韓國經營科學會誌
第23卷 第4號
1998年 12月


# 하한을 이용한 효율적인 최소차수순서화*

박찬규** · 박순달**

# An Efficient Minimum Degree Ordering Method Using the Lower Bounds of Degrees*

Chan-Kyoo Park** · Soondal Park**

■ Abstract ■

Ordering is used to reduce the amount of fill-ins in the Cholesky factor of an symmetric definite matrix. One of most efficient ordering methods is the minimum degree ordering method. In this paper, we propose the two techniques to improve the performance of the minimum degree ordering which are implemented using clique storage structure. One is node absorption which is a generalized version of clique absorption. The other technique is using the lower bounds of degree to suspend the degree updates of nodes. Finally, we provide computational results on the problems on NETLIB. These results show that the proposed techniques reduce the number of degree updates and the computational time considerably.

## 1. 서 론

선형계획해법으로 최근에 많이 사용되는 아핀법(affine method), 장벽법(barrier method) 등의 내부점 선형계획법에서는 $A\Theta A^T$ ( $A$는 제약식 행렬, $\Theta$는 대각원소가 양인 대각행렬) 형태의 대칭양정치 (symmetric positive definite) 행렬의 선형방정식을 푸는 과정이 필요하다. 일반적으로 대형선형계획법 문제의 제약식 행렬은 희소하다는 특성을 가지는데 이 경우 내부점 선형계획법에서 나타나는 대칭양정치 행렬도 역시 희소행렬이 된다. 내부점 선형계획법에서는 전체 소요 시간의 많은 부분이 대칭양정치 선형시스템을 푸는 데에 소요되므로 효율적으로

* 본 연구는 정통부 '97대학기초연구지원사업(97-G0683)의 지원을 받음.
** 서울대학교 산업공학과

이 선형방정식을 풀어내는 것이 내부점 선형계획법 프로그램의 성능을 크게 좌우한다[7].p

대칭양정치 선형시스템을 푸는 방법으로 주로 촐레스키분해(Cholesky factorization)를 사용한다. 촐레스키분해는 $A\Theta A^T = LL^T$(단, $L$은 하삼각행렬)로 분해하는 것으로 이 때 $L$을 촐레스키 인자(Cholesky factor)라 한다. 희소행렬에 대하여 촐레스키분해를 수행할 때에는 촐레스키인자의 비영요소(nonzero) 수를 줄이기 위하여 순서화(ordering) 과정을 거친다. 추가되는 비영요소의 수를 최소화하는 행순서를 구하는 문제는 NP-Hard로 알려져 있어[12], 실제 사용되는 발견적 순서화 방법으로는 최소차수순서화(minimum degree ordering)와 최소부족수순서화(minimum deficiency ordering) 등이 있는데[11][9], 특히 최소차수순서화 방법이 수행 속도가 빨라 많이 사용되고 있다[6]. 반면, 최소부족수순서화 방법은 계산량이 많으나 촐레스키인자에 추가되는 비영요소의 수가 작다는 장점이 있어 계산량을 줄이기 위한 연구가 수행되고 있다[1].

최소차수순서화 방법을 구현하는데 사용되는 자료구조로 George 등이 제안한 쿼션트 그래프(quotient graph)[3][5][6]와 Speelpenning이 제안한 클릭저장구조[10]가 있다. 또한 최소차수순서화 방법의 수행속도를 향상시키기 위한 방법으로 구별불능점(indistinguishable node), 복수삭제(multiple elimination), 외부차수(external degree), 부분 차수수정(incomplete degree update) 등의 기법들이 제안되었고[5][6][8], 특별히 클릭저장구조에서 사용될 수 있는 클릭흡수(clique absorption) 기법이 제안되었다[8]. 대부분의 내부점 선형계획법 프로그램에서는 쿼션트 그래프를 이용한 최소차수순서화 방법을 사용하고 있다.

본 연구는 클릭저장구조를 사용한 최소차수순서화 방법에서 기존에 제안된 클릭흡수를 보다 일반화한 점흡수 방법과 차수의 하한을 이용한 차수 수정 회수를 줄이는 방법을 제안한다. 아울러 실험을 통해 그 효율성을 확인하고 클릭저장구조를 사용한 최소차수순서화 방법의 수행도가 쿼션트 그래프를 이용한 최소차수순서화 방법과 경쟁성이 있음을 보인다.

## 2. 최소차수순서화와 클릭저장구조

최소차수순서화 방법(minimum degree ordering)은 매 회마다 최소의 차수를 가지는 점을 찾아서 삭제하는 방법으로 마코비츠 순서화 방법을 대칭행렬에 적용한 방법으로 볼 수 있다[3]. 최소차수순서화 방법은 일종의 발견적 기법으로서, 최소차수순서화 방법에 대한 자세한 설명은 [5]과 [6]를 참조하기 바란다.

m×m 대칭양정치 행렬 $M$에 관련된 그래프를 $G=(N, E)$ 라고 하자. $N$은 $M$의 행(또는 열)에 해당하고 $E$는 $M$의 비영요소 위치를 나타낸다. 즉, $M$의 $(i, j)$번째 요소 $M_{ij}\neq 0$ 이면 호 $(i, j)$가 $E$에 들어 있다. 그래프 $G$에서 점 $i$의 인접점 집합 $Adj_G(i)$와 점 $i$의 차수 $\deg_G(i)$를 다음과 같이 정의한다.

**정의 1. 인접점 집합과 차수**

$$Adj_G(i) = \{j \mid (i, j) \in E\}.$$

$$\deg_G(i) = |Adj_G(i)|.$$

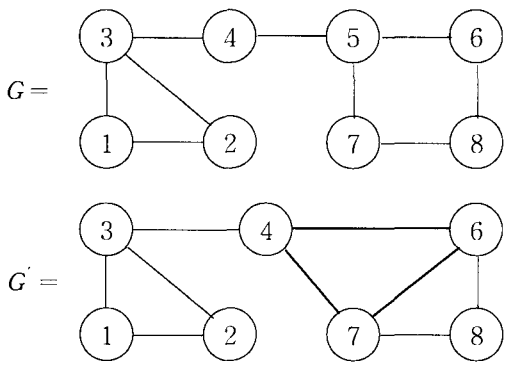
최소차수순서화 방법은 최소차수인 점을 찾아 이를 삭제하고 변형된 그래프에서 각 점의 차수를 수정하여 다시 차수가 최소인 점을 찾아 삭제하는 과정을 반복하게 된다. 순서화 과정에서 점이 삭제되면 그래프도 같이 바뀌게 되는데 이러한 일련의 그래프를 삭제그래프(elimination graph)라 부른다. $i$번째 점으로 $y$가 삭제될 때 삭제그래프 $G=$

$(N_i, E_i)$는 다음과 같이 구해진다.

$N_i \leftarrow N_{i-1} - \{y\}$

$E_i \leftarrow ((E_{i-1} - \{(u,y) | (u,y) \in E_{i-1}\}) \cup$
$\{(u,v) \notin E_{i-1} | (u,y) \in E_{i-1},$
$(v,y) \in E_{i-1}\}$
단, $G_0 = (N_0, E_0)$는 $G = (N, E)$와 같다.

예를 들어, 아래의 그래프 $G$에서 점 ⑤를 삭제하게 되면 삭제그래프 $G'$과 같이 된다.




—는 원래 있던 호, —는 새로 추가된 호

[그림 1] 점의 삭제와 삭제그래프의 변형

최소차수순서화 방법에서 가장 많은 계산시간을 차지하는 부분은 삭제그래프를 변형하고 차수를 수정하는 부분이다. 삭제그래프를 표현하기 위한 방법 중의 하나인 클릭저장구조를 살펴보자.

클릭저장구조는 쿼션트 그래프 모형과 같이 삭제그래프를 보다 효율적으로 저장하기 위한 자료구조이다. 클릭저장구조에서는 삭제그래프를 클릭으로 이루어진 부분 그래프로 쪼개어 저장하고 삭제그래프의 수정은 클릭들의 병합(merge)에 의해 수행된다.

무방향 그래프 $G = (N, E)$에 대해서, $N' \subset N$이고 $E' \subset E$인 그래프 $H = (N', E')$를 $G$의 부분그래프(subgraph)라고 한다. 그리고, 점집합이 $N' \subset N$이면서 호집합 $E(N') = \{(u,v) \in E | u \in N', v \in N'\}$인 그래프 $G(N') = (N', E(N'))$를 지역그래프(section graph)라고 한다. 지역그래프는 부분그래프의 일종임을 알 수 있다.

**정의 2. 클릭**

무방향 그래프 $C$의 모든 두 점 사이에 호가 있으면 $C$를 클릭(clique)이라 한다.

점집합으로 $V$를 가진 클릭을 $C(V)$으로 표시하자. 또한 클릭 $C$의 점집합을 $n(C)$, 호집합을 $e(C)$로 나타내기로 하고 본 연구에서 다루는 모든 그래프는 무방향 그래프이다.

**정의 3. 클릭 커버(clique cover)**

클릭 $C_1, C_2, \cdots, C_r$가 다음의 조건을 만족하면, $K = \{C_1, C_2, \cdots, C_r\}$를 그래프 $G = (N, E)$의 클릭커버라 하고 $r$를 클릭커버 $K$의 크기($|K|$로 나타내기로 함)라 한다.

(i) $n(C_1) \cup n(C_2) \cup \cdots \cup n(C_r) = N$,

(ii) $e(C_1) \cup e(C_2) \cup \cdots \cup e(C_r) = E$

$G$의 클릭커버 $K = \{C_1, C_2, \cdots, C_r\}$가 있을 때 클릭커버 $K$로부터 점 $u$의 인접점 집합 $Adj_G(u)$는 다음의 정리에 의해 구할 수 있다.

**정리 1.**

그래프 $G$의 클릭커버 $K$의 클릭 중에서 점집합에 점 $u$가 포함된 클릭들이 $C_1^u, C_2^u, \cdots, C_k^u$라 하면

$$Adj_G(u) = n(C_1^u) \cup \cdots \cup n(C_k^u) - \{u\}$$

이다.

(증명) 클릭과 클릭커버의 정의로부터 $u$에 연결된 호는 $u$를 포함하는 클릭에만 있게 된다. 또한 클릭 $C_i^u$에는 $v \in n(C_i^u) - \{u\}$인 모든 점 $v$와 $u$간의 호 $(u,v)$가 있으므로

$$Adj_G(u) = (n(C_1^u) - \{u\}) \cup \cdots \cup (n(C_k^u) - \{u\})$$
$$= n(C_1^u) \cup \cdots \cup n(C_k^u) - \{u\}$$

이다. □

삭제그래프에서 점을 삭제하고 수정된 삭제그래프가 만들어지는 과정을 클릭커버와 관련지어 생각해 보자. $G$의 클릭커버 $K=\{C_1, C_2, \cdots, C_r\}$가 있고, 여기에서 점 $i$가 포함된 클릭들을 $C_1^i$, $C_2^i$, $\cdots$, $C_k^i$ 라고 하자. 그러면 $G$에서 점 $i$가 삭제되고 난 후의 수정된 삭제그래프 $G'$의 클릭커버 $K'$는 다음과 같이 구할 수 있다. 점집합 $V$를 $V=n(C_1^i) \cup \cdots \cup n(C_k^i) - \{i\}$라 하고 점집합을 $V$로 갖는 클릭을 $C(V)$라 하자.

$$K' = (K - \{C_1^i, C_2^i, \cdots, C_k^i\}) \cup \{C(V)\}$$

즉, 삭제점 $i$를 포함하고 있는 클릭들을 $K$에서 삭제하고 점 $i$가 포함되어 있던 클릭들의 점집합을 하나로 모아 큰 클릭을 만들어 이를 클릭커버에 추가하는 것이다. 이 과정을 클릭합병(clique merge)이라 부른다[6].

내부점 선형계획법 문제에서는 순서화에 들어가기 전에 먼저 행렬 $M=A\Theta A^T$를 나타내는 관련그래프 $G=(N, E)$의 클릭커버를 결정하여야 한다. 다음의 정리는 행렬 $A$로부터 $G$의 클릭커버를 쉽게 생성할 수 있게 해준다.

**정리 2[2].**

행렬 $A$의 $s$번째 열 $A_{\cdot s}$의 비영요소들의 행번호 집합이 $N'=\{r_1, r_2, \cdots, r_l\}$이면 $G$의 지역그래프 $G(N')$은 클릭이다.

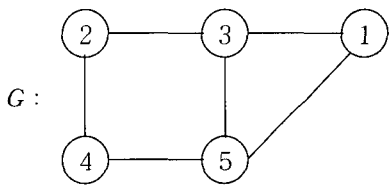
예를 들어 다음과 같이 행렬 $A$의 비영구조가 주어졌다고 하자.

$$A = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & * & & & & & \\ 2 & & * & & & & * \\ 3 & * & * & & * & & \\ 4 & & & * & & * & * \\ 5 & * & & & * & * & \end{array}$$

[그림 2] A행렬의 비영 구조

이에 해당하는 $M=A\Theta A^T$의 구조와 관련 그래프는 다음과 같다.

$$M = \begin{vmatrix} 1 & & * & & * \\ & 2 & * & * & \\ * & * & 3 & & * \\ & * & & 4 & * \\ * & & * & * & 5 \end{vmatrix}$$




[그림 3] 행렬 $M$의 비영 구조와 관련그래프 $G$

## 3. 클릭홉수의 확장

클릭모형에서는 그래프 $G$의 클릭커버의 크기가 작을수록 순서화에 유리하다. 왜냐하면 클릭커버의 크기가 작으면 차수 계산이나 삭제그래프 수정에 관계되는 클릭들의 개수도 그 만큼 작을 것이기 때문이다. 클릭의 홉수란 클릭커버의 크기를 줄이기 위해서 중복되는 클릭을 제거하는 것을 말한다. 다음의 정리는 클릭의 홉수가 가능한 이유를 제시한다.

**정리 3[6].**

그래프 $G$의 클릭커버 $K=\{C_1, C_2, \cdots, C_r\}$에

서 $n(C_i) \subset n(C_j)$ $(1 \le i, j \le r, i \ne j)$인 $i$와 $j$가 존재하면 $K' = \{C_1, C_2, \cdots, C_r\} - \{C_i\}$도 $G$의 클릭커버이고 $|K'| = |K| - 1$이다.

그래프를 최소 크기의 클릭커버로 표현하는 문제는 NP-hard로 알려져 있는데 단순히 클릭의 홉수만으로는 최소 크기의 클릭커버를 찾을 수 없다. 예를 들어 가령 점집합이 각각 $n(C_1) = \{1,2\}$, $n(C_2) = \{2,3\}$, $n(C_3) = \{1,3\}$인 3개의 클릭은 점집합이 $n(C) = \{1,2,3\}$인 하나의 클릭으로 표현할 수 있다. 그러나 정리 3은 이러한 경우에 대해서는 설명해 주지 못하고 있다.

클릭 홉수는 2개의 클릭간에 클릭 홉수가 일어날 수 있는가를 판단할 수 있게 해준다. 이를 보다 일반화하여 다음과 같은 점홉수의 개념으로 확장할 수 있다.

**정리 4.**

$G$의 클릭커버 $K = \{C_1, C_2, \cdots, C_r\}$에서 점 $u$가 포함된 클릭들이 $C_1^u, C_2^u, \cdots, C_k^u$라고 하자. 클릭 $C_j^u (1 \le j \le k)$에 대해 $I \subset \{1,2,\cdots,k\}$이고 $j \notin I$, $e(C_j^u) \subset \bigcup_{i \in I} e(C_i^u)$인 $I$가 존재하면,

$$K' = (K - \{C_j^u\}) \bigcup \{C(V)\}$$

(단, $V = n(C_j^u) - \{u\}$이고 $C(V)$는 $V$을 점집합으로 갖는 클릭이다.)

인 $K'$은 $G$의 클릭커버이다.

(증명) 임의의 점 $v$가 $v \in n(C_j^u)$이면 호 $(v,u)$가 $G$에 존재하게 된다. 정리의 가정을 만족하면 $C_j^u$에서 $u$와 $u$에 연결된 호를 제거하여도 호 $(v,u)$가 항상 다른 클릭에 포함되어 있음을 보이면 된다.

조건을 만족하는 $I$가 존재하면 $e(C_j^u) \subset \bigcup_{i \in I} e(C_i^u)$이고 $v \in \bigcup_{i \in I} n(C_i^u)$이다. 이는 $v \in C_t^u$ $(t \in I)$인 클릭 $C_t^u$가 항상 존재함을 의미한다. $C_t^u$는 호 $(u,v)$를 포함하고 있게 된다. 따라서 $C_j^u$에서 점 $u$를 제거하더라도 호 $(u,v)$는 다른 클릭 $C_t^u$에 포함되어 있으므로 $K'$은 클릭커버가 된다. □

정리 4는 점 $u$가 포함되는 클릭의 수를 줄일 수 있는 방법을 제시하고 있다. 정리 3에 의해 일어나는 모든 클릭 홉수는 정리 4에 의해 점홉수 형태로 단계적으로 일어남을 알 수 있다. 즉, 정리 3은 정리 4의 특수한 경우임을 알 수 있다.

## 4. 차수 하한을 이용한 차수수정의 효율화

클릭저장구조는 삭제그래프를 클릭커버에 의해 표현하는데 클릭커버로부터 각 점의 차수의 상한과 하한을 다음의 정리에서와 같이 쉽게 구할 수 있다.

**정리 5. 차수의 상·하한**

그래프 $G$의 클릭커버 $K$에서 점 $u$가 포함된 클릭들이 $C_1^u, C_2^u, \cdots, C_k^u$ (단, $k \ge 1$) 이면,

$$\max_{1 \le i \le k}(|n(C_i^u)| - 1) \le \deg_G(u) \le \sum_{i=1}^{k}(|n(C_i^u)| - 1)$$

이다.

(증명) 정리 1에서 $1 \le i \le k$인 모든 $i$에 대해 $(n(C_i^u) - \{u\}) \subset Adj_G(u)$이므로 $\deg_G(u) = |Adj_G(u)| \ge |n(C_i^u) - \{u\}|$이다. 즉 $\deg_G(u) \ge \max_{1 \le i \le k}(|n(C_i^u)| - 1)$이다. 또한 정리 1로부터 $|Adj_G(u)| = |(n(C_1^u) - \{u\}) \bigcup \cdots \bigcup (n(C_k^u) - \{u\})| \le \sum_{i=1}^{k}(|n(C_i^u)| - 1)$이다. □

삭제그래프 $G_i=(N_i, E_i)$에서 점 $u$와 $v$가 다음과 같은 관계식을 만족하면 구별불능점(indistinguishable node) 관계에 있다고 말한다.

$$Adj_{G_i}(u)\bigcup\{u\}=Adj_{G_i}(v)\bigcup\{v\}$$

삭제그래프 $G_i=(N_i, E_i)$에서 구별불능점 관계에 있는 점들은 그 점들이 삭제되기 전까지 계속 구별불능점 관계로 남아 있으므로[5], 구별불능점들을 하나의 점인 것처럼 처리하면 최소차수순서화 방법의 수행도를 향상시킬 수 있다. 구별불능점 관계에 있는 점들을 동시에 삭제하는 경우 차수의 하한은 다음과 같다.

**정리 6. 차수의 하한**

삭제그래프 $G_{i-1}=(N_{i-1}, E_{i-1})$에서 구별불능점 관계에 있는 점 $x_0, \cdots, x_k$(단, $k\geq 0$)가 동시에 삭제되어 삭제그래프 $G_i=(N_i, E_i)$로 변환된다고 하면 $u\in Adj_{G_{i-1}}(x_0)$이고 $u\in N_i$인 모든 점 $u$에 대해,

$$\deg_{G_i}(u)\geq \deg_{G_{i-1}}(u)-(k+1)$$

이다.

(증명)

$$\begin{aligned}
&|\deg_{G_i}(u)|=|Adj_{G_i}(u)|\\
&=|Adj_{G_{i-1}}(u)\bigcup Adj_{G_{i-1}}(x_0)-\{x_0,\cdots,x_k\}|\\
&=|Adj_{G_{i-1}}(u)\bigcup Adj_{G_{i-1}}(x_0)|-|\{x_0,x_1,\cdots,x_k\}|\\
&\geq |Adj_{G_{i-1}}(u)|-(k+1)\\
&=|\deg_{G_{i-1}}(u)|-(k+1)
\end{aligned}$$ □

최소차수순서화 방법에서는 항상 삭제점에 인접한 점들에 대해서만 차수를 수정한다. 클릭저장구조에서는 차수의 하한을 쉽게 구할 수 있으므로 삭제그래프가 바뀜에 따라 차수를 수정할 때 차수하한을 이용하여 차수를 수정해야 하는 점의 수를 줄일 수 있다.

최소차수순서화 방법의 $i$번째 단계에서의 삭제그래프 $G_{i-1}=(N_{i-1}, E_{i-1})$에서 최소차수를 가지고 구별불능점 관계에 있는 $\{x_0, \cdots, x_k\}$(단, $k\geq 0$)가 삭제되어 만들어진 삭제그래프를 $G_i=(N_i, E_i)$라 하자. 또한 $G_i$에서 점 $u$가 포함된 클릭들의 집합을 $K_u^i$라고 하자. $\deg_{G_i}(u)$의 하한 $ldeg_{G_i}(u)$를 다음과 같이 정의한다.

**정의 4. 차수의 하한**

$$ldeg_{G_i}(u)=\begin{cases}\max\{(ldeg_{G_{i-1}}(u)-(k+1)),\\ \quad(\max_{C\in K_u^i}|n(C)|-1)\}, & \text{if } u\in Adj_{G_{i-1}}(x_0)\\ ldeg_{G_{i-1}}(u), & \text{if } u\notin Adj_{G_{i-1}}(x_0)\end{cases}$$

단, 모든 점 $u\in N_0$에 대해 $ldeg_{G_0}(u)=\deg_{G_0}(u)$이다.

**정리 7. 차수의 하한**

모든 삭제그래프 $G_i=(N_i, E_i)$에서

$$\deg_{G_i}(u)\geq ldeg_{G_i}(u),\quad \forall u\in N_i$$

이다.

(증명) 수학적 귀납법을 사용하여 증명한다. $i=1$일 때는 정리 5와 6과 $ldeg_{G_0}(u)$의 정의로부터 명백히 성립한다. $i=j$(단, $j\geq 1$) 일 때 성립한다고 가정하자. $u\in Adj_{G_{i-1}}(x_0)$이면, 정리 5와 6으로부터 $\deg_{G_i}(u)\geq \max_{C\in K_u^i}(|n(C)|-1)$이다. 또한 $\deg_{G_i}(u)\geq \deg_{G_{i-1}}(u)-(k+1)\geq ldeg_{G_{i-1}}(u)-(k+1)$이므로 $\deg_{G_i}(u)\geq ldeg_{G_i}(u)$이 성립한다. □

최소차수순서화 방법에서 차수의 하한을 이용하여 다음 단계에서 최소 차수를 갖지 않을 것으로 예상되는 점들은 차수 수정을 연기할 수 있다. 수정된 최소차수순서화 방법은 아래와 같다. 배열 mark()는 삭제되지 않은 각 점의 차수가 정확히 계

산되어 있는가를 나타낸다. 즉, mark(j)가 0이면 점 $j$의 차수가 계산되어 있지 않고 차수의 하한만이 계산되어 있음을 의미하고 1이면 차수가 계산되어 있음을 나타낸다. 또한 14행에서 $\{x_0, \cdots, x_k\}$은 점 $x_0$와 구별불능점 관계에 있는 점들의 집합을 의미한다. 행렬 $M$의 관련그래프를 $G=(N, E)$라 하고 $K_u$는 각 단계의 삭제그래프에서 점 $u$가 포함된 클릭들의 집합을 나타낸다. $\deg(u)$와 $ldeg(u)$는 각각 각 단계의 삭제그래프에서 차수와 차수의 하한을 의미한다.

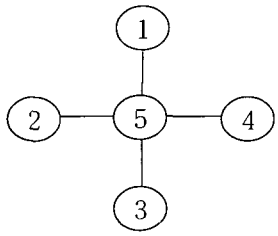
**수정된 최소차수순서화 방법**

1: $G_0 \leftarrow G=(N, E)$;
2: 모든 점 i에 대해,
  $\deg(i) \leftarrow |Adj(i)|$를 계산한다.
3: for i := 1 to m do mark(i) = 1;
4: $S \leftarrow \emptyset$;
5: do {
6:  mindeg $\leftarrow$ min{deg($j$);
    $j \in N-S$ and $mark(j)=1$};
7:  minldeg $\leftarrow$ min{$ldeg(j)$;
    $j \in N-S$ and $mark(j)=0$};
8:  if minldeg < mindeg then
9:  for $u \in \{v \mid ldeg(v) \le$ mindeg and
    $v \in N-S$},
10:  compute deg($u$);
11:  mark(u)=1;
12:  end if
13:  $x_0 \leftarrow \arg\min\{\deg(j); j \in N-S$ and
    $mark(j)=1\}$;
14:  $S \leftarrow S \cup \{x_0, x_1, \cdots, x_k\}$;
15:  $T \leftarrow Adj_{G_{i-1}}(x_0)$;
16:  construct $G_i$ from $G_{i-1}$;
17:  for each $u \in T$,
18:  if mark(u)=1 then,
19:  $ldeg(u) \leftarrow \max\{(\deg(u)-(k+1))$,
    $(\max_{C \in K_u} |n(C)|-1)\}$;
20:  mark(u) := 0;
21:  else
22:  $ldeg(u) \leftarrow \max\{(ldeg(u)-(k+1))$,
    $\max_{C \in K_u} |n(C)|-1)\}$;
23:  end if
24:  end for
25: } while( $N-S \neq \emptyset$ )

기존의 최소차수순서화 방법과 다른 점은 크게 2가지이다. 첫 번째 다른 점은 최소차수를 가진 점을 찾는 부분인데 6행~13행 부분이다. 기존의 최소차수순서화 방법에서는 항상 모든 점의 차수가 계산되어 있으므로 차수가 최소인 점을 바로 찾아 낼 수 있다. 그러나 수정된 최소차수순서화 방법에서는 일부 점들은 차수가 정확히 계산되어 있지 않고 단순히 하한만 알고 있는 상태이다. 따라서 하한이 min deg 보다 작은 점들에 대해서는 차수를 정확히 계산하여 최소차수를 가진 점을 찾는다.

두 번째 다른 점은 삭제점과 인접한 점들의 차수를 수정하는 부분으로 17행~24행까지이다. 기존의 최소차수순서화 방법에서는 점이 삭제되면 삭제점과 인접한 모든 점의 차수를 수정하지만 수정된 최소차수순서화 방법에서는 단지 차수의 하한만을 수정한다. 실제 차수의 수정이 이루어지는 부분은 8행~12행으로 최소차수의 점을 선택하는 단계이다. 현단계의 삭제그래프에서의 차수가 구해진 점은 차수를 이용하고 차수가 구해져 있지 않는 점은 차수의 하한을 이용하기 때문에 불필요한 차수 수정을 줄일 수 있게 된다. 따라서 점 $x_0$가 삭제되었을 때 차수 수정이 이루어지는 점의 수가 기존의 최소차수순서화 방법보다 작게 된다. 그러나 최악의 경우 수정된 최소차수순서화 방법도 차수가 바뀌는 점들에 대해 모두 차수 수정을 하는 경우가 있을 수 있다.

수정된 최소차수순서화 방법에서는 배열 mark()를 이용하여 일단 차수가 정확히 계산된 점의 차수의 하한은 계산된 차수로 암묵적으로 재설정함을 알 수 있다.




[그림 4] 차수하한 사용예

예를 들어 [그림 4]와 같은 삭제그래프가 주어졌다고 하자. 기존 최소차수순서화 방법에서는 ①, ②, ③, ④, ⑤ 순(단, 차수가 같을 때는 임의로 점 번호가 작은 점을 선택)으로 삭제하게 된다. 점 ①이 삭제되면 ⑤의 차수를 수정하고, ②가 삭제되면 ⑤의 차수를 수정하므로 점 ⑤의 차수는 총 4번 수정된다. 반면, 수정된 최소차수순서화 방법에서는 ①이 삭제되면 ⑤의 $ldeg(5)=3$으로 수정되고, ②가 삭제되면 $ldeg(5)=2$로 수정된다. 점 ③이 삭제되어 $ldeg(5)=1$로 바뀌면 비로소 점 ⑤의 차수 수정이 수행된다. 수정된 최소차수순서화 방법에서 점 ⑤의 차수는 총 2번 수정된다.

위의 예는 수정된 최소차수순서화 방법이 이상적으로 적용될 수 있는 단적인 예이다. 클릭저장구조에서는 쿼션트 구조에 비해 차수 계산에 많은 시간이 소요된다. 따라서 하한을 이용한 수정된 최소차수순서화 방법을 사용하면 차수 수정 회수를 줄임으로써 순서화 시간을 크게 줄일 수 있을 것이다.

## 5. 실험결과

클릭구조를 사용하여 순서화를 행하는 내부점 코드 LPABO ver 5.1f[1]에 점홉수와 차수의 하한을 이용하는 방법을 추가로 구현하여 기존의 다른 내부점 기법 프로그램의 순서화 시간과 비교 실험을 하였다. 내부점 기법 프로그램들로는 코드가 공개된 BPMPD(ver 2.13)[2], HOPDM(ver 2.13)[3]와 상용으로 널리 사용되는 CPLEX(ver 4.0.7)[4]를 선택하였다. BPMPD는 C. Meszaros 등이 개발한 내부점 기법 코드로 쿼션트 구조를 이용한 최소차수 순서화 방법과 근사 최소부족 순서화 방법을 사용하는데 이 중에서 최소차수 순서화 방법을 비교대상으로 사용하였다. HOPDM은 J. Gondzio 등이 구현한 내부점 코드로 순서화 루틴으로는 Liu가 구현한 복수삭제와 쿼션트구조를 이용한 최소차수순서화 방법[6]을 사용한다. CPLEX는 근사 최소차수순서화 방법(approximate minimum degree ordering)을 사용하는데 세부적인 구현 방법은 공개되어 있지 않고 있다. 실험 문제는 NETLIB[4]에 있는 문제와 Univ. of Iowa[5]에서 유지하는 선형계획법 문제를 사용하였다. 사용된 컴퓨터 기종은 SunSparc Ultra 170(주기억 용량 64MB)이고 BPMPD와 HOPDM는 각각 Fortran 컴파일러로 최적화 옵션 '-fast -O3'으로 컴파일 하였고 LPABO는 GCC 컴파일러로 최적화 옵션 '-O3'를 주어 컴파일 하였다. 각 프로그램에서는 순서화에 들어가기 전에 사전처리를 수행하는데 순서화 시간을 보다 정확하게 비교하기 위해 빈 행과 열을 처리하는 이외의 행과 열을 제거하는 사전처리는 수행하지 않게 하였다. 실험결과는 <표 1>에 정리하였다.

LPABO ver 5.1f에서는 클릭저장구조를 사용한 최소차수순서화 방법을 사용하고 있으며 클릭홉수, 구별불능점, 외부 차수, 복수삭제 기법 등이 사용되고 있다. 아래 표의 방법 1에서는 LPABO ver

1) 서울대학교 산업공학과 경영과학 연구실에서 개발한 내부점 선형계획법 프로그램으로 "http://orlab.snu.ac.kr/software/index.html"에서 얻을 수 있음.
2) http://ecolu-info.unige.ch/~logilab/gondzio/
3) http://www.sztaki.hu/~meszaros/bpmpd/
4) http://www.cplex.com
5) ftp://col.biz.uiowa.edu/pub/testprob/lp/

5.1f에 점흡수 기법을 추가로 구현한 방법이고, 방법 2는 LPABO ver 5.1f에 점흡수와 차수하한 사용 방법을 추가로 구현하고 외부차수는 사용하지 않는 방법이다. 방법 2에서는 정의 4의 차수하한 정의에서 $(\max_{C\in K_u}|n(C)|-1)$ 부분을 제거한 차수하한 정의를 사용하였고, 수정된 최소차수순서화 방법의 19행과 22행에서도 $(\max_{C\in K_u}|n(C)|-1)$ 부분이 제거된 방법을 사용한다. 방법 3은 방법 2와 같으나 정의 4의 차수하한 정의와 수정된 최소차수순서화 방법을 그대로 사용한다는 점에서 방법 2와 다르다.

LPABO ver 5.1f와 방법 1, 2, 3에서는 순서화 시간과 비영요소수, 차수수정 회수를 출력하였다. Time은 모두 순서화 시간을 의미하고 Nonz는 촐레스키인자의 비영요소 수를 뜻한다. NDEGUPD는 차수수정이 일어난 회수를 의미한다. 방법 1은 LPABO ver 5.1f와 촐레스키인자의 비영요소 수와 차수수정 회수가 동일하므로 표에서 생략하였다. 또한 방법 2와 방법 3은 순서화 시간에 있어서만 차이가 있고 촐레스키인자의 비영요소의 수는 같으므로 방법 2에 대해서만 촐레스키인자의 비영요소 수를 표시하였다.

실험결과에서 알 수 있듯이 방법 1의 점흡수는 약 5~10%정도의 수행시간 단축 효과를 가져왔고 방법 2의 점흡수와 하한 사용은 수행시간을 2~7배 감소시키는 효과를 나타냈다. 방법 3에서 큘릭의 원소 개수를 이용한 하한 사용법은 문제에 따라 다르지만 계산시간이 오히려 늘어나는 결과를 가져왔다. 이는 $(\max_{C\in K_u}|n(C)|-1)$을 계산하는 데 소요되는 시간이 차수 수정을 줄이는 시간보다 많기 때문이다. 따라서 방법 2가 수행시간 면에서 전반적으로 방법 3에 비해 좋다고 볼 수 있다. 촐레스키인자의 비영요소 수를 비교해 보면 방법 2의 차수하한 사용으로 비영요소수가 증가하는 경향이 있으나 감소되는 문제도 있음을 알 수 있다.

〈표 1〉 비교 실험결과

| | | | LPABO ver 5.1f | | | 방법 1 | 방법 2 | | | 방법 3 | | HOPDM | | BPMPD | | CPLEX | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 문 제 | NROW | NCOL | Time | NonZ | NDEGUPD | Time | Time | NonZ | NDEGUPD | Time | NDEGUPD | Time | NonZ | Time | Nonz | Time | Nonz |
| df001 | 6071 | 12230 | 35.03 | 1673222 | 107380 | 41.78 | 5.67 | 1639496 | 19110 | 5.78 | 17892 | 5.03 | 1610864 | 29.46 | 1604766 | * | |
| pilot87 | 2030 | 6460 | 18.74 | 430023 | 52843 | 20.11 | 1.35 | 448454 | 4580 | 1.61 | 4132 | 2.33 | 439164 | 7.34 | 496039 | 0.69 | 458132 |
| d2q06c | 2171 | 5831 | 0.75 | 137420 | 13222 | 0.67 | 0.28 | 163610 | 3544 | 0.31 | 3479 | 0.34 | 155592 | 0.43 | 175406 | 0.11 | 155279 |
| truss | 1000 | 8806 | 0.29 | 59916 | 5910 | 0.27 | 0.39 | 250682 | 925 | 1.30 | 557 | 0.14 | 58092 | 0.20 | 81044 | 0.04 | 60311 |
| bnl2 | 2280 | 4442 | 0.47 | 82968 | 11622 | 0.42 | 0.21 | 94698 | 2958 | 0.20 | 2119 | 0.19 | 82291 | 0.35 | 98116 | 0.09 | 93318 |
| woodw | 1098 | 8418 | 0.61 | 48226 | 5909 | 0.45 | 0.16 | 53524 | 1846 | 0.12 | 1232 | 0.24 | 47729 | 0.39 | 85809 | 0.13 | 54346 |
| greenbea | 2389 | 5495 | 0.81 | 77481 | 12571 | 0.81 | 0.24 | 80739 | 2614 | 0.21 | 1578 | 0.53 | 83599 | 0.50 | 105898 | 0.19 | 90330 |
| co5 | 5715 | 12125 | 4.40 | 175234 | 29780 | 4.18 | 0.84 | 225286 | 6101 | 1.65 | 4387 | 1.81 | 178409 | 2.78 | 225628 | 1.02 | 211975 |
| co9 | 10694 | 22527 | 14.67 | 471845 | 67022 | 14.28 | 2.48 | 560697 | 14111 | 5.38 | 10662 | 6.07 | 498717 | 7.40 | 576995 | 2.16 | 556481 |
| cq9 | 9247 | 21187 | 11.58 | 420213 | 52193 | 10.48 | 1.92 | 481780 | 12510 | 4.08 | 9661 | 5.15 | 439579 | 5.86 | 515342 | 1.79 | 496570 |
| nl | 7031 | 15290 | 5.04 | 276997 | 40992 | 5.16 | 1.18 | 284860 | 10671 | 1.94 | 9470 | 2.76 | 284042 | 2.47 | 313415 | 0.67 | 323739 |
| cre-b | 7240 | 77137 | 29.38 | 979004 | 64100 | 25.27 | 3.50 | 981272 | 9998 | 3.51 | 6387 | * | | 11.53 | 1205561 | 2.56 | 1000243 |
| ken-13 | 28632 | 42659 | 33.31 | 313445 | 74520 | 32.29 | 21.73 | 318346 | 32894 | 9.30 | 23740 | 2.88 | 178702 | 5.51 | 414021 | 2.00 | 422835 |
| osa-07 | 1118 | 25067 | 1.38 | 54807 | 204 | 0.81 | 0.83 | 54783 | 49 | 5.19 | 46 | 3.95 | 54783 | 0.53 | 200887 | * | |
| pds 10 | 16558 | 49932 | 108.45 | 1822541 | 168887 | 108.57 | 11.77 | 1640248 | 24744 | 12.07 | 19465 | 7.97 | 777310 | 30.75 | 1928124 | 2.02 | 1752175 |
| lp22 | 2958 | 16392 | 101.62 | 947811 | 178204 | 100.95 | 3.37 | 937741 | 12452 | 4.80 | 12262 | 25.44 | 939715 | 46.37 | 994280 | 1.42 | 1052546 |
| p10 | 10081 | 19081 | 11.45 | 497580 | 60857 | 10.88 | 8.28 | 762710 | 4772 | 11.67 | 5293 | 6.85 | 544070 | * | | 1.76 | 638571 |
| r05 | 5171 | 9671 | 3.87 | 459135 | 30684 | 3.70 | 2.90 | 643870 | 2437 | 3.53 | 2653 | 4.00 | 473825 | 1.96 | 557858 | 0.72 | 560921 |
| lpl3 | 10828 | 33686 | 6.05 | 331366 | 28410 | 6.13 | 3.86 | 332984 | 13943 | 5.34 | 8631 | * | | 2.94 | 419777 | 1.13 | 382096 |
| nemsemm2 | 6922 | 46579 | 3.32 | 111908 | 19430 | 1.54 | 1.15 | 129620 | 7021 | 0.69 | 4405 | * | | 0.76 | 272904 | 0.33 | 146494 |
| nsir2 | 4451 | 10055 | 9.73 | 312275 | 11108 | 2.89 | 1.41 | 323498 | 2575 | 1.08 | 922 | * | | 1.95 | 461832 | 0.89 | 327043 |
| aa3 | 825 | 8627 | 5.60 | 205155 | 15855 | 4.21 | 0.91 | 204867 | 2061 | 1.09 | 1945 | 1.02 | 203810 | 2.53 | 275961 | * | |
| aa5 | 801 | 8308 | 5.39 | 202314 | 15609 | 4.06 | 0.93 | 201572 | 2176 | 0.98 | 2104 | 0.96 | 201001 | 2.31 | 269499 | * | |

* : 문제 입력 또는 순서화 수행 중에 비정상적으로 종료됨을 의미함.

HOPDM과 BPMPD와의 비교에서도 예외는 있지만 방법 2가 더 수행시간 면에서 우수한 것으로 나타났고 촐레스키인자의 비영요소수도 크게 늘어나지 않는 것으로 나타났다. CPLEX는 순서화 수행 이전에 문제의 크기를 줄이는 경우가 많아 수행시간을 직접 비교할 수 없지만 방법 2가 전반적으로 약 1.5배 느린 것으로 나타났다.

## 6. 결 론

본 연구에서는 클릭구조를 사용한 최소차수순서화 방법의 수행속도를 향상시킬 수 있는 점흡수와 차수하한을 사용한 방법을 제시하였다

점흡수는 클릭흡수를 일반화한 방법으로서 차수 계산시에 탐색해야 하는 클릭의 개수를 줄이는 효과가 있다. 차수 하한을 이용한 수정된 최소차수순서화 방법에서는 삭제된 점의 인접점들의 차수를 곧바로 수정하지 않고 차수의 하한만을 수정하고 정확한 차수의 계산이 필요한 시점에 가서야 비로소 차수를 계산하기 때문에 차수 수정 회수를 줄이게 된다.

본 연구에서는 위의 두 방법의 효율성을 실험을 통해 보이고 있다. 실험 결과 점흡수는 수행시간을 약 5~10%정도 줄이는 효과가 있고 차수하한을 사용하는 방법은 수행시간을 약 40~50%정도 감소시키는 효과가 있다. BPMPD와 HOPDM과의 비교에서도 순서화 시간 면에서 빠른 것으로 나타났다.

차수 하한을 사용하는 최소차수순서화 방법은 추가되는 비영요소를 줄이기 위해 사용되는 외부차수와 함께 사용되기 어려워 추가되는 비영요소의 수가 외부차수를 사용한 최소차수순서화 방법보다 많을 수 있다. 그러나 실제 값에 가까운 외부차수의 하한을 구하는 것은 쉽지 않을 것으로 보인다. 따라서 비영요소를 줄일 수 있는 효과적인 tie-breaking 규칙을 사용하는 것도 차수 하한을 사용하는 최소차수순서화 방법의 추가되는 비영요소 수를 줄이는 방법이 될 것으로 예상된다.

## 참 고 문 헌

[1] 김병규, 성명기, 박순달, "내부점기법에 있어서 효율적인 순서화와 자료구조(최소부족순서화를 중심으로)", 「한국경영과학회지」, 제21권, 제3호(1980), pp.63-74.

[2] 모정훈, "내부점 선형계획법에서의 순서화방법과 자료구조에 관한 연구", 서울대학교 석사논문, 1995.

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

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

[5] George, A., Joseph W. H. Liu, *Computer solution of large sparse positive definite systems*, Prentice-Hall, 1981.

[6] George, A., Joseph W. H. Liu, "The evolution of the minimum degree ordering algorithm," *SIAM Review*, Vol.31, No.1(1989), pp.1-19.

[7] Jung, H.W., Roy E. Marsten and Matthew J. Saltzman, "Numerical factorization methods for interior point algorithms," *ORSA Journal on Computing*, Vol.6, No.1(1994), pp.94-104.

[8] Liu, J. W., "Modification of the minimum degree algorithm by multiple elimination," *ACM Trans. Math. Software*, Vol.11(1985), pp.141-153.

[9] Rose, D. J., "A Graph-Theoretic Study of the Numerical Solution of Sparse Positive Definite Systems of Linear Equations," in

*Graph Theory and Computing*, R.C. Read, ed., Academic Press, (1972), pp.183-217.

[10] Speelpenning, B.,The generalized element method, Tech. Rep. UIUCDCS-R-78-946, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, IL, 1978.

[11] Tinney, W. F. and John W. Walker, "Direct Optimally Ordered Triangular Factorization," *Proceedings of the IEEE*, Vol.55(1967), pp. 1801-1809.

[12] Yannakakis, M., "Computing the minimum fill-in is NP-complete," *SIAM J. Algebraic and Discrete Methods*, Vol.2(1981), pp. 77-79.