내부점 선형계획법의 쌍대문제 전환에 대하여†

On dual transformation in the interior point method of linear programming

설동렬*, 박순달*, 정호원**
* 서울대학교 산업공학과, ** 고려대학교 경영학과

Abstract

In Cholesky factorization of the interior point method, dense columns of A matrix make dense Cholesky factor L regardless of sparsity of A matrix. We introduce a method to tranform a primal problem to a dual problem in order to preserve the sparsity.

KEYWORDS: Cholesky factorization, interior point method, dense column, sparsity, primal problem, dual problem

1. 서론

모든 내부점 선형계획법은 대칭양정치 행렬의 선형방정식을 푸는 과정이 필요하다. 일반적으로 선형계획법 문제들은 희소한 특성을 가지는데, 내부점 선형계획법에서 나타나는 대칭양정치 행렬도 희소행렬이다[3]. 내부점 선형계획법에서는 전체 해법 소요 시간의 많은 부분이 선형방정식을 푸는 데에 소요되어, 효율적으로 선형방정식을 풀어내는 것이 해법의 성능을 크게 좌우한다.

내부점 선형계획법에서 나타나는 대칭양정치 행렬은 $A\Theta A^T$로 표현한다. 단, 행렬 A는 $m\times n$ 행렬이고, Θ는 $n\times n$의 대각행렬이고 모든 대각요소의 값은 양이다. Θ는 내부점 선형계획법의 종류에 따라서 다르게 계산된다. 대칭양정치 행렬 $A\Theta A^T$는 $m\times m$ 행렬이다. 내부점 선형계획법에서의 선형방정식은 다음과 같다.

$$(A\Theta A^T)x = b$$

단, x과 b는 $m\times 1$ 벡터이다.

행렬 A에 밀집열(dense column)이 존재하면 행렬 $A\Theta A^T$의 밀집도는 매우 높아지는 성질을 가지고 있다. Θ가 대각행렬이기 때문에 $A\Theta A^T$의 비영요소 구조에 영향을 주지 않는다. 따라서, $A\Theta A^T$의 비영요소 구조는 AA^T의 비영요소 구조와 동일하다. 행렬 AA^T의 (i,j)요소는 행렬 A의 i번째 행과 행렬 A^T의 j번째 열(즉, 행렬 A의 j번째 행)을 곱해서 얻는다. AA^T의 (i,j)요소가 비영요소이려면 행렬 A의 i번째 행과 j번째 행에서 모두 비영요소를 가지는 열이 적어도 하나 존재해야 한다. 거꾸로 이야기하면, 행렬 A의 k번째 열에서 비영요소를 가진 행들의 쌍들은 모두 AA^T에서 비영요소를 가지게 된다. 따라서, 행렬 A의 전체적인 밀집도는 낮다고 할지라도 특정한 열의 밀집도가 높은 경우에는 행렬 $A\Theta A^T$의 밀집도를 높이게 된다. 행렬 $A\Theta A^T$의 밀집도가 높으면 선형방정식을 푸는 데에 많은 시간이 소요된다. 이것은 내부점 선형계획법의 수행속도를 저하시키는 결과를 가져오게 된다.

행렬 A에 존재하는 밀집열을 해결하기 위한 기법으로, Schur complement를 이용하여 밀집열을 제거하는 방법[5], Conjugate gradient 방법[2] 그리고 밀집열을 여러 개로 쪼개는 분할 방법[7] 등이 개발되었다. Schur complement 방법은 수치적으로 안정적이지 않다고 알려져 있으며, splitting 방법이 가장 효율적인 방법으로 알려져 있다.[4] 이 밖에 Vanderbei[6]가 제안한 쌍대문세 전환 기법은 입력된 문제를 쌍대문제로 전환하여 밀집열을 제거하는 방법으로, 행렬분해 과정에서 밀집열을 따로 고려할 필요가 없는 방법이다.

본 논문의 구성은 다음과 같다. 1장은 내부점 선형계획법에서 밀집열의 처리에 대한 개요를 다루고 있고, 2장은 원문제로부터 쌍대문제를 생성하는 방법을 3장은 쌍대문제 전환

† 본 연구는 한국과학재단의 목적기초연구과제(과세번호 95-0200-39-01-2)에 의해 지원되었음

기법을 내부점 선형계획법에 효율적으로 적용하기 위한 방법을 제시한다.

2. 쌍대문제 생성 방법

쌍대문제 전환을 위해서 행렬 A는 열단위 압축 방법과 행단위 압축 방법을 각각 사용하여 이중으로 보관한다. 열단위 압축 방법은 행렬 A를 열단위로 비영요소만 보관하고, 열의 시작위치와 행번호는 따로 배열로 보관하는 방법이고, 행단위 압축 방법은 열단위 압축 방법을 행에 대해서 적용한 방법이다[1]. 두 가지 자료구조를 동시에 가지고 있을 경우에 쌍대문제 전화 시에 행렬 A와 행렬 A^T의 전환이 간단하게 처리된다. 또한 선형계획법 해법 수행 중의 계산 중에 계산 형태가 열단위로 발생하면 열단위 압축 자료구조를 사용하고, 행단위로 발생하면 행단위 압축 자료구조를 사용할 수 있어서 효율적인 메모리 접근이 가능하다. 문제 입력 단계에서 열단위 압축 자료구조를 생성한 다음, 행단위 압축 자료구조는 열단위 압축 자료구조로부터 생성한다.

쌍대문제 전환을 수행하기 위해서 원문제로부터 쌍대문제를 생성할 때에 원문제에 새로운 변수를 추가하여 쌍대문제의 밀집열 구조를 변형시키지 않는 것이 중요하다. 원문제에 새롭게 추가된 변수는 쌍대문제에서는 새로운 제약식이 되므로, 쌍대문제에서 행렬 $A\Theta A^T$의 밀집도가 높아질 수 있고, 또한 행렬 $A\Theta A^T$의 크기가 커지게 되어 비영요소 개수와 계산시간이 증가하게 된다. 따라서, 쌍대문제로 변환할 수 있는 형태를 만들기 위해 변수를 추가하는 방법을 피하고, 제약식을 추가하는 방법을 사용하여야 한다.

원문제를 최대화 문제로 가정하고 쌍대문제 생성 방법을 설명한다. 선형계획법 기본형 문제는 다음과 같이 쌍대문제로 변환될 수 있다.

$$\begin{array}{lll} Max\ CX & & Min\ b^TY \\ s.t.\ AX \le b & \rightarrow & s.t.\ A^TY \ge C \\ \quad X \ge 0 & & \quad Y \ge 0 \end{array}$$

따라서, 원문제로부터 쌍대문제를 생성하기 위해서 먼저 모든 제약식을 '≦' 제약식으로 바꾸어 위와 같이 쌍대문제를 생성한다. '≧' 제약식은 '-'을 곱하는 것으로 간단히 해결할 수 있다. 문제가 되는 것은 '=' 제약식, 상하한 변수제약 그리고 RANGE 제약식이다. '=' 제약식은 2개의 '≦' 제약식으로 바꾼다. '=' 제약식은 대응하는 쌍대변수를 자유변수로 처리하면 굳이 2개의 '≦' 제약식으로 바꿀 필요가 없다. 그러나, 내부점 선형계획법에서는 자유변수를 2개의 변수로 나누어 처리하는 방법 이외에 자유변수를 그대로 해결하는 방법이 없기 때문에, '=' 제약식이 많은 문제의 경우에 쌍대문제로 전환하면 매우 많은 자유변수가 생성되어 수치적으로 안정하지 않을 수 있다. 따라서, 다음과 같이 2개의 '≦' 제약식으로 처리한다.

$$AX = b \rightarrow \begin{array}{r} AX \le b \\ -AX \le -b \end{array}$$

상하한변수는 다음과 같이 각각 '≦' 제약식으로 바꾼다.

$$lb \le X \le ub \rightarrow \begin{array}{r} X \le ub \\ -X \le -lb \end{array}$$

상하한변수에 의해서 추가된 제약식은 제약식에 비영요소가 1개만 존재하므로 쌍대문제에서 $A\Theta A^T$의 밀집도에는 영향이 없다. RANGE 제약식은 2개의 '≦' 제약식으로 바꾼다. 행렬 A의 희소도만을 고려할 경우에는 RANGE 인공변수를 하나 추가하여 다음과 같이 RANGE 제약식을 처리할 수 있다.

$$b^1 \le AX \le b^2 \rightarrow \begin{array}{r} AX - Y = b^1 \\ 0 \le Y \le b^2 - b^1 \end{array}$$

그러나, 이 경우에 상한 변수제약과 '=' 제약식을 다시 처리해 주어야 하고, $A\Theta A^T$를 고려하면 변수의 추가는 바람직하지 않으므로 다음과 같이 행렬 A의 비영요소는 늘이나지만 두 개의 제약식으로 쪼개어 처리한다.

$$b^1 \le AX \le b^2 \rightarrow \begin{array}{r} AX \le b^2 \\ -AX \le -b^1 \end{array}$$

제약식을 추가하여 '≦' 제약식으로 만드는 것은 행단위 압축 자료구조를 이용하여 수행한다. 제약식 단위로 행렬 A에 비영요소가 추가되므로, 행단위 압축 자료구조를 사용하는 것이 편리하다. 그 이후에 행단위 압축 자료구조를 열단위 압축 자료구조로 사용한다. 즉, 원문제의 행단위 압축 자료구조가 쌍대문제의 열단위 압축 자료구조가 된다. 쌍대문제에서 행단위 압축 자료구조가 필요하면, 쌍대문제 생성을 위해 만든 열단위 압축 자료구조로부터 만들어 낸다. 목적함수계수와 제약식 추가와 함께 늘어난 우변상수 역시 서로 교환한다. 쌍대문제에서 모든 변수는 비음제약만을 가진다.

[표 1] 원문제와 생성된 쌍대문제의 특성

문제이름	문제형태	행개수	열개수	제약식			A의 비영요소
				≦	=	≧	
agg	원문제	488	163	405	36	47	2,410
	쌍대문제	163	524	0	0	163	2,698
israel	원문제	174	142	174	0	0	2,269
	쌍대문제	142	174	0	0	142	2,269
seba	원문제	515	1,028	0	507	1	4,352
	쌍대문제	1,028	1,537	0	0	1,028	9,205
fit1p	원문제	627	1,677	0	627	0	9,868
	쌍대문제	1,677	1,653	0	0	1,677	20,135
fit2p	원문제	3,000	13,525	0	3,000	0	50,284
	쌍대문제	13,525	13,500	0	0	13,525	108,068

[표 1]은 Netlib의 5가지 문제에 대하여 쌍대문제를 생성한 결과이다. israel은 '≦' 제약식만 있어서 비영요소의 개수에 변화가 없었다. 나머지 문제들에 대해서는 추가되는 제약식들로 인해서 행렬 A의 비영요소 개수가 많이 증가한다. 원문제에 추가된 변수가 없어서 원문제로부터 생성된 쌍대문제의 행갯수는 원문제의 열갯수와 동일하고, 쌍대문제의 열구조 역시 원문제의 행구조와 같기 때문에 $A\Theta A^T$의 비영요소 구조는 원문제에 비해서 좋은 성질을 가진다. 이에 대한 결과는 [표 2]를 통해 알 수 있다.

[표 2] 원문제와 쌍대문제의 비영요소 개수

문제 이름	A의 비영요소			L의 비대각 비영요소		
	원문제	쌍대문제	쌍대/원	원문제	쌍대문제	쌍대/원
agg	2,410	2,698	1.1195	10,854	4,740	0.4367
israel	2,269	2,269	1.0000	11,073	8,692	0.7850
seba	4,352	9,205	2.1151	56,833	7,858	0.1383
fit1p	9,868	20,135	2.0404	196,251	23,338	0.1189
fit2p	50,284	108,068	2.1492	4,498,500	190,142	0.0423

[표 2]의 결과에 의하면 israel을 제외한 모든 문제에 대해서 행렬 A의 비영요소수는 원문제보다 쌍대문제로 전환했을 경우에 더 늘어나지만, $A\Theta A^T$의 희소도가 개선되어 $A\Theta A^T$의 분해행렬 L의 비대각 비영요소수는 원문제에 비해서 매우 줄어든다는 것을 알 수 있다.

3. 쌍대문제 변환 시점

선형계획법 문제 입력 후에 밀집열 여부를 판정하여 해법을 적용하기 이전에 쌍대문제 전환이 일어난다. 쌍대문제는 밀집열에 의해서 $A\Theta A^T$ 행렬의 비영요소 수를 적게 하려는 것이 목적이므로, 원문제와 쌍대문제의 밀집열 정도를 비교하여 밀집도가 높은 것으로 선택하되 전체 해법 수행 시간을 고려하는 것이 중요하다.

본 연구에서는 다음과 같은 2가지 조건을 적용하여 쌍대문제 전환을 설정하였다.

[조건 1] 최대밀집열의 비영요소 개수 > 최대밀집행의 비영요소 개수

입력된 문제에서 최대밀집행의 비영요소 개수는 쌍대문제에서 최대밀집열의 비영요소 개수이므로 2가지 값을 비교하여 $A\Theta A^T$ 행렬의 잠재적 밀집도가 더 낮은 것을 선택하도록 하였다. Vanderbei[6]는 보다 정교한 판정을 위해서 원문제와 쌍대문제에서 $A\Theta A^T$ 행렬의 비영요소 추정치를 각각 구하여 비교하는 방법을 사용했다. 그러나, 추정치를 구하는 데에 복잡한 계산이 수행되고, 구한 값도 정확한 비영요소 개수가 아니라 추정치이므로 입력된 문제를 읽으면서 바로 계산할 수 있는 최대밀집열과 최대밀집행의 비영요소만으로 비교하는 것이 보다 효과적이다.

Netlib문제에 대해서 실험한 결과 Vanderbei가 밀집열 문제로 보고한 문제들을 모두 [조건 1]로 찾을 수 있다. [조건 1]에 해당되는 문제들은 agg(18<43), fit1p(21<627), fit2p(22<3000), israel(118<136), kb2(10<14), modszk1(21<23), sc105(4<5), sc205(4<5), sc50a(4<5), seba(17<230) 등의 10개 문제들이다. 괄호 안의 앞쪽 숫자는 최대밀집행의 비영요소수, 뒤쪽 숫자는 최대밀집열의 비영요소수이다.

[조건 2] 최대밀집열의 비영요소 개수 > 상수

[조건 1]에서 최대밀집열 문제로 판정된 문제들 가운데, kb2, sc105, sc205, sc50a, modszk1 등의 문제는 최대밀집열의 개수와 최대밀집행의 개수가 거의 비슷할 뿐만 아니라, 쌍대문제 전환에 소요되는 시간과 쌍대문제 전환에 의해서 추가되는 비영요소를 고려할 때에 오히려 원문제를 그대로 다루는 것이 적절한 문제들이다.

따라서, 최대밀집열의 비영요소 개수가 일정한 상수값 이상이 되었을 때에 비로소 [조건 1]을 사용하여 쌍대문제 전환 여부를 판정하는 효과적이다. 한편, agg 문제는 최대밀집행의 비영요소수가 18, 최대밀집열의 비영요소수가 43이기 때문에 쌍대문제 전환을 사용하

는 더욱 효과적이지만, 최대밀집열의 개수가 일정한 상수보다 적을 경우에 $A\Theta A^T$의 밀집도에 많은 영향을 주지 않으므로 계산 시간에 큰 차이를 나타내지 않는다. 상수값을 50으로 했을 때에 [조건 1]과 [조건 2]를 모두 만족하는 문제는 israel, seba, fit1p, fit2p이다.

[표 3]은 agg, israel, seba, fit1p, fit2p에 대한 실험 결과이다. 원쌍대내부점방법을 사용한 선형계획법 프로그램으로 SunSPARC Ultra 170에서 실험하였다.

[표 3] 원문제와 쌍대문제의 계산 속도

문제이름	계산회수		계산시간		
	원문제	쌍대문제	원문제	쌍대문제	쌍대/원
agg	29	53	1.450	1.320	0.9101
israel	31	26	2.860	1.650	0.5769
seba	19	36	18.620	4.790	0.2573
fit1p	19	23	110.910	13.030	0.1175
fit2p*	-	30	-	441.170	-

* fit2p는 원문제로 계산할 경우에 메모리 소요량이 매우 커서 풀지 못했다.

fit1p, fit2p 문제는 [표 1]에서 알 수 있듯이 원문제에서 분해행렬 L이 완전밀집행렬(full matrix)가 된다. 따라서, 희소행렬 기법을 적용할 경우에 오히려 추가되는 계산만 늘어나는 결과를 얻게 된다. 그러나, 쌍대문제 전환기법을 사용하면, 희소성을 유지하여 계산시간이 매우 단축된다. fit2p와 같은 경우에는 분해행렬 L의 비영요소 개수가 4,498,500가 되어 풀기 어려운 문제이지만, 쌍대문제로 전환하여 풀면 쉽게 해결할 수 있다. agg, seba, fit1p는 원문제에 비해서 쌍대문제의 계산회수가 더 많은 경우이다. 그러나, 늘어난 계산 회수에 비해서 매 회의 계산 속도가 향상되어 전체적으로 계산 속도는 향상된다.

4. 결론

원문제에서 행렬 A의 최대밀집열의 비영요소 개수 50보다 클 때에, 최대밀집행의 비영요소 개수보다 최대밀집열의 비영요소 개수가 더 크면 쌍대문제로 전환하여 내부점 선형계획법을 수행하는 것이 효율적이다. 쌍대문제로 전환할 때에는 쌍대문제에서 분해행렬의 비영요소가 증가하지 않도록 원문제에 변수는 추가하지 않고, 제약식만을 추가하는 방법을 사용하여 쌍대문제를 생성한다. 이 때에 쌍대문제로 전환된 문제에서 행렬 A의 비영요소 수는 원문제보다 증가하지만, 분해행렬의 비영요소는 감소한다.

쌍대문제 전환기법을 사용하기 위해서는 내부점 선형계획법을 수행할 때에 정확한 쌍대해를 출력하는 것이 중요하다. 해법의 수행 결과 얻어진 쌍대최적해가 원래 입력된 원문제의 원최적해이기 때문이다. 따라서, 사전처리 또는 규모화를 수행하여 문제를 변형시켰을 경우에, 변형 문제를 원래대로 복귀하여 정확한 쌍대최적해를 출력할 수 있도록 하는 것이 필요하다.

5. 참고문헌

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

[2] Adler, I., N. Karmarkar, M. G. C. Resende and G. Veiga, "An implementation of Karmarkar's algorithm for linear programming", *Math Prog.*, 44:297-235, 1989

[3] Andersen, Erling D., Jacek Gondzio, Csaba Meszaros, Xiaojie Xu, "Implementation of interior point methods for large scale linear programming", Technical Report 1996. 3, Logilab, HEC Geneva, Section of Management Studies, University of Geneva

[4] Andersen, Knud D., "A modified Schur complement method for handling dense columns in interior point methods for linear programming", Technical Report, Odense University, Dec 1, 1994

[5] Choi, I. C., C. L. Monma and D. F. Shanno, "Further development of a primal-dual interior point method", *ORSA Journal on Computing*, (2):304-311, 1990

[6] Vanderbei, Robert J., "ALPO: Another Linear Program Optimizer", *ORSA Journal on Computing*, vol.5, no.2, Spring 1993

[7] Vanderbei, Robert J., "Splitting dense columns in sparse linear systems", *Linear Algebra Appl.*, 152:107-117, 1991