經營科學
第17卷 第2號
2000年 11月

내부점 선형계획법에서의 최적기저 추출방법의 구현*

임성묵** · 박순달**

On the Implementation of an Optimal Basis Identification Procedure for Interior Point Method*

Sung-Mook Lim** · Soon-Dal Park**

Abstract

In this study, we deals with the implementation of an optimal basis identification procedure for interior point methods. Our implementation is based on Megiddo's strongly polynomial algorithm applied to Andersen and Ye's approximate LP construction. Several techniques are explained such as the use of effective indicator for obtaining optimal partition when constructing the approximate LP, the efficient implementation of the problem reduction technique proposed by Andersen, the crashing procedure needed for fast dual phase of Megiddo's algorithm, and the construction of the stable initial basis. By experimental comparison, we show that our implementation is superior to the crossover scheme implementation.

1. 서 론

선형계획법(Linear Programming)은 그 넓은 활용분야로 인해 이론적 연구 및 해법 개발이 상당히 많이 이루어져 있는 경영과학의 한 분야이다[1]. 표준적인 원, 쌍대 선형계획법 문제는 다음과 같은 형태를 갖는다.

$$\text{(P)}: \begin{array}{ll} min & \boldsymbol{c}^T\boldsymbol{x} \\ s.t. & A\boldsymbol{x}=\boldsymbol{b} \\ & \boldsymbol{x}\geq 0 \end{array} \qquad \text{(D)}: \begin{array}{ll} max & \boldsymbol{b}^T\boldsymbol{y} \\ s.t. & A^T\boldsymbol{y}+\boldsymbol{s}=\boldsymbol{c} \\ & \boldsymbol{s}\geq 0 \end{array}$$

위 문제 (P)에서 A는 $m\times n$ 행렬이고, $\boldsymbol{b}$는 m 차원 벡터, $\boldsymbol{c}$는 n차원 벡터이다.

선형계획법 문제의 해법은 지금까지 다양하게

* 본 연구는 한국과학재단의 특정기초연구과제(과제번호 98-0200-07-01-2)의 지원을 받았음
** 서울대학교 산업공학과

개발되어 왔는데 1940년대에 G.B. Dantzig에 의해 개발된 단체법(Simplex Method)이 그 시초라고 할 수 있다. 그 후 많은 연구자들에 의해 단체법의 실용적인 구현과 개선이 이루어져서, 단체법은 선형계획법을 푸는 실용적인 해법으로서 그 자리를 굳히게 되었다[5]. 그러나 1984년 N. Karmarkar에 의해 내부점 방법(Interior-point Method)이 개발되면서 이론적인 면이나 실용적인 면에서 단체법을 대체할 수 있는 해법으로 발전하였다.

단체법과 내부점 방법 사이의 큰 차이점은, 단체법이 이웃 기저들을 나열해 가면서 최적 기저해를 찾는 반면 내부점 방법은 해공간의 내부를 이동하여 최적해를 찾는 방법이라는 점이다. 따라서 단체법은 언제나 최적 기저해(Optimal basic solution)를 출력한다. 그러나, 다수의 최적해가 존재하는 경우 내부점 방법은 최적면의 내부해를 출력한다. 많은 응용에서는 기저해를 필요로 하는데, 예를 들면 기본감도분석을 수행하거나 warm-starting을 효과적으로 수행하고자 할 때에는 기저해가 필수적이라고 할 수 있다. 그러나 내부해가 더 적절한 경우도 있을 수 있다[8].

Megiddo[10]는 원, 쌍대 최적해가 주어졌을 때 최적기저해를 추출하는 강성다항식 알고리즘을 개발하였는데, Bixby는 Megiddo의 방법을 구현하여 우수한 실험결과를 보였다. Bixby[6]는 또한 내부점 방법의 최종해로부터 얻은 정보를 이용하여 기저를 구성한 다음 단체법을 적용하여 최적 기저해를 찾는 cross-over 방법을 구현하여 우수한 실험결과를 보이기도 하였다. Megiddo의 알고리즘은 원, 쌍대 최적해를 필요로 하는데 일반적으로 내부점 방법은 최적해에 근접해 가는 일련의 해들을 출력하기 때문에 정확한 최적해를 얻어내기가 어렵다. 이런 이유로 Megiddo의 알고리즘은 cross-over 방법에 비해 실용성이 떨어진다고 알려져 왔다. 그러나 내부점 방법을 유한 번만에 끝내고 정확한 최적해를 구할 수 있는 방법이 Ye[12]에 의해 개발되고, 이를 Megiddo의 알고리즘과 결합하여 최적기저해를 추출하는 방법이 Andersen과 Ye[4]에 의해 연구되었다.

Andersen과 Ye[4]의 방법의 기본 아이디어는 내부점 방법의 최종해로부터 근사 선형계획법 문제를 구성하여, 그 최종해가 최적 강상보해(Optimal strictly complementary solution)이 되도록 하는 것이다. 최적 강상보해는 Megiddo의 알고리즘을 적용할 수 있는 필요조건이 된다. 또한 그들은 근사 선형계획법 문제의 최적기저가 원래 문제의 최적기저가 되도록 하는 내부점 방법의 반복회수의 하한도 제시하였다.

Andersen과 Ye의 방법은 대략 다음과 같은 절차를 거치면서 최적기저해를 추출한다.

단계 1 : 원쌍대 내부점 방법을 주어진 반복회수만큼 수행한다.

단계 2 : 단계 1을 통해 출력된 최종해를 이용하여 근사 선형계획법 문제를 구성한다.

단계 3 : 근사 선형계획법 문제에 Megiddo의 알고리즘을 적용하여 최적 기저해를 추출한다.

위의 단계 1에서는 내부점 방법의 종료조건이 중요하다. 물론 이론적으로 알려진 반복회수의 하한이 있지만 실용적으로 볼 때 적절하지 못하다. 따라서 일반적인 구현에서는 원가능성, 쌍대가능성, 쌍대간격 등의 일정 기준이 만족되면 종료하도록 한다. 이러한 조기종료는 근사 선형계획법 문제의 최적기저가 원래 문제의 최적기저가 되는지에 영향을 미치게 된다. 즉, 조기종료로 인해 단계 3을 통해 구한 최적기저가 원래 문제의 최적기저가 되지 못할 수도 있다. 이러한 경우에는 단체법의 부가적인 수행으로 원래 문제의 최적기저를 구하는 과정이 필요하게 된다.

단계 2에서는 근사 선형계획법 문제를 구성할 때 내부점 방법의 최종해를 어떻게 이용할 것인지가 중요한 문제가 된다. 근사 선형계획법 문제를 구성할 때에는 내부점 방법의 최종해를 이용하여 최적분할(Optimal partition)을 예측하여야 하는데, 이를 위해 어떤 지시자(indicator)를 사용해야는지

가 중요한 문제가 된다.

단계 3에서는 Megiddo의 알고리즘의 효율적인 구현이 중요한 문제가 된다. Andersen[3]은 Megiddo의 알고리즘을 구현할 때 사용될 수 있는 문제 축소기법들을 여러 가지 제시하였는데 이의 효과적 구현은 해법의 성능을 크게 좌우한다.

본 논문의 각 절에서는 최적기저 추출방법에서 위에서 언급된 여러 가지 문제점들에 대해 분석하고, 각각의 문제에 대해 효과적인 구현을 위한 방법을 살펴본다. 우선 2절에서는 Megiddo의 알고리즘을 살펴보고, 3절에서는 근사 선형계획법 문제를 구성하기 위한 효과적인 지시자의 사용에 대해 알아보고, 4절에서는 Andersen이 제안한 문제 축소기법의 효율적인 구현방법에 관해 살펴본다. 그리고 5절에서는 Megiddo의 알고리즘의 쌍대국면(Dual phase)의 빠른 수행을 위해 필수적인 crashing 방법에 대해 알아보고, 6절에서는 Megiddo의 알고리즘을 시작하기 위한 초기기저의 구성방법에 대해 다루고, 7절에서는 대형 선형계획법 문제에 대한 실험결과를 분석한다.

2. Megiddo의 강성다항식 알고리즘

Megiddo는 원, 쌍대 최적해가 주어졌을 때 최적기저를 추출하는 알고리즘을 제시하고 강성다항시간(Strongly polynomial-time) 내에 그 알고리즘이 수행될 수 있음을 보였다. 또한 선형계획법을 푸는 강성다항식 알고리즘이 존재하지 않는다면, 원 최적해나 쌍대 최적해만 주어진 경우에는 최적기저를 추출하는 강성다항식 알고리즘이 존재할 수 없다는 것을 보였다.

Megiddo의 방법은 원최적기저를 구하는 원국면(Primal phase)과 쌍대최적기저를 구하는 쌍대국면(Dual phase)으로 구성된다. 주어진 선형계획법 문제 (P), (D)의 원, 쌍대 강상보최적해 $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{s}^*)$가 주어졌다고 하고, 이에 따른 최적분할을 $(P, \overline{P})$라고 하자. 즉, $P=\{i: x_i^*>0\}$이고 $\overline{P}=\{j: s_j^*>0\}$이다. 그러면 $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{s}^*)$은 다음의 최적조건을 만족한다.

$$A\boldsymbol{x}^* = \boldsymbol{b}, \quad \boldsymbol{x}^* \geq 0$$
$$A^T\boldsymbol{y}^* + \boldsymbol{s}^* = \boldsymbol{c}, \quad \boldsymbol{s}^* \geq 0$$
$$x_i^* s_i^* = 0, \quad x_i^* + s_i^* > 0, i = 1,2,\cdots,n$$

원국면에서는 현재의 원최적해에서 초기저변수(Superbasic variable)를 제거하여 원최적기저를 구한다. 초기저변수란 비기저변수 중 한계값(Bound value)를 가지지 않는 변수를 일컫는다. 우선, 임의의 기저, 비기저 분할 (B, N)이 주어졌다고 하자. 그러면 다음이 성립한다.

$$B\boldsymbol{x}_B^* + N\boldsymbol{x}_N^* = \boldsymbol{b}$$
$$\boldsymbol{x}_B^* = B^{-1}(\boldsymbol{b} - N\boldsymbol{x}_N^*) \geq 0$$

Megiddo의 알고리즘의 원국면에서는 비기저 변수 중 값이 영이 아닌 초기저 변수를 기저변수로 만들거나 값을 영으로 만드는 과정을 수행한다. $\boldsymbol{x}^0 = \boldsymbol{x}^*$라고 두고 $x_j^0 > 0, j \in N$ 인 j에 대해, x_j^0의 값을 영으로 만들기 위해 다음의 계산을 수행한다.

$$\boldsymbol{x}_N' = \boldsymbol{x}_N^0 - \alpha \boldsymbol{e}_j$$
$$\boldsymbol{x}_B' = \boldsymbol{x}_B^0 + \alpha B^{-1} N \boldsymbol{e}_j$$

해의 개선 폭인 α가 증가하면 x_j^0의 값이 감소하고, $B^{-1}N\boldsymbol{e}_j$의 부호에 따라 기저변수들 중에서도 그 값이 감소하는 변수가 존재한다. 만일 기저변수들의 값이 영으로 되기 전에 초기저변수 x_j^0의 값이 영이 되면 기저의 변화없이 초기저변수의 개수를 하나 줄이게 된다. 그러나 초기저변수 x_j^0의 값이 영이 되기 전에 기저변수 중 하나가 영이 되면 그 기저변수를 기저에서 탈락시키고 x_j^0를 기저에 포함시킨다. 이렇게 원국면의 한 단계를 수행하고 나면 초기저변수의 개수가 하나 줄어들게 된다. 초기저변수의 개수 $|N \cap P|$는 최대 $n-m$

이므로 원국면의 최대 수행회수는 $n-m$이 된다. 또한, 원국면에서 수행되는 각 단계는 쌍대퇴화이므로, 최종적으로 구해지는 기저는 원최적기저가 된다.

원국면을 자세히 적어보면 다음과 같다.

단계 0 : 임의의 초기기저 (B, N), 최적 강상보해 $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{s}^*)$, $\boldsymbol{x}=\boldsymbol{x}^*$

단계 1 : 임의의 $i\in\{k: x_k>0,\ k\in N\}$에 대해,

$$\alpha^* = max\{\alpha: \boldsymbol{x}_N-\alpha\boldsymbol{e}_i\geq 0,\ \boldsymbol{x}_B+\alpha B^{-1}N\boldsymbol{e}_i\geq 0\}$$

$$j^* = argmax\{\alpha: (\boldsymbol{x}_N-\alpha\boldsymbol{e}_i)_j\geq 0,\ (\boldsymbol{x}_B+\alpha B^{-1}N\boldsymbol{e}_i)_j\geq 0\}$$

단계 2 : $\boldsymbol{x}_N := \boldsymbol{x}_N-\alpha\boldsymbol{e}_i$, $\boldsymbol{x}_B := \boldsymbol{x}_B+\alpha B^{-1}N\boldsymbol{e}_i$ 로 수정하고,
만일 $j^*=B$이면,
기저를 $B=B\cup\{i\}\setminus\{j^*\}$로 수정한다.

쌍대국면에서는 현재의 쌍대최적해에서 초기저 쌍대여유변수(Superbasic dual variable)를 제거하여 쌍대최적기저해를 구한다. 기저변수 x_j에 대응되는 쌍대여유변수 s_j의 값이 영이 아닐 때, s_j를 초기저 쌍대여유변수라고 한다. 초기저 쌍대여유변수가 없을 때 주어진 쌍대최적해는 쌍대최적기저해가 된다. 임의의 기저, 비기저 분할 (B, N)이 주어졌다고 하자. 그러면 다음이 성립한다.

$$B^T\boldsymbol{y}+\boldsymbol{s}_B=\boldsymbol{c}_B$$
$$N^T\boldsymbol{y}+\boldsymbol{s}_N=\boldsymbol{c}_N$$

하나의 초기저 쌍대여유변수 $\boldsymbol{s}_{B_i}{}^*>0$를 선택하여, 이 값을 영으로 만들기 위해 다음의 계산을 수행한다. 단, $\boldsymbol{y}^0=\boldsymbol{y}^*$, $\boldsymbol{s}^0=\boldsymbol{s}^*$이라고 둔다.

$$\boldsymbol{y}' = \boldsymbol{y}^0+\alpha B^{-T}\boldsymbol{e}_i$$
$$\boldsymbol{s}_B' = \boldsymbol{s}_B^0-\alpha\boldsymbol{e}_i$$
$$\boldsymbol{s}_N' = \boldsymbol{s}_N^0+\alpha N^TB^{-T}\boldsymbol{e}_i$$

쌍대해의 개선 폭인 α가 증가하면 $s_{B_i}{}^0$의 값이 감소하고, $N^TB^{-T}\boldsymbol{e}_i$의 부호에 따라 $\boldsymbol{s}_N$의 변수들 중에서도 그 값이 감소하는 변수가 존재한다. 만일 $\boldsymbol{s}_N$의 값들이 영으로 되기 전에 초기저 쌍대여유변수 $s_{B_i}{}^0$의 값이 영이 되면 기저의 변화없이 초기저 쌍대여유변수의 개수가 하나 줄어들게 된다. 그러나 초기저 쌍대여유변수 $s_{B_i}{}^0$의 값이 영이 되기 전에 $\boldsymbol{s}_N$중 하나가 영이 되면 그 여유변수를 기저에 포함시키고 $s_{B_i}{}^0$를 기저에서 탈락시킨다. 이렇게 쌍대국면의 한 단계를 수행하고 나면 초기저 쌍대여유변수의 개수가 하나 줄어들게 된다. 초기저 쌍대여유변수의 개수 $|B\cap\overline{P}|$는 최대 m이므로 쌍대국면의 최대 수행회수는 m이 된다. 또한, 쌍대국면에서 수행되는 각 단계는 원퇴화이므로, 최종적으로 구해지는 기저는 쌍대최적기저가 된다. 그리고 초기기저가 원최적기저라면 쌍대국면을 통해 최적기저가 된다.

쌍대국면은 구체적으로 다음과 같은 단계를 따른다.

단계 0 : 임의의 초기기저 (B, N), 최적 강상보해 $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{s}^*)$, $\boldsymbol{y}=\boldsymbol{y}^*$, $\boldsymbol{s}=\boldsymbol{s}^*$

단계 1 : 임의의 $j\in\{k: s_{B_k}>0\}$에 대해,

$$\alpha^* = max\{\alpha: \boldsymbol{s}_B-\alpha\boldsymbol{e}_j\geq 0,\ \boldsymbol{s}_N+\alpha B^{-1}N\boldsymbol{e}_j\geq 0\}$$

$$i^* = argmax\{\alpha: [(\boldsymbol{s}_B,\ \boldsymbol{s}_N)^T-\alpha(0,\ N^TB^{-T})^T\boldsymbol{e}_j]_i\geq 0\}$$

단계 2 : 쌍대해를 $\boldsymbol{y} := \boldsymbol{y}+\alpha B^{-T}\boldsymbol{e}_j$, $\boldsymbol{s}_B := \boldsymbol{s}_B-\alpha\boldsymbol{e}_j$, $\boldsymbol{s}_N := \boldsymbol{s}_N+\alpha N^TB^{-T}\boldsymbol{e}_j$ 으로 수정한다.
만일 $i^*=N$ 이면, 기저를
$B=B\cup\{i^*\}\setminus\{B_j\}$로 수정한다.

Megiddo의 알고리즘의 특성은 전체 과정이 원, 쌍대 단체법과 유사하지만 할인연산이 없다는 점이다. 즉, 초기저 변수의 진입순서와 초기저 쌍대여유변수의 탈락순서는 임의로 결정가능하다는 것

이다. 4절에서 살펴보겠지만 이러한 특징으로 인해 문제의 축소기법을 사용할 수 있게 된다.

3. 근사 선형계획법 문제

내부점 방법으로 선형계획법 문제를 풀 때에는 최적면의 내부로 수렴하는 일련의 내부해가 구해진다. 이 내부해를 $(\hat{x}, \hat{y}, \hat{s}) \in R^n_{++} \times R^m \times R^n_{++}$ 라고 하고, 이 내부해로부터 최적분할을 예측하여 $(P, \overline{P})$를 얻었다고 하자. 그러면 우변상수와 목적함수 계수를 다음과 같이 수정하여,

$$\overline{b} = A_P \hat{x}_P, \quad \overline{c} = A^T \hat{y} + (0, \hat{s}_{\overline{P}})$$

다음과 같은 근사적 선형계획법 문제(Approximate LP)를 생성해 보자.

$$\text{(AP)} \quad \begin{array}{lll} min & \overline{c}^T x & \\ s.t. & Ax = & \overline{b} \\ & x \geq 0 & \end{array}$$

그러면 $(x, y, s) = ((\hat{x}_P, 0)^T, \hat{y}, (0, \hat{s}_{\overline{P}})^T)$는 (AP)의 강상보해가 되고, $(P, \overline{P})$는 (AP)의 최적분할이 된다는 것을 쉽게 알 수 있다. 만일 $(\hat{x}, \hat{y}, \hat{s})$이 최적해에 근접한 해라면 $(P, \overline{P})$이 원래 문제의 최적분할에 가까울 것이고, (AP)의 최적기저는 (P)에서도 최적기저가 될 것이다. Andersen과 Ye는 (AP)의 최적기저가 (P)의 최적기저가 되도록 하는 내부점 방법의 최종해 $(\hat{x}, \hat{y}, \hat{s})$를 얻기 위해 필요한 내부점 방법의 최소 반복회수를 제시하였다.

내부점 방법의 최종해를 이용하여 근사적 선형계획법 문제(AP)를 구성하면, 그 문제에 대한 강상보 최적해를 알 수 있기 때문에 Megiddo의 알고리즘을 적용하여 (AP)의 최적기저를 추출할 수 있다. 그런데, 내부점 방법의 최종해를 이용하여 (AP)를 구성할 때, 발생할 수 있는 두 가지 문제점이 있다. 첫째는 실용적으로 가치 있는 내부점 종료시점이고, 둘째는 내부점 최종해로부터 어떻게 최적분할을 예측할 것인가 하는 것이다.

본 연구에서는 내부점의 종료조건으로 상대적 쌍대간격(Relative duality gap)을 사용하였다. 상대적 쌍대간격 d_g은 다음과 같이 계산되는데,

$$d_g = \frac{| c^T x^k - b^T y^k - u^T w^k |}{1 + | b^T y^k + u^T w^k |}$$

d_g가 10^{-8}보다 작아지면 내부점 방법을 종료하도록 하였다. 원가능성과 쌍대가능성을 고려하지 않은 이유는 원가능성과 쌍대가능성의 수렴속도가 쌍대간격의 수렴속도보다 빠르기 때문인데, 이러한 사실은 이론적으로도 검증이 가능하고 실험적으로도 확인할 수 있다. 드문 경우로 쌍대간격은 많이 줄었지만 원비가능성이나 쌍대비가능성이 비교적 큰 경우가 발생할 수 있는데, 이런 경우에는 부정확한 최적분할 예측이 일어나고, 결과적으로 (AP)의 최적기저가 (P)의 최적기저가 되지 못하게 된다. 이러한 경우를 방지하기 위해 본 연구에서는 원비가능성이나 쌍대비가능성 중 하나가 10^{-9}이하가 되어야 한다는 조건을 종료조건으로 첨가하였다. 쌍대간격의 수렴속도에 비해 원비가능성이나 쌍대비가능성의 수렴속도가 비교적 느릴 때에는 그 원인이 주어진 문제의 수치적 불안정성에 있는 경우가 많은데, 이러한 문제를 내부점 방법으로 풀 때 원가능성이나 쌍대가능성을 강조하다 보면 해법이 수렴하지 못하거나 해법의 반복회수가 크게 늘어날 수 있다. 따라서, 조기종료하여 최적기저 추출방법을 수행하는 것이 더욱 안정적이다. 그리고 본 연구의 실험결과에 의하면 원가능성이나 쌍대가능성의 수렴속도가 동시에 느린 경우는 거의 없었고, 원가능성이나 쌍대가능성 중 어느 하나만 주어진 기준을 만족하면 (AP)의 최적기저가 (P)의 최적기저로 되지 못하는 경우가 발생하지 않았다.

이제 최적분할의 예측방법에 대해 살펴보자. Andersen과 Ye의 연구에서는 다음의 지시자(In-

dicator)를 사용하여 최적분할을 예측하였다.

$$P = \{ j : |\Delta x_j| / x_j \le |\Delta s_j| / s_j \},$$
$$\overline{P} = \{1, 2, \cdots, n\} \setminus P$$

위의 지시자는 Mehrotra와 Ye가 제시한 방법과 동일한 방법으로 문제의 규모화에 영향을 받지 않는다는 장점이 있다. 위의 방법과는 달리 원 변수 $\boldsymbol{x}$의 값만을 보고 최적분할을 판별하는 방법도 있다. 이 방법을 변수지시자(Variable indicator)라고 하는데, 다음의 방법을 따른다.

$$P = \{ j : x_j > \varepsilon \}, \quad \overline{P} = \{1, 2, \cdots, n\} \setminus P$$

즉, 어느 기준값을 넘는 변수는 최적해에서 양의 값을 가진다고 보고 P에 속한다고 판별하는 방법이다. 본 연구에서는 앞에서 설명한 두 가지 방법과 함께, 두 가지 방법을 혼합한 방법을 서로 비교 실험해 보았다. 두 가지 방법을 혼합한 방법은 다음과 같은 기준을 사용한다.

$$P = \{ j : x_j > \varepsilon \text{ 또는 } |\Delta x_j| / x_j \le |\Delta s_j| / s_j \},$$
$$\overline{P} = \{1, 2, \cdots, n\} \setminus P$$

세 가지 방법을 서로 비교 실험해 본 결과, 두 가지 방법을 혼합하는 방법이 가장 우수한 결과를 보였다. 그리고, 변수 지시자나 혼합 방법에서 ε의 값은 10^{-6}을 사용하는 것이 가장 좋은 결과를 보였다.

4. 문제 축소 기법

Megiddo의 알고리즘의 원국면은 원단체법과 유사하고, 쌍대국면은 쌍대단체법과 유사하다. 그러나 가장 큰 차이점은 진입변수나 탈락변수를 선정하는데 있어서 그 순서가 중요하지 않다는 점이다. 이러한 점을 이용하여 주어진 문제의 크기를 축소하는 기법을 Andersen이 제안하였는데, 본 연구에서는 그가 제안한 기법들 중 큰 효과가 있는 원 블록 고정법(Primal block fix)과 쌍대 블록 고정법(Dual block fix)을 구현하였다.

우선 원 블록 고정법에 대해 살펴보자. $\overline{N} = \{ j \in P \cap N : x_j^0 > 0 \}$이라고 정의하면, Megiddo의 원국면은 다음의 선형방정식의 기저해를 찾는 과정이라고 할 수 있다.

$$B\boldsymbol{x}_B + \overline{N}\boldsymbol{x}_{\overline{N}} = \boldsymbol{b}, \quad \boldsymbol{x}_B, \boldsymbol{x}_{\overline{N}} \ge 0$$

행렬 $[B\ \overline{N}]$이 다음과 같이 순서화 될 수 있다고 하자. 단, B_{11}은 정방행렬이다.

$$\begin{bmatrix} B_{11} & \\ B_{21} & B_{22}\ \overline{N}_{21} \end{bmatrix}$$

그러면 가역행렬 B_{11}에 해당되는 변수들은 모두 고정될 수 있다. 즉, B_{11}에 대응되는 변수들은 그 값이 미리 정해져서 원국면의 고려대상에서 제외될 수 있다. 이런 식으로 변수의 값을 미리 고정하여 문제의 크기를 줄이는 방법을 원 블록 고정법이라고 한다. 행렬 $[B\ \overline{N}]$를 위와 같이 순서화하기 위해, 우선 $I = \{ i : nonzero(\overline{N}_{i\cdot}) = 0 \}$를 정의하자. $nonzero(\boldsymbol{v})$는 벡터 $\boldsymbol{v}$의 비영요소 수를 뜻한다. 그런 후 B_I를 다음과 같이 블록 하삼각 형태로 순서화 한다.

$$\begin{bmatrix} B'_{11} & \\ B'_{21} & B'_{22} \end{bmatrix}$$

여기서 B'_{11}는 B'의 최대 정방 부분행렬이다. 그러면 B'_{11}는 B_{11}이 된다.

한편 B_I의 블록 하삼각 형태를 찾는 방법은 Pothen과 Fan[11]의 연구를 이용한다. Pothen과 Fan의 방법은 위와 같은 형태로 순서화하는 coarse decomposition과 B'_{11}과 B'_{22}를 블록단위로 더욱 세분하는 fine decomposition으로 구성되는데, 본 연구에서는 coarse decomposition만이 필요하기 때문에 더욱 빠른 시간내에 순서화가 수행될 수

있다.

원 블록 고정법의 효과를 알아보기 위해 원국면의 수행 초기에 적용하여 얼마나 많은 변수들이 고정되는지 실험해 보았다. 그 결과는 다음 표에 나타나 있는데, PFIX가 고정된 변수의 개수를 표현하고 있다. 실험대상으로 삼은 문제는 NETLIB[7] 문제들이다.

〈표 2〉 원 블록 고정법의 효과

문제	행	열	PFIX	문제	행	열	PFIX
25fv47	6856	1738	185(185)	degen3	1410	2511	934(934)
80bau3b	2016	11095	1554(1554)	greenbea	1455	4510	808(808)
bnl2	967	2906	320(320)	pds-02	1393	6123	842(1147)
cycle	1114	2560	547(644)	pds-06	6833	26149	4431(6648)
d6cube	402	6182	270(270)	ken-07	893	2069	840(840)
d2q06c	1884	5541	703(703)	ken-11	5655	12310	5481(5481)
pilot	1344	4552	177(177)	cre-a	2986	6803	2610(2610)
pilotja	754	1855	138(138)	cre-b	7094	76991	6603(6603)
pilotwe	626	2634	118(118)	osa-07	1081	25030	1057(1057)

단, 표에 나타난 행과 열의 수는 입력 자료에 대한 사전처리(Preprocessing)가 수행된 이후의 수를 나타내고 있다. 실험 결과에서 알 수 있듯이 고정되는 블록의 크기가 커서, 행의 경우에는 30~80%까지 제거되고 열의 경우에는 20~30%정도 제거되는 효과가 있었다. 반면 원 블록 고정법을 수행하는데 소요되는 계산량은 거의 무시할 만하여, 원 블록 고정법의 효과에 의한 원국면의 수행속도 향상은 컸다. 특히 NETLIB 문제들 중에서 Kennington 문제의 경우에는 다수상품문제라는 문제구조의 특성 때문에 고정되는 블록의 크기가 상당히 컸다.

이제 쌍대 블록 고정법에 대해 살펴보자. 쌍대 국면에서 가장 계산량이 많은 부분은 $\boldsymbol{\pi} = B^{-T}\boldsymbol{e}_i$를 계산하고, 기저역행렬을 수정하는 부분이다. 따라서, 기저의 크기를 줄일 수 있다면 쌍대국면의 계산량을 줄일 수 있다. 기저 B가 다음과 같이 블록 상삼각 형태로 순서화 될 수 있다고 하면, $B^T\boldsymbol{\pi} = \boldsymbol{e}_i$는 다음과 같이 된다.

$$\begin{bmatrix} B_{11}^T & \\ B_{12}^T & B_{22}^T \end{bmatrix} \boldsymbol{\pi} = \begin{bmatrix} \boldsymbol{e}_i^1 \\ \boldsymbol{e}_i^2 \end{bmatrix} = \boldsymbol{e}_i$$

위 식에서 B_{11}는 정방 가역행렬이다. 그러므로 $\boldsymbol{\pi}$는 다음과 같이 구해질 수 있다.

$$\boldsymbol{\pi} = \begin{bmatrix} 0 \\ B_{22}^{-T}\boldsymbol{e}_i^2 \end{bmatrix}$$

즉, B_{11}에 대응하는 부분의 $\boldsymbol{\pi}$는 쉽게 구해질 수 있는 것이다. 이러한 관찰을 바탕으로 기저행렬을 블록 상삼각 형태로 변환시켜 후, 일정 블록을 고정시킴으로써 기저의 크기를 줄이는 것을 쌍대 블록 고정법이라고 한다. Bixby에 따르면 현실 세계의 문제는 블록 삼각 형태를 많이 띄고 있다고 한다.

이제 다음과 같은 두가지 집합을 정의하자.

$$\hat{B} = \{ j \in B : s_j^* = 0 \}, \quad \overline{B} = B \setminus \hat{B}$$

그리고 기저행렬이 다음과 같이 순서화 된다고 하자.

$$[\hat{B}\ \overline{B}] = \begin{bmatrix} \hat{B}_{11} & \hat{B}_{12} & \overline{B}_{11} \\ & \hat{B}_{22} & \overline{B}_{21} \end{bmatrix}$$

그런데, $\hat{B}_{11}$에 대응되는 쌍대 여유변수들은 초기저 변수가 아니기 때문에 $\boldsymbol{e}_i$의 해당 부분은 모두 영이다. 따라서, $\hat{B}_{11}$에 대응되는 $\boldsymbol{\pi}$의 값도 항상 모두 영이다. 그러므로 $\hat{B}_{11}$는 쌍대국면에서 계산상 제외될 수 있다.

한편 블록 상삼각 형태를 찾는 방법은 원국면에서와 마찬가지로 Pothen과 Fan의 연구를 이용한다.

쌍대 블록 고정법의 효과를 알아보기 위해 쌍대국면의 수행 초기에 적용하여 얼마나 많은 변수들이 고정되는지 실험해 보았다. 그 결과는 <표 2>에

나타나 있는데, DFIX가 고정된 변수의 개수를 표현하고 있다.

〈표 2〉 쌍대 블록 고정법의 효과

문제	행	열	DFIX	문제	행	열	DFIX
25fv47	685	1738	638(685)	degen3	1410	2511	88(341)
80bau3b	2016	11095	1670(2016)	greenbea	1455	4510	722(1434)
bnl2	967	2906	401(965)	pds-02	1393	6123	579(1315)
cycle	1114	2560	567(1103)	pds-06	6833	26149	615(6814)
d6cube	402	6182	0(0)	ken-07	893	2069	655(655)
d2q06c	1884	5541	1701(1875)	ken-11	5655	12310	3753(5441)
pilot	1344	4552	1315(1335)	cre-a	2986	6803	353(2840)
pilotja	754	1855	124(124)	cre-b	7094	76991	369(6905)
pilotwe	626	2634	602(626)	osa-07	1081	25030	879(879)

단, 표에 나타난 행과 열의 수는 입력 자료에 대한 사전처리(Preprocessing)가 수행된 이후의 수를 나타내고 있다. 실험에 사용한 문제들은 NELIB 문제들이다. 실험 결과에서 알 수 있듯이 고정되는 블록의 크기가 커서, 행의 경우에는 30~80%까지 제거되고 열의 경우에는 20~30%정도 제거되는 효과가 있었다. 반면 쌍대 블록 고정법을 수행하는데 소요되는 계산량은 거의 무시할 만하여, 쌍대 블록 고정법의 효과에 의한 쌍대국면의 수행속도 향상은 컸다. 특히 NETLIB 문제들 중에서 Kennington 문제의 경우에는 다수상품문제라는 문제구조의 특성 때문에 고정되는 블록의 크기가 상당히 컸다

원 블록 고정법이나 쌍대 블록 고정법은 모두 기저가 바뀌면 또 다시 적용될 수 있는 방법들이다. 그러므로 본 연구에서는 기저행렬에 대한 재역산(Refactorization)이 일어날 때마다 그 방법들을 재 적용하였다. 재역산은 기저행렬의 상하분해에 대한 수치적 안정성, 희소도 등을 고려하여 수행된다. <표 1>과 <표 2>에서 괄호( ) 안의 숫자는 재역산 마다 원 블록 고정법과 쌍대 블록 고정법을 수행하였을 때, 고정된 블록의 크기의 합을 나타내고 있다. 이를 볼 때, 재역산 마다 그 방법들을 적용하는 것이 어느정도 기저의 크기를 줄이는데 효과가 있음을 알 수 있다.

5. 초기기저의 구성

Megiddo의 알고리즘은 초기기저의 설정을 필요로 한다. 초기기저에 따라 원국면 및 쌍대국면의 수행횟수가 달라지게 되므로 효과적인 초기기저의 설정은 해법의 성능 향상을 위해 필수적이다.

가장 간단하게 생각하면, 인공변수로 구성된 초기기저를 설정할 수 있다. 그러나 P로부터 최대한 많은 기저를 구성하는 것이 훨씬 효과적이다. 왜냐하면 P로부터 많은 기저를 선택할수록 초기저 변수의 개수가 줄어들게 되므로 원국면의 반복회수를 줄일 수 있고, 인공변수의 개수가 작아지므로 쌍대국면의 반복회수가 줄어들 수 있기 때문이다. 예를 들어 비퇴화 문제의 경우 P는 최적기저와 일치하고, P로부터 기저를 모두 구성할 수 있다면 바로 최적기저를 추출할 수 있다.

본 연구에서는 초기기저를 구성하는 방법으로 두 가지 방법을 고려하였다. 첫째는 (P, N)에서 기저를 구성하는 방법이고, 둘째는 (P, I)에서 기저를 구성하는 방법이다. 다시 말하면 첫째 방법은 구조변수에서 기저를 구성하는 방법이고, 둘째는 P와 인공변수에서 기저를 구성하는 방법이다. 첫번째 방법을 적용하기 위해서는 N에 선택되어 기저에 포함된 변수의 값이 변해서는 안된다. 이것은 쌍대간격을 영으로 유지하기 위함이다. 이를 위해 본 연구에서는 N에서 선택된 기저변수의 상한을 영으로 설정하여 그 값이 변하지 않도록 하는 방법을 사용하였다.

실험결과에 의하면, 최적기저 추출과정은 초기기저의 수치적 안정성에 많은 영향을 받는다. 따라서 P와 인공변수에서 기저를 구성하는 방법이 수치적 안정성 면 뿐만아니라 전체 해법의 수행성능 면에서 우수하였다.

P와 인공변수에서 기저를 구성하기 위해 본 연구에서는 가우스 소거법을 사용하였다. 즉, P에 속한 열들로 구성된 행렬에 가우스 소거법을 적용하여 일차독립성을 검사하였다. P에 속한 열들로

m개의 일차독립열을 추출하지 못했을 경우에 인공변수를 도입하였다.

6. Crashing 기법

Megiddo 알고리즘의 쌍대국면에서는 주로 인공변수를 기저에서 탈락시키는 과정이 반복된다. 인공변수의 개수는 원퇴화의 정도에 비례하므로 쌍대국면의 반복회수는 원퇴화의 정도에 의존적이라고 할 수 있다. 일반적으로 $|P|$가 행의 수 m보다 작다면 적어도 $m-|P|$번의 쌍대국면의 반복회수가 소요된다. 현실 세계의 원퇴화가 아주 심한 문제들에서는 $A_{\cdot P}$는 적절한 행 순서화를 통해 다음과 같은 형태를 가진다.

$$\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ & & & & \end{bmatrix}$$

즉, $A_{\cdot P}$는 많은 공백행을 가지고 있다. 이것은 원퇴화가 심한 선형계획법 문제는 다음과 같은 형태의 제약식을 다수 포함하고 있다는 것을 의미한다.

$$\sum_{j\in T} a_{ij}x_j = b_i$$

위 식에서 T는 $\overline{P}$의 부분집합이다. 즉, 최적해에서 $x_j^* = 0$, $j\in T$가 성립한다. 이러한 관찰을 바탕으로, 만일 Megiddo 알고리즘의 초기기저가 P와 인공변수에서 구성된다면 초기기저는 적절한 순서화를 통해 다음과 같은 형태를 가진다는 것을 알 수 있다.

$$\begin{bmatrix} * & * & * & \cdot & \cdot & \cdot & & \\ * & * & * & \cdot & \cdot & \cdot & & \\ * & * & * & \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & & & & \\ \cdot & \cdot & \cdot & & \cdot & & & \\ \cdot & \cdot & \cdot & & & \cdot & & \\ & & & & & & * & \\ & & & & & & & * \end{bmatrix} = \begin{bmatrix} B_{11} & \\ & I \end{bmatrix}$$

위의 형태에서 I에 해당하는 열들이 인공변수에 해당된다.

쌍대국면에서는 B_{11}에 속하지 않는 변수들을 먼저 탈락시킬 수 있다. 집합 T를 다음과 같이 정의하자.

$$T = \{ i : a_{ij} \neq 0 \text{이면 } j\in \overline{P}\}$$

그러면 집합 T는 인공변수로 이루어진 블록에 해당하는 행들의 집합을 표현한다. 그러므로 T에 속한 행들로 구성된 부분 문제를 구성할 수 있고, 이러한 경우에 초기기저는 단위행렬이 된다. 단위행렬이 기저행렬이 되므로 쌍대국면에서 풀어야하는 선형방정식 $B^T\pi = e_i$이 $\pi = e_i$로 쉽게 풀릴 수 있다. 그러나, 첫 번째 선회연산 후에 기저행렬은 이제 단위행렬이 아니다. 그렇지만 기저가 다음과 같이 변했다고 했을 때,

$$\begin{bmatrix} * & & * & & \\ & * & * & & \\ & & * & & \\ & & & * & \\ & & & * & * \\ & & & & * \end{bmatrix}$$

즉 4번째 기저 인공변수가 탈락되고 위 행렬의 4번째 열과 같은 형태의 구조변수가 진입했을 때, 2행과 6행은 여전히 단일요소행으로 남는다. 이것은 두 번째, 여섯 번째 행이 탈락행이 될 경우 $B^T\pi = e_i$의 해가 $\pi = e_i$로 쉽게 구해질 수 있다는 것을 의미한다. 이런 식으로 인공변수를 구조변수로 대체하는 과정을 별다른 계산없이 진행하고자 하는 것이 crashing 기법이다. Crashing 기법을 구현하기 위해서는 어떤 행이 단일요소행으로 남아있는지만 기록하고 있으면 되고, crasing 기법이 적용되는 과정에서는 기저역행렬 수정이나 π를 구하기 위한 선형방정식의 풀이도 필요없게 되어 쌍대국면의 효율화를 이룰 수 있게 된다. 다음은 쌍대국면에서 crashing을 적용할 때, 얼마나 많은 계산량 절감이 있는지를 나타내는 실험결과이

〈표 4〉 최적기저 추출과정의 계산시간

문 제	행	열	기저추출	전 체	문 제	행	열	기저추출	전체
25fv47	685	1738	0.08	1.85	degen3	1410	2511	0.75	15.57
80bau3b	2016	11095	0.14	7.11	greenbea	1455	4510	3.92	15.45
bnl2	967	2906	0.35	5.81	pds-02	1393	6123	0.50	2.33
cycle	1114	2560	0.20	3.72	pds-06	6833	26149	8.64	96.94
d6cube	402	6182	0.43	7.40	ken-07	893	2069	0.05	0.74
d2q06c	1884	5541	0.43	14.18	ken-11	5655	12310	0.49	6.62
pilot	1344	4552	1.59	37.75	cre-a	2986	6803	1.55	5.42
pilotja	754	1855	0.48	4.45	cre-b	7094	76991	50.01	396.57
pilotwe	626	2634	0.18	2.03	osa-07	1081	25030	0.24	16.75

다. 표에 나타난 결과는 기저추출과정에 소요되는 계산 시간이며, SUN Ultra Sparc 170에서 실험하였다.

〈표 3〉 Crashing의 효과

문 제	적용 후	적용 전	문 제	적용 후	적용 전
25fv47	1.15	1.64	degen3	2.92	5.61
80bau3b	6.33	10.88	greenbea	3.27	6.19
bnl2	1.44	1.63	pds-02	3.35	6.16
cycle	27.95	41.77	pds-06	1.51	2.59
d6cube	4.50	7.23	ken-07	1.94	3.11

Kennington 문제, 특히 PDS 계열의 문제들은 원퇴화가 아주 심해서 쌍대국면의 반복회수가 상당히 많다. 따라서, 쌍대국면의 계산량을 줄여주는 crashing 기법의 효과가 뛰어나다는 것을 위 실험결과를 통해 알 수 있다.

7. 실험 결과

본 연구에서는 지금까지 살펴본 여러 가지 기법들의 성능을 평가하기 위해, 모든 기법들을 구현하여 기저추출 과정을 완성하였다. 기저추출 과정을 구현한 프로그램과 결합된 내부점 방법 프로그램은 서울대학교 산업공학과 박순달 교수에 의해 개발된 LPABO[2]이다. 다음 표는 내부점 방법의 종료 후 최적기저를 추출할 때 까지 소요되는 계산시간을 전체 해법 계산시간과 함께 나타낸 것이다. 실험 대상으로 삼은 문제는 NETLIB 문제들 중 일부분이고, 모든 실험은 SUN UltraSparc 170에서 수행하였다.

이 실험결과를 통해 볼 때, 기저추출의 과정은 전체 해법 계산량의 10% 내외를 차지한다는 것을 알 수 있는데, 이는 비교적 적은 계산량으로 기저해를 추출해 낼 수 있다는 것을 보여준다(<표 4> 참조).

Bixby는 내부점 방법의 최종해로부터 얻은 정보를 이용하여 기저를 구성한 다음 단체법을 적용하여 최적 기저해를 찾는 crossover 방법을 제안하였는데, 기존의 crossover 방법과 본 연구에서 구현한 Megiddo 알고리즘을 이용한 방법의 성능을 서로 비교였다. 이를 위해 Bixby가 제안한 crossover 방법을 구현하였고, 이를 본 연구의 구현 결과와 비교하였는데, 이는 <표 5>와 같다.

〈표 5〉 Crossover 방법의 계산시간

문제	기저추출	문제	기저추출
25fv47	0.15	degen3	0.99
80bau3b	0.28	greenbea	5.14
bnl2	0.54	pds-02	3.78
cycle	0.47	pds-06	14.41
d6cube	0.87	ken-07	1.51
d2q06c	0.51	ken-11	1.91
pilot	2.97	cre-a	3.11
pilotja	1.58	cre-b	83.43
pilotwe	0.97	osa-07	1.05

위 실험결과를 통해 볼 때, 본 연구에서 구현한 방법이 crossover 방법에 비해 평균 2.5배 정도 우수함을 알 수 있다.

8. 결 론

최적기저를 이용한 전통적인 감도분석이나, 정수계획법이나 비선형계획법과 같이 서로 비슷한 일련의 선형계획법을 효과적으로 풀기 위한 warm-starting의 구현에서는 최적기저해가 반드시 필요하다. 본 연구에서는 선형계획법을 빠르게 풀어낼 수 있는 내부점 방법이 지니는 약점 중의 하나인 최적기저를 얻을 수 없다는 점을 극복하기 위해, 내부점 방법의 최종해를 이용하여 최적기저해를 추출하는 방법을 다루었다. Megiddo 알고리즘이 적용되기 위해 필요한 최적 강상보해를 얻기 위해, 문제를 변형하여 근사 선형계획법 문제를 구성한다는 Andersen과 Ye의 방법을 구현하고, Andersen이 제안한 문제 축소방법 및 crashing 기법을 활용하여 최적기저 추출과정을 효율화하였다. 근사 선형계획법 문제를 구성하기 위해서는 최적분할 판정방법을 사용하여야 한다. 본 연구에서는 변수지시자와 Mehrotra, Ye가 제시한 방법을 혼합하여 사용하였다.

문제 축소방법의 일환으로 원 블록 고정법, 쌍대 블록 고정법을 구현하여 그 효과를 실험적으로 검증하였다. 실험결과에 의하면 원, 쌍대 블록 고정법을 활용하여 50%이상의 블록이 제거되었다.

원국면이나 쌍대국면의 반복회수를 줄이기 위해서는 초기기저의 설정이 중요한데, 본 연구에서는 P와 인공변수로 초기기저를 구성하는 방법이 수치적 안정성 면이나 반복회수 면에서도 우수하다는 것을 실험적으로 검증하였다.

인공변수를 기저에서 탈락시키는 쌍대국면의 계산량을 줄이기 위해서, 원퇴화가 심한 문제의 구조적 특징을 활용한 crashing 방법을 구현하여 1.5~3배의 계산량 절감 효과를 거둘 수 있었는데, 특히 다수상품문제인 PDS계열의 문제에서 2~3배의 계산량 절감 효과를 얻었다.

내부점 방법의 최종해로부터 최적기저를 추출하는 방법은 이제 이론적 연구가 한계에 도달하였다고 할 수 있다. 즉, 강성다항식의 복잡도를 가지는 Megiddo의 알고리즘 보다 더 낮은 복잡도를 가지는 알고리즘의 개발 여부는 그리 긍정적이지 않다. 따라서 효과적인 구현을 통한 기저추출과정의 효율화가 앞으로의 과제라고 할 수 있다. 예를 들어 안정적인 새로운 지시자의 개발, 기저역행렬 연산의 효율화, 추가적인 문제 축소기법의 개발 등이 계속 추구해야할 방향이 된다.

참 고 문 헌

[1] 박순달, 선형계획법, 민영사, 1999.

[2] 박순달, LPABO ver 5.7 사용자 매뉴얼, 1999.

[3] Andersen E.D., "On Exploiting Problem Structure in a Basis Identification Procedure for Linear Programming," INFORMS Journal on Computing, Vol.11, No.1, 1999.

[4] Andersen E.D., Y. Ye, "Combining Interior-Point and Pivoting Algorithms," Management Science 42, 1719-1731, 1996.

[5] Bixby R.E., "Progress in Linear Programming," ORSA Journal on Computing 6, pp.15-22, 1994.

[6] Bixby R.E., M. J. Saltzman, "Recovering an Optimal Basis from an Interior-Point Solution," Operations Research Letters 15, pp.169-178, 1993.

[7] Gay D.M., "Electronic mail distributioin of linear programming test problems," Mathematical Programming Society Committee on Algorithms Newsletter, No.13, pp.10-12, 1985

[8] Greenberg H.J., "The Use of the Optimal Partition in a Linear Programming Solution for Postoptimal Analysis," Operations Research Letters 15, pp.179-186, 1994.

[9] Kortanek K.O., J. Zhu, "New Purification Algorithms for Linear Programming," Naval Research and Logistics Quarterly 35, pp.571-583, 1988.

[10] Megiddo N., "On finding primal and dual-optimal bases," ORSA Journal on Computing, Vol.3, No.1, Winter 1991.

[11] Pothen A., C. Fan, "Computing the Block Lower Triangular Form of Sparse Matrix," ACM Transactions on Mathematical Software 16, pp.303-324, 1990.

[12] Ye. Y., "On the Finite Convergence of Interior-Point Algorithms for Linear Programming," Mathematical Programming 57, pp.325-335 1992.