최소비용문제에서의 사전처리*
(Preprocessing for Minimum Cost Flow Problems)

엄순근, 박찬규, 박순달
서울대학교 산업공학과

초 록

본 연구는 최소비용문제에 적용할 수 있는 사전처리 기법의 이론과 그 구현에 대해서 다룬다. 일반적으로 해법을 적용하여 문제를 풀기 이전에 최적해에서 유통량을 알 수 있는 호나 중복적인 호와 점을 제거하여 문제 크기를 줄이는 과정을 사전처리(preprocessing)라 한다. 또한 문제의 비가능성이나 입력된 문제의 정확성 등을 검사하는 과정도 사전처리에 포함하기도 한다. 따라서 사전처리는 문제 축소와 입력된 문제의 정확성 검사 등을 통해 해법의 수행도와 안정성을 높이는 효과를 가져다준다.

본 연구에서는 최소비용문제의 사전처리로 비가능성 판정, 중개지에 대한 사전처리, 병렬호에 대한 사전처리, 호의 유통상한과 유통하한을 이용한 유통량고정에 대한 사전처리, 우회경로에 대한 사전처리 등을 연구하였다.

본 연구에서는 네트워크 단체법 프로그램에 최소비용문제에서의 사전처리기법을 각각 구현하여 이러한 사전처리를 하지 않았을 때와 비교하여 문제의 크기를 줄일 수 있었고 수행시간을 16%정도 줄일 수 있다는 것을 실험적으로 보였다.

1. 서론

현대사회로 오면서 네트워크로 모형화되고 있는 문제들이 급격히 늘어나고 있는 추세를 보여왔다. 또한 네트워크 문제의 해를 실시간(Realtime processing)동안 풀어야 할 때 네트워크 문제 해법 프로그램의 고속화는 필수 불가결해진다. 그리고 네트워크 문제 해법 프로그램이 다른 해법의 서브루틴으로 사용될 경우가 있는데 이러한 경우에는 네트워크 문제를 여러 번 풀어야 하므로 고속의 네트워크 프로그램의 필요성은 더욱 중요해 진다.

네트워크 문제 중에 흔히 나타나는 유형으로 크게 최단경로문제, 최대유통문제, 최소비용문제를 들 수 있다([1][4]). 이러한 문제를 입력받아 해법을 적용하여 문제를 풀기 이전에 최적해에서 유통량을 알 수 있는 호나 중복적인 호와 점을 제거하여 문제 크기를 줄이는 과정을 사전처리(preprocessing)라 한다([2][3][6]). 또한 문제의 비가능성이나 입력된 문제의 정확성 등을 검사하는 과정도 사전처리에 포함하기도 한다([2][3][7]).

사전처리로 얻을 수 있는 이익은 문제 축소와 입력된 문제의 정확성 검사 등을 통해 해법의 수행도와 안정성을 높이는 효과 등을 들 수 있다([2][5]). 더욱이 문제를 여러 번 반복해서 풀어야 할 경우 사전처리에 의해서 줄여진 문제를 반복해서 풀게 되면 수행시간을 줄이는 측면에서 더욱 큰 효과를 얻게 된다.

사전처리에 대한 기존 연구로는 주로 선형계획법 문제나 정수계획법 문제 등에서의 사전처리 연구가 있었다. 선형계획법에서의 사전처리는 변수나 제약식의 제거, 문제의 비가능성 판정, 변수의 상하한 수정 등의 과정을 거친다([2][3][8]).

본 연구에서는 최소비용문제에 적용할 수 있는 사전처리 기법의 이론과 그 구현에 대해서 다룬다. 네트워크 단체법 프로그램에 여러 가지 네트워크 사전처리 기법을 구현하여 사전처리를 하지 않았을 때와 비교 실험을 하여 네트워크 문제에서의 사전처리의 효과를 알아보고자 한다.

본 연구에서 다루는 최소비용문제에서의 사전처리는 중개지에 대한 사전처리, 병렬호에 대한 사전처리, 호의 유통상한과 유통하한을 이용한 유통량고정에 대한 사전처리, 우회경로에 대한 사전처리 등으로 구성되어 있다.

2. 기본적인 사전처리

본 연구에서 다루는 최소비용문제에서의 기본적인 사전처리는 문제의 비가능성 판정, 네트워크의 연결성과 고립된 점에 대한 처리 의 세 가지가 있다.

첫째, 문제의 비가능성 판정에 대한 사전처리가 있다. 이것은 해법을 수행하기 전에 문제의 비가능성을 미리 알아내어 불필요한 해법수행을 막고자 하는데 그 목적이 있다. 그 방법으로 어떤 점 i로 들어오는 모든 호들의 유통하한의 합과 점 i의 수요량 또는 공급량의 합이 점 i에서 나가는 모든 호들의 유통상한의 합보다 클 경우, 또 어떤 점 i로 들어오는 모든 호들의 유통상한의 합과 점 i의 수요량 또는 공급량이 점 i에서 나가는 모든 호들의 유통하한의 합보다 작을 경우에 비가능으로 판정하고 해법수행을 종료한다. 비가능성에 대한 자세한 내용은 유통량고정에 대한 사전처리 부분에서 다룬다.

둘째, 네트워크에서 고립되어 있는 점에 대한 처리가 있다. 최소비용문제에서 입력된 문제에 고

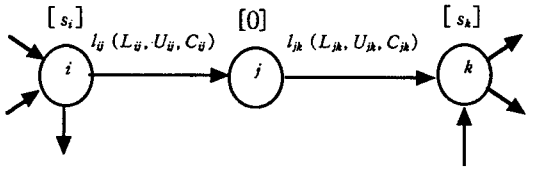
* : 본 연구는 정보통신부 '97년도 대학기초연구지원사업(97-G0683)의 지원을 받음.

립되어 있는 점이 있을 수 있다. 이때 이 점에서의 수요량 또는 공급량이 영이 아니면 이 문제는 명백히 가능해가 없으므로 비가능 판정을 하고 해법을 종료한다. 만약 이 점에서의 수요량 또는 공급량이 영이면 이 점을 제거하고 문제를 축소할 수 있다. 셋째, 모든 점이 연결되어 있지 않는 네트워크에 대한 처리가 있다. 최소비용문제에서 네트워크가 두 개 이상의 부분으로 분할되어 있는 경우가 있을 수 있다. 이 때 각 부분의 수요량과 공급량의 합이 영이 되지 않으면 명백히 가능해가 존재하지 않으므로 비가능 판정을 하고 해법 수행을 종료한다.

3. 점과 호의 제거에 대한 사전처리

본 연구에서 다루는 최소비용문제에서의 점과 호의 제거에 대한 사전처리는 중개지에 대한 사전처리, 병렬호에 대한 사전처리, 호의 유통상한과 하한을 이용한 유통량고정에 대한 사전처리, 우회경로에 대한 사전처리의 네 부분으로 구성되어 있다.

어떤 점에서의 수요량 또는 공급량이 영일 때 그 점을 중개지라고 한다([1]). 중개지로 들어오는 호와 나가는 호가 각각 하나씩 있을 때 이런 중개지는 제거를 하게되고 아울러 이 중개지에서 나가는 호도 함께 제거를 한다. 최소비용문제의 경우에는 이 중개지로 들어오는 호의 유통상한, 유통하한, 비용을 수정해야하고 이 호가 도달하는 점도 수정을 해야한다. 최소비용문제에서는 중개지를 다음과 같이 처리할 수 있다.



단, (L_{ij}, U_{ij}, C_{ij})=(호 l_{ij}의 유통하한, 호 l_{ij}의 유통상한, 호 l_{ij}의 비용)

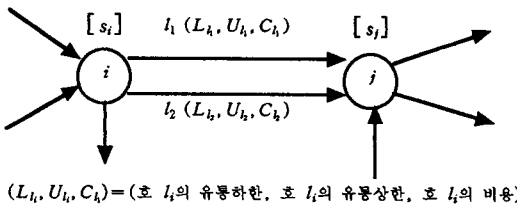
s_i= 점 i의 수요 또는 공급량

[그림 1] 중개지에 대한 사전처리 전의 네트워크

여기서 점 j는 중개지이면서 들어오는 호와 나가는 호가 각각 한 개이기 때문에 제거를 한다. 그리고 호 l_{jk}도 함께 제거를 한다. 그리고 호 l_{ij}은 다음과 같이 처리를 해준다.

- $L_{ik} = \max\{L_{ij}, L_{jk}\}$
- $U_{ik} = \min\{U_{ij}, U_{jk}\}$
- $C_{ik} = C_{ij} + C_{jk}$

어떤 호들의 시점과 종점이 각각 같은 점이면 이 호들을 병렬호라고 한다([4]). 병렬호는 입력자료 수준에서 발견되는 경우와 중개지에 대한 사전처리 후에 발생할 수 있다. 이러한 병렬호는 제거를 하게된다. 최소비용문제의 경우에는 병렬호들의 비용이 모두 같을 때에만 병렬호를 제거할 수 있고 남아 있는 호의 유통상한과 유통하한을 수정해야한다. 최소비용문제에서는 병렬호를 다음과 같이 처리할 수 있다.



단, (L_l, U_l, C_l)=(호 l_i의 유통하한, 호 l_i의 유통상한, 호 l_i의 비용)

s_i= 점 i의 수요량 또는 공급량

[그림 2] 병렬호에 대한 사전처리 전의 네트워크

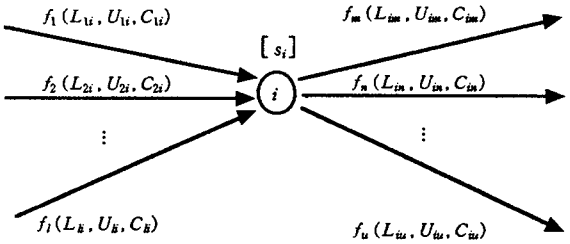
여기서 호 l_1과 호 l_2는 병렬호이다. 따라서 만약 $C_{l_1} = C_{l_2}$이면 호 l_2는 제거를 한다. 그리고 호 l_1은 다음과 같이 처리를 해준다.

- $L_l = L_{l_1} + L_{l_2}$
- $U_l = U_{l_1} + U_{l_2}$

병렬호가 생기는 경우는 입력수준에서 병렬호가 있는 경우와 중개지에 대한 사전처리를 수행하고 난 후에 병렬호가 생기는 경우가 있다. 후자의 경우에도 앞에서 설명한 방법과 동일한 방법으로 처리할 수 있다.

최소비용문제에서 어떤 점으로 들어오는 모든 호들의 유통상한과 유통하한, 그리고 나가는 모든 호들의 유통상한과 유통하한을 보고 그 호들의 유통량을 고정할 수 있는지를 검사할 수 있고 또한 비가능성을 판정할 수 있다. 호의 유통상한과 유통하한을 이용한 유통량고정에 대한 사전처리는 다음의 네 경우로 분류할 수 있다. 비가능일 때는 비가능 판정을 하고 해법수행을 종료하고, 유통량을 고정할 수 있을 때에는 유통량을 고정하고 해당하는 점과 호들을 제거한다.

[그림 3]과 같이 임의의 점 i를 고려할 때 다음의 네 가지 경우로 분류할 수 있다.



단, (L_{ij}, U_{ij}, C_{ij})=(호 l_{ij}의 용량하한, 호 l_{ij}의 용량상한, 호 l_{ij}의 비용)

s_i= 점 i의 수요 또는 공급량

f_j= 호 j의 유량 (j=1,2,…, u)

[그림 3] 유통량고정에 대한 사전처리 전의 네트워크

[경우1]: 점 i로 들어오는 모든 호들의 유통하한의 합 + 점 i의 수요량 또는 공급량 > 점 i에서 나가는 모든 호들의 유통상한의 합

$$\sum_{j=1}^{l} L_{ji} + s_i > \sum_{j=m}^{n} U_{ij} \qquad (1)$$

[경우2]: 점 i로 들어오는 모든 호들의 유통상한의 합 + 점 i의 수요량 또는 공급량 < 점 i에서 나가

는 모든 호들의 유통하한의 합

$$\sum_{j=1}^{l} U_{ji} + s_i < \sum_{j=m}^{n} L_{ij} \qquad (2)$$

[경우3]: 점 i로 들어오는 모든 호들의 유통하한의 합 + 점 i의 수요량 또는 공급량 = 점 i에서 나가는 모든 호들의 용량상한의 합

$$\sum_{j=1}^{l} L_{ji} + s_i = \sum_{j=m}^{n} U_{ij} \qquad (3)$$

[경우4]: 점 i로 들어오는 모든 호들의 유통상한의 합 + 점 i의 공급량 = 점 i에서 나가는 모든 호들의 유통하한의 합

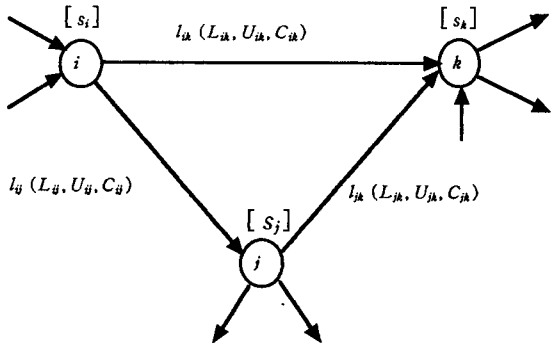
$$\sum_{j=1}^{l} U_{ji} + s_i = \sum_{j=m}^{n} L_{ij} \qquad (4)$$

[경우1]과 [경우2]일 때는 각 호에 흐르는 유통은 어떠한 경우에도 가능해를 가질 수 없다. 따라서 이 경우에는 비가능 판정을 하고 해법수행을 종료한다.

[경우3]과 [경우4]일 때는 유통을 고정할 수 있다. 왜냐하면 [경우3]일 때는 호 i로 들어오는 모든 호들은 유통하한 만큼의 유통만 흐르고, 호 i에서 나가는 모든 호들은 유통상한 만큼의 유통이 흐를 때에만 가능해가 존재할 수 있기 때문이고, [경우4]일 때는 호 i로 들어오는 모든 호들은 유통상한 만큼의 유통이 흐르고, 호 i에서 나가는 모든 호들은 유통하한 만큼의 유통만 흐를 때에만 가능해가 존재 할 수 있기 때문이다. 따라서 [경우3]일 때는 호 i로 들어오는 모든 호들은 유통을 유통하한으로 고정하고, 호 i에서 나가는 모든 호들은 유통을 유통상한으로 고정할 수 있다. 그리고 [경우4]일 때는 호 i로 들어오는 모든 호들은 유통을 유통상한으로 고정하고, 호 i에서 나가는 모든 호들은 유통을 유통하한으로 고정한다. 유통이 고정된 모든 호들을 제거하게 되면 네트워크가 끊어지게 되어 해법을 수행할 수 없게 된다. 따라서 점 i로 들어오는 호와 점 i에서 나가는 호 하나씩을 남겨 두고 유량이 고정된 모든 호들을 제거한다. 그리고 가능해가 존재할 수 있도록 유지하면서 원문제와 동등한 문제로 만들어 주기 위해서 유통이 고정된 각 호들의 시점과 종점의 수요량 또는 공급량을 수정해 주어야 한다. 그리고 고정된 유통에 대한 비용을 미리 계산하여 두었다가 해의 복원 과정에서 목적함수 값에 더해 주어야 한다.

그리고 유통을 고정할 수 있는 경우인 즉 식(3)이나 식(4)를 만족하는 특수한 경우에 해당하는 것으로 임의의 점 j로 들어오는 호들만 있는 경우나 또는 점 j에서 나가는 호들만 있는 경우를 생각할 수 있다. 점 j에서 나가는 호들만 있고 이 호들의 유통상한과 하한이 식(3)이나 식(4)를 만족할 경우 점 j에서 나가는 모든 호들의 유통을 고정하고 점 j와 함께 유통이 고정된 모든 호들을 제거한다. 그리고 점 j로 들어오는 호들만 있고 이 호들의 유통상한과 하한이 식(3)이나 식(4)를 만족할 경우 점 j로 들어오는 모든 호들의 유량을 고정하고 점 j와 함께 유통이 고정된 모든 호들을 제거한다. 그리고 이러한 호의 유통상한과 하한을 이용한 유통량고정을 하고 나면 각 점에서의 공급량이 변하게 되고 또한 각 점으로 들어오는 호의 수와 각 점에서 나가는 호의 수가 변하게 되어 제거가 가능한 중개점이 생길 수 있다. 이 경우 중개지에 대한 사전처리에 의해서 처리할 수 있다.

최소비용문제에서 어떤 한 점에서 또 다른 어떤 한 점으로 가는 두 경로가 있다면 이 두 경로 중에서 명백히 비용이 높은 경로가 있고, 비용이 낮은 경로의 유통상한과 유통하한이 유통량을 제약하지 않는다면 비용이 높은 경로를 제거한다. 최소비용문제에서는 우회경로를 다음과 같이 처리할 수 있다.



단, (L_{ij}, U_{ij}, C_{ij})=(호 l_{ij}의 유통하한, 호 l_{ij}의 유통상한, 호 l_{ij}의 비용)

s_i= 점 i의 수요량 또는 공급량

[그림 4] 우회경로에 대한 사전처리 전의 네트워크

[그림 4]에서와 같이 점 i에서 점 k로 가는 경로는 $i \rightarrow k$, $i \rightarrow j \rightarrow k$의 두 경로가 있다. 여기서 점 j로 들어오는 호가 한 개만 있을 때 C_{ik}와 $C_{ij}+C_{jk}$를 비교하여 비용이 큰 쪽의 경로를 제거한다. 즉 $C_{ik} > C_{ij}+C_{jk}$이면 호 l_{ik}를 제거하고, $C_{ik} < C_{ij}+C_{jk}$이면 호 l_{jk}를 제거한다. 이때 제거되지 않는 경로에 속한 호의 유통상한은 전체네트워크의 공급량을 흘려줄 수 있을 만큼 커야하고 제거되는 호의 유통하한은 영이라야 한다.

4. 실험

앞에서 설명한 사전처리의 성능을 평가하기 위해 네트워크 단체법 프로그램에 구현하여 실험하였다. 사전처리를 적용하지 않았을 때를 비교대상으로 삼았다. 푸는 문제는 NETGEN으로 만든 DIMACS형태의 문제 30개를 대상으로 삼았다. 그리고 [표 7.1]은 그 실험결과를 나타낸 표이다. 네트워크 단체법 프로그램 NETSIMP version 1.1로 컴파일 옵션 GCC -lm -O3을 사용하여 SUN Ultra 170에서 실험하였다.

점의 개수에 대해 호의 개수가 1배에서 2.4배 정도인 문제에서 중개지를 제거할 수 있었고 점의

개수에 대해 호의 개수가 희소할수록 많은 중개지를 제거할 수 있었다. 그리고 수요지와 공급지의 수가 작을수록 많은 중개지를 제거할 수 있었다. 일반적으로 점에 대해 호가 희소할수록 많은 중개지가 제거되고 해법의 수행시간도 비례하여 줄어들었고, 수요지와 공급지의 수가 작을수록 많은 중개지가 제거되고 해법의 수행시간도 비례하여 줄어들었다 (stndrd36, mid*). 그리고 점의 개수에 대해 호의 개수가 희소하지 않더라도 수요지로 들어오는 호들의 유통하한의 합과 수요량이 같은 경우에 이러한 모든 호들의 유통량이 고정되고 그 수요지와 유통량이 고정된 모든 호들이 제거되었다 (stndrd4*, stndrd5*, flow*). 그리고 몇 개씩의 우회경로에 제거된다 (stndrd1*, stndrd2*, stndrd3*, big1, ...). 점의 개수에 대해 호의 개수가 희소할수록 제거되는 점과 호의 수가 많고 수행시간도 많이 줄어들게 된다.

DIMACS형태의 문제 30개에 대해 수행하여 수행시간을 16%정도 줄일 수 있다는 것을 실험적으로 보였다.

[표 1] 실험문제의 정보

문제이름	(1)	(2)	문제이름	(1)	(2)	문제이름	(1)	(2)
stndrd16	400	1306	stndrd46	8000	35000	mid2015	9000	18000
stndrd18	400	1306	stndrd47	8000	35000	mid2017	13000	26000
stndrd28	1000	2900	stndrd48	3000	15000	mid2020	20000	40000
stndrd29	1000	3400	stndrd54	1400	12000	mid2021	20000	40000
stndrd30	1000	4400	big1	25000	120000	mid2022	20000	40000
stndrd33	1500	4385	big3	2000	170000	mid2034	10000	20000
stndrd34	1500	5107	big5	5000	80000	flow0515	5000	30000
stndrd36	8000	15000	big6	5000	60000	flow0516	5000	35000
stndrd37	5000	23000	big7	5000	40000	flow0518	5000	45000
stndrd42	10000	30000	mid2014	7000	14000	flow0531	5000	20000

(1): 점의 개수
(2): 호의 개수

[표 2] 사전처리 실험결과

문제이름	(3)	(4)	사전처리 안한 경우			사전처리 한 경우		
			(5)	(6)	(7)	(8)	(9)	(10)
stndrd16	0	1	0.01	0.04	0.05	0.01	0.03	0.04
stndrd18	0	1	0.00	0.04	0.04	0.00	0.04	0.04
stndrd28	0	2	0.00	0.16	0.16	0.00	0.15	0.15
stndrd29	0	2	0.01	0.19	0.20	0.01	0.15	0.16
stndrd30	0	2	0.01	0.20	0.21	0.01	0.19	0.20
stndrd33	0	1	0.01	0.44	0.45	0.01	0.37	0.38
stndrd34	0	2	0.01	0.44	0.45	0.01	0.35	0.36
stndrd36	788	790	0.02	9.52	9.54	0.08	7.92	8.00
stndrd37	0	3	0.03	6.08	6.11	0.08	5.00	5.08
stndrd42	32	32	0.05	21.95	22.00	0.10	19.27	19.37
stndrd46	563	4435	0.02	42.89	42.91	0.14	30.08	30.22
stndrd47	540	4222	0.03	37.24	37.27	0.10	33.56	33.66
stndrd48	440	3094	0.02	3.06	3.08	0.03	2.40	2.43
stndrd54	97	1533	0.01	0.88	0.89	0.02	0.75	0.77
big1	0	1	0.12	268.28	268.40	0.57	254.02	254.59
big3	121	20543	0.14	5.37	5.51	0.61	4.43	5.04
big5	117	2360	0.07	16.32	16.39	0.24	14.48	14.72
big6	110	1649	0.05	13.31	13.36	0.20	12.40	12.60
big7	106	1044	0.04	11.00	11.04	0.13	10.22	10.35
mid2014	776	777	0.02	4.15	4.17	0.07	3.45	3.52
mid2015	1064	1065	0.02	6.47	6.49	0.09	5.31	5.40
mid2017	1572	1574	0.03	15.54	15.57	0.17	10.60	10.77
mid2020	2041	2763	0.05	54.05	54.10	0.31	38.63	38.94
mid2021	2034	2791	0.04	42.47	42.51	0.33	32.83	33.16
mid2022	1977	2590	0.03	34.25	34.28	0.34	28.95	29.29
mid2034	851	1610	0.03	11.44	11.47	0.12	9.47	9.59
flow0515	359	3050	0.02	6.44	6.46	0.10	4.52	4.62
flow0516	375	3612	0.04	7.03	7.07	0.14	5.17	5.33
flow0518	351	4329	0.04	7.73	7.77	0.16	6.36	6.52
flow0531	307	1767	0.01	4.20	4.21	0.06	3.03	3.09

(3): 제거된 점의 개수
(4): 제거된 호의 개수
(5): star-form 자료구조 저장 시간 (단위:초)
(6): 사전처리 안 했을 때 해법수행시간 (단위:초)
(7): 사전처리 안 했을 때 총수행시간 (단위:초)
(8): 사전처리 수행시간 (단위:초)
(9): 사전처리 했을 때 해법수행시간 (단위:초)
(10): 사전처리 했을 때 총수행시간 (단위:초)

5. 결론

본 연구에서는 최소비용문제에 적용할 수 있는 사전처리 기법의 이론과 그 구현에 대한 연구 및 실험결과의 분석을 행하였다.

최소비용문제에서의 사전처리기법으로 중개지에 대한 사전처리, 병렬호에 대한 사전처리, 호의 유통상한과 유통하한을 이용한 유통량고정에 대한 사전처리, 우회경로에 대한 사전처리 등을 구현하여 이러한 사전처리를 하지 않았을 때와 비교 실험을 하였다. 그 결과 문제를 축소하고 수행시간을 16%정도 줄이는 효과가 있다는 것을 보였다.

최소비용문제에서의 사전처리 기법으로 문제 축소와 입력된 문제의 정확성 검사 등을 통해 해법의 수행도와 안정성을 높이는 효과를 가져다준다. 더욱이 네트워크 프로그램이 다른 해법의 서브루틴으로 사용될 경우가 있는데 이러한 경우에 네트워크 문제를 여러 번 반복해서 풀어야 할 경우가 생기는데 이 때 사전처리에 의해서 점과 호를 제거하여 축소된 문제를 반복해서 풀게 되면 수행시간을 줄이는 측면에서 더욱 큰 효과를 얻게 된다.

6. 참고문헌

[1] 박순달, *경영과학*, 제3판, 민영사, 1998

[2] 성명기, 박순달 대형선형계획문제의 사전처리, *Proceedings of KORS/MS '96 Autumn Conference*, 285-288, 1996

[3] 박순달, 김우제, 박찬규, 임성묵, "단체법 프로그램 LPAKO 개발에 관한 연구", *경영과학 제15권 제1호*, 1998

[4] Ahuja Ravindra K., Thomas L. Magnanti, James B. Orlin, *Network Flows theory, algorithms, and applications*, Prentice-Hall, 1993

[5] Andersen Erling D., Knud D. Andersen, "Presolving in Linear Progrmming", *Mathematical Programming 71*, 221-245, 1995

[6] Brearley, A. L., G. Mitra, H. P. Williams, "Analysis of Mathematical Progamming Problems Prior to Applying the Simplex Algorithm", *Mathematical Programming 8*, 54-83, 1975

[7] Gondzio Jacek , "Presolve Analysis of Linear Progrmming Prior to Applying an Interior Point Method", Technical Report, 1994

[8] Savelsbergh, M. W. P., "Preprocessing and Probing Techniques for Mixed Integer Programming Problems", *ORSA Journal on Computing 6*, 445-454, 1994