Community Detection in Various Network Datasets Using Genetic Algorithms

201835707 이동원

202135321 류종현

201835601 가한솔

201839486 안창훈

# Introduction

정보 기술의 발전으로 복잡한 네트워크의 규모는 계속 커지고 있다. 소셜 네트워크가 대표적인 예이다. 소셜 네트워크 분석(SNA – Social Network Analysis)은 복잡한 소셜 네트워크를 복잡한 관계를 가진 그래프 구조로 추상화한다. 많은 연구가 복잡한 소셜 네트워크를 바탕으로 수행될 수 있다. 일부 연구자들은 복잡한 네트워크의 커뮤니티 구조 분석을 통해 전염병 확산 메커니즘을 발견했다. Kim , et al. 은 Twitter 네트워크에 적용된 클러스터링 기술과 트위터 트렌드 분석을 다루었다. 실시간 사용자 생성 콘텐츠를 수집하여 선거 결과를 예측하는 데 사용할 수 있다 [2].

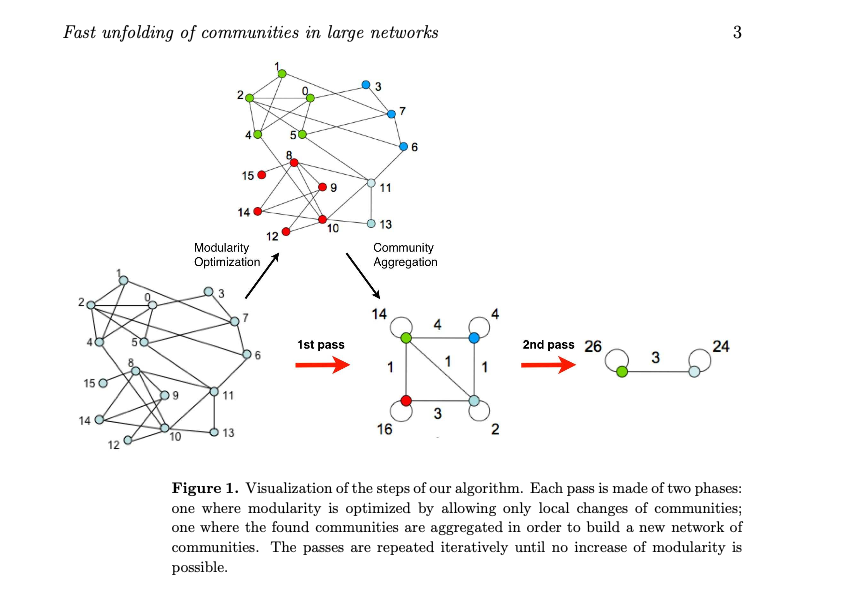
소셜 네트워크나 여러가지 일상 생활에서 주목받고 있는 연구주제가 Community Detection이다. 이는 대규모 Network에서 NP – Hard Problem으로 간주되며 이를 찾기 위해 많은 Meta – Heuristic Algorithm들이 제안되었다. Algorithm들의 대부분은 Modularity를 Objective Function으로 사용하고 있으며 Modularity의 최적화 측면은 네트워크 내 존재하는 Edge의 수에 따라 감지된 커뮤니티의 크기가 달라지는 해상도 한계(Resolution Limits)라는 문제와 관련이 있다 [1]. 이 논문(보고서)은 Large – Scale Network에 대한 Community Detection의 중요성에 대해서 다룬다.

Community Detection은 복잡한 Network분석에서 같은 커뮤니티의 노드들이 강한 결속력으로 인해 유사한 특성을 가진다. 네트워크의 내부 구조를 식별하고 시각화하며, 유용한 정보를 탐지하고, 개인 간의 관계를 분석하는 데 도움이 된다. 지난 10년 동안 다양한 Community Detection 방법들이 발견되었다. 이에는 Global Community Detection알고리즘, Local Community Detection알고리즘, Nested Community Detection 알고리즘이 포함된다. 네트워크에서 Community Detection가 수년 동안 연구되었음에도 불구하고, 대규모 네트워크 환경에서 고속이면서 고품질의 Community Detection 알고리즘은 여전히 중요하다[2].

Community Detection 문제는 Network를 밀접하게 연결된 Node들의 Community로 분할하는 것을 요구하며, 다른 커뮤니티에 속한 Node들은 드물게 연결되어 있다. Modularity는 Community 내부의 Link 밀도와 Community 간의 Link와 비교하여 측정한다. Modularity의 공식은 아래와 같다. 이는 Section 3 : Methodology 에서 더 자세하게 다룰 예정이다.



대규모 Network에서 Modularity를 최적화하기 위해 가장 빠른 근사 Algorithm은 Clauset et al.에 의해 제안되었다. 이 방법은 Greedy Algorithm으로 Modularity의 생산을 최적화하는 Community를 반복적으로 병합하는 것으로 구성된다[8].



<Louvain Algorithm [8]>

지난 십년간 많은 Community Detection 알고리즘이 제안되었다. 최적화 기반 방법은 Community Detection에 인기 있는 방법이다. 최적화 기반 방법에서는 네트워크 분할의 품질을 측정하는 목적 함수(Objective Function)가 정의된다. 알고리즘의 효율성은 최적화 알고리즘의 검색 능력과 그들이 사용하는 목적 함수에 달려 있다. 가장 인기 있는 목적 함수는 Newman과 Girvan의 Modularity이다

GA(Genetic Algorithm)은 본질적으로 병렬이며 목적 함수와 제약 조건의 미분 가능성을 요구하지 않으며, 한 번의 실행에서 가능한 해답의 집합을 다룬다. 표준 Genetic Algorithm에서는 문제에 대한 Candidate Solution들이 존재하며 이들은 염색체형으로 숫자형으로 Encoding된다. 또한 Solution공간에서 최적화될 목적함수(Objective Function)이 있다. 목적함수는 염색체에 대한 생물학적 적합성의 역할을 한다. 일반적으로 Random한 후보 해답 집합에서 시작하여, 실제 염색체와 관련된 생물학적 과정에 영감을 받은 Manipulation을 통해 점진적으로 변경된다. 이러한 Manipulation에는 Point – Mutation (염색체의 일부 부분의 임의 변형)와 Crossover (기존 염색체의 부분을 병합하여 새로운 염색체 생성)가 포함된다. 후보들의 새로운 Generation의 Fitness이 계산되고 가장 높은 fitness value를 가진 염색체가 다음 세대에서 살아남을 가능성이 가장 크다. 여러 반복 후에는 높은 fitness value를 가진 해집단(Population)만이 살아남는다[7].

Louvain Algorithm은 그래프 이론에 기반한 효율적인 계층적 클러스터링 알고리즘이다. 이 알고리즘의 원리는 Node를 지속적으로 이동시키며 커뮤니티 분할 결과의 Modularity를 최대치로 만드는 것이다[2]. Louvain Algorithm의 계산 효율성은 분할 결과를 보장하는 전제 하에 다른 Community Detection 알고리즘들보다 우수하며, Louvain Algorithm은 병렬 처리가 가능하다[16]. Section 2 ‘Related Works’에서 조금 더 자세하게 다룰 예정이다.

이 논문(보고서)의 목표는 Louvain 알고리즘을 사용하여 대규모 네트워크의 커뮤니티 구조를 빠르게 탐지하는 것이며 GA(Genetic Algorithm)을 이용하여 Modularity 점수를 높이는 것에 목표를 두고 있다. 소규모 데이터셋 1개, 중간 규모 1개 대규모 데이터셋 1개의 실험적 테스트를 통해 Genetic Algorithm + Louvain Algorithm으로 Community Detection의 Modularity 점수를 높이는 것에 대해서 서술하겠다.

이 논문은 다음과 같은 형식으로 구성이 된다. Section 2에서는 주제와 관련된 주제, Section 3, 4에서는 우리의 방법론이랑 실험 방법, Section5에서는 실험의 결과, Section6에는 결론과 실험의 한계에 대해서 설명을 하겠다.

# Related Work

Community Detection과 관련된 가장 유명한 알고리즘은 Newman과 Girvan에 의해 제안되었다. 이 방법은 Network에서 Edge를 반복적으로 제거하는 계층적 분할 Clustering 방법이다. 제거되는 Edge들은 중간성-측정(betweenness measures)을 사용하여 선택된다. 간선 중간성의 아이디어는 두 커뮤니티가 소수의 커뮤니티 간 Edge로 연결되어 있을 때, 한 커뮤니티의 정점에서 다른 커뮤니티의 정점으로 가는 모든 경로가 이 Edge들을 통과해야 한다는 관찰에서 나온다. Edge를 통과하는 모든 경로를 세고, 가장 높은 값을 기록한 간선을 제거함으로써 네트워크 내의 연결이 끊어진다. 이 과정은 네트워크가 더 작은 구성요소로 나뉠 때까지 반복되며, 분할을 멈추는 기준은 Modularity이다. Modularity은 주어진 k개의 커뮤니티에 대해 다음과 같이 정의된다. eij 는 네트워크 내에서 그룹i 의 정점과 그룹 j 의 정점을 연결하는 간선의 비율이며, ai = ∑ j eij 이다. 그러면 M = ∑i (eii − a2i ) 는 커뮤니티 내부의 간선 비율에서 무작위로 연결되었을 때 예상되는 간선 비율을 뺀 값이다. 1에 가까운 값은 강한 Community 구조를 나타낸다. 따라서 이 알고리즘은 계층적 접근을 적용하여 얻은 모든 Cluster의 Modularity를 계산하고, 가장 높은 Modularity 값을 가진 Clustering을 결과로 반환한다[5][10][11][12].

Radicchi et al.은 커뮤니티(community)에 대한 두 가지 양적(quantative) 정의와 커뮤니티(community)를 식별하는 알고리즘을 제안한다. 커뮤니티의 양적 정의는 노드의 정도(degree)에 기반한다. 강한 의미에서의 커뮤니티는 각 노드가 그래프의 나머지 부분보다 커뮤니티 내에 더 많은 연결을 가지고 있는 부분 그래프다. 약한 의미에서의 커뮤니티는 V(vertex) 내의 모든 입도(in-degrees)의 합이 출도(out-degrees)의 합보다 클 때의 부분 그래프다. 이 알고리즘은 간선 클러스터링 계수(edge-clustering coefficient) 개념에 기반한 계층적 분할 방법이다. 이는 인접 노드의 정도를 고려하여, 간선이 참여하는 삼각형의 수를 해당 간선이 속할 수 있는 삼각형의 수로 나누어 정의된다. 이 알고리즘은 Newman과 Girvan의 방법과 유사하게 작동하지만, 가장 높은 간선 중간성을 가진 간선을 제거하는 대신 가장 낮은 간선 클러스터링 계수 값을 가진 간선을 제거한다. [13][14].

유전 알고리즘(GA : Genetic Algorithm)을 기반으로 한 Community Detection 접근법은 [9,15]에서 찾을 수 있다. [9]에서 저자들은 Newman과 Girvan에 의해 제안된 네트워크 Modularity를 Fitness 함수로 사용하는 유전 알고리즘(GA – Genetic Algorithm)을 제시한다. 개체는 N개의 유전자로 구성되며, N은 객체의 수다. i번째 유전자는 i번째 노드에 해당하며, 그 값은 노드 i의 커뮤니티 식별자다. [15]에서는 소셜 네트워크를 Clustering하기 위해 유전 알고리즘(Genetic Algorithm)에 통합된 random walk distance measure측정 방법이 설명되어 있다. 이들이 사용하는 표현은 k-medoids이며, 여기서 각 Cluster 중심은 네트워크의 한 노드에 의해 표현된다. 물론 이는 Cluster의 수 k가 사전에 알려져 있어야 함을 의미한다. Fitness 함수는 노드 간의 모든 pair-wise 거리의 합을 최소화하려고 시도한다.

Tasgin et al.의 연구에서는 chromosome들은 Partition으로 정의되고 Modularity가 적합성 함수로 사용된다. 염색체의 수, Mutation 및 Crossover Rate과 같은 적절한 매개 변수 선택을 통해, Tasgin et al.은 벤치마크 네트워크에서 Greedy Modularity Optimization와 비교할 수 있는 품질의 결과를 얻을 수 있었다[9].

Jicun Zhang et al.의 연구에서는 Community Detection을 위하여 Louvain Algorithm에 대해서 소개하고 있다[2].

Louvain Algorithm의 주요 단계는 다음과 같다:

1단계:

Community를 초기화하고 각 노드를 별도의 Community로 설정한다. 예를 들어, 커뮤니티 1: (노드1), 커뮤니티 2: (노드2) 등이다.

2단계:

Node 1과 연결된 모든 Community를 찾아 Node 1을 각 이웃 커뮤니티(neighbor. community)로 이동시킨 후 Modularity의 변화를 계산한다. Modularity를 최대한 증가시킬 수 있는 커뮤니티로 Node 1을 이동시킨다.

3단계:

모든 Node에 대해 반복하여 2단계를 실행하고, 이동할 노드가 없을 때까지 계속하여 한 층의 커뮤니티 분할을 얻는다.

4단계:

3단계에서 각 한Community를 새로운 Node로 병합한다. 새 Node 간의 관계는. 원래 Community 간의 관계다. 모든 노드가 하나의 Community로 최종 병합될 때까지 1단계로 돌아간다. 여러 레벨의 커뮤니티 분할이 얻어지며, 가장 높은 모듈성을 가진 분할을 최종 결과로 선택한다.

Fast Newman Algorithm과 비교할 때, Louvain Algorithm은 반복 당 계산 횟수가 적다는 장점이 있다. 한편, 단일 노드의 이동을 계산할 때 이동이 발생하는 두 커뮤니티에서 노드의 Modularity 이득만 계산하면 되므로, 계산 효율성이 빠른 Newman Algorithm보다 뛰어나다. 이 알고리즘의 시간 복잡도는 대략 o(m + n)이다[2][16].

# Methodology

Genetic Algorithm:

Genetic Algorithm은 1970년대에 J. Holland에 의해 소개된 자연 선택과 자연 유전학의 개념을 기반으로 하는 search 알고리즘이다[17]. Genetic Algorithm은 초기 무작위로 생성 된population에서 시작하여 더 좋은 품질의population으로 세대를 반복적으로 거듭하면서 진화시킨다 [18]. 좋은 품질은 genetic algorithm을 적용할 문제에 적합한 fitness Score 결과 값에 의해 결정된다[18]. GA는 각 세대마다, 세 가지 기본적인 연산, Selection, Crossover, Mutation을 진행한다[18]. GA population을 대체하는 비율에 따라 Steady-State GA, Generational GA 로 구분된다.

Community Detection을 하기 위해 한 세대가 population 내 가장 낮은 fitness 값을 가진 chromosome을 20개 대체한다.

Pseudo code:

Generate n random initial solution and generate n solution using Louvain local optimization;

repeat {

for i <- 1 to k {

choose two parent chromosomes and ; // tournament selection

offspring <- crossover (, ); // one- or two-point crossover

offspring <- mutation (offspring);

offspring <- Louvain\_local\_optimization(offspring);

}

Replace offspring with k chromosomes in the population;

} until (stop condition);

return the best in the population;

스크린샷, 텍스트, 디자인이(가) 표시된 사진

자동 생성된 설명

Figure 1. Flowchart of GA System

Representation:

Community Detection을 위한 그래프를 표현하기 위해 partition 배열을 생성한다. partition 배열의 크기는 그래프가 가진 노드의 개수이다. partition 배열의 인덱스 i는 그래프의 노드 번호를 의미하며, partition[i]의 값, j는 해당 노드가 소속된 community를 의미한다. j의 범위는 최소 커뮤니티 개수와 최대 커뮤니티 개수의 사이이며, 이는 데이터셋으로 사용하는 그래프마다 적절한 개수를 정의해 사용한다.

Evaluate fitness:

GA는 좋은 solution인지 비교하기 위해 fitness score를 도입해서 사용한다. 네트워크의 community detection은 edge 배열과 상관관계가 있는 modularity 측정 값을 사용하여 정량화 할 수 있다 [19]. 따라서, population내 chromosome의 비교를 위해 modularity 측정 값을 fitness score로 사용한다.

Modularity는 edge가 무작위로 배치된 그래프에서 community에 속하는 edge의 수에서 예상되는 수를 뺀 값이다[19]. Modularity를 구하는 식은 다음과 같다.

Q: modularity

M: 전체 링크 수

N: 전체 노드 수

: i, j간 링크 (있을 경우 1, 없을 경우 0)

: 각 노드가 지니는 링크의 개수는 그대로 유지, 대상을 무작위적으로

재연결 하였을 때 노드 i, j 간 링크(있을 경우 1, 없을 경우 0)

<>: t\_ij의 기대치

C(i): 노드 i가 속하는 커뮤니티

δ[C(i), C(j)]: C(i)와 C(j)가 같은 커뮤니티일 때 1, 다를 때 0

Initializing Population:

GA는 초기 population을 무작위로 생성해 GA를 진행한다. 그러나 community detection 데이터의 크기가 큰 경우, 최적의 solution을 얻는데 많은 시간이 걸리고, 최적의 solution을 쉽게 찾지 못했다. 이런 문제를 해결하기 위해 초기 population에 local optimization을 적용하여 생성된 chromosome을 일부 대체하여 진행한다. Population의 25%를 local optimization 이 적용된 chromosome으로 대체하여 진행한다.

Selection:

Selection Operator는 offspring chromosome을 생성하기 위해 population에서 좋은 두 개의 parents chromosome을 선택하는 과정이다[18]. Selection Operator는 roulette wheel selection, tournament selection, rank-based selection, elitism selection 등이 있다[18]. 이 중 tournament의 크기가 1인 tournament selection를 사용한다. 일반적인 tournament selection은 일정 확률을 정의하여 더 안 좋은 parent chromosome 역시 선택되게 하지만, 해당 GA는 성능적인 부분을 고려하여, 항상 더 좋은 parent chromosome이 선택된다. chromosome들이 저장된 population 배열에서 무작위 난수를 생성해 두 개의 chromosome을 선택하여 더 높은 fitness 값을 가진 chromosome이 선택되며, 이 과정을 두 번 실행해 parent chromosome 가 선택된다.

Crossover:

Crossover operator는 GA의 성공 여부가 결정되는 중요한 operator 중 하나이다[2]. Crossover operator는 Selection Operator로 선택된 parent chromosome인 를 offspring을 만들기 위해 재조합 하는 과정이다. Crossover operator의 종류로는 one point crossover, two point crossover, uniform crossover 등이 있다[2]. 이 중 two point crossover를 사용해 두 개의 offspring을 생성한다.

무작위 난수, 추출하여 selection operator로 선택된 를 재조합해 두 개의 offspring을 만든다. 하나의 offspring은 의 0번째 인덱스부터 인덱스까지, 의 인덱스부터 인덱스까지, 의 인덱스부터 마지막 인덱스까지 재조합해 생성한다. 다른 offspring은 의 0번째 인덱스부터 인덱스까지, 의 인덱스부터 인덱스까지, 의 인덱스부터 마지막 인덱스까지 재조합해 생성한다.

Mutation:

Crossover operator가 두 개의 parent chromosome을 재조합해 offspring을 만드는 반면, Mutation operator는 chromosome의 일부분을 mutation rate를 가지고 무작위로 변경한다 [19]. Mutation Operator는 Steady-State GA 실행으로 인해 발생하는 premature convergence의 속도를 늦춰줄 수 있다. Mutation operator는 정의된 mutation rate의 의거해, 각각의 인덱스의 정의된 커뮤니티들을 정의된 최대 community 크기 범위 내로 무작위로 변경한다. Mutation rate와 최대 community 크기는 해당 데이터 셋에 맞게 적절히 변경하여 사용한다.

이후, local optimization을 적용하고, 최소 fitness score를 가진 chromosome과 offspring을 대체한다.

Local optimization:

그래프에서의 Global search는 community detection을 하기 위해서 전체 그래프를 모두 탐색할 필요가 있다[20]. 이는 그래프의 크기가 커진다면 사용할 수 없는 방법이다[4]. 이 의미는 현실에 존재하는 네트워크를 그래프로 표현한 것에는 적용하기 어렵다는 것을 의미한다. 따라서, 현실에 존재하는 네트워크 그래프를 community detection 하기 위해선 local optimization 과정이 필요하다.

Local optimization 과정은 offspring에 mutation을 적용한 후, community detection을 하기 위한 알고리즘 중에 하나인 Louvain Algorithm을 적용하여 진행한다. Local optimization을 통해 큰 규모의 그래프에서도 community detection을 가능하게 하고, community detection 하는 시간을 효과적으로 줄일 수 있다.

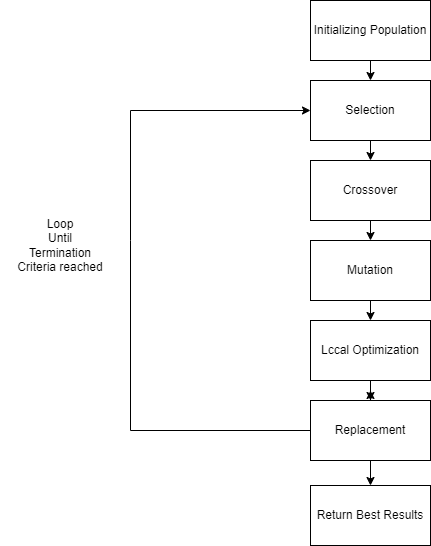


Figure 2. Flowchart of Project

# Experiment Setup

본 연구에서 사용된 Data는 총 3 가지이다. Nodes : 34이고 Edges : 78인 상대적으로 작은 크기의 데이터인 Zachary karate club[21], Nodes : 115이고 Edges : 613인 American Football[22], Nodes : 4039이고 Edges : 88234인 상대적으로 큰 크기의 데이터인 Facebook[23] 데이터를 사용하였다.

Zachary Karate Club 데이터는 1977년 Wayne Zachary가 대학 Karate Club 회원으로부터 수집한 것이다. 각 노드는 클럽 회원을 나타내고 각 가장자리는 클럽 회원 두 명 간의 연결을 나타낸다. 네트워크 방향은 지정되지 않았고(Undirected) 이 데이터 세트를 사용하여 자주 논의되는 문제는 교사 간의 논쟁 후 Karate Club이 분할된 그룹의 사람들을 찾는 것이다.

American football 데이터는 American football games between Division IA colleges during regular season Fall 2000에 관한 데이터로 각 Node는 대학 축구팀을 나타내고, Edge는 두 팀 간의 경기 결과를 나타낸다.

Facebook 데이터는Facebook의 친구목록에 관한 데이터로 Node는 사용자를 나타내고, Edge는 노드들의 관계를 나타낸다. 두 노드가 친구상태일 경우 Edge가 형성된다.

문제를 해결하기 위해 본 연구에서 사용한 알고리즘 기법은 Genetic algorithm, Louvain algorithm, Local optimization algorithms등을 결합한 Hybrid Genetic Algorithm을 사용하였다. 초기 파티션을 Louvain Algorithm을 사용하여 구성 후 Genetic Algorithm을 사용해 선택(Selection), 교차(Crossover), 돌연변이(Mutation) 등의 과정을 통해 더 좋은 값을 찾아가고 Local Optimization Algorithm을 사용해 일정한 빈도마다 Louvain Algorithm을 사용한 초기해를 기반으로 지역 최적화를 수행한다.

Population\_size는 Genetic algorithm에서 각 Generation에서 생성되는 커뮤니티 파티션(Community Partition)의 수로 값이 높을수록 더 많은 다양성을 제공하지만 계산 비용이 증가할 수 있다. Generation은 알고리즘의 총 반복 횟수로 generation이 진행될수록 해결책이 점진적으로 개선될 수 있다. Crossover는 두 부모 커뮤니티 파티션을 결합하여 새로운 파티션을 생성하는 데 사용된다. 값이 높을수록 부모들이 파티션을 교차하는 가능성이 높아져 다양성을 유지할 수 있다. Mutation는 개별 Partition 내의 커뮤니티를 무작위로 변경하는데 사용된다. 값이 낮으면 다양성이 감소할 수 있으나, 높은 값은 해결책의 질을 떨어뜨릴 수 있어 적절한 값을 적용해야 한다. min\_partition\_size, max\_partition\_size는 최적화 과정에서 생성된 파티션의 크기를 제한하는 데 사용된다. 이를 통해 원하는 크기의 커뮤니티가 유지되도록 하거나, 해결책의 다양성을 유지할 수 있다. 앞의 파라미터의 경우 Genetic algorithm을 구현하는데 필수적인 파라미터이다. local\_optimization\_frequency는 최적화 과정에서 파티션을 지역적으로 개선하기 위한 빈도를 나타낸다. 이는 Louvain algorithm을 사용하여 얻은 초기 해결책을 개선하기 위해 사용되었다.

Zachary karate club 데이터에 코드를 적용시켰을 때 파라미터 값을 Population\_size 50, generation 100, Crossover 1.0, mutation 0.01, min\_partition\_size 2, Max\_partition\_size 5, local\_optimization\_frequency 10으로 설정하였다. 데이터 자체의 크기가 작기 때문에 빠른 시간 내에 수렴할 것으로 예상되었다.

|  |  |
| --- | --- |
| **Zachary\_karate\_club Parameters** | |
| **Population\_size** | **50** |
| **generation** | **100** |
| **Crossover** | **1.0** |
| **mutation** | **0.01** |
| **min\_partition\_size** | **2** |
| **Max\_partition\_size** | **5** |
| **local\_optimization\_frequency** | **10** |

American football 데이터에 코드를 적용시켰을 때 파라미터 값을 Population\_size 200, generation 500, Crossover 1.0, mutation 0.05, min\_partition\_size 5, Max\_partition\_size 15, local\_optimization\_frequency 5로 설정하였다. 데이터의 크기가 크지는 않지만 population\_size가 200에 generation이 500임으로 코드 자체의 시간은 오래 걸릴 것으로 예상되었다. Zachary karate club에서 너무 빨리 수렴할 경우 코드에 이상이 있는 걸 확인하기 어렵고 진행 과정을 알기 어렵겠지만 American football로 진행 과정과 코드 점검이 가능해질 것을 기대할 수 있었다.

|  |  |
| --- | --- |
| **American\_football Parameters** | |
| **Population\_size** | **200** |
| **generation** | **500** |
| **Crossover** | **1.0** |
| **mutation** | **0.05** |
| **min\_partition\_size** | **5** |
| **Max\_partition\_size** | **15** |
| **local\_optimization\_frequency** | **5** |

Facebook 데이터에 코드를 적용시켰을 때 파라미터 값을 Population\_size 200, generation 500, Crossover 1.0, mutation 0.01, min\_partition\_size 5, Max\_partition\_size 15, local\_optimization\_frequency 5로 설정하였다. 앞서 Zachary karate club과 American football 데이터로 실행시켜 본 결과 코드에 문제점이 없다는 것을 확인할 수 있었고 이를 통해 데이터 크기가 큰 facebook 데이터를 기반으로 작동하였을 때 문제가 없을 것이라 예상할 수 있었다. 다만 Population\_size 값과 generation값이 높고 데이터 크기가 커 시간은 가장 오래 걸릴 것이라 예상할 수 있었다.

|  |  |
| --- | --- |
| **Facebook Parameters** | |
| **Population\_size** | **200** |
| **generation** | **500** |
| **Crossover** | **1.0** |
| **mutation** | **0.01** |
| **min\_partition\_size** | **5** |
| **Max\_partition\_size** | **15** |
| **local\_optimization\_frequency** | **5** |

본 연구에서 사용된 장비의 성능은 다음과 같다

|  |  |
| --- | --- |
| **장비의 성능** | |
| **프로세서** | **13th Gen Intel(R) Core(TM) i5-13500 2.5 GHz** |
| **RAM** | **32GB** |
| **디바이스** | **NVIDA GeForce RTX 3060** |

소프트웨어로는 Jupyter Notebook으로 Python을 사용하였다.

# Result

각 데이터셋의 Louvain 알고리즘에 대한 Modularity값을 Figure3의 표로 나타내었다.

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Karate Club | American\_football | Facebook |
| Modularity | 0.44385412567 | 0.60418368731853 | 0.8349793603940 |

Figure 3. Modularity value of each dataset

앞서 설명한 Hybrid GA를 이용하여Karate Club, American Football, Facebook 데이터셋에 적용한 결과를 Figure4, Figure5, Figure6의 표로 나타내었다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G |
| Population\_size | 50 | 50 | 50 | 50 | 50 | 50 | 50 |
| Generation | 100 | 100 | 100 | 100 | 100 | 200 | 200 |
| Mutation\_prob | 0.01 | 0.01 | 0.01 | 0.05 | 0.05 | 0.01 | 0.05 |
| CrossOver | one\_point | one\_point | two\_point | one\_point | two\_point | two\_point | two\_point |
| Community\_Size | 5-15 | 5-15 | 5-15 | 5-15 | 5-15 | 5-15 | 5-15 |
| Init | 0% | 25% | 25% | 25% | 25% | 25% | 25% |
| Avg | 0.4412927 | 0.4291323 | 0.4430020 | 0.4215863 | 0.4428286 | 0.4439892 | 0.4444283 |
| Max | 0.4449035 | 0.4449035 | 0.4449035 | 0.4449035 | 0.4449035 | 0.4449035 | 0.4449035 |

Figure 4. Karate Club Dataset Hybrid GA Result

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G |
| Population\_size | 200 | 200 | 200 | 200 | 200 | 200 | 200 |
| Generation | 500 | 500 | 500 | 500 | 500 | 1000 | 1000 |
| Mutation\_prob | 0.01 | 0.01 | 0.01 | 0.05 | 0.05 | 0.01 | 0.05 |
| CrossOver | one\_point | one\_point | two\_point | one\_point | two\_point | two\_point | two\_point |
| Community\_Size | 5-15 | 5-15 | 5-15 | 5-15 | 5-15 | 5-15 | 5-15 |
| Init | 0% | 25% | 25% | 25% | 25% | 25% | 25% |
| Avg | 0.5942273 | 0.5129481 | 0.5980024 | 0.5781133 | 0.6038408 | 0.6010421 | 0.6021836 |
| Max | 0.6045695 | 0.5706910 | 0.6045695 | 0.6045695 | 0.6045695 | 0.6045695 | 0.6045695 |

Figure 5. American\_football Dataset Hybrid GA Result

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G |
| Population\_size | 200 | 200 | 200 | 200 | 200 | 200 | 200 |
| Generation | 500 | 500 | 500 | 500 | 500 | 1000 | 1000 |
| Mutation\_prob | 0.01 | 0.01 | 0.01 | 0.05 | 0.05 | 0.01 | 0.05 |
| CrossOver | one\_point | one\_point | two\_point | one\_point | two\_point | two\_point | two\_point |
| Community\_Size | 5-15 | 5-15 | 5-15 | 5-15 | 5-15 | 5-15 | 5-15 |
| Init | 0% | 25% | 25% | 25% | 25% | 25% | 25% |
| Avg | 0.8264841 | 0.8226007 | 0.8244438 | 0.7912876 | 0.8347456 | 0.8262464 | 0.8339266 |
| Max | 0.8357896 | 0.8350447 | 0.8357999 | 0.8358139 | 0.8358409 | 0.8358007 | 0.8358409 |

Figure 6. Facebook Dataset Hybrid GA Result

Figure4, Figure5, Figure6에서 공통적으로 Louvain 알고리즘보다 조금 더 향상된 결과를 보인다. Stady-State GA는 한 세대에서 일부 개체만 교체하는 특성 때문에 Generational GA보다 높은 Mutation Probability을 적용했을 때 지역 최적화에 갇힐 가능성이 낮다.[24]

Two-Point Crossover연산은 One-Point Crossover연산에 비해 두 지점 간의 부분 교차를 통해 다양한 Offspring을 생성할 수 있다.[25] 이러한 특성들 때문에 더 좋은 결과가 도출되었다고 예상되었다.

Hybrid GA의 실행 시간을 측정하였을 때

|  |  |
| --- | --- |
| 실행시간 | |
| Karathe\_club | 7s |
| American\_football | 130s |
| Facebook | 25656s |

이러한 결과가 나왔다. Karate\_club 의 경우 데이터셋의 크기가 작아 7s 가 경과된 반면 Football 의 경우 130s 가 경과된 것을 확인해 볼 수 있었다. Facebook 의 경우 25656s 으로 다른 결과들에 비해 상대적으로 높다는 것을 확인해 볼 수 있었다. 이를 통해 데이터의 크기가 클수록 기하급수적으로 시간이 오래 걸린다는 결과를 도출할 수 있었다.

Louvain의 알고리즘 실행 시간을 측정하였을 때

|  |  |
| --- | --- |
| 실행시간 | |
| Karathe\_club | 0.003s |
| American\_football | 0.012s |
| Facebook | 2.867s |

이러한 결과가 나왔다. Hybrid GA에 비해 실행 시간이 매우 짧은 것을 볼 수 있다.

# Conclusion & Limitations

위에서 언급했듯이, 대규모 네트워크에서Community Detection은 NP-Hard Problem으로 간주하며, Heuristic 기반 알고리즘은 NP-Hard 문제에 효율적인 것으로 입증되었다. [1] 이러한 이유로, 본 연구에서는 Modularity값을 기준으로, 3가지 데이터셋을 사용하여Louvain 알고리즘을 활용한 휴리스틱 초기화(Heuristic Initialization) 및 Hybrid GA를 이용하여 Community Detection문제를 해결하고자 했다.

기존 Louvain 알고리즘에 대한 비교 분석을 통하여 아래와 같은 결과를 도출하였다.

Hybrid GA가 Louvain 알고리즘에 비해 값이 소폭 향상되었지만, 실행 시간이 매우 오래 소요된다는 단점이 있다. 이는 Heuristic Algorithm에서 매우 큰 단점으로 작용한다. [26]

추후 Hybrid GA의 실행시간 단축과 성능 향상은 필수적이다. 따라서 데이터셋의 Preprocessing 및Modularity 연산 과정 최소화를 통한 실행 시간 단축, Simulated Annealing 적용 및 repair을 통한 성능 향상에 관한 연구들을 진행할 예정이다.

# Reference

[1] Genetic algorithm-based community detection in large-scale social networks , Ranjan Kumar Behera1 • Debadatta Naik2 • Santanu Kumar Rath1 • Ramesh Dharavath1

[2] An Improved Louvain Algorithm for Community Detection , Jicun Zhang ,1,2 Jiyou Fei ,1 Xueping Song ,1 and Jiawei Feng 2

[3] Community detection in graphs , Santo Fortunato

[4] Community Detection In Social Networks, Punam Bedi, Chhavi Sharma,

Department of Computer Science, University of Delhid

[5] GA-Net: A Genetic Algorithm for Community Detection in Social Networks, Clara Pizzuti

[6] Community detection based on modularity and an improved genetic algorithm , Ronghua Shang ∗, Jing Bai, Licheng Jiao, Chao Jin

[7] A Genetic Algorithm for Modularity Density Optimization in Community Detection , Ali Ghorbanian \*, Bahman Shaqaqi

[8] Fast unfolding of communities in large networks, Vincent D. Blondel1;a, Jean-Loup Guillaume1,2;b, Renaud Lambiotte1,3;c and Etienne Lefebvre1

[9] M. Tasgin, A. Herdagdelen, and H. Bingol, “Community detection in complex networks using genetic algorithms,”

[10] Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proc. National. Academy of Science.

[11] Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E

[12] Dubes, R.C., Jain, A.K.: Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs

[13] Radicchi,F.,Castellano,C.,Cecconi,F.,Loreto,V.,Parisi,D.:Defining and identifying communities in networks.

[14] Newman, M.E.J.: The structure and function of complex networks. SIAM Review 45, 167– 256 (2003)

[15] Firat, A., Chatterjee, S., Yilmaz, M.: Genetic clustering of social networks using random walk. Computational Statistics and Data Analysis 51

[16] S. Ghosh, M. Halappanavar, A. Tumeo et al., “Distributed louvain algorithm for graph community detection,” in Pro- ceedings of the 2018 IEEE International Parallel and Dis- tributed Processing Symposium (IPDPS) IEEE Computer Society

[17] Genetic Algorithm, TV Mathew - Report submitted at IIT Bombay, 2012 - datajobstest.com

[18] A study on genetic algorithm and its applications, L Haldurai, T Madhubala, R Rajalakshmi - Int. J. Comput. Sci. Eng, 2016 - academia.edu

[19] Modularity and community structure in networks,

MEJ Newman - Proceedings of the national academy of …, 2006 - National Acad Sciences

[20] Local search of communities in large graphs

W Cui, Y Xiao, H Wang, W Wang - Proceedings of the 2014 ACM …, 2014 - dl.acm.org

[21] UCI Network Repository. Zachary's karate club network. -http://konect.cc/networks/ucidata-zachary/

[22] DIMACS (Discrete Mathematics and Theoretical Computer Science) Network Data. Football network. -http://konect.cc/networks/dimacs10-football/

[23] Stanford Large Network Dataset Collection. Facebook dataset. -https://snap.stanford.edu/data/egonets-Facebook.html

[24] A Steady-State and Generational Evolutionary Algorithm for Dynamic Multiobjective Optimization

Shouyong Jiang; Shengxiang Yang

[25] A formal analysis of the role of multi-point crossover in genetic algorithms

Kenneth A. De Jong & William M. Spears

[26] Experimental Evaluation of Heuristic Optimization Algorithms: A Tutorial

Ronald L. Rardin & Reha Uzsoy