韓國IT서비스學會誌
第7卷 第4號
2008年 12月

불리언 행렬의 모노이드에서의 J 관계 계산 알고리즘*

한 재 일**

Algorithm for Computing J Relations in the Monoid of Boolean Matrices*

Jae-Il Han**

■ Abstract ■

Green's relations are five equivalence relations that characterize the elements of a semigroup in terms of the principal ideals. The J relation is one of Green's relations. Although there are known algorithms that can compute Green relations, they are not useful for finding all J relations in the semigroup of all n x n Boolean matrices. Its computation requires multiplication of three Boolean matrices for each of all possible triples of n x n Boolean matrices. The size of the semigroup of all n x n Boolean matrices grows exponentially as n increases. It is easy to see that it involves exponential time complexity. The computation of J relations over the 5 x 5 Boolean matrix is left an unsolved problem. The paper shows theorems that can reduce the computation time, discusses an algorithm for efficient J relation computation whose design reflects those theorems and gives its execution results.

Keyword : Semigroup, Ideal, Green Relation, Boolean matrix, Algorithm, Computational Complexity

논문투고일 : 2008년 10월 15일 논문수정완료일 : 2008년 12월 12일 논문게재확정일 : 2008년 12월 12일
* 본 연구는 2008년도 국민대학교 교내연구비를 지원받아 수행되었음.
** 국민대학교 전자정보통신대학 컴퓨터공학부

1. 서 론

대수학에서의 Green 관계(relation)는 반군(semigroup)에 속한 원소의 특성을 보여주는 다섯 개의 동치관계(equivalence relation)이다[4]. Green 관계는 반군에서의 분할성(divisibility)을 이해하는데 매우 유용한 개념으로써 변환반군(transformation semigroup), 추계적 행렬(stochastic matrix), 다항식(polynomials) 등 여러 분야에서 사용되고 있다[5-7]. 불리언 행렬은 원소가 0(거짓)이나 1(참) 값을 갖는 행렬이며, 구문분석, 선형 우선순위 함수 계산, 논리 최적화 등 다양한 분야에서 유용하게 사용되고 있다[8-12]. J 관계는 Green 관계에 속한 하나의 동치관계로써 불리언 행렬의 반군에서의 J 관계는 암호학 등 여러 분야에 응용할 수 있는 가능성이 있으나, 불리언 행렬 곱셈에서 야기되는 기하급수적인 계산복잡도로 인해 극히 제한된 크기의 불리언 행렬에 대한 결과만이 알려져 있다[1, 13].

본 논문은 $n \times n$ 불리언 행렬의 반군에서의 J 관계 특성을 밝히기 위해 벡터 기반의 불리언 행렬 곱셈 이론[2]과 반군에서의 Green 관계에 대한 정리를 이용하여 보다 개선된 모든 J 관계의 계산 알고리즘을 제시한다. 본 논문의 구성은 다음과 같다. 제 2장은 기존에 제시된 불리언 행렬 사이의 곱셈 알고리즘과 Green 관계 계산 알고리즘을 살펴보고, 제 3장은 본 논문에서 사용할 용어와 기호를 정의한다. 제 4장은 [2]에서 제시한 벡터기반 불리언 행렬 곱셈 이론과 반군에서의 Green 관계에 대한 정리를 기반으로 보다 효율적인 J 관계 계산을 위한 수학적 이론과 이를 적용하여 설계한 J 관계 계산 알고리즘을 기술하고, 제 5장은 알고리즘의 계산복잡도와 실행결과에 대하여 논한다. 제 6장은 결론 및 향후 연구방향에 대하여 논한다.

2. 관련 연구

불리언 행렬은 여러 분야에서 유용하게 사용되고 있다[8-12]. 이러한 대부분의 응용은 두 불리언 행렬의 효율적인 곱셈에 초점을 두고 있으며 많은 연구를 통해 여러 응용에 적합한 다양한 알고리즘이 제시되었다[14-21]. 반면 모든 불리언 행렬 사이의 곱셈은 극히 소수의 연구[2, 3]에서 다루고 있으며, 모든 불리언 행렬의 중첩곱셈에 대한 연구는 기하급수적 계산복잡도와 효율적 곱셈의 어려움, 상대적으로 적은 응용 분야 등으로 인해 관심 밖에 있다. 그러나 D-클래스[13]와 같은 계산은 이러한 중첩곱셈을 요구하며 기하급수적 계산의 어려움으로 인해 극히 제한된 결과만이 알려져 있다. [3]은 [2]에서 제시한 벡터기반 불리언 행렬 곱셈 알고리즘을 기반으로 중복된 중첩곱셈을 최소화하여 D-클래스 계산을 보다 최적화한 알고리즘을 기술하였다.

반군에서의 Green 관계나 다른 구조적 특성을 연구하기 위해 여러 알고리즘이 제시되었다[20-26]. 그러나 이 연구들은 모두 변환반군만을 고려하였으며 제시된 알고리즘은 변환반군의 특성에 의존한다. [25]는 불리언 행렬 반군에서의 알고리즘을 보이고 있으나 변환반군의 특성에 의존한다. 그러나 [26-30]에서 보듯이 변환반군이 아닌 불리언 행렬이나 다른 행렬의 반군도 매우 자주 나타난다. [29]는 변환반군이나 행렬의 반군 등 반군 원소의 특성에 상관없이 주어진 원소의 집합으로부터 반군을 생성할 수 있는 범용 알고리즘을 제시하였다. 이 알고리즘은 현재 유사 알고리즘 중에서 가장 빠른 알고리즘으로 알려져 있으며 반군 S와 원소 집합 A에 대해 $O(|S||A|)$의 시간복잡도(time complexity)를 갖는다.

[29]의 알고리즘은 Green 관계의 계산을 위한 범용 알고리즘에 사용 될 수 있다. 그러나 불리언 행렬 반군에서의 Green 관계 계산은 모든 불리언 행렬 사이의 곱셈이 기본적으로 요구되어 [29]의 알고리즘만으로는 효율적인 계산을 할 수 없다. $n \times n$불리언 행렬의 반군은 $|S| = 2^{n^2}$개의 원소가 존재하여 Green 관계를 계산하려면 $O(2^{n^2})$의 시간복잡도를 보이기 때문이다. 더구나 [29]의 알고

리즘은 주어진 반군에서 하나의 Green 관계 계산에 대해 논하고 있으며 모든 Green 관계 계산은 다루고 있지 않다. Green 관계를 구성하는 다섯 동치관계 중 모든 J 관계나 D 관계의 계산은 다른 동치관계와 달리 모든 불리언 행렬 사이의 이중 연속곱셈이 요구되어, 계산복잡도는 다른 동치관계와 같이 $O(2^{n^2})$이나 실제 계산시간은 다른 동치관계보다 훨씬 많이 소요된다. 따라서 불리언 행렬의 반군에서 모든 J 관계를 효율적으로 계산할 수 있는 알고리즘에 대한 연구가 필요하다.

3. 정 의

본 논문은 다음과 같이 용어와 기호를 정의한다. 이항연산 $\cdot$이 하나 주어진 집합 S가 다음 특성을 만족하면 반군(semigroup)으로 부른다.

$x \cdot y \in S$ for any $x, y \in S$
$x \cdot (y \cdot z) = (x \cdot y) \cdot z$ for any $x, y, z \in S$

모든 $a \in S$에 대해서 $a \cdot e = e \cdot a = a$ 인 원소 e를 단위 원소(identity element)라고 하며, $e \cdot e = e$인 원소 $e \in S$를 멱등원(idempotent element)이라 부른다. 단위원소를 가지는 반군 M은 모노이드(monoid)로 정의된다.

$I \subseteq S$일 때, 모든 $x \in I$와 $s \in S$에 대해 $x \cdot s \in I$이면 I를 우 아이디얼(right ideal)이라 부르고 $s \cdot x \in I$이면 I를 좌 아이디얼(left ideal)이라 부른다. 주 아이디얼(principal ideal)은 하나의 원소 $a \in S$에 의해 생성되는 아이디얼을 의미하며, a에 대해 S의 모든 원소를 연산한 결과의 집합으로써 구체적으로 다음과 같이 정의된다. 하나의 원소 $a \in S$에 대하여 $Sa = \{sa : s \in S\}$, $aS = \{as : s \in S\}$는 각각 a에 의해 생성되는 좌 주 아이디얼(left principal ideal), 우 주 아이디얼(right principal ideal)이며, $SaS = \{s_1 a s_2 : s_1, s_2 \in S\}$는 a에 의해 생성되는 양측 주 아이디얼(two-sided principal ideal)이고, a는 자신이 생성하는 각 아이디얼의 생성자(generator)로 부른다.

M을 유한개의 원소를 갖는 모노이드, a, b를 M에 속한 임의의 두 원소라고 할 때 Green 관계는 다음 다섯 개의 R, L, J, H, D 관계로 정의된다.

aRb if and only if $aM = bM$
aLb if and only if $Ma = Mb$
aJb if and only if $MaM = MbM$
aHb if and only if aRb and aLb
aDb if and only if there exists a c in S such that aRc and cLb

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, m 차원 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 m 차원 행벡터의 집합이며 $(V(m))_k$는 행벡터를 k개 조합하여 생성한 모든 $k \times m$ 불리언 행렬의 집합이다. $V^T(m)$은 모든 m 차원 열벡터의 집합이며 $(V^T(m))^k$는 열벡터를 k 번 조합하여 생성한 모든 $m \times k$ 불리언 행렬의 집합이다.

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

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

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

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

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

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

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

4. J 관계 계산 알고리즘

본 장은 $M_n^n(F)$를 대상으로 모든 J 관계를 효율적으로 계산할 수 있는 알고리즘에 대하여 기술한다. $M_n^n(F)$는 모든 $n \times n$ 불리언 행렬의 집합이다. $M_n^n(F)$의 두 불리언 행렬의 연산을 불리언 연산에 의한 행렬 곱셈으로 정의하면, 모든 $A, B \in M_n^n(F)$에 대해 $A \cdot B \in M_n^n(F)$이고 $A \cdot (B \cdot C) = (A \cdot B) \cdot C$이며 단위행렬(identity matrix)를 가지고 있으므로 모노이드임을 알 수 있다.

[정리 1] M 이 주기적 반군(periodic semigroup)이라면 $D = J$이다.

(증명) 임의의 두 원소 $a, b \in M$에 대해 aJb라고 하자. J 관계 정의에 의해 $xay = b$이고 $ubv = a$인 x, y, u, v가 M에 존재한다. $ubv = a$에서 b를 xay로 반복 치환하면

$$a = (ux)a(yv) = (ux)^2 a(yv)^2 = (ux)^3 a(yv)^3 = \cdots$$

을 얻으며 같은 방법으로

$$b = (xu)a(vy) = (xu)^2 a(vy)^2 = (xu)^3 a(vy)^3 = \cdots$$

를 얻는다. M이 주기적이므로 $(ux)^m$이 멱등원인 m이 존재한다([4]의 정리 1, 2, 3). $c = xa$라고 하면,

$$a = (ux)^m a(yv)^m = (ux)^m (ux)^m a(yv)^m = (ux)^m a = (ux)^{m-1} uc$$

이므로 aLc이다. $cy = xay = b$이므로 $(vy)^n$이 멱등원인 n에 대해

$$c = xa = x(ux)^{n+1} a(yv)^{n+1} = (xu)^{n+1} xay(vy)^n v = (xu)^{n+1} b(vy)^{2n} v = (xu)^{n+1} b(vy)^{n+1} (vy)^{n-1} v = b(vy)^{n-1} v$$

이고, cRb이다. 따라서 aLc이고 cRb인 c가 항상 존재하므로 $D = J$ 이다. ■

[정리 1]은 유한 모노이드에서 J 관계와 D 관계가 같음을 보인다[4]. 따라서 $M_n^n(F)$는 유한 모노이드이므로 $M_n^n(F)$에서 $J = D$이다.

[정리 2] $a, b \in M$일 때 R_a와 L_b를 다음과 같이 정의하자.

$$R_a = \{ c \in M \mid aM = cM \}$$
$$L_b = \{ c \in M \mid Mb = Mc \}$$

M이 유한 모노이드라면 임의의 두 원소 $a, b \in M$에 대해서

$$aJb \text{ if and only if } R_a \cap L_b \neq \varnothing$$

이다.

(증명) M이 유한 모노이드라면 임의의 두 원소 $a, b \in M$에 대해서 $aJb = aDb$이다. aDb라면 정의로부터 aRc이고 cLb인 c가 존재하므로 $R_a \cap L_b \neq \varnothing$이다. ■

[정리 2]는 매우 단순하고 직관적인 정리이나 $M_n^n(F)$에서의 효율적인 J 관계 계산에 매우 중요한 의미를 가지고 있다. [정리 2]의 R_a와 L_b는 각각 같은 우 주 아이디얼과 좌 주 아이디얼을 생성하는 생성자 집합을 의미한다. 따라서 [정리 2]는 $M_n^n(F)$의 모든 좌 주 아이디얼과 우 주 아이디얼의 생성자 집합을 얻을 수 있다면, 공통원소를 가지고 있는 모든 좌 주 아이디얼, 우 주 아이디얼의 생성자 집합의 합집합을 계산함으로써 효율적으로 J 관계를 얻을 수 있음을 보인다. 그러나 $M_n^n(F)$의 모든 좌/우 주 아이디얼에 대한 생성자 집합 계산은 $M_n^n(F)$의 모든 원소에 대해 좌/우 주 아이디얼의 계산을 요구하여 $O(|M_n^n(F)|) = O(2^{n^2})$의 불리언 행렬 곱셈을 수행해야 하므로 근본적으로 기하급수적인 계산복잡도를 가지고 있다.

이 생성자 집합 계산 문제는 [2]에서 제시한 다음 두 정리를 적용하여 $O(2^{n^2})$의 불리언 행렬 사이의 곱셈 대신 $O(2^n)$의 불리언행렬과 벡터 사이의 곱셈을 수행함으로써 같은 결과를 보다 효율적으로 얻을 수 있다.

[정리 3] $M_n^m(F)$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.

(증명) H를 $(C_A)^k$에 속한 임의의 $n \times k$ 불리언 행렬이라 하고 $H \notin R_A$라고 가정하자. $H \notin R_A$이므로 R_A에 속한 모든 $n \times k$ 불리언 행렬 G에 대하여 $H \neq G$이다. 따라서 H의 어떤 i열이 R_A에 속한 모든 $n \times k$ 불리언 행렬 G의 i열과 달라야 한다. $M_n^m(F)$는 모든 $n \times m$ 불리언 행렬의 집합이므로 V가 m 차원 불리언 벡터의 전체 집합일 때

$$M_m^k(F) = \{(v_0^T v_1^T \cdots v_{k-1}^T) \mid v_i \in V,\ 0 \le i \le k-1\}$$

이고

$$R_A = \{((A_0 u_0^T A_1 u_0^T \cdots A_{n-1} u_0^T)^T \\ (A_0 u_1^T A_1 u_1^T \cdots A_{n-1} u_1^T)^T \\ \cdots \\ (A_0 u_{k-1}^T A_1 u_{k-1}^T \cdots A_{n-1} u_{k-1}^T)^T) \\ \mid u_i \in V,\ 0 \le i \le k-1\}$$

이 된다. H는 $(C_A)^k$에 속한 $n \times k$ 불리언 행렬이므로 H의 각 열 H^i는 V에 속한 어떤 m차원 벡터 u에 대하여 $H^i = (A_0 u^T A_1 u^T \cdots A_{n-1} u^T)$이 되므로 H가 R_A에 속하게 되어 모순이다.

D를 R_A에 속한 임의의 $n \times k$ 불리언 행렬이라 하자. C_A는 $n \times m$ 불리언 행렬 A의 각 행에 V^T에 속한 각 m 차원 불리언 벡터 v^T를 곱하여 얻은 n 차원 벡터의 전체 집합이다. 따라서 0과 $k-1$ 사이의 모든 i에 대해 D가 C_A에 속하며, $D \in (C_A)^k$이다. ■

[정리 3]은 주어진 $n \times m$ 불리언 행렬에 모든 $m \times k$ 불리언 행렬을 곱하여 얻는 불리언 행렬의 집합이 $n \times m$ 불리언 행렬에 모든 m 차원 열벡터를 곱하여 얻는 열벡터들을 모든 가능한 k 번의 조합으로 얻는 불리언 행렬 집합과 같다는 것을 보인다. [정리 4]는 [정리 3]과 유사하게 증명되며

모든 $l\times n$ 불리언 행렬을 하나의 $n\times m$ 불리언 행렬에 곱하여 얻는 불리언 행렬의 집합이 모든 n 차원 행벡터를 $n\times m$ 불리언 행렬에 곱하여 얻는 행벡터들을 모든 가능한 k 번의 조합으로 얻는 불리언 행렬 집합과 같다는 것을 보인다.

[정리 4] $M_n^m(F)$에 속한 임의의 $n\times m$ 불리언 행렬 A에 대해 L_A, T_A를

$$L_A=\{BA \mid B\in M_l^n(F)\},$$
$$T_A=\{(vA^0vA^1\cdots vA^{m-1}) \mid v\in V(n)\}$$

로 정의할 때 $L_A=(T_A)_l$이다.

[그림 1]은 [정리 1~4]를 적용하여 설계한 $M_n^n(F)$의 모든 J 관계 계산 알고리즘의 개요를 보이고 있다. 알고리즘의 개요는 다음과 같다. 먼저 [정리 3]을 적용하여 $M_n^n(F)$의 모든 우 주 아이디얼의 생성자 집합을 계산하고, 다음 [정리 4]를 적용하여 모든 좌 주 아이디얼의 생성자 집합을 계산한다. 이제 $M_n^n(F)$의 모든 좌/우 주 아이디얼의 생성자 집합을 얻었으므로 [정리 2]를 적용하여 공통원소를 가지고 있는 모든 우 주 아이디얼과 좌 주 아이디얼의 생성자 집합의 합집합을 얻어 모든 J 관계를 계산한다.

5. 계산 복잡도 및 실행 결과

모든 불리언 행렬에 대한 곱셈은 기하급수적인 시간 복잡도를 가진다. 본 장의 공간 및 시간 복잡도에 대한 분석은 곱셈 알고리즘의 개선정도를 알

〈표 1〉 공간 및 시간 복잡도 비교

구 분	행렬곱셈 알고리즘	정리적용 알고리즘
시간 복잡도	2^{mk}	2^m
공간 복잡도	$nk2^n$	$n2^n$

〈표 2〉 실행시간 비교

행렬 크기	행렬곱셈 적용 알고리즘	벡터곱셈 적용 알고리즘	벡터곱셈/동치관계 적용 알고리즘
2×2	0.006초	0.02초	0.007초
3×3	9.682초	0.049초	0.048초
4×4	29312169초	3.013초	2.82초
5×5	5.126×10^{15}초 이상	168,877초	122,909초

```
RiList = ∅
RiGenList = ∅
for each boolean matrix A in M_n^n(F)
  compute a right principal ideal RI_A of A
  if (RI_A ∈ RiList)
    insert A into GEN(RI_A)
  else
    make a new generator set GEN(RI_A) for RI_A
    insert GEN(RI_A) into RiGenList
    insert A into GEN(RI_A)
LiList = ∅
LiGenList = ∅
for each boolean matrix A in M_n^n(F)
  compute a left principal ideal LI_A of A
  if (LI_A ∈ LiList)
    insert A into GEN(LI_A)
  else
    make a new generator set GEN(LI_A) for LI_A
    insert GEN(LI_A) into LiGenList
    insert A into GEN(LI_A)
JRel = ∅
for each GEN(RI_A) ∈ RiGenList
  J_A = GEN(RI_A)
  for each GEN(LI_A) ∈ LiGenList
    if GEN(RI_A) ∩ GEN(LI_A) ≠ ∅
      J_A = J_A ∪ GEN(LI_A)
  if J_A ∉ JRel
    insert J_A into JRel

* RI_A : a right principal ideal of A
* LI_A : a left principal ideal of A
* RiList : a list of right principal ideals
* LiList : a list of left principal ideals
* GEN(PIDL) : a set of generators of PIDL
* RiGenList : a list of generator sets for right
              principal ideals
* LiGenList : a list of generator sets for left
              principal ideals
* JRel : a set of J relations
```

〈그림 1〉 J 관계 계산 알고리즘

기 위한 노력이며, <표 1>은 불리언 행렬을 직접 곱한 경우와 [정리 3~4]를 적용한 경우의 공간 및 시간 복잡도를 보이고 있다. $n \times m$ 불리언 행렬에 모든 $m \times k$ 불리언 행렬을 곱하는 경우 2^{mk}개의 $m \times k$ 불리언 행렬을 곱하는 대신 2^m개의 m 차원 벡터를 곱하여 결과를 얻을 수 있어 실제 실행 시간에 상당한 성능 개선을 가져올 수 있다. 또한 곱셈 결과로 n 차원 벡터의 집합을 생성하여 이 집합에 속한 벡터들을 모든 가능한 방법으로 k번 조합하면 결과로 얻을 모든 $n \times k$ 불리언 행렬을 얻을 수 있으므로 최악의 경우에도 $n2^n$에 비례한 메모리 공간이 요구되므로 불리언 행렬의 집합을 직접 나타낼 때 최악의 경우 요구되는 메모리 공간이 $nk2^{nk}$에 비례하는 것과 비교하면 중간결과 저장을 위한 메모리 공간을 고려하지 않더라고 많은 메모리 공간이 절약되는 것을 알 수 있다[2].

J 관계 계산 알고리즘은 Java 언어로 구현하였으며 2개의 2.8 GHz Zeon CPU, 1 GB RAM, 250 GB HDD, Fedora 9.0 환경에서 실행하였다. 불리언 행렬과 벡터의 곱셈은 행 OR-연산 기반 불리언 행렬 곱셈[4]을 비트사이의 논리연산에 적합하도록 변형하여 수행하였다. <표 2>는 불리언 행렬을 직접 곱한 곱셈, 벡터를 이용한 행렬 곱셈([정리 3-4] 적용), 벡터 이용 곱셈과 J/D 동치관계를 모두 이용([정리 1~4] 적용)한 J 관계 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 기하급수적이나 <표 1>에서 보는 것처럼 정리를 모두 적용한 J 관계 계산 알고리즘의 실제 실행시간은 다른 알고리즘과 비교하여 상당한 개선을 보이고 있다.

6. 결 론

불리언 행렬과 관련된 대부분의 연구와 응용은 두 불리언 행렬의 최적화된 곱셈에 중점을 두고 있으나, J 관계 계산은 모든 불리언 행렬 사이의 이중 연속곱셈을 요구한다. 이러한 곱셈은 기하급수적인 계산복잡도를 가지므로 공간 및 시간 복잡도 개선에 근본적인 어려움이 내포되어 있다. 본 논문은 $n \times n$ 불리언 행렬의 반군을 대상으로 한 모든 J 관계의 효율적인 계산을 위해 기존의 연구와 문제점에 대하여 논하고, 벡터 기반의 불리언 행렬 곱셈 이론과 반군에서의 Green 관계에 대한 정리를 이용하여 보다 개선된 모든 J 관계 계산 알고리즘을 제시하였다.

본 연구에서 제시한 J 관계 계산 이론은 분산 및 병렬 컴퓨팅 등의 실행환경에 상관없이 성능개선 등을 위해 적용가능 하다. 또한 원소가 세 개 이상의 값을 갖는 일반 행렬에도 이론을 확대 적용할 수 있다. 그러나 아직 여러 부분에서 알고리즘의 최적화가 필요하며, 정규 D-클래스(regular D class)를 신속히 계산할 수 있는 알고리즘, 분산 또는 병렬 알고리즘의 설계 및 구현 등에 대한 연구가 필요하다.

참 고 문 헌

[1] 김창범, 한재일, "Classifications of D-classes in the semigroup $M_n(F)$ of all n x n Boolean matrices over F = {0, 1}", 「퍼지 및 지능시스템학회 논문지」, 제16권, 제3호(2006), pp.338-348.

[2] 한재일, "모든 l x n, n x m, m x k 불리언 행렬 사이의 중첩곱셈에 대한 연구", 「한국IT서비스학회지」, 제5권, 제1호(2006), pp.191-198.

[3] 한재일, "효율적인 D-클래스 계산을 위한 알고리즘", 「한국IT서비스학회지」, 제6권, 제1호(2007), pp.151-158.

[4] Howie, J. M., *An Introduction to semigroup theory*, Oxford University Press, 1995.

[5] Linton, S. A., G. Pfeiffer, E. F. Robertson,

and N. Ruskuc, "Computing Transformation Semigroups", *Journal of Symbolic Computation*, Vol.33(2002), pp.145-162.

[6] Wall, J. R., "Green's relations for stochastic matrices", *Czechoslovak Mathematical Journal*, Vol.25, No.2(1975), pp.247-260.

[7] Heyworth, A., "One-Sided Noncommutative Grobner Bases with Applications to Computing Green's Relations", *Journal of Algebra*, Vol.242(2001), pp.401-416.

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

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

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

[11] 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*, 1995, pp. 653-657.

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

[13] Rim, D. S. and Kim, J. B., "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.

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

[15] Yelowitz, L., "A Note on the Transitive Closure of a Boolean Matrix", *ACM SIGMOD Record*, Vol.25, No.2(1978), p.30.

[16] Comstock, D. R., "A note on multiplying Boolean matrices II", *CACM*, Vol.7, No.1 (1964), p.13.

[17] 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(1995), pp.799-801.

[18] 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.

[19] Booth, K. S., "Boolean matrix multiplication using only bit operations", *ACM SIGACT News*, Vol.9, No.3(1977), p.23.

[20] Cousineau, G., J. F. Perrot, and J. M. Rifflet, "APL programs for direct computation of a finite semigroup", *APL Congres*, Vol.73(1973), pp.67-74.

[21] Lallement, G., *Semigroups and Combinatorial Applications*, John Wiley and Sons, New York, 1979.

[22] Champarn명, J. M. and G. Hansel, "AUTOMATE, a computing package for automata and finite semigroups", *Journal of Symbolic Computation*, Vol.12(1991), pp.197-220.

[23] Sutner, K., "Finite State Machines and Syntatic Semigroups", *The Mathematica Journal*, Vol.2(1991), pp.78-87.

[24] Lallement, G. and R. McFadden, "On the determination of Green's relations in finite transformation semigroups", *Journal of Symbolic Computation*, Vol.10(1990), pp.481 -498.

[25] Konieczny, J., "Green's equivalences in finite semigroups of binary relations", *Semigroup Form*, Vol.48(1994), pp.235-252.

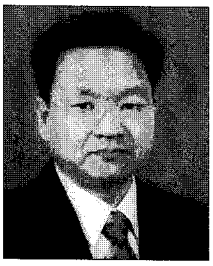
[26] Plemmons, R. J. and M. T. West, "On the semigroup of binary relations", *Pacific Journal of Mathematics*, Vol.35(1970), pp.743-753.

[27] Simon, I., "On semigroups of matrices over the tropical semiring", *Informatique Theorique et Applications*, Vol.28(1994), pp.277-294.

[28] Straubing, H., "The Burnside problem for semigroups of matrices", *in Combinatorics on Words, Progress and Perspectives, L. J. Cummings(ed.)*, Academic Press, (1983), pp.279-295.

[29] Froidure, V. and J. Pin, "Algorithms for computing finite semigroups", Foundations of Computational Mathematics, 1997, pp.112-126.

◈ 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 서비스 사이언스, 서비스 컴퓨팅, 분산 시스템, 객체지향 시스템, 미들웨어, 컴퓨터 및 네트워크 보안, 지능형 시스템, 공개 소프트웨어 등이다.