논리 대수의 연산 특성을 이용한 D-클래스 계산에 대한 연구

한재일*
*국민대학교 컴퓨터학부

A Study on the Computation of D-Classes based on the Properties of Operations of Boolean Algebras

Jae-Il Han*
*Kookmin University
E-mail : jhan@kookmin.ac.kr

요 약

D-클래스는 암호학 등의 보안 분야에 응용될 수 있는 가능성을 가지고 있다. 그러나 D-클래스 계산은 NP-완전 문제로서 현재 4 × 4 이하 크기의 행렬에 대한 D-클래스 만이 알려져 있어 D-클래스의 특성에 대한 연구가 제대로 이루어지지 못하고 있다. 최근 새로운 D-클래스를 얻기 위하여 보다 효율적인 D-클래스 계산에 대한 연구가 수행되었으나 아직 새로운 D-클래스를 계산할 수 있을 정도의 효율적인 연구결과는 보이지 않고 있다. 본 논문은 논리 대수의 연산 특성을 이용하여 D-클래스 계산을 보다 효율적으로 할 수 있는 수학적 이론과 방법을 제시한다.

1. 서론

불리언 행렬은 원소가 참 또는 거짓 값을 갖는 행렬을 말하며, $F=\{0,1\}$, $M_n(F)$는 모든 $n \times n$ 불리언 행렬의 집합, A는 $M_n(F)$에 속한 임의의 불리언 행렬일 때 D-클래스는 다음과 같이 정의된다[6].

$$D_A = \{ B \in M_n(F) : \exists C,X,Y,U,V \in M_n(F) \text{ such that } AX = C,\ CY = A,\ UC = B,\ VB = C \}$$

위 정의에서 보듯이 D-클래스 계산은 $n \times n$ 불리언 행렬의 전체 집합을 대상으로 모든 불리언 행렬에 대한 이중 삼중의 곱셈을 요구하는 NP-완전 계산복잡도를 가지고 있으며 이로 인해 현재 4 × 4 이하 크기의 행렬에 대한 D-클래스 만이 알려져 있어 D-클래스의 특성에 대한 연구가 제대로 이루어지지 못하고 있다.

최근 새로운 D-클래스를 얻기 위하여 효율적인 D-클래스 계산에 대한 연구[1-5]가 수행되었으나 새로운 D-클래스를 계산할 수 있을 정도의 효율적인 연구결과는 보이지 않고 있다. 본 논문은 기존의

연구와 달리 논리 대수의 연산 특성을 이용하여 모든 불리언 행렬 사이의 중첩된 곱셈을 효율적으로 할 수 있는 이론을 정립하고, 이를 이용하여 D-클래스 계산을 보다 효율적으로 할 수 있는 D-클래스 계산에 대하여 논한다.

본 논문의 구성은 다음과 같다. 2장은 관련연구에 대하여 기술하며 3장은 논리 대수의 연산 특성을 이용하여 모든 $n \times n$ 불리언 행렬 곱셈과 D-클래스 계산을 보다 효율적으로 할 수 있는 수학적 이론을 정립하고 이를 이용한 D-클래스 계산에 대하여 기술한다. 4장은 제시한 이론을 $n \times n$ 불리언 행렬의 곱셈과 D-클래스 계산에 적용했을 때 예상되는 실행결과를 기술하며 5장은 결론 및 향후 연구방향에 대하여 논한다.

2. 관련 연구

불리언 행렬은 많은 분야에 응용되어 사용되고 있다[9-19]. 그러나 이러한 응용의 대부분은 두 불리언 행렬 사이의 효율적인 곱셈에 관심을 두고 있어[13-19] 불리언 행렬의 곱셈에 대한 모든 연구가 두 불리언 행렬의 최적 곱셈에 집중되었고 [7, 8]과 같은 서로 다른 방식의 두 불리언 행렬에 대한 최적 곱셈 알고리즘이 제시되었다. 그러나 D-클래스 계산은 다른 응용과 달리 $n \times n$ 불리언 행렬의 전체 집합을 대상으로 이 집합으로부터 조합할 수 있는 모든 행렬 사이의 이중 삼중 곱셈을 요구한다. 이러한 불리언 행렬 곱셈은 NP-완전 문제이며 적은 필요성으로 인해 관심밖에 있어 현재 이에 대해 알려진 연구결과가 없다.

최근 D-클래스 계산에 대한 연구[1-5]가 수행되었으나 매우 기초적이고 국부적인 연구로서 실행시간 개선이 매우 미흡하거나 과도한 메모리 요구 등의 문제로 인해 새로운 D-클래스의 계산에 연구결과를 활용하기 어렵다.

```
for each A in M_n(F)
    D_A = φ
    for each X in M_n(F)
        C ← AX
        for each Y in M_n(F)
            if A = CY
                for each U in M_n(F)
                    B ← UC
                    for each V in M_n(F)
                        if C = VB
                            insert B to D_A
```

[그림 1] 정의기반 단순 알고리즘

3. 논리 대수의 연산 특성을 이용한 D-클래스 계산

D-클래스 계산은 NP-완전 문제이다. 따라서 본 논문은 D-클래스 계산 성능에 대하여 논할 때 각 불리언 행렬 크기에서 실제 성취될 수 있는 실행 성능을 대상으로 한다.

[그림 1]은 위의 D-클래스 정의를 그대로 적용하여 설계된 단순한 D-클래스 계산 알고리즘이다. 알고리즘은 $n \times n$ 불리언 행렬을 대상으로 할 때 다섯 개의 for 순환문에서 전체 불리언 행렬 개수만큼, 즉 2^{n^2} 번씩 반복 순환하므로 2^{5n^2}에 비례하는 계산복잡도를 보인다. 이러한 계산복잡도로 인해 현재 4 × 4 이하 크기의 행렬에 대한 D-클래스 만이 알려져 있다. 본 장에서는 5 × 5 이상 크기의 행렬에 대한 D-클래스 계산을 가능하게 하기 위해 논리 대수의 연산 특성을 이용하여 모든 $n \times n$ 불리언 행렬 곱셈과 D-클래스 계산을 보다 효율적으로 할 수 있는 수학적 이론을 정립하고 이를 이용한 D-클래스 계산에 대하여 논한다. 본 논문은 설명을 위해 용어와 기호를 다음과 같이 정의한다. A가 $n \times n$ 불리언 행렬일 때 A_i와 A^i는 각각 A 행렬의 i 번째 행과 열을 의미한다.

$$A^T = (a_{ji}) \ \ where \ \ A = (a_{ij}),\ 0 \le i, j \le n-1$$

v 는 원소가 F에 속하는 n 차원 벡터로서 $n \times n$ 불리언 행렬의 행에 대응하여 행벡터로 부르며 $1 \times n$ 불리언 행렬과 동등하게 다룬다. v^T 는 v 의 열과 행번호를 바꾼 $n \times 1$ 행렬과 같으며 $n \times n$ 불리언 행렬의 열에 대응하여 열벡터로 부른다.

$$v = (e_0\ e_1 \Lambda\ \ e_{n-1}) \ \ \ where \ \ e_i \in F,\ 0 \le i \le n-1$$

$$v^T = \begin{pmatrix} e_0 \\ e_1 \\ \mathrm{M} \\ e_{n-1} \end{pmatrix} \ \ where \ \ \ v = (e_0\ e_1 \Lambda\ \ e_{n-1})$$

V 는 모든 행벡터의 집합이며 $(V)_n$ 는 행벡터를 n 개 조합하여 생성한 모든 $n \times n$ 불리언 행렬의 집합이다. V^T 는 모든 열벡터의 집합이며 $(V^T)^n$ 은 n 차원 열벡터를 n 개 조합하여 생성한 모든 $n \times n$ 불리언 행렬의 집합이다.

$$V = \{(e_0\ e_1\ \Lambda\ \ e_{n-1}) |\ e_i \in F \ for \ 0 \le i \le n-1\}$$

$$\begin{bmatrix} v_0 \\ v_1 \\ \mathrm{M} \\ v_{n-1} \end{bmatrix} = \begin{bmatrix} e_0^0 & e_1^0 & \Lambda & e_{n-1}^0 \\ e_0^1 & e_1^1 & \Lambda & e_{n-1}^1 \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} \\ e_0^{n-1} & e_1^{n-1} & \Lambda & e_{n-1}^{n-1} \end{bmatrix}$$

$$where \ \ v_i \in V, e_j^i \in v_i \ \ for \ 0 \le i, j \le n-1$$

$$(V)_n = \{ \begin{bmatrix} v_0 \\ v_1 \\ \mathrm{M} \\ v_{n-1} \end{bmatrix} | \ v_i \in V \ \ for \ 0 \le i \le n-1 \}$$

$$V^T = \{(e_0\ \ e_1\ \Lambda\ \ e_{n-1})^T \mid \ e_i \in F \ for \ 0 \le i \le n-1\}$$

$$[v_0^T\ v_1^T \Lambda\ v_{n-1}^T] = \begin{bmatrix} e_0^0 & e_0^1 & \Lambda & e_0^{n-1} \\ e_1^0 & e_1^1 & \Lambda & e_1^{n-1} \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} \\ e_{n-1}^0 & e_{n-1}^1 & \Lambda & e_{n-1}^{n-1} \end{bmatrix}$$

$$where \ \ v_i \in V, e_j^i \in v_i \ \ for \ 0 \le i, j \le n-1$$

$$(V^T)^n = \{ [v_0^T\ v_1^T \Lambda\ v_{n-1}^T] | \ v_i \in V \ \ for \ 0 \le i \le n-1 \}$$

$n \times n$ 불리언 행렬과 n 차원 열벡터의 곱셈은 n 차원 열벡터를, n 차원 행벡터와 $n \times n$ 불리언 행렬의 곱셈은 n 차원 행벡터를 생성한다.

$$Av^T = (A_0 v^T\ A_1 v^T\ \Lambda\ \ A_{n-1} v^T)^T \ \ where \ \ v \in V$$

$$vA = (vA^0\ vA^1\ \Lambda\ \ vA^{n-1}) \ \ where \ \ v \in V$$

3.1 모든 불리언 행렬에 대한 곱셈

위의 D_A 정의에서 $AX = C$ 를 $UC = B$ 에, $VB = C$ 를 $CY = A$ 에 대입하면 다음 D_A 정의를 얻는다.

$$D_A = \{\ B \in M_n(F) :\ \ \exists\ X, Y, U, V \in M_n(F)$$
$$such \ that \ UAX = B, VBY = A\ \}$$

이 정의에 따라 D_A 를 계산하는 간단한 방법은 먼저 불리언 행렬 A 에 $M_n(F)$ 에 속한 각 불리언 행렬 U와 X를 곱하여 UAX 를 계산하고 이렇게 얻은 행렬 B를 다시 $M_n(F)$ 에 속한 각 불리언 행렬 V와 Y에 대하여 VBY 를 계산한 행렬 A' 을 하나씩 얻어서 A 와 A' 이 같은지 비교하는 것이다. 이 경우 UAX 와 VBY 가 순차적으로 계산되고 있으나 UAX 와 VBY 는 세 불리언 행렬 곱셈의 전처리(preprocessing) 또는 병렬처리 등의 방법을 통해 독립적으로 계산할 수도 있다. 어느 경우든 D_A 계산에 가장 기본적으로 요구되는 것은 하나의 $n \times n$ 불리언 행렬과 $M_n(F)$의 모든 불리언 행렬과의 곱셈과 이 결과에 다시 $M_n(F)$ 의 모든 불리언 행렬을 곱하는 이중 곱셈을 효율적으로 하는 것이다.

다음 정리는 논리 대수의 연산 특성을 이용하여 유도 되었으며 하나의 $n \times n$ 불리언 행렬과 $M_n(F)$ 의 모든 불리언 행렬의 곱셈이 벡터집합을 이용하여 효율적으로 계산될 수 있는 기반을 제공한다.

[정리 1] $M_n(F)$에 속한 임의의 $n \times n$ 불리언 행렬 A 에 대해 M_A 와 T_A 를 $\mathrm{M}_A = \{ AX \mid X \in M_n(F) \}$,

$T_A = \{ Av^T \mid v \in V \}$로 정의하면 $\mathrm{M}_A = (T_A)^n$이다.

(증명) $P \in \mathrm{M}_A$이라 하자. 이 경우 P의 열 P^i는 어떤 $v \in V$에 대해 $P^i = Av^T$이다. T_A는 불리언 행렬 A에 n개의 원소를 가지는 모든 벡터 v^T를 곱하여 얻은 열벡터의 전체 집합이므로 0과 $n-1$ 사이의 모든 i에 대해 $P^i \in T_A$ 이며, 따라서 $P \in (T_A)^n$이다.

$P \in (T_A)^n$ 이라 하자. 만약 $P \notin \mathrm{M}_A$ 이라면 M_A에 속하는 모든 불리언 행렬 D에 대해 $P \neq D$이고 따라서 P의 어떤 열벡터 P^i가 모든 행렬 D의 i번째 열벡터 D^i와 달라야 한다. $M_n(F)$는 모든 $n \times n$ 불리언 행렬의 집합이므로 $M_n(F) = (V^T)^n$이고 $X \in M_n(F)$이므로

$$\mathrm{M}_A = \{ [Av_0^T \Lambda \ Av_{n-1}^T] \mid v_i \in V \ for \ 0 \le i \le n-1 \}$$

이다. 가정에 의해 $P \in (T_A)^n$이므로 V의 어떤 벡터 v에 대하여 $P^i = Av^T$이므로 P가 M_A에 속하게 되어 모순이 된다.』

[정리 2] $M_n(F)$에 속한 임의의 $n \times n$ 불리언 행렬 A에 대해 M_A와 T_A를 $\mathrm{M}_A = \{UA \mid U \in M_n(F)\}$, $T_A = \{vA \mid v \in V\}$로 정의하면 $\mathrm{M}_A = (T_A)_n$이다.

(증명) $P \in \mathrm{M}_A$이라 하자. 이 경우 P의 행 P_i는 어떤 $v_k \in V (0 \le k \le 2^n - 1)$에 대해 $P_i = v_k A$ 이다. T_A는 n개의 원소를 가지는 모든 벡터 v를 불리언 행렬 A에 곱하여 얻은 행벡터의 전체 집합이므로 0과 $n-1$ 사이의 모든 i에 대해 $P_i \in T_A$이며, 따라서 $P \in (T_A)_n$이다.

$L \in (T_A)_n$ 이라 하자. 만약 $L \notin \mathrm{M}_A$ 이라면 M_A에 속하는 모든 불리언 행렬 D에 대해 $L \neq D$이고 따라서 L의 어떤 행벡터 L_i가 모든 행렬 D의 i번째 행벡터 D_i와 다르다. $M_n(F)$는 모든 $n \times n$ 불리언 행렬의 집합이므로 $M_n(F) = (V)_n$이고 $U \in M_n(F)$이므로

$$\mathrm{M}_A = \{ \begin{bmatrix} v_0 A \\ v_1 A \\ \mathrm{M} \\ v_{n-1} A \end{bmatrix} \mid v_i \in V \ for \ 0 \le i \le n-1 \}$$

이다. 가정에 의해 $L \in (T_A)_n$이므로 V의 어떤 벡터 $v_k \ (0 \le k \le 2^n - 1)$에 대하여 $L_i = v_k A$이므로 L이 M_A에 속하게 되어 모순이 된다.』

[정리 1]은 $M_n(F)$의 어떤 불리언 행렬 A에 V^T의 2^n개 열벡터를 곱하여 나오는 열벡터의 집합을 얻으면, 이 집합에 있는 열벡터를 n번 조합하여 얻을 수 있는 모든 행렬의 집합이 A에 $M_n(F)$의 모든 행렬을 직접 곱하여 얻는 행렬의 집합과 동일하다는 것을 보이고 있다. [정리 2]도 유사하게 $M_n(F)$의 모든 불리언 행렬을 $M_n(F)$의 모든 2^{n^2}개 불리언 행렬을 곱하는 대신 V의 모든 행벡터에 곱하여 행벡터의 집합을 얻으면 동일한 결과를 얻을 수 있음을 보인다. [정리 1]과 [정리 2]는 하나의 $n \times n$ 불리언 행렬과 $M_n(F)$의 2^{n^2}개 불리언 행렬의 곱셈을 하나의 $n \times n$ 불리언 행렬과 V의 2^n개 벡터의 곱셈으로 대신할 수 있도록 함으로써 효율적인 D-클래스 계산을 위한 기반을 제공한다.

3.2 D-클래스 계산

3.1절의 정의에 따른 D-클래스 계산은 $M_n(F)$의 모든 불리언 행렬에 대하여 UAX와 VBY 계산이 필요하다. 따라서 같은 불리언 행렬 사이의 곱셈 계산이 여러 번 중복될 수 있으나 중복 계산은 효율적인 계산을 위해 가능한 한 피해야 한다. 다음 정리는 이러한 중복계산을 줄여 주는데 도움이 된다.

[정리 3] $B \in D_A$ *if and only if* $B^T \in D_{A^T}$.

(증명) $B \in D_A$ 이라 가정하자. 이 경우 D-클래스 정의에 의해 $AX = C$, $CY = A$, $UC = B$, $VB = C$인 관계를 만족하는 불리언 행렬 X, Y, U, V, C가 $M_n(F)$에 존재한다. 따라서 $A^T X^T = C^T$, $C^T Y^T = A^T$, $U^T C^T = B^T$, $V^T B^T = C^T$ 이고, X^T, Y^T, U^T, V^T, C^T가 $M_n(F)$에 존재하므로 $B^T \in D_{A^T}$ 이

다. $B^T \in D_{A^T}$ 이면 $B \in D_A$ 인 경우도 유사하게 증명된다.』

4. 성능개선 분석

[정리 1-2]는 벡터집합을 이용하여 두 불리언 행렬 사이의 곱셈을 가능하게 한다. 따라서 하나의 $n \times n$ 불리언 행렬에 2^{n^2} 개의 불리언 행렬을 곱하는 대신 2^n 개의 n 차원 벡터를 곱하여 결과를 얻을 수 있어 실제 실행시간에 상당한 성능 개선을 가져온다. 또한 곱셈 결과로 n 차원 벡터의 집합을 생성하여 이 집합에 속한 벡터들을 모든 가능한 방법으로 n 번 조합하면 결과로 얻어야 할 모든 $n \times n$ 불리언 행렬을 간접적으로 얻을 수 있어 최악의 경우에도 $2^n \times n$ 에 비례하여 메모리 공간이 요구되므로 불리언 행렬의 집합을 직접 나타낼 때 최악의 경우 요구되는 메모리 공간이 $2^{n^2} \times n^2$ 에 비례하는 것과 비교하면 대폭 개선된 것을 알 수 있다.

$M_n(F)$ 의 불리언 행렬 M에 대해 $M_n(F)$ 에 속한 다른 두 불리언 행렬 C와 D를 곱하여 얻을 수 있는 모든 CMD 불리언 행렬의 집합을 다음과 같이 정의하자.

$$S_M = \{CMD \mid C, D \in M_n(F)\}$$

하나의 불리언 행렬에 대한 S_M 을 계산할 때 직접 불리언 행렬 사이의 곱셈을 수행하는 경우 각 불리언 행렬마다 평균 $2^{n^2} \times 2^{n^2}$ 에 비례한 불리언 행렬 사이의 곱셈이 필요하나 [정리 1-2]를 이용하면 m 이 $\{MD \mid D \in M_n(F)\}$ 의 원소 개수일 때($0 < m \le 2^{n^2}$) 평균 $2^n(m+2^n)$ 에 비례한 행렬과 벡터 사이의 곱셈을 수행하여 실제 실행시간의 상당한 개선을 가져온다.

3.1절의 D_A 정의를 분석해 보면 A와 동치관계에 있는 불리언 행렬 B를 찾기 위해 UAX 와 VBY 를 순차적으로 반복 계산하는 대신, $M_n(F)$ 의 모든 불리언 행렬 M에 대해 S_M 을 계산한 후 A가 S_B 에 속하고 B가 S_A 에 속하는지를 보고 A와 B가 동치관계인 것을 알 수 있으며 이러한 사실은 쉽게 증명된다. 이와 같이 $M_n(F)$ 의 모든 불리언 행렬 A에 대해 D_A 를 계산하는 경우 VBY 에 해당하는 불리언 행렬의 중복된 곱셈을 피할 수 있는 장점이 있으나 중간결과를 저장하기 위해 메모리 공간이 필요하다. 최악의 경우 각 불리언 행렬에 대해 필요한 메모리 공간은 직접 불리언 행렬 곱셈을 수행할 때 $2^{n^2} \times n^2$ 에 비례하나 [정리 1-2]를 이용하면 벡터집합으로 중간결과를 저장함으로써 중간결과를 저장하기 위한 메모리 공간이 ${}_nC_{\lfloor n/2 \rfloor} \times 2^n \times n$ 에 비례하도록 줄어든다.

또한 D_A 계산 도중에 A에 동치관계인 B를 찾을 경우 [정리 3]에 의해 A^T 와 B^T 가 동치관계인 것을 알 수 있으므로 A^T 와 B^T 를 제외시키고 D_A 계산을 수행해 나가면 되므로 보다 실행시간을 개선시킨다.

5. 결론 및 향후 연구방향

D-클래스 계산은 NP-완전 문제로서 현재 4 × 4 이하 크기의 행렬에 대한 D-클래스 만이 알려져 있어 D-클래스의 특성에 대한 연구가 제대로 이루어지지 못하고 있다. 최근 새로운 D-클래스를 얻기 위하여 보다 효율적인 D-클래스 계산에 대한 연구가 수행되었으나 아직 새로운 D-클래스를 계산할 수 있을 정도의 효율적인 연구결과는 보이지 않고 있다. 본 논문은 논리대수의 연산 특성을 이용하여 D-클래스 계산을 보다 효율적으로 할 수 있는 수학적 이론과 방법을 제시하고 예상되는 성능개선에 대하여 논하였다.

본 논문의 결과는 단지 시작일 뿐이며 앞으로 효율적인 D-클래스 계산을 위한 이론, 이론을 적용한 알고리즘의 설계 및 구현, 알고리즘의 최적화, 병렬처리의 적용 등에 대한 많은 연구가

필요하다.

[참고문헌]

[1] 신철규, 한재일, “그리드 컴퓨팅 환경에서의 D-클래스 계산 병렬 알고리즘”, 한국정보처리학회 춘계학술대회 논문집, 제12권, 제1호, pp. 929-932, 2005

[2] 신철규, 한재일, “내부 순환문 개선을 통한 Linux 기반의 D-클래스 계산 고효율 순차 알고리즘”, 한국SI학회 춘계학술대회 논문집, pp. 526-531, 2005

[3] 신철규, 한재일, “공유 메모리 기반의 고성능 D-클래스 계산 병렬 알고리즘”, 한국컴퓨터종합학술대회 논문집, 제32권, 제1호, pp. 10-12, 2005

[4] 신철규, 한재일, “D-클래스 계산을 위한 병렬 알고리즘”, 한국정보처리학회 추계학술대회 논문집, 제11권, 제2호, pp. 1009-1012, 2004

[5] 신철규, 한재일, “분산 메모리 MIMD 구조를 기반으로 한 D-클래스 계산 Java 패키지”, 한국SI학회 추계학술대회 논문집, pp. 479-485, 2004

[6] Rim, D. S., and Kim, J. B., "Tables of D-Classes in the semigroup Bnof the binary relations on a set X with n-elements,"Bull. Korea Math. Soc., Vol. 20 No. 1, pp. 9-13, 1983

[7] Angluin, D., "The four Russians'algorithm for boolean matrix multiplication is optimal in its class," ACM SIGACT News, Vol. 8 No. 1, pp. 29-33, 1976

[8] Booth, K. S., "Boolean matrix multiplication using only $O(n^{\log_2 7}\log n)$ bit operations," ACM SIGACT News, Vol. 9 No. 3, pp. 23, 1977

[9] Atkinson, D. M., Santoro, N., and Urrutia, J., "On the integer complexity of Boolean matrix multiplication," ACM SIGACT News, Vol. 18 No. 1, pp. 53, 1986

[10] Yelowitz, L., "A Note on the Transitive Closure of a Boolean Matrix," ACM SIGMOD Record, Vol. 25 No. 2, pp. 30, 1978

[11] Comstock, D. R., "A note on multiplying Boolean matrices II," CACM, Vol. 7 No. 1, pp. 13, 1964

[12] Macii, E., "A Discussion of Explicit Methods for Transitive Closure Computation Based on Matrix Multiplication," 29th Asilom Conference on Signals, Systems and Computers, Vol 2, pp. 799-801, 1995

[13] Lee, L., "Fast context-free grammar parsing require fast Boolean matrix multiplication," JACM, Vol. 49 No. 1, pp. 1-15, 2002

[14] Satta, G., "Tree-adjoining grammer parsing and Boolean matrix multiplication,"Computational Linguistics, Vol. 20. No. 2, pp. 173-191,1994

[15] Martin, D. F., "A Boolean matrix method for the computation of linear precedence functions," CACM, Vol. 15 No. 6, pp. 448-454,1972

[16] Nakamura, Y., and Yoshimura T., "A partitioning-based logic optimization method for large scale circuits with Boolean matrx," Proceedings of the 32nd ACM/IEEE conference on Design automation, pp. 653-657, 1995

[17] Pratt, V. R., "The Power of Negative Thinking in Multiplying Boolean matrices,"Proceedigs of the annual ACM symposium on Theory of computing, pp. 80-83, 1974

[18] Yi, X., et al., "Fast Encryption for Multimedia," IEEE Transactions on Consumer Electronics, Vol. 47, No. 1, pp. 101-107, 2001

[19] Alon, N., and Galil, Z., "Witnesses for Boolean Matrix Multiplication and for Shortest Paths," 33rd Annual Symposium on Foundations of Computer Science, pp. 417-426, 1992