韓國經營科學會誌
第21卷 第3號
1996年 12月



# 내부점기법에 있어서 효율적인 순서화와 자료구조*
## (최소부족순서화를 중심으로)

박순달** · 김병규** · 성명기**

# An Efficient Ordering Method and Data Structure of the Interior Point Method
## (Putting Emphasis on the Minimum Deficiency Ordering)

Soondal Park** · Byunggyoo Kim** · Myeongki Seong**

## Abstract

Ordering plays an important role in solving an LP problem with sparse matrix by the interior point method. Since ordering is NP-complete, we try to find an efficient heuristic method.

The objective of this paper is to present an efficient heuristic ordering method for implementation of the minimum deficiency method. Both the ordering method and the data structure play important roles in implementation. First we define a new heuristic pseudo-deficiency ordering method and a data structure for the method - quotient graph and clique storage. Next we show an experimental result in terms of time and nonzero numbers by NETLIB problems.

## 1. 서 론

최근에는 선형계획법의 한 해법으로 내부점기법이 많이 연구되고 있다[6]. 내부점기법은 계산복잡도면에서 뿐만 아니라 수행속도면에서 우수한 결과를 보이고 있다[2][7]. 내부점기법은 단체법의 경우와는 달리 문제의 크기에 민감하지 않아서 대부분의 문제들이 적은 회수에 풀린다[9].

내부점기법은 매회 마다 $A\Theta A^T\mathbf{y}=\mathbf{b}$의 방정식을 풀어야 하는데 이 과정이 전체 수행시간의 대부분을 차지한다[6]. 따라서 내부점기법의 수행속도는 이 부분을 효율적으로 푸는 데

* 본 연구는 일주문화재단의 지원을 받은 것이다.
** 서울대학교 공과대학 산업공학과

달려 있다. 대체적으로 내부점기법은 대형문제를 고려하며 이 때 행렬 $A\Theta A^T$는 희소행렬일 경우가 많다. 그래서 이 행렬을 희소행렬이라고 가정한다.

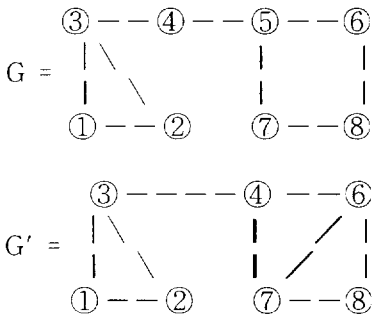
대칭양정치행렬(symmetric positive definite matrix)인 $A\Theta A^T$ 행렬은 보통 촐레스키 분해(Cholesky factorization)을 통해 $LL^T$ 형태로 상하분해하여 하삼각행렬 L만을 보관한다. L을 생성할 때 가능한 적은 수의 비영요소를 생성하는 것이 중요하다. 즉, 촐레스키분해 과정에서 새로 생기는 비영요소인 추가요소(fill-in)를 최소화하는 것이 중요하다. $A\Theta A^T$ 행렬에서 A의 행의 순서를 달리하면 이 추가요소의 수가 달라진다. 추가요소를 적게 하기 위하여 A행렬의 행의 순서를 달리하는 것을 순서화(ordering)라고 한다. 추가요소를 최소화하는 최적 순서화는 NP-complete 문제로 알려져 있다. 그래서 주로 발견적 기법인 최소차수순서화(minimum degree ordering)방법이 사용되고 있다[5]. 최소차수순서화는 가우스소거 연산의 각 단계에서 선회점으로 삭제그래프에서 이웃점의 수가 가장 적은 점을 선택하는 방법으로 구현이 간단하기 때문에 이에 대한 연구가 많이 진행되어 왔다[4][5][8]. 본 연구의 목적은 최소차수순서화와는 다른 1957년에 Tinney와 Walker에 의해 제안된 최소부족순서화(minimum deficiency ordering)를 효율적으로 구현하는 것이다. 이 방법에 대해서는 간단한 연구만이 되어 있다[1]. 그래서 최소부족순서화 방법의 특성을 연구하고 이를 효율적으로 구현하는 방법과 자료 구조에 관해 연구하고자 한다.

## 2. 순서화

최소부족순서화는 현재의 삭제그래프에서 한 점이 삭제될 때 추가되는 호의 수가 가장 적은 점을 선택하는 방법이다. 즉, 현재의 소거 단계에서 추가요소를 제일 작게 하는 방법이다. 삭제그래프에서 한 점과 연결된 호의 수, 즉, 인접하고 있는 점의 수를 차수라고 하며 D(i)로 표시하고, A(i)는 점 i의 인접점의 집합이라고 정의한다. 그러면, 점 i의 부족수는 그 점이 삭제되었을 때 생성되는 호의 수로 다음과 같이 계산할 수 있다.

$$\text{부족수(i)} = \binom{D(i)}{2} - (\text{A(i)의 점끼리 연결된 호의 수})$$

즉, 삭제그래프에서 점 i를 삭제할 때, 점 i의 인접점은 모두 연결되므로, 이 때 생성되는 호의 수는 인접점 간의 모든 연결상태에서 이미 연결되어 있는 호의 수를 뺀 값이 된다. 예를 들어 [그림 1]의 그래프 G에서 점 5를 삭제하게 되면 삭제그래프 G′과 같이 된다.




[그림1] 그래프 G 와 삭제그래프 G′

점 5의 부족수는 A(5)={4, 6, 7}, D(5)=3이기 때문에 부족수(5)$=\binom{3}{2}-0=3$ 이다.

이 방법은 다른 순서화 방법에 비해 새로 생

기는 비영요소의 수는 적으나 부족수의 계산이 복잡하다는 단점이 있다. 즉, 부족수를 계산하기 위해서는 각 점에서 인접점의 연결상태를 모두 알아야 하기 때문이다.

최소부족순서화를 수행하기 위해서는 일단 주어진 그래프에서 각 점들의 부족수를 모두 알아야만 한다. 이를 위해서는 각 점별로 인접점의 연결상태를 모두 파악해야 한다. 각 점별로 인접점의 연결상태를 파악하면 호의 연결상태가 중복되게 파악되기 때문에 그렇게 효과적인 방법이 되지 못한다. 여기에서는 각 점의 부족수를 호 중심으로 중복없이 구하는 방법을 제시하겠다.

먼저 다음과 같은 관계를 정의하겠다.

**정의 1 삼각관계** 세 개의 점 i, j, k가 서로 인접해 있으면 이것을 삼각관계라고 한다. 그리고 이를 〈i, j, k〉로 표시하겠다.

**정리 1** 주어진 무방향 그래프 G에서 임의의 점 i의 부족수는 $\binom{D(i)}{2}$−(점 i가 포함된 삼각관계의 수)이다.

**(증명)** 점 i의 부족수는 정의에 따라서 $\binom{D(i)}{2}$−(A(i)의 점끼리 연결된 호의 수)이다.

점 i의 인접점 중 점 j, k가 서로 인접하고 있다고 하자. 그러면 〈i, j, k〉는 삼각관계를 하나 형성한다. 즉, 점 i의 인접점들을 서로 연결하는 호의 수는 점 i가 포함된 삼각관계의 수와 서로 같다. 즉, (A(i)의 점끼리 연결된 호의 수)=(점 i가 포함된 삼각관계의 수)가 된다. 따라서 위의 정리가 성립한다. ■

위의 정리를 이용하면 주어진 무방향 그래프의 모든 호의 부족수는 그래프의 모든 삼각관계를 파악하면 구할 수 있다.

최소차수순서화에서 차수 수정과 삭제그래프 변형을 감소시키기 위하여 구별할 필요가 없는 점들이 있고 이것을 최소부족순서화에서 사용하면 편리하다. 어떤 점 i에 대해서 A(i)∪{i}, 즉, 자기 자신과 인접점의 합집합을 이웃집합(neighborhood set)이라고 한다. 그러면 다음을 정의할 수 있다.

**정의 2 구별불능점** 두 개의 점 i, j의 이웃집합이 같거나 인접점이 같으면 두 개의 점은 구별불능(indistinguishable)이라고 한다. 즉, A(i)∪{i} = A(j)∪{j}이거나 A(i)=A(j)이면 구별불능점이 된다[5].

이렇게 정의된 구별불능점은 다음의 정리에 의해 동시에 삭제가 가능하다.

**정리 2** 점 i가 현재의 최소부족점이고, 두 개의 점 i, j가 그래프 G에서 구별불능이면, 삭제점 i을 삭제한 후의 삭제그래프에서 j는 최소부족점이 된다[5]. ■

위의 정리는 점 i와 점 j가 구별불능이면 현재 점 i를 삭제하면 그 다음에는 점 j를 선택할 수 있기 때문에 점 i와 점 j를 함께 삭제할 수 있음을 의미한다. 따라서, 구별불능점들을 동시에 삭제함으로써 순서화 과정에서 대부분의 시간을 차지하는 삭제그래프의 변형과 각 점들의 부족수 수정을 감소시킬 수 있다.

**정리 3** 두 개의 점 i, j가 그래프 G에서 구별불능하면, 어떤 최소부족점 y를 삭제하고 난 후에도 두 점은 구별불능하다[5]. ■

위의 정리는 어느 한 단계에서 구별불능이 되면 이후에는 계속 구별불능이 되고 따라서 구별불능인 점들은 같이 취급할 수 있음을 의미한다. 구별불능점들은 동시에 같이 제거할 수 있는 성질이 있다. 또한 각 점들 간에는 삭제되는 선후 관계가 있다.

**정의 3 우세점** 두 점 i, j의 이웃집합 사이에 A(i)∪{i}⊂A(j)∪{j} 와 같은 포함관계가 있거

나 인접점이 A(i)⊂A(j)인 포함관계를 만족할 때 점 i가 점 j보다 우세하다(j is out matched by i)고 한다[5].

**정리 4** 점 i가 점 j보다 우세하면 점 i는 점 j보다 먼저 삭제된다.

**(증명)** 먼저 인접점 간에 포함관계가 있는 경우를 보면 점 i의 인접점이 점 j의 인접점에 포함되기 때문에 점 j의 부족수는 점 i의 부족수에다 점 i의 인접점과 점 i의 인접점이 아닌 집합간의 부족수를 더해 주어야 한다. 점 i의 부족수가 점 j보다 작거나 같은 것은 분명하다. 이웃 집합들 간에 포함관계가 있는 경우도 인접점 간에 포함관계가 있는 경우와 마찬가지다. 따라서 점 i가 점 j보다 항상 먼저 삭제된다. ■

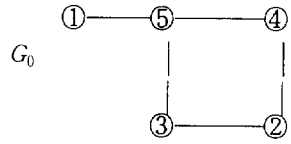
정리 4는 점 i가 점 j보다 우세하면 점 j의 부족수는 점 i를 삭제하면서 수정되어야 하기 때문에 점 i을 삭제하기 이전에는 부족수를 수정할 필요가 없음을 말한다. 따라서 점 i을 삭제할 때까지 부족수의 수정을 하지 않아도 된다. 이는 최소부족순서화에서 계산량의 상당부분을 차지하는 부족수 수정을 줄임으로써 수행속도를 개선할 수 있다.

## 3. 자료 구조

최소부족순서화는 삭제그래프를 이용하는데 이 삭제그래프의 표현은 최소부족순서화의 효율에 있어서 매우 중요하다. 삭제그래프를 나타내기 위한 자료구조로 쿼션트(quotient)그래프모형이 주로 사용되었다[1][4]. 본 논문에서는 삭제그래프를 표현하는 방법으로 쿼션트그래프모형과 클릭(clique)모형을 함께 제시하고 이를 구현하는 방법들도 제시한다.

### 쿼션트그래프모형

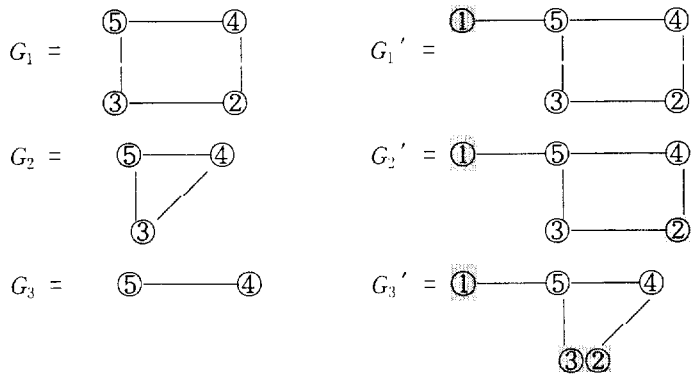
쿼션트그래프모형은 A. George와 W. H. Liu에 의해서 제안된 방법이다[4]. 이 방법은 인접구조(adjacency structure)를 이용한다. 인접구조는 그래프를 각 점의 인접점으로 표현하는 방법이다. 다음의 삭제그래프가 있다고 하자.




[그림 2] 삭제그래프 $G_0$

삭제그래프에서 어느 한 점을 삭제하면 그래프를 표현한 인접구조도 바뀌어야 한다. 이러한 삭제그래프의 변형은 많은 비용을 동반한다. 쿼션트그래프의 개념은 삭제그래프의 변형을 줄이자는 것이다. 점이 삭제될 때마다 그래프를 변형하지 않고 삭제된 점에 삭제표시만 한다. 그래프가 변형되는 경우는 두 개 이상의 삭제점이 인접한 경우이다. 이 때 이 점들은 하나의 점으로 통합한다. 이 방법을 예로 표시하면 [그림 3]과 같다.

[그림 3]에서 $G_1$, $G_2$, $G_3$는 삭제그래프의 변형과정이고 $G_1'$, $G_2'$, $G_3'$은 그에 해당하는 쿼션트그래프이다. $G_1'$, $G_2'$은 그래프의 변형 없이 삭제표시만 했다. 점 3이 삭제되는 경우 점 2와 점 3을 하나로 통합한다. 이 경우는 쿼션트그래프가 변하게 된다. 음영이 들어간 것은 삭제 표시이다. $G_3'$의 경우 점 2와 점 3이 통합되어 하나의 점이 되므로 구조가 변한다.

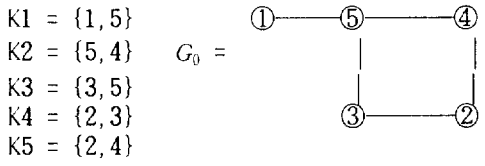



[그림 3] 삭제그래프와 대응하는 쿼션트 그래프

쿼션트그래프 구조의 다른 장점은 추가의 기억공간이 불필요하다는 것이다. 처음 삭제그래프를 표시할 때 필요한 공간 이상이 소용되지 않는다[4]. 이러한 성질에 의해 쿼션트그래프를 표현하는 데 정적자료구조(static data structure)를 이용할 수 있다. 쿼션트그래프를 통해서 삭제그래프상의 인접점을 구할 때는 인접점이 삭제표시되어 있으면 삭제표시된 점의 인접점까지 파악해야 한다.

## 클릭모형

클릭은 모든 점들이 연결된 그래프를 말하며 Duff는 이를 이용해서 삭제그래프를 표시하였다[3]. 가장 간단한 방법은 주어진 그래프를 호의 수만큼의 두 개의 점으로 구성된 클릭의 집합으로 표시하는 것이다. [그림 4]는 주어진 그래프를 호의 개수만큼의 클릭으로 표시한 것이다.




[그림 4] $G_0$의 클릭구조 표현

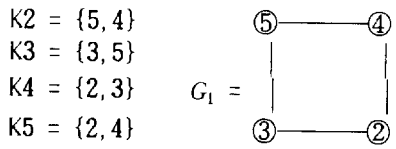
삭제그래프를 클릭의 집합으로 표시하면 이것은 삭제과정을 보는 다른 관점을 제공할 뿐만 아니라 삭제그래프를 표시할 수 있는 효율적인 방안도 제시한다. 삭제과정에서 삭제되는 점의 인접점이 클릭을 형성하기 때문에 삭제그래프를 클릭의 집합으로 표시하면 삭제과정을 효과적으로 나타낼 수 있다. 클릭의 집합 $\{K_1, K_2, \cdots, K_q\}$이 현재의 그래프 G를 나타내고 점 i가 삭제될 점이라고 할 때 집합 $\{K_{s_1}, \cdots, K_{s_t}\}$를 점 i가 속한 클릭들이라고 하자. 그러면 삭제그래프를 변형하는 것은 다음과 같은 단계로 나타낼 수 있다.

1) 집합 $\{K_1, K_2, \cdots, K_q\}$에서 $K_{s_1}, \cdots, K_{s_t}$를 삭제한다.

2) 새로운 클릭 $K=(K_{s_1} \cup \cdots \cup K_{s_t})-\{i\}$를 클릭 집합에 추가한다.

즉, 삭제점 i가 있는 모든 클릭을 삭제하고 이들 클릭의 합집합으로 이루어진 하나의 클릭을 추가하는 것이다. [그림 4]의 그래프에서 점 1을 삭제하는 과정은 1이 포함된 클릭 $K_1$을 삭제하고 새로운 클릭 $K_1'$를 만든다. $K_1'=\{5\}$이다. 클릭의 요소수가 1인 경우는 삭제해도

되므로 결과로 생기는 $G_1$의 클릭구조는 [그림 5]와 같다.




[그림 5] $G_1$의 클릭구조 표현

쿼션트그래프와 마찬가지로 기억공간의 양에 대하여 $|K| \leq \sum_{i=1}^{t} |K_{s_i}|$와 같은 관계가 성립한다 [4].

위의 관계는 새로 형성된 클릭집합 K가 기존의 클릭집합보다 적은 양의 기억공간을 사용한다는 것이다. 따라서 기억공간의 양은 처음에 필요한 양보다 증가하지는 않는다.

임의의 그래프가 주어졌을 때 초기의 클릭구조는 모든 호에 해당하는 클릭으로 그래프를 표현할 수 있지만 클릭의 수가 많으면 클릭을 관리하는 데 많은 계산비용이 들기 때문에 바람직하지 못하다. 관심의 대상인 $AA^T$행렬을 표현하는 클릭구조를 결정하기 위해서는 행렬 A의 각 열의 비영요소는 $AA^T$에 해당하는 그래프에서 클릭을 형성한다는 사실을 이용한다. 이 경우 삭제그래프를 적은 수의 클릭으로 표현할 수 있다. $M=AA^T$에서 $m_{ij}$가 비영요소이면 $a_{ik} \neq 0$이고 $a_{jk} \neq 0$인 열 k가 존재한다. 즉, 행렬 A의 임의의 열 k에서 비영요소들의 행이 $I=\{i_1, i_2, \cdots, i_l\}$이면 I의 원소로 이루어지는 모든 쌍 (p, q)에 대하여 $m_{pq} \neq 0$이다. 이것은 그래프의 관점에서 보면 I에 해당하는 점들이 클릭을 형성함을 의미한다.

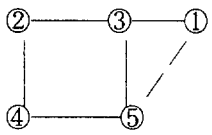
다음과 같은 비영요소를 가진 A행렬이 주어졌다고 하자. *는 비영요소이다.

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

[그림 6] 임의의 A행렬의 구조

이에 해당하는 $AA^T$의 구조와 그래프는 다음과 같다.

$$AA^T = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} & & & & \end{array} \\ \begin{pmatrix} * & & * & & * \\ & * & * & * & \\ * & * & * & & * \\ & * & & * & * \\ * & & * & * & * \end{pmatrix} \end{array}$$




[그림 7] $AA^T$의 구조와 해당하는 그래프

[그림 7]을 보면 A의 각 행의 비영요소의 열은 각각 {1, 3, 5}, {2, 3}, {4}, {3, 5}, {4, 5}, {2, 4}이다. A행렬의 각 열들이 $AA^T$에 해당하는 그래프의 클릭집합을 형성함을 알 수 있다. {4}는 형성할 수 있는 호가 없으므로 의미가 없다. {3, 5}⊂{1, 3, 5}이므로 필요가 없다. 따라서 위의 그래프는 {1, 3, 5}, {2, 3}, {4, 5}, {2, 4}의 4개의 클릭으로 표시됨을 알 수 있다.

각 열의 비영요소들의 행의 첨자로 클릭의 집합을 표시하면 그것은 $AA^T$에 해당하는 그래프를 표현하는 클릭의 집합이 된다. 따라서 $AA^T$을 표현하는 전체 클릭의 수는 행렬 A의 열의 수를 초과하지 않는다. 이 때 비영요소의 수가 1인 열은 제외한다.

클릭 저장구조에서는 그래프를 나타내는 클릭의 수는 삭제변형을 용이하게 하기 위해서는 작을수록 유리하다. 클릭흡수(clique absorption)란 클릭의 수를 줄이기 위해서 중복되는 클릭을 없애는 것이다. 클릭의 집합 $\{K_1, K_2, \cdots, K_q\}$이 현재의 그래프 G를 나타낸다고 하자.

**정리 5** 어떤 s와 t에 대하여 $K_s \subset K_t$이면 그래프 G는 $\{K_1, K_2, \cdots, K_q\} - K_s$로 표현될 수 있다[5][8]. ■

위의 정리에서 클릭 $K_s$는 그래프 G를 중복적으로 표현하고 있으므로 $K_t$에 의해서 흡수될 수 있다. 삭제그래프의 변형에서 점 i가 삭제될 때 점 i가 포함되는 클릭을 모두 삭제하는 것은 바로 새로 형성되는 클릭에 이 모든 점들이 포함되어 있기 때문이다. 단순한 클릭의 흡수만이 아니라 여러 개의 클릭을 하나로 합병하여 표현할 수도 있다. 예를 들어, 3개의 클릭 K1={1, 2}, K2={2, 3}, K3={1, 3}의 경우는 하나의 클릭 {1, 2, 3}으로 표현할 수 있다. A의 임의의 열 j의 비영요소의 행번호의 집합을 R(j)로 표현하면 정리 5에 의해서 $R(s) \subset R(t)$이면 s열에 의해서 형성되는 클릭은 t열에 의해 흡수되므로 보관할 필요가 없다.

클릭구조에서 가장 중심적인 연산은 여러 클릭을 하나로 합치는 합병이다. 클릭구조의 합병을 위해서는 임의의 점 i가 포함되어 있는 클릭을 파악하는 연산과 이들 클릭을 합쳐주는 연산이 필요하다.

## 4. 계산법

앞서 언급한 바와 같이 부족수를 계산하는 데에는 인접점의 연결관계를 파악해야 하기 때문에 이에 많은 시간이 소요된다. 그래서 여기서는 두 가지 방법을 비교하고자 한다. 하나는 정리 1을 이용하여 정확하게 부족수를 계산해 가면서 진행하는 방법이고 또 하나는 부족수의 근사치를 구해서 진행하는 방법이다. 부족수 계산은 국지적으로 최적화하는 방법이기 때문에 정확한 부족수를 계산한다고 전체 최적순서화를 구하는 것은 아니기 때문에 국지적 순서화와 부족수 계산의 취사선택의 문제가 일어난다. 그래서 두 가지 방법을 비교하고자 한다. 추가요소의 수가 다소 증가하더라도 정확한 부족수를 계산하기보다는 계산하기 용이한 유사부족수를 정의해서 이를 사용하는 것이 효율적이다.

### 정확한 부족수 계산법

부족수는 정리 1에 의해서 구할 수 있다. 주어진 무방향 그래프 G에서 임의의 점 i의 부족수는 점 i의 부족수=$\binom{D(i)}{2}$-(점 i가 포함된 삼각관계의 수)이다.

이 식에서 점 i가 포함된 삼각관계를 파악하는 데 많은 시간을 소요하게 된다.

정확한 부족수에 의한 최소부족순서화의 알고리듬을 나타내면 다음과 같다.

### 알고리듬 1 최소부족순서화

```
begin
  G는 AA^T에 해당하는 무방향 그래프
  while 무방향 그래프 G≠φ do
  begin
    G에서 최소부족점을 선택
    이 점을 G에서 삭제하고 삭제그래프를
    변형한다.
  end
end
```

실제로 프로그램에서는 이 밖에도 해싱(hashing)함수를 이용한 구별불능점을 사용하여 계산효율을 높인다.

### 유사부족수 계산법

이제 위에서 살펴본 자료구조의 특성을 이용해서 각각의 자료구조에 적합하고 계산하기 용이한 유사부족수를 정의한다.

이제부터 E(i)는 점 i의 인접점 중 삭제된 점들의 집합, U(i)는 점 i의 인접점 중 삭제되지 않은 점들의 집합, $D_u(i)$는 U(i)에 있는 점들의 수라고 하자.

**정의 4 유사부족수(쿼션트그래프모형)** 점 i의 쿼션트그래프상에서 유사부족수는 다음과 같이 정의한다.

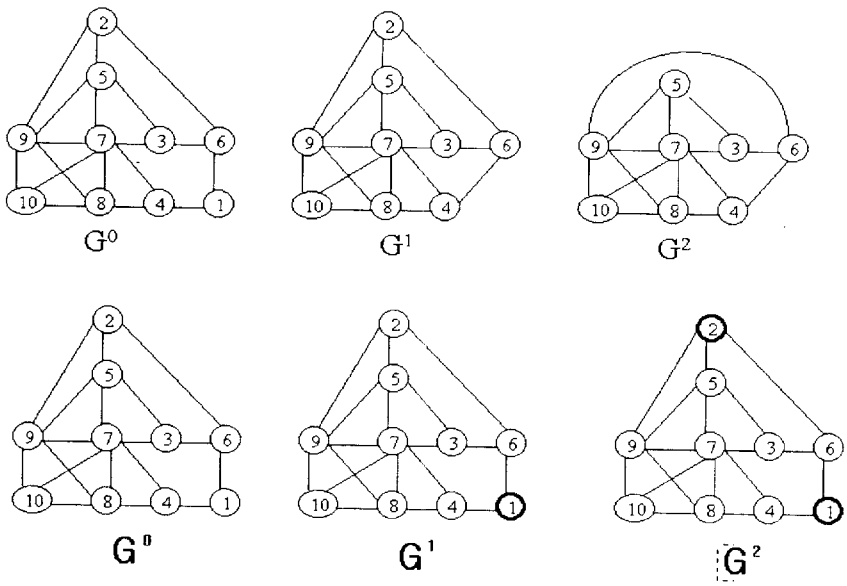
유사 부족수(i)=$\binom{D_u(i)}{2}$−(U(i)의 점끼리 연결된 호의 수)+$\alpha$(E(i))

$\alpha$(E(i))는 E(i)에 있는 점들의 차수의 합을 의미한다.

즉, 부족수를 구하기 위해서는 인접점의 연결관계를 알아야만 하는데 쿼션트그래프에서 인접점이 삭제되었을 경우 다시 이 삭제된 점의 인접점을 찾아가서 이 점들 간의 연결관계를 알아보아야 하지만 인접점 중에서 삭제되지 않은 점들만으로 대강의 부족수를 계산하고 삭제된 점의 인접점과의 부족수는 $\alpha$(E(i))로 간단하게 계산을 하고자 하는 것이다.

예를 들어, [그림 8]을 보면 $G^i$는 삭제그래프이고 $\mathbf{G}^i$는 쿼션트그래프를 나타낸다.

[그림 8]에서 굵게 표시된 점은 삭제된 점이다. $G^1$에서 4번 점은 6, 7, 8과 연결되어 있고 이들 사이에는 7, 8만이 연결되어 있으므로 부족수는 2이다. 이를 쿼션트그래프 $\mathbf{G}^1$에서 보면 7, 8과 이미 삭제된 1의 인접점 6을 생각해야 한다. 하지만 이미 삭제된 점 6의 인접점을 찾




[그림 8] 삭제그래프와 쿼션트 그래프

는 것은 그리 쉽지 않고 찾더라도 찾아진 점들과 7, 8과의 연결관계를 파악하는 데 많은 계산량을 필요로 한다. 따라서 이미 삭제된 점의 경우 그 점의 차수에 비례하는 적당한 수 $\alpha$ (E(i))를 그냥 더해주는 계산법이 효율적이다. 즉, 차수가 (2n−1)에서 2n 사이에 있으면 n을 더해 주는 것이 하나의 방법이 될 수 있다. 물론 다른 공식에 의한 방법도 여러 가지 있을 수 있다.

$G^1$에서는 4번 점에 인접한 점은 7, 8이고 7, 8 사이에는 호가 있으므로 제거된 1번 점의 차수는 2이므로 1을 더한다. 따라서 유사부족수는 1이다. 이 경우 실제의 부족수보다 1이 작다. 다른 예를 들면, 위의 $G^2$에서 5번 점의 경우, 이는 3, 6, 7, 9와 연결되어 있고 이들 사이에는 {3, 6}, {3, 7}, {7, 9}, {6, 9} 등이 연결되어 있으므로 부족수는 2이다. $G^2$에서는 5번 점에 인접한 점은 3, 7, 9이므로 유사부족수는 3이다. 이 경우는 실제 부족수보다 1이 많다. 쿼션트그래프에서 유사부족수를 이용한 최소부족순서화의 알고리듬을 기술하면 다음과 같다.

알고리듬 2 쿼션트그래프에서 유사부족수를 이용한 순서화

```
begin
  무방향 그래프에 대응하는 쿼션트그래프 Q
  를 생성
  while Q≠φ do
  begin
    각 점의 유사부족수를 계산
    최소부족점을 선택해서 Q에서 삭제
    이웃한 삭제점이 없으면 삭제표시만 하
    고, 있으면 Q를 변형한다.
  end
end
```

클릭 구조를 이용할 경우 점 i의 부족수를 계산하기 위해서는 점 i가 있는 클릭들은 모두 살표보아야 하기 때문에 효율적이지 못하다. 따라서 계산하기 쉬운 유사부족수를 정의한다.

**정의 5 유사부족수(클릭모형)** 임의의 점 i가 들어있는 클릭을 $K_1$, $K_2$, …, $K_n$, 각각의 클릭에서 점 i를 뺀 원소의 수를 $c_1$, $c_2$, …, $c_n$, 인접점의 수를 D(i)라고 할 때

$$\text{점 i의 부족수} = \binom{D(i)}{2} - \sum_{i=1}^{n} \binom{c_i}{2}$$

으로 정의할 수 있다.

위에서 정의한 부족수는 클릭들 간의 연결상태는 고려하지 않는다. 하지만 클릭흡수를 사용할 경우에 중복적인 클릭이 제거되기 때문에 점 i가 있는 클릭들 간에 중복적인 요소는 많지 않게 되고 A행렬 자체가 상당히 희소하기 때문에 각 열에서 중복되는 비영요소의 수는 작게 된다. 따라서 위에서 정의한 부족수는 매우 적절한 유사부족수가 되며 클릭간의 연결상태를 고려하지 않기 때문에 정확한 부족수와 같이 부족수의 상한이 된다. 다음의 행렬에서 부족수와 $a_{4,6}\neq 0$이면 행렬의 모양과 그 때의 부족수의 변화는 [그림 9]와 같다.

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | ■ | ■ | ■ | | | |
| 2 | ■ | | | ■ | | |
| 3 | | ■ | | | ■ | |
| 4 | | | ■ | | | ■ |
| 5 | | | | ■ | ■ | ■ |
| 6 | ■ | | | | | ■ |

| 노드 | 정확한 부족수 | 유사 부족수 |
|---|---|---|
| 1 | 4 | 5 |
| 2 | 2 | 2 |
| 3 | 1 | 1 |
| 4 | 2 | 2 |
| 5 | 5 | 5 |
| 6 | 4 | 4 |

[그림 9] A의 구조 및 정확한 부족수와 유사부족수

**정리 6** 정의 5에서 정의한 점 i의 유사부족수는 점 i의 부족수의 상한(upper bound)이다.

**(증명)** 점 i가 들어 있는 클릭을 $\{K_1, K_2, \cdots, K_q\}$라고 할 때 위의 정의된 유사부족수는 클릭간에 연결관계를 고려하지 않았기 때문에 실제 부족수는 이 값보다 항상 작거나 같게 된다. ■

클릭구조에서 유사부족수를 이용한 최소부족 순서화의 알고리듬은 다음과 같다.

**알고리듬 3** 클릭구조에서 유사부족수를 이용한 순서화

```
begin
  무방향 그래프를 표현하는 클릭을 생성
  repeat
    각 점의 유사부족수를 계산
    최소부족점을 선택
    이 점을 포함한 클릭들을 제거
    제거된 클릭들로 새로운 클릭을 생성
  until 모든 점들이 삭제
end
```

## 5. 실험 결과 및 분석

본 논문에서는 쿼션트모형과 클릭모형에서 정확한 부족수와 앞서 정의한 유사부족수 간의 수행속도와 비영요소의 수를 비교하고자 한다. 실험 결과는 [표 1]에 제시된 10개의 실험문제를 대상으로 하였다. 이 문제들은 NETLIB 문제에서 발췌하였다. 문제들은 문제크기에 따라 정렬하였다. 본 실험은 HP 9000/730 워크스테이션에서 수행한 결과이다.

[표 1] 실험 문제

| 문제이름 | 문제 크기 | 비영요소수 | 밀도(%) |
|---|---|---|---|
| stocfor1 | 118 * 111 | 447 | 3.44 |
| israel | 175 * 142 | 2269 | 9.18 |
| woodlp | 245 * 2594 | 70215 | 11.09 |
| scorpion | 389 * 358 | 1426 | 1.03 |
| degen2 | 445 * 534 | 3978 | 1.68 |
| agg3 | 517 * 302 | 4300 | 2.76 |
| bnl1 | 644 * 1175 | 5121 | 0.68 |
| 25fv47 | 822 * 1571 | 10400 | 0.81 |
| scfxm3 | 991 * 1371 | 7777 | 0.57 |
| truss | 1001 * 8806 | 27836 | 0.32 |

[표 2]는 쿼션트모형에서 정확한 부족수와 유사부족수를 비교 실험한 것이다.

[표 2] 쿼션트 모형에서의 실험 결과

| | 정확한 부족수 | | 유사 부족수 | |
|---|---|---|---|---|
| 문제이름 | 수행시간 | 비영요소수 | 수행시간 | 비영요소수 |
| stocfor1 | 0.900 | 702 | 0.800 | 703 |
| israel | 79.650 | 10963 | 71.417 | 11309 |
| woodlp | 136.600 | 11643 | 131.900 | 11643 |
| scorpion | 1.717 | 1779 | 1.667 | 1814 |
| degen2 | 796.483 | 15223 | 714.650 | 15223 |
| agg3 | 240.500 | 19078 | 220.217 | 20191 |
| bnl1 | 126,650 | 10491 | 115.700 | 10491 |
| 25fv47 | 693.417 | 27649 | 546.967 | 27649 |
| scfxm3 | 50.933 | 11632 | 42.433 | 11632 |
| truss | 474.667 | 56135 | 418.483 | 56178 |

[표 2]를 보면 정확한 부족수의 수행시간에 비해 유사부족수의 수행시간이 적게는 3.4%에서 많게는 21.0%까지 향상을 가져왔다. 비영요소의 수면에서는 약간 증가하거나 거의 같은 수를 나타내고 있다. 따라서 유사부족수가 두 가지 면에서 모두 정확한 부족수에 비해 우수함을 알 수 있다.

[표 3]은 클릭모형에서 실험한 것이다.

[표 3] 클릭 모형에서의 실험 결과

| | 정확한 부족수 | | 유사 부족수 | |
|---|---|---|---|---|
| 문제이름 | 수행시간 | 비영요소수 | 수행시간 | 비영요소수 |
| stocfor1 | 0.567 | 845 | 0.133 | 768 |
| israel | 1.050 | 11216 | 0.317 | 10953 |
| wood1p | 24.150 | 18326 | 1.333 | 11826 |
| scorpion | 7.433 | 2352 | 0.967 | 1904 |
| degen2 | 227.267 | 16668 | 12.883 | 16031 |
| agg3 | 197.217 | 20409 | 14.150 | 19574 |
| bnl1 | 98.583 | 11846 | 11.433 | 11458 |
| 25fv47 | 513.50 | 29788 | 32.533 | 31639 |
| scfxm3 | 143.50 | 13017 | 18.500 | 13327 |
| truss | 702.167 | 61478 | 83.283 | 60529 |

[표 3]을 보면 정확한 부족수의 경우 인접점들 간의 연결상태를 파악하기 위해서 많은 클릭들을 찾아보아야 하기 때문에 수행시간이 상당히 길어지게 된다. 이에 비해 정확한 부족수는 쿼션트모형에 비해 수행시간이 개선되는 것은 분명하지만 25fv47같은 문제에서는 지나치게 비영요소의 수가 많아진다. 따라서 문제에 따라서는 유용할 수 있지만 일반적으로 비영요소수 면에서 단점을 지니게 된다. 유사부족수를 보면 수행시간이 가장 뛰어날 뿐만 아니라 비영요소수도 정확한 부족수에 비해 크게 늘어나지 않고 오히려 감소하는 경우도 생기는 것을 볼 수 있다. 클릭모형에서는 유사부족수가 가장 효율적임을 알 수 있다.

한편, 쿼션트모형과 클릭모형을 비교해 보면 [표 2]와 [표 3]에서 보는 바와 같이 비영요소수에서는 쿼션트모형이 수행시간에서는 클릭모형이 우수한 것으로 나타났다. 그리고 전체적으로 쿼션트모형보다는 클릭모형이 우수하다.

## 6. 결 론

본 연구는 내부점기법에서 최소부족순서화에 대해 다루었다. 순서화는 대칭양정치 선형방정식 $A\Theta A^T\mathbf{y}=\mathbf{b}$를 효과적으로 풀기 위해서 $A\Theta A^T$를 촐레스키분해할 때 비영요소의 수를 감소시키는 방법이다.

최소부족순서화에서 정확한 부족수 대신 삭제그래프를 표현하기 위한 자료구조인 쿼션트모형과 클릭모형에 적합한 유사부족수를 정의하였다. 클릭모형에서는 $AA^T$에 해당하는 그래프를 표현하는 데 있어서 A만을 이용해서 효과적으로 수행할 수 있음을 보였다.

유사부족수는 수행속도면에서 상당한 향상을 가져왔을 뿐만 아니라 비영요소수 면에서도 정확한 부족수와 비교해서 큰 차이를 보이지 않았으며 오히려 감소하는 경우도 있었다. 특히 클릭모형의 유사부족수는 수행속도와 비영요소수 두 가지 면에서 모두 좋은 성과를 나타냈다. 그리고 쿼션트모형보다는 클릭모형이 우수한 것으로 나타났다.

## 7. 참 고 문 헌

[1] 모정훈, "내부점 선형계획법에서의 순서화 방법과 자료구조에 관한 연구", 서울대학교 대학원 산업공학과 석사학위 논문, 1995

[2] 박순달, 선형계획법, 3판, 민영사, 1992

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

[4] George, J.A. and J.W.H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Engle wood Cliffs, NJ, 1981

[5] George, J.A. and J.W.H. LIU, "The Evolution of the Minimum Degree Ordering Algorithm", *SIAM Review* Vol.31, No.1 (1989), pp.1-19

[6] Jung, H.W., "Direct Sparse Matrix Methods for Interior Point Algorithms", Ph. D. Dissertation

[7] Karmarkar, N., "A New Polynomial-Time Algorithm for Linear Programming", *Combinatorica 4(4)* (1985), pp. 373-395

[8] Liu, J.W.H., "Modification of the Minimum-Degree Algorithm by Multiple Elimination", *ACM Transactions on Mathematical Software*, Vol.11, No.2 (1985), pp.141-153

[9] Marsten R., R. Subramanian, and M. Saltzman, "Interior Point Methods for Linear Programming: Just Call Newton Lagrange, and Fiacco & McCormick", *Interfaces*, Vol.20, No.4 (1990), pp. 105-116