불리언 행렬의 곱셈에 관한 연구

A Study on the Multiplication of Boolean Matrices

한재일, 전성택*
국민대학교, 영산대학교*

Han Jae-Il, Jun Sung-Taeg*
Kookmin Univ., Youngsan Univ.*

요약

불리언 행렬은 다양한 분야에 응용되어 유용하게 사용되고 있으며, 불리언 행렬의 응용과 곱셈에 대하여 많은 연구가 수행되었다. 대부분의 연구에서는 불리언 행렬의 곱셈을 다루고 있으나 모두 두 불리언 행렬 곱셈에 관심을 두고 있으며 많은 불리언 행렬 쌍의 곱셈은 극히 소수의 연구에서 보이고 있다. 본 논문에서는 기존에 제시된 두 불리언 행렬의 최적 곱셈 알고리즘이 많은 불리언 행렬 쌍에 대한 곱셈을 해야 하는 경우 부적합함을 보이고 하나의 $n \times m$ 불리언 행렬과 모든 $m \times k$ 불리언 행렬의 곱셈을 개선시킬 수 있는 방법을 제시한다.

Abstract

Boolean matrices are applied to a variety of areas and used successfully in many applications. There are many researches on the application and multiplication of boolean matrices. Most researches deal with the multiplication of boolean matrices, but all of them focus on the multiplication of just two boolean matrices and very few researches deal with the multiplication of many pairs of two boolean matrices. The paper discusses it is not suitable to use for the multiplication of many pairs of two boolean matrices the algorithm for the multiplication of two boolean matrices that is considered optimal up to now, and suggests a method that can improve the multiplication of a $n \times m$ boolean matrix and all $m \times k$ boolean matrices.

I. 서 론

불리언 행렬은 원소가 0(거짓)이나 1(참) 값을 갖는 행렬로 정의되며 단순하고 논리적인 특성으로 인해 여러 분야에 응용되어 유용하게 사용되고 있다. 다양한 분야에서의 응용을 위해 불리언 행렬에 대하여 많은 연구가 수행되었으며[4-9] 대부분의 연구에서 불리언 행렬의 곱셈을 다루고 있다. 그러나 이 연구들은 모두 두개의 불리언 행렬 곱셈에 관심을 두고 있으며 극히 소수의 연구[1-3]만이 많은 불리언 행렬 쌍의 곱셈을 다루고 있다.

본 논문은 기존에 제시된 두 불리언 행렬의 최적(optimal) 곱셈 알고리즘이 많은 불리언 행렬 쌍에 대한 곱셈을 해야 하는 경우 부적합함을 보이고 하나의 $n \times m$ 불리언 행렬과 모든 $m \times k$ 불리언 행렬의 곱셈을 보다 효율적으로 할 수 있도록 수학적 이론을 바탕으로 곱셈 시간을 개선시킬 수 있는 방법을 제시한다. 본 논문의 구성은 다음과 같다. 2장은 기존에 제시된 두 불리언 행렬의 최적 곱셈 알고리즘을 살펴보고 하나의 $n \times m$ 불리언 행렬과 모든 $m \times k$ 불리언 행렬의 곱셈에 적용할 경우 나타나는 문제점에 대하여 기술한다. 3장은 하나의 $n \times m$ 불리언 행렬과 모든 $m \times k$ 불리언 행렬의 곱셈을 보다 효율적으로

할 수 있는 수학적 이론과 방법에 대하여 논하고, 4장은 결론 및 향후 연구방향에 대하여 기술한다.

II. 관련 연구 및 문제점

두 불리언 행렬의 최적 곱셈에 대한 연구가 많이 수행되었으나 행을 이용한 곱셈 알고리즘 중에서 $O(n^2/logn)$ 번의 행 OR-연산을 보이는 [8]이, 비트 기반 연산을 이용한 알고리즘 중에서는 $O(n^{\log_2 7}\log n)$ 번의 비트 연산을 요구하는 [9]가 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 행 OR-연산이나 비트 연산에 필요한 정보를 미리 계산할 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. 또한 모든 $n \times n$ 불리언 행렬과 모든 $n \times n$ 불리언 행렬의 곱셈을 하는 경우 위의 두 알고리즘은 두 불리언 행렬 곱셈이 수행되기 전에 요구되는 정보를 생성하기 위하여 각 곱셈마다 [8]은 최악의 경우 $O(2^n)$, [9]는 $O(n^2\log n)$의 계산 시간이 추가적으로 필요하다.

위에 언급한 바와 같이 [8, 9]의 알고리즘은 불리언 행렬에 여러 불리언 행렬을 곱할 때 바로 사용되기 어렵다. 그러나 [8]의 경우 알고리즘의 핵심 아이디어가 불리언 행렬을 여러 불리언 행렬에 곱할 때 부분적으로 사용될 수 있는 여지가 있으므로 간단히 [8]의 핵심내용을 살펴본다. $n \times n$ 불리언 행렬 A, B가 주어졌을 때 두 행렬의 곱 $C = AB$는

$$C_{i,j} = \bigcup_{k=1}^{n} (A_{ik} \cap B_{kj}) \quad \text{for } 1 \le i, j \le n$$

으로 정의된다. $M_{k\cdot}$를 M의 k 행에서 1 값을 가지는 열 번호의 집합 즉

$$M_{k\cdot} = \{\, j \mid A_{kj} = 1 \,\}$$

으로 정의하였을 때 [8]은 다음 결과를 보인다.

$$C_{k\cdot} = \bigcup_{j \in A_{k\cdot}} B_{j\cdot}$$

즉, [8]은 두 불리언 행렬 곱셈에서 앞의 불리언 행렬A를 불리언 행렬 B의 행에 대한 OR-연산 지시자로 사용하여 곱셈 결과를 얻는다.

III. 일 대 다수의 불리언 행렬 곱셈

본 논문은 하나의 $n \times m$ 불리언 행렬과 모든 $m \times k$ 불리언 행렬의 곱셈을 보다 효율적으로 하기 위해서 불리언 행렬 연산의 수학적 특성을 이용하여 곱셈 시간을 개선할 수 있고 또한 곱셈 결과 생성되는 모든 불리언 행렬의 집합을 쉽게 나타낼 수 있는 수학적 이론을 찾는데 중점을 두었다. 본 장에서는 하나의 $n \times m$ 불리언 행렬과 모든 $m \times k$ 불리언 행렬의 곱셈을 $n \times m$ 불리언 행렬과 m 차원 벡터의 곱셈으로 할 수 있는 정리에 대하여 논한다. 본 장의 설명을 위해 다음과 같이 정의한다. 임의의 $n \times m$ 불리언 행렬 A가 주어지고 $F = \{0, 1\}$ 이라 할 때

A_i : A의 i 행

A^i : A의 i 열

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

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

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

$V^T = \{ v^T | v \in V \}$

$V^n = \{ [v_0^T v_1^T \cdots v_{m-1}^T] | v_i \in V \text{for } 0 \le i \le m-1 \}$

$Av = \{ (A_0 v^T A_1 v^T \cdots A_{m-1} v^T)^T | v \in V \}$

$M_n^m(F) = \{ A | A \text{ is an } n \times m \text{ boolean matrix} \}$

위 정의를 사용하여 다음과 같은 정리를 얻을 수 있다.

[정리 1] $M_n^m(F)$의 임의의 $n \times m$ 불리언 행렬 A에 대해 V를 m 차원 불리언 벡터의 전체 집합, $R_A = \{ AB | B \in M_m^k(F) \}$, $C_A =$ $\{ (A_0 v^T A_1 v^T \cdots A_{n-1} v^T)^T | v \in V \}$로 정의할 때 $C_A^k \subset R_A$ 이다.

(증명) H를 C_A^k에 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 임의의 i열(H^i)은 R_A에 속한 모든 $n \times k$ 불리언 행렬 G의 i 열(G^i)\과 다르다. $M_n^m(F)$는 모든 $n \times m$ 불리언 행렬의 집합이므로 V가 m 차원 불리언 벡터의 전체 집합일 때

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

이고 $B \in M_m^k(F)$이므로

$$R_A = \{ [(A_0 u_0^T A_1 u_0^T \cdots A_{n-1} u_0^T)^T \\ (A_0 u_1^T A_1 u_1^T \cdots A_{n-1} u_1^T)^T \\ \cdots \\ (A_0 u_{k-1}^T A_1 u_{k-1}^T \cdots A_{n-1} u_{k-1}^T)] \\ | u_i \in V \text{ for } 0 \le i \le k-1 \}$$

이 된다. H는 C_A^k에 속한 $n \times k$ 불리언 행렬이므로 V에 속한 어떤 m 차원 불리언 벡터 u_j $(0 \le j \le k-1)$에 대하여

$H^i = (A_0 u_j^T A_1 u_j^T \cdots A_{n-1} u_j^T)$가 되므로 모순이다.

[정리 2] V를 m 차원 불리언 벡터의 전체 집합이라 하자. $M_n^m(F)$의 임의의 불리언 행렬 A에 대해 $R_A = \{ AB | B \in M_m^k(F) \}$, $C_A =$ $\{ (A_0 v^T A_1 v^T \cdots A_{n-1} v^T)^T | v \in V \}$로 정의하면 $R_A \subset C_A^k$ 이다.

(증명) D를 R_A에 속한 임의의 $n \times k$ 불리언 행렬이라 하자. C_A는 $n \times m$ 불리언 행렬 A의 각 행에 V에 속한 각 m 차원 불리언 벡터 v의 v^T를 곱하여 얻은 n 차원 벡터($n \times 1$ 행렬)의 전체 집합이다. 따라서 0과 $k-1$ 사이의 모든 i 에 대해 D^i가 C_A에 속하며, $D \in C_A^k$ 이다.

위의 두 정리로부터 다음 부속정리를 얻을 수 있다.

[정리 3] $M_n^m(F)$에 속한 임의의 $n \times m$ 불리언 행렬 A에 대해 R_A와 C_A^k를 다음과 같이 정의할 때 $R_A = C_A^k$이다.

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

$R_A = \{ AB | B \in M_m^k(F) \}$,

$C_A = \{ (A_0 v^T A_1 v^T \cdots A_{n-1} v^T)^T | v \in V \}$

로 정의할 때 $C_A^k \subset R_A$이다.

위의 정리에 의해 하나의 $n \times m$ 불리언 행렬과

모든 $m \times k$ 불리언 행렬의 곱셈을 $n \times m$ 불리언 행렬과 모든 m 차원 벡터의 곱셈으로 할 수 있다. $n \times n$ 불리언 행렬의 경우 하나의 $n \times n$ 불리언 행렬에 2^{n^2} 개의 $n \times n$ 불리언 행렬을 곱하는 대신 2^n 개의 n 차원 벡터를 곱하여 결과를 얻을 수 있어 실제 실행 시간에 상당한 성능 개선을 가져올 것으로 기대된다. 또한 곱셈 결과로 n 차원 벡터의 집합을 생성하여 이 집합에 속한 벡터들을 모든 가능한 방법으로 k 번 조합하면 결과로 얻어야 할 모든 $n \times k$ 불리언 행렬을 간접적으로 얻을 수 있어 최악의 경우에도 2^n에 비례하여 메모리 공간이 요구되므로 불리언 행렬의 집합을 직접 나타낼 때 최악의 경우 요구되는 메모리 공간이 2^{n^2}에 비례하는 것과 비교하면 대폭 개선된 것을 알 수 있다.

IV. 결론 및 향후 연구방향

다양한 분야에서의 응용을 위해 불리언 행렬에 대하여 많은 연구가 수행되었으며 대부분의 연구에서 불리언 행렬의 곱셈을 다루고 있다. 그러나 이 연구들은 모두 두개의 불리언 행렬 곱셈에 관심을 두고 있으며 많은 불리언 행렬 쌍의 곱셈에 대한 연구는 극히 소수의 연구가 보이고 있다.

본 논문은 기존에 제시된 두 불리언 행렬의 최적 곱셈 알고리즘이 많은 불리언 행렬 쌍에 대한 곱셈을 해야 하는 경우 실행시간과 메모리 공간의 문제점으로 인해 부적합함을 보이고 하나의 $n \times m$ 불리언 행렬과 모든 $m \times k$ 불리언 행렬의 곱셈을 보다 효율적으로 할 수 있는 수학적 이론을 제시하였다. 또한 이를 바탕으로 벡터 집합을 이용하여 실행시간과 메모리 공간에 대한 요구를 개선시킬 수 있는 방법을 제시하고 기존의 방법과 비교하여 실행시간과 메모리 공간 요구를 어느 정도 개선할 수 있는지 분석하였다.

모든 불리언 행렬 쌍에 대한 곱셈은 NP-완전 계산 복잡도를 가지므로 해결 방안을 찾기 어려우나, 어느 정도 많은 불리언 행렬 쌍에 대한 곱셈은 각 경우에 적용할 수 있는 수학적 이론을 정립할 수 있다면 적은 메모리 공간으로 빠른 실행이 가능할 수도 있음을 본 논문의 결과는 보이고 있다. 이러한 관점에서 본 논문의 결과는 단지 시작일 뿐이며 앞으로 많은 불리언 행렬 쌍에 대한 곱셈을 효율적으로 할 수 있는 알고리즘을 개발하기 위해서는 이론 뿐아니라 최적화 방법, 병렬 컴퓨팅의 적용 등에 대한 많은 연구가 필요하다.

▌참 고 문 헌▐

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

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

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

[7] Macii, E., "A Discussion of Explicit Methods for Transitive Closure Computation Based on Matrix Multiplication," 29th Asilom Conference on Signals, Systems and Computers, Vol2, pp.799-801, 1995.

[8] 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.

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