)」, 민영사, 1992

[2] 김우제, 강완모, 김민정, 박순달, "일반한계 선형계획법에서 비영요소만 보관하는 자료구조와 평가전략의 효율성에 관한 연구", 전산활용연구, 제6권 제1호, pp 55-66, 1993

[3] 진희채, "선형계획법을 위한 아핀법과 장벽법의 통합화 및 알고리듬의 효율화", 공학박사학위 논문, 서울대학교, 1995

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

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

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

[7] Dembart, B., Erisman, A. M, "Hybrid Sparse-Matrix Methods", *IEEE Transactions on Curcuit theory*, Vol. CT-20, No. 6, pp. 641-649, 1973

[8] Duff, I. S., "A Survey of Sparse Matrix Research", *Proceedings of the IEEE*, Vol. 65, No. 4, pp 500-535, 1977

[9] Duff. I. S., A. M. Erisman, J. K. Reid, *Direct Methods for Sparse Matrices,* Clarendon Press, Oxford 1986

[10] Forrest, J. J. H., J.A. Tomlin, "Implementing the Simplex Method for the Optimization Subroutine Library", *IBM Systems Journal*, Vol. 31, No. 1, pp 11-25, 1992

[11] Forrest. J. J. H., J. A. Tomlin, "Updating triangular factors of the basis to maintain sparsity in the product-form simplex method." *Math. Prog.* 2, pp 263-278, 1972

[12] Murtagh, B. A., *Advanced Linear Programming : Computation and Practice*, McGraw-Hill, 1981

[13] Murty, K. G., *Linear Programming*, Wiley, 1983

[14] Perold, A. F., "A degeneracy exploiting LU factorization for the Simplex method", *Math. Prog. 19*, pp.239-254, 1980

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

[16] 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 pp 1-23, 1975

[17] Sedgewick, R., *Algorithms*, 2nd ed., Addison Wesley, New york, 1988

[18] Saunders, M. A., "A fast, stable implementation of the Simplex method using Bartels-Golub updating", 1975

[19] Zlatev, Z., "On some pivotal strategies in Gaussian elimination by sparse technique", *SIAM J. Numer. Anal. 17*, pp. 18-30