커뮤니티 검출기법을 이용한 소프트웨어 아키텍쳐 모듈 뷰 복원†

(Recovering Module View of Software Architecture using Community Detection Algorithm)

김정민‡ 이찬근§
(Jungmin Kim) (Changun Lee)

요 약 본 논문은 소프트웨어 클러스터링 기법과 커뮤니티 검출 기법의 비교를 통하여 아키텍쳐 모듈 복원 프로세스에 커뮤니티 검출 알고리즘의 적용가능성을 제시한다. 또한, 대표적인 클러스터링 알고리즘과 커뮤니티 검출 알고리즘의 값과 나눠진 모듈간의 상관관계와 차이점을 분석한다. 이를 통하여 커뮤니티 검출 알고리즘이 소프트웨어 아키텍쳐 모듈 뷰 복원에 활용되어질 수 있다는 몇 가지 근거를 제시하였고, 기존의 클러스터링 결과와 커뮤니티 알고리즘의 결과치를 비교함으로써, 서로의 결과 데이터가 어떠한 연관성을 가지는지 제시하였다.

키워드 소프트웨어 모듈 뷰, 커뮤니티 스트럭쳐, 소프트웨어 클러스터링

Abstract This article suggests applicability to community detection algorithm from module recovering process of software architecture through compare to software clustering metric and community dectection metric. in addition to, analyze mutual relation and difference between separated module and measurement value of typical clustering algorithms and community detection algorithms. and then only sugeested several kinds basis that community detection algorithm can use to recovering module view of software architecture and, by so comparing measurement value of existing clustering metric and community algorithms, this article suggested correlation of two result data.

Key words : Software module view, Community Structure, Software Clustering

1. 서 론

네트워크의 현대적인 연구는 복잡계 네트워크의 융합 학문분야에 접목되어지는 과학에 집중되어 있다.[3] 많은 복잡계 네트워크 시스템들은 네트워크의 모양으로 나타낼 수 있고, 시스템의 최소 단위를 이루는 부분들로 이루어지는데, 그러한 네트워크는 노드와 링크를 통해 서로의 상호 연관성을 만들어 낸다. 이와 관련하여 복잡계 네트워크 안의 노드들은 한 모듈 안의 내부 링크끼리의 강한 밀집도를 통하여 노드들의 집합의 모양으로 나타내어 지는데, 반대로 다른 모듈간의 이어진 링크의 밀집도를 최소화 시켜서 각 모듈간의 독립성을 최대화 시키는 것이 연구의 핵심이다. 여기에서의 네트워크를 그래프로 표현할 때, 그래프 안의 내부 링크간의 밀집도가 높게 구성되어진 서브그래프를 모듈(클러스터링 기법) 또는 커뮤니티(커뮤니티 검출 기법)라 칭하게 된다.

복잡계 네트워크의 노드들을 구분하는 방법을 찾는 것은 그 시스템의 특성과 그들의 규칙에 의존

† 본 연구는 한국연구재단 기초연구사업(과제번호 2010-0015636, 2011-0013924)의 지원을 받았습니다.

‡ 학생회원 : 중앙대학교 컴퓨터공학부, jungmink26@lycos.co.kr

§ 종신회원 : 중앙대학교 컴퓨터공학부 교수, cglee@cau.ac.kr

하게 되는데, 그러므로 네트워크상의 커뮤니티들을 찾는 것은 네트워크 과학의 가장 기본적인 문제이다.

많은 방법들이 이미 제시되었고, 측정 도구는 물리학, 생물학, 그래프 이론에서의 수학적인 접근 그리고 컴퓨터와 소셜과학의 분야를 통틀어 활용되어지고 있다. 현재 네트워크(그래프)의 모듈화의 방법은 위에서 말했듯이 두 가지 접근 방법으로 설명이 가능한데, 그 두 기법은 약간의 차이가 있다.[1]

첫 번째로 소프트웨어 클러스터링의 주된 목적은 이미 개발자에 의해 구축되어진 아키텍쳐를 비교군으로 잡아, 그 소프트웨어 구조를 알고리즘으로 하여금 분류(군집)하게 하여 기존 아키텍쳐와 알고리즘을 통한 모듈화 결과의 일치성을 비교한다. 이는, 좋은 알고리즘이 소프트웨어 설계구조를 정의할 수 있다는 것으로 해석될 수 있으며, 이를 위해 몇 가지 솔루션을 연구 중에 있다.

두 번째로 커뮤니티 검출 기법은 “척도없는 네트워크”라는 소셜네트워크를 기반으로 한 네트워크 구조상에서의 적절한 커뮤니티를 나누고 그 나눠진 커뮤니티의 모듈성 측정을 목적으로 한다. 커뮤니티 검출의 경우 기본 컨셉이 매우 많은데, 각 노드의 연결관계를 통해 특정 노드의 영향력을 나타낸다거나, 네트워크의 특성에 따른 노드와 간선간의 의미를 어떻게 정의하느냐에 따라서 커뮤니티를 검출하는데에 큰 영향력을 끼친다고 할 수 있다. 하지만 기본적인 컨셉(네트워크의 특성)은 그래프 생성 단계에서 그래프의 구조적 방법으로 정의되기 때문에 알려진 몇몇 대표적인 알고리즘들이 존재한다.

다음으로는 이와 관련한 본 논문의 실험의 접근 방법 및 실험내용과 결과를 알아보도록 하겠다.

2. 접근 방법

커뮤니티 검출 기법의 본래 목적은 복잡계 네트워크상에서의 각 정점들끼리의 강한 연결 요소를 찾아내어 서로 다른 커뮤니티끼리의 독립성을 보장하는 데에 있다.

본 연구의 접근 방법은 이러한 커뮤니티 검출 기법이 실제 소프트웨어 아키텍쳐 모듈 복원에 활용되어 질 수 있는지를 알아본다. 이 실험을 통해 기존 클러스터링 알고리즘의 접근 방법(closeness, centrality)등에 비해 커뮤니티 검출 알고리즘(betweenness, modularity) 최대화 기법이 어느 정도의 효율을 보일 수 있고, 그 효과가 기존 알고리즘 등과 비교했을 때 적용가치가 있는지 등을 알아보겠다. 기본 동기는 현재 클러스터링 연구 분야가 기존 네트워크의 구조적 특성이나 노드의 각각의 특정 수치에 의존하여 모듈을 나눔으로써 의미를 파악하는데에 집중하고 있는 점을 비추어 볼때, 순수하게 노드의 영향력적인 가치를 반영하는 커뮤니티 검출방법이 이러한 모듈 뷰 복원 일치성을 높이는 노력에 도움이 될 수 있다 생각했기 때문이다.

3. 연구 내용

검출 데이터는 기본적으로 클래스 의존 관계와 우수한 package 모듈을 가지고 있는 자바 라이브러리를 사용하였다. 모든 데이터는 기본적으로 50개 이상의 클래스로 구성된 중/대규모 패키지이며, 이는 복잡계 네트워크를 표현할 수 있는 규모를 가지고 있다고 판단하였다. 자바 패키지가 척도없는 네트워크의 특성인 멱함수 분포를 보인다는 것은 이미 증명되어 있다. 이러한 실험을 진행하기 위한 추출되어질 라이브러리는 다음과 같다.

표 1 검출 데이터

검출 데이터	설 명
javax.swing	javax swing package
java org	java org library
com imageio	java com library
java.util	java util package

3.1. 추출 알고리즘

기존 클러스터링 기법과 커뮤니티 검출 알고리즘 2가지를 사용하여 같은 데이터에 대한 아키텍쳐 복원율을 비교한다. 대표적인 클러스터링 알고리즘으로서는 WeakComponentClusterer(WC) 기법을 활용할 것이다.[6]

커뮤니티 검출 알고리즘으로써 사용한 기법은 Girven-Newman의 EdgeBetweenness(EB), Modularity maximization(MM)이다. 이 두 알고리즘은 서브그래프를 추출하는 점에서는 클러스터링 알고리즘과 같으나, 커뮤니티를 검출한다는 점이 차이점이다.

3.2. 소프트웨어 아키텍쳐 복원율 측정

실험은 크게 두 가지로 나뉜다. 첫째로 클러스터링 계수측정을 통하여 커뮤니티 검출 알고리즘의 효용성을 판단하고 두 번째로, 실제 커뮤니티 검출 알고리즘이 소프트웨어 모듈 뷰 복원율 향상에 어느 정도 효과가 있는지를 알아볼 것이다. 실험 순서는 다음과 같다.

(1) 주어진 DataSet의 클래스 의존 그래프를 추출하고 커뮤니티 검출 알고리즘을 통해 DataSet의 모듈성을 측정한다.
(2) 각 알고리즘들에 관하여 클러스터링 계수를 측정해 본다.
(3) 클러스터링 알고리즘과 커뮤니티 검출 알고리즘을 이용하여 그래프를 모듈화 시킨 후, 그 결과를 MoJo알고리즘을 이용하여 기존 소프트웨어의 패키지구조와 비교한다.[5]
(4) 그 결과를 토대로 (1)번에서의 모듈성 측정값과, 복원율 측정값의 비례여부를 조사한다.

3.4. 실 험

3.4.1. 각 데이터의 클래스를 그래프화하여 정점과 간선의 구조로 나타내었다.

표 2 클래스 의존 그래프의 각 패키지 정보

데이터	정점수	간선수	방향성
javax.swing	2021	5874	없음
java org	767	373	없음
com imageio	211	726	없음
java.util	772	1239	없음

각 데이터를 살펴보면, swing 패키지는 매우 많은 수의 정점과 간선의 수가 존재한다. 그러므로 알고리즘 수행 시간이 오래걸리는 반면 가장 신뢰성 있는 결과를 기대할 수 있다. util 패키지는 그보다 약간 소규모 패키지로써, 정점수가 간선수보다 적다. org패키지와 com패키지는 서로 대조적인 정점수와 간선수를 보여주고 있다.

위의 클래스 의존 그래프를 커뮤니티 검출 알고리즘을 통해 모듈화 하였다. modularity는 0부터 1까지의 값으로써, 각각의 커뮤니티가 얼마나 서로 독립적인지를 수치로 나타내어 준다. 만약 나눠진 모듈안의 모든 정점들이 서로 다른 모듈의 정점들과 연결되는 간선이 없다면 그 측정치는 1이 되며, 완전 그래프의 경우에는 측정치가 0이 된다. 아래의 표는 검출 알고리즘 시행 후, 그 결과이다.[7]

org, util 패키지는 상대적으로 높은 모듈성을 가진것과 달리 swing, imageio 패키지는 낮은 모듈

성을 보이고 있다. 이는 패키지의 노드 대비 간선수와 비례한다고 볼 수 있는데, 간선이 많다는 것은 그만큼 전체 클래스 의존성이 서로 얽혀 있어서 독립적인 모듈이 나오기 힘들다는 것을 의미한다. 유사실험 논문에서 모듈성 측정값을 0.5이상을 기준으로 좋은 결과라 생각하는 것을 참고했을때, 노드 대비 간선수가 비교적 적을수록 직관적으로 커뮤니티 독립성이 강하다는 점을 알 수 있다.[8] 또한, EB의 경우 알고리즘의 시간 복잡도가 높은 대신, MM에 비해 우수한 커뮤니티 검출 능력을 보여주는 것으로 실험 결과 나왔다.

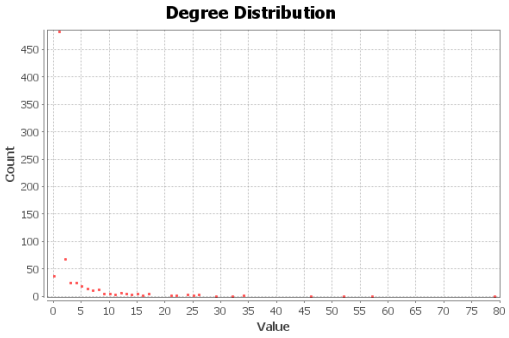

그림 1 java.util의 차수 분포

Value값이 차수이고, Count는 해당 차수를 가진 노드의 수를 의미한다.

표 3 데이터 모듈성 측정

데이터	MM	EB
javax.swing	0.494	0.598
java org	0.782	0.836
com imageio	0.407	0.505
java.util	0.692	0.842

3.4.2 클러스터링 계수는 그래프에서의 각 노드가 얼마나 강하게 밀집되어져 있는지를 나타내는 척도이다. 외부 모듈끼리의 독립성을 커뮤니티 모듈이 보장하였을때, 클러스터링

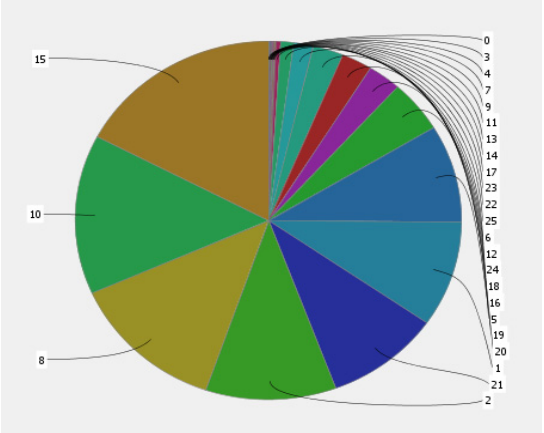

그림 2 MM-검출커뮤니티(swing)

계수가 높게 나온다면 직관적으로 커뮤니티 알고리즘 검출 결과가 아키텍쳐 모듈 뷰의 가치로써도 높을 거라 예상 할 수 있다. 반면에, 허브 노드의 존재로 인해 클러스터링 계수가 높아진다면 그로 인하여 커뮤니티 검출에 좋지 않은 영향을 끼친다고 볼 수 있다. 그런 이유에서 각 데이터의 클러스터링 계수를 구한다.

표 4 그래프 클러스터 일치성 측정(노드들의 평균값)

데이터	AVG.cluster coefficient	노드	간선
javax.swing	0.156(3211 triangle)	2021	5874
java org	0.091(28 triangle)	767	373
com imageio	0.182(264 triangle)	211	726
java.util	0.2(253 triangle)	772	1239

[표 4]의 결과에서 org 패키지의 클러스터링 계수의 수치가 매우 낮다. 이것은 노드 대비 간선의 수가 매우 적기 때문에 기본 클러스터링 계수측정 알고리즘의 트라이앵글을 만족하는 노드 집합생성 확률이 다른 데이터에 비해 적기 때문인 것으로 분석된다. 이 값이 낮다는 것은 반대로 커뮤니티 모듈성이 높아지는 결과를 야기한다는 것을 [표 3]과의 비교과정에서 알 수 있는데, 그래프 각 노드를

연결하는 간선이 밀집되지 않는다는 것이 커뮤니티를 나누었을 때 모듈의 크기에 따라 결과가 더 큰 독립성을 보장할 수 있기 때문이다.

3.4.3. 다음으로 클러스터링 기법과 (3.4.1.)번 실험에서 쓰여진 MM, EB의 모듈성 측정 결과를 기존 소프트웨어 패키지와 비교해 보겠다. 클러스터링 알고리즘으로는 WC를 썼다. 세 알고리즘의 모듈 복원율을 측정하는 알고리즘은 MoJo알고리즘(Move & Join 알고리즘)을 활용하였다. 이 알고리즘은 Move와 Join연산을 통하여 두 모듈간의 유사성을 0에서 100사이의 값으로 나타낸다. 아래 식은 이 알고리즘의 정의이다.[5]

$$Q(M) = (1 - \frac{MoJo(A,B)}{n}) \times 100\%$$

그림 3 Mojo FM

표 5 데이터 복원율 측정

(각 수치는 100%로 환산한 Mojo FM 측정치이다.)

데이터	MM	EB	WC
javax.swing	47.1	32.0	66.5
java org	66.2	31.5	58.6
com imageio	43.5	25.5	54.1
java.util	63.6	40.6	57.1

실험 결과 값을 종합 해 보았을 때, 같은 커뮤니티 검출 기법 중에서도 특히 modularity maximization의 경우 상당히 좋은 복원율을 보였다. 그에 반해 EB는 괄목할만한 성과가 나오지 않았으며, 그 이유는 MM이 모듈성을 높이기 위해 기존 클러스터링 알고리즘과 마찬가지로 노드들 사이의 연결 밀도를 중점적으로 높인 알고리즘이기 때문이다. EB는 알고리즘 기본 컨셉이 내부 노드의 강한 연결성 보다는 영향력이 큰 간선들을 잘라서 나누어진 커뮤니티이기 때문에 패키지 일치성 측면에서는 좋은 결과가 나오지 않았다고 생각해볼 수 있다.

(3.4.1.)번에서 실험한 커뮤니티 모듈성 측정값과 MoJo FM 값의 연관성을 살펴보겠다. MM의 경우엔 커뮤니티 모듈성 측정 방법과 일치성 측정값이 매우 유사한 패턴을 보였고, EB는 크게 일치하지는 않았으나, 모듈성 결과에 따라 근소한 차이를 보이는 것을 알 수 있었다. 이는 커뮤니티의 모듈성 수치가 패키지 복원율의 수치와 무관하지 않다라는 것을 말해준다.

5. 결 론

실험결과에 의하여 커뮤니티를 검출해 내는 특정 알고리즘에 한하여서는 상당히 흥미로운 결과가 나왔다. 커뮤니티 검출 결과에 따른 모듈성이 소프트웨어 아키텍쳐 복원 알고리즘의 복원율과 유사한 수치를 보였기 때문이다. 이 실험이 정확히 증명되기 위해서는 좀 더 정확하고 많은 숫자의 데이터를 통한 동일 실험이 선행되어야 할 것이다. 한편, 현재 각 커뮤니티 기법의 stability를 검증하는 실험 또한 진행 중에 있으며, 그 실험에 대해서는 stability 측정에 필요한 몇몇의 알고리즘과 도구의 구현이 필요할 것으로 보이기 때문에 개발 중에 있다. 커뮤니티 검출 기법이 아키텍쳐 모듈 복원 분야에서 어떠한 성능을 내는가에 관하여 특정 일치하는 부분을 찾는다면 클러스터링과 커뮤니티 검출의 융합이 가능할 것으로 보인다.

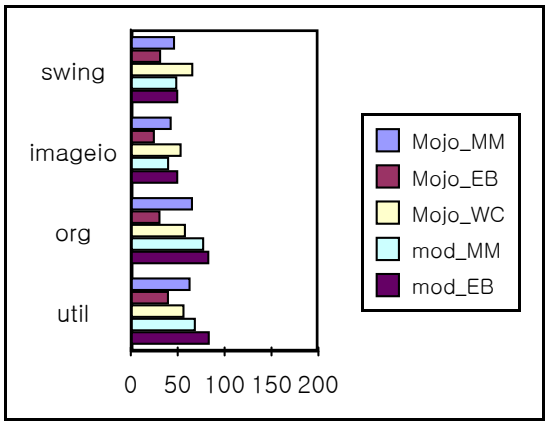

그림 4 각 실험 데이터 분포표

각 데이터의 MojoFm 측정 결과값과 모듈성 측정결과이다. 커뮤니티 모듈성과 데이터 복원율이 비례하는 것을 알 수 있다.

6. 참고문헌

[1] A.L. barabasi, "Link," 2002.

[2] S.Fortunato, "Community detection in graphs," Physics Reports, 2009.

[3] M.V. SteenGraph, Theory and Complex Networks, Maarten van Steen, 2010.

[4] L.Šubelj, M.Bajec, " Community structure of complex software systems: Analysis," Physica A: vol.390, pp.2968-2975, 2011.

[5] V.Tzerpos, R.C.Holt "MoJo, A Distance Metric for Software Clusterings," proc. WCRE, 1999.

[6] Network Work bench, "Weak Component Clustering", https://nwb.slis.indiana.edu/community/?n=AnalyzeData.WeakComponentClustering.

[7] M. E. J. Newman, "Detecting community structure in networks" Europian Physics Journal. B vol.38 pp.321-330, 2004.

[8] B. H. Good, Y. A. de Montjoye and A. Clauset, "The performance of modularity maximization in practical contexts," Physical Review. E pp.81, 2010.

저자소개

김 정 민

2013년 중앙대학교 컴퓨터공학부 학사.
2013년~현재 중앙대학교 컴퓨터공학과 석사과정.

<관심분야> 실시간 소프트웨어, 소프트웨어 클러스터링, 척도없는 네트워크

이 찬 근

1996년 중앙대학교 전자계산학과 학사
1998년 KAIST 전산학과 석사.
2005년 Univ. of Texas at Austin 전산학과 박사.
2005년~2007년 미국 인텔 소프트웨어 엔지니어.
2007~현재 중앙대학교 컴퓨터공학부 조교수.

<관심분야> 실시간 소프트웨어, 수행시간 모니터링, 소프트웨어 테스팅