그래프를 이용한 기계 학습

#5 그래프의 구조를 어떻게 분석할까?

신기정
(KAIST AI대학원)

boostcamp AI Tech

1. 군집 구조와 군집 탐색 문제

2. 군집 구조의 통계적 유의성과 군집성

3. 군집 탐색 알고리즘

4. 중첩이 있는 군집 탐색

5. 실습: Girvan-Newman 알고리즘 구현 및 적용

1. 군집 구조와 군집 탐색 문제

1.1 군집의 정의
1.2 실제 그래프에서의 군집들
1.3 군집 탐색 문제

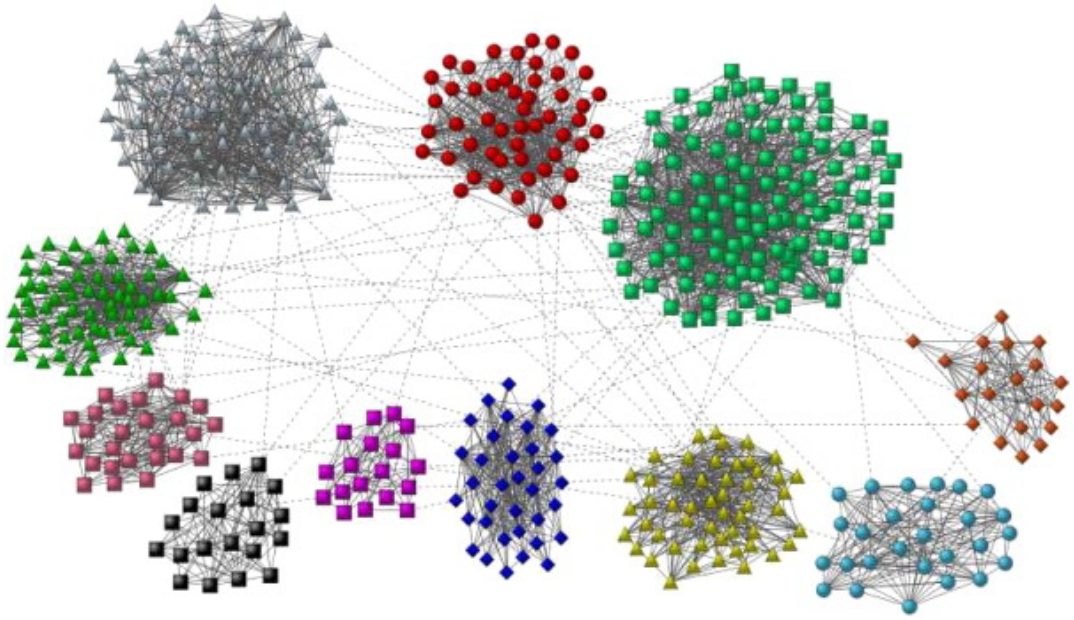
1.1 군집의 정의

군집(Community)이란 다음 조건들을 만족하는 정점들의 집합입니다

(1) 집합에 속하는 정점 사이에는 많은 간선이 존재합니다
(2) 집합에 속하는 정점과 그렇지 않은 정점 사이에는 적은 수의 간선이 존재합니다

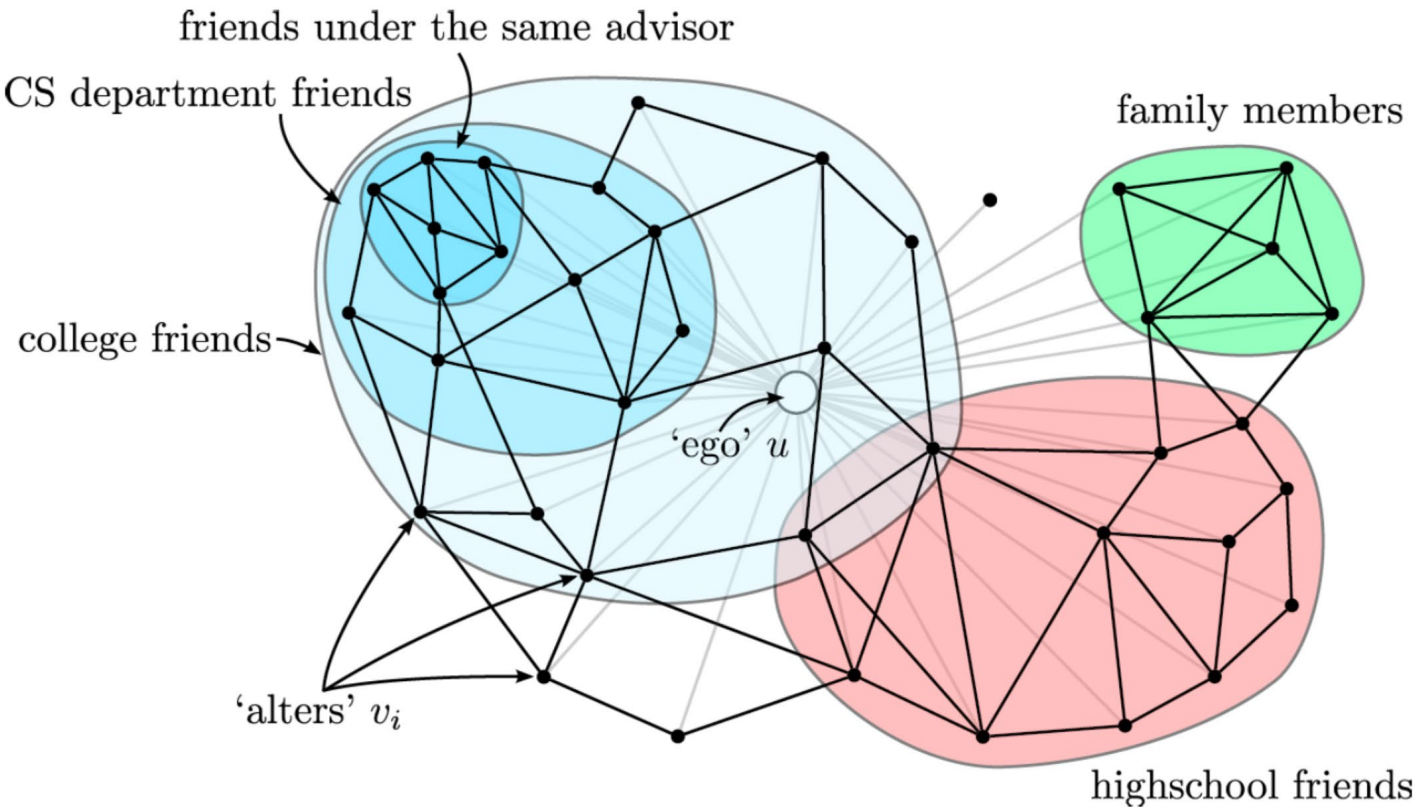
수학적으로 엄밀한 정의는 아닙니다

예시 그래프에는 11 개의 군집이 있는 것으로 보입니다



1.2 실제 그래프에서의 군집들

온라인 소셜 네트워크의 군집들은 사회적 무리(Social Circle)을 의미하는 경우가 많습니다



이미지 출처: https://arxiv.org/pdf/1210.8182.pdf

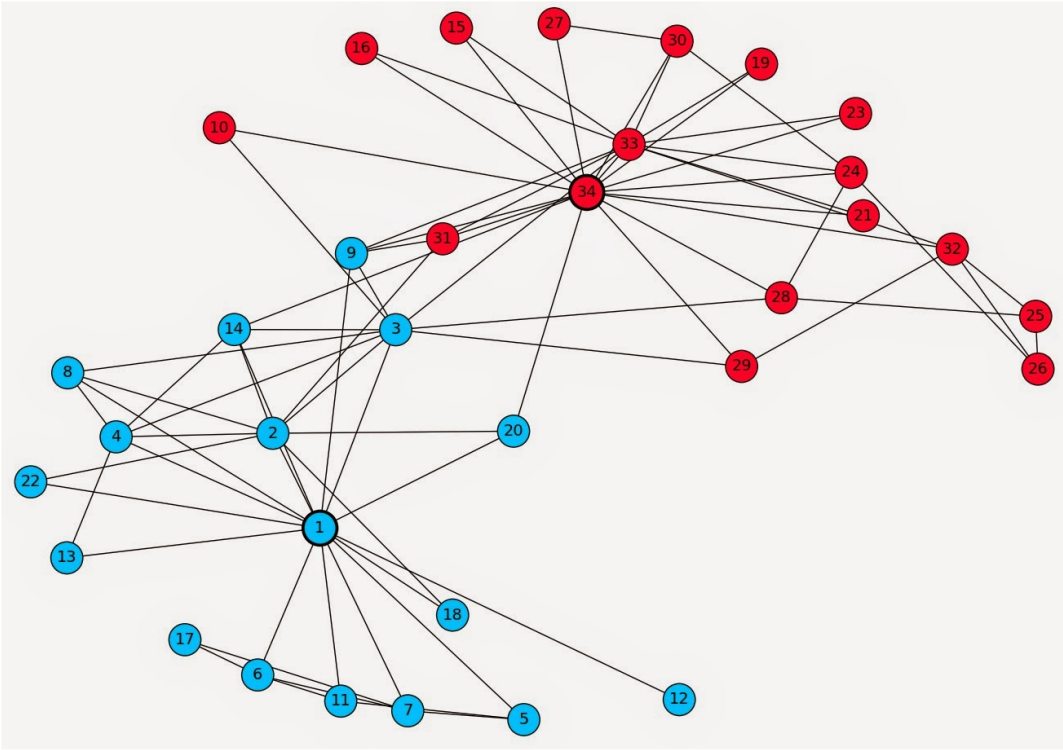
1.2 실제 그래프에서의 군집들

온라인 소셜 네트워크의 군집들이 부정 행위와 관련된 경우도 많습니다

왼쪽 부정 행위에 연루된 계정들이
군집을 형성한다는 것을 발견했습니다

1.2 실제 그래프에서의 군집들

조직 내의 분란이 소셜 네트워크 상의 군집으로 표현된 경우도 있습니다



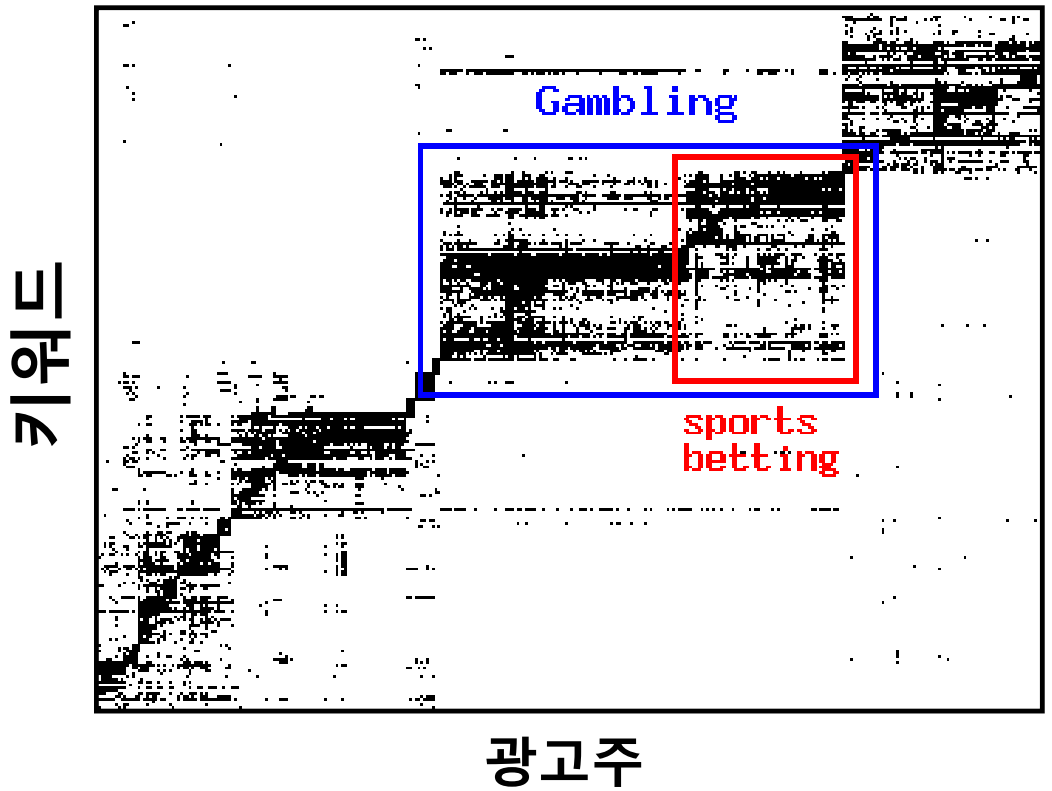
왼쪽의 그래프는 대학교 가라테 동아리 내 친구 관계를 보여줍니다

내분으로 동아리가 둘로 나뉘어지는 사건이 발생했습니다

그 결과 두 개의 군집이 형성되었습니다

1.2 실제 그래프에서의 군집들

키워드 – 광고주 그래프에서는 동일한 주제의 키워드들이 군집을 형성합니다



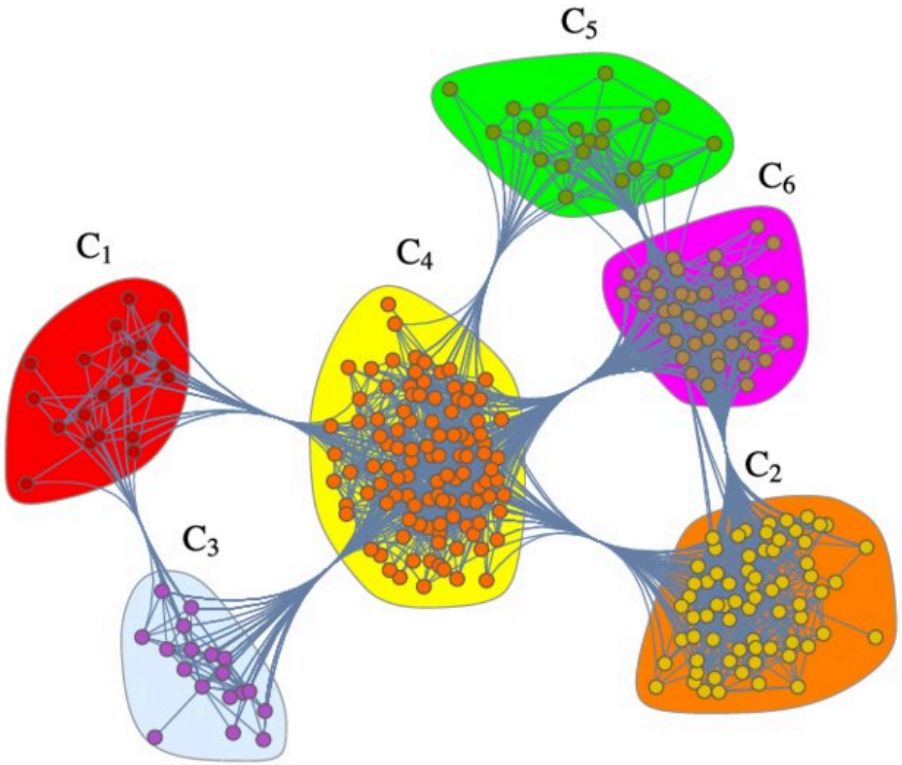
오른쪽은 키워드-광고주 그래프의
인접행렬을 시각화한 것입니다

원소 중 0은 흰색, 나머지는 검은색으로
표시한 것입니다

이미지 출처: http://www2006.thewebconf.org/programme/files/pdf/3095.pdf

1.2 실제 그래프에서의 군집들

뉴런간 연결 그래프에서는 군집들이 **뇌의 기능적 구성 단위**를 의미합니다



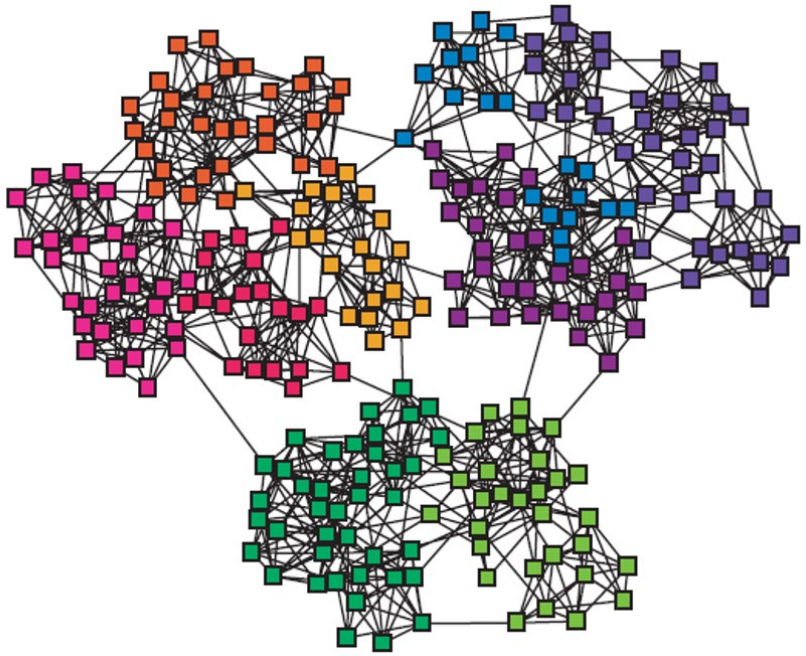
이미지 출처: https://pubmed.ncbi.nlm.nih.gov/26826854/

1.3 군집 탐색 문제

그래프를 여러 군집으로 '잘' 나누는 문제를 군집 탐색(Community Detection) 문제라고 합니다

보통은 각 정점이 한 개의 군집에 속하도록 군집을 나눕니다
비지도 기계학습 문제인 클러스터링(Clustering)과 상당히 유사합니다

먼저 성공적인 군집 탐색부터 정의합시다



이미지 출처: http://infolab.stanford.edu/~ullman/mmds/ch10n.pdf

2. 군집 구조의 통계적 유의성과 군집성

2.1 비교 대상: 배치 모형

2.2 군집성의 정의

2.1 비교 대상: 배치 모형

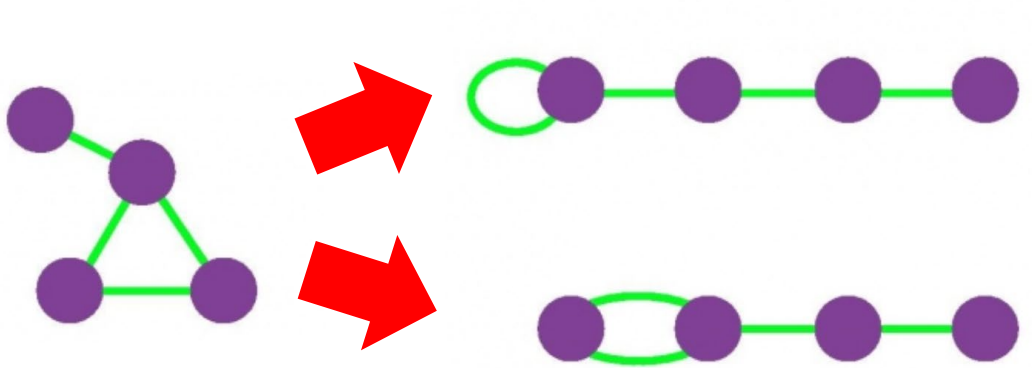
성공적인 군집 탐색을 정의하기 위해 먼저 배치 모형(Configuration Model)을 소개합니다

주어진 그래프에 대한 배치 모형은,

1) 각 정점의 연결성(Degree)을 보존한 상태에서

2) 간선들을 무작위로 재배치하여서 얻은

그래프를 의미합니다



배치 모형에서 임의의 두 정점 i와 j 사이에 간선이 존재할 확률은
두 정점의 연결성에 비례합니다

2.2 군집성의 정의

군집 탐색의 성공 여부를 판단하기 위해서, 군집성(Modularity)가 사용됩니다

그래프와 군집들의 집합 S가 주어졌다고 합시다
각 군집 $s \in S$가 군집의 성질을 잘 만족하는 지를 살펴보기 위해,
군집 내부의 간선의 수를 그래프와 배치 모형에서 비교합니다

구체적으로, 군집성은 다음 수식으로 계산됩니다

$$\frac{1}{2|E|}\sum_{s\in S}(\text{그래프에서 군집 } s \text{ 내부 간선의 수} - \text{배치 모형에서 군집 } s \text{ 내부 간선의 수의 기댓값})$$

즉, 배치 모형과 비교했을 때,
그래프에서 군집 내부 간선의 수가 월등이 많을 수록 성공한 군집 탐색입니다

2.2 군집성의 정의

즉, 군집성은 무작위로 연결된 배치 모형과의 비교를 통해 통계적 유의성을 판단합니다

군집성은 항상 –1과 +1 사이의 값을 갖습니다

보통 군집성이 0.3 ~ 0.7 정도의 값을 가질 때,
그래프에 존재하는 통계적으로 유의미한 군집들을 찾아냈다고 할 수 있습니다

3. 군집 탐색 알고리즘

3.1 Girvan-Newman 알고리즘

3.2 Louvain 알고리즘

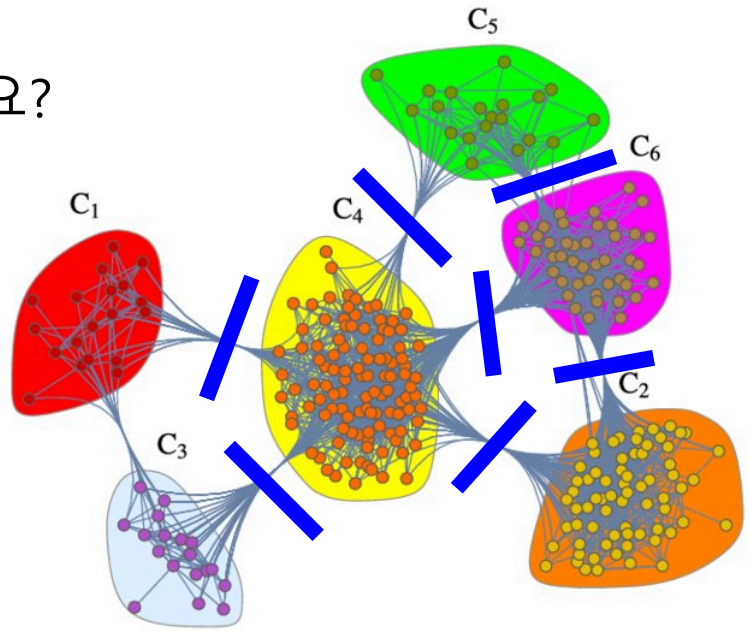
3.1 Girvan-Newman 알고리즘

Girvan-Newman 알고리즘은 대표적인 하향식(Top-Down) 군집 탐색 알고리즘입니다

Girvan-Newman 알고리즘은 전체 그래프에서 탐색을 시작합니다
군집들이 서로 분리되록, 간선을 순차적으로 제거합니다

어떤 간선을 제거해야 군집들이 분리될까요?
바로 서로 다른 군집을 연결하는
다리(Bridge) 역할의 간선입니다

오른쪽 예시에서 파란색 선을 따라
간선을 제거한다고 생각해보세요



이미지 출처: https://pubmed.ncbi.nlm.nih.gov/26826854/

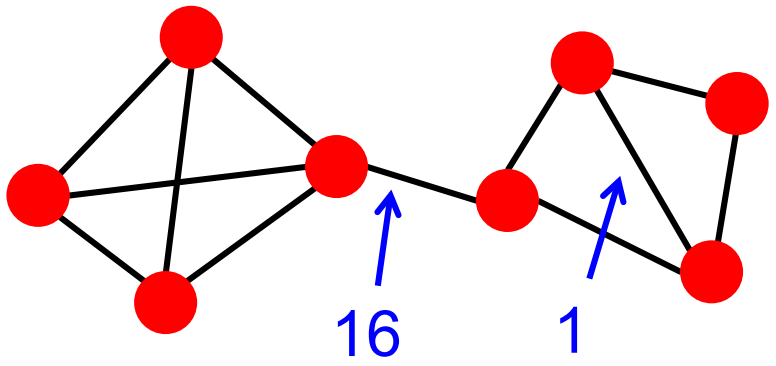
3.1 Girvan-Newman 알고리즘

서로 다른 군집을 연결하는 다리 역할의 간선을 어떻게 찾아낼 수 있을까요?

간선의 매개 중심성(Betweenness Centrality)을 사용합니다
이는 해당 간선이 정점 간의 최단 경로에 놓이는 횟수를 의미합니다

정점 i로 부터 j로의 최단 경로 수를 $\sigma_{i,j}$라고 하고
그 중 간선 (x, y)를 포함한 것을 $\sigma_{i,j}(x, y)$ 라고 합시다
간선 (x, y)의 매개 중심성은 다음 수식으로 계산됩니다

$$\sum_{i<j} \frac{\sigma_{i,j}(x, y)}{\sigma_{i,j}}$$

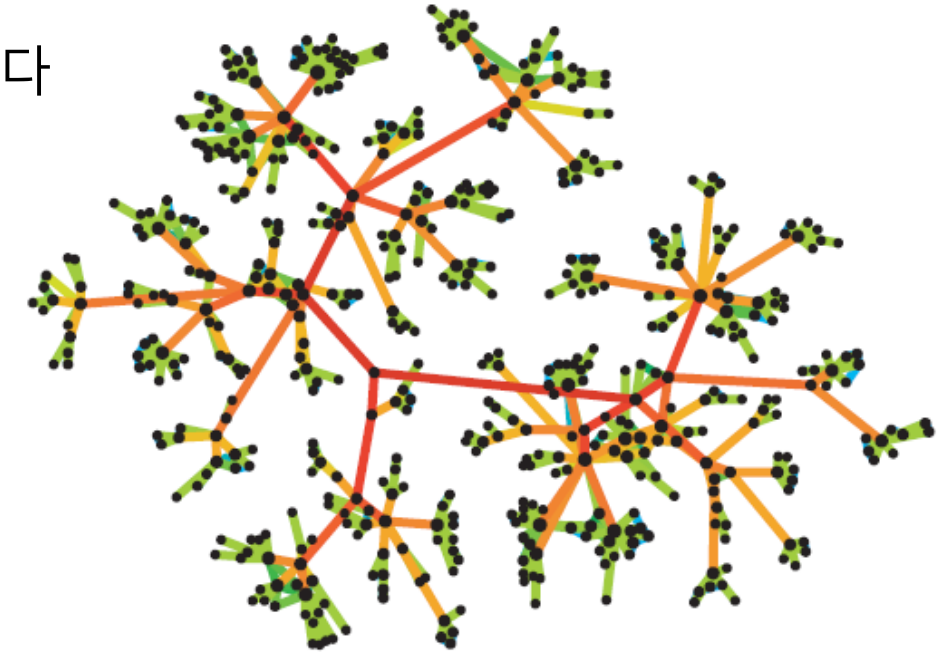


3.1 Girvan-Newman 알고리즘

매개 중심성을 통해 서로 다른 군집을 연결하는 다리 역할의 간선을 찾아낼 수 있습니다

오른쪽 그림은 전화 그래프입니다
매개 중심성이 높을 수록 붉은 색에 가깝습니다

실제로 매개 중심성이 높은 간선들이
서로 다른 군집을 연결하는 다리 역할을
하는 것을 확인할 수 있습니다

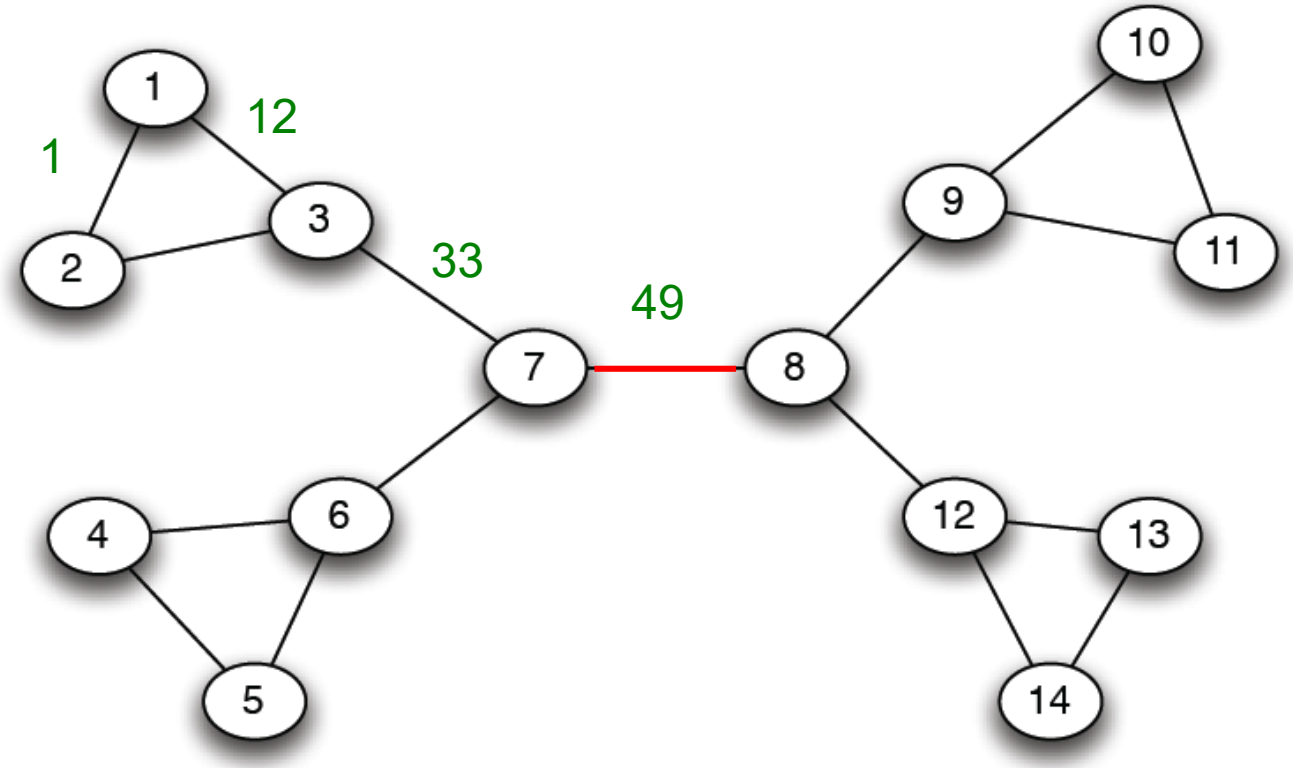


이미지 출처: http://infolab.stanford.edu/~ullman/mmds/ch10n.pdf

3.1 Girvan-Newman 알고리즘

Girvan-Newman 알고리즘은 매개 중심성이 높은 간선을 순차적으로 제거합니다

간선이 제거될 때마다, 매개 중심성을 다시 계산하여 갱신합니다

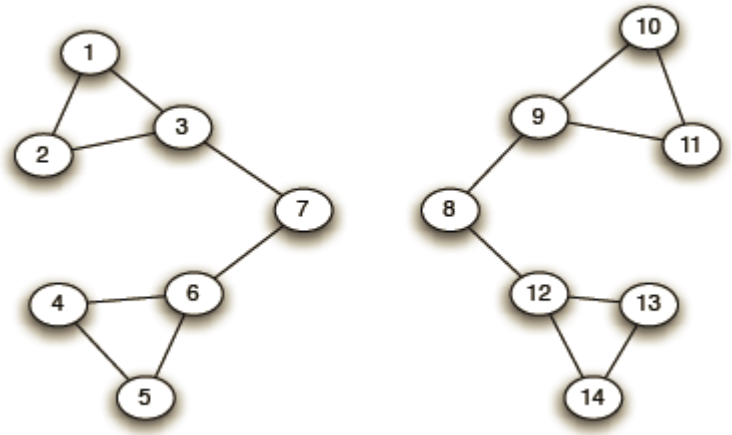


이미지 출처: http://infolab.stanford.edu/~ullman/mmds/ch10n.pdf

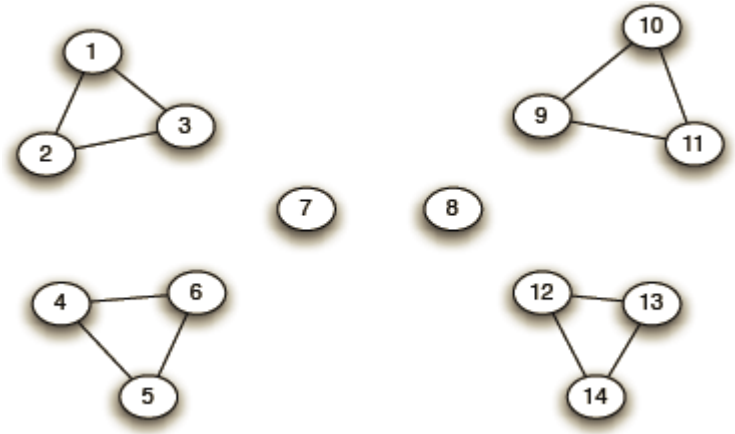
3.1 Girvan-Newman 알고리즘

Girvan-Newman 알고리즘은 매개 중심성이 높은 간선을 순차적으로 제거합니다

단계 1:



단계 2:

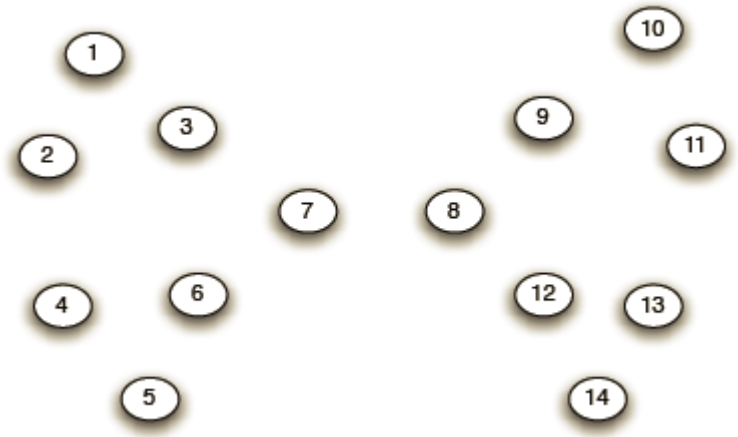


이미지 출처: http://infolab.stanford.edu/~ullman/mmds/ch10n.pdf

3.1 Girvan-Newman 알고리즘

Girvan-Newman 알고리즘은 매개 중심성이 높은 간선을 순차적으로 제거합니다

간선이 모두 제거될 때까지 반복합니다

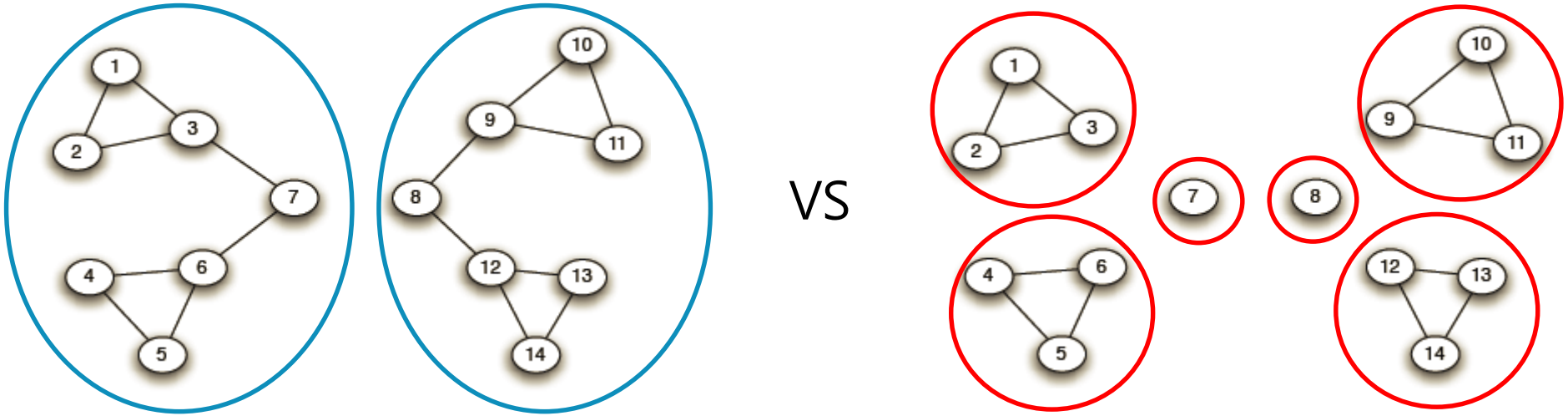


이미지 출처: http://infolab.stanford.edu/~ullman/mmds/ch10n.pdf

3.1 Girvan-Newman 알고리즘

Girvan-Newman 알고리즘은 매개 중심성이 높은 간선을 순차적으로 제거합니다

간선의 제거 정도에 따라서 다른 입도(Granularity)의 군집 구조가 나타납니다



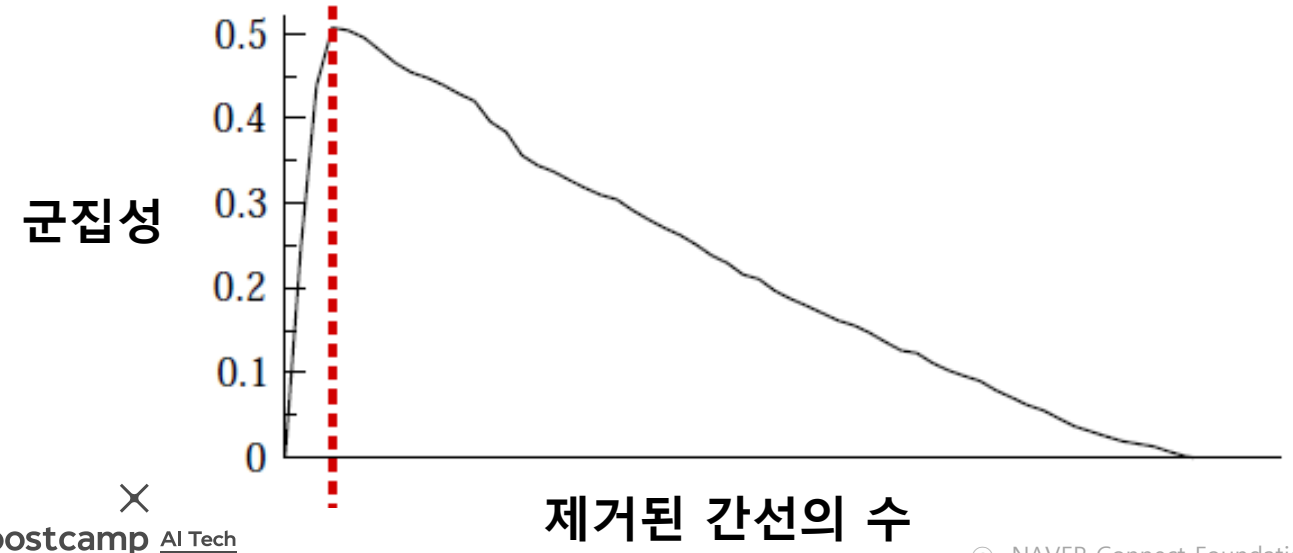
이미지 출처: http://infolab.stanford.edu/~ullman/mmds/ch10n.pdf

3.1 Girvan-Newman 알고리즘

간선을 어느 정도 제거하는 것이 가장 적합할까요?

앞서 정의한 군집성을 그 기준으로 삼습니다
즉, 군집성이 최대가 되는 지점까지 간선을 제거합니다

단, 현재의 연결 요소들을 군집으로 가정하되, 입력 그래프에서 군집성을 계산합니다



이미지 출처: http://infolab.stanford.edu/~ullman/mmds/ch10n.pdf

3.1 Girvan-Newman 알고리즘

정리하면, Girvan-Newman 알고리즘은 다음과 같습니다

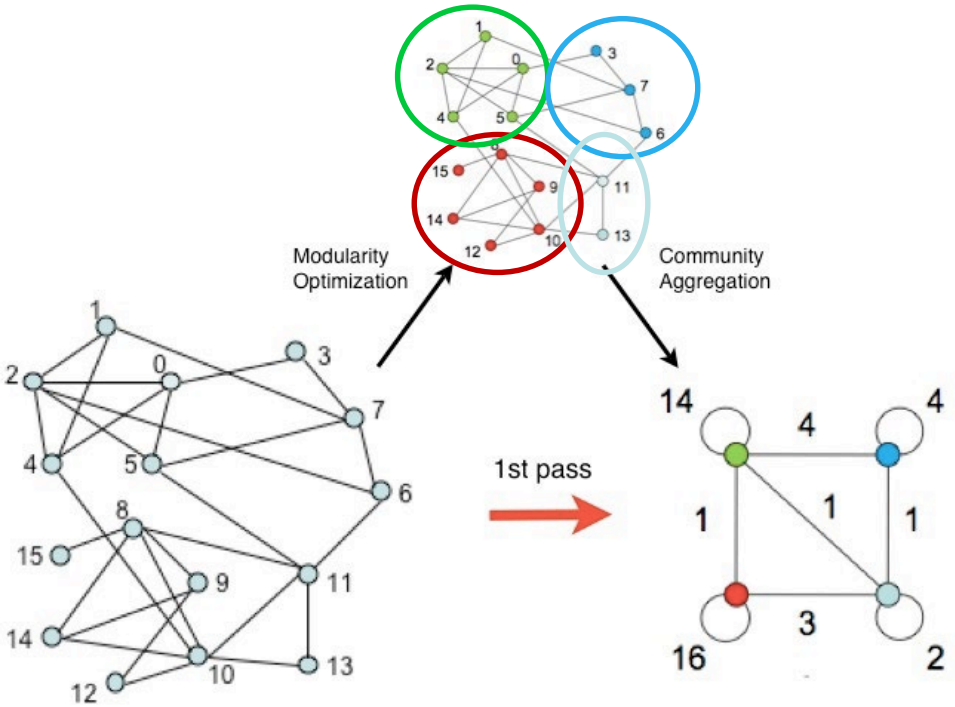
1) 전체 그래프에서 시작합니다
2) 매개 중심성이 높은 순서로 간선을 제거하면서, 군집성을 변화를 기록합니다
3) 군집성이 가장 커지는 상황을 복원합니다
4) 이 때, 서로 연결된 정점들, 즉 연결 요소를 하나의 군집으로 간주합니다

즉, 전체 그래프에서 시작해서 점점 작은 단위를 검색하는 하향식(Top-Down) 방법입니다

3.2 Louvain 알고리즘

Louvain 알고리즘은 대표적인 상향식(Bottom-Up) 군집 탐색 알고리즘입니다

Louvain 알고리즘은 개별 정점에서 시작해서 점점 큰 군집을 형성합니다

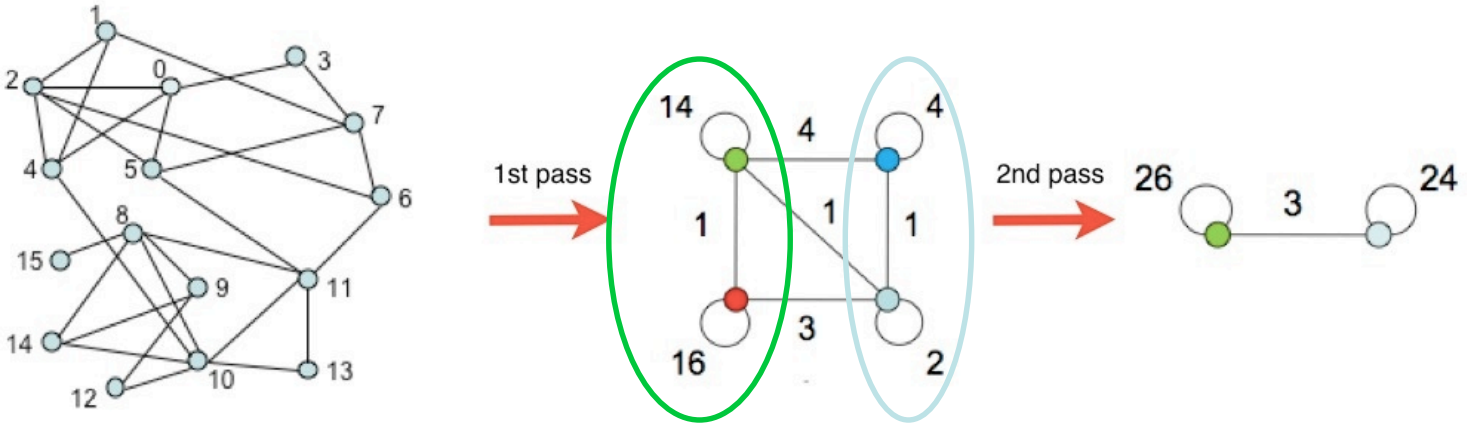


이미지 출처: https://sites.google.com/site/findcommunities/

3.2 Louvain 알고리즘

Louvain 알고리즘은 대표적인 상향식(Bottom-Up) 군집 탐색 알고리즘입니다

Louvain 알고리즘은 개별 정점에서 시작해서 점점 큰 군집을 형성합니다



이미지 출처: https://sites.google.com/site/findcommunities/

3.2 Louvain 알고리즘

그러면 어떤 기준으로 군집을 합쳐야 할까요?

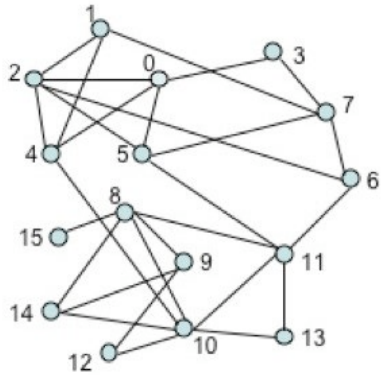
정답은 이번에도 군집성입니다!

이미지 출처: https://sites.google.com/site/findcommunities/

3.2 Louvain 알고리즘

구체적으로 Louvain 알고리즘의 동작 과정은 다음과 같습니다

1) Louvain 알고리즘은 개별 정점으로 구성된 크기 1의 군집들로부터 시작합니다

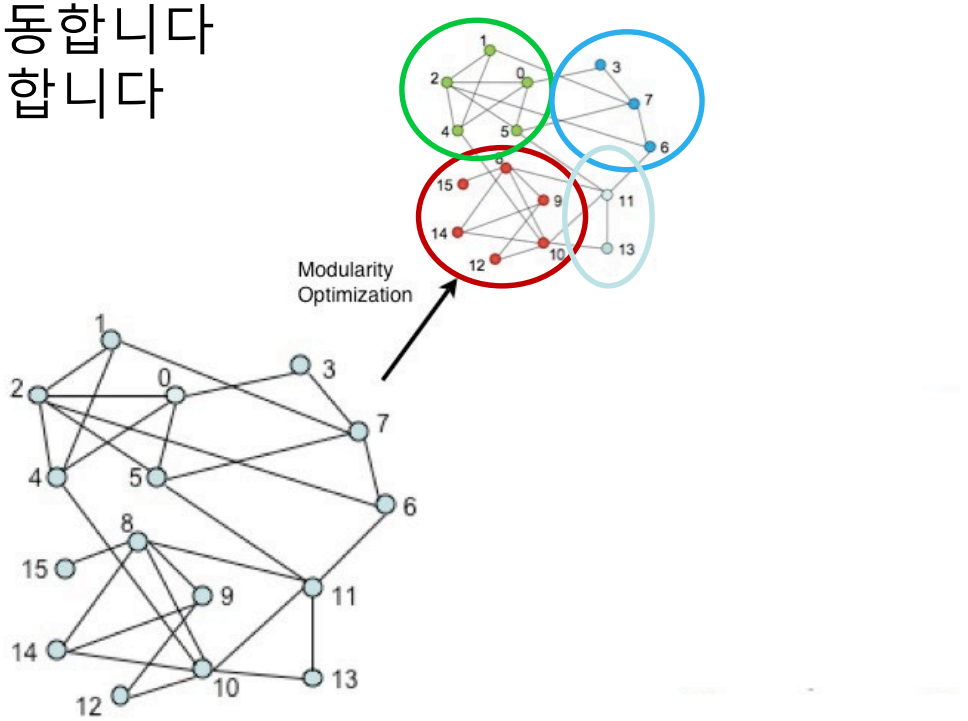


이미지 출처: https://sites.google.com/site/findcommunities/

3.2 Louvain 알고리즘

구체적으로 Louvain 알고리즘의 동작 과정은 다음과 같습니다

1) Louvain 알고리즘은 개별 정점으로 구성된 크기 1의 군집들로부터 시작합니다
2) 각 정점 u를 기존 혹은 새로운 군집으로 이동합니다
   이 때, 군집성이 최대화되도록 군집을 결정합니다
3) 더 이상 군집성이 증가하지 않을 때까지
   2)를 반복합니다

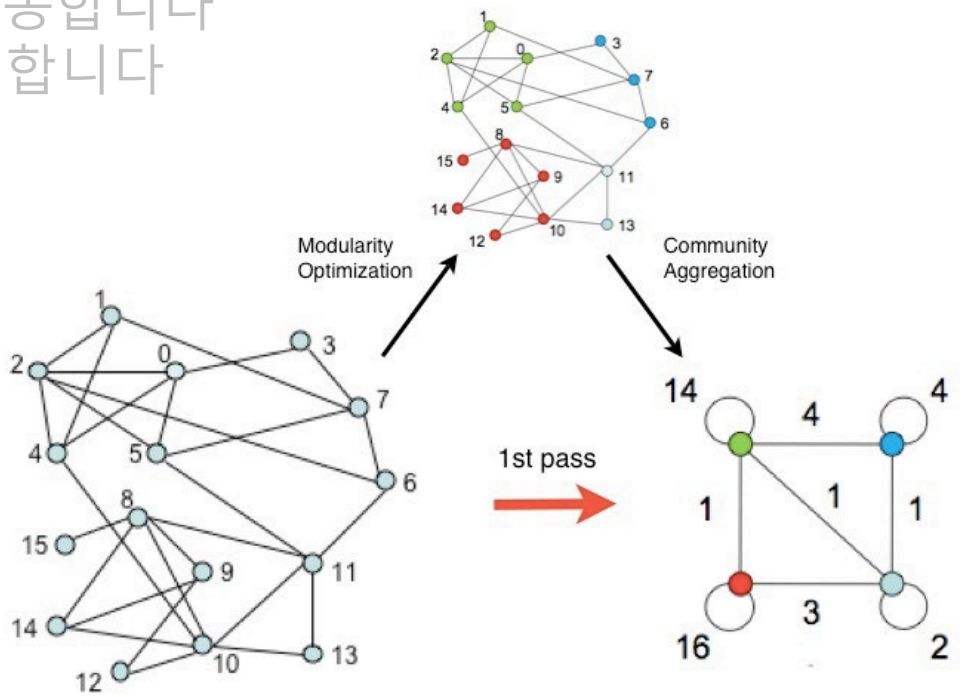


이미지 출처: https://sites.google.com/site/findcommunities/

3.2 Louvain 알고리즘

구체적으로 Louvain 알고리즘의 동작 과정은 다음과 같습니다

1) Louvain 알고리즘은 개별 정점으로 구성된 크기 1의 군집들로부터 시작합니다
2) 각 정점 u를 기존 혹은 새로운 군집으로 이동합니다
   이 때, 군집성이 최대화되도록 군집을 결정합니다
3) 더 이상 군집성이 증가하지 않을 때까지
   2)를 반복합니다
4) 각 군집을 하나의 정점으로하는
   군집 레벨의 그래프를 얻은 뒤
   3)을 수행합니다

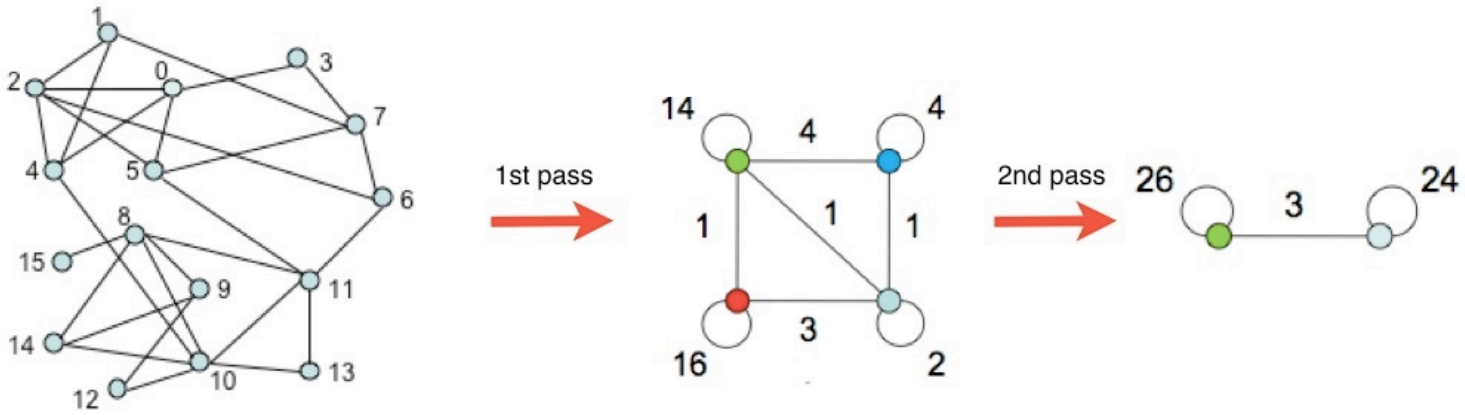


이미지 출처: https://sites.google.com/site/findcommunities/

3.2 Louvain 알고리즘

구체적으로 Louvain 알고리즘의 동작 과정은 다음과 같습니다

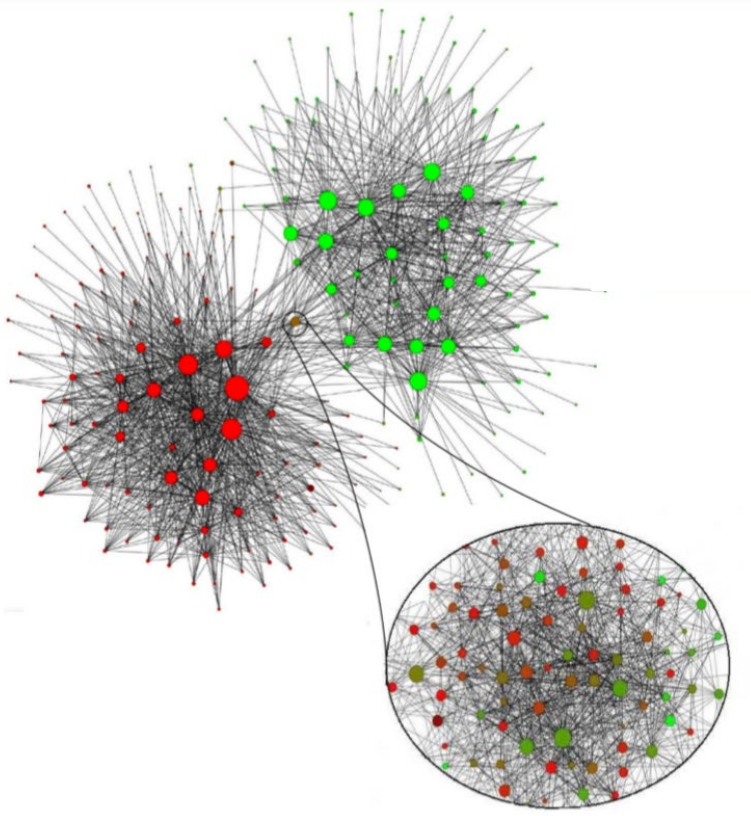
1) Louvain 알고리즘은 개별 정점으로 구성된 크기 1의 군집들로부터 시작합니다
2) 각 정점 u를 기존 혹은 새로운 군집으로 이동합니다
   이 때, 군집성이 최대화되도록 군집을 결정합니다
3) 더 이상 군집성이 증가하지 않을 때까지
   2)를 반복합니다
4) 각 군집을 하나의 정점으로하는
   군집 레벨의 그래프를 얻은 뒤
   3)을 수행합니다
5) 한 개의 정점이 남을 때까지
   4)를 반복합니다



이미지 출처: https://sites.google.com/site/findcommunities/

3.2 Louvain 알고리즘

Louvain 알고리즘을 이용해 소셜 네트워크의 군집을 탐색한 결과를 살펴봅시다



그래프의 정점들은 실제로는 여러 정점들로 구성된 군집입니다
불어를 사용하는 사람의 비율이 높을 수록 녹색에,
독어를 사용하는 사람의 비율이 높을 수록 붉은색에 가깝습니다

대부분 군집들이
높은 비율의 불어를 사용하는 사람들 혹은
높은 비율의 독어를 사용하는 사람들로 구성되어 있습니다

예외적으로 가운데 군집 하나는
두 언어를 사용하는 사람들이
혼합되어 있으며, 다리 역할을 합니다

이미지 출처: https://sites.google.com/site/findcommunities/

4. 중첩이 있는 군집 탐색

4.1 중첩이 있는 군집 구조

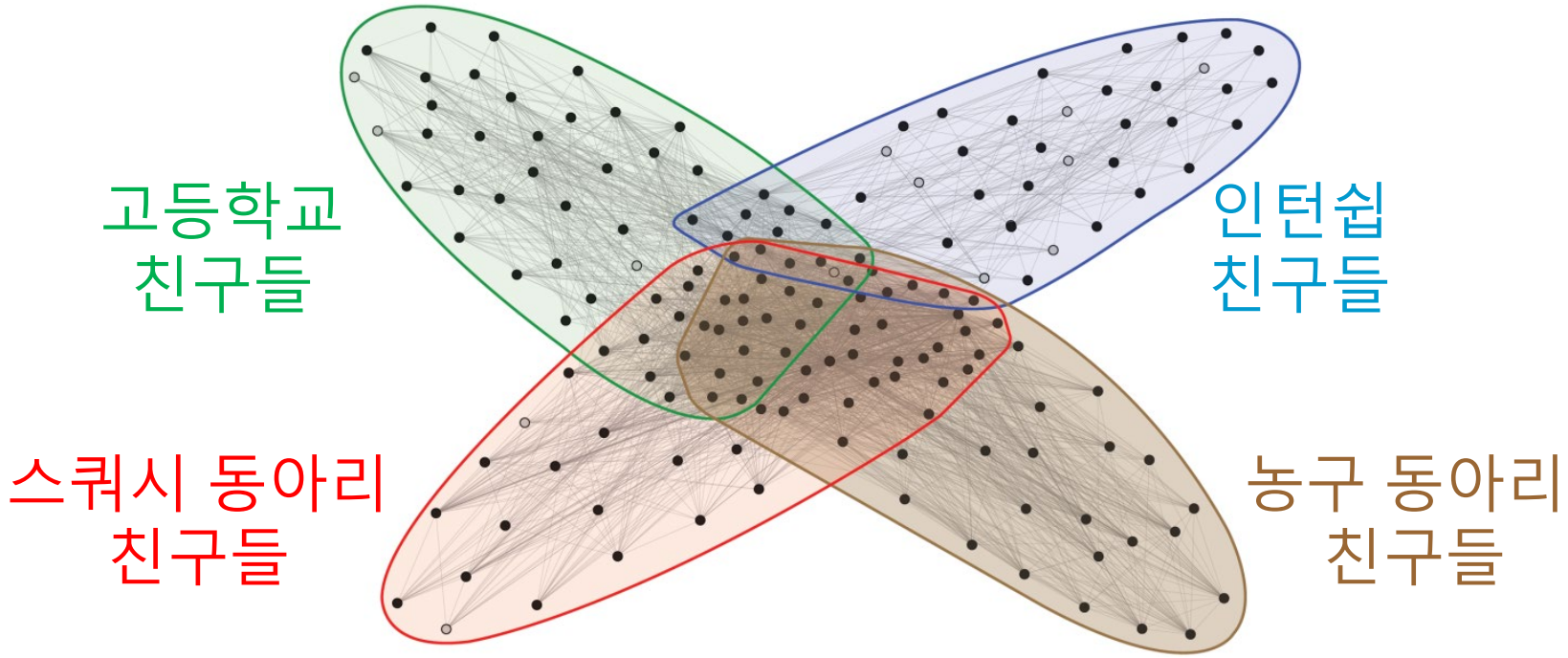
4.2 중첩 군집 모형

4.3 완화된 중첩 군집 모형

4.1 중첩이 있는 군집 구조

실제 그래프의 군집들을 중첩되어 있는 경우가 많습니다

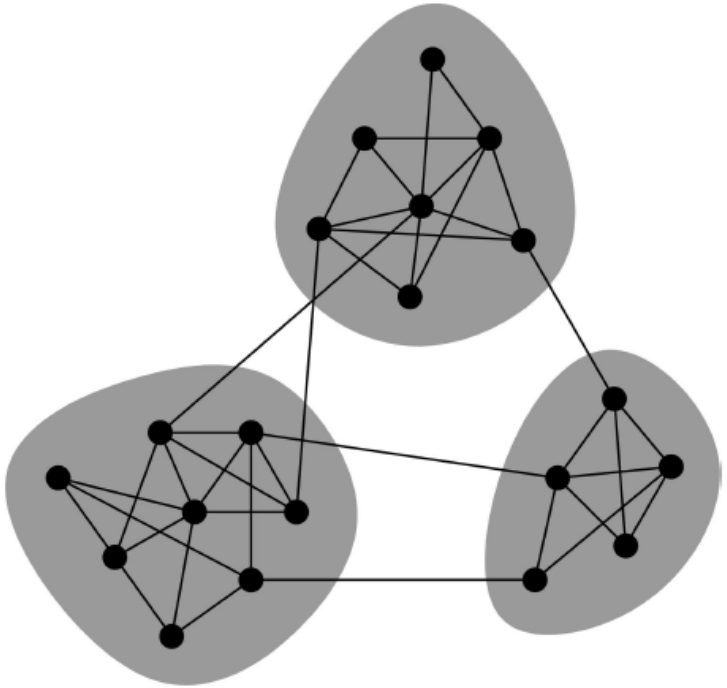
예를 들어 소셜 네트워크에서의 개인은 여러 사회적 역할을 수행합니다
그 결과 여러 군집에 속하게 됩니다



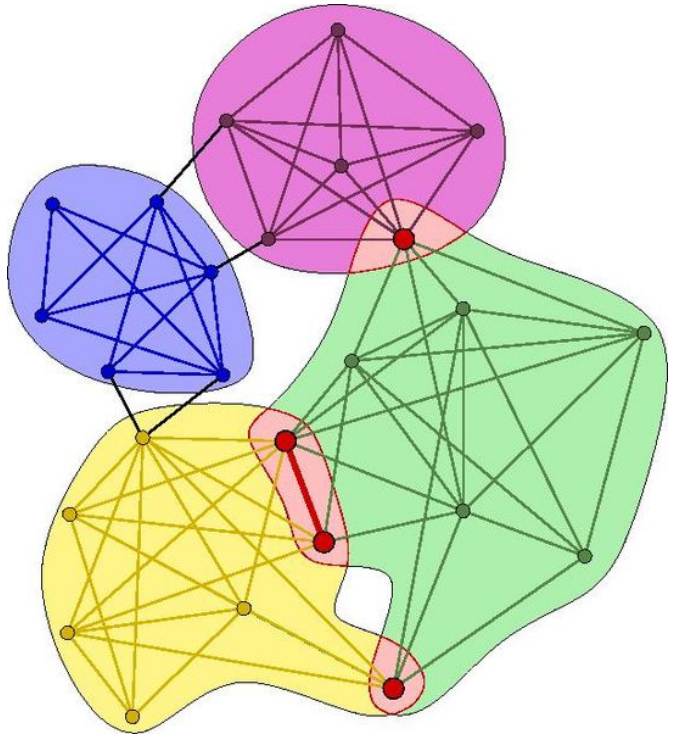
이미지 출처: https://sites.google.com/site/findcommunities/

4.1 중첩이 있는 군집 구조

앞서 배운 Girvan-Newman 알고리즘, Louvain 알고리즘은 군집 간의 중첩이 없다고 가정합니다
그러면 중첩이 있는 군집은 어떻게 찾아낼 수 있을까요?



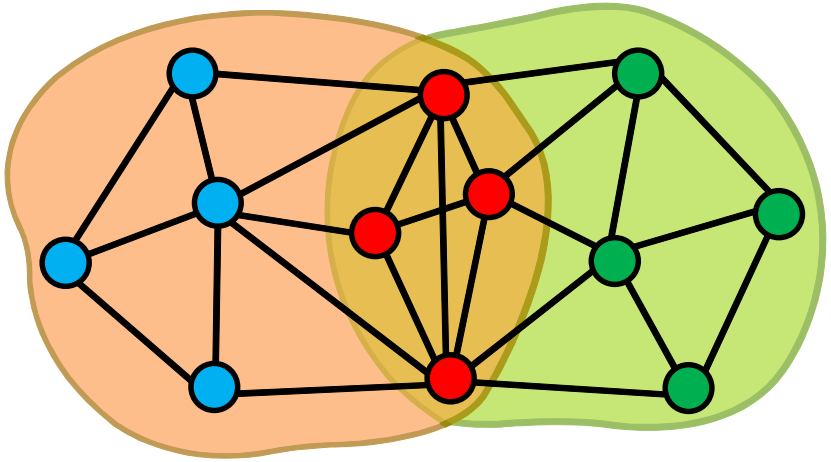
VS



이미지 출처: https://sites.google.com/site/findcommunities/

4.2 중첩 군집 모형

이를 위해 아래와 같은 중첩 군집 모형을 가정합니다



1) 각 정점은 여러 개의 군집에 속할 수 있습니다

2) 각 군집 A에 대하여, 같은 군집에 속하는
두 정점은 P_A 확률로 간선으로 직접 연결됩니다

3) 두 정점이 여러 군집에 동시에 속할 경우 간선 연결 확률은 독립적입니다
예를 들어, 두 정점이 군집 A와 B에 동시에 속할 경우
두 정점이 간선으로 직접 연결될 확률은 $1-(1-P_A)(1-P_B)$입니다

4) 어느 군집에도 함께 속하지 않는 두 정점은 낮은 확률 ϵ으로 직접 연결됩니다

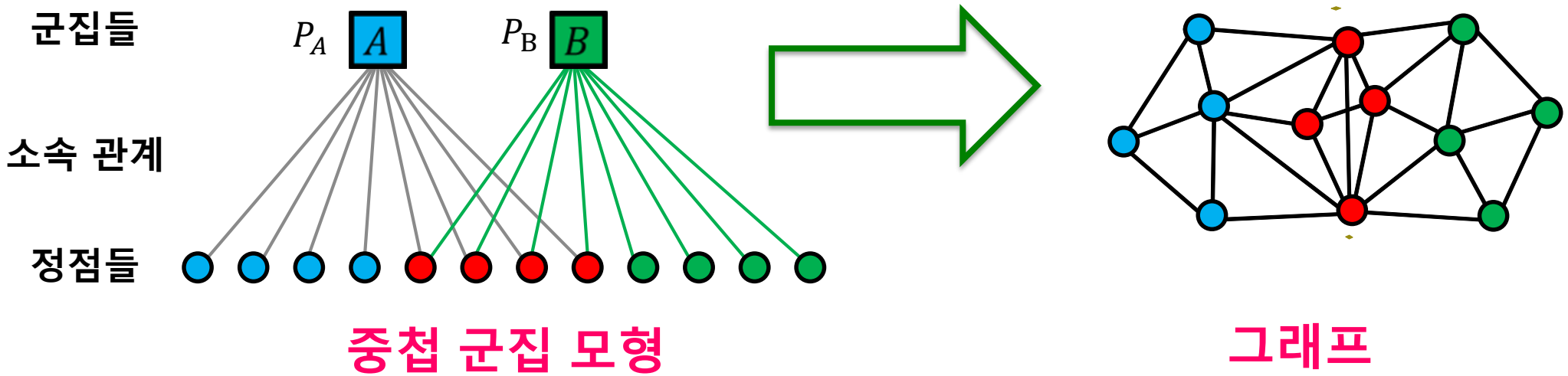
이미지 출처: https://sites.google.com/site/findcommunities/

4.2 중첩 군집 모형

중첩 군집 모형이 주어지면, 주어진 그래프의 확률을 계산할 수 있습니다

그래프의 확률은 다음 확률들의 곱입니다

1) 그래프의 각 간선의 두 정점이 (모형에 의해) 직접 연결될 확률
2) 그래프에서 직접 연결되지 않은 각 정점 쌍이 (모형에 의해) 직접 연결되지 않을 확률

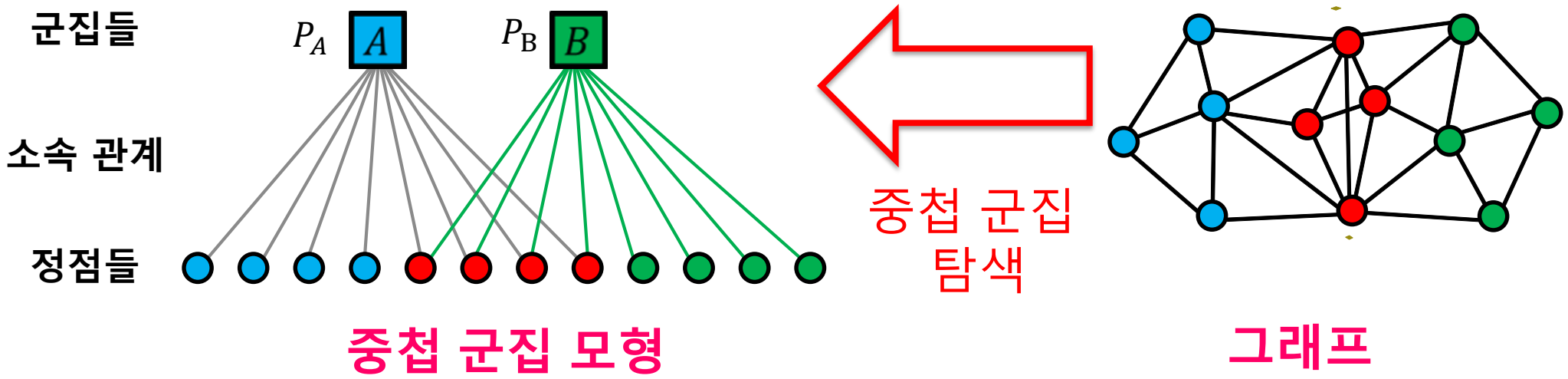


이미지 출처: https://sites.google.com/site/findcommunities/

4.2 커뮤니티 모형과 완화된 커뮤니티 모형

중첩 군집 탐색은 주어진 그래프의 확률을 최대화하는 중첩 군집 모형을 찾는 과정입니다

통계 용어를 빌리면, 최우도 추정치(Maximum Likelihood Estimate)를 찾는 과정입니다



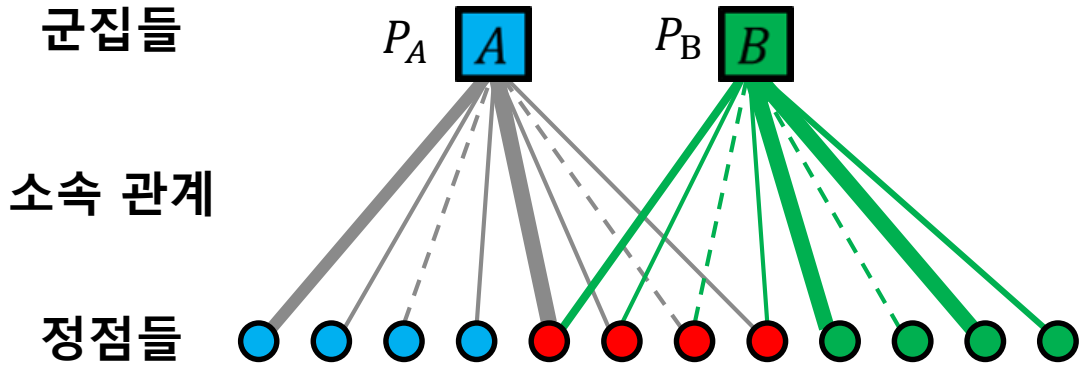
이미지 출처: https://sites.google.com/site/findcommunities/

4.3 완화된 중첩 군집 모형

중첩 군집 탐색을 용이하게 하기 위하여 완화된 중첩 군집 모형을 사용합니다

완화된 중첩 군집 모형에서는 각 정점이 각 군집에 속해 있는 정도를 실숫값으로 표현합니다

즉, 기존 모형에서는 각 정점이 각 군집에 속하거나 속하지 않거나 둘 중 하나였는데,
중간 상태를 표현할 수 있게 된 것입니다



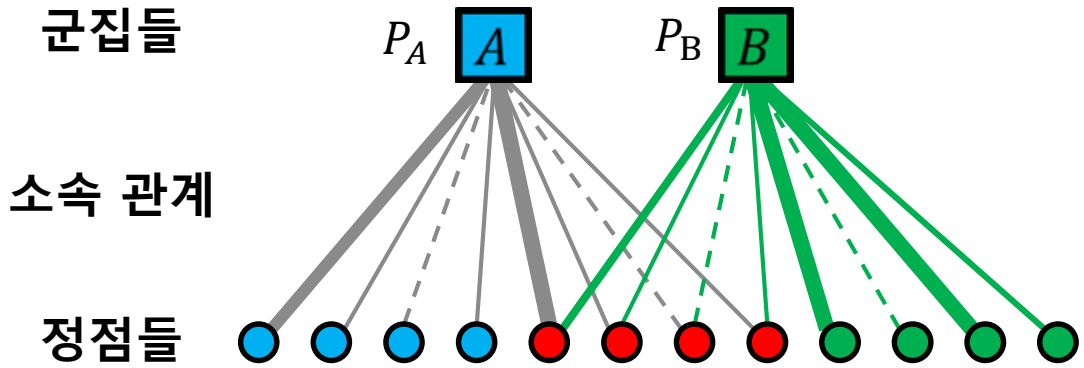
완화된 중첩 군집 모형

이미지 출처: https://sites.google.com/site/findcommunities/

4.3 완화된 중첩 군집 모형

중첩 군집 탐색을 용이하게 하기 위하여 완화된 중첩 군집 모형을 사용합니다

최적화 관점에서는, 모형의 매개변수들이 실수 값을 가지기 때문에
익숙한 최적화 도구 (경사하강법 등)을 사용하여 모형을 탐색할 수 있다는 장점이 있습니다



완화된 중첩 군집 모형

이미지 출처: https://sites.google.com/site/findcommunities/

5. 실습: Girvan-Newman 알고리즘 구현 및 적용

5.1 실습용 데이터 불러오기

5.2 Girvan-Newman 알고리즘 구현

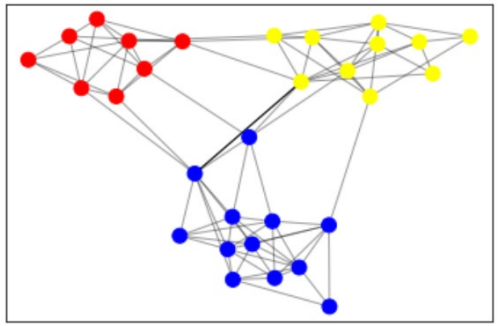
5.3 가라테 동아리 내의 군집 분석

5.4 NetworkX 제공 군집 분석 알고리즘

5.1 실습용 데이터 불러오기

필요한 라이브러리를 불러오고, 실습에 사용할 그래프를 파일에서 읽어옵니다

```
import matplotlib.pyplot as plt
import numpy as np
import networkx as nx
from networkx.algorithms import community
```



```
G = nx.Graph()
f = open('drive/MyDrive/data/community_graph.txt')
for line in f:
    v1, v2 = map(int, line.split())
    G.add_edge(v1, v2)
```

5.2 Girvan-Newman 알고리즘 구현

먼저 각종 변수들을 초기화합니다

```
[ ]  def GirvanNewmanAlgorithm(G, nodeColorList):

        copyG = G.copy()
        step = 0
        logModularityList = []
        maxModCom = []
        maxMod = -1
        maxStep = 0
```

5.2 Girvan-Newman 알고리즘 구현

그래프 현재 상태에서 군집성을 계산하여 관련 정보를 갱신합니다

```
[ ]  def GirvanNewmanAlgorithm(G, nodeColorList):
         ...
         while len(copyG.edges()) > 0:
             recComList = sorted(nx.connected_components(copyG), key=len, reverse=True)
             recMod = community.modularity(G, communities=recComList)
             if recMod > maxMod:
                 maxModG = copyG.copy()
                 maxMod = recMod
                 maxModCom = []
                 for j in range(len(recComList)):
                     maxModCom = maxModCom + [recComList[j]]
                 maxStep = step
```

5.2 Girvan-Newman 알고리즘 구현

매개 중심성이 가장 큰 간선을 찾아 순차적으로 제거합니다

```
[ ]  def GirvanNewmanAlgorithm(G, nodeColorList):
         ...
         while len(copyG.edges()) > 0:
             ...

             step = step + 1
             betweenness = nx.edge_betweenness_centrality(copyG)
             maxEdge = max(betweenness, key=betweenness.get)
             copyG.remove_edge(maxEdge[0], maxEdge[1])
```

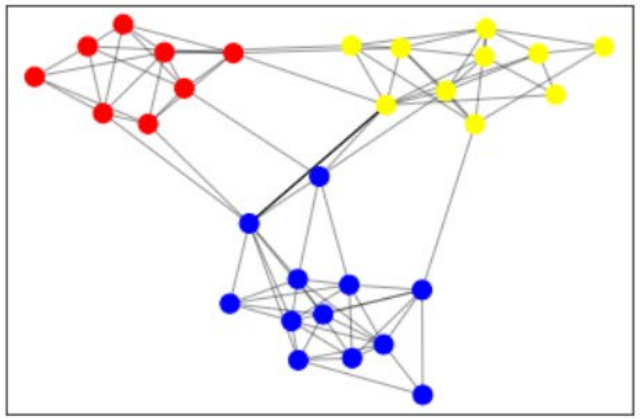
5.2 Girvan-Newman 알고리즘 구현

두 부분을 함께 적으면 아래와 같습니다

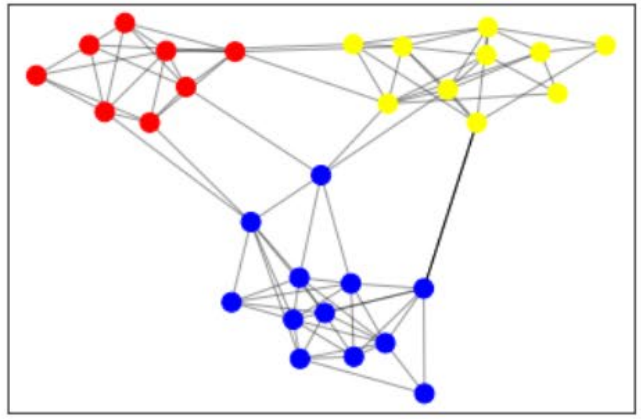
```
[ ]  def GirvanNewmanAlgorithm(G, nodeColorList):
         ...
         while len(copyG.edges()) > 0:
             recComList = sorted(nx.connected_components(copyG), key=len, reverse=True)
             recMod = community.modularity(G, communities=recComList)
             if recMod > maxMod:
                 maxModG = copyG.copy()
                 maxMod = recMod
                 maxModCom = []
                 for j in range(len(recComList)):
                     maxModCom = maxModCom + [recComList[j]]
                 maxStep = step
             step = step + 1
             betweenness = nx.edge_betweenness_centrality(copyG)
             maxEdge = max(betweenness, key=betweenness.get)
             copyG.remove_edge(maxEdge[0], maxEdge[1])
```

5.2 Girvan-Newman 알고리즘 구현

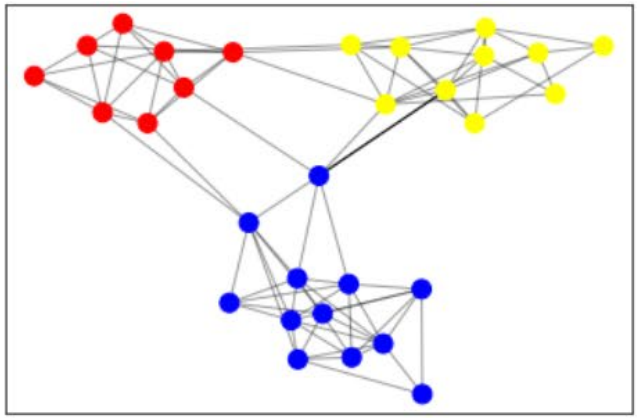
단계별 그래프의 상태는 다음과 같습니다



단계 1



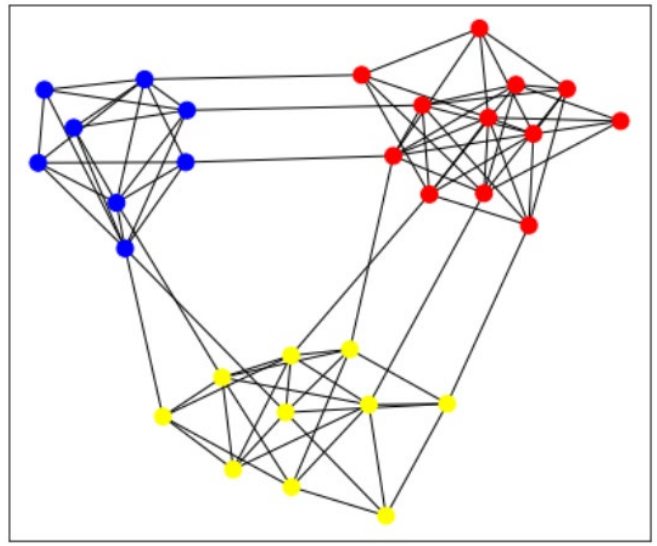
단계 2



단계 3

5.2 Girvan-Newman 알고리즘 구현

군집성을 토대로 선정된 최종 군집을 시각화합니다

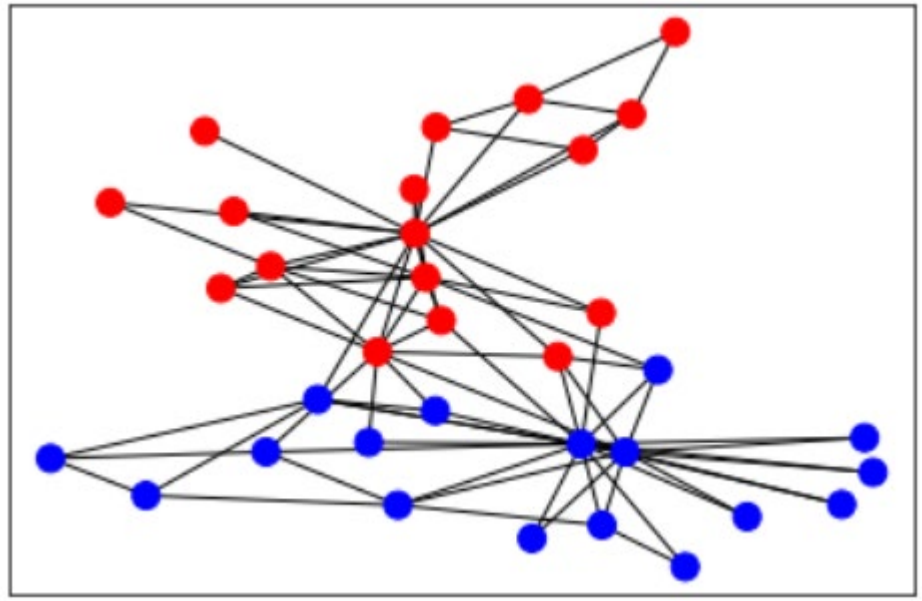


```
pos = nx.spring_layout(G)
fig = plt.figure(figsize=(7, 6))
predictedNodeColorList = []
for i in range(len(G.nodes)):
    if i in maxModCom[0]:
        predictedNodeColorList = predictedNodeColorList + ['red']
    elif i in maxModCom[1]:
        predictedNodeColorList = predictedNodeColorList + ['yellow']
    elif i in maxModCom[2]:
        predictedNodeColorList = predictedNodeColorList + ['blue']
nx.draw_networkx_nodes(G, pos, node_color=predictedNodeColorList, node_size=100)
nx.draw_networkx_edges(G, pos)
plt.show()
```

5.3 가라테 동아리 내의 군집 분석

가라테 동아리 그래프를 불러와서 시각화합니다

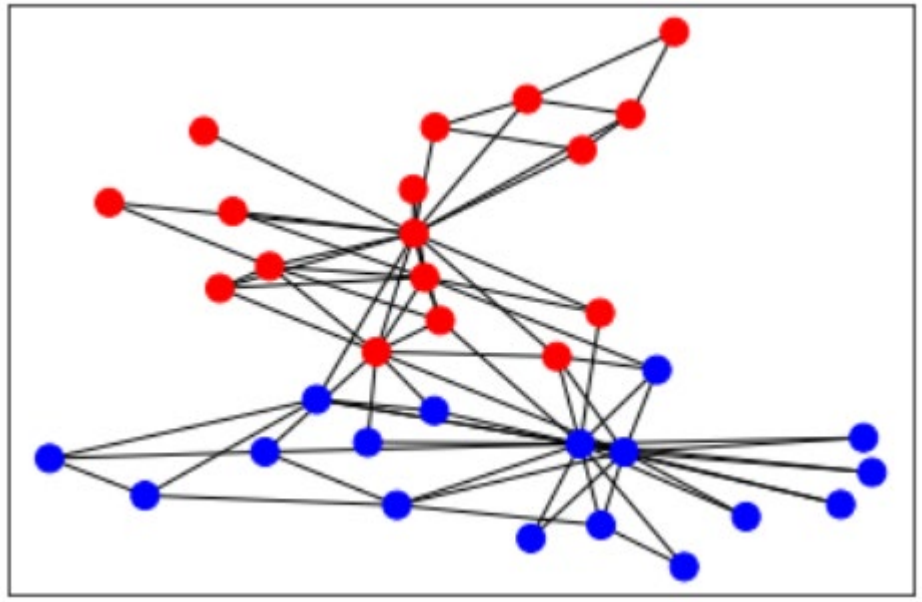
주어진 레이블은 동아리가 둘로 분열 되었을 때,
어느 쪽에 속했는지를 의미합니다



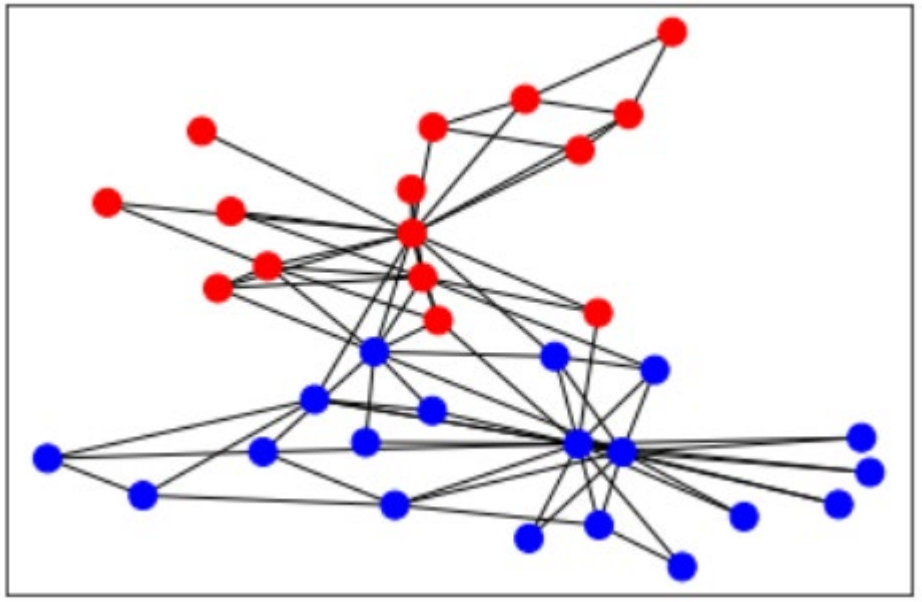
```
G = nx.karate_club_graph()
label = [G.nodes[i]["club"] for i in range(len(G.nodes))]
nodeColorList = []
for i in range(len(G.nodes)):
    if label[i] == 'Mr. Hi':
        nodeColorList = nodeColorList + ['red']
    elif label[i] == 'Officer':
        nodeColorList = nodeColorList + ['blue']

nx.draw_networkx_nodes(G,pos, node_color=nodeColorList, node_size=100)
nx.draw_networkx_edges(G, pos)
plt.show()
```

5.3 가라테 동아리 내의 군집 분석

주어진 레이블과 군집 분석을 통해 얻은 결과를 비교합니다



군집 분석



주어진 레이블

5.4 NetworkX 제공 군집 분석 알고리즘

NetworkX에서는 다양한 군집 분석 알고리즘을 제공합니다

```
[ ]  girvan_newman(G)
     greedy_modularity_communities(G)
     kernighan_lin_bisection(G)
```

5강 정리

1. 군집 구조와 군집 탐색 문제

- 실제 그래프의 군집들은 무엇을 의미할까?
- 군집을 어떻게 찾아낼까?

2. 군집 구조의 통계적 유의성과 군집성

- 배치 모형과의 비교를 통해 군집성을 측정

3. 군집 탐색 알고리즘

- 하향식 알고리즘: Girvan-Newman 알고리즘
- 상향식 알고리즘: Louvain 알고리즘

4. 중첩이 있는 군집 탐색

- (완화된) 중첩 군집 모형

5. 실습: Girvan-Newman 알고리즘 구현 및 적용