특집 07

빅 그래프 데이터의 삼각 개수의 의미와 처리 기술

민준기 (한국기술교육대학교)

목 차 »

1. 서 론
2. 그래프 상의 삼각 개수의 의미
3. 삼각 개수의 응용분야
4. 그래프 상의 삼각형처리 기법
5. 결 론

1. 서 론

점 (vertex) 와 간선 (edge) 으로 이루어진 그래프는 개체들 간의 관계를 모델링하는 중요한 자료구조로 간주되어 왔으며 그래프 데이터를 분석하여 유의한 정보를 추출하는 다양한 기법들이 제안되었다. 최근에는 소셜 네트워크의 등장, 바이오 정보기술의 발전 등에 따라서 그래프는 다양한 응용분야에서 활용됨으로써 더욱더 중요한 자료구조로 인정받고 있으며 그래프의 크기 또한 기하급수적으로 증가되고 있다. 예를 들어 트위터 (twitter)의 경우 9 억명의 사용자 중 5억 명이상의 활동 중으로 활용 사용자들 간의 관계는 약 20억 개의 간선으로 표시될 수 있으며 야후 (Yahoo) 에 등록된 웹페이지는 약 14억 개로 웹페이지간의 연결 관계는 약 66억 개이다[1]. 이와 같이 거대한 그래프 구조를 통칭 “빅 그래프 데이터“ 라고 한다.

최근에 들어와서 소셜 네트워크, 블로그 네트워크, 단백질-상호작용 네트워크 등과 같은 빅 그래프 데이터의 특성을 파악하고 이를 활용하는 소셜 광고, 선거 예측, 추천 시스템 등 다양한 응용 분야들이 등장하고 있다. 특히, *삼각형 (triangle)* 은 복잡한 그래프 분석에 있어서 중요한 역할을 한다. 특히, 소셜 네트워크에서 삼각형은 동류성 (homophily) 과 이행성 (transitivity) 을 설명하는 주요한 구성요소이다. 동류성이란 유유상종으로 한 사용자는 자신과 비슷한 사용자들 친구로 삼는다는 것이고 이행성이란 동일한 친구들을 가진 사용자들이 서로 친구가 된다는 특성이다.

대량의 점과 간선들로 이루어진 빅 그래프 데이터에서 삼각 개수 (triangle count)를 구하는 것은 많은 시간과 비용을 요구하는 작업이다. 따라서, 최근 빅데이터 그래프에서 효율적으로 삼각 개수를 구하는 다양한 연구들이 진행되고 있다. 본 논문에서는 그래프 데이터 상의 삼각개수의 의미와 이를 활용한 응용 분야들 및 삼각형 및 개수

를 구하는 대표적인 기법들을 설명하고자 한다.

2. 그래프 상의 삼각 개수의 의미

2.1 배경지식

그래프의 삼각 개수에 대한 내용에 앞서 우선 그래프에 대한 기본 용어를 설명한다. 그래프 $G = (V, E)$는 정점들의 집합 V와 간선들의 집합 E로 구성되며, V의 크기 $|V|$ 는 n, E의 크기 $|E|$는 m이라 가정한다. 간선 집합 $E \subset V \times V$ 의 원소 e 는 두 개의 정점 $v, u \in V$을 연결하며 $e = <v,u>$로 표현한다. 이 경우 (즉, $e = <v, u> \in E$), v와 u를 '인접하다' (adjacent) 라고 하며 e는 v와 u에 '연결된다' (incident) 라고 한다.

정점 $v \in V$ 에 대하여 $N(v)$는 v와 인접한 정점들의 집합으로 $N(v) = \{u \in V \mid <v, u> \in E\}$ 이며 v의 차수(degree) $d(v)$ 는 v와 연결된 간선들의 수로 $d(v) = |N(v)|$이다. 이 때, 그래프 G의 최대 차수는 $d_{max}(G) = max\{d(v) : v \in V\}$로 정의된다.

그래프 G에서 순환 간선 (looping edge)이 없다면 (즉, $<u. u> \notin E$), G를 단순 그래프 (simple graph) 라고 하며 본 논문에서는 단순 그래프만을 고려한다. 단순 그래프 G에서 간선의 수 m은 ${}_nC_2$ 보다 항상 작거나 같으며 (즉, $m \leq n(n-1)/2$), 만약 $m = n(n-1)/2$ 이라면 G를 완벽 그래프 (complete graph) 또는 n-클리크 (n-clique) 라고 한다.

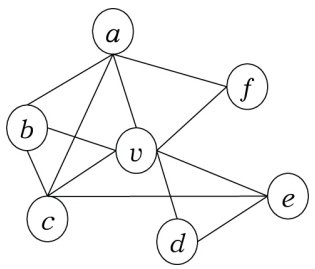
그래프 $G = (V, E)$ 의 부분 집합 $V' \subset V$, $E' \subset E$ 에 대하여, 각 간선 $e= <v, u> \in E'$ 와 연결된 정점 v와 u가 V' 에 존재한다면 $G' =(V', E')$은 G의 부분 그래프 (subgraph) 라고 한다. 정점의 부분집합 $V' \subset V$ 에 대하여 정점 축소 (node induced) 부분 그래프 $G[V']$ 은 정점 집합 V'와 간선 집합 $E'= \{<v, u> \in E \mid v, u \in V'\}$으로 구성된다. 간선 축소 부분 그래프 $G[E']$도 이와 유사하게 정의된다.

2.2 클러스터 계수 (cluster coefficient)와 이행도 (transitivity ratio)

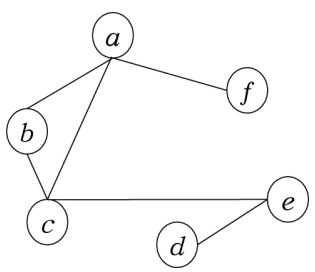
단순 그래프 G의 밀집도 (density)는 n개의 정점들로 만들 수 있는 최대 간선 수 ${}_nC_2$ 와 실제 G에 존재하는 간선들의 수 m 의 비율로 $p(G) = m/{}_nC_2$ 로 나타내며 $p(G)$가 1에 가까울수록 밀집하다 (dense) 라고 한다. 그림 1의 단순 그래프의 밀집도는 $12/{}_7C_2 = 12/21$이다. 그래프 밀집도는 계산이 편리하다는 장점과 모든 정점들이 얼마나 연결되어 있는지에 대한 정보를 제공한다. 그러나 한 정점에 인접한 정점들 간의 연관성 또는 밀집도를 표현하기에는 다소 무리가 있다. 한 정점의 인접한 정점들 간의 연관성을 나타내는 것이 클러스터 계수 (cluster coefficient)[2]이다.

정점 v의 클러스터 계수 $c(v)$는 v에 인접한 정점들로 이루어진 부분그래프의 밀집도로 표현된다. 즉 v의 인접 정점 집합 $N(v)$에 대한 정점 축소 부분 그래프 $G[N(v)]$의 밀집도이다. 그림 2는 그림 1의 그래프 G의 $N(v)$ 정점 축소 부분 그래프 $G[N(v)]$를 나타내고 밀집도 $p(G[N(v)])$는 $6/{}_{N(v)}C_2 = 6/15$이며 이는 v의 클러스터 계수 $c(v)$ 이다.

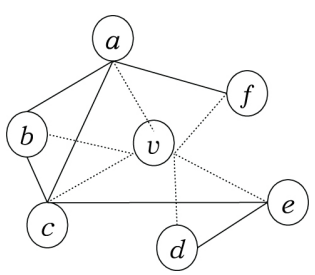
정점 v의 클러스터 계수를 구할 때, $|N(v)|$ (즉, v의 차수 $d(v)$)는 쉽게 구하지만 $G[N(v)]$의 간선들의 수를 계산하는 것이 필요하다. 여기서 흥미롭게도 $G[N(v)]$ 의 간선들의 수가 v를 꼭짓점으로 하는 삼각형의 개수와 동일하다. 그림 4에서는 v를 꼭짓점으로 하는 삼각형을 구성하는 3개의 간선들 중 v와 연결된 간선 두개는 점선으로 v와 연결되지 않는 하나의 간선은 실선으로 표시하였다. 이 때, 그림 3에서 실선으로 표시된 간선들의 집합이 그림 2의 간선들의 집합과 동일함을



(그림 1) 단순 그래프 G



(그림 2) $G[N(v)]$



(그림 3) 삼각형들

알 수 있다. 즉, v를 꼭짓점으로 하는 삼각형의 개수를 $\delta(v)$ 라고 할 때 $c(v) = \delta(v)/{}_{d(v)}C_2$ 이다.

그래프 G의 클러스터 계수 $c(G)$ $(= \sum_{v \in V} c(v)/|V|)$ 는 한 정점의 인접한 정점들이 평균적으로 얼마나 연관되어 있는지 나타는 것으로 모든 정점들 v의 클러스터 계수 $c(v)$의 평균으로 정의된다.

그래프 G의 이행도 (transitivity ratio) $t(G)$는 그래프 G 에서 이행관계 (transitive relation)이 발생하는 확률을 나타내며 $t(G) = (\sum_{v \in V} \delta(v))/ (\sum_{v \in V} {}_{d(v)}C_2)$ 로 정의된다. 그래프 상에 존재하는 삼각형의 개수를 $\Delta(G)$ 일 때, $\sum_{v \in V} \delta(v) = 3\Delta(G)$ 임으로 $t(G) = 3\Delta(G)/(\sum_{v \in V} {}_{d(v)}C_2)$ 로 표현된다.

즉, $t(G)$는 사용자 a, b, c에 대하여 a와 b가 친구이고 \ b와 c가 친구일 때, a와 c가 친구일 가능성을 표현한다. 그래프 G의 클러스터 계수 $c(G)$와 이행도 $t(G)$는 유사하지만 $c(G)$는 작은 차수를 가지는 정점들로 구성된 삼각형에서 좀 더 많은 가중치를 부여하고 $t(G)$는 이에 반하여 높은 차수를 가지는 정점들로 구성된 삼각형에 더 많은 가중치를 부여한다.

3. 삼각 개수의 응용분야

그래프 상의 삼각 개수 및 삼각 개수로 유도된 클러스터 계수는 다양한 응용분야에서 활용되고 있다. [3]에서는 웹-스팸(web-spam) 탐지에 클러스터 계수를 적용할 수 있음을 보였다. 웹-스팸이란 웹서치 엔진을 기만하여 성인물이나 광고물을 포함한 자신의 웹페이지를 상위 랭크에 올리는 것을 말한다. 이 경우 웹페이지를 상위로 올리기 위하여 호스트가 여러 웹 페이지들을 유지하고 이 웹 페이지들 간에 상호 연결이 되어 있는 것을 확인했다. 따라서 웹-스팸의 경우, 클러스터 계수 또는 삼각 개수가 일반적인 경우보다 매우 높다는 것으로부터 파악된 것이다. 또한 [3]에서는 소셜 네트워크에서 사용자의 역할을 파악하기 위하여 삼각 개수와 클러스터 계수를 활용할 수 있음을 보였다. 특히, Yahoo의 질의-응답 웹 페이지를 분석해 본 결과 좋은 질문 또는 좋은 응답을 하는 사용자들은 주로 삼각 개수가 크고 클러스터 계수가 작다는 것을 파악하였다.

[4]에서는 웹 페이지들의 내용을 보지 않고도 중요한 웹 페이지들을 파악하는 방법을 제시하면

서 클러스터 계수를 사용하였다. 이를 위하여 우선 웹 페이지들을 클러스터링하여 각 클러스터를 하나의 노드로 표현하고 노드들 간의 링크들을 기반으로 클러스터 계수를 계산하였다. 이를 통하여 클러스터 계수가 높은 노드들이 웹 환경에서 중요한 역할을 함을 보였다.

[5]에서는 트위터 (Twitter)나 링크드인 (LinkedIn)과 같은 소셜 네트워크에서 삼각 개수 기반의 링크 추천 (또는 친구 추천) 방법을 제시하였다. 기본적으로 친구의 친구는 친구인 경향이 있음으로 이는 삼각형으로 표현되고 이에 따라서, 가능한 많은 삼각형을 생성할 수 있는 링크들을 추천하는 것이다. 이외에도 다양한 분야에서 삼각 개수 및 클러스터 계수를 활용하고 있다.

4. 그래프 상의 삼각형 처리 기법

2절과 3절에서 삼각 개수의 의미와 삼각 개수를 활용하는 응용 분야에 대하여 살펴보았다. 본 절에서는 그래프 상에 존재하는 삼각 개수를 구하는 기법들에 대하여 기술한다.

4.1 일반적인 삼각형 처리기법

그래프 상의 삼각형의 개수를 구하는 가장 명확한 방법은 모든 가능한 3개의 정점들에 대하여 삼각형을 구성하는지 안하는지 조사해 보는 것이다. n개 정점들로 구성된 그래프 G 에 대하여 이 방법의 시간 복잡도는 $\Theta({}_nC_3)$로 $O(n^3)$ 이다.

삼각 개수를 구하는 또 다른 방법은 행렬의 곱을 이용하는 것이다. 그래프 G가 $n \times n$ 크기의 인접행렬 A로 표현되어 있다고 하자. 즉 정점 v_i가 정점 v_j가 인접하다면 $A[i][j]$가 1로 지정되어 있다. 이 경우 A^3의 i번째 대각 원소 (즉, $A^3[i][i]$)는 v_i에서 임의 두 정점 v_j, v_k 을 거쳐 v_i로 다시 돌아오는 경로의 수를 나타난다. 이때, v_i를 꼭짓점으로 하는 하나의 삼각형에서 대하여 v_i로 돌아오는 경로는 2개 ($v_i \rightarrow v_j \rightarrow v_k \rightarrow v_i$와 $v_i \rightarrow v_k \rightarrow v_j \rightarrow v_i$)가 존재하게 된다. 따라서 $A^3[i][i]$ 는 $2\delta(v_i)$ 와 같다. 또한 이 경우 $A^3[j][j]$와 $A^3[k][k]$ 에도 이 삼각형 개수의 2배가 포함되어 있게 된다. 따라서 A^3의 대각원소들의 합 ($\sum_{i=1\ldots n} A^3[i][i]$) 은 $2\sum_{v \in V} \delta(v) = 6\Delta(G)$ 이다. 따라서, 행렬의 곱을 통하여 삼각 개수를 구할 수 있다. 단순한 행렬의 곱셈 알고리즘의 시간 복잡도는 $\Theta(n^3)$ 이다. 그러나, [6]에서 행렬 곱의 시간 복잡도를 $O(n^{2.376})$ 으로 줄였으며 현재까지 알려진 가장 빠른 행렬의 곱셈 알고리즘이다.

*NodeIterator*는 그래프 상의 각 정점 v 에 대하여 인접한 두 정점 u와 w가 인접하였는지를 검사하여 삼각형을 파악한다. *NodeIterator* 알고리즘은 그림 4와 같으며 알고리즘의 맨 마지막 라인 각 삼각형이 한번만 출력되도록 한다. 정점 v에 대하여 모든 인접한 정점 쌍을 고려함으로

```
Algorithm NodeIterator(G = (V,E))
for v ∈ V do
    for each pair {u, w} in N(v) do
        if {u, w} ∈ E then
            if v < u < w then output triangle {v, u, w}
```

(그림 4) *NodeIterator* 알고리즘

NodeIterator 알고리즘의 복잡도는 $\Theta(\sum_{v \in V} d(v) C_2)$ 이며 이는 점근적으로 $O(n\ d_{max}(G)^2)$ 이다[7].

[8]에서는 *NodeIterator* 방법과 행렬의 곱셈 방법을 결합한 알고리즘을 제안하였다. 이 알고리즘은 정점 집합 V를 낮은 차수를 가지는 정점 집합 $V_{low} = \{v \in V | d(v) \leq \beta\}$와 높은 차수를 가지는 정점 집합 $V_{high} = V - V_{low}$ 나누었다. 이 때 $\beta = m^{(2.367-1)/(2.376+1)}$이다. 그리고 V_{low} 에 대해서는 *NodeIterator* 를 적용한다. 각 $v \in V_{low}$ 에 대하여 v에 인접한 $u, w \in V$ 가 삼각형을 형성할 때, u와 w가 모두 V_{low} 에 속하면 $\delta(v), \delta(u), \delta(w)$ 에 각각 1/3 을 더하고 u와 w가 모두 V_{high} 에 속하면 $\delta(v), \delta(u), \delta(w)$ 에 각각 1을 더하며, u와 w 중 하나만 V_{low} 에 속하면 $\delta(v), \delta(u), \delta(w)$ 에 각각 1/2를 더한다. 그리고 정점 축소 부분 그래프 $G[V_{high}]$ 에 대하여 행렬의 곱셈 방법을 적용하여 V_{high} 에 속한 정점들로만 이루어진 삼각형의 개수를 구한다. [8]에서 제안한 삼각 개수 알고리즘의 복잡도는 $O(m^{1.41})$이다.

EdgeIterator 알고리즘은 각 간선에 대하여 해당 간선을 포함하는 삼각형 구하는 방법이다. 간선 $\langle u, w \rangle \in E$ 에 대하여 $\{v, u, w\}$가 삼각형을 구성하려면 정점 $v \in V$는 $N(u)$와 $N(w)$에 공통으로 존재해야 한다. 따라서, 모든 간선 $e = \langle u, w \rangle \in E$ 에 대하여 $|N(u) \cap N(w)|$ 을 구하면 삼각형의 개수를 구할 수 있다. N(u)와 N(w)가 정렬된 배열로 구성되어 있다면 *EdgeIterator* 의 시간 복잡도는 $\sum_{\langle u,w \rangle \in E} (\delta(u)+\delta(w))$이며 이는 점근적으로 $O(m\ d_{max}(G)) \subset O(nm)$ 이다. *EdgeIterator*의 성능을 향상하기 위하여, $|N(u) \cap N(w)|$를 구하는데 해쉬을 사용한 *EdgeIterator-Hashed* 기법, u와 w 의 모든 인접 노드들을 비교하는 대신 부분 집합만을 비교하는 *EdgeIterator-forward* 기법 등이 제안되었다.

4.2 빅 그래프 데이터를 위한 삼각형 처리 기법

4.1절에서 전통적인 삼각형 처리 기법들을 살펴보았다. 이러한 전통적인 기법들은 인-메모리(in-memory) 알고리즘들로써 SNS 등과 같은 빅 그래프 데이터에는 적절하지 못하다. 따라서 최근 빅 그래프 데이터를 대상으로 하는 다양한 삼각형 처리 기법들이 제안 되었다.

빅 그래프 데이터를 크기가 매우 큼으로 근사적으로 삼각형 개수를 파악하고자 하는 연구들이 있었다. [9]에서는 그림 5와 같은 3-pass 알고리즘을 제안하였다.

T_i를 i개의 간선을 지니는 node-triple의 집합이라고 하면 $|T_1| + 2|T_2| + 3|T_3| = |E|(|V|-2)$ 와 같다. 3-pass알고리즘의 반환값 β의 예측값은 $E(\beta) = 3|T_3| / (|T_1| + 2|T_2| + 3|T_3|)$이다. 따라서, 구하고자 하는 삼각형의 개수 $|T_3|$ 는 $E(\beta)|E|(|V|-2)/3$이다.

Algorithm 3-pass($G = (V,E)$)
pass 1: count # of edges
pass 2: Sample an edge $e = \langle v, u \rangle$ uniformly and choose node w from $V-\{v,u\}$ randomly
pass 3: if $\langle v,w \rangle \in E$ and $\langle u,w \rangle \in E$ then $\beta = 1$ else $\beta = 0$
return β

(그림 5) 3-pass 알고리즘

따라서 3-pass알고리즘을 s번 시행한다면 $|T_3|$의 예측값, $|T_3^{\wedge}| \sim (1/s\sum_{i=i}^{s}\beta)\ |E|(|V|-2)/3$ 이다.

EdgeIterator 알고리즘에서 $|N(u) \cap N(w)|$ 을 구하는데 이는 Jaccard coefficient J(A,B) = |A∩B|/|A∪B|를 구하는 것과 유사하다. 따라서 [3]에서는 Jaccard coefficient를 근사적으로 구하는 Min-Hash를 활용하여 근사적으로 삼각형의 개수를 구하였다.

[10]에서는 간선 e를 p의 확률로 랜덤 샘플링하고 샘플링된 간선들의 집합 E'만으로 구성된 간선 축소 부분 그래프 $G[E']$ 에 대하여 삼각형 개수를 구하여 전체 그래프의 삼각형 개수를 구하는 *Doulion* 알고리즘을 제안하였다. 그림 6은 *NodeIterator*를 사용한 *Doublion* 알고리즘이다.

그림 6의 *Doulion-NodeIterator* 에서 하나의 간선이 G'에 추가될 확률은 p이다. 따라서, 하나의 삼각형이 G'에 유지될 확률은 $1/p^3$과 같다. 이에 따라서, 간선 축소 부분 그래프 G'의 삼각형의 개수에 $1/p^3$을 곱하면 전체 삼각형의 개수를 추정할 수 있다.

[5]에서는 삼각형의 개수를 근사적으로 구하기 위하여 행렬의 고유값(eigen value)을 활용하였다. 4.1절에서 언급한 바와 같이, 그래프 G의 인접행렬 A에 대하여 A^3의 대각원소들의 합 ($\sum_{i=1\ldots n} A^3[i][i]$) 은 $2\sum_{v\in V}\delta(v) = 6\Delta(G)$ 이다. 이 때, A는 대칭 행렬 (symmetric matrix)로써 고유값 분해 (eigen decomposition) 을 통하여 $A = U\Lambda U^T$로 분해가 가능하다. Λ 는 대각 행렬 (diagonal matrix)로 고유값 ($\lambda_1,\ldots,\lambda_n$) 을 가지고 있으며 U는 고유 벡터(eigen vector) $u_1,\ldots,u_n$들이 열 (column)으로 이루어진 직교행렬 (orthogonal matrix)이다 (즉, $U^TU = I$). 따라서, $A^3 = U\Lambda U^TU\Lambda U^TU\Lambda U^T = U\Lambda^2U^TU\Lambda U^T = U\Lambda^3U^T$ 와 같다.

이에 따라서, $\delta(v_i) = (\sum_j \lambda^3_j u^2_{ij})/2$ 이다. 여기서 u_{ij}는 열벡터 u_i의 j번째 값이다. 또한, $\Delta(G)$는 $1/6\ \sum_j\lambda^3_j$ 와 같다. [5]에서는 실제 그래프들을 고유값 분해를 해 본 결과 몇 개의 고유값들만 큰 절대값을 가지고 대부분은 매우 작은 절대값을 가짐을 확인하였다. 따라서, 큰 절대값을 가지는 고유값들만을 빠르게 추출하여 그래프의 삼각형 개수를 계산하는 방법을 제안하였다.

근사적으로 삼각형의 개수를 파악하는 기법 이외에 빅 그래프 데이터의 정확한 삼각형 개수를 파악하기 위하여 MapReduce[11] 및 Hadoop [12]과 같은 분산 병렬 프레임워크를 활용하는 기법들이 제안되었다.

[13]에서는 *NodeIterator*를 기반으로 두 단계 MapReduce 알고리즘을 제안하였다. 중복적으로 삼각형이 생성되는 방지하기 위하여 첫 번째 맵 리듀스 단계에서 맵 함수는 간선 $\langle v, u\rangle$ 에 대하여 차수가 작은 정점을 키로 하여 간선 (즉, $[min(d(v), d(u)), \langle v,u\rangle]$) 을 출력한다. 이 후 리듀스 함수는 동일한 키 u 를 가지는 간선들을 결합하여 모든 node-triple $\{v, u, w\}$들을 생성하고 키

Algorithm Doulion($G = (V,E)$)
for each $e \in E$ **do**
    put e to G' with probability p
$\Delta'(G) = NodeIterator(G')$
return $1/p^3\ \Delta'(G)$

(그림 6) *Doulion-NodeIterator* 알고리즘

로 바깥쪽에 붙은 두 정점에 대하여 차수가 높은 순으로하여 $\{v, w\}$을 출력한다 (즉, [$\{v,w\}$, $\{v,u,w\}$], 여기서 $d(v) > d(w)$). 이 후에는 간선 $<v,w>$가 존재하는지 만을 파악하면 된다. 따라서, 두 번째 단계의 맵 함수에서는 각 간선을 출력하는데 키를 정점을 차수가 높은 순으로 출력한다 (즉, [$(max(d(v), d(u)), mix(d(v), d(u))), <v,u>$]). 따라서, 이전 맵리듀스 단계의 결과 [$\{v,w\}$, $\{v,u,w\}$]와 이번 맵 함수의 결과를 합쳐서 두 번째 리듀스 함수에서는 동일 한 키 $\{v,w\}$ 에 대하여 $<v,w>$ 간선이 존재하는지 파악하여 $\{v,u,w\}$가 삼각형을 형성하는지 파악할 수 있게 된다.

[14]와 [15]에서는 그래프 분할을 이용한 알고리즘들을 제안하였다. [14]에서는 그래프 G의 정점집합 V를 p개의 부분집합들 $V_1, V_2, ..., V_p$로 나눈다. p개의 부분 집합들 중에서 임의의 3개의 부분집합들을 합한 집합을 V_{ijk} $(=V_i \cup V_j \cup V_k)$ $(i<j<k \leq p)$라 할 때, 각 V_{ijk}에 대한 정점축소 부분 그래프 $G[V_{ijk}]$에 대하여 독립적으로 삼각형의 개수를 계산한다. 이때, 하나의 삼각형이 중복되어 계산되어질 수 있다. 즉, 삼각형 $\{v,u,w\}$에 대하여 정점 v,u,w 가 V_i에 속한다면 ${}_iC_2 + i(p-i-1) + {}_{p-i-1}C_2$ 번 중복 계산되며 $\{v,u,w\}$가 V_i와 V_j에 나누어 속한다면 $p-2$번 중복 계산된다. 이에 따라서, [15]에서는 삼각형의 개수가 중복 계산되는 문제를 해결한 보다 효율적인 MapReduce 알고리즘을 제안하였다. [15]에서는 우선 p개의 부분 집합 $V_1, V_2, ..., V_p$에 대하여 두개의 부분 집합들의 합집합 V_{ij}에 $(i<j \leq p)$ 대하여 삼각형 개수를 구한다. 이 경우 삼각형 $\{v,u,w\}$의 정점들이 모두 같은 부분집합 V_i (또는 V_j) 에 있다면 $p-1$번만 중복 계산된다. 그렇지 않다면 한번만 계산되어진다. 이후 남아있는 것은 삼각형을 구성하는 정점이 서로 다른 부분집합에 포함되어 있는 경우임으로 $G[V_{ijk}]=(V_{ijk}, E')$에서 $\{e =<v,u> \in E' | v \in V_i, u \in V_j, V_i \neq V_j\}$ 로만 구성된 부분 그래프에 대하여 삼각형의 개수를 구하여 정확한 삼각형의 개수를 효율적으로 구한다.

5. 결 론

앞에서 언급한 바와 같이 그래프는 소셜 네트워크의 등장, 바이오 정보기술의 발전 등에 따라서 그래프는 다양한 응용분야에서 활용되고 있다. 이에 따라서, 그래프 마이닝 등 다양한 기술들이 개발 발전되고 있다. 특히, 소셜 네트워크에서 삼각형은 동류성 (homophily)과 이행성 (transitivity)을 설명하는 주요한 구성요소이다.

이에 따라서, 본 연구에서는 이러한 특성을 설명하기 위하여 그래프 상의 삼각 개수를 효율적으로 구하는 다양한 알고리즘 기법들을 조사하였다. *NodeIterator, EdgeIterator* 등과 같은 전통적인 기법들의 인-메모리 알고리즘으로 그래프가 메인 메모리에 적재될 수 있을 경우에만 적용이 가능하다. 그러나 최근에 활용되는 그래프는 정점 및 간선들의 개수가 수억 개에서 수십억 개에 달할 만큼 매우 큼으로 전통적인 알고리즘을 사용하기 어렵다. 이에 따라서, 샘플링 기법에 기반한 근사 알고리즘 및 MapReduce기반의 병렬처리 알고리즘들이 최근에 제안되고 있다. 이러한 알고리즘들은 빅 그래프 데이터에 대한 효율적인 처리 방안 및 그래프 데이터의 활용에 기본 기술을 제공함으로써 국내 그래프 데이터 처리 개발에 도움이 되리라 기대한다.

참 고 문 헌

[1] U. Kang, B. Meeder, C. Faloutsos, "Spectral Analysis for Billion-Scale Graphs: Discoveries and Implementation", In

Proceedings of PAKDD, pp. 13-25, 2011.
[2] D. J. Watts and Steven Strogatz, "Collective dynamics of 'small-world' networks", Nature 393 (6684), pp. 440–442, 1998.
[3] L. Becchetti, P. Boldi, C. Castillo. A. Gionis, "Efficient Semi-streaming algorithms for local triangle couting in massive graphs", In Proceddings of ACM KDD, pp. 16-24, 2008
[4] J. P. Eckmann, E. Moses, "Curvature of co-links uncovers hidden thematic layers in the world wide web", In Proceedings of the national academy of sciences, 99(9), pp. 5825-5829, 2002.
[5] C. E. Tsourakakis, P. Drineas, E. Michelakis, I. Koutis, C. Faloutsos, "Spectral counting of triangles via element-wise sparsification and triangle-based link recommendation", Social Network Analysis and Mining 1(2), pp. 75-81, 2011.
[6] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions", In Proceedings of ACM conference on Theory of computing, pp. 1-6, 1987.
[7] T. Schank, D. Wagner, "Finding, counting and listing all triangles in large graphs, an experimental study" In Experimental and Efficient Algorithms (LNCS), pp. 606-609, 2005.
[8] N. Alon, Y. Raphael, and U. Zwick. "Finding and counting given length cycles", Algorithmica, 17(3), pp. 209-223, 1997
[9] L. S. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccalema, C. Sohler, "Counting triangles in data streams", In Proceedings of ACM PODS, pp. 253- 262, 2006.
[10] C. E. Tsourakakis, U. Kang, G. L. Miller, C. Faloutsos, "Doulion: counting triangles in massive graphs with a coin", In Proceedings of ACM SIGKDD, pp. 837-846, 2009.
[11] J. Dean, S. Ghemawat, "Mapreduce: simplified data processing on large clusters", CACM, 51(1), pp. 107-113, 2008.
[12] T. White, "Hadoop: the definitive guide", O'Reilly Media, Inc., 2012.
[13] J. Cohen, "Graph twiddling in a mapreduce world," Computing in Science and Engineering, 11(4), pp. 29-41, 2009.
[14] S. Suri, S. Vasslivitskii, "Counting triangles and the curse of the last reducer", In Proceedings of WWW, pp. 607-614, 2011.
[15] H-M. Park, C-W. Chung, "An efficient mapreduce Algorithm for counting triangles in a very large graph", In Proceedings of ACM CIKM, pp.539-548, 2013.

저 자 약 력

민 준 기

이메일 : jkmin@koreatech.ac.kr

- 1995년 숭실대학교 전자계산학과(학사)
- 1997년 한국과학기술원 전산학과(석사)
- 2002년 한국과학기술원 전산학과(박사)
- 2002년~2004년 한국과학술원 연수연구원/초빙교수
- 2004년~2005년 한국전자통신연구원 선임연구원
- 2005년~현재 한국기술교육대학교 컴퓨터공학부 교수
- 관심분야 : 데이터베이스, 센서 데이터, 맵리듀스, 질의 최적화