ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

대규모 네트워크에서 Modularity를 이용한 향상된 커뮤니티 추출 알고리즘

An Enhanced Community Detection Algorithm Using Modularity in Large Networks

한 치 근* 조 무 형**
Chi-Geun Han Moo-Hyoung Jo

요 약

본 논문에서는 modularity를 기반으로 한 향상된 커뮤니티 추출 알고리즘을 제안한다. 기존의 알고리즘은 modularity 값을 증가시키는 커뮤니티를 구축할 때 노드가 갖고 있는 정보를 고려하지 않음으로써, 계산을 비효율적으로 반복하여 수행한다. 제안하는 알고리즘은 노드의 degree(weight)를 계산하고 그것을 내림차순으로 정렬하고, 정렬된 순서대로 modularity 값의 증가여부를 확인함으로써, 반복되는 계산과정을 줄여 기존의 알고리즘보다 빠르게 최종 결과를 도출해낸다. 실험계산을 통해 제안하는 알고리즘이 더 짧은 시간 내에, 기존알고리즘이 구한 modularity 값보다 같거나, 향상된 값을 찾는다는 것을 보인다.

ABSTRACT

In this paper, an improved community detection algorithm based on the modularity is proposed. The existing algorithm does not consider the information that the nodes have in checking the possible modularity increase, hence the computation may be inefficient. The proposed algorithm computes the node degree (weight) and sorts them in non-increasing order. By checking the possible modularity value increase for the nodes in the nonincreasing order of node weights, the algorithm finds the final solution more quickly than the existing algorithm does. Through the computational experiments, it is shown that the proposed algorithm finds a modularity as good as the existing algorithm obtains.

☞ keyword : 소셜 네트워크, modularity, 커뮤니티 추출, 그래프 밀도

1. 서 론

소셜 네트워크 서비스는 대규모의 사용자(수백만~수천만)를 갖고 있으며, 사용자들의 관계는 follower (Twitter), 친구(Me2day, Facebook) 등으로 정의된다.

'친구'와 같은 단순한 관계로만 연결되는 대규모의 소셜 네트워크 내에서는 커뮤니티를 확인, 추출(community detection)하는 것이 용이한 일이 아니다. 수학적으로는 partition 문제로 정의되고, 이는 NP-hard의 계산 복잡도를 갖는다. 그렇기 때문에 그동안 소셜 네트워크 분석(SNA: Social Network Analysis) 방법을 이용하여 네트워크 내에 존재하는 커뮤니티의 구성을 확인하는 많은 연구가 진행되어 왔다[1].

본 논문에서는 소셜 네트워크 내에 존재하는 커뮤니티를 추출해내는 기존의 여러 방법들 중, Louvain method [2]을 개선시키는 연구를 수행하였다. 그리고, 동일한 네트워크 내에서 커뮤니티를 추출하였을 때, 같거나 향상된 결과 보다 빠르게 도출해내는 연구에 대해 기술한다.

본 논문은 다음과 같이 구성되어 있다. 저 이전에 연구된 소셜 네트워크 내의 커뮤니티 추출 방법들에 대해 간략히 살펴보며, 3장에서는 본 논문에서 제안하는 방법에 대해 설명하고, 제안하는 방법과 기존 방법과의 성능 차이를 확인하기 위한 실험계산결과를 보인다. 마지막으로 4장에서는 결론을 기술한다.

2. 관련 연구

1970년대에 진행되었던 연구 중, 소셜 네트워크에서 네트워크의 중앙(centrality, 중심)을 정의하기 위한 measure

* 정 회 원 : 경희대학교 컴퓨터공학과 교수
cghan@khu.ac.kr
** 정 회 원 : 경희대학교 일반대학원 컴퓨터공학과 재학중
(박사과정) for2cho@khu.ac.kr
[2011/12/30 투고 - 2012/01/05 심사(2012/05/02 2차) - 2012/06/01 심사완료]

http://dx.doi.org/10.7472/jksii.2012.13.3.75

를 제안한 연구[3]가 있다.

2000년대에 접어들면서 노드들 간의 밀접도(betweenness)를 이용하여 에지를 순차적으로 제거하는 방법을 통해 커뮤니티를 추출하는 방법[4]도 연구되었으며, 네트워크의 modularity(여기서 modularity는 네트워크 내에서 구분된 그룹(division, partition) 내에서는 많은 연결이 있고, 그 그룹 간에는 적은 수의 연결이 있는 성질을 나타내는 하나의 척도이다)를 이용하여, 그 값을 향상시키는 노드들의 결합여부를 결정하고 modularity를 향상시킬 수 있는 결합이 있는 한 계속해서 반복 수행하여 커뮤니티를 추출하는 방법[5]도 연구되었다.

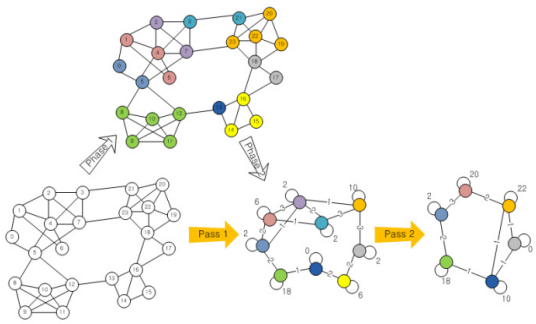
2000년대 후반에는 보다 더 활발하게 연구가 진행되었는데, 유향그래프에서 커뮤니티를 찾는 방법[6], 커뮤니티 내의 이메일 등의 정보 교류량을 측정하여, 그 변화가 있는지를 확인하는 통계에 기반을 둔 방법[7], 그래프의 계층적, 다차원적인 구조를 확인하기 Potts라는 모델을 제안한 연구[8], 앞서 언급한 [5]의 단점을 보완하여 마찬가지로 modularity 값을 이용해 노드들을 결합하고 분해하며 최고의 modularity 값을 갖는 커뮤니티를 추출하는 방법[2], modularity를 사용한 방법들과는 반대로 modularity를 사용하는 방법이 일정 규모 이하의 문제인 경우 커뮤니티를 찾지 못할 수도 있다는 것을 밝힌 연구[9]도 진행되었다. 또한, 다양한 네트워크 내에서 커뮤니티를 찾는 알고리즘들을 검증하기 위한 샘플 그래프를 생성해주는 연구[10]도 존재하며, 구현된 커뮤니티 추출 알고리즘들을 서로 비교분석한 연구[11]도 진행되었다.

가장 최근에는 Generative network 모델에서 principled 통계적 방법을 사용하는 커뮤니티 추출 방법[12], 공통 이웃을 가진 노드들의 그래프 밀도를 계산하여 유사도를 정의하고, 유사도 전파(affinity propagation) 알고리즘과 접목시켜 소셜 네트워크의 커뮤니티를 추출하는 방법[13] 등이 연구되었다. 또한, 커뮤니티 추출 방법에 대해 문제를 정의하고, 2010년 이전의 커뮤니티 추출 알고리즘들의 비교분석 등을 정리하여 방대한 내용의 논문으로 구성한 연구[1]도 존재한다.

아래에서는 본 논문에서 제안하는 방법의 근간이 되는 Modularity 기반의 Blondel et al. 방법(2008, 이하 Louvain method)[2]에 대해 자세히 설명한다.

2.1 Louvain method

Louvain method는 2개의 phase로 나뉘어 반복적으로 알고리즘을 수행한다. N개의 노드를 갖는 네트워크에서



(그림 1) Louvain method의 phase와 pass를 나타낸 그림.

계산을 수행한다고 가정했을 때, 우선, 각각의 노드에 서로 다른 커뮤니티를 할당하여 노드 개수만큼의 커뮤니티 번호를 부여한다. 그리고 각각의 노드 i에 대해 노드 i의 이웃노드인 j와 커뮤니티를 이루도록 노드 i가 속해있던 커뮤니티에서 노드 i를 빼내어 노드 j가 속한 커뮤니티에 속하도록 삽입하여 modularity 값을 구한다. 만약, 여기서 구한 modularity 값이 이전에 구한 modularity 값보다 크다면, 현재의 커뮤니티를 유지하고 다음 계산을 진행하며, 이전에 구한 modularity값보다 작다면, 이전의 커뮤니티 구성 상태로 되돌아가 다음의 이웃 노드에 대해 연산을 수행한다. 이러한 계산은 노드 i의 모든 이웃노드에 대해 순차적으로 수행되고 다른 모든 노드에 대해서 반복적으로 수행된다. 이 과정은 modularity값이 더 이상 증가하지 않을 때까지 계속되며, 노드의 커뮤니티간 이동으로 modularity값이 더 이상 증가하지 않을 때, 첫 번째 phase가 종료된다.

다음 phase는 이전 단계에서 커뮤니티로 묶인 노드들의 집합을 하나의 노드로 만드는 과정인데, 커뮤니티 내부 노드들의 연결은 self-loop로 바뀌어 연결의 개수만큼의 가중치가 있는 self-loop를 갖게 되고, 커뮤니티 사이의 연결은 각각의 커뮤니티에 속한 노드들의 연결을 모두 더한 만큼의 가중치를 갖는 하나의 연결로 바뀌게 된다. 2번째 phase를 거치고 나서 생성된 네트워크는 또다시 1번째 phase를 적용할 수 있는 형태가 되고 앞서 설명한 두개의 phase를 합쳐 하나의 pass로 칭하며 반복적으로 계산을 수행하게 된다. 이러한 구조로 인해 커뮤니티의 개수는 각각의 pass가 반복되며 감소하게 되고, 계산 시간의 대부분을 첫 번째 pass의 계산에 사용한다. (그림 1)에서 연산과정을 간단히 나타내었다.

알고리즘의 수행은 modularity의 값이 더 이상 증가하

지 않을 때 정지하고, 그 때까지 결합된 노드들의 커뮤니티를 추출하여, 소셜 네트워크의 커뮤니티를 구성한다.

$$Q = \sum_i (e_{ij} - a_i^2). \quad \text{(식 1)}$$

이 방법에서 사용되는 modularity의 값은 Q로 표현[5]하며, 식 1과 같다. 여기서 e_{ij}는 커뮤니티 i와 커뮤니티 j를 연결하는 에지들의 비율을 나타내며, a_i는 i에 연결되어 있는 에지들의 비율을 나타낸다. 식 1의 Q를 이용하여 노드 i를 커뮤니티 C에 넣을 때의 modularity 변화량을 수식으로 표현하면 다음과 같다.

$$\Delta Q = \left[\frac{\sum_{in} + k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right]. \quad \text{(식 2)}$$

위 식에서 $\sum_{in}$(이하 in)은 C내부에 연결된 weight의 총합이고, $\sum_{tot}$(이하 tot)은 커뮤니티 C에 인접하는 에지들의 weight들의 총합이다. k_i는 노드 i에 연결된 에지들의 weight의 합이고, $k_{i,in}$은 커뮤니티 C내에 있는 노드 i에 연결된 에지들의 weight합이다. m은 네트워크에 존재하는 에지들의 weight의 총합이다.

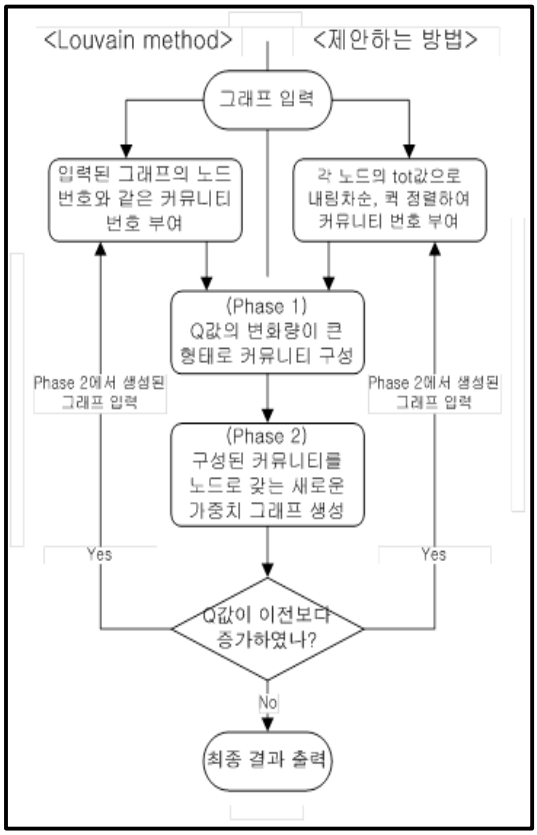
식 2와는 반대로 노드 i를 커뮤니티 C로부터 제거하였을 때 modularity변화량은 다음과 같다.

$$\Delta Q = \left[\frac{\sum_{in} - k_{i,in}}{2m} - \left(\frac{\sum_{tot} - k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 + \left(\frac{k_i}{2m} \right)^2 \right]. \quad \text{(식 3)}$$

위의 식 2와 식 3를 이용하여 ΔQ를 증가시킬 수 있는 노드의 커뮤니티간 이동을 확인하여 modularity의 값을 pass마다 서서히 증가시키는 방법이 바로 Louvain method이다.

3. 본 론

본 장에서는 앞서 2.1에서 기술한 Louvain method를 발전시켜 보다 빠른 시간 내에 유사한 결과를 얻을 수 있도록 개선된 방법을 제안한다. 또한, 실험계산을 통해 Louvain method와 제안하는 방법의 성능을 비교분석한다.

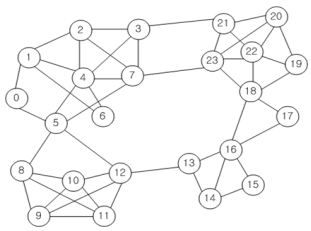


(그림 2) Louvain method와 제안하는 방법의 알고리즘 비교.

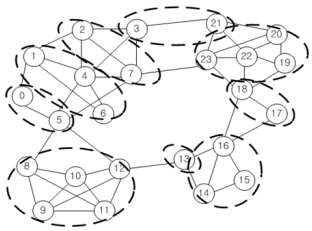
3.1 개선된 방법

Louvain method의 phase 1에서는 커뮤니티에서 빼거나 삽입할 노드를 임의로, 혹은 첫 번째 노드부터 순차적으로 정하여 고려한다. 본 논문에서는 Louvain method의 노드를 고려하는 순서를 바꿔 계산 시간을 단축할 수 있도록 하는 방법을 제안한다.

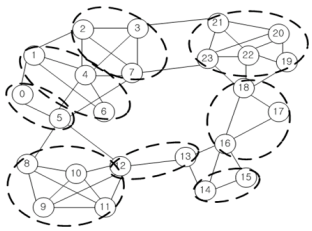
제안하는 방법은 (그림 2)의 알고리즘에서도 볼 수 있듯이 phase 1에 진입하여 연산을 수행하기 이전에 노드들 각각의 tot값으로 퀵 정렬을 이용하여 내림차순(non-increasing)으로 정렬한다. 이러한 정렬 과정을 거친 이후, 커뮤니티를 형성하는데 구심점이 될 수 있는 높은 weight



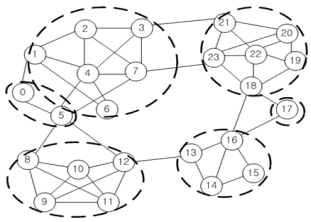
(그림 3) 그래프의 초기 상태.



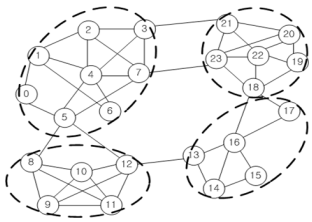
(그림 4) 기존 방법, pass 1 후의 모습.



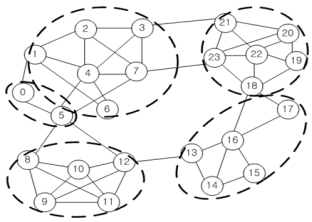
(그림 5) 제안하는 방법, pass 1 후의 모습.



(그림 6) 기존 방법, pass 2 후의 모습.



(그림 7) 제안하는 방법, pass 2 후의 모습.



(그림 8) 기존 방법, pass 3 후의 모습.

값들의 합(비가중치 그래프의 pass 1에서는 각 노드들의 degree를 의미하며, 이후 커뮤니티를 형성하여 커뮤니티를 노드로 간주하는 시점에서부터 각 커뮤니티 간에 연결된 에지들의 합을 weight로 칭함)을 갖는 노드부터 고려하여 커뮤니티를 구성함으로써 tot값은 낮아지며(외부와의 연결은 줄어들며), in값은 커지게(내부의 연결은 많아짐) 된다. 결과적으로 phase가 진행될수록 기존의 방법보다 빠르게 커뮤니티의 크기가 커지게 되고, 연산의 수행 횟수가 줄며, 횟수가 줄어들게 된 만큼 계산 시간이 단축된다.

반면, 노드를 정렬하여 정렬된 순서대로 연산을 수행하기 때문에 정렬을 하는 데 시간이 소요된다. 따라서, 정렬하는데 필요한 시간을 최대한 단축시키기 위하여 $O(nlog_2n)$의 복잡도를 갖는 퀵 정렬(Quick sort)을 사용하여 노드의 정렬에 요구되는 시간을 최대한 줄인다.

제안하는 방법의 알고리즘의 수행 순서는 앞선 2.1절의 기존 방법과 같으나, 알고리즘의 수행 이전에 노드를 tot값의 내림차순으로 정렬하여, 정렬된 순서대로 커뮤니티 번호를 부여해 노드의 고려순서를 바꿔주는 작업을 수행하는 차이점이 있다.

3.2 개선된 방법의 수행 예

(그림 3)의 그래프를 이용하여 기존의 방법과 제안하는 방법을 적용 시, 연산 과정의 차이점을 (그림 4)~(그림 8)의 예시를 통해 알 수 있다. (그림 4)~(그림 8)에서 실선은 노드 사이의 관계를 나타내며, 굵은 점선은 pass이후에 새롭게 형성된 커뮤니티를 나타낸다.

기존의 방법은 노드 번호 0번부터 순차적으로 고려하여 커뮤니티를 생성하게 되고, 제안하는 방법은 노드의 외부와 연결된 에지의 수(weight)가 많은 순서대로 고려하여 4번 노드부터 알고리즘을 수행한 결과를 볼 수 있다. tot값이 같을 경우, 노드 번호가 앞서는 노드를 우선적으로 고려한다. (그림 4)과 (그림 5)에서 볼 수 있듯이 노드를 고려하는 순서가 달라짐에 따라 pass 1이 지난 후의 결과가 약간 다르며, 다음 pass에서 기존의 방법은 9개의 노드, 제안하는 방법은 8개의 노드로 알고리즘을 수행하게 된다.

Pass 2가 지난 후의 결과를 (그림 6)와 (그림 7)에서 볼 수 있다. (그림 7)에서 볼 수 있는 4개의 커뮤니티는 두 번의 pass를 더 거쳐 modularity 값이 증분임계치(0.000001)보다 증가하지 않음을 확인하고 최종적으로 도출된 결과와 일치한다.

기존의 방법은 pass 4를 거쳐도 (그림 8)의 상태를 유지하며, pass 5가 지나서야 0번과 5번 노드의 커뮤니티가 4번 노드가 속한 커뮤니티에 포함되며 modularity 값이 증가하지 않음을 확인하고 알고리즘의 수행을 종료하게 된다.

최종적으로 도출된 결과는 기존의 방법과 차이가 없음에도 불구하고 제안하는 방법이 알고리즘을 수행하는 과정에서 노드를 고려하는 순서를 바꿔줌으로써 기존의 방법보다 계산 과정을 단축시켜 빠른 시간 내에 효과적으로 커뮤니티의 추출이 가능함을 알 수 있다.

3.3 개선된 방법의 Pseudo-code

제안하는 방법의 알고리즘을 아래에 Pseudo -code로 나타내었다.

```
input graph data;
compute newQ;
do while{
for each node, compute tot value;
Quick Sort(tot values);
 //tot(n_0) ≥ tot(n_1) ≥ .... ≥ tot(n_numofnodes-1)
 for(i=0;i<number of nodes;i++)
 {
   for(j=0;j<number of n_i's neighbor;j++)
   {
     n_i and neighbor n_i(=j) grouped;
     calculated newQ;
     if(newQ>beforeQ)
     {
       Q=newQ;
       keep current community group;
     }
     else
       go back previous community group;
   }
 }
 update community graph;
}(Q-beforeQ>threshold)
 //threshold is 0.000001.
show result;
```

3.4 실험 환경

실험에서 사용된 컴퓨터의 CPU는 Intel Core 2 Duo CPU E7500 2.93Ghz이며, 메모리는 2GB이다.

모든 실험에 사용된 그래프는 [10]에서 제안한 방법을 사용하여 생성하였으며, 같은 조건으로 생성한 그래프이더라도 생성할 때마다 당시의 time-seed 값에 따라 다양한 모양의 그래프가 생성될 수 있으므로 같은 조건으로 10개의 그래프를 생성하였다. 또한, 하나의 문제에 대해 알고리즘 수행 시점에 따라 계산 시간이 달라질 수 있으므로 같은 그래프에 대해 두 가지 방법으로 각 10번씩 풀이하였으며(이는 실험계산을 수행한 컴퓨터의 다른 프로세스들의 영향으로 인하여 계산하는데 걸리는 시간이 약간씩 차이가 나기 때문에 취한 방법임), 계산된 10개의 결과에 대한 평균값을 구해 최종적인 결과 데이터로 사용하였다.

실험은 크게 다섯 가지로 나눌 수 있다. 실험 1과 실험 2는 전체 네트워크의 크기가 커지는 상황을 가정하여 실험하였고, 실험 3과 실험 4는 네트워크의 크기가 일정한 상황에서 밀도가 증가하는 상황을 가정하여 실험하였다. 마지막으로 실험 5는 네트워크의 크기가 일정한 상황에서 대부분의 노드와 연결된 특정 노드가 존재하는 상황을 가정하여 실험하였다.

(표 1) 실험에 사용된 그래프의 속성

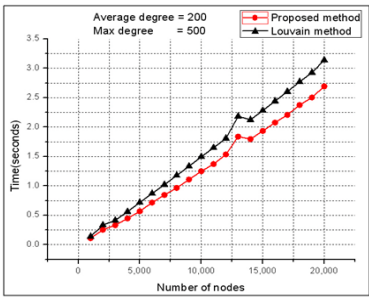
	노드개수	평균 degree	최대 degree	총 에지 개수
실험 1	1,000 ~20,000	200.1575	500	99,254 ~2,001,525
실험 2	10,000 ~100,000	199.7858	500	1,000,645 ~9,987,187
실험 3	10,000	49.5134 ~249.6321	500	247,567 ~1,248,161
실험 4	10,000	99.12606 ~498.9371	1,000	495,630 ~2,494,686
실험 5	10,000	198.5923	1,000 ~5,000	984,026 ~1,004,831

각 실험에 사용된 그래프의 속성을 (표 1)에 간단히 표현하였다. 실험 1과 2에 사용된 그래프는 평균 degree와 최대 degree를 고정시키고 실험 1은 노드의 개수를 1,000개에서 20,000개까지 1,000개 단위로, 실험 2는 노드의 개수를 10,000개에서 100,000개까지 10,000개 단위로 증가시키며 생성하였다. 또 실험 3과 4에서 사용된 그래프는 노드의 개수와 최대 degree를 고정시키고 실험 3은 평균 degree를 50개에서 250개까지 50개 단위로, 실험 4는 평균 degree를 100개에서 500개까지 100개 단위로 증가시키며 그래프를 생성하였다. 마지막으로 실험 5에서 사용된 그래프는 노드의 개수와 평균 degree를 고정시키고 최대 degree를 1,000개에서 5,000개까지 1,000개 단위로 증가시키며 생성하였다.

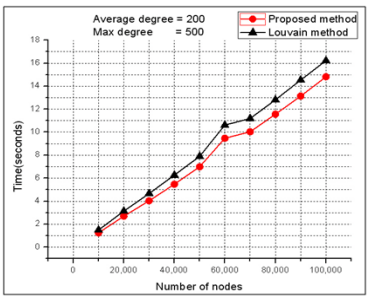
또한, (표 1)에서 볼 수 있는 실험 1과 실험 2에서 사용된 평균 degree의 값과 최대 degree의 값은 인터넷 사이트 (http://www.sysomos.com/insidetwitter/appendix/)를 참고하였으며, 소셜 네트워크 서비스인 Facebook을 통해 20대 남성과 여성 100여명을 대상으로 설문조사를 진행한 결과를 반영하여 얻어진 현실적으로 의미가 있다고 판단되는 값 200과 500을 기준으로 정하였다.

3.5 실험 결과 및 분석

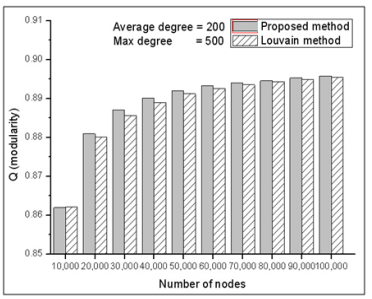
노드의 개수가 1,000개에서 20,000개까지 증가하는 동안 제안하는 방법이 기존의 방법보다 빠른 알고리즘 수행 속도를 보여주는 모습을 (그림 9)에서 볼 수 있으며,



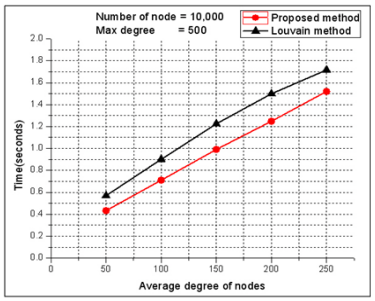
(그림 9) 실험 1의 결과. 계산 시간이 최대 0.5초가량 차이남.



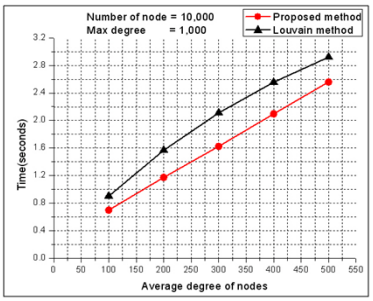
(그림 10) 실험 2의 결과. 계산 시간이 최대 1.3초가량 차이남.



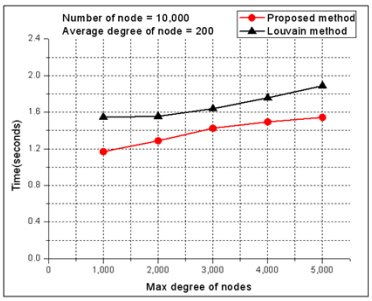
(그림 11) 실험 2에서 얻어진 Q값의 비교.



(그림 12) 실험 3의 결과. 계산 시간이 최대 0.1초가량 차이남.



(그림 13) 실험 4의 결과. 그림 12와 유사한 형태의 그래프가 생성됨.



(그림 14) 실험 5에 대한 결과. 계산 시간이 많게는 0.3초까지 차이남.

노드의 개수가 많아질수록 계산 시간의 차이도 점차 벌어지는 것을 확인할 수 있다.

다음으로, 노드의 개수를 10,000개에서 100,000개까지 증가시키며 알고리즘을 수행한 실험 2의 결과를 (그림 10)에서 볼 수 있는데, 실험 1과 마찬가지로 기존의 방법보다 좋은 성능을 보여주고 있다. 또한, 실험 1에서 확인한 바와 같이 노드의 개수가 증가할수록 기존의 방법보다 우수한 성능을 보여준다. 노드의 개수가 더 늘어날 경우 계산 시간은 기존의 방법보다 더 많이 감소할 것으로 기대된다.

다음으로 실험 2에서 최종적으로 얻어진 modularity의 값, 즉, Q값을 (그림 11)에서 기존의 방법과 비교해보았다. 기존의 방법과 제안하는 방법 모두 최종 Q값이 같을 것으로 예상했던 것과 다르게 노드의 수가 10,000개일 때엔 기존 방법의 Q값이 약 0.0003정도 제안하는 방법보다 높게 나왔고, 노드의 개수가 20,000개 이상부터는 제안하는 방법이 기존 방법보다 최대 0.0015이상 높은 Q값을 보여준다.

다음으로, 그래프의 밀도를 높여가며 실험한 실험 3과 실험 4의 계산 결과에 대한 그래프를 보면, 실험 1과 실험 2와 마찬가지로 제안하는 방법이 기존의 방법보다 계산 시간에 있어서 우수한 성능을 보여주는 것을 알 수 있다. 또한, (그림 12)와 (그림 13)에서 볼 수 있듯이 계산 시간이 linear하게 평균 degree의 증가에 비례하여 커지는데, 이는 각 노드에서 고려해야할 이웃노드의 수가 증가하기 때문이다.

마지막으로 실험 5에 대한 결과를 (그림 14)에서 볼 수 있다. 계산 시간이 (그림 12), (그림 13)보다는 완만하게 증가하는 것을 볼 수 있는데, 이것은 평균 degree가 최대 degree보다 계산 시간에 많은 영향을 끼치는 것을 의미한다.

실험 결과에 따라 모든 경우에 제안하는 방법이 기존 방법보다 최소 9.5%에서 최대 32.2%까지 계산 시간을 단축하였음을 알 수 있었다. Q의 값 또한 전체 문제 중 약 26%의 문제에서 기존 방법보다 0.02%~0.18% 가량 높은 결과를 얻었고, 약 71%의 문제에서 기존 방법과 같은 Q값을 얻었다.

4. 결 론

본 논문에서는 소셜 네트워크 내에서의 커뮤니티 추출을 위해 modularity를 사용한 기존의 방법을 개선한 방법을 제안하고 기존 방법과 성능을 비교, 분석하였다.

기존 방법이 처음 노드부터 순차적으로 연산을 반복하여 비효율적으로 수행하던 것과 달리, 제안하는 방법은 외부와의 연결이 많은 커뮤니티의 순서대로 내림차순 정렬을 수행하고, 정렬된 결과를 이용하여 외부와의 연결이 많은 커뮤니티에 대해 먼저 연산을 수행한다. 그로 인해 기존의 방법보다 빠르게 많은 외부와의 연결이 내부의 연결로 점차 바뀌며, 보다 많은 노드들을 연산 초기에 커뮤니티에 합류시켜 반복문의 수행 횟수를 줄여서 계산 시간을 단축시키는 방법이다. 제안하는 방법의 이러한 계산 시간의 단축은 네트워크 환경에서 요구되는 정보들이 실시간으로 제공되어야 하는 것을 감안했을 때 긍정적으로 볼 수 있다.

기존 방법과 제안하는 방법의 성능을 비교해본 결과 본 논문에서 제안하는 방법이 모든 경우에 기존 방법보다 최소 9.5%에서 최대 32.2%까지 시간을 단축시켰으며, 최종적으로 얻어진 Q의 값 또한 전체 문제 중 약 71%의 문제에서 기존 방법과 동일한 결과가 나왔으며, 약 26%의 문제에서 0.02%~0.18% 가량 기존 방법보다 높은 결과를 얻어낸 사실을 확인할 수 있었다. 향후 실제적으로 존재하는 1,000,000개 이상의 노드를 갖는 대형 네트워크에서 성능을 실험하여 제안하는 방법이 우수함을 입증할 계획이다.

참 고 문 헌

[1] Santo Fortunato, Community Detection in Graphs, Physics Reports 486, 75-174 (2010).

[2] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte and Etienne Lefebvre, Fast Unfolding of Communities in Large Networks, Journal of Statistical Mechanics Volume 2008 October (2008).

[3] Linton C. Freeman, Centrality in Social Networks Conceptual Clarification, Social Networks, 1, 215-239 (1978/79).

[4] M.E.J. Newman and M. Girvan, Finding and Evaluating Community Structure in Networks, Phys. Rev. E 69, 026113 (2004).

[5] Aaron Clauset, M. E. J. Newman, and Cristopher Moore, Finding Community Structure in Very Large Networks, Phys. Rev. E 70, 066111 (2004).

[6] E.A. Leicht and M.E.J. Newman, Community Structure in Directed Networks, Phys. Rev. Letter, 100(11), 118703, (2007).

[7] Ian A. McCulloh and Kathleen M. Carley, Social Network Change Detection, Institute for Software Research, Carnegie Mellon, Working Paper CMU-CS-08-116, (2008).

[8] Peter Ronhovde, Zohar Nussinov, Multireso-lution Community Detection for Megascale Networks by Information-Based Replica Correlation, Phys. Rev. E 77, 036122 (2009).

[9] Santo Fortunato, Marc Barthelemy, Resolution Limit in Community Detection, Proc. Natl. Acad. Sci. USA 104 (1), 36-41 (2007).

[10] A. Lancichinetti, S. Fortunato, and F. Radicchi, Benchmark Graphs for Testing Community Detection Algorithms, Phys. Rev. E 78(4), 046110 (2008).

[11] A. Lancichinetti, S. Fortunato, Community Detection Algorithms: A Comparative Analysis, Phy. Rev. E 80, 056117 (2009).

[12] Brian Ball, Brian Karrer, and M. E. J. Newman, An Efficient and Principled Method for Detecting Communities in Networks, Phys. Rev. E 84, 036103 (2011).

[13] 강윤섭, 최승진, 공통 이웃 그래프 밀도를 사용한 소셜 네트워크 분석, 정보과학회 논문지 : 컴퓨팅의 실제 및 레터, 제 16권 제 4호(2010).

◐ 저 자 소 개 ◑

한 치 근

1983년 서울대학교 산업공학과 졸업(학사)
1988년 펜실베니아주립대학교 대학원 computer science학과 졸업(석사)
1991년 펜실베니아주립대학교 대학원 computer science학과 졸업(박사)
1992년~현재 경희대학교 컴퓨터공학과 교수
관심분야 : 알고리즘, 유전자알고리즘
E-mail : cghan@khu.ac.kr

조 무 형

2010년 경희대학교 전자정보학부 컴퓨터공학과 졸업(학사)
2012년 경희대학교 일반대학원 컴퓨터공학과 졸업(석사)
2012년~현재 경희대학교 일반대학원 컴퓨터공학과 재학중(박사과정)
관심분야 : 알고리즘, 유전자알고리즘, 클러스터링
E-mail : for2cho@khu.ac.kr