# 대용량 그래프에서의 삼각형 검색 연구: 알고리즘과 응용

박하명, 강유
한국과학기술원

## 요 약

본 고에서는 다양한 네트워크를 표현하는 그래프에서 삼각형을 검색하는 알고리즘과 그 응용을 다룬다. 삼각형은 그래프에서 서로가 연결된 세 정점의 집합을 의미한다. 삼각형 검색 문제는 폭 넓은 응용이 가능하기 때문에 데이터 마이닝, 네트워크 분석 등 다양한 분야에서 중요하고 기본적인 문제로서 인식되어왔다. 삼각형 검색 문제의 중요성이 널리 인식되면서 여러 알고리즘이 제안 되어 왔지만, 최근의 소셜 네트워크, 웹 등의 크기가 방대해 기존의 방법은 이러한 네트워크를 분석하기가 사실상 불가능하다. 최근 맵리듀스를 활용한 분산/병렬 처리를 통해 대용량 그래프에서 삼각형을 검색하는 알고리즘들이 여럿 제안되었다. 본 논문에서는 지금까지 제안된 알고리즘들을 설명하고 삼각형 검색의 응용에 대해서 소개한다.

## Ⅰ. 서 론

소셜 네트워크에서 광고를 주 목적으로 하는 스팸 사용자들은 어떻게 검출할 수 있을까? 소셜 네트워크에 범람하는 컨텐츠들의 질을 어떻게 평가할 수 있을까? 웹에서 스팸 사이트들은 어떻게 걸러낼 수 있을까?

위의 예제들은 모두 그래프에서 삼각형을 찾는 문제와 관련이 깊다. 그래프는 소셜 네트워크의 친구 망을 포함하여 월드 와이드 웹(WWW), 인터넷, 통화 네트워크 등의 다양한 네트워크들을 정점과 간선으로 표현한다. 그래프에서 삼각형이란 서로 연결된 세 정점의 집합을 의미한다. 소셜 네트워크에서 예를 들면, 한 사용자에게 두 친구가 있고, 그 두 친구 역시 서로 친구라면, 이 세 사용자는 친구 망 그래프에서 삼각형을 이룬다.

그래프에서 삼각형을 찾는 문제는 오래 전부터 데이터 마이닝 분야와 그래프 분석 분야에서 중요한 문제로 인식되어왔다. 한 예로, 군집 계수 (clustering coefficient) [1] 는 그래프에서 한 정점이 이웃 정점들과 군집하려는 정도를 나타내는 수치로 두 분야에서 중요하게 여겨지는데, 이 계수를 계산하기 위해서는 삼각형의 수를 계산해야만 한다. 군집 계수는 소셜네트워크의 가짜 사용자를 찾아내거나 [2], 웹에서 악의적 페이지(malicious pages)나 성인 사이트등을 걸러내는 등 [3] 다양한 방면에서 활용된다. 이 외에도 삼각형을 찾는 문제는 네트워크 스펙트럼 분석(spectral analysis) [4] 과 커뮤니티 탐색(community defection) [5], 소셜네트워크의 컨텐츠 분석 및 평가(qualifying quality of contents) 등에도 응용된다 [3]. 이 외에도 다양한 응용들이 이전 연구들에서 소개되었다 [1][12][14][15].

삼각형 검색 문제의 다양한 활용과 그 중요성 덕분에, 사회 과학, 정보 공학, 전산학 등의 많은 연구자들에게 높은 관심을 받아왔다 [16][17][18]. 하지만, 대부분의 연구들은 인-메모리 알고리즘 (in-memory algorithm)에 관한 내용으로 [6][7][8], 이러한 알고리즘들은 최근 방대한 크기의 네트워크들을 사실상 처리가 불가능하다. 유명한 소셜 네트워크 서비스인 Facebook과 Twitter에는 각각 800만명 [1], 300만명 [2] 이상의 사용자가 활발하게 서비스를 이용 중이며, 2008년 구글의 발표에 따르면 1조 개 이상의 URL이 웹상에 존재 [3] 한다. 이러한 방대한 네트워크를 다루는 한 가지 방법은, 여러 대의 연산장치를 이용해서 병렬적으로 처리하는 것이다. 최근, 구글에서 제안한 맵리듀스(MapReduce) [9] 와 이의 오픈소스(open-source) 버전인 하둡(Hadoop) [4] 이 분산/병렬 처리의 사실상의 표준으로 자리매김하고 있다. 삼각형을 검색하는 알고리즘 역시 최근 몇몇의 연구자들에 의해 맵리듀스 알고리즘으로 제안되었다.

본 논문에서는 대용량 그래프에서 삼각형 검색을 위한 맵리듀스 알고리즘과 그 응용을 설명한다. Ⅱ 장에서는 삼각형 검색 문제를 정의하고 기존 삼각형 검색 알고리즘을 소개한다. Ⅲ

※본 연구는 산업통상자원부 및 한국산업기술평가관리원의 산업융합원천기술개발사업(정보통신)의 일환으로 수행하였음. [10044970, Symbolic Approach 기반 인간보사형 자가학습 지능 원천 기술 개발]

[1] http://newsroom.fb.com/company-info

[2] https://about.twitter.com/company

[3] http://googleblog.blogspot.kr/2008/07/we-knew-web-was-big.html

[4] http://hadoop.apache.org/

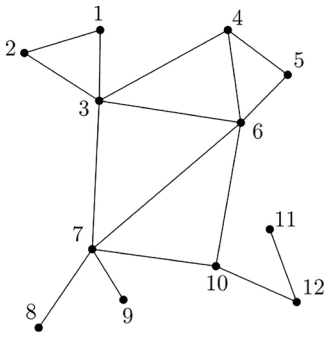



그림 1. 입력 그래프 예제

장에서는 맵리듀스와 하둡에 대한 소개와 맵리듀스 프로그램의 한 예를 보인다. Ⅳ 장에서는 최근에 제안된 삼각형 검색을 위한 맵리듀스 알고리즘들을 소개한다. Ⅴ 장에서는 삼각형 검색 문제의 몇 가지 응용을 소개하고, Ⅵ 장에서 결론과 함께 끝을 맺는다.

## Ⅱ. 삼각형 검색 알고리즘

삼각형 검색 문제는 주어진 그래프에서 삼각형을 모두 열거하는 문제다. 그래프 $G = (V, E)$ 는 정점 집합 $V$과 간선 집합 $E$의 쌍이다. 정점 집합 $V$ 에 포함되는 두 정점 $u, v$에 대해서, $u$ 와 $v$ 를 연결하는 간선은 $\{u, v\}$ 혹은 $\{v, u\}$로 표현되며 이는 정점 $u$와 $v$가 연결되어 있음을 의미한다. 그래프에서 삼각형이란 서로 연결된 세 정점 집합을 의미하는데, 즉, $\{u, v\}$, $\{v, n\}$, $\{u, n\}$ 가 간선 집합 $E$ 에 모두 포함될 때 $\{u, v, n\}$ 은 하나의 삼각형이다. 즉, 삼각형 검색 문제는 다음과 같이 정의할 수 있다.

정의 1. (삼각형 검색 문제) 그래프 $G = (V, E)$가 주어졌을 때, $\{u, v\}$, $\{v, n\}$, $\{u, n\}$ 이 모두 간선 집합 $E$에 포함되며 $u$, $v$, $n$ 이 모두 정점 집합 $V$ 에 포함되는 세 정점 집합 $\{u, v, n\}$을 모두 열거하라. 즉, $\{\{u, v, n\} \mid \{u, v\}, \{v, n\}, \{n, u\} \in E, u, v, n \in V\}$를 구하라.

〈그림 1〉 은 12 개의 정점으로 구성된 그래프의 한 예이다. 이 그래프에는 총 5 개의 삼각형이 존재하며, 삼각형 검색 문제의 결과는 {{1,2,3}, {3,4,6}, {4,5,6}, {3,6,7}, {6,7,10}} 이다.

삼각형 검색을 위한 다양한 알고리즘들이 수 년에 걸쳐 제안되었다[6][7][8]. 정점 순회 알고리즘(Vertex iteration algorithm)과 간선 순회 알고리즘(Edge iteration algorithm)이 대표적이다. 한 정점 $u \in V$ 와 연결된 모든 정점 집합을 $N(u)$로 표현하고 이웃 정점 집합이라고 정의한다. 즉, $N(u) = \{v \in V \mid \{u, v\} \in E\}$. 정점 순회 알고리즘은 모든 정점 $u \in V$ 를 순회하며 모든 가능한 $v, n \in N(u)$ 에 대해 $\{v, n\}$ 이 간선 집합 $E$ 에 포함되는지 확인한다. 포함되면 $\{u, v, n\}$을 출력한다. 간선 순회 알고리즘은 모든 간선 $\{u, v\} \in E$ 를 순회하며 $n \in N(u) \cap N(v)$ 인 모든 $n$에 대해 $\{u, v, n\}$을 출력한다. $d_{max}$ 가 모든 정점에 대해 최대 이웃 정점 집합의 크기를 의미할 때, 두 알고리즘 모두 같은 시간 복잡도 $O(|V|{d_{max}}^2)$ 를 갖는다. 두 알고리즘 모두 한 삼각형을 세 번씩 출력하는 단점이 있는데, 이는 정점 집합 $V$가 순완전순서(static total order)를 만족하도록 정점에 순서를 부여하여 해결할 수 있다. 이 때, 정점의 순서를 이웃 정점 집합의 크기로 정하면 두 알고리즘 모두 $O(|E|^{3/2})$ 의 시간 복잡도를 갖게 된다. 최근 여러 연구에서 정점 순회 알고리즘과 간선 순회 알고리즘의 변종인 Forward, Compact Forward [7], NodeIterator+ [10], NodeIteratorN [11] 등의 삼각형 검색 알고리즘들이 제안 되었다. 이들은 모두 같은 시간 복잡도 $O(|E|^{3/2})$ 를 갖으며 입력 데이터의 형태에 따라 서로 약간의 수행시간 차이를 보인다. 하지만, 이러한 알고리즘들은 메모리의 크기가 무한대라는 비현실적인 가정을 전제하기 때문에, 근래에 생성되는 방대한 크기의 그래프를 처리하지 못한다. 이와 같이, 메모리 크기보다 큰 그래프를 처리하기 위한 한 가지 방법은 분산 병렬 처리가 가능하도록 알고리즘을 설계하는 것이다. 하지만, 이러한 분산 병렬 처리 알고리즘을 작성하는 일은 결함 내성, 동기성 등 신경 써야 하는 부분이 많기 때문에 쉽지 않다. 다음 장에서 설명할 맵리듀스는 프로그래밍 모델에 몇 가지 제한을 둠으로써 이러한 분산 병렬 처리 알고리즘을 쉽게 작성할 수 있도록 한다.

## Ⅲ. 맵리듀스와 하둡

맵리듀스 [9] 와 그의 오픈소스 버전인 하둡은 대용량의 데이터를 분산된 연산장치에서 병렬로 처리를 위해 제안된 소프트웨어 프레임워크이다. 맵리듀스라는 이름은 함수형 언어들에 자주 등장하는 맵(map) 함수와 리듀스(reduce) 함수로부터 유래되었다. 맵리듀스를 통해 분산/병렬 컴퓨팅을 활용하기 위해서는, 맵 함수와 리듀스 함수만을 설계하면 된다. 일반적으로 맵 함수는 주어진 데이터를 여러 연산장치에 분배하도록 데이터를 변형하는 역할을 하고, 리듀스 함수는 분배된 데이터를 처리하는 역할을 한다. 맵리듀스는 맵 단계(map step), 셔플 단계(shuffle step), 리듀스 단계(reduce step)의 세 단계를 거쳐 진행된다.

맵 단계에서는 각 연산장치에서 맵 함수를 이용해 데이터를

병렬적으로 처리한다. 데이터는 한 줄씩 맵 함수에 전달되고, 맵 함수는 이 한 줄을 처리하여 〈키;값〉의 쌍을 출력한다. 출력되는 〈키;값〉의 쌍은 0개일 수도 있고 많을 수도 있는데 이는 맵 함수에 따라 다르다.

**셔플 단계**에서는 맵 단계에서 출력한 모든 〈키; 값〉 쌍들 중, 같은 키를 갖는 값들이 한데 모인다. 즉, 〈키;값 집합〉 의 형태로 데이터가 변형되어 출력된다.

**리듀스 단계**에서는 각 연산장치에서 리듀스 함수를 이용해 데이터를 병렬적으로 처리한다. 맵 함수의 입력은 셔플 단계에서 출력된 〈키; 값집합〉이고, 리듀스 함수는 이를 처리하여 〈키; 값〉들을 출력한다. 맵 함수와 마찬가지로, 리듀스 함수에 따라 출력되는 〈키; 값〉의 쌍이 없을 수도, 많을 수도 있다.

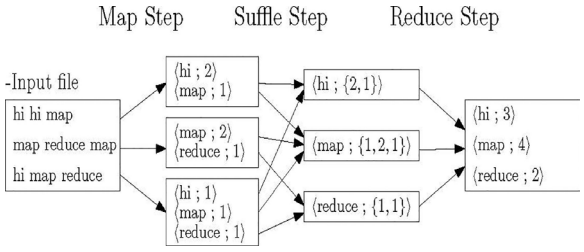



그림 2. 맵리듀스 단어 수 세기 예제

〈그림 2〉 는 맵리듀스를 이용한 단어 수 세기 예제이다. 이 예에서는 세 개의 매퍼(mapper)와 세 개의 리듀서(reducer)가 있고, 입력 데이터의 각 줄이 한 매퍼에서 실행 되며 셔플로부터 출력된 〈키; 값 집합〉들이 각 리듀서에서 하나씩 입력으로 주어질 때를 가정한다.

## Ⅳ. 대용량 그래프에서 삼각형 검색을 위한 맵리듀스 알고리즘

최근 몇 년 사이에 삼각형 검색을 위한 맵리듀스 알고리즘이 여럿 제안되었다. 본 장에서는 최근 제안된 맵리듀스 삼각형 검색 알고리즘을 소개한다.

### Cohen 알고리즘 [12]

처음 소개할 알고리즘은 정점 순회 알고리즘을 맵리듀스 알고리즘으로 변형 한 것으로, 연구자의 이름에 따라 Cohen 알고리즘이라 부른다. Cohen 알고리즘은 두 번의 맵리듀스 작업을 필요로 한다. 첫 번째 작업의 맵 함수에서는 간선 하나를 입력으로 받아 순서가 낮은 정점 $u \in V$ 를 키로, 순서가 높은 정점 $v \in V$ 를 값으로 하는 $< u; v >$ 쌍 하나를 출력한다. 리듀스 함수에서는 한 정점 $u \in V$ 을 키로, 그 정점의 이웃 정점들 중 순서가 높은 정점들의 집합 $N^+(u)$ 을 값으로 하는 $< u; N^+(u) >$ 쌍 하나를 입력으로 받는다. 그리고 $v, n \in N^+(u)$ 이고 $v$ 이 $n$ 보다 순서가 낮은 모든 경우에 대해서, $(v, n)$ 을 키로, $u$ 를 값으로 하는 $< (v, n), u >$ 쌍을 출력한다. 즉, $N^+(u)$ 에 포함된 정점의 수를 $d$ 라 할 때 $\binom{d}{2} = d(d-1)/2$ 개의 쌍을 출력한다. 이 키 값은 정점 $v$ 와 $n$ 이 정점 $u$ 에 의해 연결되어 있음을 의미하며, 만일 간선 $\{v, n\}$ 이 간선 집합 $E$ 에 포함된다면 $u, v, n$ 은 삼각형을 이루게 된다. Cohen 알고리즘의 두 번째 맵리듀스 작업은 정확하게 이와 같은 원리에 따라 동작한다. 두 번째 작업의 맵 단계의 입력은 입력 그래프의 간선 집합과 첫 번째 작업의 결과물을 모두 받는다. 즉, 맵 함수는 간선 하나 또는 첫 번째 작업의 결과물 중 한 $< (v, n), u >$ 쌍을 입력으로 받는다. 만일 맵 함수의 입력이 간선이고, 간선의 두 정점 중 $u$ 가 순서가 낮은 정점, $v$ 가 순서가 높은 정점이라면, $(u, v)$ 를 키, 특수문자(예를 들면 $)를 값으로 하는 $< (u, v) ; \$ >$ 쌍을 출력한다. 만일 맵 함수의 입력이 첫 번째 작업의 결과물 $< (v, n); u >$ 쌍이라면 그대로 출력한다. 리듀스 함수에서는 정점 쌍 $(u, v)$를 키로, 집합 $C \subseteq V \cup \{\$\}$ 를 값으로 하는 $< (u, v), C >$ 를 입력으로 받는다. 만일 $C$ 가 특수문자 $ 를 포함하면 각각의 $n \in C \setminus \{\$\}$ 에 대해서 $\{n, u, v\}$ 를 출력하고, $ 를 포함하지 않을 경우에는 아무것도 출력하지 않는다. 특수문자 $ 는 키의 두 정점 $u, v$ 에 대해서, 간선 집합 $E$에 $\{u, v\}$ 가 포함되어 있음을 의미하고, $C$ 에 포함된 각 정점은 $u, v$ 를 연결하는 형태임을 의미하기 때문에, $ 가 있을 경우에는 $C$ 에 포함된 모든 정점 $n$ 에 대해서 삼각형 $\{n, u, v\}$ 를 이루는 것이다.

정점 순회 알고리즘과 마찬가지로, 정점의 순서를 이웃 정점 집합의 크기로 정하면 Cohen 알고리즘 역시 총 $O(|E|^{3/2})$ 의 연산량을 나타낸다. 하지만, 첫 번째 작업의 결과 데이터의 크기 역시 $O(|E|^{3/2})$ 이며, 이는 두 번째 작업의 셔플 단계에서 역시 $O(|E|^{3/2})$ 의 데이터가 셔플되어야 함을 의미한다. 입/출력 데이터의 크기에 비해 중간 데이터가 폭증(intermediate data explosion)하는 현상이 발생하기 때문에, 맵리듀스 알고리즘임에도 불구하고 대용량 데이터를 처리하는데 있어서 제약사항이 있다. 이 뿐만 아니라 한 정점이 메모리에 들어차지 못하는 매우 많은 이웃 정점을 가질 경우에 이 알고리즘은 해당 그래프를 처리할 수 없다.

### GP 알고리즘 [10]

GP알고리즘은 그래프 분할(Graph Partitioning) 기법을 사

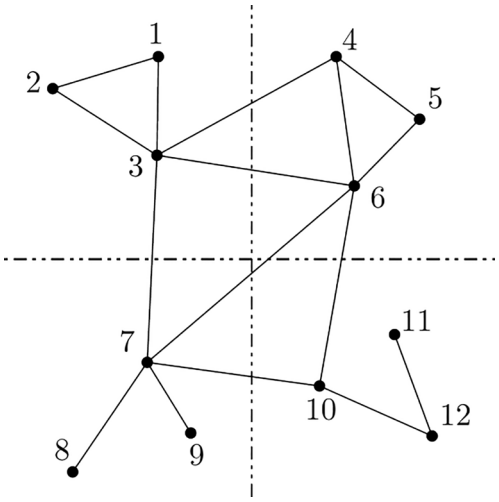



그림 3. 그래프 분할 예제

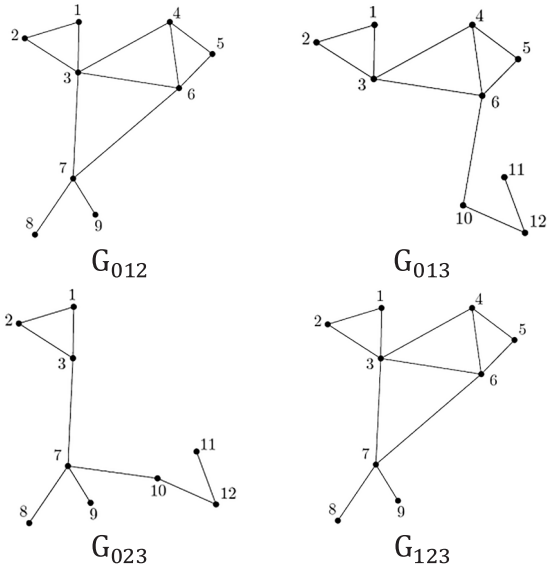


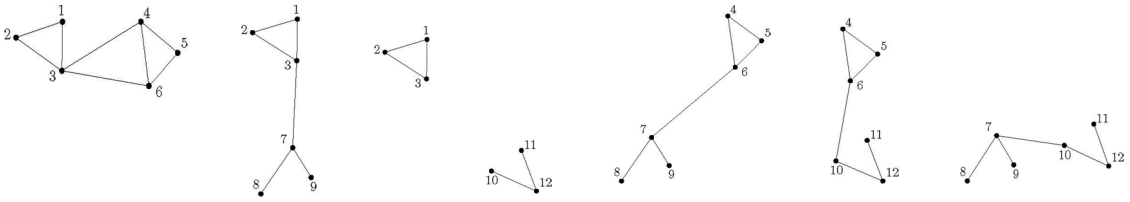

그림 4. GP 알고리즘 유도 서브그래프 예제

용하여 삼각형을 검색한다. 삼각형은 세 정점 집합이기 때문에, 그래프의 정점 집합 V 를 2 이상인 p 개의 부분 집합으로 나눌 경우에, 한 삼각형의 세 정점은 최대 세 개의 부분 집합안에 반드시 포함된다. GP 알고리즘은 이러한 원리를 이용하여 삼각형을 검색한다. 그래프의 정점집합 V 를 서로 독립적인 p 개의 부분 집합 $V_0$ , … , $V_{p-1}$ 으로 나누고, 이 부분 집합들을 세 개씩 조합하여 조합된 정점 집합들로부터 유도된 서브 그래프 (vertex-induced subgraph) 를 만든다. 유도 서브 그래프는, 그래프 G = (V, E) 와 부분 정점 집합 V' ⊆ V 이 주어졌을 때, V' 를 정점 집합으로 하며 간선 집합 E 에서 간선의 두 정점이 모두 V' 에 포함되는 모든 간선의 집합 E' 를 간선 집합으로 하는 그래프 G' = (V', E') 이다. 이 때, 세 정점 집합 $V_i$ , $V_j$ , $V_k$ 의 합집합에서 유도된 서브 그래프를 $G_{ijk} = (V_{ijk}, E'_{ijk})$ 로 표현하며, p개의 부분 집합이 있을 때, 총 $\binom{p}{3} = p(p-1)(p-2)/6$ 개의 서브 그래프가 만들어진다. 각 서브 그래프에서 기존의 삼각형 검색 알고리즘을 수행하면 모든 삼각형을 빠짐없이 찾을 수 있다. 〈그림 3〉 은 〈그림 1〉 의 입력 그래프가 4 개로 분할된 경우를 표현하는 예제이다. 〈그림 4〉 는 GP 알고리즘에 의해 만들어지는 서브 그래프들의 예제이다.

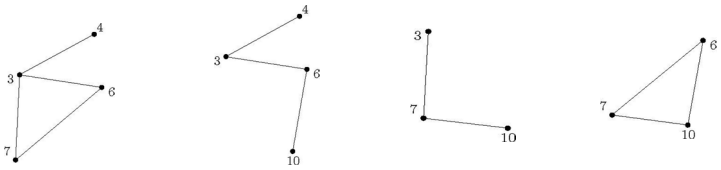
GP 알고리즘에서 맵 단계는 입력 그래프를 서브 그래프들로 나누는 일을 하고, 리듀스 단계에서는 서브 그래프들을 입력으로 받아 기존의 삼각형 검색 알고리즘을 수행한다. 맵 함수는 한 간선 {u, v} 를 입력으로 받아 해당 간선이 포함되는 서브 그래프를 나타내는 {i, j, k} 를 키로 하고 해당 간선을 값으로 하는 < {i, j, k}; {u, v} > 를 출력한다. 간선의 정점이 어떤 부분 정점 집합에 포함되는지는 임의의 해시 함수(hash function) h: V → [p] 에 의해 결정된다. 해시 함수 h 는 정점 u ∈ V 를 입력으로 받아 0보다 크거나 같고 p보다 작은 정수 (즉, [p])를 출력한다. 즉, h(u)는 정점 u 가 포함되는 부분 집합의 번호를 나타내며, 달리 표현하면 정점 u 가 부분 정점 집합 $V_{h(u)}$ 에 포함됨을 뜻한다. 리듀스 함수는 $< \{i, j, k\}; E'_{ijk} >$ 를 입력으로 받아 $G_{ijk}$ 에 존재하는 삼각형을 모두 출력한다.

m 을 하나의 리듀스 함수에서 사용 가능한 메모리 크기라 할 때, 부분 정점 집합의 수인 p 는 $\Omega(\sqrt{|E|/m})$ 로 정해진다. 그래프 G = (V, E) 에서 E 의 간선 중 두 정점 u, v 가 각각 $V_i$ , $V_j$ 에 포함되며 u 의 순서가 v 의 순서보다 우선하는 모든 간선의 집합을 $E_{ij}$ 라 하면, 즉 $E_{ij} = \{\{u, v\} \in E \mid u \in V_i, v \in V_j, u < v\}$, 총 $p^2$ 개의 $E_{ij}$ 가 존재한다. 모든 간선은 하나의 $E_{ij}$ 에 포함되므로, 간선이 고르게 분포된다고 가정하면 한 $E_{ij}$ 는 $|E|/p^2$ 개의 간선을 포함하게 된다. 서브 그래프 $G_{ijk} = (V_{ijk}, E'_{ijk})$ 의 간선 집합 $E'_{ijk}$ 는 $E_{ii}$ , $E_{ij}$ , $E_{ik}$ , $E_{ji}$ , $E_{jj}$ , $E_{jk}$ , $E_{ki}$ , $E_{kj}$ , $E_{kk}$ 의 합집합이므로, 한 서브 그래프는 $9|E|/p^2$ 의 간선 수를 가진다. 서브 그래프에서 기존의 삼각형 검색 알고리즘을 수행하려면 모든 간선이 메모리에 들어가야 하므로 한 간선을 표현하기 위한 데이터 크기를 e 라 할 때 $9|E|e/p^2 \le m$ 가 성립 되어야 한다. e 는 상수이므로 $p = \Omega(\sqrt{|E|/m})$라 할 수 있다.

GP 알고리즘의 리듀스 함수에서는 기존의 어떤 삼각형 검색 알고리즘을 사용해도 무방하다. 한 서브 그래프의 간선의 수는 $9|E|e/p^2$ 이므로, $O(|E|^{3/2})$ 의 복잡도를 보이는 알고리즘을 사용하면 서브 그래프를 처리하는데 드는 작업 량은 $O((|E|^{3/2}/p^3))$이다. 총 $p(p-1)(p-2)/6 = O(p^3)$ 개의 서브 그래프를 처리해야 하므로, 전체 필요한 작업의 수는 $O((|E|^{3/2}/p^3$



(a) 유형 1과 유형2 삼각형 검색을 위한 $G_{ij}$ 서브 그래프



(b) 유형3 삼각형 검색을 위한 $G_{ijk}^{-}$ 서브 그래프

그림 5. TTP 알고리즘 유도 서브그래프 예제

$)) \times O(p^3) = O(|E|^{3/2})$ 으로, GP 알고리즘은 기존 알고리즘과 동일한 시간 복잡도를 갖는다.

간선 집합 E에 포함된 간선 $\{u, v\}$ 의 두 정점이 동일한 부분 정점 집합에 포함되는, 즉 $\{u, v\} \in E_{ii}$, 내부 간선(inner-edge)은 맵 함수에서 $\binom{p-1}{2} = \frac{(p-1)(p-2)}{2}$ 번 중복하여 출력 되고, u 와 v 가 다른 부분 정점 집합에 포함되는 외부 간선(outer-edge)은 $p-2$ 번 출력된다. E 에 포함되는 내부 간선의 수의 기대값은 $\frac{|E|}{p^2} \times p = \frac{|E|}{p}$ 이고, 내부 간선이 $\frac{(p-1)(p-2)}{2}$ 번 출력 되므로, 맵 단계는 내부 간선을 $\frac{|E|(p-1)(p-2)}{2p}$ 번 출력한다. 외부 간선 역시 비슷한 계산 방법으로 $\frac{|E|(p-1)(p-2)}{p}$ 번 출력됨을 알 수 있다. 이는 셔플 단계에서 셔플되는 간선의 수가 $\frac{3}{2} \times \frac{|E|(p-1)(p-2)}{p}$ 라는 것을 의미하며, p 를 $\Theta(\sqrt{|E|/m})$ 로 설정하면 $\Theta(|E|^{3/2}/\sqrt{m})$ 가 된다. Cohen 알고리즘에 비해 GP는 중간 데이터의 크기가 $O(1/\sqrt{m})$ 의 요소만큼 적다. 하지만, 내부 간선이 맵 함수에 의해 $O(p^2)$ 번이나 중복되어 출력된다는 점과, 내부 간선들로만 이뤄진 삼각형이 리듀스 함수들에서 $O(p^2)$ 번 중복하여 출력되는 등의 단점을 지닌다.

## TTP 알고리즘 [13]

TTP 알고리즘은 GP 알고리즘과 마찬가지로 그래프 분할 기법을 사용하여 삼각형을 검색한다. GP 알고리즘의 단점인 내부 간선의 중복 출력 문제를 삼각형의 유형(Triangle Type)을 고려한 방법으로 해결한다. 그래프 $G = (V, E)$의 정점 집합 V 가 p 개의 부분 집합 $V_0$ , … , $V_{p-1}$ 으로 분할 되었을 때, 삼각형은 다음 세 가지로 분류할 수 있다.

유형 1. 삼각형의 모든 정점이 동일한 부분 정점 집합에 속한다.

유형 2. 삼각형의 모든 정점이 두 부분 정점 집합에 속한다.

유형 3. 삼각형의 모든 정점이 모두 다른 부분 정점 집합에 속한다.

GP 알고리즘은 유형 1과 유형 2의 삼각형을 불필요하게 중복하여 계산하는 문제점이 있는데, 그 이유는 이들 유형의 삼각형을 계산하는데 있어 부분 정점 집합이 하나 혹은 두 개만 있으면 되는데 반해 GP 알고리즘은 무조건 세 개의 부분 정점 집합을 조합한 서브 그래프에서 삼각형을 계산하기 때문이다.

TTP 알고리즘은 유형 1과 유형 2의 삼각형을 두 개의 부분 정점 집합 $V_i$ , $V_j$ 에서 유도된 서브 그래프 $G_{ij} = (V_{ij}, E'_{ij})$ 에서 검색하고, 유형 3의 삼각형을 세 개의 부분 정점 집합에서 유도된 내부 간선을 포함하지 않는 서브 그래프 $G_{ijk}^{-} = (V_{ijk}, E_{ijk}^{-})$ 에서 검색한다. 즉, $E'_{ij}$ 는 $E_{ii}$ , $E_{ij}$ , $E_{ji}$ , $E_{jj}$ 의 합집합이고, $E_{ijk}^{-}$ 는 $E_{ij}$ , $E_{ik}$ , $E_{ji}$ , $E_{jk}$, $E_{ki}$ , $E_{kj}$ 의 합집합이다. 유형 1 과 유형 2 의 삼각형을 별도로 계산함으로써 유형 3 의 삼각형 검색이 내부 간선 없이 외부 간선 만으로 가능해진다. 유형 1의 삼각형을 부분 정점 집합 하나에서 유도된 서브 그래프에서 찾지 않는 이유는, 유형 1의 삼각형을 이러한 방식으로 모두 검색을 하였더라도, 유형 2의 삼각형을 검색 할 때, 유형 1의 삼각형이 다시 검색되는 문제가 발생하기 때문이다. 〈그림 5〉 는 TTP 알고리즘에 의해 생성되는 서브 그래프들의 예제이다. 〈그림 5 (a)〉에 나열된 6 개의 서브 그래프에서 유형 1과 유형 2의 삼각형을 검색하고, 〈그림 5 (b)〉에 나열된 4 개의 서브 그래프에서 유형 3의 삼각형을 검색한다.

TTP 알고리즘의 맵 단계는 GP 알고리즘의 맵 단계와 마찬가지로 입력 그래프를 서브 그래프들로 나누는 일을 하고, 리듀스 단계 역시 서브 그래프들을 입력으로 받아 기존 알고리즘으로

삼각형을 검색한다. 맵 함수는 한 간선 {u, v} 를 입력으로 받아 해당 간선이 포함되는 서브 그래프를 나타내는 {i, j, − 1} 혹은 {i, j, k} 를 키로 설정한다. 여기서 키 {i, j,−1} 은 $G_{ij}$ 를, 키 {i, j, k} 는 $G_{ijk}^{-}$ 를 의미한다. 값은 각 그래프의 입력 간선 이다. h(u)를 정점 u가 속한 분할의 번호라고 하면, 입력 간선이 내부 간선인 경우, 즉 h(u) = h(v) 인 경우, 맵 함수는 i < h(u) < j 를 만족하는 모든 i 와 j 에 대해서 < {h(u), j, −1}; {u,v} > 와 < {i, h(u), −1}; {u, v} > 를 출력한다. 입력 간선이 외부 간선인 경우, 즉 h(u) ≠ h(v) 인 경우, h(u) 와 h(v) 중 더 큰 값을 high, 낮은 값을 low라고 했을 때, i < low < j < high < k 를 만족하는 모든 i, j, k 에 대해 < {i,low,high}; {u,v} >, < {low,j, high}; {u, v} >, < {low, high, k}; {u, v} >, < {low, high,−1}; {u, v} > 를 출력한다. 리듀스 함수는 < {i, j, −1}; $E'_{ij}$ > 또는 < {i, j, k}; $E_{ijk}^{-}$ > 를 입력 받아 기존 삼각형 검색 알고리즘으로 삼각형을 모두 출력한다.

부분 정점 집합의 수 p 는 GP 알고리즘에서와 마찬가지로 $\Omega(\sqrt{|E|/m})$ 로 정해진다. 서브 그래프의 종류에 따라 $9|E|/p^2$ 가 아닌 $6|E|/p^2$ 혹은 $4|E|/p^2$의 간선이 존재한다는 점 만을 주의하면 GP 알고리즘에서 p를 정하는 방법과 동일하다. TTP 알고리즘의 맵 단계에서는 내부 간선과 외부 간선 모두 p − 1 번 출력 된다. 또한 GP 알고리즘에서는 유형 2의 삼각형이 p − 2번, 유형 1의 삼각형이 (p − 1)(p − 2)/2 번 중복하여 출력되는데, TTP 알고리즘에서는 유형 1 의 삼각형 만이 p − 1번 중복하여 출력된다. 즉, 맵 단계에서 출력되는 간선의 수는 |E|(p − 1)로, GP 알고리즘이 출력하는 간선의 수 $\frac{3}{2}\times\frac{|E|(p-1)(p-2)}{p}$ 에 비해 $\frac{3}{2}\times\frac{p-2}{p}$ 배 적다.

TTP 알고리즘은 GP 알고리즘의 중복 현상을 삼각형 유형을 고려함으로써 효과적으로 제거하였다. 하지만, TTP 알고리즘 역시 $O(|E|^{3/2}/\sqrt{m})$ 만큼의 데이터가 셔플되어야 하고, 이는 입력 데이터의 크기 |E|에 비해 상당히 크기 때문에 중간 데이터의 폭증 현상이 여전히 남아있다고 할 수 있다.

## CTTP 알고리즘 [19]

Cohen, GP, TTP 알고리즘 모두 중간 데이터 폭증 문제를 안고 있다. 이들 알고리즘 모두는 하나 혹은 두 개의 맵리듀스 작업만을 수행하기 때문에 그러한 문제가 발생한다는 사실에 근거하여, CTTP 알고리즘은 다단계 접근법(Multi-round approach)를 통해 중간 데이터 폭증 문제를 해결한다. 맵리듀스 시스템의 모든 연산 장치에 대한 전체 가용 메모리 크기를 M 이라고 했을 때, 셔플되는 데이터의 크기를 M보다 작게 하는 것이 CTTP 알고리즘의 목표이다.

CTTP 알고리즘은 TTP 알고리즘과 동일한 서브 그래프들을 생성하고, 각 서브 그래프에서 삼각형을 검색한다. TTP 알고리즘과의 차이점은, TTP 알고리즘은 모든 서브 그래프를 하나의 맵 리듀스 작업에서 모두 처리하는 반면에, CTTP 알고리즘은 여러 맵리듀스 작업에서 나누어 처리한다. CTTP 알고리즘에서 총 맵리듀스 작업의 수를 R 이라고 할 때, TTP 알고리즘의 맵 단계에서 생성되는 데이터의 양이 $O(|E|^{3/2}/\sqrt{m})$ 이므로 $O(|E|^{3/2}/\sqrt{m})/R<M$ 이 만족되도록 R을 $O(|E|^{3/2}/\sqrt{m})$ M) 으로 설정한다. 그리하여, 각 맵리듀스 작업에서는 O(M)의 데이터를 셔플하게 된다.

CTTP 알고리즘에서 삼각형을 검색해야 하는 서브 그래프의 수를 K라 하면, K는 $G_{ijk}^{-}$ 서브 그래프의 수 $\binom{p}{3}$ 과 $G_{ij}$ 서브 그래프의 수 $\binom{p}{2}$ 의 합인 $p(p^2-1)/6$ 이다. CTTP는 매 맵리듀스 작업마다 K/R개의 서브 그래프를 처리한다. r번째 맵리듀스 작업에서 어떤 서브 그래프들을 처리할 지는 다음과 같이 정해진다.

*$i<j<k$ 이면서, $i+j+k\equiv r \mod R$ 인 모든 i, j, k에 대해서 $G_{ijk}^{-}$ 는 r번째 맵리듀스 작업에서 처리된다.*

*$i<j$ 이면서 $i+j\equiv r \mod R$ 인 모든 i, j 에 대해서 $G_{ij}$ 는 r번째 맵리듀스 작업에서 처리된다.*

CTTP 알고리즘의 맵 단계에서는 TTP 알고리즘의 맵 단계와 같은 일을 하지만, 각 r 번째 맵리듀스 작업에서 처리되는 서브 그래프만을 리듀서에 전달해야한다. 맵 함수는 간선 {u, v} 를 입력 받아 다음과 같이 처리한다. h(u) 와 h(v)가 다른 경우, h(u) 와 h(v) 중 큰 값을 high, 작은 값을 low라 할 때, i < low < j < high < k 를 만족하면서 i ≡ r−low−high mod R, j ≡ r−low−high mod R, k ≡ r−low−high mod R 를 만족하는 모든 i, j, k 에 대해서 < {i, low, high}; {u, v} >, < {low, j, high}; {u, v} >, < {low, high, k}; {u, v} > 를 출력한다. 또한, low+high≡r mod R 을 만족한다면 < {low, high, −1}; {u, v} > 도 출력한다. 만일 h(u)와 h(v) 가 같을 경우, i < h(u) < j 를 만족하면서 i ≡ r−h(u) mod R, j ≡ r−h(u) mod R 를 만족하는 모든 i, j 에 대해서 < {i, h(u)}, {u, v} >, < {h(u), j}, {u, v}> 를 출력한다. 리듀스 함수는 TTP 알고리즘과 마찬가지로 < {i, j, −1}; $E'_{ij}$ > 또는 < {i, j, k}; $E_{ijk}^{-}$ > 를 입력 받아 기존 삼각형 검색 알고리즘으로 삼각형을 모두 출력한다.

CTTP 알고리즘은 TTP 알고리즘과 완전히 동일한 서브 그래프들에서부터 삼각형을 검색한다. 그렇기 때문에 TTP와 마찬가지로 총 작업의 양은 $O(|E|^{3/2})$ 로 알려진 가장 빠른 삼각형 검색 알고리즘과 동일한 복잡도를 갖으며, 다단계 접근법을 통해 매 맵리듀스 작업마다 셔플되는 데이터의 수를 O(M)으로 줄임으

로써 중간 데이터 증폭 현상을 없앴다. 단, CTTP 알고리즘은 매 맵리듀스 작업마다 전체 데이터를 반복하여 읽어야 한다는 단점이 있지만, 데이터를 한번 읽는 작업은 삼각형을 검색하는 작업이나, 데이터를 셔플 하는 시간에 비해 부하가 크지 않다.

# V. 삼각형 검색의 응용

본 장에서는 삼각형 검색 문제를 활용하여 소셜 네트워크, 웹 그래프 등 다양한 네트워크를 분석하는 여러 가지 사례에 대해 소개하고자 한다.

## 소셜 네트워크 분석

소셜 네트워크 데이터는 다음 두 종류의 네트워크를 포함한다.

- 친구 망 (friendship network): 소셜 네트워크 이용자와, 이들 간의 친구 관계 정보를 담고 있다. 이는 이용자를 정점으로, 친구 관계를 간선으로 하는 그래프로 표현할 수 있다. 예를들면, Facebook에서는 사용자들의 친구 관계가, Twitter에서는 follow 관계가 친구 망으로 표현된다.
- 구독 망 (subscription network): 소셜 네트워크 서비스에서는 사용자가 글이나 사진 등의 컨텐츠를 서비스에 게시한다. 다른 사용자는 이와 같은 컨텐츠를 '구독'할 수 있다. 예를 들어, Facebook에서 사용자들은 '좋아요' 기능을 이용해 특정 컨텐츠에 대한 관심을 표현할 수 있고, Twitter 역시 'retweet'이라는 유사한 기능이 있다. 즉, 구독 망은 사용자와 컨텐츠가 정점이고, 컨텐츠에 대한 사용자의 관심을 간선인 그래프로 표현된다.

다음은 소셜 네트워크 데이터 분석에 삼각형 검색을 활용한 사례이다.

**스팸 사용자 탐지** [2]: 친구 망 데이터에서 군집 계수를 활용하여 스팸 사용자들은 탐지할 수 있다. 스팸 사용자는 광고를 주 목적으로 하기 때문에, 자신의 컨텐츠를 많은 사람들에게 알리려는 목적으로 많은 사람들과 친구관계를 맺는다. 하지만, 일반 사용자와는 달리, 이러한 친구들 사이에는 어떤 연관성도 없기 때문에, 서로서로 친구일 확률이 낮다. 다시 말하면, 낮은 군집 계수를 갖는다. 이러한 특징을 이용해서 일반 사용자와 스팸 사용자를 구분할 수 있다.

**컨텐츠의 질 측정** [3]: 기존에는 컨텐츠가 구독된 수를 기준으로 컨텐츠의 질을 평가하였지만, 이 방법에는 한가지 맹점이 존재한다. 예를 들어서, 어떤 사용자가 극단적으로 많은 사용자에게 동시 다발적으로 특정 컨텐츠를 홍보한다면 (즉, 스팸), 이 컨텐츠는 낮은 질로 평가되어야 함에도 불구하고 많은 구독 수 덕분에 좋은 질로 평가된다. 이러한 문제를 해결하기 위해 친구 망과 구독 망을 합친 복합 망에서 군집 계수를 계산함으로써 보다 정확하게 컨텐츠의 질을 측정할 수 있다. 홍보성 컨텐츠의 경우, 구독자가 많을 지라도, 구독자 사이에 관계가 없기 때문에 서로 친구일 확률이 낮다. 즉, 낮은 군집 계수를 보이는데, 일반적인 컨텐츠의 경우에 게시자 혹은 구독자의 주변인들이 마찬가지로 구독을 하게 되어 높은 군집 계수를 보이는 것과 상반된다. 이러한 특징을 이용하여 컨텐츠의 질을 보다 정확하게 평가할 수 있다.

## 웹 그래프 분석

웹 그래프는 웹에서 페이지 혹은 사이트를 정점으로 하고, 페이지들 사이의 하이퍼링크를 간선으로 하는 그래프다. 다음은 웹 그래프에서 삼각형 검색을 활용한 사례다.

**스팸 페이지 검출** [3]: 구글, 네이버 등과 같은 웹 검색 서비스들은 사용자가 키워드로 검색을 하면, 이를 포함하는 웹 페이지들 중에 순위가 높은 순으로 사용자에게 보여준다. 페이지들의 순위는 PageRank 등의 알고리즘으로 결정된다. 스팸 페이지는 PageRank와 같은 페이지 순위 결정 알고리즘을 속여 자신의 페이지의 순위를 높게 하여 사용자들에게 더 잘 노출되도록 한다. 이러한 스팸 페이지들을 검출하기 위해서 삼각형 검색이 활용된다. 스팸 페이지들은 순위를 올리기 위해서 서로가 서로를 하이퍼링크로 연결하기 때문에 일반적인 페이지들에 비해 많은 삼각형을 포함하게 된다. 비 정상적으로 많은 삼각형을 갖는 페이지들을 검출함으로써 스팸 페이지를 판별해 낼 수 있다.

**웹 테마 검색** [15]: 웹 상에서 특정 주제를 갖는 사이트를 찾

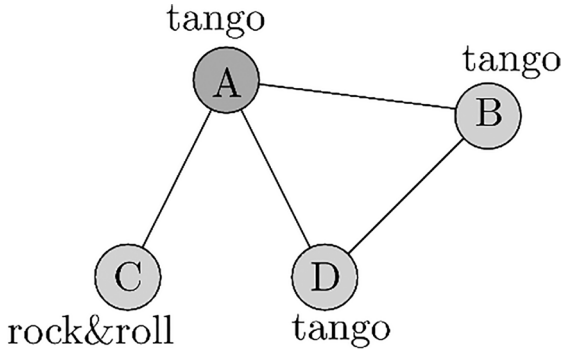



그림 6. 웹 테마 검색 예제

아내기 위해서 삼각형 검색이 활용된다. 〈그림 6〉은 '탱고'에 관한 사이트 A 와, 이 사이트와 하이퍼링크로 연결된 사이트 B, C, D를 표현한 예제이다. 이 에에서 사이트 B와 D는 A와 마찬가지로 '탱고'에 관한 사이트이기 때문에 서로 하이퍼링크로 연결되어있다. 반면, 사이트 C는 B, D와는 다른 주제를 다루기 때문에 이들 사이는 연결되지 않았다. 이처럼, 같은 주제를 다루는 사이트의 경우는 서로 하이퍼링크로 연결되어 있을 확률이 높지만, 연관이 없는 주제를 다루는 사이트들은 서로 연결되어 있을 확률이 낮다. 즉, 같은 주제를 다루는 사이트들은 서로 연결되어 삼각형을 이룬다. 이러한 특성에 따라 특정 사이트와 유사한 주제를 다루는 사이트를 찾기 위해서 삼각형 검색이 활용될 수 있다. 더욱이, 사이트가 포함되는 삼각형의 수에 따라서 사이트의 중요도를 평가할 수 있다.

## Ⅵ. 결 론

본 논문에서는 대용량 그래프에서 삼각형 검색을 위한 맵리듀스 알고리즘과 그 응용에 대해 설명하였다. Cohen 알고리즘은 정점 순환 알고리즘을 변형한 맵리듀스 알고리즘이고, GP 알고리즘은 그래프 분할 기법을 활용하여 Cohen 알고리즘에서의 중간 데이터 증폭 문제를 줄였다. TTP 알고리즘은 삼각형 유형을 고려함으로서 GP 알고리즘에서의 불필요한 중복 출력을 줄였으며 CTTP 알고리즘은 다단계 접근법으로 중간 데이터 증폭 현상을 해결하였다.

월드 와이드 웹, 인터넷, 소셜네트워크 등 다양한 네트워크가 그래프로 표현될 수 있고, 그래프에서 삼각형을 검색하여 소셜네트워크의 가짜 계정, 스팸 계정들을 검출하는 등 다양한 분야에 활용될 수 있다. 최근 여러 연구자들에 의해 외부 메모리 알고리즘, 분산 알고리즘 등 다양한 방면에서 삼각형 검색 알고리즘들이 제안되고 있기 때문에, 지속적으로 발전 중인 컴퓨팅 환경에 발맞춰 개선된 삼각형 검색 알고리즘이 제안될 것으로 기대한다. 더불어, 이러한 알고리즘들의 활용으로, 최근 급속히 크기가 증가하는 다양한 네트워크들을 분석함으로써 그 특성을 이해하고, 이형데이터 검출(anomaly detection)을 통한 데이터 마이닝의 활성화를 기대한다.

## 참 고 문 헌

[1] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. Nature, 393(6684):440-442, 1998.

[2] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai. Uncovering social network sybils in the wild. In SIGCOMM, pages 259-268, 2011.

[3] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In SIGKDD, pages 16-24. ACM, 2008.

[4] U Kang, B. Meeder, E. E. Papalexakis, and C. Faloutsos. Heigen: Spectral analysis for billion-scale graphs. TKDE, 26(2), 350-362, 2014.

[5] J. W. Berry, B. Hendrickson, R. A. LaViolette, and C. A. Phillips. Tolerating the community detection resolution limit with edge weighting. Physical Review E, 83:056119, 2011.

[6] A. Itai and M. Rodeh. Finding a minimum circuit in a graph. SIAM Journal on Computing, 7(4):413-423, 1978.

[7] M. Latapy. Main-memory triangle computations for very large (sparse (power-law)) graphs. Theoretical Computer Science, 407(1):458-473, 2008.

[8] T. Schank and D. Wagner. Finding, counting and listing all triangles in large graphs, an experimental study. In Experimental and Efficient Algorithms, pages 606-609. Springer, 2005.

[9] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. OSDI, 2004.

[10] Siddharth Suri and Sergei Vassilvitskii. Counting triangles and the curse of the last reducer. In WWW, pages 607-614, 2011.

[11] Shaikh Arifuzzaman, Maleq Khan, Madhav V. Marathe. PATRIC: a parallel algorithm for counting triangles in massive networks. In CIKM 2013, pages 529-538, 2013

[12] Jonathan Cohen. Graph twiddling in a mapreduce world. Computing in Science & Engineering, 11(4):29-41, 2009.

[13] Ha-Myung Park, Chin-Wan Chung. An efficient MapReduce algorithm for counting triangles in a very large graph. In CIKM, pages 539-548, 2013

[14] Shumo Chu and James Cheng. Triangle listing in

massive networks and its applications. In KDD, pages 672-680, 2011.

[15] Jean-Pierre Eckmann and Elisha Moses. Curvature of co-links uncovers hidden thematic layers in the world wide web. PNAS, 99(9):5825-5829, 2002.

[16] Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. SIAM J. Comput., 14(1):210-223, 1985.

[17] Xiaocheng Hu, Yufei Tao, and Chin-Wan Chung. Massive graph triangulation. In SIGMOD, pages 325-336. ACM, 2013.

[18] Rasmus Pagh and Francesco Silvestri. The Input/Output complexity of triangle enumeration. In PODS, pages 224-233, 2014.

[19] Ha-Myung Park, Francesco Silvestri, U Kang, Rasmus Pagh. MapReduce Triangle Enumeration With Guarantees. In CIKM, pages to appear, 2014.

## 약 력

박 하 명

2011년 성균관대학교, 컴퓨터공학 학사
2011년~2013년 KAIST 웹사이언스공학 석사
2013년~현재 KAIST 전산학과 박사과정
관심분야: 대용량 그래프 마이닝, 대용량 기계학습

강 유

2003년 서울대학교, 컴퓨터공학 학사
2012년 Carnegie Mellon University, Computer Science 박사
2013년~현재 KAIST 전산학과 조교수
관심분야: 데이터 마이닝, 빅 데이터