通信情報保護學會誌
第2卷 第4號, 1992.12

효율적인 다중 일치 프로토콜
An Efficient Multiparty Consensus Protocol

김수진*, 류재철*

요 약

본 논문에서는 시스템 내의 모든 site들에게 분산되어 있는 정보들을 수렴하여 일치를 이루고, 그 결과를 모든 site들이 알도록 하는 다중 일치 프로토콜을 위한 효과적인 통신 방법을 제안하고자 한다. 분산 시스템에 참여하는 computer 또는 site들의 수를 N이라 할때, $O(N^2)$의 message를 필요로 하면서 한 round안에 일치를 이룰 수 있는 프로토콜은 message의 수가 너무 많다는 것이 단점이다. 이에 본 논문에서는 Finite Projective Planes을 이용하여 message의 수를 줄이면서 두 round 안에 일치를 이룰 수 있는 통신 방법을 제안한다. 이때, 각 round마다 필요한 message의 수는 $O(N\sqrt{N})$이다. 또한, 이 통신 방법에서 이용되는 Finite Projective Planes을 구축하는 알고리즘을 제안하고자 한다.

1. 개 요

본 논문에서 고려하는 분산시스템은 point-to-point 통신 네트워크에 의해 연결된 독립된 computer (Autonomous computer) 또는 site들로 이루어져 있으며, 미리 정해진 중앙 조절자(central controller)를 포함하고 있지 않다. 또한 시스템 내의 모든 site들은 동등하며, 서로 message를 통해 통신한다. 이러한 시스템하에서 여러 site에 분산되어 있는 정보를 매개 변수로 갖는 associative함수나 명제(predicate)를 계산하여 그 결과를 모든 site들이 알도록 하는 것이 본 논문의 목적이다.

예를 들어, 각 site는 'Yes'(1) 또는 'No'(0)에 대한 값을 하나씩 가지고 있고, 각 site들은 이러한 값들을 수렴하여 전체 의견을 알고 싶다고 하자. 이러한 계산을 하기 위해서는 여러 site에 분산되어 있는 정보들을 수집해야만 한다. 고려하는 시스템 내에 중앙 조절자가 없으므로 이 계산은 여러 site들의 협조와 조정 아래 이루어져야만 한다. 이와 같이 여러 site들의 참여 아래 일치를 이루는 다중일치 문제를 해결하기 위해 많은 연구가 진행되었으며, 이에 대한 연구결과로는 최대, 최소값 찾기,[6,12,3] 분산 checkpoint의 조정,[17] 트랜잭션 원시성(transaction atomicity)의 유지[15,16]등이 있다.

위와 같은 계산을 위한 가능한 방법으로 각 site가 자신의 정보('Yes' 또는 'No')를 시스템내의 다른 모든 site에게 전달함으로서 전체의견을 수렴하는 방법이 있을 수 있다. 이 방법은 한 round[1]의

* 충남대 전산학과

[1] 각 site가 message를 전달해 주고 다른 site로부터 필요한 모든 message를 받을 때 까지를 한 round이라한다.

message 교환을 통해 일치를 이룰 수는 있지만, 시스템내의 site의 갯수를 N이라 할때, N×(N-1)개의 message가 필요하다는 것이 단점이다. 이에 본 논문에서는 message의 갯수를 줄이면서 두 round안에 일치를 이룰 수 있는 효과적인 통신 방법을 제안하고자 한다. 이 통신 방법은 Finite Projective Planes 을 기초로 하며 각 round마다 $O(N\sqrt{N})$의 message 만을 사용하여 일치를 이룰 수 있다. 이러한 방법은 암호학에서도 효율적인 알고리즘 개발에 널리 이용되리라 기대된다.

다음장에서는, Finite Projective Planes을 개략적으로 소개하고 필요한 정리들을 서술한다. 3장에서는 Finite Projective Planes을 이용한 통신 방법이 제안되며, 4장에서는 이러한 통신 방법을 응용한 최대, 최소값을 찾는 프로토콜을 다룬다. 5장에서는 시스템에 분산되어 있는 숫자들의 합을 계산하는 프로토콜을 제안한다. 6장에서는, 두 round안에 일치를 이루기 위해 필요한 message갯수의 lower bound는 $O(N\sqrt{N})$임을 보여 주고, 7장에서는 Finite Projective Planes을 구축하는 알고리즘을 제안한다. 8장에서 앞으로의 연구방향과 결론을 정리한다.

2. Finite Projective Planes

모든 일치 프로토콜에 있어서, 각 site는 자신과 통신하는 다른 site들의 집합과 연관되어 있다. 앞장에서 보았듯이 이 집합은 자신을 제외한 다른 모든 site들로 이루어질 수도 있고, ring구조로 연결된 시스템에서는 자신의 이웃 site가 이 집합을 구성할 수도 있다. 이러한 집합의 크기(cardinality), 구성원소(membership), 교차(intersection) 등의 특성은 일치 프로토콜의 message복잡도(complexity), 대칭성(symmetry), message교환의 round횟수에 영향을 미친다.

본 논문에서 제안하는 통신 방법은 message의 갯수를 줄이기 위해 각 site는 시스템내의 일부 site들과만 통신 할 것을 요구한다. 시스템내의 일부 site들과만 통신하면서도 필요한 모든 message를 받은 것과 같은 효과를 얻기 위해서는 각 site들과 연관된 집합을 잘 선택하여야만 한다. 또한 각 site들은 일치를 이루기 위해 서로 협조하고 조정하는데 있어서 같은 양의 역할을 담당해야 한다. 즉, 각 site들 간에 대칭(symmetry)을 유지해야만 한다. 이러한 대칭을 이루기 위해 다음과 같은 조건을 만족해야 한다.

1) 모든 site들은 각 round마다 같은 수의 message를 보내야 한다.
2) 각 site와 통신하는 집합의 크기는 서로 같아야 한다.
3) 각 site는 같은 수의 집합에 포함되어야 한다.

본 논문에서 제안하는 통신 방법은 각 round에 기껏해야 $O(N\sqrt{N})$개의 message를 보낼 수 있으므로 각 site와 통신하는 집합의 크기는 $O(\sqrt{N})$이어야 함을 알 수 있다. 또한 각 site는 두 round안에 다른 site에서 수행되는 계산에 영향을 미칠 수 있어야 하므로, 각 site와 연관된 집합들의 교차 그래프(intersection graph)[2]는 연결 그래프(connected graph)이고 동시에 완전 그래프(complete graph)이어야 함을 알 수 있다.

이와 유사한 특성을 만족하는 집합들이 Mutual exclusion을 위해 사용되었고,[10] Mullender와 Vitanyi는 분산 match-making문제에 있어서 $O(\sqrt{N})$ message가 lower bound임을 증명했다.[11] 위의 특성을 만족하는 집합을 만드는 한가지 가능한 방법은 Finite Projective Planes을 사용하는 것이다. (Finite Projective Planes의 구축은 7장에서 논의한다.)

Finite Projective Planes은 다음의 공리를 만족하는 유한개의 점(points)과 선(lines:점들로 이루어진 집합)들로 이루어진다.

공리 1 : 서로 다른 두 점은 오직 하나의 공통 선위에 놓여 있다.

공리 2 : 서로 다른 두 선은 오직 하나의 공통 점을 지난다.

공리 3 : 서로 다른 4개의 점들이 있다. 이 점들 중 3개는 같은 선위에 있지 말아야 한다.

[2] $S=\{S_1, S_2, \cdots, S_n\}$이라 하고 S_i를 집합이라 하자. S를 절점(Vertex)으로 갖고, $S_i \cap S_j \neq \phi$이라면 두 절점 사이에 절선(edge)(S_i, S_j)가 존재하도록 하는 그래프를 S의 교차 그래프(intersection graph)라 한다.

공리 3은 한 선위에 모든 점들이 놓여있는 degenerate finite projective plane을 제외시키기 위한 조건이다. Finite Projective Plane의 각 점을 site로 간주하며, 선들을 각 site와 연관된 집합으로 간주한다. 7개의 점을 가진 Projective Plane이 그림 1에 보여 진다.

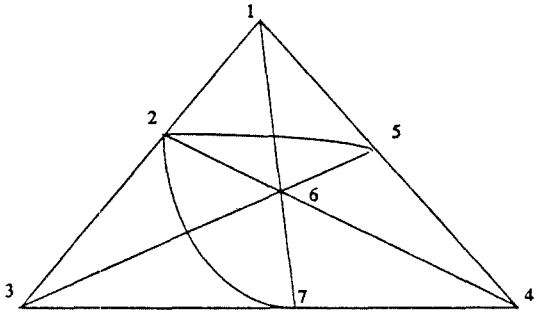


그림 1. 7개의 점을 가진 Finite Projective Plane

다음은 본 논문에서 제안한 통신 방법을 위해 필요한 Finite Projective Plane에 관한 정리들이다.

정리 2.1. Finite Projective Plane에서, 모든 점은 같은 갯수의 선위에 놓여 있고, 모든 선은 같은 갯수의 점을 지난다.

정리 2.2. Finite Projective Plane에서, 각 점을 지나는 선들의 수는 각 선분상의 점들의 갯수와 같다.

정리 2.3. 각 선분 상에 m+1개의 점이 있고, 각 점을 지나는 m+1개의 선을 가진 Projective Plane은 m^2+m+1개의 점과 m^2+m+1개의 선을 가진다.

정리 2.4. 소수 p와 양의 정수 k에 대해 $m=p^k$이라면, order m의 Projective Plane이 존재한다.

정리 2.3에서 m을 Finite Projective Plane의 order라고 한다.

3. 통신방법

이 장에서 논의되는 통신 방법은 Finite Projective Planes에 의해서 얻어진 집합에 따라 각 site들은 시스템 내의 일부 site들과만 통신하며, 각 site는 Finite Projective Planes의 한 점으로 간주된다. (site와 점은 서로 같은 의미로 번갈아 사용하였다). 일단 시스템내의 site들의 갯수 N에 대하여 Finite Projective Planes이 존재한다고 가정하며, 존재하지 않는 경우에 대해서는 이 장의 끝에서 논하기로 한다.

Projective Plane의 각 점 i는 전단사 함수 F에 의하여 그 점 i와 인접한(incident) 유일한 선 L_i와 연관된다. 이 선 L_i는 site i와 연관된 집합이 될 것이다. 함수 F의 존재 여부는 연역된 이분 그래프(induced bipartite graph)에서 saturating matching의 존재를 증명함으로서 확인된다. 증명은 부록에서 주어지며, 증명에서 볼 수 있듯이 함수 F를 결정하기 위해 이분 그래프 matching 알고리즘이 사용될 수 있다. 그림 1의 Projective Plane에 대한 각 site i와 연관된 집합 L_i는 그림 2에서 주어진다.

L_1 : {1,2,3}
L_2 : {2,4,6}
L_3 : {3,5,6}
L_4 : {4,5,1}
L_5 : {5,2,7}
L_6 : {6,7,1}
L_7 : {7,3,4}

그림 2. Projective Plane의 각 site와 연관된 집합들

이러한 선 L_i들의 특성들을 살펴보면 다음과 같다.

Finite Projective Planes의 각 선들의 특성 :

(1) 각 선, L_i, 의 크기, $|L_i|$, 는 m+1이다. $(1 \le i \le N)$
(2) 임의의 점 j, $1 \le j \le N$, 는 m+1개의 선, L_i, 들에 포함된다.
(3) 임의의 선 L_i와 L_j, $i \neq j$, 에 대하여, 두 선의 공통 점의 갯수는 하나이다. 즉, $|L_i \cap L_j| = 1$이다.
(4) 각선, L_i, 는 점 i를 포함한다.

특성 1과 2는 정리 2.1, 2.2와 2.3의 결과임을 알

수 있다. 또한, 특성 3은 공리 2의 결과이다.

한점 i를 고려해 보자. 점 i는 정리 2.3에 따라 그 점에 인접한(incident) 선 L_i와 m개의 다른 선을 가지고 있다. 그 m개의 선은 함수 F의 역함수인 F^{-1}에 의해 유일한 점과 사상(mapping)될 수 있다. 이러한 점을 각각 $i_1, i_2, \cdots, i_m$이라 하자. 본 논문에서 제안하는 다중 일치 프로토콜의 각 단계마다 다음의 통신 방법에 따라 통신한다.

통신방법 : site i는 $j \in L_i$이거나 $i \in L_j$이도록 하는 site $j(i \neq j)$들에게만 message을 보내고 받는다. 물론 $i \in L_j$이도록 하면서 $i \neq j$인 site들은 F^{-1}에 의해서 결정된 $i_1, i_2, \cdots, i_m$임을 알 수 있다. 그러므로, 각 단계마다 한 site는 2m개의 message를 보내면 된다. 점 i가 받는 message의 수는 다음 보조정리(lemma) 에 의해서 알 수 있다.

보조정리 3.1. : site i는 2m개의 site들로부터 message를 받는다. 즉 $j \in L_i$이거나 $i \in L_j$이면서 $i \neq j$인 site j들로부터 message를 받는다.

증명 : 각 점은 그 점에 인접한(incident) 모든 선과 연관된 점들에게 message를 보내야 한다. 그러므로, 선 L_i상의 모든 점들은 i에게 message를 보낸다. 그리고, 정리 2.3에 따라 그런 점들이 m개 있음을 알 수 있다. 또한, 각 점은 그 점과 연관된 선 상의 모든 다른 점들에게도 message를 보내야 한다. 점 i에 인접한 선은 L_i를 제외하고 m개가 있으며, 각 선에 연관된 유일한 한 점이 있다. 이런 m개의 점들 각각은 i에게 message를 보내야 한다. 공리 2에 의하여, i에 인접한(incident) m+1개의 선은 i를 제외한 어떠한 점도 공통으로 가질 수 없다는 것을 알수 있다. 그러므로 i가 받은 2m개의 message는 2m개의 서로 다른 점들로부터 온 것임을 알 수 있다. ■

Site들과 유도된 통신 경로들(communication paths)은 degree 2m의 regular 그래프를 형성한다. 그래프의 각 절선(edge)은 양방향 통신 경로(bidirectional communication path)를 나타낸다. 그림1과 관련된 통신 경로 그래프는 그림 3에서 보여준다.

보조정리 3.2. : 다중 일치 프로토콜의 각 단계마다, 각 site는 $O(\sqrt{N})$개의 message를 보낸다.

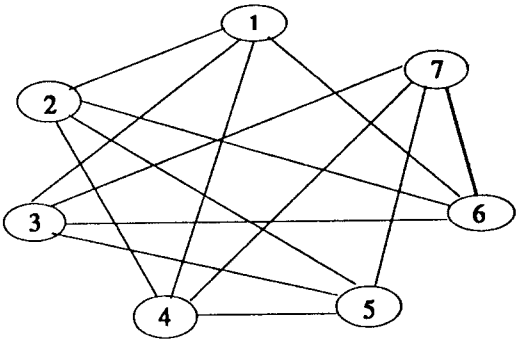


그림 3. 통신 경로 그래프

증명 : 고려되어야 할 두가지 경우가 있다. 주어진 N에 대하여,

1) Finite Projective Planes이 존재한다.

각 site는 각 단계마다 2m개의 message를 보낸다. $N=m^2+m+1$이므로, $2m=-1+\sqrt{4N-3}$이다. 그러므로, message의 갯수는 $O(\sqrt{N})$이다.

2) Finite Projective Planes이 존재하지 않는다.

$m_0=\lfloor -1+\sqrt{4N-3}/2 \rfloor$이라 하고, m_0보다 더 큰 정수 중에서 소수의 멱승(Power)인 가장 작은 정수를 m_1이라 하자. 참고문헌[3]에서 어떠한 $n \geq 1$에 대하여, $n<p \leq 2n$이도록 하는 소수 p가 적어도 하나 존재함을 보여주었다. 그러므로, $m_0<m_1 \leq 2m_0$이며, m_1이 $O(\sqrt{N})$임을 알 수 이다. 정리 2.4에 따라, $N^*=m_1^2+m_1+1$을 가진 Finite Projective Planes을 만들 수 있다. $N^*>N$이므로, N^*-N개의 가상(virtual) site가 시스템에 첨가되어야 하며, 각 단계마다 $2m_1=O(\sqrt{N})$개의 message가 한 site에 의해서 보내진다. ■

가상 site들은 계산되어질 함수나 명제(Predicate)에 따라 초기값이 다르다는 것을 빼고는, 실제 site들과 같은 알고리즘을 실행한다. 합 계산에 있어서, 가상 site들의 초기값은 0이 될 것이고, 최대, 최소값 계산에 있어서는, MININT나 MAXINT가 될 것이다. 가상 site의 참여에도 불구하고, $N^*=O(N)$이므로 각 단계마다 오직 $O(N\sqrt{N})$개의 message가 보내진다.

4. 최대, 최소값 계산을 위한 분산 프로토콜

분산 시스템내에 N개의 서로 다른 값들이 각 site 마다 하나씩 분산되어 있다고 하자. 문제는 그러한 값들 중 최대값 또는 최소값을 시스템내의 모든 site 들이 알도록 하는 것이다. 앞 장에서 소개된 통신 방법을 이용한 프로토콜을 이 장에서 설명하였다. 이 프로토콜은 참고문헌[7]에 근거한 것이다.

모든 site는 같은 프로토콜을 수행하고, site i에서 최대값을 찾기 위해 수행하는 프로토콜은 다음과 같다.

최대값 계산 프로토콜

단계 1. $j \in L_i$이거나 $i \in L_j$이면서 $i \neq j$인 site j에게 자신의 값 V_i를 보낸다.

단계 2. (1) 보조정리 3.1에서 명시된 2m개의 site 들로부터 값을 받을 때까지 기다린다.
(2) 받은 값들과 V_i 사이에 최대값 MAX_i'을 계산한다.

단계 3. $j \in L_i$이거나 $i \in L_j$이면서 $i \neq j$인 site j에게 MAX_i'을 보낸다.

단계 4. (1) 보조정리 3.1에서 명시된 2m개의 site 들로부터 값을 받을 때까지 기다린다.
(2) 받은 값들과 MAX_i'사이에 최대값 MAX_i을 계산한다.

다음 정리는 이 프로토콜의 타당성을 증명한다.

정리 4.1. 한 site가 단계 4)의 실행을 끝냈다면, 그 site는 초기에 존재했던 값들 중에 최대값을 소유한다.

증명 : 초기에 존재했던 최대값이 site j가 가지고 있는 값 V_j라 하자. site i가 단계 4)의 실행을 끝냈고 V_j가 아닌 다른 값을 최대값으로 가지고 있다고 하자. 또한, i와 연관된 선을 L_i라 할때, 고려되어야 할 두가지 경우는 다음과 같다.

1) $j \in L_i$: 이 경우에, site i가 V_j를 받지 않고 단계 2)의 (1)을 끝낼 수는 없다. 그러므로, 단계 2)의 (2)에서 계산된 MAX_i'은 V_j이어야만 한다. 단계 4)의 실행을 끝내기 이전에 site i는 MAX_i'과 받은 값들 중에서 최대값, MAX_i를 계산한다. 이 값은 분명히 V_j일 것이다. 그러므로 위의 가정이 모순임을 알 수 있다. (j=i인 경우도 여기에 준한다.)

2) $j \notin L_i$: [공리 1]에 따라 site j는 i에 인접한 (incident) m개의 선들 중, 하나에 속해야만 한다. 이 선과 연관된 점이 i_k이라 하자. 첫번째 round에서, site j는 자신의 값 V_j를 i_k에게 보내야하므로, i_k가 단계 2)에서 계산한 MAX_{i_k}'은 V_j이어야만 한다. 두번째 round에서 site i_k는 이 값을 i에게 보내야만 한다. 그러므로, site i가 단계 4)에서 계산한 최대값은 V_j이어야만 한다. 그러므로, 위의 가정은 모순이다. ■

그림 1의 Projective Plane에 대한 최대값 계산은 표 1에서 보여준다. 각 site i가 가지고 있는 값 V_i를 자신의 site번호라 가정 하였다. 또한, Finite Projective Planes가 존재하지 않는 경우에 대하여 다음 예제를 통하여 설명한다.

표 1. N=7일때, 최대값 계산

최 대 값 계 산		
Site no.	단 계 2	단 계 4
1	max{1,2,3,4,6}=6	max{6,6,7,7,7}=7
2	max{2,1,6,5,4}=6	max{6,6,7,7,7}=7
3	max{3,1,7,6,5}=7	max{7,6,7,7,7}=7
4	max{4,2,1,7,5}=7	max{7,6,6,7,7}=7
5	max{5,4,3,2,7}=7	max{7,7,7,6,7}=7
6	max{6,3,2,1,7}=7	max{7,7,6,6,7}=7
7	max{7,3,4,5,6}=7	max{7,7,7,7,7}=7

예제 4.1. N=5인 경우의 최대값 계산

보조정리 3.2의 증명에서 보았듯이 Finite Projective Planes가 존재하지 않는 경우에 가상 site들의 참여가 요구된다. N=5보다 크면서 Finite Projective Planes가 존재하는 가장 작은 N^*는 7이다. 그러므로, 2개의 가상 site들이 필요하다. 이 가상 site들의 번호를 각각 6과 7이라 하자. 이제 7개의 site들은

최대값 계산 프로토콜을 수행하면 된다. 이때, 가상 site들의 초기값은 MININT(시스템에서 표현할 수 있는 가장 작은 정수)이어야 한다. 각 site가 가지고 있는 값 V_i, $1 \le i \le 5$, 를 자신의 site 번호라 하고, $V_6=0$이고 $V_7=0$이라 할 때 최대값 계산결과는 표 2에 주어진다.

표 2. N=5일때, 최대값 계산

최 대 값 계 산		
Site no.	단 계 2	단 계 4
1	max{1,2,3,4,0}=4	max{4,5,5,5,3}=5
2	max{2,1,0,5,4}=5	max{2,4,3,5,5}=5
3	max{3,1,0,0,5}=5	max{5,4,5,3,5}=5
4	max{4,2,1,0,5}=5	max{5,5,4,5,5}=5
5	max{5,4,3,2,0}=5	max{5,5,5,5,5}=5
6(0)	max{0,3,2,1,0}=3	max{3,5,5,4,5}=5
7(0)	max{0,3,4,5,0}=5	max{5,5,5,5,3}=5

최소값 계산 프로트콜은 각 단계에서 최대값 대신 최소값을 계산한다는 것과 Finite Projective Planes가 존재하지 않는 경우에 가상 site들이 초기값으로 MAXINT(시스템에서 표현할 수 있는 가장 큰 값)를 갖는다는 것을 제외하고는 위의 프로토콜과 같다.

5. 합 계산 프로토콜

분산시스템내에 N개의 값들이 각 site마다 하나씩 분산되어 있다고 하자. 문제는 그러한 값들의 합을 구하는 것이며, 동시에 시스템 내의 모든 site들이 그 값을 알도록 하는 것이다. 이 프로토콜은 참고문헌[7]에 근거한 것이다.

한 site i에서 수행되는 프로토콜은 다음과 같다.

합 계산 프로토콜

단계 1. $i \in L_j$이고 $i \neq j$인 site j에게 자신의 값 V_i를 보낸다.

단계 2. (1) $j \in L_i$이고 $i \neq j$인 m개의 site j들로 부터 값을 받을 때까지 기다린다.

(2) 받은 값들과 V_i의 합, SUM_i'을 계산한다.

단계 3. $j \in L_i$이고 $i \neq j$인 site j에게 SUM_i'을 보낸다.

단계 4. (1) $i \in L_j$이고 $i \neq j$인 m개의 site j들로 부터 값을 받을 때까지 기다린다.

(2) 받은 값들과 SUM_i'의 합, SUM_i''을 계산한다.

(3) $SUM_i = SUM_i'' - m * V_i$를 계산한다.

다음 정리는 이 프로토콜의 타당성을 보여준다.

정리 5.1. 한 site가 단계 4)의 실행을 끝냈다면, 그 site는 초기에 존재했던 값들의 합을 소유한다.

증명 : site i가 단계 4)의 실행을 끝냈다고 하자. site i는 단계 2)에서 L_i에 포함되어 있는 모든 site들이 가지고 있는 값들과 V_i의 합 SUM_i'을 계산한다. 또한 $i_1, i_2, \cdots, i_m$들도 $L_{i_1}, L_{i_2}, \cdots, L_{i_m}$상의 모든 site들의 값의 합을 계산한다. $i_1, i_2, \cdots, i_m$에 의해서 계산된 합들은 단계 4)에서 site i가 받게 된다. 그러므로, 프로토콜이 끝나기 전에 site i는 i에 인접한(incident) m+1개의 선들(L_i와 $L_{i_1}, L_{i_2}, \cdots, L_{i_m}$)에 대응하는 값들의 합을 계산한다. [공리 2]에 따라, i는 이러한 m+1개의 선들 중 어떠한 두 선에 대해서도 공통인 점이 오직 하나뿐이다. 또한 공리 1에 의하여, Projective Plane의 모든 점은 이러한 m+1개의 선들 중 하나에 놓여 있어야만 한다. 그러므로, 계산된 합은 i를 제외한 모든 값들이 정확하게 한번 더해지게 된다. i의 값은 i가 m+1개의 선상에 있었으므로 m+1번 더해진다. 이것은 합으로부터 i의 값, V_i을 m번 빼줌으로서 해결된다. ■

그림 1의 Projective Plane에 대한 합 계산은 표 3에서 보여진다. 각 site가 가지고 있는 값은 자신의 site 번호이다. Finite Projective Planes가 존재하지 않는 경우, 앞 장의 예제에서와 마찬가지로 필요한 가상 site들의 참여하에 합 계산 프로토콜을 수행하면 된다. 각 site들이 소유한 값의 합을 계산하는 것이므로 가상 site들의 초기값으로 0을 가지고 수행하면 된다. N=5인 경우의 합계산 결과는 표 4에 주어진다.

표3. N=7일때, 합 계산

합 계 산		
Site no.	단 계 2	단 계 4
1	1+2+3 = 6	10+14+ 6−2*1 = 28
2	2+4+6 = 12	6+14+12−2*2 = 28
3	3+5+6 = 14	6+14+14−2*3 = 28
4	4+5+1 = 10	12+14+10−2*4 = 28
5	5+2+7 = 14	14+10+14−2*5 = 28
6	6+7+1 = 14	12+14+14−2*6 = 28
7	7+3+4 = 14	14+14+14−2*7 = 28

표4. N=5일때, 합 계산

합 계 산		
Site no.	단 계 2	단 계 4
1	1+2+3 = 6	10 + 1+ 6−2*1 = 15
2	2+4+0 = 6	6 + 7+ 6−2*2 = 15
3	3+5+0 = 8	6 + 7+ 8−2*3 = 15
4	4+5+1 = 10	6 + 7+10−2*4 = 15
5	5+2+0 = 7	8+10 + 7−2*5 = 15
6(0)	0+0+1 = 1	6 + 8 + 1−2*0 = 15
7(0)	0+3+4 = 7	7 + 1 + 7−2*0 = 15

6. Lower Bounds

지금까지 논의된 문제들에 있어서, site i가 가진 값 V_i는 일치를 이루기 위해 모든 다른 site에서 실행되는 계산에 영향을 미쳐야만 하며, 이것은 두 round안에 이루어져야 한다. 각 단계에서 site i가 통신하는 site들의 최대 갯수를 k라 하자. 두 round의 message교환이 끝나면 V_i는 기껏해야 $(k+1)*k$개의 site에 영향을 미칠 수 있다. 만약 두 round안에 일치가 이루어 진다면, $(k+1)*k=N$이어야 한다. 그것은 k가 $O(\sqrt{N})$임을 의미한다. 만약 각 site마다 동등한 프로토콜을 실행한다면, 모든 프로토콜들에 대한 k의 값은 같다. 그러므로, 일치를 이루기 위해 $O(N\sqrt{N})$ message가 필요하며, 본 논문의 프로토콜이 optimal임을 알 수 있다. 그리고 이 통신방법은 c가 round횟수와 연관된다고 할때, $O(N*N^{1/c})$의 분산일치 프로토콜로 확장될 수 있으리라 기대된다. 이것은 사용된 총 message의 갯수와 round 횟수 사이의 tradeoff의 결과일 것이다.

7. Finite Projective Planes의 구축

7.1. 직교 라틴 방격(Orthogonal Latin Squares)

Finite Projective Planes을 구축하기 위해 제안될 알고리즘은 라틴 방격에 바탕을 두고 있으므로, 연관된 특성들을 소개하기로 한다. 자세한 사항은 참고논문[2]를 참고하기 바란다.

정의 7.1. n개의 원소를 어느 행, 어느 열에도 한 번씩만 있도록 하여 n^2개의 entry를 가진 정방 행렬(Square Matrix)이 되도록 한 것을 order n의 라틴 방격(Latin Square)이라 한다.

라틴 방격(Latin Square)의 원소들을 간략히 1, 2, …, n이라 하자. 두 라틴 방격 $S_1 = \|a_{ij}\|$과 $S_2 = \|b_{ij}\|$에 대하여, n^2개의 정렬된 쌍인 (a_{ij}, b_{ij})가 모두 서로 다르다면, S_1과 S_2는 직교(orthogonal)한다고 말한다. (단, i=1,2,…,n 그리고 j=1,2,…,n이다.) 가장 작은 order를 가진 라틴 방격이 예제 7.1에 주어지고, 대응하는 정렬된 쌍인 (a_{ij}, b_{ij})가 주어진다.

예제 7.1. order n=3인 직교 라틴 방격

$$S_1=\begin{bmatrix}2&3&1\\1&2&3\\3&1&2\end{bmatrix} \quad S_2=\begin{bmatrix}2&1&3\\1&3&2\\3&2&1\end{bmatrix} \quad \begin{matrix}(2,2)(3,1)(1,3)\\(1,1)(2,3)(3,2)\\(3,3)(1,2)(2,1)\end{matrix}$$

다음으로, 두개 이상의 라틴 방격들로 이루어진 집합을 고려해 보자. 이러한 집합에 있는 라틴 방격의 각 쌍이 직교이면, 이 집합을 서로 직교인 라틴 방격들 (mutually orthogonal latin squares)로 이루어진 집합이라 한다.

정리 7.1. order n인 서로 직교인 라틴 방격들은 n−1개 이상 존재할 수 없다.

n-개의 order n인 서로 직교인 라틴 방격들로 이루어진 집합을 완전 집합(complete set)이라 한다. 서로 직교인 두 라틴 방격을 겹쳐 놓았을때, 겹쳐진 entry들의 각 정렬된 쌍은 n^2개의 쌍들 중 오직 한번 나타난다. 직교 특성을 보존하기 위해 두 라틴 방격에서 서로 대응하는 행들을 동시에 재배열해야 한다. 완전 집합에 대하여는, 각 라틴 방격의 제 1 행의 원소들이 자연수 순서, 즉, 1,2,…,n,대로 배열되도록 하면 된다.

정의 7.2. 서로 직교인 라틴 방격들로 이루어진 완전 집합이 다음 두 조건을 만족 할때 완전 표준 집합 (complete standardized set)이라 한다.

(1) 각 라틴 방격의 제 1행의 원소들이 자연수 순서대로 배열되어 있다.

(2) 이 집합의 한 라틴 방격의 제 1열이 자연수 순서대로 배열되어 있다.

정리 7.2. 소수의 멱승으로 표현될 수 있는 모든 정수 n에 대하여, order n인 서로 직교인 라틴 방격들로 이루어진 완전 표준 집합이 적어도 하나 존재한다.

이러한 집합을 만드는 방법은 참고문헌 [4]에서 찾을 수 있다.

7.2. Finite Projective Planes의 생성 알고리즘

이장에서는 Finite Projective Planes를 구축하는 알고리즘을 제안한다. Finite Projective Planes의 각 선들은 직교 라틴 방격의 완전 표준 집합에 의해서 만들어 진다. 정리 7.1과 7.2에 따라 이러한 라틴 방격의 order n이 소수의 멱승일 때 존재함을 알 수 있다.

알고리즘 : order m의 Finite Projective Plane의 구축

K=m+1이라 하고 K-2개의 표준 직교 라틴 방격들, $S_1, S_2, \cdots, S_{k-2}$이 주어진다고 하자. Finite Projective Plane의 각 선들은 다음과 같은 N×K 행렬의 각 행에 의해서 표현된다.

	W				
M_1	N_1				
	⋮				
	N_P				
M_2	N_1^T	V_{11}	V_{12}	····	V_{1q}
M_3	N_1^T	V_{21}	V_{22}	····	V_{2q}
⋮	⋮	⋮	⋮	⋯	⋮
M_k	N_1^T	V_{p1}	V_{p2}	⋯	V_{pq}

N × K

단, p=K-1이고 q=K-2 이며,

$W=[1\ 2\ \cdots\ K]_{1\times k}$

$$M_i = \begin{bmatrix} i \\ i \\ \vdots \\ i \end{bmatrix}_{(K-1)\times 1} \qquad 1 \le i \le K$$

$N_i=[K+1\ \ K+2\ \cdots\ K+(K-1)]_{1\times(k-1)} + [r\ r\ \cdots\ r]_{1\times(k-1)}$, $1 \le i \le K-1$이고 $r=(K-1)\times(i-1)$이다. (N_p의 마지막 원소는 N이어야 한다.)

N_1^T는 N_1의 행과 열을 교환함으로서 얻을 수 있는 N_1의 전치 행렬이다. 즉,

$$N_1^T = \begin{bmatrix} K+1 \\ K+2 \\ \vdots \\ K+(K+1) \end{bmatrix}_{(K-1)\times 1}$$

V_{ij} (i=1,2,…,K-1이고 j=1,2,…,K-2)는 크기가 (K-1)×(K-1)인 직교 라틴 방격들, $S_1, S_2, \cdots, S_{K-2}$,로 이루어진 완전 표준 집합으로 부터 얻어진다. 라틴 방격들을 다음과 같이 표시하자.

$S_1=[l_{ij}^1]_{(K-1)\times(K-1)}$

$S_2=[l_{ij}^2]_{(K-1)\times(K-1)}$

…

$S_{K-2}=[l_{ij}^{K-2}]_{(K-1)\times(K-1)}$

단, $l_{ij} \in \{1,2,\cdots,K-1\}$이다.

N_i와 라틴 방격들, $S_1, S_2, \cdots, S_{K-2}$에 대하여, V_{ij}는 다음과 같이 만들어 진다.

$$V_{ij} = \begin{bmatrix} N_{j+1}^{1} \text{ at } l_{i1}^{1} \text{ th row} \\ N_{j+1}^{2} \text{ at } l_{i2}^{1} \text{ th row} \\ \vdots \\ N_{j+1}^{K-1} \text{ at } l_{i(K-1)}^{1} \text{ th row} \end{bmatrix}_{(K-1)\times 1}$$

단, N_i^j는 N_i의 j번째 열에 있는 원소를 나타낸다.

알고리즘에 따라, V_{ij}는 N_{j+1}의 원소들로 이루어지고, 라틴 방격 S_j의 i번째 행이 V_{ij}의 각 원소의 위치를 결정하기 위해 사용되었음을 알 수 있다.

예제 7.2. N=13일때, N은 m=3의 함수로 표현될 수 있다. 즉, $13=3^2+3+1$이다. 그리고 3은 소수이므로 정리 2.4에 따라 order가 3인 Finite Projective Planes이 존재함을 알 수 있다. Finite Projective Planes의 각 집합들을 구축하기 위해, 직교 라틴 방격의 완전 표준 집합이 필요하다. m=3이므로, 직교 라틴 방격의 완전 표준 집합은 크기가 3×3인 2개의 라틴 방격, S_1과 S_2로 이루어 진다. S_1과 S_2는 다음과 같다.

$$S_1 = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix} \qquad S_2 = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix}$$

S_1과 S_2의 제 1행이 자연수 순서, 즉 1,2,3, 이고 S_1의 제 1열 또한 자연수 순서이므로, S_1과 S_2는 표준 직교 라틴 방격임을 알 수 있다. 알고리즘에 따라, Finite Projective Planes의 선들은 다음과 같은 크기가 13×4인 행렬로 표현될 수 있다.

	W		
M_1	N_1		
	N_2		
	N_3		
M_2	N_1^T	V_{11}	V_{12}
M_3	N_1^T	V_{21}	V_{22}
M_4	N_1^T	V_{31}	V_{32}

13×4

단, W=[1 2 3 4] N_1 = [5 6 7]
N_2 = [8 9 10] N_3 = [11 12 13]

$$N_1^T = \begin{bmatrix} 5 \\ 6 \\ 7 \end{bmatrix}$$

$$M_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad M_2 = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} \quad M_3 = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix} \quad M_4 = \begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix}$$

V_{11}, V_{21}과 V_{31}은 각각 S_1의 제 1행, 제 2행, 그리고 제 3행에 의해서 만들어 진다.

$$V_{11} = \begin{bmatrix} 8 \text{ at } 1 \text{ th row} \\ 9 \text{ at } 2 \text{ th row} \\ 10 \text{ at } 3 \text{ th row} \end{bmatrix} = \begin{bmatrix} 8 \\ 9 \\ 10 \end{bmatrix}$$

$$V_{21} = \begin{bmatrix} 8 \text{ at } 2 \text{ th row} \\ 9 \text{ at } 3 \text{ th row} \\ 10 \text{ at } 1 \text{ th row} \end{bmatrix} = \begin{bmatrix} 10 \\ 8 \\ 9 \end{bmatrix}$$

$$V_{31} = \begin{bmatrix} 8 \text{ at } 3 \text{ th row} \\ 9 \text{ at } 1 \text{ th row} \\ 10 \text{ at } 2 \text{ th row} \end{bmatrix} = \begin{bmatrix} 9 \\ 10 \\ 8 \end{bmatrix}$$

V_{12}, V_{22}과 V_{32} 역시 S_2를 사용하여 같은 방법으로 만들 수 있다.

$$V_{12} = \begin{bmatrix} 11 \text{ at } 1 \text{ th row} \\ 12 \text{ at } 2 \text{ th row} \\ 13 \text{ at } 3 \text{ th row} \end{bmatrix} = \begin{bmatrix} 11 \\ 12 \\ 13 \end{bmatrix}$$

$$V_{22} = \begin{bmatrix} 11 \text{ at } 3 \text{ th row} \\ 12 \text{ at } 1 \text{ th row} \\ 13 \text{ at } 2 \text{ th row} \end{bmatrix} = \begin{bmatrix} 12 \\ 13 \\ 11 \end{bmatrix}$$

$$V_{32} = \begin{bmatrix} 11 \text{ at } 2 \text{ th row} \\ 12 \text{ at } 3 \text{ th row} \\ 13 \text{ at } 1 \text{ th row} \end{bmatrix} = \begin{bmatrix} 13 \\ 11 \\ 12 \end{bmatrix}$$

마지막으로 위의 열 벡터(column vector)들을 이용하여 13개의 각 선들을 다음과 같이 생성할 수 있다.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 5 & 6 & 7 \\ 1 & 8 & 9 & 10 \\ 1 & 11 & 12 & 13 \\ 2 & 5 & 8 & 11 \\ 2 & 6 & 9 & 12 \\ 2 & 7 & 10 & 13 \\ 3 & 5 & 10 & 12 \\ 3 & 6 & 8 & 13 \\ 3 & 7 & 9 & 11 \\ 4 & 5 & 9 & 13 \\ 4 & 6 & 10 & 11 \\ 4 & 7 & 8 & 12 \end{bmatrix}$$

7.3. Labeling 알고리즘

7.2절의 알고리즘에 의해서 생성된 N×K행렬의 각 행은 전단사 함수 F에 의하여 유일한 점과 연관될 수 있다. 이 절에서는 전단사 함수 F를 이용하지 않고 N×K행렬의 각 행들을 유일한 점과 연관시키는 Labeling 알고리즘을 소개하기로 한다.

N×K행렬에서 각 행들은 순차적으로 i번째 행이 i번째 선이 되도록 라벨(Label)이 붙여질 수 있다. 그러나, 이 방법은 각 점 i가 선 L_i에 있으리라는 것을 보증 할 수 없다. 즉, 3장에서 Projective Plane의 각 점 i가 전단사 함수 F에 의하여 그 점 i와 인접한 유일한 선 L_i와 연관되므로 이러한 특징을 만족하도록 각 행들은 적절하게 라벨이 붙여져야 한다.

알고리즘 : Row Labeling
제 1행은 L_1으로 라벨을 붙인다.
for j : =1 **to** m **begin**
  N_j의 첫번째 원소를 i라 하자.
  N_j를 포함하고 있는 행을 L_i로 라벨을 붙인다.
end ;
for j : =1 **to** m **begin**
  for i : =N_j 의 두번째 원소 **to** N_j의 마지막 원소
    i와 j+1을 포함하고 있는 행을 L_i로 라벨을 붙인다.
  라벨이 붙여지지 않았으면서 j+1을 포함하고 있는 행을 L_{j+1}로 라벨을 붙인다.
end ;

다음 예제는 N=13인 경우에 Labeling 알고리즘을 수행한 결과이다.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 5 & 6 & 7 \\ 1 & 8 & 9 & 10 \\ 1 & 11 & 12 & 13 \\ 2 & 5 & 8 & 11 \\ 2 & 6 & 9 & 12 \\ 2 & 7 & 10 & 13 \\ 3 & 5 & 10 & 12 \\ 3 & 6 & 8 & 13 \\ 3 & 7 & 9 & 11 \\ 4 & 5 & 9 & 13 \\ 4 & 6 & 10 & 11 \\ 4 & 7 & 8 & 12 \end{bmatrix} \begin{matrix} L_1 \\ L_5 \\ L_8 \\ L_{11} \\ L_2 \\ L_6 \\ L_7 \\ L_{10} \\ L_3 \\ L_9 \\ L_{13} \\ L_4 \\ L_{12} \end{matrix}$$

7.4. 복잡도(Complexity)

직교 라틴 방격의 완전 표준 집합이 주어진다면, 이 알고리즘의 time complexity는 O(N×K)이다. K=m+1이고 N=m(m+1)+1이므로, 복잡도는 $O(N^{3/2})$ 또는 $O(K^3)$임을 알 수 있다. 또한, order (K-1)의 직교 라틴 방격의 완전 표준 집합이 $O(K^3)$안에 생산될 수 있다. 그러므로, 이 알고리즘을 사용해서, Finite Projective Planes의 구축은 $O(N\sqrt{N})$의 time complexity를 가지고 생성될 수 있다.

8. 결 론

본 논문에서는 시스템 내의 모든 site들에게 분산되어 있는 정보들을 매개변수로 갖는 함수를 계산하고, 그 계산 결과가 모든 site들에게 알려지도록 하는 다중 일치 프로토콜을 위한 효과적인 통신 방법을 제안했다. 각 site가 시스템내의 다른 모든 site들에게 자신의 정보를 전해주고 다른 모든 site들로부터 필요한 정보를 받음으로서 관련된 함수를 계산하는 방법은 한 round의 message 교환에 의해

일치를 이룰 수는 있지만 $O(N^2)$개의 message가 필요한 것이 단점이다. (단, N은 분산시스템에 참여하는 site들의 수이다.) 이러한 단점을 보완하기 위해, 본 논문에서는 Finite Projective Planes를 이용하여 각 site들이 자신과 관련된 일부 site들에게만 message를 보냄으로서 두 round의 message 교환과 각 round마다 $O(N\sqrt{N})$의 message를 사용하여 일치를 이루는 통신 방법을 제안했다. 또한, 이 통신 방법에서 이용되는 Finite Projective Planes의 각 site와 연관된 집합들을 구축하는 생성 알고리즘을 제안하였고, 이러한 통신방법이 최대, 최소값 계산, 합 계산과 같은 다중 일치 문제에 적용될 수 있음을 보였다.

암호학에 있어서, 각 가입자들은 필요한 message를 전달하는 것 외에도 message의 인증 또는 서명등을 위해 부가적인 message교환이 필요하다. 특히, 다자간의 통신하에서 시스템내의 모든 가입자들로부터 정보를 수집해야하는 회의용 키 분배 방식 (Conference Key Distribution System)에서 본 논문이 제시한 통신방법이 message의 수를 줄이기 위해 효과적으로 응용될 수 있으리라 기대된다.

참 고 문 헌

1. A.A. Albert and R. Sandler, *An Introduction to Finite Projective Planes*, New Tork: Holt, Rinehart and Winston, 1968.

2. J. Denes and A.D. Keedwell, *Latin Square and Their Applications*, Academic Press, New York, 1974.

3. D. Dolev, M. Klawe, and M. Rodeh, "An O(n log n) unidirectional distributed algorithm for extrema finding in a circle," *J. Algorithms*, vol. 3, pp. 245-260, Sept. 1982.

4. R.A. Fisher and F. Yates, *Statistical Tables for Biological Agricultural and Medical Research*, New York, Hafner Publishing Co., 1957.

5. G.H. Hardy and E.M. Wright, Theorem 418, p.343, in *An Introduction to the Theory of Numbers*, London, England: Oxford University Press, 1954.

6. D.S. Hirschberg and J.B. Sinclair, "Decentralized extrema-finding in circular configurations of processors," *Commun. ACM*, vol.23, pp. 627-628, Nov. 1980.

7. T.V. Lakshman and A.K. Agrawala, "Efficient Decentralized Consensus Protocols," IEEE Trans. Software Eng., vol. SE-12, May 1986, pp.600-607.

8. L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Commun. ACM*, pp.558-565, July. 1978.

9. C.L. Liu, "Matching Theory," in *Introduction to Combinatorial Mathematics*, New York: McGraw-Hill, 1968.

10. M. Maekawa, "A SQRT(N) algorithm for mutual exclusion in decentralized systems," *ACM Trans. Comput. Syst.*, pp.145-159, May. 1985.

11. S.J. Mullender and P.M. Vitanyi, "Distributed match-making for processes in computer networks," Centrum voor Wiskumde en Informatica, The Netherlands, Rep. CS-R8424, Dec. 1984.

12. G.L. Peterson, "An O(n log n) unidirectional algorithm for the circular extrema problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, pp.758-762, 1982.

13. F.S. Roberts, Sections 9.3.3, 9.3.4, and 9.5.2, in *Applied Combinatorics*, Englewood Cliffs, NJ: Prentice-Hall, 1984.

14. J.C. Ryou and J.Y. Juang, " A State Table Design for Load Balancing in Distributed Computing Systems," to appear in *IEEE Trans. on computers*.

15. D. Skeen, "Nonblocking commit protocols," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 1981, pp. 133-142.

16. D. Skeen and M. Stonebraker, "A formal model of crash recovery in a distributed system," *IEEE Trans. Software Eng.*, vol. SE-9, pp.219-

228, May 1983.

17. S.H. Son and A.K. Agrawala, "A non-intrusive checkpointing scheme in distributed database systems," in *Proc. 15th Annu. Int. Symp. Fault-Tolerant Comput.*, June. 1985, pp.99-104.

부 록

부록에서는 3장에서 사용된 전단사 함수의 존재를 증명한다.

$\pi=(P,L,I)$를 점들의 집합 P와 선들의 집합 L, 인접관계(incidence relation) I를 갖는 Finite Projective Plane을 나타낸다고 하자. P의 원소들을 P_1, P_2,…이라 하고, L의 원소들을 L_1, L_2,…라 하자. 한점 P_i가 선 L_i에 인접해 있을 때, (P_i,L_i)는 I에 속하게 된다. 마찬가지로, 선 L_i가 점 P_i를 지나면, (L_i,P_i)는 I에 속한다. 관계 I가 대칭(symmetric)임은 분명하다.

정리 : 어떤 $\pi=(P,L,I)$에 대하여, $F(P_i)=L_i$이라면, $(P_i,L_i)\in I$이도록 하는 전단사 함수 $F:P\rightarrow L$가 존재한다.

증명 : 그래프 $G(P\cup L,I)$를 고려해 보자. 그래프 G는 절점들의 집합 P와 L을 가진 undirected 이분 그래프이다. 정리 2.1, 2.2, 2.3에 따라, m을 Projective Plane의 order라 할때, $|P| = |L| = m^2+m+1$이다. 또한, G는 degree m+1의 regular 그래프이다. 그래프 G의 P-saturating matching, M은 구하고자 하는 전단사 함수 F를 주어진다. 그러므로, P-saturating matching, M의 존재를 증명하는 것으로 충분하다.

S을 P의 임의의 부분집합이라하고, N(S)를 S에 속해 있는 어떤 P_i와 연관된 모든 절점 L_i들의 집합이라 하자. Hall의 결혼정리(marriage theorem)에 따라, P의 모든 부분집합 S에 대하여, $|N(S)| \geq |S|$ 이라면, P-saturating matching이 존재함을 알 수 있다.[6)]

아래에서, 이것이 실제로 그래프 $G(P\cup L,I)$를 위한 경우임을 보여준다. 어떤 부분집합 S_k에 대해, $|N(S_k)| < |S_k|$라 가정하자. $|N(S_k)|$는 S_k의 점들과 인접한(incident) 서로 다른 선들의 총 갯수이다. 특별한 선은 하나 이상의 S_k의 점에 인접할지도 모른다. 그러므로 $|N(S_k)|$를 계산하기 위해, S_k의 각점에 인접한 선들의 갯수를 더하는 것으로 충분하지 않다. 이 합을 $|N(S_k)|$에 속해있는 한 선상에 있는 S_k의 평균 점들의 갯수 N_{av}에 의해서 나누어야 한다. 정리 2.1에 의해서

$$|N(S_k)| = \frac{|S_k| * \text{한점에 인접한 선들의 갯수}}{N_{av}}$$

이다. 정리 2.3에 따라, 이것은

$$|N(S_k)| = \frac{|S_k| * (m+1)}{N_{av}}$$

이다. 그러므로 가정에 따라,

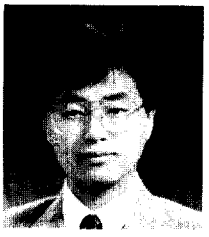
$$\frac{|N(S_k)|}{|S_k|} = \frac{m+1}{N_{av}} < 1$$

이다. 그러므로 $m+1 < N_{av}$이며, 정리 2.3에 모순이다.

□ 著者紹介

김 수 진

1991년 2월 충남대학교 통계학과 졸업
1991년 3월～현재 충남대학 대학원 전산학과 석사과정



류 재 철

1985년 2월 한양대학교 산업공학 학사
1988년 5월 Iowa State University 전산학 석사
1990년12월 Northwestern University 전산학 박사
1991년 2월～현재 충남대학교 전산학과 전임강사
관심분야 : 암호학, 분산시스템