韓國IT서비스學會誌
第6卷 第1號
2007年 4月

효율적인 D-클래스 계산을 위한 알고리즘*

한 재 일**

Algorithm for Efficient D-Class Computation*

Jae-Il Han**

■ Abstract ■

D-class computation requires multiplication of three Boolean matrices for each of all possible triples of $n \times n$ Boolean matrices and search for equivalent $n \times n$ Boolean matrices according to a specific equivalence relation. It is easy to see that even multiplying all $n \times n$ Boolean matrices with themselves shows exponential time complexity and D-Class computation was left an unsolved problem due to its computational complexity. The vector-based multiplication theory shows that the multiplication of three Boolean matrices for each of all possible triples of $n \times n$ Boolean matrices can be done much more efficiently. However, D-Class computation requires computation of equivalent classes in addition to the efficient multiplication. The paper discusses a theory and an algorithm for efficient D-class computation, and shows execution results of the algorithm.

Keyword : D-Class, Boolean Matrix, Algorithm, Computational Complexity, NP-Complete

* 본 연구는 2007년도 국민대학교 교내연구비를 지원받아 수행되었음.
** 국민대학교 전자정보통신대학 컴퓨터공학부

1. 서 론

불리언 행렬은 원소가 0(거짓)이나 1(참) 값을 갖는 행렬이다. $F=\{0,1\}$, $M_n(F)$를 원소가 F에 속하는 모든 $n\times n$ 불리언 행렬의 집합, A, B를 $M_n(F)$에 속하는 임의의 $n\times n$ 불리언 행렬이라 할 때 동치관계(equivalence relation) $\sim_R$와 D-클래스는 다음과 같이 정의된다[7].

$$A\sim_R B \text{ if } \exists X, Y, U, V\in M_n(F) \text{ such that } UAX=B \text{ and } VBY=A.$$

$$D_A=\{B\in M_n(F) \mid A\sim_R B\}$$

최근 제시된 벡터기반의 불리언 행렬 중첩곱셈[1, 2]은 이러한 모든 세 $n\times n$ 불리언 행렬 사이의 효율적 중첩곱셈을 수행할 수 있는 이론과 알고리즘을 제시하였으며 아직 다른 연구는 보이지 않고 있다. 본 논문은 벡터기반의 불리언 행렬 중첩곱셈 이론을 이용하여 D-클래스 계산 알고리즘을 설계하고 그 실행결과에 대하여 논한다.

본 논문의 구성은 다음과 같다. 제 2장은 기존에 제시된 두 불리언 행렬 사이의 곱셈 알고리즘을 중첩곱셈에 적용할 때의 문제점을 살펴보고, 제 3장은 본 논문에서 사용할 용어와 기호를 정의한다. 제 4장은 [1, 2]에서 제시한 수학적 이론과 이를 이용하여 설계한 D-클래스 알고리즘을 기술하고, 제 5장은 알고리즘의 실행결과에 대하여 논한다. 제 6장은 결론 및 향후 연구방향에 대하여 논한다.

2. 관련 연구 및 문제점

불리언 행렬에 대한 연구는 대부분 두 불리언 행렬의 효율적인 곱셈에 초점을 두고 있다. 행을 이용한 곱셈 알고리즘 중에서는 $O(n^2/\log n)$번의 행 OR-연산 알고리즘[3]이, 비트 기반 연산을 이용한 알고리즘 중에서는 $O(n^{\log_2 7}\log n)$번의 비트 연산 알고리즘[4]이 현재 최적의 알고리즘으로 나타나고 있다. 그러나 이 알고리즘들은 $n\times n$ 불리언 행렬을 대상으로 단지 두개의 불리언 행렬 곱셈만을 다루고 있으며, 행렬 곱셈을 수행하기 전에 주어진 특정 행렬에 대해서 행 OR-연산이나 비트 연산에 필요한 정보를 미리 계산하여 저장할 것을 요구한다. 하나의 $n\times n$ 불리언 행렬과 모든 $n\times n$ 불리언 행렬의 곱셈을 하는 경우 위의 두 알고리즘 중 어떤 알고리즘이 사용되는가에 상관없이 2^{n^2}번의 두 불리언 행렬 곱셈이 요구되며 곱셈 결과로 나오는 불리언 행렬을 저장할 때 최악의 경우 2^{n^2}에 비례하는 메모리 공간이 필요하다. 따라서 하나의 불리언 행렬과 모든 불리언 행렬의 곱셈에 두 불리언 행렬의 최적 곱셈 알고리즘을 그대로 적용하여 각각의 두 불리언 행렬 곱셈을 하는 경우 효율적인 곱셈이 어렵다. 더구나 D-클래스 계산과 같이 모든 $n\times n$ 불리언 행렬과 모든 $n\times n$ 불리언 행렬의 곱셈이 요구되는 경우 위의 두 알고리즘은 두 불리언 행렬 곱셈이 수행되기 전에 요구되는 정보를 생성하기 위하여 각 불리언 행렬마다 [3]은 최악의 경우 $O(2^n)$, [4]는 $O(n^2\log n)$의 계산 시간이 추가적으로 필요하다. 따라서 [3, 4]의 알고리즘은 D-클래스의 계산에 사용되기 어렵다.

$n\times n$ 불리언 행렬에 모든 $n\times n$ 불리언 행렬을 이중으로 연속 곱하는 경우 불리언 행렬을 직접 곱하면 항상 $2^{n^2}\cdot 2^{n^2}$에 비례하는 시간 복잡도를 가지나, 최근 제시된 벡터기반의 불리언 행렬 곱셈[1]과 중첩곱셈 개선 이론[2]을 적용할 경우 최악의 경우에도 $2^n\cdot 2^{n^2}\cdot 2^n$에 비례하는 시간 복잡도를 가진다. 또한 불리언 행렬 직접 곱셈은 공간 복잡도가 최악의 경우 $n^2\cdot 2^{n^2}$에 비례하나 두 이론을 적용하면 $n\cdot 2^n\cdot 2^{n^2}$에 비례하여 실제 실행 시간은 상당한 개선을 보였다.

D-클래스 계산은 이와 같은 불리언 행렬 중첩곱셈의 개선에 더해 동치관계에 있는 불리언 행렬

에 대한 효율적인 계산을 요구하나 현재 특정 상황에 적용할 수 있는 연구결과만이 알려져 있다. [5]는 D-클래스 계산과 관련하여 최선의 알고리즘으로 알려져 있으나 특정 부분집합 A로부터 생성되는 유한 세미그룹 S의 D-클래스만을 다루고 있다. D-클래스는 정규(regular) D-클래스와 비정규(non-regular) D-클래스로 구성되나 [6]은 모든 $n\times n$ 불리언 행렬의 세미그룹 B_n에 있는 정규(regular) D-클래스 생성 이론만을 제시한다.

3. 용어 및 기호 정의

본 논문의 설명을 위해 다음과 같이 용어와 기호를 정의한다. 임의의 $n\times m$ 불리언 행렬 A가 주어지고 $F=\{0,1\}$이라 할 때, $M_n^m(F)$는 모든 $n\times m$ 불리언 행렬의 집합을 정의하며, A_i와 A^i는 각각 A 행렬의 i행과 i열을 의미한다. A^T는 A의 전치(transpose)행렬이며, m차원 벡터 v는

$$v=(b_0 b_1 \cdots b_{m-1}),\ b_i\in F,\ 0\le i\le m-1$$

로 정의하고 $n\times m$ 불리언 행렬의 행에 대응하여 행벡터로 부른다. v^T는 v의 열과 행 번호를 바꾼 $m\times 1$ 불리언 행렬로서

$$v^T=\begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{m-1}\end{pmatrix}\quad where\ v=(b_0 b_1 \cdots b_{m-1})$$

으로 정의되며 $m\times n$ 불리언 행렬의 열에 대응하여 열벡터로 부른다. $V(m)$은 모든 m차원 행벡터의 집합이며$(V(m))_k$는 행벡터를 k개 조합하여 생성한 모든 $k\times m$ 불리언 행렬의 집합이다. $V^T(m)$은 모든 m차원 열벡터의 집합이며$(V^T(m))^k$는 열벡터를 k번 조합하여 생성한 모든 $m\times k$ 불리언 행렬의 집합이다.

$$V(m)=\{(b_0 b_1 \cdots b_{m-1}) \mid b_i\in F \text{ for } 0\le i\le m-1\}$$

$$(V(m))_k=\left\{\begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{k-1}\end{pmatrix} \mid v_i\in V(m) \text{ for } 0\le i\le k-1\right\}$$

$$V^T(m)=\{v^T \mid v\in V(m)\}$$

$$(V(m))^k=\{(v_0^T v_1^T \cdots v_{k-1}^T) \mid v_i\in V(m) \text{ for } 0\le i\le k-1\}$$

$n\times m$ 불리언 행렬 A와 m차원 열벡터 v^T의 곱셈은 n차원 열벡터를, m차원 행벡터 v와 $m\times n$ 불리언 행렬 B의 곱셈은 n차원 행벡터를 생성한다.

$$vB=(vB^0 vB^1 \cdots vB^{n-1})\ where\ v\in V(m)$$

$$Av^T=\begin{pmatrix} A_0 v^T \\ A_1 v^T \\ \vdots \\ A_{n-1} v^T\end{pmatrix}\quad where\ v=V(m)$$

4. D-클래스 계산 알고리즘

[1]은 다음 세 개의 정리를 통해 모든 $l\times n$, $n\times m$, $m\times k$ 불리언 행렬의 이중 연속곱셈을 불리언 행렬 사이의 곱셈대신 n차원 벡터와 m차원 벡터의 곱셈을 이용하여 보다 효율적으로 같은 결과를 얻을 수 있음을 증명하고 벡터기반 중첩곱셈 알고리즘 및 실행결과를 기술하였다.

[정리 1] $M_n^m(F)$에 속한 임의의 $n\times m$ 불리언 행렬 A에 대해 R_A, C_A를

$$R_A=\{AB \mid B\in M_m^k(F)\},$$
$$C_A=\{(A_0 v^T A_1 v^T \cdots A_{n-1} v^T)^T \mid v\in V(m)\}$$

로 정의할 때 $R_A=(C_A)^k$ 이다.

[정리 2] $M_n^m(F)$에 속한 임의의 $n\times m$ 불리언 행렬 A에 대해 L_A, T_A를

$$L_A=\{BA \mid B\in M_l^n(F)\},$$

$$T_A = \{(vA^0 vA^1 \cdots vA^{m-1}) \mid v \in V(n)\}$$

로 정의할 때 $L_A = (T_A)_l$ 이다.

[정리 3] A를 $M_n^m(F)$에 속한 임의의 $n \times m$ 불리언 행렬, Z를 $M_n^k(F)$에 속한 임의의 $n \times k$ 불리언 행렬이라 할 때 S_{AX}, S_{UAX}, T_Z를

$$S_{AX} = \{AX \mid X \in M_m^k(F)\},$$
$$S_{UAX} = \{UAX \mid U \in M_l^n(F), X \in M_m^k(F)\},$$
$$T_Z = \{(vZ^0 vZ^1 \cdots vZ^{k-1}) \mid v \in V(n)\}$$

로 정의하면 $S_{UAX} = \bigcup_{Z \in S_{AX}} (T_Z)_l$ 이다.

[정리 1]은 $n \times m$ 불리언 행렬에 모든 $m \times k$ 불리언 행렬을 곱하여 얻는 불리언 행렬의 집합이 $n \times m$ 불리언 행렬에 모든 m차원 열벡터를 곱하여 얻는 열벡터들을 모든 가능한 k번의 조합으로 얻는 불리언 행렬 집합과 같다는 것을 보인다. [정리 2]는 [정리 1]과 유사하게 모든 $l \times n$ 불리언 행렬을 $n \times m$ 불리언 행렬에 곱하여 얻는 불리언 행렬의 집합이 모든 n차원 행벡터를 $n \times m$ 불리언 행렬에 곱하여 얻는 행벡터들을 모든 가능한 k번의 조합으로 얻는 불리언 행렬 집합과 같다는 것을 보인다. [정리 3]은 $n \times m$ 불리언 행렬에 모든 $l \times n$, $m \times k$ 불리언 행렬을 직접 곱하여 얻은 결과가 $n \times m$ 불리언 행렬에 모든 m차원 벡터와 n차원 벡터를 차례로 곱하여 얻은 결과와 같다는 것을 보인다. 따라서 [정리 1]에 의하여 주어진 $n \times m$ 불리언 행렬 뒤에 모든 $m \times k$ 불리언 행렬을 곱하는 대신 모든 m차원 벡터를 곱하여 n차원 열벡터 집합을 얻고 이 집합의 벡터를 k번 조합하여 $n \times m$ 불리언 행렬과 모든 $m \times k$ 불리언 행렬을 곱할 때와 같은 $n \times k$ 불리언 행렬의 집합을 얻고, 다음 [정리 2]를 적용하여 $n \times k$ 불리언 행렬의 집합의 각 불리언 행렬의 앞에 모든 n차원 벡터를 곱하여 k차원 행벡터 집합을 얻은 후 이 집합의 벡터를 l번 조합하여 $l \times k$ 불리언 행렬 집합을 얻으면 주어진 $n \times m$ 불리언 행렬과 모든 $l \times n$, $m \times k$ 불리언 행렬을 중첩하여 곱할 때와 같은 결과를 얻을 수 있다. 모든 $l \times n$, $n \times m$, $m \times k$ 불리언 행렬 사이의 연속곱셈 결과는 $M_n^m(F)$의 각 $n \times m$ 불리언 행렬에 대해 위 연속곱셈을 수행하여 얻을 수 있다.

그러나 이 경우 알고리즘을 설계할 때 각 불리언 행렬에 대하여 모든 벡터와의 첫 번째 곱셈에서 얻은 벡터 집합으로부터 불리언 행렬을 하나씩 생성하여야 하는 오버헤드를 포함하며 많은 계산이 중복된다. 이러한 중복계산을 줄이기 위해[2]는 [정리 4]를 통해 임의의 두 $n \times n$ 불리언 행렬 A, B에 모든 $n \times n$ 불리언 행렬을 곱하여 얻는 행렬 집합이 같으면 A, B에 모든 $n \times n$ 불리언 행렬을 뒤와 앞 순서로 연속 곱하여 얻는 행렬 집합도 같다는 것을 보임으로써, $M_n^n(F)$의 어떤 불리언 행렬에 모든 $n \times n$ 불리언 행렬을 곱하여 얻는 행렬 집합이 이미 존재하는 경우 모든 $n \times n$ 불리언 행렬에 대한 다음 곱셈을 수행하여 행렬 집합을 계산할 필요 없이 이전의 연속곱셈 결과를 사용하여 앞의 정리만 적용하였을 경우 나타나는 오버헤드와 많은 중복계산이 상당부분 개선됨을 보였다.

[정리 4] $M_n^n(F)$에 속한 임의의 두 불리언 행렬 A, B에 대해 S_A, S_B, T_A, T_B를

$$S_A = \{AX \mid X \in M_n^n(F)\},$$
$$S_B = \{BX \mid X \in M_n^n(F)\},$$
$$T_A = \{UAX \mid U, X \in M_n^n(F)\},$$
$$T_B = \{UBX \mid U, X \in M_n^n(F)\}$$

로 정의할 때 $S_A = S_B$이면 $T_A = T_B$이다.

본 연구는 D-클래스 계산 알고리즘 설계시 벡터기반 불리언 행렬 중첩곱셈의 적용과 더불어 다음 두 개의 정리를 적용하여 불리언 행렬의 동치관계 검사에 대한 중복계산을 줄임으로써 보다 개

선된 알고리즘을 설계 하였다.

[정리 5] $M_n^n(F)$에 속한 임의의 두 불리언 행렬 A, B에 대해 S_{UAX}, S_{VBY}를

$$S_{UAX}=\{UAX \mid U, X \in M_n^n(F)\},$$
$$S_{VBY}=\{VBY \mid V, Y \in M_n^n(F)\}$$

로 정의하면, $A \sim_R B$ *iff* $S_A = S_B$ 이다.

(증명) I가 $n \times n$ 단위행렬(identity matrix)이라고 하면, $A=IAI$, $B=IBI$이므로 $A \in S_{UAX}$, $B \in S_{VBY}$ 이다. $S_{UAX}=S_{VBY}$ 라면 $B \in S_{UAX}$, $A \in S_{VBY}$ 이므로, $UAX=B$이고 $VBY=A$를 만족하는 U, V, X, Y가 $M_n^n(F)$에 존재한다. 따라서 $A \sim_R B$ 이다.

$A \sim_R B$라고 가정하자. 그러면 정의에 의해 $UAX=B$이고 $VBY=A$인 U, V, X, Y가 $M_n^n(F)$에 존재한다. C를 S_{UAX}에 속한 임의의 불리언 행렬이라고 하면 $C=U'AX'$을 만족하는 U', X'이 항상 $M_n^n(F)$에 있으므로 A대신 VBY를 대입하여 $C=U'VBYX'$을 얻는다. 이때 $U'V$와 YX'은 모두 $M_n^n(F)$에 속하므로 $C \in S_{VBY}$이며 $S_{UAX} \subseteq S_{VBY}$이다. $S_{VBY} \subseteq S_{UAX}$도 같은 논리로 증명되며 따라서 $S_{UAX}=S_{VBY}$이다. ■

[정리 6] A, B를 $M_n^n(F)$에 속한 임의의 두 불리언 행렬이라 하고 S_{UAX}, S_{VBY}, S_{AX}, S_{BY}, M_{AX}, M_{BY}를

$$S_{UAX}=\{UAX \mid U,X \in M_n^n(F)\},$$
$$S_{VBY}=\{VBY \mid V,Y \in M_n^n(F)\},$$
$$S_{AX}=\{AX \mid X \in M_n^n(F)\},$$
$$S_{BY}=\{BY \mid Y \in M_n^n(F)\},$$
$$M_{AX}=\{C \in M_n^n(F) \mid S_{CX}=S_{AX}\},$$
$$M_{BY}=\{C \in M_n^n(F) \mid S_{CY}=S_{BY}\}$$

으로 정의할 때, $S_{UAX}=S_{VBY}$라면 $M_{AX} \cup M_{BY}$에 속한 임의의 두 불리언 행렬 K, L에 대해 $K \sim_R L$이다.

(증명) $K, L \in M_{AX}$이거나 $K, L \in M_{BY}$인 경우 [정리 4]와 [정리 5]에 의해 $K \sim_R L$이다.

$K \in M_{AX}$, $L \in M_{BY}$이라 가정하자. $S_{AX}=S_{BY}$라면 $M_{AX}=M_{BY}$가 되어 앞의 경우와 같이 [정리 4]와 [정리 5]에 의해 $K \sim_R L$ 이다. $S_{AX} \neq S_{BY}$라 가정하자. 이 경우 $M_{AX} \cap M_{BY} = \varnothing$가 된다. 주어진 조건이 $S_{UAX}=S_{VBY}$이므로 [정리 5]에 의해 $A \sim_R B$이고, $\sim_R$은 동치관계이므로 $K \sim_R L$이다. ■

[정리 5]와 [정리 6]은 임의의 두 $n \times n$ 행렬 A, B에 대해 $S_{UAX}=S_{VBY}$이면 S_{AX} 집합이나 S_{BY} 집합을 생성하는 모든 불리언 행렬이 동치관계, 즉 같은 D-클래스에 속한다는 것을 의미한다. [그림 1]은 벡터기반 불리언 행렬 중첩곱셈 이론과 [정리 5-6]을 적용하여 설계한 D-클래스 계산 알고리즘의 개요를 보이고 있다. 알고리즘은 먼저 [2]의 벡터기반 불리언 행렬 중첩곱셈 알고리즘에 의해 트리플 집합 $\mathbb{M}$을 생성한 후, $\mathbb{M}$에서 임의의 (S_c, S_M, c_A) 트리플을 선택하여 세 번째 원소가 c_A와 같은 트리플을 모두 찾아 같은 그룹으로 묶는다. 이 그룹이 하나의 D-클래스를 나타내며 그룹의 각 트리플에 있는 S_M을 유니온하여 D-클래스를 얻는다. 그룹에 속한 트리플은 $\mathbb{M}$에서 제거한다. 이 과정을 $\mathbb{M}$에 트리플이 남아 있는 동안 반복하며 루프가 종료되면 모든 D-클래스를 찾게 된다.

5. D-클래스 계산 알고리즘 및 실행결과

D-클래스 계산 알고리즘은 Java 언어로 구현하였으며 2개의 2.8GHz Zeon CPU, 1GB RAM, 250GB HDD, Fedora 9.0 환경에서 실행하였다. 불리언 행렬과 벡터의 곱셈은 행 OR-연산 기반 불리언 행렬 곱셈[3]을 비트사이의 논리연산에 적합하도록 변형하여 수행하였다. <표 1>은 불리언

```
M = ∅
for each boolean matrix A in M_n^n(F)
    S_c = ∅
    for each row vector v in V(n)
        compute an Av^T vector
        insert the vector into S_c
    if (S_c ∉ M[0])
        S_M = ∅
        insert A into S_M
        for each matrix M synthesized from S_c
            S_r = ∅
            for each row vector v in V(n)
                compute an vM vector
                insert the vector into S_r
            insert S_r into C_A
        insert a triple (S_c, S_M, C_A) into M
    else
        insert A into M(S_c)[1]
while (M ≠ ∅)
    take one S_c from M[0]
    remove M(S_c) from M
    insert M(S_c) into D_Sc
    for each triple (S'_c, S'_M, C'_A) in M
        if (C'_A = M(S_c)[2])
            remove M(S'_c) from M
            insert M(S'_c) into D_Sc
    insert D_Sc into DClassList
```

* S_c : a set of n-dimensional column vectors
* S_r : a set of n-dimensional row vectors
* S_M : a set of $n \times n$ boolean matrices
* $\mathbb{C}_A$: a class of S_r's for S_c
* $\mathbb{M}$: a set of $(S_c, S_M, \mathbb{C}_A)$ triples
* $\mathbb{M}[i]$: a set of the i'th elements in all triples
* $\mathbb{M}(S_c)$: a triple whose first element matches S_c
* $\mathbb{M}(S_c)[i]$: the i'th element in $\mathbb{M}(S_c)$
* $\mathbb{D}_{S_c}$: a set of $(S_c, S_M, \mathbb{C}_A)$ triples whose S_M belong to the same D-class

[그림 1] D-클래스 계산 알고리즘

〈표 1〉 실행시간 비교

행렬 크기	행렬기반 중첩곱셈 알고리즘	벡터기반 중첩곱셈 알고리즘	D-클래스 계산 알고리즘
2×2	0.006초	0.002초	0.02 초
3×3	9.682초	0.021초	0.049초
4×4	29312169초	1.586초	3.013초
5×5	5.1257314×10^{15} 초 이상	15596초	168877초

행렬을 직접 곱한 중첩곱셈, 벡터기반 중첩곱셈, D-클래스 계산 알고리즘의 실행시간을 보이고 있다. 행렬기반 중첩곱셈 알고리즘의 4×4 이상 크기의 실행 결과는 가장 내부에 있는 루프의 평균 실행시간을 필요한 외부 루프 수만큼 곱하여 얻은 최소 예상 실행시간이다. 모든 알고리즘의 계산복잡도는 NP-완전문제이나 <표 1>에서 보는 것처럼 D-클래스 계산 알고리즘의 실제 실행시간은 행렬중첩곱셈 알고리즘과 비교하여도 상당한 개선을 보이고 있다.

6. 결론 및 향후 연구방향

D-클래스 계산은 모든 세 $n\times n$ 불리언 행렬 사이의 중첩곱셈과 불리언 행렬의 동치관계에 대한 효율적인 계산을 요구한다. 불리언 행렬의 곱셈에 대한 많은 연구는 모두 두 불리언 행렬 사이의 최적곱셈에 관심을 두고 있으나[3, 4], 최근 제시된 벡터기반의 불리언행렬 중첩곱셈[1, 2]은 이러한 모든 세 $n\times n$ 불리언 행렬 사이의 효율적인 중첩곱셈에 대한 이론 및 알고리즘을 제시하고 있다. 본 논문은 벡터기반 불리언행렬 중첩곱셈 이론과 알고리즘을 기반으로 동치관계에 있는 불리언 행렬을 효율적으로 계산할 수 있는 이론을 제시하였으며 이를 바탕으로 설계한 D-클래스 알고리즘과 실행결과를 논하였다.

D-클래스 계산 알고리즘은 NP-완전 계산 복잡도를 가지므로 다차원 함수의 계산복잡도를 가

지는 알고리즘을 찾기 어려우나, 공간 및 시간 복잡도를 개선할 수 있는 수학적 이론을 정립함으로써 보다 나은 결과를 얻을 수 있다. 앞으로 D-클래스 계산을 최적화할 수 있는 불리언 행렬 중첩곱셈 및 동치 클래스 계산 이론에 대한 연구, 이론을 적용한 알고리즘의 설계 및 최적화, D-클래스 계산 병렬 알고리즘 등에 대한 연구가 필요하며, 이를 바탕으로 아직 알려지지 않은 D-클래스를 찾아 내어 D-클래스의 특성을 파악한 후 이를 활용할 수 있는 응용분야에 대한 연구도 요구된다.

참 고 문 헌

[1] 한재일, "모든 $1\times n$, $n\times m$, $m\times k$ 불리언 행렬 사이의 중첩곱셈에 대한 연구", 한국IT서비스학회지, 제5권, 제1호(2006), pp.1209-1220.

[2] 한재일, "모든 n 차 정사각 불리언 행렬 사이의 효율적 이중 연속곱셈에 관한 연구", *Journal of The Korean Data Analysis Society*, Vol.3, No.3(2006), pp.526-531.

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

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

[5] Froidure, V. and J. E. Pin, "Algorithms for computing finite semigroups", Cucker (ed.), Foundations of Computational Mathemtics, Berlin, Springer, 1997, pp.112-126.

[6] Kim, K. H. and F. W. Roush, "On generating regular elements in the semigroup of binary relations", Semigroup Forum No.14 (1997), pp.29-32.

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

◈ 저 자 소 개 ◈

한 재 일 (jhan@kookmin.ac.kr)

연세대학교에서 이학사, 미국 Syracuse University에서 전산학 석사와 박사학위를 취득하고, 국민대학교 컴퓨터학부 교수로 재직 중이다. 현재 분산처리, 객체지향 시스템과 RFID/USN 미들웨어를 연구 중이다. 관심분야는 분산 시스템, 객체지향 시스템, 미들웨어, 컴퓨터 및 네트워크 보안, 지능형 시스템, 공개 소프트웨어 등이다.