ANKARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DOKTORA TEZİ

AĞIRLIKLARIN BELİRSİZ OLDUĞU ÇİZGE ÜZERİNDE TOPLULUK TESPİTİ: SOSYAL AĞ ÜZERİNDE BİR ÇALIŞMA

Pelin ÇETİN

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ANKARA
2022

ÖZET

Doktora Tezi

AĞIRLIKLARIN BELİRSİZ OLDUĞU ÇİZGE ÜZERİNDE TOPLULUK TESPİTİ: SOSYAL AĞ ÜZERİNDE BİR ÇALIŞMA

Pelin ÇETİN

Ankara Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Şahin EMRAH

Karmaşık ağlar kişi veya nesneleri aralarındaki ilişkilerle birlikte gösteren yapılardır. Örneğin; insanlar, proteinler veya internet sayfaları aralarındaki etkileşimlerle beraber karmaşık ağ olarak modellenebilir. Karmaşık ağların analizi, sosyal ağlarda kişiye özel reklam vermek, protein ağlarında benzer proteinleri tespit etmek, arama motorlarında ilgili internet sayfalarını bulmak gibi bu ağı ortaya çıkaran probleme bağlı olarak farklı nedenlerden dolayı yapılmaktadır. Ancak karmaşık ağların büyüklüğü her ögeyi ilişkileri ile beraber incelemeyi zorlaştırmaktadır. Bu nedenle de karmaşık ağlar topluluk adı verilen alt ağlara ayrılarak incelenmektedir. Topluluk tespiti problemi, her üyenin en fazla bir toplulukta bulunduğu ayrık ve birden fazla toplulukta bulunduğu örtüşen topluluk tespiti olmak üzere iki sınıfa ayrılmaktadır.

Bu tezin amacı farklı topluluk yapılarını tespit eden yeni ve verimli algoritmalar önermektir. Yapılan çalışmada karmaşık ağlarda topluluk tespiti yapabilen mevcut algoritmalar incelenmiş, farklı benzerlik ölçütleri ile başlangıç toplulukları kullanmanın etkisi gözlemlenmiştir. Birçok ağda daha verimli olan ve farklı topluluk yapılarını tespit edebilen iki yeni ayrık ve örtüşen topluluk tespiti algoritması önerilmiştir. Önerilen algoritmalar birbirine benzeyen iki komşu üyenin aynı toplulukta bulunabileceği varsayımından yola çıkarak tasarlanmıştır. Algoritmaların başarılarını ölçmek için normalleştirilmiş ortak bilgi, F-skor, modülarite ve performans gibi farklı metriklerle birlikte, bu çalışmada örtüşen topluluklar için uyarlanan kapsama metriği kullanılmıştır. Önerilen algoritmalar gerçek ve sentetik ağlar üzerinde test edilmiş ve mevcut algoritmalarla karşılaştırılmıştır. Ayrık topluluk tespitinde toplulukların doğruluğu hedeflenirken, örtüşen topluluk tespitinde algoritmanın çalışma zamanı da dikkate alınarak başarılı sonuçlar elde edilmiştir.

Eylül 2022, 69 sayfa

Anahtar Kelimeler: karmaşık ağlar, topluluk tespiti, çizge, ayrık topluluklar, örtüşen topluluklar

ABSTRACT

Ph.D. Thesis

COMMUNITY DETECTION ON A GRAPH WHICH WEIGHTS ARE UNKNOWN: A CASE STUDY ON SOCIAL NETWORK

Pelin ÇETİN

Ankara University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Prof. Dr. Şahin EMRAH

Complex networks are structures that represent objects together with their relationships. The connections between people, proteins, and web pages can be modeled as complex networks. Complex networks are analyzed for different reasons depending on the problem that creates this network, such as giving personalized advertisements in social networks, detecting similar proteins in protein networks, and finding relevant web pages in search engines. However, the size of complex networks makes it difficult to examine each user or object individually with their relationships. Therefore, complex networks are examined by dividing them into sub-networks called communities. The community detection problem is divided into two classes: discrete, where each member is in at most one community, and overlapping, where there is more than one community.

In this thesis, new discrete and overlapping community detection algorithms that are more efficient in many networks and can detect different community structures are proposed. Existing algorithms are examined, and the effect of using initial communities and different similarity criteria is observed. Besides different metrics such as normalized mutual information, F-score, modularity, and performance, the coverage metric is adapted for overlapping communities in this study and used for measuring the success of the algorithms. The proposed algorithms are tested on real and synthetic networks and compared with existing algorithms. While the accuracy of the communities is taken into account in discrete community detection, the running time of the algorithm is also considered in overlapping community detection, and successful results are obtained for both algorithms.

September 2022, 69 pages

Key Words: complex networks, community detection, graph, discreate communities, overlapping communities

ÖNSÖZ ve TEŞEKKÜR

Doktora eğitimim boyunca bilgi, birikim ve tecrübeleri ile yol gösterici olan ve olumlu tavrıyla beni destekleyen değerli danışman hocam Prof. Dr. Şahin EMRAH'a teşekkür ve saygılarımı sunarım.

Tez sürecim boyunca yönlendirmeleriyle önemli katkılarda bulunan değerli hocalarım Prof. Dr. Ferda Nur ALPASLAN ve Dr. Öğretim Üyesi Bülent TUĞRUL'a, yorum ve önerileri ile tezime katkı sağlayan değerli jüri üyelerim Prof. Dr. Suat ÖZDEMİR ve Prof. Dr. İman ASKERBEYLİ'ye saygılarımla teşekkür ederim.

Ayrıca bu süreçte bana destek olan aileme en içten teşekkürlerimi sunarım.

Pelin ÇETİN
Ankara, Eylül 2022

İÇİNDEKİLER

KISALTMALAR DİZİNİ

BNMF	Bayesian Negatif Olmayan Matris Çarpanlara Ayırma
BNMTF	Sınırlı Negatif Olmayan Matris Üçlü Çarpanlara Ayırma
CDlib	Topluluk Tespiti Kütüphanesi
CONGA	Örtüşen Küme Newman Girvan Algoritması
CONGO	Optimize Edilmiş Örtüşen Küme Newman Girvan Algoritması
COPRA	Topluluk Örtüşme Yayılım Algoritması
DEMON	Bir Ağın Modüler Organizasyonunun Demokratik Tahmini
GCE	Açgözlü Klik Genişleme
LPANNI	Komşu Köşe Etkili Etiket Yayılımı Algoritması
NMI	Normalleştirilmiş Ortak Bilgi
NMF	Negatif Olmayan Matris Faktorizasyonu
SLPA	Konuşmacı-Dinleyici Etiket Yayılım Algoritması

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Bilgisayar bilimleri, sosyal bilimler, biyoloji gibi farklı bilim dallarında veriler ve bu veriler arasındaki ilişkiler ağ ile modellenmektedir. Ağlar nesneleri ve bu nesneler arasındaki ilişkileri temsil eder. Örneğin, sosyal ağlarda nesneler insanları, bu nesneler arasındaki ilişkiler ise arkadaşlık bağını temsil etmektedir. Sosyal ağ yöneticileri kullanıcıları daha iyi tanıyarak onlara uygun reklamları vermek için karmaşık ağlardan faydalanırlar. Ancak ağın boyutu büyüdükçe her kullanıcıyı ilişkileri ile beraber ele almak zorlaşmaktadır. Bu problem büyük boyutlardaki karmaşık ağın alt ağlara ayrılmasıyla çözülebilir. Ağın alt ağlara ayrılması ağın yönetilebilirliğini kolaylaştırarak sosyal ağlarda ilgili kişilerle bilgi paylaşımı yapmaya, tavsiye sistemlerinde benzer profildeki kişi veya nesneleri bulmaya, arama motorlarında benzer konuları bulmaya, ilaç üretiminde benzer proteinleri bulmaya olanak tanır.

Literatürde ağı alt ağlara ayrıma işlemi için hem kümeleme hem de topluluk tespiti algoritmaları kullanılmaktadır. Nesneler, kümeleme algoritmalarında özelliklerine göre gruplandırılırken topluluk tespiti algoritmalarında aralarındaki ilişkilere göre gruplandırılır. Topluluk tespiti, kümelemenin ağlar için özelleştirilmiş halidir. Topluluk tespitinde içeriğe bağlı veriler çizge ile modellenerek yapısal verilere dönüştürülür. Böylece alan bağımsız, genel çözümler üretilir. Bu tezde topluluk tespiti probleminin çözümü üzerinde çalışılmıştır.

Topluluk tespiti problemi kendi içinde zorluklar barındırmaktadır. Bunların başında topluluğu tanımlamak gelmektedir. Topluluk tanımı nesneler arasındaki ilişkilere göre yapılmaktadır ve topluluk yapısında hangi tür ilişkilerin bulunacağı görecelidir. Örneğin, sosyal ağ için bütün üyelerin birbiri ile iletişimde olduğu yapılara topluluk denildiğinde, üyelerden biri başka bir üye ile iletişimi kestiğinde diğer bütün üyelerle iletişimi devam etmesine rağmen topluluktan ayrılmış olur. Benzer şekilde bir ünlünün hayran kitlesini o ünlünün topluluğu olarak tanımlandığında hayranlar, aralarında herhangi bir bağ bulunmamasına rağmen aynı topluluğa ait olacaklardır. Bu nedenlerle topluluk matematiksel olarak kesin bir şekilde tanımlanamamakta ve sezgisel yöntemler kullanılmaktadır.

Sezgisel yöntemlerde topluluklar, üyeler arasındaki bağlantının veya benzerliğin gücüne göre belirlenmektedir. Bu yöntemler en yüksek benzerlik değeri veren üyelerin aynı topluluğa dahil edilmesi prensibine dayanır. Ancak sezgisel algoritmalarla elde edilen sonuçlar, hesaplanan değerlere, topluluğa ilk dahil edilen üyelere, aynı değere sahip üyeler arasındaki seçimlere göre değişiklik göstermektedir. Bazı sezgisel algoritmalar aynı değere sahip üyeler arasında rastgele seçim yapmakta ve bu nedenle her çalıştığında farklı bir sonuç üretebilmektedir. Topluluk kesin bir şekilde tanımlanamasa da en genel anlamda, üyeleri arasındaki bağın topluluğun dışındaki üyelerle bağlarına göre daha yoğun olduğu yapılar olarak tanımlanmaktadır.

Topluluğun belirli bir tanımı olmadığından bulunan toplulukları doğrulamak da zordur. Küçük ağlarda bu doğrulama işlemi göreceli de olsa gözle yapılabilirken büyük ağların görselleştirilmesi ve toplulukların görsel olarak doğrulanması mümkün değildir. Toplulukların önceden tanımlandığı durumlarda, elde edilen sonuçlar mevcut metriklerle doğrulanabilmektedir. Toplulukların önceden bilinmediği durumlarda ise doğrulama işlemi genel topluluk tanıma uygun metriklerle yapılmaktadır. Ancak bu metriklerde de sorunlar bulunmaktadır. Örneğin, geliştirilen metrik çizgenin boyutuna bağlı ise farklı boyuttaki çizgelerin karşılaştırmasında kullanılamamaktadır, bazı metrikler her köşelerin bir topluluğa dahil edilmesini gerektirmektedir. Bu nedenle topluluğu belirlenemeyen veya birden fazla topluluğa dahil edilen üyeler için kullanılamamaktadırlar.

Yukarıda da belirtildiği gibi bazı durumlarda bir üye birden fazla topluluğa dahil edilebilmektedir. Örneğin, bir kişi hem okul hem de iş ağımızda bulunabilir. Bir tıraş makinesi hem saç hem de sakal ürünleri kategorisine dahil edilebilir. Bir kelime birden fazla anlama sahip olabildiği için arama motorlarında farklı sonuçlar çıkabilir. Ayrıca alt kategoriler bulunabilir; cep telefonu, elektronik kategorisine dahil edildikten sonra cep telefonu kategorisine alınabilir ve markalarına veya belirli bir özelliği sağlayıp sağlamadığına göre daha alt kategorilere ayrılabilir. Bu problemler ise örtüşen topluluk tespiti algoritmaları ile çözülmektedir. Bu algoritmalar üyeleri birden fazla topluluğa dahil ederek topluluk içinde kalan toplulukları ve birden fazla çözümü bir arada görmeye olanak sağlar.

Örtüşen topluluk tespiti veri yapısını detaylı bir şekilde görme imkânı sunarken bazı du-

rumlarda aşırı detaylı sonuç üretilmesi temel yapının tespitini zorlaştırmaktadır. Ayrıca iki topluluğa eşit uzaklıktaki üyelerin hangi topluluğa dahil edilmesi gerektiği konusunda da farklı görüşler bulunmaktadır. Bazı algoritmalar bu üyeleri iki topluluğa da dahil etmezken, bazıları rastgele seçim yapmakta, bazıları işlem sırasını dikkate almakta, bazıları ise iki topluluğa da dahil etmektedir.

Bu nedenlerden dolayı topluluk tespiti araştırılmaya ve geliştirilmeye açık bir konudur. Geliştirilen her algoritma probleme farklı bir bakış açısıyla çözüm sunmaktadır. Bu tezin amacı farklı problemlerin çözümünde kullanılabilecek verimli bir algoritma geliştirmektir. Bu amaç doğrultusunda yönsüz ve ağırlıksız çizgeler ile çalışılmış, ayrık ve örtüşen toplulukları tespit eden iki farklı algoritma geliştirilmiştir. Ayrıca kapsama ölçütü örtüşen topluluklar için uyarlanmıştır. Algoritmalar hem gerçek hem de sentetik küçük ve büyük veriler üzerinde test edilmiştir. Algoritmaların başarıları görsel olarak ve değerlendirme metrikleri ile mevcut algoritmalarla karşılaştırmalı bir şekilde gösterilmiştir. Ancak metriklerin doğruluğu tartışmalı olduğu için farklı topluluk yapılarından oluşan ağlar tanımlanarak algoritmalar bu ağlar üzerinde de test edilmiştir. Mevcut metriklerde başarılı sonuçlar veren algoritmaların bu ağlar üzerinde yeterince başarılı sonuçlar vermediği gözlemlenmiştir. Önerilen örtüşen topluluk tespiti algoritması farklı topluluk yapılarını başarılı bir şekilde tespit edebilmektedir. Bunlara ek olarak önerilen algoritmalar farklı benzerlik metrikleriyle, başlangıç topluluklarıyla, topluluk sayısını parametre olarak almaya, yönlü ve ağırlıklı çizgelerde çalışmaya uygun bir şekilde tasarlanmıştır. Bu da probleme özgü kullanım sağlamaktadır.

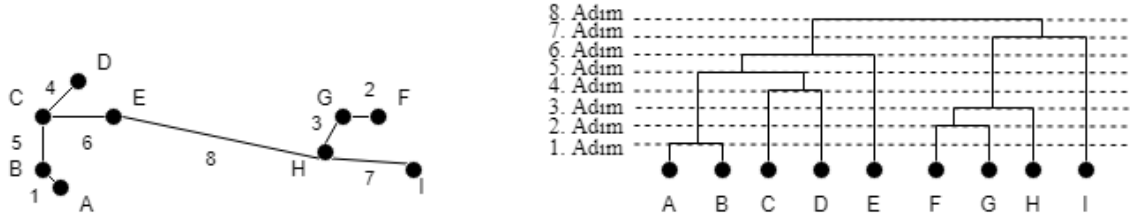
2. KURAMSAL TEMELLER ve KAYNAK ÖZETLERİ

2.1 Kümeleme Problemi

Artan veri ile birlikte verinin yönetilebilirliği zorlaşmaktadır. Gruplara ayrılarak organize edilmiş veri, genel yapının anlaşılmasını ve yönetilebilirliği kolaylaştırır. Bu nedenle kümeleme algoritmaları geliştirilmektedir. Ancak kümeleme problemi kendi içinde zorluklar barındırmaktadır. Kleinberg (2002) aşağıdaki üç kriteri aynı anda sağlayan kümeleme yöntemi bulunmadığını belirtmektedir.

- **Ölçek Değişmezliği:** Herhangi bir d uzaklık fonksiyonunu belli bir α sabiti ile çarpınca kümelemenin sonucu değişmez.
- **Zenginlik:** Sadece noktalar biliniyorsa ve bu noktalar arasındaki uzaklıklar bilinmiyorsa bile olası bütün kümeler üretebilir. Bu durum küme sayılarını belirleyebilmeyi gerektirir.
- **Tutarlılık:** Küme içindeki uzaklıklar azaltılıp kümeler arasındaki uzaklık artırıldığında kümeleme sonucu değişmez.

Kleinberg bu teoremi Tek Bağlantı Kümeleme (Single Linkage Clustering) algoritması üzerinden açıklamaktadır. Tek Bağlantı Kümeleme Algoritması bir hiyerarşik kümeleme algoritmasıdır. Başlangıçta bütün noktalar ayrı bir kümenin elemanıdır. İki farklı kümeye ait en yakın iki nokta arasındaki mesafe iki küme arasındaki mesafe olarak belirlenir. Her adımda birbirine en yakın iki küme birleştirilerek mesafeler yeniden hesaplanır ve birleştirme işlemine tek bir küme kalana kadar devam edilir.



Şekil 2.1 Tek bağlantı kümeleme algoritması

Tek Bağlantı Kümeleme algoritmasının nasıl çalıştığı şekil 2.1'de gösterilmiştir. Şekil 2.1'de kenarların ağırlıkları iki köşe arasındaki mesafeyi belirtmektedir ve başlangıçta her köşe kendi başına bir kümedir. İlk olarak birbirine en yakın mesafeye sahip A ile B köşelerinin kümeleri birleştirilir ve kümeler arasındaki mesafeler yeniden hesaplanır. Bu durumda bir önceki adımda 6 olan A ile C köşelerinin kümelerinin arasındaki uzaklık B köşesinin de A köşesinin kümesine eklenmesiyle A ile C ve B ile C köşeleri arasındaki minimum uzaklık olan 5 birime düşer. Daha sonra en yakın mesafeye sahip iki küme birleştirilerek işlemler tekrarlanır. Bu algoritmayı durdurmak için üç farklı kriter kullanılarak ölçek değişmezliği, zenginlik ve tutarlılık kriterlerinden hangilerinin karşılandığı incelenmiştir.

- Küme sayısı k olduğunda: Kümeler bu kritere göre belirlendiğinde her zaman iki küme bulunur. Bu durum tutarlılık ve ölçek değişmezliği kriterini sağlasa da zenginlik kriteri ile çelişmektedir.

- Kümeler arasındaki uzaklık d olduğunda: Küme içindeki uzaklıklar azaltılıp kümeler arasındaki uzaklıklar artırıldığında kümelemenin sonucu değişmeyeceğinden tutarlılık kriteri sağlanmaktadır. Ancak ölçek değiştiğinde farklı kümeler bulunacaktır. Bu durum zenginlik kriterini sağlasa da ölçek değişmezliği kriteri ile çelişmektedir.

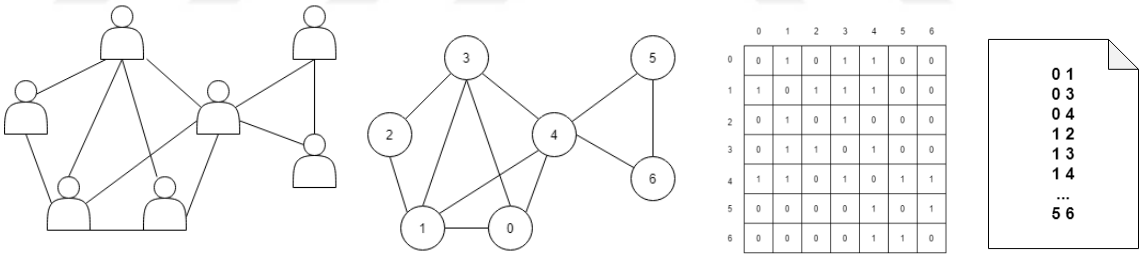
- Kümeler arasındaki uzaklık *d/maksimum uzaklık* olduğunda: Ölçek değiştiğinde kriter de değişeceğinden ölçek değişmezliği ve zenginlik kriterleri sağlanacaktır. Ancak aynı kümeye ait üyeler arasındaki mesafe azaltılıp farklı kümedekiler arasındaki mesafe artırıldığında maksimum uzaklık da artacağı için kümelemenin sonucu değişecek ve tutarlılık kriteri sağlanamayacaktır.

Bu üç kriteri aynı anda sağlayan bir algoritma bulmak imkansızdır. Önerilen algoritmalar bu üç kriterden en fazla ikisini sağlayabilmektedir. Buna bağlı olarak bir çizge için hangi parçalamanın daha iyi olacağının net bir tanımı bulunmamaktadır ve probleme göre değişkenlik göstermektedir. Bu durum topluluk tespiti algoritmalarında da geçerlidir.

2.2 Topluluk Tespiti Problemi

Topluluk tespiti, ağ üzerindeki ögelerin ilişkilerine göre kümelenmesi problemidir. Bir ağdaki üyeler V köşeleri, üyeler arasındaki ilişkileri ise E kenarları ile temsil edildiğinde bütün ağ $G = (V, E)$ çizgesiyle ifade edilebilmektedir. Üyeler arasındaki ilişkinin sadece varlığı biliniyorsa ağırlıksız, gücü belli ise ağırlıklı çizgeler kullanılmaktadır. Üyeler arasındaki ilişki karşılıklıysa yönsüz, tek taraflı ise yönlü çizgeler ile modellenir. Örneğin, sosyal bir ağda arkadaşlık ilişkisi karşılıklıdır ve yönsüz çizgeler kullanılır. Ancak tanışıklık ilişkisi tek taraflı olabilmektedir ve yönlü çizgeler kullanılır.

Çizgelerle modellenebilen ağlar matematiksel olarak komşuluk matrisi veya komşuluk listeleri ile temsil edilmektedir. Komşuluk listeleri bellekte daha az yer tuttuğu için çoğunlukla tercih edilen yöntemlerdir. Bu nedenle birçok çizge sırasıyla kaynak köşe, hedef köşe ve ağırlık bilgisinin bulunduğu kenar listesi dosyalarında tutulmaktadır. Şekil 2.2'de (a) arkadaşlık ağı (b) çizgesi ile modellenmiştir. Komşuluk matrisi (c)'de, ağırlık bilgisinin bulunmadığı kenar listesi dosyası ise (d)'de gösterilmiştir.



(a) (b) (c) (d)

Şekil 2.2 Arkadaşlık ağının (a), çizge (b), komşuluk matrisi (c) ve kenar listesi (d) ile gösterimi

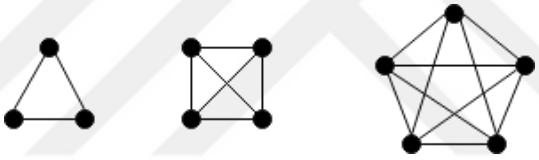
2.3 Topluluk Tanımı

Çizge üzerinde topluluk tespitinde en temel problem topluluğun tanımıdır. Çünkü algoritmaların başarısı kullandıkları topluluk tanımına göre değişmektedir. Belirli bir yapıyı başarılı bir şekilde tespit eden algoritmalar bu yapıda en ufak bir değişiklik olduğunda

başarısız olabilmektedir. Topluluk tanımı, probleme bağlı olmakla birlikte bir problemin farklı yapılarda topluluklar içermesi de mümkündür. Bu nedenle literatürde topluluklar başlangıçta matematiksel olarak tanımlansa da son zamanlarda daha esnek olan sezgisel tanımlar kullanılmıştır. Bu bölümde literatürde bulunan matematiksel ve sezgisel tanımlara değinilmiştir.

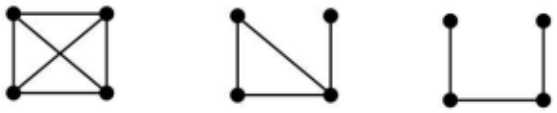
2.3.1 Matematiksel tanımlar

Topluluğun en bilinen matematiksel tanımlarından biri kliklerdir. k köşeden oluşan ve bütün köşelerin birbiri ile bağlantılı olduğu, maksimum kenar sayısına sahip tam çizgelere **k-klik** denir. Şekil 2.3'te bazı k-klik örnekleri verilmiştir. Birçok ağda büyük kliklere rastlamak pek mümkün değildir. Çünkü sadece bir kenarın eksik olması bile alt çizgeyi klik olmaktan çıkarmaktır. Bu nedenle daha esnek tanımlara ihtiyaç duyulmuştur.



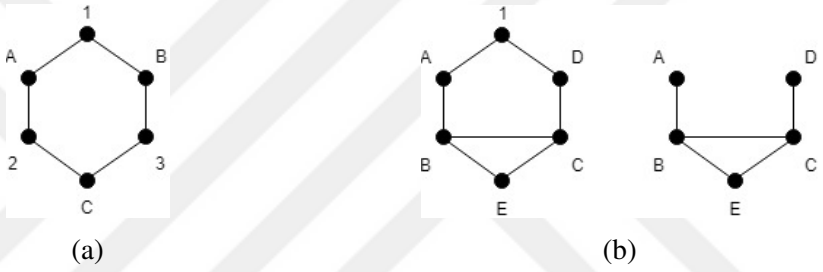
Şekil 2.3 Sırasıyla 3-klik, 4-klik ve 5-klik

Luce (1950) iki köşe arasındaki uzaklığın maksimum n olduğu kümeleri **n-klik** olarak tanımlamıştır. Bu tanıma göre $n = 1$ için orijinal k-klik tanımı geçerlidir. Şekil 2.4'te görüldüğü gibi şekil 2.3'te verilen 4-klik'in iki kenarı çıkarıldığında şekil 2.4'te verilen ve maksimum uzaklığı 2 olan 2-klik elde edilmektedir. İki kenarının eksik olması şekil 2.3'teki yapıyı 4-klik olmaktan çıkarırken Luce'nin tanımına göre klik özelliğini sağlamaya devam etmektedir. Bu tanım n değeri artırılarak daha da esnetilebilmektedir. Böylece şekil 2.3'te verilen 4-klik'in 3 kenarı çıkarıldığında bile klik olarak tanımlanması sağlanabilir.



Şekil 2.4 Sırasıyla 1-klik, 2-klik ve 3-klik

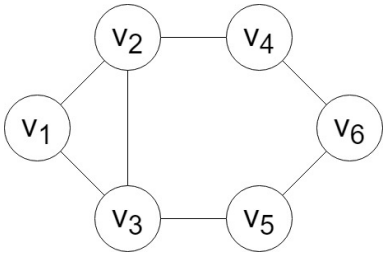
Alba (1973) şekil 2.5'te verdiği örnek ile bu yöntemle elde edilen alt çizgenin bağlantısız olabileceğini göstermiştir. Şekil 2.5a'da $\{A, B, C\}$ ve $\{1, 2, 3\}$ alt çizgelerinin ikisi de 2-klik tanımına uymaktadır ancak bu çizgeler bağlantısızdır. Ayrıca n uzaklıktaki köşeler aranırken klik dışına çıkılabildiğinden köşeler arasındaki uzaklık n'den az olsa bile klik çapı yani en uzak iki köşe arasındaki en kısa uzaklığın değeri n'den fazla olabilmektedir. Şekil 2.5b'de solda verilen örnekte $\{A, B, C, D, E\}$ alt çizgesi 2-klik olmasına rağmen çapı 3'tür. Çünkü 1 köşesi alt çizgede bulunmadığından A köşesinden D köşesine en kısa uzaklık 3'tür. Bu nedenle Alba (1973) n-kliği çapı n olan klik olarak tanımlamıştır. Bu tanım daha sonra Mokken (1979) tarafından **n-klan** olarak isimlendirilmiştir. n-klanda, Luce'nin n-klik tanımına uyma ve maksimal n-klik olma zorunluluğu vardır.



(a) (b)

Şekil 2.5 n-klik (a) bağlantısız çizge problemi, (b) maksimum çap problemi

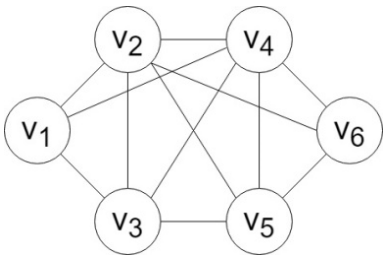
Daha sonra bu tanım da esnetilerek n-klik tanımına uyma zorunluluğu kaldırılmış ve maksimum çapının n olduğu alt çizgeye **n-kulüp** denilmiştir. Bu tanımlar şekil 2.6'da bir örnek üzerinde gösterilmiştir. n-klan, hem n-klik olma zorunluluğu hem de maksimum n çapında olma zorunluluğu taşımaktadır. Bu nedenle, n-klik ve n-kulübün kesişimi n-klanı vermektedir.



2-klik = $\{v_1, v_2, v_3, v_4, v_5\}, \{v_2, v_3, v_4, v_5, v_6\}$
2-klan = $\{v_2, v_3, v_4, v_5, v_6\}$
2-kulüp = $\{v_1, v_2, v_3, v_4\}, \{v_1, v_2, v_3, v_5\}, \{v_2, v_3, v_4, v_5, v_6\}$

Şekil 2.6 n-klik, n-klan ve n-kulüp örneği

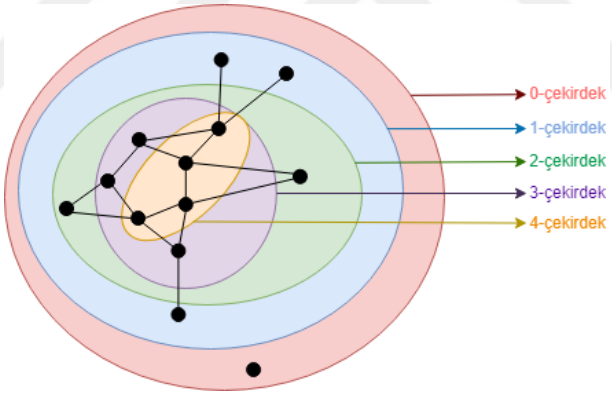
Bir diğer topluluk tanımı ise köşelerin diğer köşelerle olan sınırlı ilişkisine dayanmaktadır. Her köşesi en fazla $k-1$ kadar köşeyle bağlantılı olmayan çizgeye ise **k-plex** denir. k-plex örneği şekil 2.7'de verilmiştir.



1-plex = $\{v_2, v_3, v_4, v_5\}$
2-plex = $\{v_1, v_2, v_3, v_4, v_5\}, \{v_2, v_3, v_4, v_5, v_6\}$
3-plex = $\{v_1, v_2, v_3, v_4, v_5, v_6\}$

Şekil 2.7 k-plex örneği

k-çekirdek (k-core) ise her köşesi en az k sayıda köşe ile bağlantılı çizgedir (Batagelj ve Zaversnik 2011). Bu nedenle k arttıkça topluluk içindeki kenar yoğunluğu da artar. k-çekirdeklerin gösterildiği örnek bir çizge şekil 2.8'de verilmiştir.



Şekil 2.8 k-çekirdek örneği

F-grup ise X ve Y köşesi arasında ve Y ile Z köşesi arasında bir bağ varsa bu bağları güçlü olarak tanımlar ve ek olarak X'ten de Z'ye zayıf bir bağ olduğunu belirtir. **LS kümesi** ise iç derecelerinin dış derecelerinden büyük olduğu çizgedir. Ancak bu yöntemle elde edilen topluluklar çok kuvvetli bağlı topluluklardır ve bu nedenle daha esnek tanıma sahip lambda kümeleri tanımı tercih edilebilmektedir. **Lambda kümelerinde** her köşe ikilisi biri küme içinden biri küme dışından olan herhangi iki köşeye göre daha fazla bağ yoğunluğu içerir.

Topluluk için birçok matematiksel tanım geliştirilmiştir. Geliştirilen her tanımın avantajı ve dezavantajı bulunmaktadır. Bununla birlikte topluluk tespiti NP-Zor sınıfında bir problemdir. Bu nedenle matematiksel tanım kullanan algoritmalar çok fazla işlem zamanı gerektirmekte ve büyük ağlarda kullanılamamaktadır. Bu durumlarda yaklaşık sonuç üreten sezgisel algoritmalar tercih edilmektedir.

2.3.2 Sezgisel tanımlar

Fortunato (2010) toplulukları tanımlamanın ancak çizgenin seyrek olması ile mümkün olacağını belirtmiştir. Çizgedeki kenar sayısının köşe sayısından çok büyük olduğu durumlarda kenarlar, toplulukların tespitini mümkün kılamayacak şekilde homojen dağılır. Bu durumda topluluk tespiti yapısal olarak yapılamamaktadır. Bu nedenle çizge üzerinde topluluk tespiti kenar yoğunluğuyla ilgilidir.

n_A köşeden oluşan A çizgesi, n köşeden oluşan G çizgesinin bir alt çizgesi olsun. $\forall v \in A$ için iç derece $k_v^{\text{iç}}$, dış derece $k_v^{\text{dış}}$ olarak gösterildiğine;

- $k_v^{\text{dış}} = 0$ ise v köşesi sadece A çizgesindeki köşelerle bağlantılı demektir. Bu da tespit edilen toplulukların başarılı olduğunu göstermektedir.
- $k_v^{\text{iç}} = 0$ ise v köşesinin A çizgesi ile bir bağlantısı olmadığını, yeni bir topluluk olarak ayrılması gerektiğini göstermektedir.
- A çizgesindeki köşelerin iç derecelerinin toplamına $k_A^{\text{iç}}$, dış derecelerin toplamına $k_A^{\text{dış}}$ denilirse; A çizgesindeki derecelerin toplamı $k_A = k_A^{\text{iç}} + k_A^{\text{dış}}$ olur.

A'nın dış kenarlarının sayısı $n_A^{\text{dış}}$, iç kenarlarının sayısı ise $n_A^{\text{iç}}$ ile ifade edildiğinde, A çizgesinin dış ve iç bağlantılarının yoğunluğu, olası bağlantı sayılarına göre sırasıyla formül 2.1 ve formül 2.2 ile hesaplanır.

$$\delta_{\text{dış}}(A) = \frac{n_A^{\text{dış}}}{n_A(n - n_A)} \tag{2.1}$$

$$\delta_{\text{iç}}(A) = \frac{n_A^{\text{iç}}}{n_A(n_A - 1)/2} \tag{2.2}$$

Toplulukların başarılı bir şekilde belirlenmiş olması için $\delta_{\text{dış}}(A)$'nın en küçük, $\delta_{\text{iç}}(A)$'nın ise en büyük değeri alması gerekmektedir. Bu nedenle $\delta_{\text{iç}}(A) - \delta_{\text{dış}}(A)$ değeri maksimize edilmelidir (Mancoridis vd. 1998). Sezgisel yöntemler de bu temele dayanmaktadır.

2.4 Kenar Ağırlıklarının Belirlenmesi

Şekil 2.1'de verilen noktalar kümelenirken noktalar arasındaki mesafe dikkate alınmış ve birbirine yakın noktalar aynı kümeye dahil edilmiştir. Bu problem çizge ile modellendiğinde kenarların ağırlıkları noktalar arasındaki mesafeyi ifade eder. Ancak çizge üzerindeki noktalar kişileri veya ürünleri temsil ettiğinde kenar ağırlıkları, fiziksel mesafeler yerine noktalar arasındaki benzerliği veya bağlantının gücünü temsil eder.

İçerik verisinin bilinmediği durumlarda ağırlıklar hesaplanırken ortak komşuluklar veya bir köşeden diğerine ulaşmak için gerekli minimum adım sayısı gibi yapısal özelliklerden faydalanılmaktadır. Aynı zamanda literatürdeki uzaklık formülleri de kullanılmaktadır. Bu formüllerin kullanılabilmesi için köşeler komşuluk vektörleri ile ifade edilir. n elemanlı bir veri kümesinde her köşe için n uzunlukta bir vektör oluşturulur. Bu vektördeki her alan veri kümesindeki bir köşeyi temsil eder. Bu köşeler arasında bağlantı varsa vektörde ilgili alanda 1, yoksa 0 değeri yer alır. Kullanılan metriğe göre elde edilen hesaplama sonucu kenarın ağırlığını belirler.

Topluluk tespiti yapılırken köşeler arasındaki mesafeden çok benzerlikler dikkate alınır. Bu nedenle köşeler arasındaki uzaklıklar hesaplandığında mesafenin en az olduğu köşe çifti birbirine en bezer köşe çifti olarak yorumlanır. Mesafe hesaplayan ölçütlerden bazıları aşağıda incelenmiştir.

Manhattan Uzaklığı'nda iki nokta arasındaki mesafe x ve y düzlemlerindeki toplam değişime göre belirlenmektedir. $p = (x_1, y_1)$ ve $q = (x_2, y_2)$ noktaları için Manhattan uzaklığı formül 2.3 ile hesaplanmaktadır.

$$\text{ManhattanUzaklığı}(p, q) = |x_1 - x_2| + |y_1 - y_2| \tag{2.3}$$

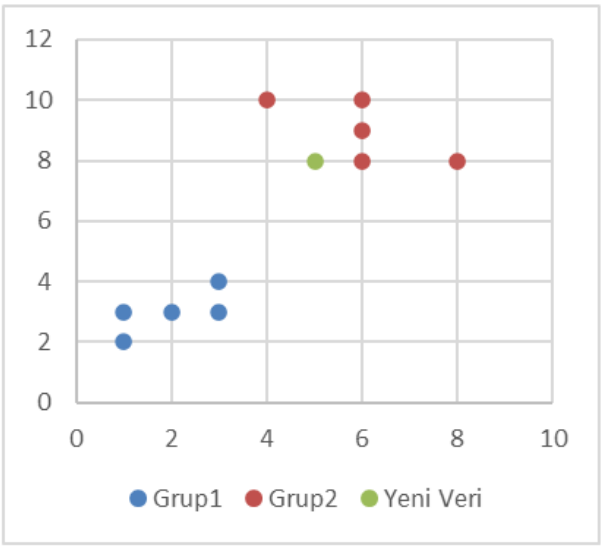
Euclid Uzaklığı iki nokta arasındaki en kısa mesafeyi hesaplar. $p = (x_1, y_1)$ ve $q = (x_2, y_2)$ noktaları arasındaki Euclid uzaklığı formül 2.4 ile hesaplanır.

$$\text{EuclidUzaklığı}(v_i, v_j) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{2.4}$$

Daha sonra bu iki uzaklığı da kapsayan ve genelleştirilmiş bir formül olan **Minkowski Uzaklığı** önerilmiştir. $p = (x_1, y_1)$ ve $q = (x_2, y_2)$ noktaları için Minkowski uzaklığı formül 2.5 ile hesaplanır. Formüldeki λ katsayısı 1 değerini aldığında Manhattan, 2 değerini aldığında Euclid uzaklığının formülünü verir.

$$\text{MinkowskiUzaklığı}(p, q) = (|x_1 - x_2|^\lambda + |y_1 - y_2|^\lambda)^{\frac{1}{\lambda}} \tag{2.5}$$

Mahalanobis Uzaklığı, bu uzaklıklardan farklı olarak değişkenlerin tanımlı olduğu boyuttaki değişimlerini de dikkate alır. Yani bir boyuttaki 1 birimlik değişim diğer boyuttaki 100 birimlik değişime karşılık gelebilir. Mahalanobis uzaklığı kovaryans matrisini kullanarak bu dengeyi sağlar.



Şekil 2.9 Mahalanobis uzaklığı örneği

Şekil 2.9'da mavi ile işaretlenen Grup 1 ve kırmızı ile işaretlenen Grup 2 verilerine ye-

şil ile işaretlenen p verisinin eklenmesi durumunda $p = (x, y)$ noktası için Mahalanobis uzaklığı formül 2.6 ile hesaplanır. Formüldeki μ aritmetik ortalamayı, S ise kovaryans matrisini temsil eder. Kovaryans matrisinin köşegeni ise varyansa eşittir.

$$\text{MahalanobisUzaklığı}(\vec{p}) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})} \tag{2.6}$$

Öncelikle p noktasının Grup 1'e uzaklığına bakılır. Bunun için örnek sayısının n ile gösterildiği formül 2.7 ile x ve y için varyans hesaplanır.

$$\text{Varyans} = \frac{\sum_{i=1}^{n} (x_i - \mu_x)^2}{n - 1} \tag{2.7}$$

Daha sonra x'in x, x'in y, y'nin y ve y'nin x ile kovaryansı formül 2.8 ile hesaplanarak kovaryans matrisi oluşturulur.

$$\text{Kovaryans} = \frac{1}{n - 1} \sum_{i=1}^{n} (x_i - \mu_x)(y_i - \mu_y) \tag{2.8}$$

Bulunan değerler formül 2.6'da yerine konulduğunda elde edilen sonuç p'nin Grup 1'e uzaklığını belirler. Aynı adımlar Grup 2 için tekrarlanarak uzaklık hesaplandığında p noktası yakın olduğu gruba dahil edilir.

Bu çalışmada sadece köşeler arasındaki komşuluklar dikkate alındığından tek boyutta karşılaştırma söz konusudur. Uzaklıklar sadece kenarların değerlendirme sırasını belirlediğinden önemli olan köşeler arasındaki mesafeyi yani farklı komşu sayısını belirlemektir. Bu nedenle önerilen algoritmada Euclid, Minkowski veya Mahalanobis uzaklığının kullanılması sonucu etkilemeyecek, ancak işlem sayısını artıracaktır. Buna bağlı olarak işlem kolaylığı açısından Manhattan uzaklığı tercih edilmiştir. Ayrık topluluk tespiti yapılırken Manhattan uzaklığı ile birlikte aşağıda formülleri verilen Kosinüs ve Jaccard benzerliklerinin etkisi incelenmiştir.

Kosinüs Benzerliği'nde köşeleri komşu köşelere bağlayan her kenar Euclid uzayında bir boyutu temsil eder. İki vektörün skaler çarpımı ortak köşe sayısını, köşeye bağlı kenar sa-

yısının karekökü ise vektörün boyutunu gösterir. İki vektör arasındaki kosinüs benzerliği formül 2.9 ile hesaplanır.

$$\text{KosinüsBenzerliği}(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{2.9}$$

Çizge üzerindeki u köşesinin komşuları $N(u)$ ile gösterildiğinde u ve v köşeleri için kosinüs benzerliği formül 2.10 ile hesaplanır.

$$\text{KosinüsBenzerliği}(u, v) = \frac{|N(u) \bigcap N(v)|}{|N(u)||N(v)|} \tag{2.10}$$

Aralarında bağlantı olmayan köşeler için kosinüs benzerliği sıfırdır. Ancak bazı durumlarda köşeler birbirine bağlı olsa da ortak komşuları bulunmadığı için kosinüs benzerlikleri sıfır çıkmaktadır. Bu durumu önlemek için köşenin kendisi de komşuluk listesine eklemektedir.

Jaccard Benzerliği, iki köşe arasındaki ortak komşuların iki köşeye komşu olan tüm köşelere oranıdır. u köşesinin komşularının sayısına $N(u)$ denildiğinde u ve v köşeleri arasındaki Jaccard benzerliği formül 2.11 ile hesaplanır.

$$\text{JaccardBenzerliği}(u, v) = \frac{|N(u) \bigcap N(v)|}{|N(u) \bigcup N(v)|} \tag{2.11}$$

2.5 Değerlendirme Metrikleri

Toplulukların başarısını ölçmek için kullanılan değerlendirme metrikleri Wagenseller vd. (2018) tarafından yapısal ve fonksiyonel olmak üzere iki başlığa ayırmıştır. Yapısal metrikler çizgenin özelliklerini kullanmaktadır. Performans, modülarite ve kapsama yapısal metriklerdir. Fonksiyonel metrikler ise gerçek toplulukların bilindiği durumlarda kullanılır ve tahmin edilen sonuçlarla gerçek sonuçlar arasındaki farkı hesaplar. F-Skor ve Normalleştirilmiş Ortak Bilgi (Normalized Mutual Information, NMI) ise fonksiyonel metriklerdir.

2.5.1 Fonksiyonel metrikler

F-Skor kesinlik (precision) ve duyarlılık (recall) kullanılarak hesaplanmaktadır. **Kesinlik** pozitif olarak tahmin edilen örneklerin gerçekte ne kadarının pozitif olduğudur. **Duyarlılık** ise gerçekte pozitif olan örneklerin ne kadarının pozitif tahmin edildiğidir. F-Skor bu iki metrik arasında bir denge sağlar ve formül 2.12 ile hesaplanır.

$$F\text{-}Skor = \frac{Kesinlik \times Duyarlılık}{Kesinlik + Duyarlılık} \tag{2.12}$$

Normalleştirilmiş Ortak Bilgi (Normalized Mutual Information, NMI) ise bir bilginin bilinmesi durumunda çıkarılabilen bilgi miktarıdır. Bilgi teorisindeki entropiden türetilmiştir. Süreksiz rastgele değişken X için x etiketinin olasılığı $p(x)$ ile gösterildiğinde entropi $H(X)$ formül 2.13 ile hesaplanır.

$$H(X) = -\sum_{x} p(x)\, log\, p(x) \tag{2.13}$$

X ile Y'nin ortak entropisi $H(X, Y)$ ise formül 2.14 ile hesaplanmaktadır.

$$H(X, Y) = -\sum_{x}\sum_{y} p(x, y)\, log\, p(x, y) \tag{2.14}$$

Ortaklaşa rastgele X ile Y değişkenleri için NMI formül 2.15 ile hesaplanır. Ancak bu formül örtüşen topluluklar için çalışmamaktadır.

$$NMI(X, Y) = \frac{H(X) + H(Y) - H(X, Y)}{\frac{H(X)+H(Y)}{2}} \tag{2.15}$$

Lancichinetti vd. (2009) örtüşen topluluklarda NMI değerini hesaplamak için formül 2.16'da verilen koşullu entropi $H(Y|X)$'i kullanarak formül 2.17'yi önermiştir.

$$H(Y|X) = -\sum_{x}\sum_{y} p(x,y)\, log\, p(y|x) \tag{2.16}$$

$$NMI(X,Y)_{LFK} = 1 - \frac{1}{2}\left(\frac{H(X|Y)}{H(X)} + \frac{H(Y|X)}{H(Y)}\right) \tag{2.17}$$

Ortaklaşa rastgele X ile Y değişkenlerinin karşılıklı bilgisi $I(X:Y)$ formül 2.18 ile hesaplanır.

$$I(X:Y) = \frac{1}{2}[H(X) - H(X|Y) + H(Y) - H(Y|X)] \tag{2.18}$$

McDaid vd. (2011) NMI_{LFK} için kullanılan normalleştirmenin iki topluluğun benzerliğinde fazla büyük değer verdiğini belirtmiş ve karşılıklı bilgiyi de kullanarak kendi normalleştirme yöntemiyle, bu çalışmada da örtüşen topluluklar için kullanılan formül 2.19'u önermiştir.

$$NMI(X,Y)_{MGH} = \frac{I(X:Y)}{max(H(X), H(Y))} \tag{2.19}$$

2.5.2 Yapısal metrikler

Performans kriteri, doğru yorumlanan köşe çiftlerinin sayısının, olası köşe çiftlerinin sayısına oranıdır (Fortunato 2010). Köşelerin doğru yorumlanması, aynı topluluktaki köşelerin arasında doğrudan bir bağ bulunduğu, farklı topluluklardaki köşeler için doğrudan bir bağ bulunmadığı anlamına gelmektedir. Örneğin şekil 2.6'daki çizge için $\{v_1, v_2, v_3\}$ ve $\{v_4, v_5, v_6\}$ topluluklarının bulunduğunu varsayalım. Bu durumda 7 kenarlı bu çizgenin 5 kenarı doğru yorumlanmıştır. v_2 ile v_4 ve v_3 ile v_5 köşeleri farklı topluluklara ait olmasına rağmen aralarında bağlantı bulunduğu için bu kurala göre doğru yorumlanmamıştır. Olası kenar sayısı ise $6 * 5/2 = 15$ olduğundan, performans değeri $5/15 = 3,33$ olarak hesaplanır. Bu metriğin yüksek sonuç vermesi için farklı topluluklara ait köşeler arasındaki bağlantının en az olması gerekmektedir. Bu durumda elde edilebilecek maksimum değer *kenar sayısı/olası kenar sayısı*'na eşittir.

Modülarite literatürde en çok kullanılan değerlendirme kriterinden biridir. Newman ve Girvan (2004) tarafından rastgele bir çizgenin topluluk içermesinin beklenmediği fikrinden yola çıkılarak önerilmiştir. Orijinal çizgenin topluluk yapısı dışındaki bazı yapısal özellikler korunmaya çalışılarak rastgele çizge oluşturulmuş ve verilen çizgenin yoğunluğu ile rastgele çizgenin yoğunluğu karşılaştırılmıştır. Modülaritenin nasıl hesaplandığı formül 2.20'de verilmiştir (Newman 2010).

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij}) \delta(C_i, C_j) \tag{2.20}$$

Formülde toplam kenar sayısı m, komşuluk matrisi A, rastgele seçilen modelde i ve j köşeleri arasında beklenen kenar sayısı P ile ifade edilmektedir. δ ise i ve j köşeleri aynı topluluğa aitse 1, farklı topluluklarda ise 0 sonucunu veren bir fonksiyondur.

Bernoulli rastgele çizgesinde toplam kenar sayısı korunur ve kenarlar, köşeler arasına aynı olasılıkla yerleştirilir. Bernoulli rastgele çizgesinde P olasılık fonksiyonu, toplam kenar sayısının maksimum kenar sayısına oranıdır ve formül 2.21'de verildiği gibi hesaplanır. Formüldeki n köşe sayısını ifade etmektedir.

$$P_{ij} = \frac{2m}{n(n-1)}, \forall i, j \tag{2.21}$$

Ancak bu dağılım gerçek hayattaki çizgelerle uyuşmamaktadır. Bu nedenle orijinal çizgeyle aynı derece dağılımının korunduğu rastgele çizgeler tercih edilmektedir. Molloy ve Reed (1995) tarafından geliştirilen modelde k_i ve k_j dereceli i ve j köşeleri $p_i = k_i/2m$ ve $p_j = k_j/2m$ olasılıkları ile bağlıdır ve i ile j arasında kenar olma olasılığı $P_{ij} = k_i k_j/2m$'dir. Bu durumda modülarite formül 2.22 ile hesaplanır.

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(C_i, C_j) \tag{2.22}$$

Sadece aynı topluluktaki köşeler için toplama işlemi yapıldığından modülarite fonksiyonu; n_c'nin toplam topluluk sayısını, l_c'nin c topluluğundaki toplam kenar sayısını,

d_c'nin c topluluğundaki toplam derece sayısını belirttiği formül 2.23 ile ifade edilebilir. Ancak bu formül örtüşen topluluklar için kullanılamamaktadır.

$$Q = \sum_{c=1}^{n_c} \left[\frac{l_c}{m} - \left(\frac{d_c}{2m} \right)^2 \right] \tag{2.23}$$

Nepusz vd. (2008) formül 2.22'deki $\delta(C_i, C_j)$ yerine köşelerin üyelik derecelerinin çarpımı olan $s_{ij} = \sum_{c=1}^{k} a_{ic} a_{jc}$ benzerlik ölçeğini kullanarak örtüşen topluluklarda çalışabilen ve formül 2.24'te verilen modülarite fonksiyonunu önermiştir. Formülde i köşesinin c topluluğuna aidiyeti a_{ic}, topluluk sayısı ise k ile gösterilmektedir.

$$Q_N = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] s_{ij} \tag{2.24}$$

Shen vd. (2009) ise modülarite formülündeki $\delta(C_i, C_j)$ yerine $1/(O_i O_j)$'yi kullanarak formül 2.25'i önermiştir. Formülde i köşesinin dahil olduğu topluluk sayısı O_i ile belirtilmektedir.

$$Q_S = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \frac{1}{O_i O_j} \tag{2.25}$$

Çeşitli modülarite fonksiyonları önerildikten sonra D. Chen vd. (2010) daha genel bir ifade olan formül 2.26'yı önermiştir ve i ile j köşelerinin c topluluğuna aidiyet derecelerinin çarpımı, ortalaması, maksimumu vb. ile ifade edilebilen $f(a_{ic}, a_{jc})$'yi kullanmıştır.

$$Q_C = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] f(a_{ic}, a_{jc}) \tag{2.26}$$

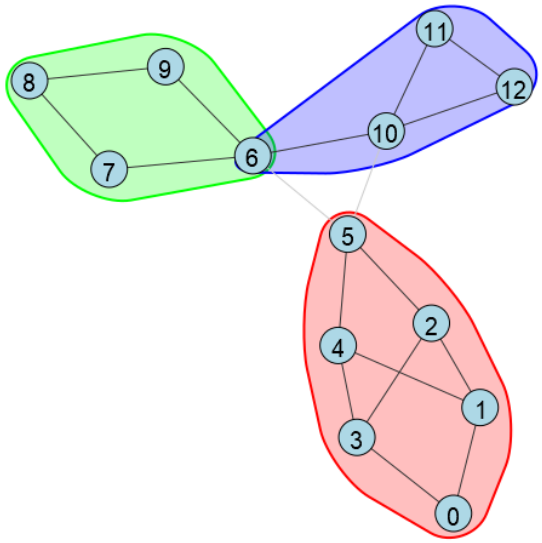
Modülaritenin 0,3 ve 0,7 arasında çıkması topluluk tespitinin başarılı olduğunu göstermektedir. Yüksek değerler çok nadir elde edilir. Ayrıca çizge boyutunun artmasıyla modülarite de arttığından modülarite ile farklı boyutlardaki çizgelerin karşılaştırılması uygun değildir.

Kapsama (Coverage), topluluk içi kenarların toplam kenarlara oranıdır (Fortunato 2010). **Topluluk içi kenarlar (intra-community edges)** aynı topluluktaki iki köşeyi bağlayan kenarlardır. **Topluluklar arası kenarlar (inter-community edges)** ise farklı iki topluluğa ait iki köşeyi bağlayan kenarlardır. Bu metrik ayrık topluluklar için geçerlidir. Örtüşen topluluklarda, iki köşe aynı zamanda hem aynı toplulukta hem de iki farklı topluluklarda bulunabilmektedir. Bu çalışmada topluluk içi kenar tanımı örtüşen topluluklar için uyarlanmıştır.

$G = (V, E)$ çizgesi için $k >= 1$ sayıda ayrık veya örtüşen $C_1, C_2, ...C_k$ topluluklarının bulunduğunu varsayalım. Bu durumda, örtüşen topluluklar için uyarlanan kapsama ölçütü formül 2.27'te verilmiştir.

$$Kapsama = \begin{cases} 0, & if\ k = 1 \\ \frac{|E_{in}|}{|E|}, & if\ k > 1 \end{cases} \tag{2.27}$$

Her $(u, v) \in E$ için eğer hem $u \in C_j$ hem de $v \in C_j$ koşullarını sağlayan bir C_j topluluğu varsa (u, v) kenarına topluluk içi kenar E_{in} denir. Bazı algoritmalar bütün köşeleri tek bir topluluğa atayarak verilen çizgenin tamamını bir topluluk olarak tanımlamaktadır. Bu durumu önlemek için $k = 1$ koşulu tanımlanmıştır.

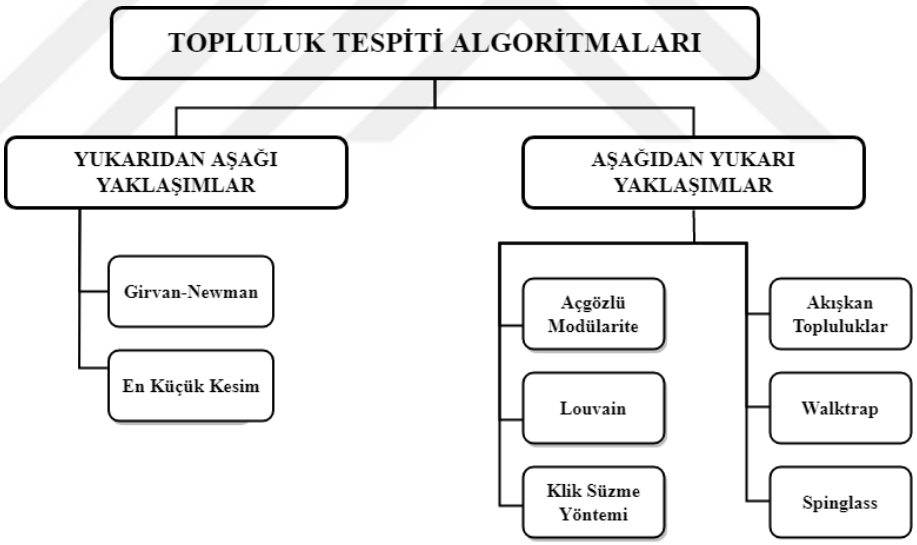


Şekil 2.10 Kapsama örneği

Önerilen uyarlanmış kapsama metriği tanımına göre şekil 2.10'da bulunan $(5, 6)$ ve $(5, 10)$ kenarları hariç bütün kenarlar topluluk içi kenardır. Örneğin $(6, 10)$ kenarı mavi topluluğa ait 6 ve 10 köşelerini bağladığı için bir topluluk içi kenardır. Bu örnek için formül 2.27'de verilen $|E_{in}| = 16$ ve $|E| = 18$'dir. Kapsama sonucu ise 0.89'dur.

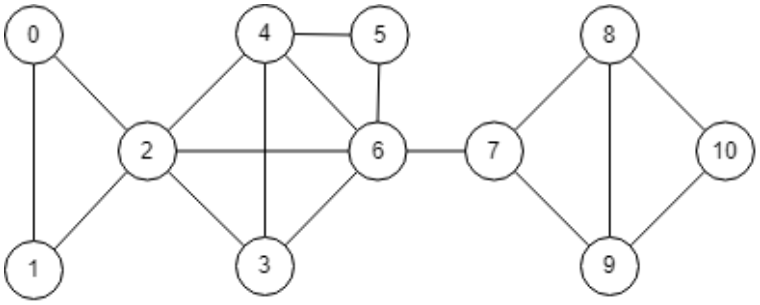
2.6 Topluluk Tespiti Algoritmaları

Topluluk tespitinde temel olarak aşağıdan-yukarı ve yukarıdan-aşağı olmak üzere iki yaklaşım izlenmektedir. Aşağıdan-yukarı yaklaşımla tasarlanan algoritmalar köşeleri birleştirerek toplulukları bulmayı hedeflerken, yukarıdan-aşağı yaklaşımlar ağı parçalayarak toplulukları tespit etmeyi hedefler. İki yaklaşımın da avantajları ve dezavantajları vardır. Bu bölümde Şekil 2.11'de verilen topluluk tespiti algoritmaları açıklanmış ve bu algoritmalardan türetilen algoritmalara yer verilmiştir.



Şekil 2.11 Topluluk tespiti algoritmaları

Bu algoritmalar Şekil 2.12'deki çizge üzerinde çalıştırılarak bulunan topluluklar incelenmiştir ve kullandıkları topluluk tanımına bağlı olarak bulunan topluluklarda da farklılıklar gözlemlenmiştir. Bir alanda başarılı sonuçlar veren algoritma, farklı bir topluluk tanımının geçerli olduğu bir alanda aynı başarıyı gösterememektedir. Bu nedenle topluluk tespiti araştırılmaya açık bir konudur ve önerilen her algoritma önemlidir.



Şekil 2.12 Örnek çizge

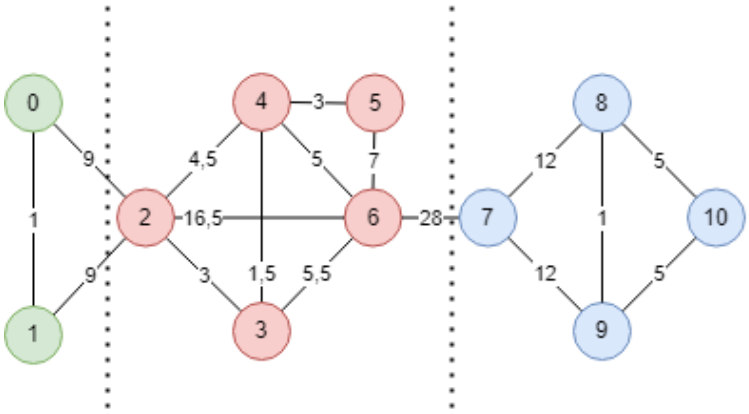
2.6.1 Yukarıdan aşağı yaklaşımlar

Yukarıdan aşağı yaklaşımlar çizgeyi parçalarına ayırarak toplulukları tespit eder. Köprü görevi gören ve en kısa yollar üzerinde bulunan kenarlar çizgeden çıkarılarak topluluklar belirlenir. Köprüler, çıkarıldığı zaman çizgedeki bağlı birleşen sayısını artıran yapılardır ve iki topluluğu birbirine bağladığı için farklı topluluktaki iki köşe arasındaki bütün yollar bu köprülerin üzerinden geçmek zorundadır. Köprüleri çizgeden çıkararak da toplulukları tespit etmek mümkündür. Ancak birbirine birden fazla kenarla bağlı topluluklar için en kısa yol algoritmaları kullanılır. Girvan-Newman ve En Küçük Kesim algoritmaları bu yaklaşımla tasarlanan algoritmalardır.

Girvan-Newman Algoritması

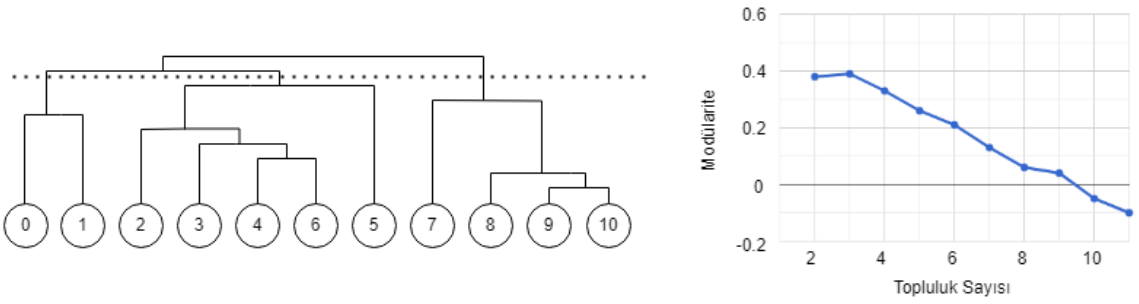
Girvan ve Newman yaptıkları çalışmalar ile topluluk tespiti problemi için önemli kavramlar oluşturmuşlardır (Girvan ve Newman 2002, Newman ve Girvan 2004, Newman 2006). Bunlardan ilki Freeman (1977) tarafından köşeler için tanımlanan arasındalık (betweeness) kavramının kenarlara uyarlanarak kenar arasındalığının literatüre kazandırılmasıdır. **Kenar arasındalığı (edge-betweeness)** bir kenarın, köşeler arasındaki en kısa yolların kaçında bulunduğunu gösterir. Bu nedenle kenar arasındalığı, bir köşeden diğerine bir tane kısa yol varsa 1, n_s tane kısa yol varsa $1/n_s$ kadar artar.

Bu algoritmanın ilk adımında en kısa yol üzerinde bulunamayacağı ve topluluk yapısını etkilemeyeceği için varsa köşeler üzerindeki döngüler kaldırılır. Kalan kenarların ağırlıkları kenar arasındalığına göre belirlenir. Sırasıyla ağırlığı en yüksek olan kenardan başla-



Şekil 2.13 Girvan-Newman algoritmasının örnek çizge üzerindeki sonucu

yarak bütün köşeler bağlantısız kalıncaya kadar kenarlar çizgeden çıkarılır. Şekil 2.13'te Girvan-Newman algoritmasının örnek çizge üzerinde nasıl çalıştığı gösterilmiştir. (3, 4) kenarı 3 ve 4 numaralı köşeler arasındaki tek en kısa yoldur. 3 numaralı köşeden 5 numaralı köşeye ise en kısa yol 2 adımdır. Bu yollardan biri (3, 4) diğeri (3, 6) kenarından geçer, bu nedenle (3, 4) kenarı 3'ten 5'e giden en kısa yolların yarısından geçmektedir. (3, 4) kenarı 3'ten 4 ve 5 numaralı köşelere gidenler hariç diğer köşelere giden en kısa yolların hiçbirinde bulunmamaktadır. Bu nedenle kenar arasındalık değeri $1 + 1/2 = 1,5$'tur. Benzer şekilde (8, 10) kenarı da 10'dan 8'e olan en kısa yol üzerinde bulunur ve 9 hariç kalan 8 köşeye olan en kısa yolların yarısı (8, 10), diğer yarısı (9, 10) üzerinden geçer ve kenar arasındalığı $1 + 8(1/2) = 5$'tir. Bu şekilde bütün kenar arasındalıkları hesaplandıktan sonra bu kenarlar büyükten küçüğe doğru sırasıyla çizgeden çıkarılarak şekil 2.14'te verilen hiyerarşik ağaç oluşturulur.



Şekil 2.14 Girvan-Newman algoritması

Hiyerarşik yapı oluşturulduktan sonra toplulukları belirlemek için bu yapının nereden ke-

sileceği ve en iyi topluluk sayısının kaç olduğunu bilmek gerekmektedir. Bu nedenle en uygun topluluk sayısını belirlemek için literatüre kazandırılan bir diğer önemli kavram modülaritedir. Newman (2010) tarafından önerilen modülarite formül 2.20'de verilmiştir. Örnek çizgede hiyerarşik ağaç en yüksek modülarite değerini veren yerden kesildiğinde $\{\{0,1\},\{2,3,4,5,6\},\{7,8,9,10\}\}$ toplulukları elde edilir.

Bu algoritma ile çıkarılan hiyerarşik yapı sayesinde bütün ağ incelenebilir ancak hem modülarite hesabı maliyetli olduğundan hem de hiyerarşik yapı oluşana kadar bütün kenarlar değerlendirildiğinden işlem zamanı fazladır. Bu nedenle Fang-ju (2017) kenarları çıkarırken arasındalık merkeziyeti yerine ortak komşuları, komşularının %20'sinden daha az olan kenarları kullanarak daha hızlı bir algoritma önermiştir. Arasteh ve Alizadeh (2019) ise aynı anda birden fazla kenarın çıkarılmasına olanak sağlayan bir yöntem önermiştir.

Gregory (2007) tarafından Girvan-Newman algoritmasının örtüşen versiyonu olan Örtüşen Küme Newman Girvan Algoritması (Cluster-Overlap Newman-Girvan Algorithm, CONGA) önerilmiştir. Gregory (2008) daha sonra yerel arasındalık kullanara Optimize Edilmiş Örtüşen Küme Newman Girvan Algoritmasını (CONGA Optimized, CONGO) önermiştir. Ma vd. (2016) Girvan-Newman algoritmasına benzer bir mantıkla kenar ağırlıklarını ortak komşuların geometrik ortalamasını alarak belirlemiş ve döngüsel kenar silme algoritması ile toplulukları tespit etmiştir.

En Küçük Kesim (Min-Cut) Algoritması

En küçük kesim algoritmasına göre farklı iki topluluk arasındaki bağlantı sayısı minimum olmalıdır. Yani çizge bir doğru ile c ve $\bar{c}$ olmak üzere iki topluluğa ayrıldığında bu doğrunun kestiği kenar sayısı $kesim(c,\bar{c})$ minimum olmalıdır. Bu durumda C topluluklar kümesi için kesim noktasının değeri formül 2.28 ile hesaplanır.

$$\text{KesimNoktasınınDeğeri(C)} = \frac{1}{|C|}\sum_{c\epsilon C}\frac{kesim(c,\bar{c})}{|c|} \tag{2.28}$$

Ancak formül 2.28'in dezavantajı bulunmaktadır. Örneğin, bir köşe çizgeye tek bir kenar ile bağlı ise en küçük kesim noktası da bu kenardan geçecektir ve bulunan topluluklar bu

köşe ile çizgenin diğer köşeleri olacaktır. Bu durumu önlemek için normalizasyon yapılır. En küçük kesim algoritmasında kesim noktalarının değerleri, c topluluğunun derecelerinin $deg(c)$ ile gösterildiği formül 2.29 ile hesaplanır.

$$\text{NormalleştirilmişKesim(C)} = \frac{1}{|C|}\sum_{c\epsilon C}\frac{kesim(c,\bar{c})}{\sum \text{deg(c)}} \tag{2.29}$$

Şekil 2.14'te 6 ve 7 numaralı köşeleri birbirine bağlayan kenar üzerinden kesim yapıldığında, $\{\{0,1,2,3,4,5,6\},\{7,8,9,10\}\}$ topluluklarını oluşturan kesim noktasının değeri formül 2.30'daki gibi hesaplanır.

$$\text{NormalleştirilmişKesim(C)} = \frac{1}{2}\left(\frac{1}{11}+\frac{1}{23}\right) = 0,067 \tag{2.30}$$

Bu algoritmada kaç kesim yapılacağı topluluk sayısına göre belirlendiğinden topluluk sayısı önceden bilinmelidir. Bu nedenle Bai vd. (2018) topluluk sayısı yerine λ parametresini önermişlerdir ve [0,3, 0,6] aralığında bir değer seçildiğinde gerçek topluluk sayısına yakın bir sonuç elde edildiğini belirtmişlerdir. λ değeri büyüdükçe tespit edilen topluluk sayısı da artmaktadır. Ayrıca ağı basitleştirerek ağırlıklı bir ağaca çevirdikten sonra en küçük kesim algoritması uygulandığında daha iyi topluluk tespiti yapıldığını belirtmiştir.

Lierde vd. (2019) ise spektral kümeleme algoritmasını kesim minimizasyonunda kullanarak örtüşen toplulukları tespit etmiştir. [0, 0,5] aralığında bir α değişkeni kullanılarak bu değerin altında sonuç alındığında köşenin topluluğa ait olmadığı belirlenmiştir. Birden fazla topluluk için bu değerin üstünde sonuç elde edilmesiyle de örtüşen topluluklar tespit edilmiştir. Topluluk sayısının bilinmediği durumlarda ise algoritmanın hiyerarşik versiyonu kullanılmıştır. Hiyerarşik ağacın nerede kesileceğini belirlemek için de [0, 1] aralığındaki β değişkeni kullanılmıştır ve 0,5 değeri için iyi sonuçlar elde edildiği belirtilmiştir.

2.6.2 Aşağıdan yukarı yaklaşımlar

Aşağıdan-yukarı yaklaşımlarda öncelikle merkezi köşeler ile ilk topluluklar belirlenir. Bu merkezlere en yakın köşeler de topluluklara dahil edilerek genişleme işlemi uygulanır. Sonlandırma kriteri sağlandığında ise genişleme işlemi durdurularak topluluk tespiti tamamlanır. Sonlandırma işlemi genellikle modülarite, topluluk sayısı gibi parametrelerle kontrol edilir.

Açgözlü Modülarite (Greedy Modularity)

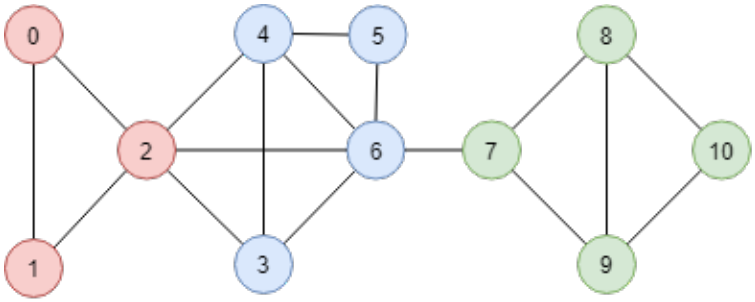
Modülarite kavramı literatüre kazandırıldıktan sonra Clauset vd. (2004) aşağıdan-yukarı yaklaşımla tasarlanan açgözlü modülarite algoritmasını önermiştir. Bu algoritmada başlangıçta her köşe kendi başına bir topluluktur ve tek bir topluluk oluşuncaya kadar modülariteyi en çok artıran topluluklar birleştirilmektedir. Çizgedeki kenar sayısı m, dereceler k ile gösterildiğinde i ve j topluluklarının birleştirilmesiyle modülaritede oluşacak değişim $\Delta Q_{i,j}$ formül 2.31 ile hesaplanmaktadır. Aralarında bağ olmayan toplulukları birleştirmek modülariteyi artırmayacağından sadece aralarında bağ olan topluluklar için modülarite hesaplanır.

$$\Delta Q_{i,j} = \begin{cases} \frac{1}{2m} - \frac{k_i k_j}{(2m)^2} & \text{, i ve j bağlıysa} \\ 0 & \text{, diğer durumlarda} \end{cases} \tag{2.31}$$

i topluluğuna bağlı kenar sayısının toplam kenar sayısına oranını $a_i = k_i/2m$'dir. Algoritma $\Delta Q_{i,j}$ ve a_i'nin ilk değerlerini hesapladıktan sonra $\Delta Q_{i,j}$ değerlerini max-heap yapısında tutar. Bu sayede hem bellek hem de zamandan kazanılır. En yüksek $\Delta Q_{i,j}$ değerini veren toplulukları birleştirerek max-heap'i günceller ve $\Delta Q_{i,j}$ ile a_i değerlerini yeniden hesaplar.

$$\Delta Q'_{j,c} = \begin{cases} \Delta Q_{i,c} + \Delta Q_{j,c} & \text{, c topluluğu i ve j'nin ikisine de bağlıysa} \\ \Delta Q_{i,c} - 2a_j a_c & \text{, c topluluğu sadece i'ye bağlıysa} \\ \Delta Q_{i,c} - 2a_i a_c & \text{, c topluluğu sadece j'ye bağlıysa} \end{cases} \tag{2.32}$$

Güncelleme işlemi için formül 2.32 kullanılarak sadece ilgili toplulukların modülarite değişimleri hesaplanır. Bu işlemler bütün köşeleri içeren tek bir topluluk oluşana kadar devam eder. Şekil 2.15'te açgözlü algoritma ile elde edilen topluluklar gösterilmiştir.



Şekil 2.15 Açgözlü modülarite algoritmasının örnek çizge üzerindeki sonucu

Sonraki yıllarda Meghanathan (2016) NOVER açgözlü modülarite algoritmasını önermiştir. Bu algoritmada kenar ağırlıkları önerilen NOVER formülü ile hesaplanır. Kenarlar belirlenen eşik değerine göre zayıf ve güçlü olmak üzere iki gruba ayrılır ve çizgeden çıkarma işleme sırasında sadece zayıf kenarlar kullanılır. Bu sayede işlem sayısı azaltılmış olur. Elde edilen topluluklar için modülarite değerleri hesaplanarak toplam modülarite değeri belirlenir ve toplam modülaritenin artması durumunda bulunan topluluklar güncellenir.

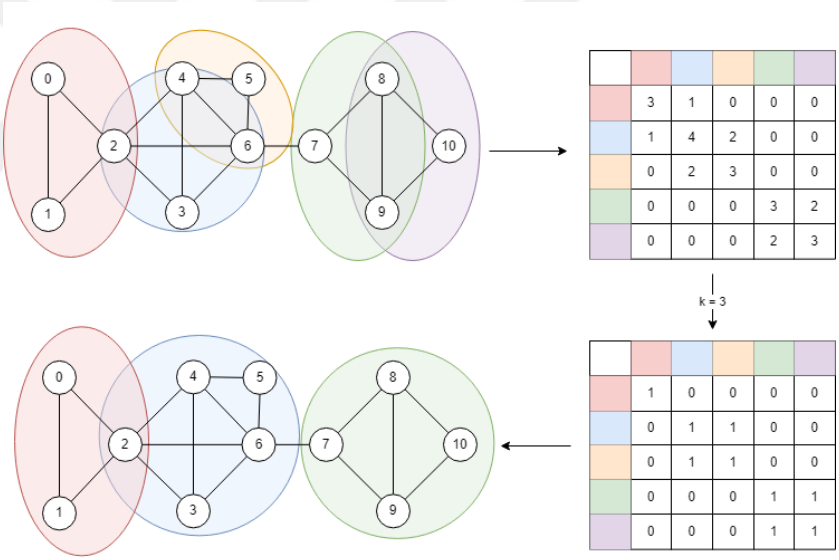
Arasteh ve Alizadeh (2019) de aç gözlü yaklaşımla tasarlanan bir algoritma önermiştir. Önerilen algoritmada her köşe farklı bit topluluğa atanır ve rastgele köşeler seçilerek komşuları değerlendirilir. Maksimum pozitif modülarite kazancını veren komşu ile birleştirme işlemi yapılır. Bu işlemler bütün köşeler değerlendirilene kadar tekrarlanır.

Klik Süzme Yöntemi (Clique Percolation Method, CPM)

Klik süzme yönteminde (Palla vd. 2005) ilk olarak kaba kuvvet yöntemi ile klikler tespit edilir. Klikler tespit edilirken başlangıçta her köşe tek başına 1-klik kabul edilir. Derecesi $k-1$'e eşit veya küçük olan köşeler sırasıyla çizgeden çıkarılarak k'dan büyük klikler bulunur. Klikler bulunduktan sonra komşu klikler tespit edilerek birleştirilir. $k-1$ köşesi örtüşen k-klikler **komşu k-klik** olarak tanımlanır. Örneğin şekil 2.16'da soldaki iki 3-klik komşu iken sağdaki iki 3-kliğin sadece 1 köşesi ortak olduğundan komşu k-klik değildir.

Şekil 2.16 Komşu (soldaki) ve komşu olmayan (sağdaki) klik örneği

Komşu kliklerin birleştirilmesi için Everett ve Borgatti (1998) tarafından önerilen yöntem ile n tane klik için nxn 'lik klik-klik örtüşme matrisi oluşturulur. Bu matrisin elemanları ilgili satır ve sütundaki kliklerin örtüşen köşe sayısını belirtmektedir. Önceden belirlenen k topluluk sayısına göre bu matrisin köşegeni üzerinde bulunan ve k'den küçük değerler ile diğer hücrelerdeki $k-1$'den küçük değerler sıfırlanır. Kalan diğer değerler 1 olarak aktarılır. Matriste karşılığı 1 olan topluluklar birleştirilerek nihai topluluklar elde edilir. Örnek çizge üzerinde $k = 3$ için algoritmanın adımları şekil 2.17'de verilmiştir.



Şekil 2.17 Klik süzme yöntemi ve örnek çizge üzerindeki sonucu

Bu yöntemde kenarların ağırlıkları dikkate alınmamaktadır. Farkas vd. (2007) yoğunluğa göre eşik değeri belirleyerek ağırlıklı çizgede çalışan versiyonunu önermiştir. Ancak eşik değerinin düşük seçilmesi çok büyük topluluklar elde etmeye, yüksek seçilmesi ise topluluk bulamamaya neden olabilmektedir. Ayrıca klik tabanlı yöntemler işlem zamanı açısından maliyetlidir. Bu nedenle, Pattabiraman vd. (2015) sezgisel yaklaşım kullanarak paralelleştirilebilen ve daha hızlı versiyonunu önermiştir.

Literatürde klik süzme yöntemini kullanan melez yöntemler de mevcuttur. Wen vd. (2019) örtüşen topluluk sayısını kısıtlayarak ve evrişimsel bir yöntem kullanarak toplulukları tespit etmiştir. Kasoro vd. (2019) ise ilk aşamada klik süzme yöntemini kullanarak toplulukları belirlemiştir. Bu aşamada topluluğu bulunamayan köşelerin için ikinci aşamada özvektör merkeziyetini kullanarak toplulukları belirleyen PercoMVC algoritmasını önermiştir.

Walktrap Algoritması

Pons ve Latapy (2005) tarafından önerilen Walktrap algoritması komşu köşeler arasındaki benzerliği hesaplamak için köşeler arasındaki uzaklığı kullanır. Birbirine en yakın köşeler en benzer köşeler olarak tanımlanır. Köşeler arasındaki mesafe ağ üzerinde rastgele yürüyüş yapan bir yürüyüşçü tarafından belirlenir. Bu yürüyüşçünün gidebileceği maksimum uzaklık algoritmanın başında tanımlanır. Genellikle 4 veya 5 birimlik mesafe tanımlanır. Yoğun ağlarda bu değerin daha küçük seçilmesi gerekir. Örnek çizge için bu algoritma da şekil 2.15'teki sonucu verir.

Daha sonra Reichardt ve Bornholdt (2006) yürüyüşçünün ağı Walktrap'ın aksine istatistiksel verilere göre dolaştığı **Spinglass** algoritmasını önermiştir. Bu yürüyüşlerde yaklaşık maksimum topluluk sayısı olan spin sayısını kullanılır. Aynı spinde gezilen köşeler aynı topluluğa atanır. Bu algoritmanın çalışma süresi fazla olduğundan genellikle küçük ağlarda kullanılmaktadır.

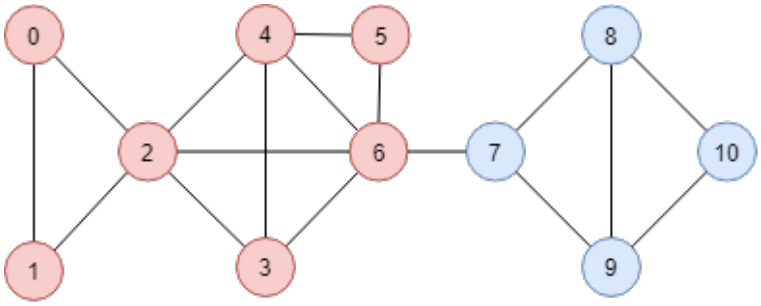
Okuda vd. (2021) de rastgele yürüyüşü köşelerin benzerliklerini belirlemek için kullanmıştır. Zhi-Xiao vd. (2016) Page Rank algoritması ile köşelerin yoğun olduğu yerlere ve köşelerin konumlarına göre örtüşen toplulukları belirlemektedir. Bunun için köşeleri tepede, eğimde ve düzlükte olarak gruplandırmıştır. Tepedeki köşeler topluluk merkezlerini, eğimdekiler topluluk içindeki köşeleri ve düzlüktekiler örtüşen köşeleri temsil etmektedir ve merkezi köşeler bütün köşeler dikkate alınarak güncellenmektedir. Gao vd. (2021), Page Rank algoritmasının doğruluğunu arttırmak için yürüyüşçüleri alakasız topluluklardan uzak tutan bir yöntem önermiştir. Bunun için yürüyüşçüyü alakasız bir topluluğa girdiğinde topluluktan çıkmaya zorlamışlardır.

Etiket Yayılımı (Label Propagation) Algoritması

Etiket yayılımı yönteminde (Raghavan vd. 2007) başlangıçta her köşenin özgün bir etiketi vardır. Her adımda komşular arasındaki en yaygın etikete göre etiket güncelleme işlemi yapılır. Bu koşulu sağlayan birden fazla köşe olması durumunda, köşeler arasında rastgele seçim yapılır ve bu adım durma kriteri sağlanıncaya kadar devam eder. Durma kriteri olarak genellikle köşelerin etiketlerinin değişmemesi kuralı kullanılmaktadır. Bu yöntem yönlü çizgelerde çalışmamaktadır.

Komşuların etiketlerinin aynı anda güncellendiği senkron yöntemlerde sonsuz döngüye girme problemi görülmektedir. Bu nedenle rastgele belirlenen sırayla etiketleri güncelleyen asenkron yöntemler geliştirilmiştir. Ancak asenkron yöntemler işlem zamanı açısından senkron yöntemlere göre daha maliyetlidir. Daha sonra iki yönteminde avantajlarından faydalanmak için yarı senkron yöntemler önerilmiştir.

Yarı senkron yöntemde köşeler minimum sayıda etiket kullanılarak ve aynı etikete sahip iki komşu yan yana gelmeyecek şekilde renklendirme işlemi yapılır. Aynı renge sahip köşeler eş zamanlı güncellenir ancak renkler sıralı bir şekilde seçilir. Cordasco ve Gargano (2010) bu yöntemle tasarlanan bir algoritma önermiştir. Örnek çizge için bu algoritmanın bulduğu topluluklar şekil 2.18'de gösterilmiştir.



Şekil 2.18 Etiket yayılımı yöntemi algoritmasının örnek çizge üzerindeki sonucu

Etiket yayılımı yöntemindeki rastgele seçim, algoritmanın her çalıştığında farklı bir sonuç üretmesine neden olmaktadır. Bu durumu önlemek için literatürde pek çok çalışma yapılmıştır. Jokar ve Mosleh (2018) kenarların ağırlıklarını belirleyip rastgele seçim yerine en yüksek ağırlıkları kullanmıştır. Yang vd. (2020) ve You vd. (2020) benzerlik değer-

lerini kullanmıştır. Hu vd. (2017) ise köşelerin rollerini tanımlayıp (merkezi, çevresel, sıkı örülmüş, köprü gibi) bu rollere göre seçim yapmıştır. Zhang, Liu vd. (2020) ise hem değişkenlik sorununu çözmek hem de doğruluğu artırmak için etiketi güncelleme, kabul etme ve yayınlama adımları tanımlamış ve bu adımlar için kurallar belirlemiştir. Bouyer ve Roghani (2020) doğruluğu artırmak için düşük dereceli köşelerden başlamıştır ve etiketleri bu köşelerden yaymıştır. Daha sonra birleştirme işlemi uygulamıştır.

Gregory (2010) örtüşen topluluklarda çalışan Topluluk Örtüşme Yayılım Algoritmasını (Community Overlap Propagation Algorithm, COPRA) önermiştir. Xie vd. (2011) bir köşeye birden fazla etiket atanmasına izin veren Konuşmacı-Dinleyici Etiket Yayılım Algoritmasını (Speaker-Listener Label Propagation Algorithm, SLPA) önermiştir. Coscia vd. (2012) ise her köşenin kendi topluluğunu oylayarak seçmesine izin veren Bir Ağın Modüler Organizasyonunun Demokratik Tahmini (Democratic Estimate of the Modular Organization of a Network, DEMON) algoritmasını önermiştir.

Etiket yayılımı algoritması literatürde sıklıkla çalışılmıştır. Lu vd. (2019) etiketleri önemlerine göre sıralayıp artan şekilde güncelleyen bir etiketleme stratejisi kullanarak Komşu Köşe Etkili Etiket Yayılımı Algoritması'nı (Label Propagation Algorithm with Neighbor Node Influence, LPANNI) önermiştir. Kouni vd. (2020) de benzer şekilde köşelerin önem derecelerini kullanmıştır. Joghan vd. (2019) kenarların ağırlıklarını Girvan-Newman algoritmasını kullanarak belirlemiş ve sonrasında etiket yayılımı algoritmasını uygulamıştır. Hosseini ve Rezvanian (2020) karınca kolonisi optimizasyon algoritmasıyla beraber modülariteyi ve etiket yayılımı algoritmasını kullanmıştır.

Louvain Algoritması

Louvain algoritması modülarite tabanlı bir algoritmadır. Formül 2.33'te verilen modülarite kazancına göre köşeleri birleştirerek ilerler (Blondel vd. 2008). Formül 2.33'de C topluluğundaki ağırlıkların toplamı $\sum_{in}$, C'deki köşelere bağlı kenarların ağırlıklarının toplamı $\sum_{tot}$, i köşesine bağlı kenarların ağırlıkların toplamı k_i , i köşesi ile C topluluğu arasındaki kenarların ağırlıklarının toplamı $k_{i,in}$ ve ağdaki bütün kenarların ağırlıklarının toplamı m ile ifade edilmektedir.

$$\Delta Q = \left[\frac{\sum_{in} + 2k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right] \tag{2.33}$$

Bu algoritma iki aşamadan oluşur. Öncelikle her köşe sırasıyla bütün komşularının topluluklarına eklenerek modülarite kazancı hesaplanır. Maksimum pozitif kazancı veren köşeler aynı topluluğa atanır. Artık kazanç elde edilemeyince ilk aşama tamamlanır.

İlk aşama tamamlandıktan sonra köşelerin toplulukları temsil ettiği yeni bir ağ oluşturularak ikinci aşama başlar. İki topluluk arasında kalan kenarların ağırlıkları toplanarak iki topluluğu temsil eden köşelerin arasındaki kenarın ağırlığı belirlenir. Aynı topluluğa ait köşelerin arasında kalan kenarlar ise köşeden kendisine olan bir kenar ile gösterilir ve kenarların ağırlıklarının toplamı bu kenarın ağırlığı olarak belirlenir. Yeni ağ oluşturulduktan sonra ikinci aşama da tamamlanır ve bu yeni ağ üzerinde birinci aşamaya geri dönülür. Bu işlemler maksimum modülaritede bir değişiklik olmayana kadar tekrarlanır. Örnek çizge için Louvain algoritması da şekil 2.15'te olduğu gibi açgözlü modülarite algoritması ile aynı sonucu vermektedir.

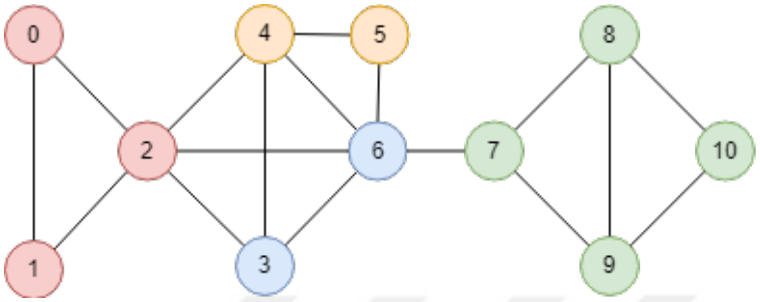
Bu yöntem modülarite kullandığı için toplulukları başarılı bir şekilde tespit eder ancak bellek problemi vardır. Bu nedenle Mohammadi vd. (2020) bellek problemini çözmek için CUDA kullanmıştır.

Akışkan Topluluklar (Fluid Communities)

Akışkan topluluklar algoritması Pares vd. (2017) tarafından önerilen ve bağlı çizgeler üzerinde çalışan bir algoritmadır. İlk adımda, belirtilen topluluk sayısı kadar rastgele köşe seçilir ve bu köşeler sırasıyla birer topluluğa atanarak oluşturulan toplulukların yoğunlukları hesaplanır. Topluluğun yoğunluğu c_n'nin c topluluğundaki köşe sayısını temsil ettiği $1/c_n$ formülü ile bulunur. 1 köşeden oluşan topluluk maksimum yoğunluğa sahip topluluktur ve yoğunluğu 1'dir.

Algoritma rastgele bir köşeden başlayarak bütün köşeleri rastgele gezer. Her köşe için komşu köşeleri arasında en sık geçen topluluklar aday topluluklar olur ve bu topluluklar arasında maksimum yoğunluğa sahip topluluk en iyi topluluk olarak belirlenir. En iyi top-

luluğun birden fazla olması durumunda topluluklar arasından rastgele seçim yapılır. Topluluklarda değişiklik olduğu sürece köşeler karıştırılarak en baştan başlanır ve işlemler tekrarlanır. Örnek çizge için topluluk sayısı 3 seçildiğinde şekil 2.15'teki, 4 seçildiğinde şekil 2.19'daki sonuç elde edilmektedir.



Şekil 2.19 Akışkan topluluklar algoritmasının örnek çizge üzerindeki sonucu

Genişleme Tabanlı Yöntemler

Genişleme tabanlı yöntemlerde genellikle merkezi köşelerden başlayarak bu köşelerin ait olduğu topluluklar genişletilerek topluluk tespiti yapılır. Lancichinetti vd. (2009) bu yaklaşımı kullanarak hem örtüşen toplulukları hem de hiyerarşik yapıyı bulan LFM algoritmasını önermiştir. Merkezi köşeler rastgele belirlenmiş ve topluluklar yerel optimizasyon kullanan bir uygunluk fonksiyonu ile genişletilmiştir.

Genişleme tabanlı yöntemlerde merkezi köşelerin seçimi, bulunan toplulukları etkilemesi nedeniyle çok önemlidir. Buna bağlı olarak literatürde farklı köşe seçimi yöntemleri kullanılmıştır. Lee vd. (2010) rastgele merkezi köşe seçimi yerine maksimal klikleri kullanan Açgözlü Klik Genişleme (Greedy Clique Expansion, GCE) algoritması önermiştir. Choumane vd. (2020) merkezi köşeleri en yüksek komşuluk örtüşme değerine göre belirleyen Çekirdek Genişleme (Core Expansion) algoritmasını önermiştir. Aghaalizadeh vd. (2021) önemli köşelerin birbirinden uzak olduğunu ancak düşük önem derecesine sahip köşelerle çevrili olduğunu belirtmiştir. Bu nedenle Kirşof (Kirchhoff) kanunundan yola çıkmış ve komşu köşelerin dereceleri ile ters orantı kurarak merkezi köşeleri belirlemiştir.

Hollocou vd. (2016) merkezi köşeleri seçtikten sonra Page Rank kümeleme skoruna göre bu köşeleri genişleten Walkscan algoritmasını önermiştir. X. Wang vd. (2017) genişle-

meyi yerel arama ile yapan bir yöntem önermiştir. Ağırlıklar yapısal benzerlikler kullanılarak belirlenmiştir ve bu ağırlıklar azalan şekilde sıralandıktan sonra merkezi düğümler belirlenmiştir. Her merkezi düğümün tüm komşularına bakarak merkezi yoğunluğu hesaplamış ve merkezi düğümün topluluğuna bir düğüm eklenip eklenmeyeceğine karar vermiştir. Bütün komşular değerlendirildiğinde algoritma sonlanmaktadır. Long (2018) ise ilk olarak kenar yoğunluklarını hesaplayan ve belli bir eşik değerinin üstündeki kenarlarla ağın iskeletini çıkaran bir yöntem önermiştir. İskeleti belirleyen kenarların ucundaki köşeler merkezi köşeleri oluşturmaktadır. Merkezi köşeler içinden olası örtüşen köşeler bularak çoğaltılır ve köşeler aidiyet derecelerine göre topluluklara atanır. Son olarak aynı topluluğa ait çoğaltılmış düğümler birleştirilir.

Genişleme tabanlı yöntemlerde benzerlik ölçütleri de sıklıkla kullanılmaktadır. T. Wang vd. (2015) merkezi düğümleri tespit etmek için kosinüs benzerliğine benzer bir metrik önermiştir. Önerilen metrik iki köşe arasındaki ortak köşe sayısı yerine en kısa yolun uzunluğunu kullanmaktadır. Z. Chen vd. (2015) ise köşeler arasındaki Jaccard benzerliklerini hesapladıktan sonra en yüksek benzerliğe sahip iki köşeden başlayarak diğer köşeleri sırasıyla gruplandırmış ve toplulukları tespit etmiştir.

Kenar Tabanlı Yöntemler

Birleştirme tabanlı yöntemlerin çoğu köşe tabanlı olmakla beraber kenarları kullanarak topluluk tespiti yapan yöntemler de mevcuttur. Bu yöntemler köşeler yerine kenarları gruplandırır ve daha sonra kenar topluluklarını köşe topluluklarına çevirerek toplulukları belirler.

Örtüşen topluluklarda köşeler birden fazla topluluğa ait olabilirken kenarlar bir topluluğa ait olabilmektedir. Bu nedenle Ahn vd. (2010) Jaccard benzerliği ve tek bağlantı kümeleme algoritmasını kullanarak kenarları kümeleyen bir yöntem önermiştir.

Lim vd. (2014) ise öncelikle ağı kenar ağına çevirmiş ve bağlar arasındaki benzerlikleri kullanan LinkSCAN algoritmasını önermiştir. Kim vd. (2018) LinkSCAN'ın örtüşen topluluk tespiti yeteneği ve BlackHole'un topluluklar arasındaki karışmayı önleyen özelliğini kullanan LinkBlackHole algoritmasını önermiştir.

Huang vd. (2016) kosinüs benzerliği kullanmıştır ve hızlı arama kümeleme (fast search clustering) algoritması ile kenarları kümelemiştir. Daha sonra kenar topluluklarını köşe topluluklarına çevirmişlerdir.

İstatistiksel Yöntemler

Örtüşen topluluk tespitinde istatistiksel yöntemler de kullanılmaktadır. Zhang vd. Negatif Olmayan Matris Ayrıştırma Yöntemi (Non-negative Matrix Factorization, NMF) yöntemi ile örtüşen toplulukları tespit etmiştir. Bu yöntemde topluluk sayısı yerel modülarite kullanılarak belirlenmektedir. Psorakis vd. (2011) ise NMF'nin işlemsel performansını artıran Bayesian Negatif Olmayan Matris Çarpanlara Ayırma (Bayesian Non-negative Matrix Factorization, BNMF) algoritmasını önermiştir. Zhang ve Yeung (2012) NMF'nin kümeleme performansını üç faktör kullanarak artıran (Bounded-NMF,(BNMTF) algoritmasını önermiştir. Ancak matrisi çarpanlarına ayırma (matrix factorization) yöntemleri seyrek ağlarda başarılı olsa da karmaşık ağlar için çok maliyetlidir.

Gopalan ve Blei (2013) k sayıda topluluk olduğunu ve her düğümün bu topluluklarla olasılıksal bir ilişkisi olduğunu varsayan Bayes tabanlı bir model geliştirmiştir. Bunun için k tane topluluktan birine işaret eden topluluk göstergelerini kullanmışlardır. Her bir düğüm çifti için göstergeler aynı topluluğa işaret ediyorsa algoritma düğümleri yüksek olasılıkla bağlamaktadır. Jin vd. (2015) hem köşe hem de kenar topluluklarını değerlendiren bir yöntem önermiştir. Bu modelin parametreleri NMF, topluluk sayısı ortak karar, topluluklar ise açgözlü optimizasyon algoritması ile belirlenmiştir. Zhou vd. (2019) köşelerin diğer köşelere etki gücünü hesaba katan Karma Üyelik Stokastik Blok Modeli (Mixed-Membership Stochastic Block Model) algoritmasını önermiştir.

3. MATERYAL ve YÖNTEM

Bu çalışmada topluluk, genel tanıma bağlı kalarak üyeleri arasındaki bağın diğer üyelerle bağlarına göre daha yoğun olan yapılar olarak kabul edilmiştir. Önerilen algoritmalarda bu yapıların tespit edilmesi amaçlanmıştır. Ağlar çizge ile modellenmiş, ağırlıksız ve yönsüz çizgeler üzerinde ayrık ve örtüşen topluluk tespitleri yapılmıştır. İki algoritmada da aşağıdan-yukarı yaklaşım uygulanmıştır. Öncelikle ayrık topluluk tespiti üzerinde çalışılmış ve benzerlik ölçütleri incelenmiştir. Daha sonra başlangıç toplulukları kullanmanın etkisi araştırılmıştır. Önerilen ayrık topluluk tespiti algoritması ile başarılı sonuçlar elde edildikten sonra örtüşen topluluk tespiti algoritması tasarlanmıştır.

3.1 Veri Kümesi

3.1.1 Sentetik veri kümeleri

Bu çalışmadaki sentetik veriler Lancichinetti vd. (2009) tarafından önerilen LFR-benchmark ile oluşturulmuştur. LFR benchmark oluşturmak için kullanılan parametrelerin açıklamaları çizelge 3.1'de verilmiştir.

Çizelge 3.1 LFR parametreleri

-N	köşe sayısı
-k	ortalama derece
-maxk	maksimum derece
-mut	topoloji karıştırma parametresi
-muw	ağırlık karıştırma parametresi
-beta	ağırlık dağılımı için üs
-t1	derece dağılımı için negatif üs
-t2	topluluk boyutu dağılımı için negatif üs
-minc	minimum topluluk boyutu
-maxc	maksimum topluluk boyutu
-on	örtüşen köşe sayısı
-om	örtüşen köşelerin üyelik sayısı
-C	ortalama kümeleme katsayısı

3.1.2 Gerçek veri kümeleri

Bu çalışmada çeşitli alanlardan ve literatürde sıkça tercih edilen gerçek veri kümelerinden bazıları kullanılmıştır. Bu veri kümelerinin özellikleri çizelge 3.2'de verilmiştir.

Çizelge 3.2 Gerçek veri kümeleri ve açıklamaları

Veri Kümesi	Köşe Sayısı	Kenar Sayısı	Topluluk Sayısı	Açıklama
Karate	34	78	2	Üniversitedeki karate kulübünün üyeleri arasındaki sosyal arkadaşlık ağıdır. Görüş farklılığı sonucu ayrılmış iki topluluktan oluşmaktadır.
Football	115	613	12	2000 Sonbahar sezonunda Division IA kolejleri arasındaki Amerikan futbolu oyunlarının ağıdır.
Dolphins	62	159	2	62 yunusun bulunduğu bir ağdır. Sık sık bir araya gelen yunuslara göre topluluklar belirlenmiştir.
Polbooks	105	441	3	Siyasi kitap ağıdır. Kitaplar arasındaki kenarlar, aynı alıcılar tarafından sıklıkla birlikte satın alınan kitapları bağlamaktadır.
Les Misérables	77	254	-	Sefiller romanındaki karakterlerden oluşan bir ağdır. Bu ağdaki topluluklar romanda beraber geçen kararterden oluşmaktadır.
NetScience	379	914	-	Ağ teorisi ve deneyi üzerinde çalışan bilim adamlarının ortak yazarlık ağıdır.
Email	1133	5451	-	Univeristy Rovira i Virgili (Tarragona) üyeleri arasındaki e-posta ağıdır.
Yeast	2375	11693	-	Protein-protein etkileşim ağıdır.
Web spam	4767	37375	-	Spam internet sayfalarının bulunduğu ağdır.
Router	5022	6258	-	Otonom sistemler düzeyinde internet yapısının anlık görüntüsünü gösteren ağdır.
Bio dmela	7393	25569	-	Protein-protein etkileşim ağıdır.
PGP	10680	24316	-	PGP kullanıcılarının grafiğindeki en büyük bağlantılı bileşendir.

3.2 Ayrık Topluluk Tespiti

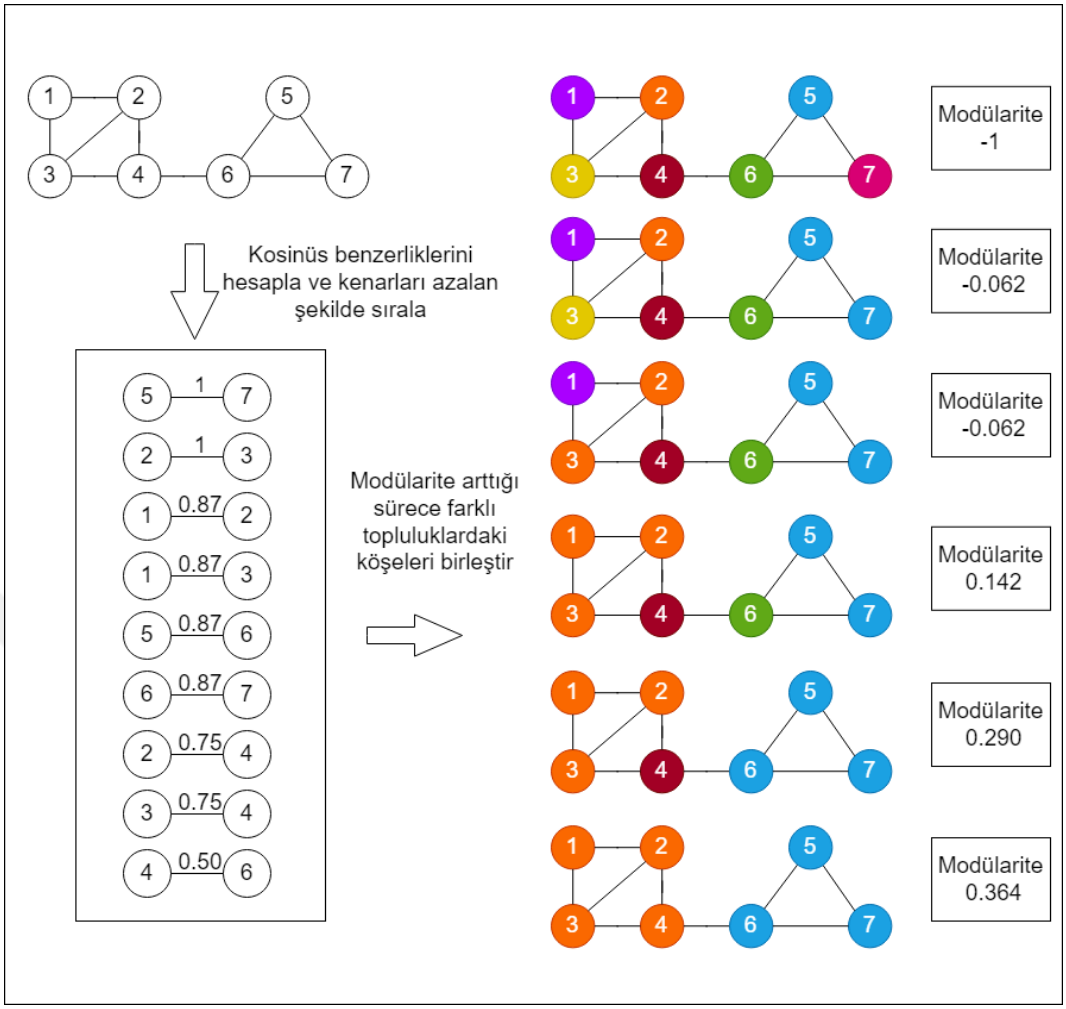
Bu bölümde ayrık topluluk tespiti için önerilen algoritma açıklanarak yapılan deneysel çalışmalar ile elde edilen sonuçlara yer verilmiştir. Ayrık topluluk tespiti algoritması test edilirken bölüm 3.1.2'de sunulan gerçek veri kümeleri kullanılmıştır. Ayrıca aşağıdan yu-

karı yaklaşımlarda algoritmaların sonucunu en çok etkileyen merkezi topluluk kullanımı ile kenar ağırlıklarını belirlemede kullanılan benzerlik ölçütlerinin etkisi incelenmiştir.

3.2.1 Önerilen algoritma

Ayrık topluluk tespiti algoritması geliştirilirken benzer köşelerin aynı toplulukta bulunması varsayımından yola çıkılmıştır. Bu amaçla köşeler arasındaki benzerlikler hesaplanarak benzerlik oranı yüksek köşeler aynı topluluklara dahil edilmiştir. Köşelerin kendileri de komşuluk listelerine dahil edilerek kosinüs benzerlikleri hesaplanmış ve kenarların ağırlıkları belirlenmiştir. Böylece ağırlıksız çizge ağırlıklı çizgeye dönüştürülmüştür. Daha sonra belirlenen ağırlıklar büyükten küçüğe sıralanmıştır ve en büyük ağırlığa sahip kenardan başlayarak bu kenarın birbirine bağladığı köşe ikilisi için birleştirme işlemi yapılmıştır. Bu iki köşe mevcut durumda aynı topluluğa ait ise birleştirme işlemine gerek olmadığından bir sonraki en büyük ağırlık değerine sahip kenardan devam edilmektedir. Birleştirme işlemi sonucunda modülarite hesaplanmış ve modülaritede artış olması durumunda birleştirme işlemi kabul edilerek yeni topluluklar belirlenmiştir. Aksi takdirde yapılan birleştirme işlemi geçersiz sayılmıştır. Bütün kenarların birbirine bağladığı köşe ikilileri değerlendirildikten sonra algoritma sonlandırılmış ve son topluluklar elde edilmiştir.

Önerilen algoritma şekil 3.1'de bir örnek üzerinde gösterilmiştir. İlk adımda köşeler arasındaki benzerlikler hesaplanarak büyükten küçüğe sıralandıktan sonra benzerlik oranı en yüksek köşelerin 5 ile 7 ve 2 ile 3 olduğu görülmektedir. Çünkü bu köşelerin bütün komşuları ortaktır. 5 ile 7 ve 2 ile 3 köşeleri birleştirildiğinde iki topluluğa ait merkez belli olmaktadır. Birleştirme işlemleri yapılırken modülariteler hesaplanmakta ve birleştirme işleminin modülariteyi artırıp artırmayacağı kontrol edilmektedir. Sonrasında benzerlik oranı en yüksek ikililerden 1 ve 2 numaralı köşeler birleştirilmekte ve modülaritenin arttığı gözlemlenmektedir. 1 ile 3 köşesi artık aynı toplulukta olduğundan bu ikili için bir işlem yapılmamaktadır. Benzer şekilde 5 ile 6 köşeleri birleştirildikten sonra 6 ile 7 numaralı köşeler aynı topluluğa ait olduğundan herhangi bir işlem yapılmamaktadır. 2 ile 4 köşeleri birleştirildikten sonra da 3 ile 4 aynı topluluğun üyeleri olmaktadır. Bu durumda



Şekil 3.1 Ayrık topluluk tespiti algoritmasının çalışma şekli

elde edilen modülarite değeri 0,364'tür. Son adımda 4 ile 6 köşeleri birleştirildiğinde ise hesaplanan modülarite de düşüş olduğundan birleştirme işlemi iptal edilir. Bütün kenarlar değerlendirildikten sonra algoritma sonlanmakta ve $\{1, 2, 3, 4\}$ ile $\{5, 6, 7\}$ toplulukları elde edilmektedir.

Algoritma 1'de u köşesinin kendisinin de dahil olduğu komşularının listesi N_u, u köşesinin ait olduğu topluluk ise C_u ile gösterilmektedir. $Q(C)$ ise C toplulukları için formül 2.23 ile hesaplanan modülarite değeridir. Algoritmanın karmaşıklığı incelendiğinde ise kosinüs benzerlikleri $O(n^3)$, sıralama işlemi $O(m\,lg\,m)$, *for* döngüsü $O(m\,n)$ zamanda, algoritma ise toplam $O(n^3)+O(m\,lg\,m)+O(m\,n)$ zamanda çalışmaktadır. Bu nedenle önerilen algoritmanın çalışma zamanı $O(n^3)$'tür.

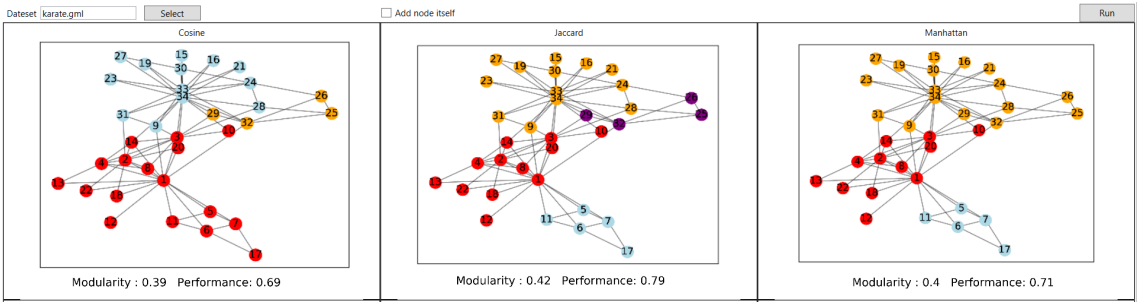
Algoritma 1 Ayrık topluluk tespiti algoritması

Input : $G = (V, E)$ ağırlıksız basit çizgesi
Output: C topluluk listesi

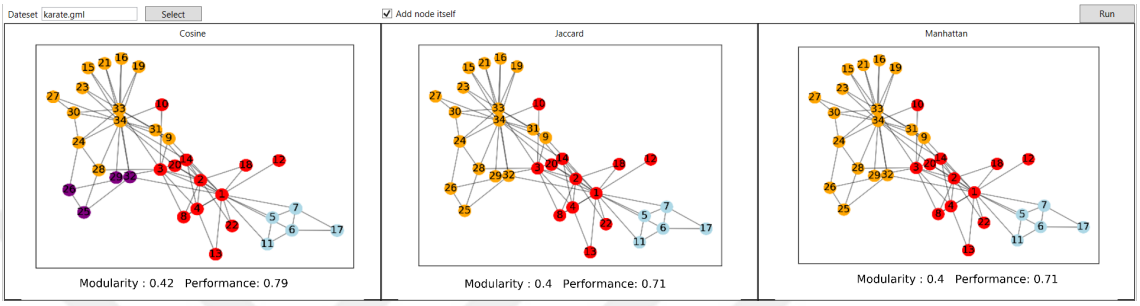
for her $(u, v) \in E$ için **do** ▷ u ve v köşeleri arasındaki benzerliği hesapla
    $w(u, v) = |N_u \cap N_v| / \sqrt{|N_u||N_v|}$
end for
$C = \phi$
for her $v \in V$ **do** ▷ tek köşeden oluşan $\{v\}$ topluluklarını C topluluk listesine ekle
    $C = C \cup \{v\}$
end for
$(u, v) \in E$ kenarlarını $w(u, v)$ ağırlıklarına göre azalan sırada sırala
$Q' = -1$
$Q'' = -1$
for her $(u, v) \in E$ **do**
    $C' = C$
    if $C_u \neq C_v$ **then**
        $C' \leftarrow C' - C_v$
        $C_v \leftarrow C_v \cup C_u$
        $Q'' = Q(C')$
    end if
    if $Q'' >= Q'$ **then**
        $C = C'$
        $Q' = Q''$
    end if
end for
return C

3.2.2 Deneysel çalışma

Ayrık topluluk tespiti algoritması kosinüs benzerliğinin dışında Jaccard ve Manhattan uzaklıkları ile de test edilmiştir. Bu metrikler kullanılarak *karate* veri kümesi üzerinde bulunan topluluklar şekil 3.2'de verilmiştir. Köşenin kendisinin komşuluk listesine eklendiği ve kosinüs benzerliği kullanılarak bulunan topluluklar, köşenin kendisinin komşuluk listesine dahil edilmediği ve Jaccard benzerliği kullanılarak bulunan topluluklarla aynıdır. Ancak Jaccard benzerliği için köşenin komşuluk listesine dahil edilmesi bulunan topluluk sayısını, modülarite ve performans değerlerini düşürmüştür. Manhattan uzaklığı kullanıldığında ise farklı komşu sayıları değerlendirildiği için köşenin kendisinin komşuluk listesine dahil edilip edilmemesi sonucu etkilememektedir.



(a) Köşe komşuluk listesine dahil edilmediğinde



(b) Köşe komşuluk listesine dahil edildiğinde

Şekil 3.2 *karate* verisi üzerinde benzerlik ölçülerinin karşılaştırması

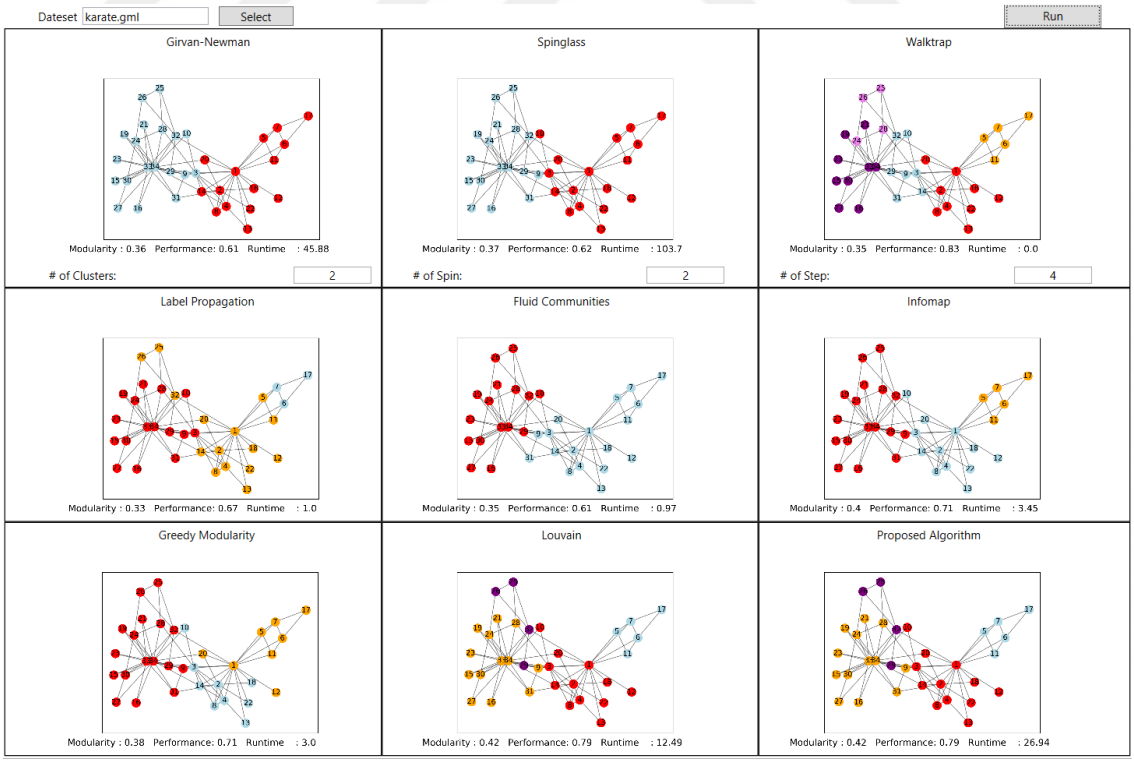
Bu metrikler, kosinüs benzerliğinde köşenin kendisi de komşu listesine eklenerek Jaccard benzerliğinde ise eklenmeden çizelge 3.2'de verilen veri kümelerinde de test edilmiştir. Çizelge 3.3'te verilen sonuçlar incelendiğinde *football* veri kümesinde bütün metriklerle aynı sonucun elde edildiği görülmektedir. Diğer veri kümelerinde ise en yüksek NMI değerleri, her veri kümesi için farklı bir metrikle elde edilmiştir. *karate* veri kümesinde daha önce de belirtildiği gibi kosinüs ve Jaccard benzerlikleri kullanılarak aynı modülarite ve performans değerleri elde edilmiştir. *polbooks* ve *dolphins*'te kosinüs benzerliği ile yüksek modülarite değeri elde edilmektedir. *dolphins*'te kosinüs benzerliği en yüksek performans değerini verirken *polbooks*'ta Manhattan en yüksek performans değerini vermiştir.

Bu deney bütün metriklerde en iyi sonucu veren bir benzerlik ölçütü olmadığını göstermektedir. Ancak kosinüs benzerliği veri kümelerinde yüksek sonuç verdiği için önerilen algoritmada da kosinüs benzerliği kullanılmıştır. Önerilen algoritmanın diğer algoritmalarla karşılaştırması şekil 3.3'te verilmiştir. Şekil 3.3'te de görüldüğü gibi her algoritmanın bulduğu topluluklar birbirinden farklıdır. Bu da farklı topluluk tanımlarından kaynak-

Çizelge 3.3 Geçek veri kümeleri üzerinde benzerlik ölçütlerinin sonuçları

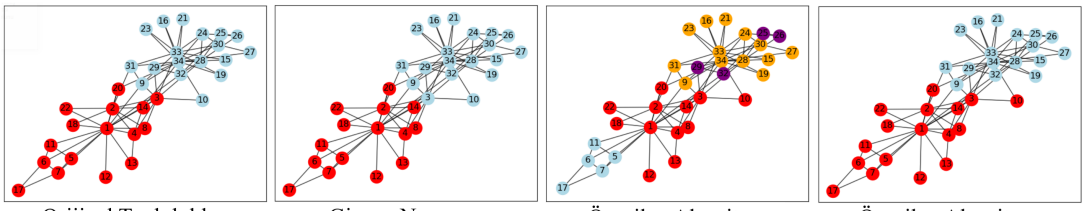
Veri Kümesi	Benzerlik Ölçeği	Modülarite	Performans	NMI	Topluluk Sayısı
karate	jaccard	**0,4156**	**0,7861**	0,6301	4
	cosine	**0,4156**	**0,7861**	0,6301	3
	manhattan	0,4020	0,7112	**0,7112**	3
football	jaccard	0,6044	0,9246	0,8734	9
	cosine	0,6044	0,9246	0,8734	9
	manhattan	0,6044	0,9246	0,8734	9
dolphins	jaccard	0,5153	0,7911	**0,6831**	4
	cosine	**0,5203**	**0,8091**	0,6570	4
	manhattan	0,5034	0,8006	0,5762	4
polbooks	jaccard	0,5223	0,7286	0,5585	4
	cosine	**0,5267**	0,7313	**0,5930**	4
	manhattan	0,5157	**0,7476**	0,5414	4

lanmaktadır. En yüksek modülarite ve performans değerini veren algoritmaların bulduğu topluluklar incelendiğinde gerçek topluluklardan farklı olduğu görülmektedir.



Şekil 3.3 *karate* verisi için algoritmaların sonuçları

Girvan-Newman algoritması *karate* veri kümesi üzerinde orijinale en yakın sonucu veren algoritmadır. Girvan-Newman algoritması ile elde edilen modülarite, performans ve NMI değerleri sırasıyla 0,36, 0,6114 ve 0,8333'tür. Önerilen algoritma ile bu algoritmanın sonuçları karşılaştırıldığında, önerilen algoritma ile elde edilen performans ve modülarite değerlerinin daha yüksek olduğu görülmektedir. Ancak *karate* verisi iki topluluk içermesine rağmen önerilen algoritma ile dört topluluk bulunduğundan NMI değeri daha düşüktür.



Şekil 3.4 *karate* verisi için Girvan-Newman algoritması ve ayrık topluluk tespiti algoritması ile elde edilen sonuçlar

Bu noktada topluluk sayısı önceden verilseydi algoritmanın başarısı ne olurdu sorusuna cevap aramak için bir deney daha yapılmıştır. Önerilen algoritmanın durma kriteri olarak modülaritedeki azalmama yerine verilen topluluk sayısına ulaşma belirlenmiştir. Bu nedenle istenilen sayıdaki topluluklar bulunana kadar birleştirme işlemi yapılmaya devam edilmiştir. Şekil 3.4'te verilen sonuçlarda görüldüğü gibi 10 numaralı köşe hariç bütün köşeler doğru bir şekilde topluluklara ayrılmıştır. Girvan-Newman algoritması ise sadece 3 numaralı köşeyi yanlış topluluğa ayırmaktadır. Bu durumda önerilen algoritma ile elde edilen modülarite, performans ve NMI değerleri Girvan-Newman algoritması ile elde edilen değerden daha yüksektir (Çizelge 3.4). Ancak daha yüksek modülarite elde edilmek isteniyorsa bu veri kümesinde ikiden fazla topluluk tespit edilmelidir.

Önerilen algoritma k toplulukla çalıştırıldığında diğer veri kümelerinde elde dilen sonuçlar da çizelge 3.4'te verilmiştir. *dolphins* verisinde orijinal topluluklar olduğu gibi tespit edilmiştir. *polbooks* verisinin modülarite ve NMI sonuçlarında büyük bir değişiklik olmadığı ancak performansın azaldığı gözlemlenmemiştir. *football* verilerinde ise önerilen algoritma ile bulunan topluluk sayısı gerçek topluluk sayısından daha az olduğu için bu

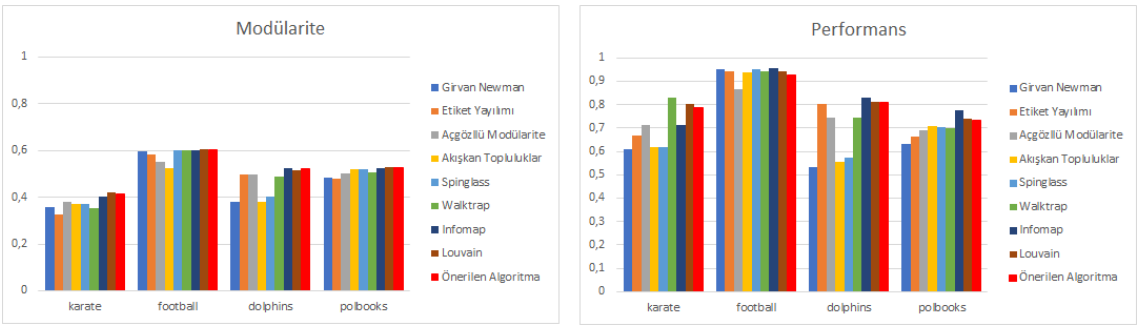
veri kümesi karşılaştırmaya dahil edilememiştir.

Çizelge 3.4 Önerilen algoritma k-toplulukla çalıştırıldığında elde edilen sonuçlar

Veri Kümesi	**Modülarite**	**Performans**	**NMI**	**k**
Karate	0.3718	0.6185	0.8371	2
Dolphin	0.3735	0.5219	1.0	2
Polbooks	0.5216	0.7119	0.5934	3

Elde edilen sonuçlar aynı zamanda değerlendirme metriklerinin de yeterince iyi olmadığını göstermektedir. Gerçek topluluklar bire bir tespit edildiğinde bile elde edilen modülarite ve performans değerlerinin yeterince yüksek olmadığı görülmektedir. Modülarite ve performans değerlerinde gözlemlenen bu durum metriklerin kullandığı topluluk tanımından kaynaklanmaktadır. NMI ise gerçek toplulukların bilindiği durumlarda karşılaştırma yapmak için uygundur.

Kullanılan gerçek verilerin çoğunda topluluklar bilinmediğinden bu çalışmada NMI yerine modülarite ve performans metrikleri ile karşılaştırma yapılmıştır. Önerilen algoritma ve diğer algoritmaların gerçek veri kümeleri üzerindeki sonuçları şekil 3.5'te verilmiştir. Sonuçlar incelendiğinde önerilen algoritmanın bütün verilerde yüksek modülarite ve performans sonucu verdiği görülmektedir.



Şekil 3.5 Ayrık topluluk tespiti algoritmaların modülarite ve performans sonuçları

Önerilen algoritma, başarılı sonuçlar vermesine rağmen işlem zamanı kosinüs benzerliğinin hesaplanma süresine bağlı olduğu için $O(n^3)$'te çalışmaktadır. Ayrıca modülarite hesabı da işlem zamanı açısından maliyetlidir. Bu nedenle algoritmayı hızlandırmak için başlangıç toplulukları oluşturulmuş ve algoritmanın başarısına etkisi test edilmiştir. Bu

amaçla Algoritma 2 ile merkezi köşeler tespit edilmiştir. Bulunan merkezi köşelere komşu olan köşeler de bu köşelerin topluluk listelerine eklenmiştir. Daha sonra birden fazla topluluğa dahil edilen köşeler topluluklardan çıkarılarak hangi topluluğa dahil edileceğine Algoritma 1 ile karar verilmiştir. Algoritma 2'deki V_s sıralı köşe listesini temsil etmektedir. n_u ise u köşesinin komşularının listesidir.

Algoritma 2 Merkezi köşeleri bulma algoritması

Input : $G = (V, E)$ ağırlıksız basit çizgesi
Output: l merkezi köşe listesi

$l = \phi$
köşeleri derecelerine göre büyükten küçüğe doğru sıralayarak $V_s = \{v_1, v_2, ...v_n\}$'yi oluştur
$l = \{v_1\}$
$V_s \leftarrow V_s - v_1$
$i = 1$
while $V_s \neq \phi$ **do**
    for her $q \in n_{v_{i-1}}$ **do**
        if $q \in V_s$ **then**
            $V_s \leftarrow V_s - q$
        end if
    end for
    $V_s = \{v_1, v_2, ...v_k\}$ olacak şekilde indisleri güncelle
    if $|V_s| > 0$ **then**
        $l \leftarrow l \cup v_1$
    end if
    $i = i + 1$
end while
return l

Başlangıç köşeleri kullanıldığında ve topluluk sayısı bilindiğinde bulunan topluluklar için elde edilen sonuçlar çizelge 3.5'te verilmiştir. *karate* ve *dolphins* verilerinde orijinal topluluklar olduğu gibi tespit edilmiştir. Ancak *polbooks* verisinde tüm metriklerin sonuçlarında düşüş görülmüştür.

Çizelge 3.5 Önerilen algoritma topluluk sayısı ve başlangıç topluluklarıyla çalıştırıldığında elde edilen sonuçlar

Veri Kümesi	**Modülarite**	**Performans**	**NMI**	**k**
Karate	0.3715	0.6168	1.0	2
Dolphin	0.3735	0.5219	1.0	2
Polbooks	0.4881	0.6471	0.5480	3

Çizelge 3.6 Önerilen algoritmanın gerçek veri kümesi üzerindeki sonuçları ve başlangıç toplulukları kullanıldığında elde edilen sonuçlar

Veri Kümesi	**Köşe Sayısı**	**Kenar Sayısı**	**k**	**Önerilen Algoritma**		**Önerilen Algoritma Başlangıç Topluluklarıyla**	
				Modülarite	**k**	**Modülarite**	**k**
Karate	34	78	2	**0.4156**	4	0.3774	4
Football	115	613	12	**0.6044**	9	0.5395	8
Dolphin	62	159	2	**0.5203**	5	0.5188	5
Polbooks	105	441	3	**0.5267**	4	0.5141	4
Les Misérables	77	254	-	**0.5555**	7	0.5225	7
NetScience	379	914	-	0.7585	10	**0.7989**	31
Email	1133	5451	-	0.3936	5	**0.5130**	27
Yeast	2375	11693	-	**0.7213**	22	0.7181	168
Web spam	4767	37375	-	0.3880	58	**0.4336**	209
Router	5022	6258	-	**0.8777**	36	0.7850	623
Bio dmela	7393	25569	-	0.3125	26	**0.3277**	394
PGP	10680	24316	-	0.8020	112	**0.8065**	1240

Bu deneylerden sonra algoritma gerçek veriler üzerinde de test edilmiştir. Çizelge 3.6'da verilen sonuçlar önerilen algoritmanın yüksek modülarite değerlerine sahip olduğunu göstermektedir. Başlangıç toplulukları kullanmak *football* hariç bütün verilerde elde edilen topluluk sayısını artırmıştır. Bu artış *yeast* ve *router* verilerinde daha başarılı sonuç almayı sağlamıştır. Ancak iki versiyonun da eşit sayıda topluluk tespit ettiği durumlarda başlangıç toplulukları kullanmanın modülariteyi düşürdüğü görülmüştür. Bununla birlikte *PGP* verisinde 112 veya 1240 topluluk bulmanın çok yakın modülarite değeri verdiği görülmüştür. Bu nedenle her ne kadar hızı artırsa da topluluk sayısının bilinmediği durumlarda başlangıç topluluğu kullanma önerilmemektedir.

3.3 Örtüşen Topluluk Tespiti

Topluluk tespiti yapılırken bazı durumlarda bir köşeyi belli bir topluluğa dahil etmek mümkün olmamaktadır. Örneğin bir köşe iki topluluğa da eşit uzaklıkta olabilir veya iki topluluğa eşit ölçüde bağlı olabilir. Bu problemin çözümlerinden biri de köşeyi iki topluluğa da dahil etmektir. Bu durumda çizge üzerindeki toplulukların kesişmesi yani örtüşmesi söz konusudur. Bir köşenin birden fazla topluluğa ait olduğu ağlarda örtüşen topluluk tespiti yapılmaktadır. Bu nedenle örtüşen topluluklar için ayrı bir algoritma tasar-

lanmıştır. Önerilen algoritma hem gerçek hem de sentetik veri kümeleri ile test edilmiştir. Sentetik ağlar ise parametreleri bölüm 3.1.1'de açıklanan LFR algoritması ile oluşturulmuştur. Kullanılan gerçek veri kümeleri ise bölüm 3.1.2'de açıklanmıştır.

3.3.1 Önerilen algoritma

Örtüşen topluluklar için de $G = (V, E)$ ağırlıksız ve yönsüz basit çizgesinde çalışan, aşağıdan-yukarı yaklaşım uygulayan bir algoritma önerilmiştir. Önerilen algoritma (Algoritma 3) üç aşamadan oluşmaktadır. İlk aşamada her u köşesi n_u ile gösterilen komşularının listesine eklenerek $N_u = \{u\} \cup n_u$ komşuluk listesi elde edilir. Daha sonra bir kenarla birbirine bağlanan tüm köşelerin kosinüs benzerliği hesaplanır. Köşelerin kendilerinin de komşuluk listesine dahil edilmesi, bağlantılı ancak ortak bir komşusu bulunmayan köşelerin benzerlik puanını sıfırdan fazla yaptığından kosinüs benzerliğinin başarısını artırmaktadır. Hesaplanan benzerlikler köşeleri birleştiren kenarlara ağırlık olarak atanır. Böylece başlangıçtaki ağırlıksız G çizgesi ağırlıklı çizgeye dönüştürülür. Ardından her köşe için yalnızca kendisinden oluşan bir topluluk oluşturulur ve başlangıç topluluk listesi C elde edilir. G çizgesinin kenarları azalan ağırlık sırasına göre sıralanarak ilk aşama tamamlanır.

İkinci aşamada $C = \{C_1, C_2, ...C_k\}$ topluluk listesi oluşturulur. İlk aşama sonucunda sıralanan kenar listesindeki ilk kenardan başlayarak tüm kenarlar sırasıyla değerlendirilir. Değerlendirilen kenar ile birbirine bağlanan u ve v köşelerinin her ikisi de sadece kendilerinden oluşan tek elemanlı topluluklarsa bu iki topluluk tek bir toplulukta birleştirilir. Aksi takdirde, bu köşelerin bir arada bulunduğu bir topluluk olup olmadığına bakılır. Önerilen algoritmada u köşesini içeren tüm topluluklar S_u ile gösterilmektedir. Bu nedenle eğer $S_u \cap S_v > 0$ ise iki köşenin de üyesi olduğu ortak bir topluluk var demektir. Böyle bir topluluk varsa, hiçbir işlem yapılmadan bir sonraki kenara geçilir. Eğer böyle bir topluluk yoksa, formül 3.1'de verilen kriterlere göre bu köşelerden hangisini diğerinin topluluğuna eklemenin daha avantajlı olacağına karar verilir. (u, v) kenarı için, belirlenen kritere göre u köşesinin v'nin C_v topluluğuna eklenmesi daha avantajlıysa, C_v topluluğu u köşesi eklenerek genişletilir. Ayrıca C'de $\{u\}$ topluluğu varsa listeden çıkarılır. Bu şekilde her köşe, topluluğundaki köşelerden bağımsız olarak değerlendirilir. Tüm kenarlar değerlendirildi-

Algoritma 3 Örtüşen topluluk tespiti algoritması

Input : $G = (V, E)$ ağırlıksız basit çizgesi
Output: C topluluk listesi

▷ **Aşama 1:** İlklendirme
$C = \phi$
for her $v \in V$ **do**
    $C = C \cup \{v\}$ ▷ her köşeyi ayrı bir topluluk olarak belirle
end for
for her $(u, v) \in E$ **do**
    $w(u, v) = |N_u \cap N_v| / \sqrt{|N_u||N_v|}$ ▷ köşelerin benzerliğini hesapla ve kenarların ağırlıklarını
end for
her $(u, v) \in E$'yi $w(u, v)$'ye göre azalan sırada sırala

▷ **Aşama 2:** Toplulukları bulma
for her $(u, v) \in E$ **do**
    if $C_u = \{\{u\}\}$ ve $C_v = \{\{v\}\}$ **then** ▷ eğer u ve v sadece kendilerinden oluşan bir topluluk ise
        $C \leftarrow C \cup \{u, v\} - \{u\} - \{v\}$ ▷ $\{u, v\}$'yi topluluk listesine ekle ve $\{u\}$ ile $\{v\}$'yi listeden çıkar
    else if $S_u \cap S_v = \phi$ **then** ▷ eğer u ve v'nin beraber bulunduğu bir topluluk yoksa
        $CN(u, v) = \max_{C_v \in C} |C_v \cap n_u|$
        $C_v^* = arg\,max\ CN(u, v)$
        $CN(v, u) = \max_{C_u \in C} |C_u \cap n_v|$
        $C_u^* = arg\,max\ CN(v, u)$
        if $CN(u, v) > CN(v, u)$ veya $(CN(u, v) = CN(v, u)$ ve $deg(u) < deg(v))$ **then**
            $C \leftarrow C - C_v^*$
            $C_v^* \leftarrow C_v^* \cup \{u\}$
            $C \leftarrow C \cup C_v^*$
            if $\{u\} \in C$ **then**
                $C \leftarrow C - \{u\}$
            end if
        else
            $C \leftarrow C - C_u^*$
            $C_u^* \leftarrow C_u^* \cup \{v\}$
            $C \leftarrow C \cup C_u^*$
            if $\{v\} \in C$ **then**
                $C \leftarrow C - \{v\}$
            end if
        end if
    end if
end for

▷ **Aşama 3:** Toplulukları birleştirme ▷ $C = \{C_1, C_2, ...C_k\}$
$k \leftarrow length(C)$
C'yi toplulukların eleman sayılarına göre azalan sırada sırala
$l \leftarrow 1$
for i = 2'den k'ya kadar **do**
    for $j = l$'den 1'e kadar **do**
        if $\left|C_i \cap C_j\right| > |C_i|/2$ or $(|C_i| = 2$ ve $\left|C_i \cap C_j\right| = 1)$ **then**
            $C \leftarrow C - C_j - C_i$
            $C_i \leftarrow C_i \cup C_j$
            $C \leftarrow C \cup C_i$
        end if
    end for
    C listesinde indisleri i'den küçük veya eşit olan toplulukları yeniden numaralandır ve l'ye ata
end for
return C

ğinde, ikinci aşama tamamlanır ve C topluluk listesi elde edilir.

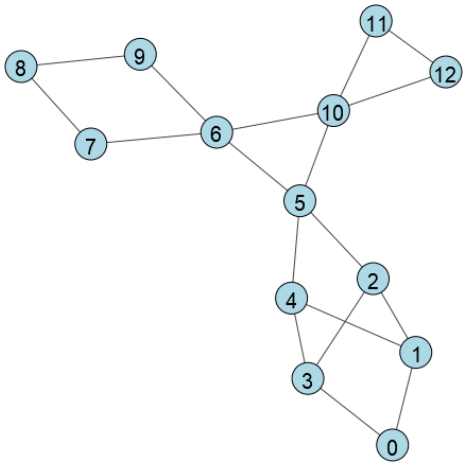
Algoritmanın üçüncü aşaması, C topluluk listesinin iyileştirilmesi aşamasıdır. Bu aşamada C listesindeki topluluklar eleman sayılarına göre azalan sırada sıralanır. Ardından listedeki ikinci topluluktan başlayarak tüm topluluklar değerlendirilinceye kadar sıradaki işlemler yapılır. Sırayla mevcut topluluk ile bir önceki toplulukların kesişimine bakılır. Eğer kesişimdeki eleman sayısı mevcut topluluğun eleman sayısının yarısından fazla ise bu iki topluluk birleştirilir. Özel bir durum olarak, mevcut toplulukta sadece iki eleman varsa ve biri kesişimde ise de birleştirme işlemi uygulanır.

İki köşenin topluluğun birleştirilmesi durumunda hangi birleştirme işleminin daha avantajlı olacağını değerlendirmek için kullanılan kriter formül 3.1'de verilmiştir. Formül 3.1'de u köşesinin komşuluk listesi n_u ve v köşesini içeren bir topluluk C_v ile gösterilmektedir. $CN(u, v)$ değeri u'nun v'nin topluluğundaki komşularının sayısını gösterdiğinden, kriter "Common Neighbors" ifadesinin baş harfleriyle temsil edilmiştir.

$$CN(u, v) = \max_{C_v \in C} |C_v \cap n_u| \tag{3.1}$$

Bu kriterle olası bütün kombinasyonlar değerlendirilmektedir. İlk olarak formül 3.1'e göre $CN(u, v)$'nin en yüksek değeri ve bu değeri sağlayan topluluk C_v^* belirlenir. Sonrasında en yüksek değere sahip $CN(v, u)$ ve bu değeri sağlayan C_u^* belirlenir. *CN(u,v) > CN(v,u)* ise, u'yu C_v^* topluluğuna eklemek daha avantajlıdır. *CN(u,v) < CN(v,u)* ise, v'yi C_u^* topluluğuna eklemek daha avantajlıdır. *CN(u,v) = CN(v,u)* ise u ve v köşelerinin dereceleri $deg(u)$ ve $deg(v)$'ye bakılır ve küçük dereceli köşe diğerinin topluluğuna eklenir. Derecelerin eşitliği durumunda v köşesi C_u^* topluluğuna eklenir.

Önerilen algoritmanın toplulukları nasıl bulduğu şekil 3.6'da verilen örnek çizge üzerinden açıklanmaktadır. İlk olarak, 18 kenarın tümü için kosinüs benzerliği kullanılarak ağırlıklar belirlenir. Daha sonra bu 18 kenar, her bir köşenin ayrı bir topluluk oluşturduğu başlangıç durumundan başlayarak azalan ağırlık sırasına göre değerlendirilir. Her adımdan sonraki sonuçlar şekil 3.7'de verilmiştir.



Şekil 3.6 Örtüşen topluluk tespiti için örnek çizge

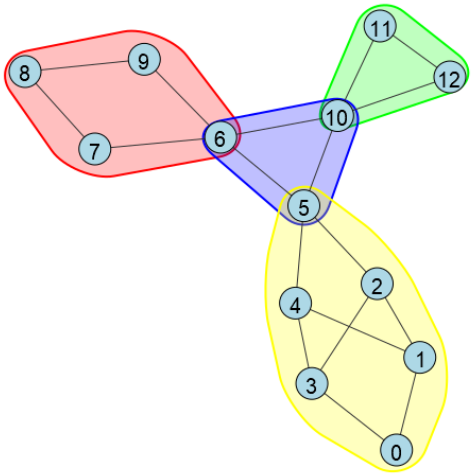
Adım	Kenar (u, v)	Kenar Ağırlığı	CN(u, v)	CN(v, u)	u ve v'nin Derecelerinin Karşılaştırması	Topluluklar
0	-	-	-	-	-	{{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10}, {11}, {12}}
1	(11, 12)	1.00	-	-	-	{{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10}, {11, 12}}
2	(10, 11)	0.78	2	1	-	{{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10, 11, 12}}
3	(10, 12)	0.78	-	-	-	{{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10, 11, 12}}
4	(7, 8)	0.67	-	-	-	{{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7, 8}, {9}, {10, 11, 12}}
5	(8, 9)	0.67	1	1	eşit	{{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7, 8, 9}, {10, 11, 12}}
6	(5, 6)	0.60	-	-	-	{{0}, {1}, {2}, {3}, {4}, {5, 6}, {7, 8, 9}, {10, 11, 12}}
7	(5, 10)	0.60	1	2	-	{{0}, {1}, {2}, {3}, {4}, {5, 6, 10}, {7, 8, 9}, {10, 11, 12}}
8	(6, 10)	0.60	-	-	-	{{0}, {1}, {2}, {3}, {4}, {5, 6, 10}, {7, 8, 9}, {10, 11, 12}}
9	(0, 1)	0.58	-	-	-	{{0, 1}, {2}, {3}, {4}, {5, 6, 10}, {7, 8, 9}, {10, 11, 12}}
10	(0, 3)	0.58	1	1	deg(v) büyük	{{0, 1}, {2}, {0, 3}, {4}, {5, 6, 10}, {7, 8, 9}, {10, 11, 12}}
11	(6, 7)	0.52	2	1	-	{{0, 1}, {2}, {0, 3}, {4}, {5, 6, 10}, {6, 7, 8, 9}, {10, 11, 12}}
12	(6, 9)	0.52	-	-	-	{{0, 1}, {2}, {0, 3}, {4}, {5, 6, 10}, {6, 7, 8, 9}, {10, 11, 12}}
13	(1, 2)	0.50	1	1	eşit	{{0, 1, 2}, {0, 3}, {4}, {5, 6, 10}, {6, 7, 8, 9}, {10, 11, 12}}
14	(1, 4)	0.50	1	1	eşit	{{0, 1, 2, 4}, {0, 3}, {5, 6, 10}, {6, 7, 8, 9}, {10, 11, 12}}
15	(2, 3)	0.50	1	3	-	{{0, 1, 2, 3, 4}, {0, 3}, {5, 6, 10}, {6, 7, 8, 9}, {10, 11, 12}}
16	(3, 4)	0.50	-	-	-	{{0, 1, 2, 3, 4}, {0, 3}, {5, 6, 10}, {6, 7, 8, 9}, {10, 11, 12}}
17	(2, 5)	0.45	1	2	-	{{0, 1, 2, 3, 4, 5}, {0, 3}, {5, 6, 10}, {6, 7, 8, 9}, {10, 11, 12}}
18	(4, 5)	0.45	-	-	-	{{0, 1, 2, 3, 4, 5}, {0, 3}, {5, 6, 10}, {6, 7, 8, 9}, {10, 11, 12}}

Şekil 3.7 Örtüşen topluluk tespiti algoritmasının ikinci aşama adımları

Başlangıçta, en yüksek ağırlığa sahip kenar $(11, 12)$'dir ve 11 ile 12 köşelerinin her ikisi de tek elemanlı topluluklara aittir. Bu nedenle, algoritma 1. adımda bu köşeleri birleştirir. 2. adımda belirlenen kriter ile ikinci en yüksek ağırlığa sahip $(10, 11)$ kenarı değerlendirilir. 10 köşesi, 11 köşesinin topluluğunda daha fazla komşuya sahip olduğundan 10, 11'in topluluğuna eklenir. 3. adımda, $(10, 12)$ kenarı ile bağlanan köşeler aynı toplulukta olduğundan hiçbir değişiklik yapılmaz. 5. adımda, $CN(u, v)$, $CN(v, u)$'ya eşit olduğundan

ve bu köşelerin dereceleri de eşit olduğundan 9, 8'in topluluğuna eklenir. 10. adımda, $CN(u, v)$ ve $CN(v, u)$ da eşittir, ancak 0'ın derecesi 3'ün derecesinden daha yüksektir. Bu nedenle 3, 0'ın topluluğuna eklenir.

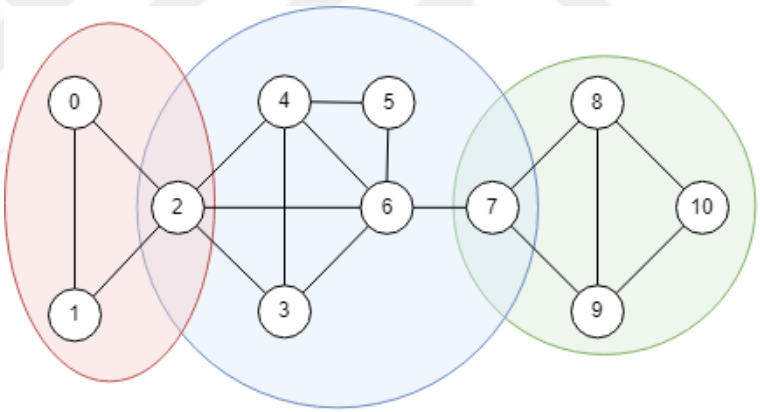
18. adımdan sonra C listesi elde edilir. Bu liste topluluklardaki eleman sayılarına göre azalan sırada sıralandığında $C = \{\{0,1,2,3,4,5\},\{6,7,8,9\},\{5,6,10\},\{10,11,12\},\{0,3\}\}$ listesi elde edilir. İkinci topluluk $\{6,7,8,9\}$'dan başlanır. $\{0,1,3,4,5\}$ topluluğu, $\{6,7,8,9\}$'dan önceki tek topluluktur. Bu toplulukların kesişimi boş küme olduğu için herhangi bir değişiklik yapmadan bir sonraki topluluğa geçilir. Bir sonraki topluluk $\{5,6,10\}$'dur. $\{5,6,10\}$ ile önceki $\{6,7,8,9\}$ topluluğunun kesişimi sadece bir elemana sahiptir. Bu sayı $\{5,6,10\}$ topluluğunun eleman sayısının yarısından daha küçük $(1 < 3/2)$ olduğundan birleştirme işlemi yapılmaz. $\{5,6,10\}$ ile $\{1,2,3,4,5\}$ topluluklarının kesişimi de yalnızca bir elemana sahiptir ve birleştirme koşulu karşılanmamaktadır. Bir sonraki $\{10,11,12\}$ topluluğu, öncekilerden yalnızca $\{5,6,10\}$ topluluğu ile kesişir. Bu kesişim de tek elemanlı olduğu için birleşme şartı sağlanmamaktadır. Son olarak, $\{0,3\}$ topluluğu ile önceki toplulukların kesişimleri incelenir. Yalnızca $\{0,1,2,3,4,5,6\}$ bu toplulukla kesişir ve bu kesişimin eleman sayısı 2'dir. Bu nedenle bu iki topluluk birleştirilir. Sonuç olarak, analiz edilen çizge için algoritma tarafından bulunan topluluklar şekil 3.8'de verilen $C = \{\{0,1,2,3,4,5\},\{6,7,8,9\},\{5,6,10\},\{10,11,12\}\}$ topluluklarıdır.



Şekil 3.8 Örtüşen topluluk tespiti algoritmasının şekil 3.6'da verilen çizge için tespit ettiği topluluklar

Algoritmanın karmaşıklığı hesaplanırken $G = (V, E)$ çizgesindeki köşe sayısı $n = |V|$, kenar sayısı $m = |E|$ ve ikinci aşamadan sonra bulunan topluluk sayısı ise k ile temsil edilmektedir. Burada k, n ile karşılaştırıldığında küçük bir sayıdır. Algoritmanın ilk aşamasında iki döngü vardır. Kosinüs benzerliklerini hesaplayan ilk döngü $O(m)$ zamanda, ikinci döngü $O(n)$ zamanda çalışır. Kenarların sıralanması $O(m\,lg\,m)$ zamanda gerçekleştirilir. Bu nedenle, önerilen algoritmanın ilk aşamasının karmaşıklığı $O(n + m + m\,lg\,m) = O(m\,lg\,m)$'dir. İkinci aşamadaki döngü ise $O(m)$, üçüncü aşama $O(k^2)$ zamanda çalışır. Bu nedenle önerilen algoritmanın toplam karmaşıklığı $O(m\,lg\,m + k^2)$'dir. k'nin $O(n)$'ye eşit olduğu kabul edilebilir. Bu durumda yoğun grafikler için algoritmanın karmaşıklığı $O(m\,lg\,m)$'dir.

Önerilen algoritmanın bölüm 2.6'da verilen temel algoritmalarla karşılaştırılabilmesi için şekil 2.14'te verilen örnek çizge için de topluluklar tespit edilerek sonuçlar şeki l3.9'da verilmiştir.



Şekil 3.9 Örtüşen topluluk tespiti algoritmasının şekil 2.14'te verilen çizge için tespit ettiği topluluklar

3.3.2 Deneysel çalışma

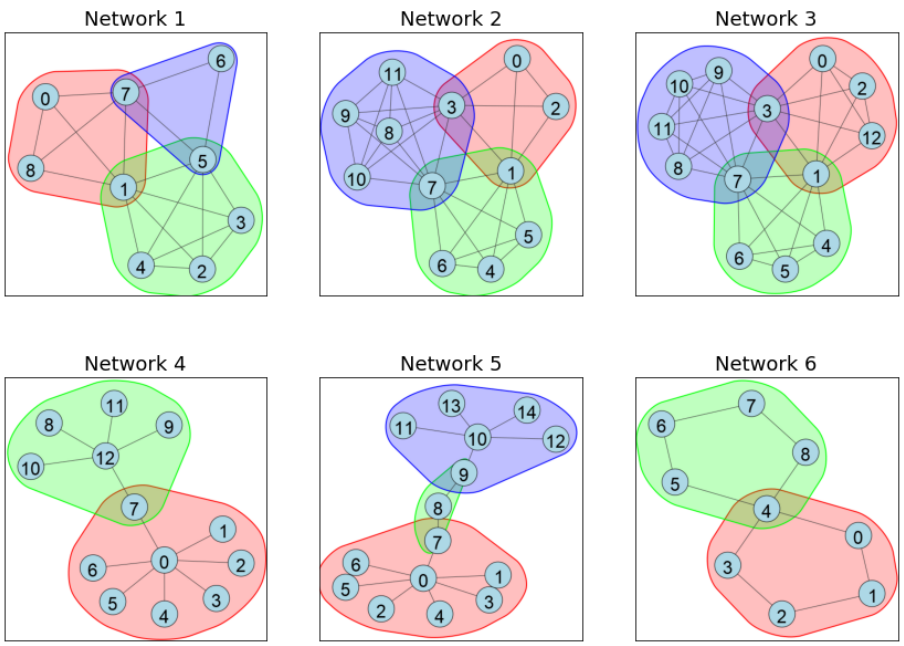
Önerilen algoritma geliştirilirken için öncelikle örnek ağlar ve bu ağlar üzerindeki topluluklar tanımlanmıştır. Sonrasında algoritmanın bu toplulukları başarılı bir şekilde tespit etmesi için çalışılmıştır. Örnek ağlar üç tip topluluk yapısından oluşmaktadır:

- **Klik:** her üyenin birbiri ile bağlantılı olduğu yapıdır.

- **Yıldız:** sadece topluluk liderinin bütün üyelerle bağlantısı olduğu ve üyelerin birbiri ile bağlantısının olmadığı yapıdır.

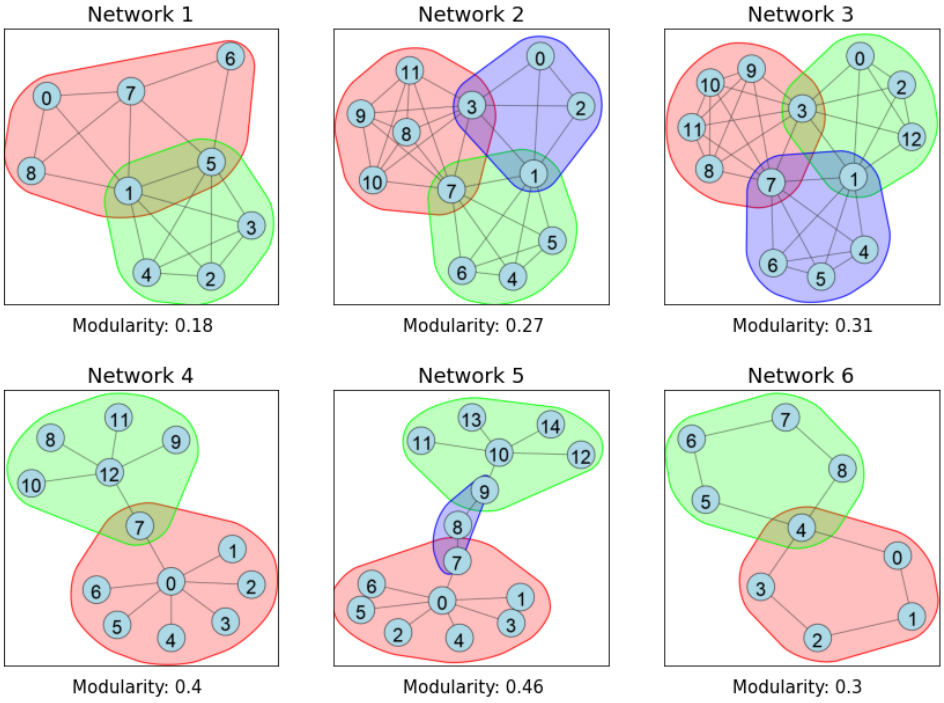
- **Çember:** her üyenin birbirine çembersel bir zincirle bağlı olduğu yapıdır.

Bu yapıları içeren 6 ağ şekil 3.10'da verilmiştir. Bu ağlara bakıldığında doğal topluluklar kolayca görülebilir. 1., 2. ve 3. ağlar farklı boyutlardaki kliklerden oluşmaktadır. 4. ve 5. ağlar ise farklı boyutlarda yıldızlardan oluşmaktadır. 5. ağda ayrıca zincir şeklinde bir topluluk bulunmaktadır. 6. ağ ise eşit boyuttaki iki çemberden oluşmaktadır. Bu topluluklar şekil 3.10 üzerinde işaretlenmiştir.



Şekil 3.10 Örnek ağlar

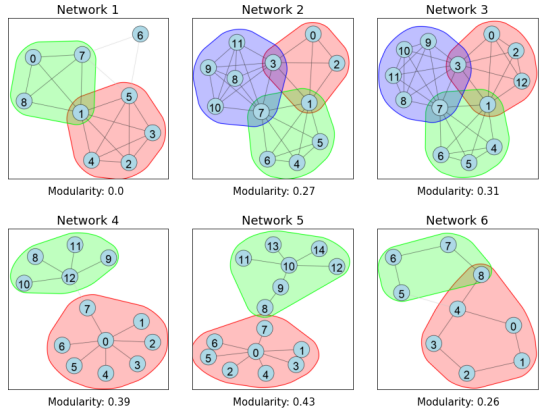
Önerilen algoritmanın başarısını ölçmek için gerçek ve sentetik ağlar üzerinde modülarite, NMI ve F-Skor değerleri kullanılmıştır. Ayrıca Community Discovery Library (CDlib)'de bulunan 34 algoritma incelenerek bu algoritmalardan ön bilgi gerektirmeyen, örnek ağlar üzerinde başarılı sonuçları veren, statik ve işlem zamanı çok fazla olmayan algoritmalarla yapılan karşılaştırmaların sonuçları sunulmuştur. Bu algoritmalar Core Expansion, Ego Networks, LPANNI ve PercoMVC'dir.



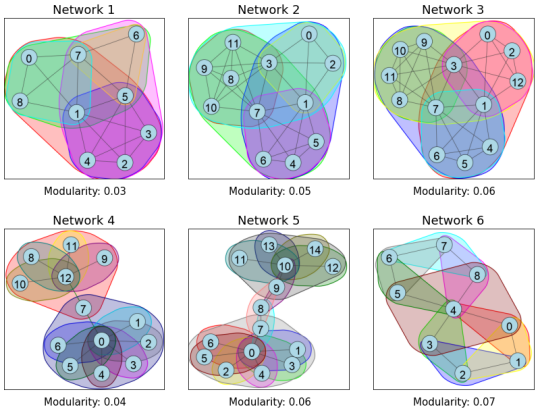
Şekil 3.11 Örtüşen topluluk tespiti algoritmasının örnek ağlar üzerindeki sonuçları

Önerilen algoritmanın bu ağlar üzerindeki sonuçları şekil 3.11'de verilmiştir. 1. ağ hariç diğer bütün ağlarda beklenen topluluklar doğru bir şekilde tespit edilmiştir. 1. ağ için {(0, 1, 7, 8), (1, 2, 3, 4, 5), (5, 6, 7)} topluluklarının tespit edilmesi beklenmektedir, ancak önerilen algoritma ile $\{\{0, 1, 5, 6, 7, 8\}, \{1, 2, 3, 4, 5\}\}$ toplulukları bulunmaktadır. Bu durumun nedeni araştırıldığında önerilen algoritma ile {{0, 8, 7}, {1, 2, 3, 4, 5}, {6}} toplulukları bulunduktan sonra (6, 7) kenarı değerlendirildiği görülmektedir. Bu kenar değerlendirilirken $CN(u, v) = CN(v, u)$ ve 7'nin derecesi 6'dan büyük olduğundan 6, 7'nin topluluğuna eklenir. Bu kural tanımlanmasaydı beklenen topluluklar elde edilirdi. Ancak bu kural 4 ve 5 numaralı ağlardaki gibi yıldız toplulukları bulmak için gereklidir. Ayrıca (1, 5, 7) ve (5, 6, 7) komşu klik olduğundan bu toplulukları birleştirmek yanlış değildir.

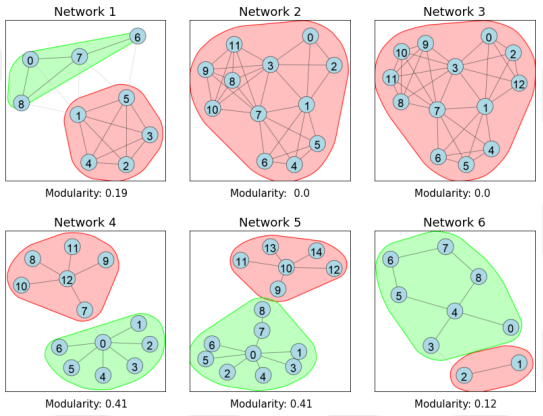
Diğer dört algoritmanın sonuçları ise şekil 3.12'de ve çizelge 3.7'de verilmiştir. Önerilen algoritmaya en yakın sonucu Core Expansion algoritması vermektedir. Ancak bu algoritmanın topluluğunu belirleyemediği köşeler bulunmaktadır ve son üç ağ için örtüşen toplulukları bulmada başarılı değildir. İlk üç ağda başarılı sonuç veren bir diğer algoritma ise PercoMVC'dir ancak bu algoritma yıldız ve çember toplulukları bulamamaktadır. Ego



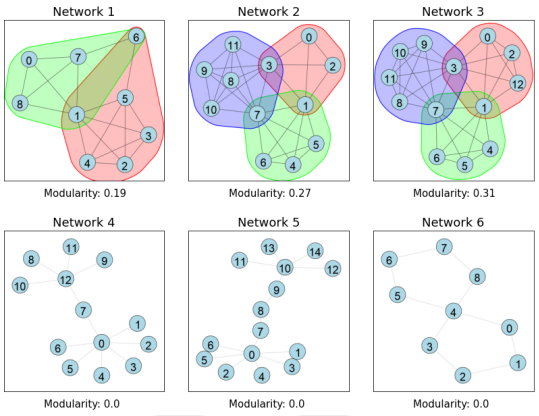
(a) Core Expansion



(b) Ego Networks



(c) LPANNI



(d) PercoMVC

Şekil 3.12 Mevcut algoritmaların örnek ağlar üzerindeki sonuçları

Network ise olası bütün toplulukları bulmaktadır ancak istenilen toplulukları bulmak için filtreleme yapmak gerekmektedir. LPANNI 2 ve 3 numaralı ağlarda toplulukları tespit edememiş ve diğer ağlarda ayrık topluluklar tespit etmiştir.

Çizelge 3.7 Örtüşen topluluk tespiti algoritmalarının örnek ağlar üzerindeki sayısal sonuçları

	Core Expansion		Ego Networks		LPANNI		PercoMVC		Önerilen	
	Q	*Kapsama*	*Q*	*Kapsama*	*Q*	*Kapsama*	*Q*	*Kapsama*	*Q*	*Kapsama*
Network 1	0	0.84	0.03	1	0.19	0.74	0.19	0.95	0.18	1
Network 2	0.27	1	0.05	1	0	0	0.27	1	0.27	1
Network 3	0.31	1	0.06	1	0	0	0.31	1	0.31	1
Network 4	0.39	0.92	0.04	1	0.41	0.92	0	0	0.40	1
Network 5	0.43	0.93	0.06	1	0.41	0.93	0	0	0.46	1
Network 6	0.26	0.90	0.07	1	0.12	0.80	0	0	0.30	1

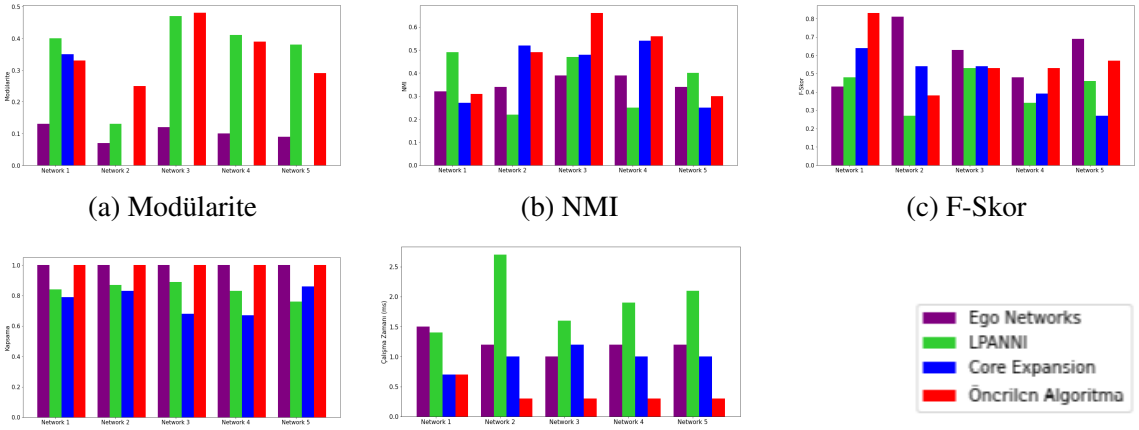
Aynı zamanda bu algoritmaların modülarite sonuçları da karşılaştırılmıştır. 0,3 ve 0,7 ara-

sındaki modülarite başarılı sayılmaktadır ve önerilen algoritma 1. ve 2. ağ hariç bütün ağlarda bu aralıkta sonuç vermektedir. 2 numaralı ağ incelendiğinde istenilen topluluklar bulunmasına rağmen elde edilen modülaritenin 0.27 olduğu görülmektedir. Önerilen algoritma ile elde edilen modülarite değerleri 4 ağda da en yüksek değerlerdir. 1 ve 4 numaralı ağlarda ise en yüksek modülarite değerine çok yakın sonuçlar elde edilmiştir. Ancak önerilen algoritma ile istenilen yapılar tespit edilmesine rağmen elde edilen en yüksek modülarite değeri 0.42'dir. Bununla birlikte Ego Network olası bütün toplulukları bulmasına rağmen modülarite sonuçlarının çok düşük olduğu görülmektedir. Ayrıca Core Expansion her düğümü bir topluluğa atayamadığı için modülarite hesaplanamamıştır. Bu nedenle farklı bir karşılaştırma metriğine ihtiyaç duyulmuş ve ayrık topluluklar için önerilen kapsama metriği örtüşen topluluklar için uyarlanmıştır.

Uyarlanmış kapsama metriği ile elde edilen sonuçlar çizelge 3.7'de karşılaştırılmıştır. PercoMVC 4., 5. ve 6. ağlar için bütün ağı tek bir topluluk olarak gösterdiğinden sonuç 0'dır. Benzer şekilde LPANNI de 2. ve 3. ağlarda 0 değerini vermiştir. Bunların dışındaki sonuçlar incelendiğinde Core Expansion'un 6. ağda, LPANNI'nin ise 1. ve 6. ağlarda bulduğu topluluklar hariç bulunan bütün toplulukların başarılı olduğu söylenebilir. Bu üç topluluk için uyarlanmış kapsama metriği en düşük sonuçları vermektedir. Bu da uyarlanmış kapsama metriğinin karşılaştırma için uygun olduğunu göstermektedir. Önerilen algoritmanın uyarlanmış kapsama metriğine göre elde ettiği sonuçlar ise bütün ağlarda 1'dir. Bu durumun nedeni metriğin köşelerin ne derece sosyal olduğunu yani bir köşenin komşularıyla beraber aynı toplulukta ne sıklıkta bulunduğunu ölçmesidir. Önerilen algoritma da sosyalliği artırma üzerine kurulduğundan bu metriğe göre en iyi sonuçları sağlamaktadır.

Algoritmanın sentetik ağlardaki başarısı ise öncelikle küçük ağlar üzerinde test edilmiştir. Köşe sayının 15, karıştırma parametresinin 0.1, örtüşen köşe sayısının 3 ve örtüşen köşelerin 2 topluluğa ait olduğu sentetik ağlar üretilmiştir. Şekil 3.13'de ve EK 1'de verilen sonuçlar, önerilen algoritmanın yüksek modülarite, NMI, F-Skor ve kapsama değerleri verdiğini göstermektedir. Ayrıca işlem zamanı da düşüktür.

Bu deneyden sonra gerçek veri kümeleriyle test yapılmıştır. En popüler veri kümelerinden *karate* üzerindeki sonuçlar şekil 3.14 (a)'da verilmiştir. Bu veri kümesinde önerilen algo-

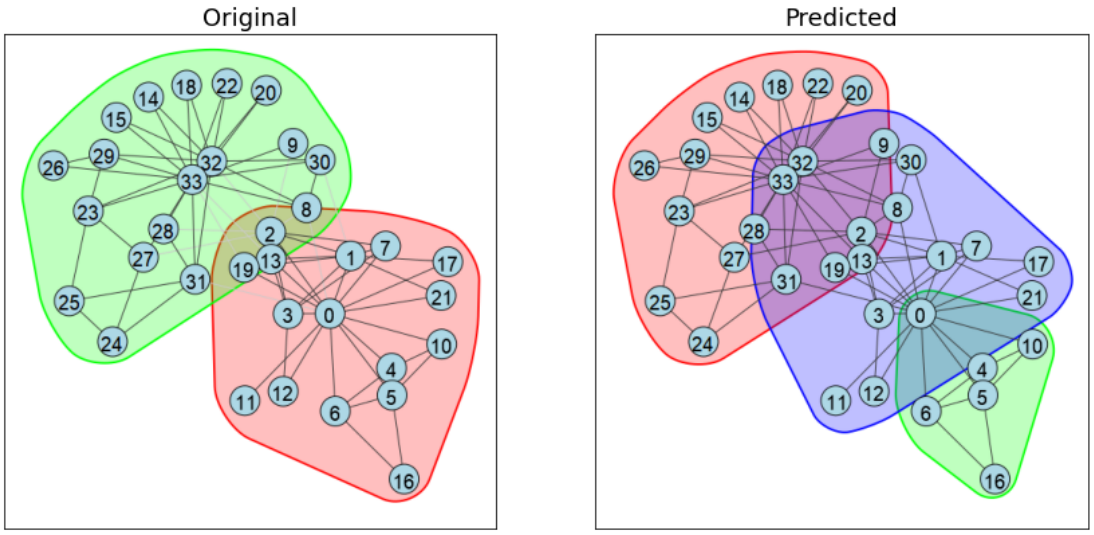


(a) Modülarite (b) NMI (c) F-Skor

(d) Kapsama (e) Çalışma Zamanı

Şekil 3.13 Algoritmaların sentetik veri kümesi üzerindeki sonuçların grafikleri (15 köşe)

ritma ile 3 topluluk bulunmuş ve 0,29 NMI, 0,52 F-Skor ile 0,25 modülarite değerleri elde edilmiştir. Önerilen ayrık topluluk tespiti algoritmasında olduğu gibi bu algoritmaya da topluluk sayısı bilgisi verilseydi bulunacak topluluklar şekil 3.14 (b)'de gösterilmiştir. Bu durumda yine ayrık topluluk tespitindeki gibi aynı köşe farklı topluluğa atanmış olacaktır.



(a) Gerçek ve örtüşen topluluk tespiti algoritmasının bulduğu topluluklar

Köşe	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
Tahmin edilen	1,2	1	1,3	1	2	2	2	1	1,3	1	2	1	1	1	3	3	2	1	3	1	3	1	3	3	3	3	3	1,3	1,3	3	1,3	1,3	3	1,3
k toplulukla	1	1	1,2	1	1	1	1	1	1,2	1	1	1	1	1	2	2	1	1	2	1	2	1	2	2	2	2	2	1,2	1,2	2	1,2	1,2	2	1,2
Gerçek	1	1	1	1	1	1	1	1	2	2	1	1	1	1	2	2	1	1	2	1	2	1	2	2	2	2	2	2	2	2	2	2	2	2

(b) köşelerin topluluk etiketleri

Şekil 3.14 Örtüşen topluluk tespiti algoritmasıyla *karate* veri kümesinde elde edilen sonuçlar

Benzer şekilde *dolphins*, *lesmis* ve *politics* veri kümelerinde elde dilen sonuçlar çizelge 3.8'de ve EK 2'de verilmiştir. Tahmin edilen toplulukların gerçek topluluklara yakın ve daha detaylı olduğu görülmektedir.

Çizelge 3.8 Örtüşen topluluk tespiti algoritmasının gerçek veri kümeleri üzerindeki sonuçları

	Gerçek			Önerilen			
	k	Q	Kapsama	k	Q	NMI	F-Skor
karate	2	0.37	0.87	3	0.25	0.29	0.52
lesmis	7	0.56	0.76	7	0.39	0.46	0.45
polbooks	3	0.41	0.84	4	0.43	**0.35**	0.22
dolphins	2	0.37	0.96	5	0.34	0.30	**0.34**

Diğer algoritmaların sonuçları ise çizelge 3.9'da verilmiştir. Core Expansion ve PercoMVC algoritmaları her düğümü bir topluluğa atayamamaktadır. Ego Network çok sayıda topluluk bulmaktadır. LPANNI en yüksek modülarite ve NMI değerlerine sahip olmasına rağmen F-Skor değerlerinde aynı başarıyı gösterememektedir. Önerilen algoritma ise *polbooks* verisinde en yüksek NMI ve *dolphins* verisinde en yüksek F-Skor değerini vermektedir. Ayrıca bulunan topluluk sayıları incelendiğinde gerçek topluluklara en yakın sonuçlar önerilen algoritma ile elde edilebilmektedir.

Çizelge 3.9 Mevcut algoritmaların gerçek veri kümeleri üzerindeki sonuçları

Core Expansion

	k	Q	NMI	F-Skor	Kapsama
karate	2	0	0.36	0.67	0.54
lesmis	13	0	0.40	**0.67**	0.89
polbooks	16	0	0.13	0.08	0.76
dolphins	12	0	0.13	0.15	0.65

Ego Network

	k	Q	NMI	F-Skor	Kapsama
karate	34	0.04	0.14	0.05	1
lesmis	63	0.07	0.25	0.16	1
polbooks	105	0.04	0.10	0.02	1
dolphins	62	0.05	0.10	0.02	1

LPANNI

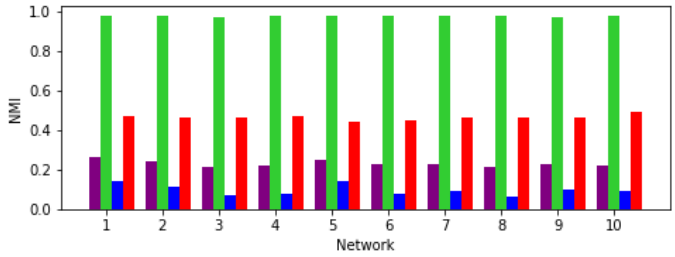
	k	Q	NMI	F-Skor	Kapsama
karate	3	**0.40**	**0.63**	0.60	0.82
lesmis	10	**0.52**	**0.57**	0.61	0.87
polbooks	6	**0.50**	0.34	0.14	0.91
dolphins	6	**0.51**	**0.33**	0.26	0.81

PercoMVC

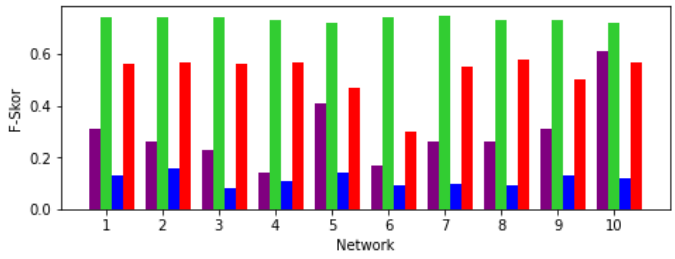
	k	Q	NMI	F-Skor	Kapsama
karate	3	0	0.38	**0.69**	0.87
lesmis	4	0	0.40	0.32	0.93
polbooks	6	0	0.29	**0.34**	0.84
dolphins	4	0	0.18	0.27	0.57

Son olarak önerilen algoritma büyük sentetik ağlarda test edilmiştir. Köşe sayının 1000, ortalama derecenin 30, maksimum derecenin 100, karıştırma parametresinin 0.1, örtüşen köşe sayısının 100 ve örtüşen köşelerin 2 topluluğa ait olduğu 10 sentetik ağ üretilmiştir. Elde dilen NMI ve F-Skor değerleri ile çalışma zamanları şekil 3.15'te verilmiştir. PercoMVC klik süzme yöntemi kullandığından her ağ için sonuç üretememiştir, bu nedenle

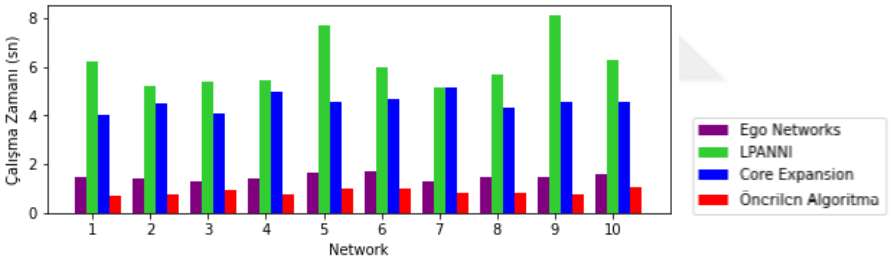
kıyaslamaya alınamamıştır. LPANNI en yüksek NMI ve F-Skor değerini vermektedir ancak aynı zamanda en uzun işlem süresine sahiptir. Önerilen algoritma çalışma zamanına göre en iyi sonucu vermektedir.



(a) NMI



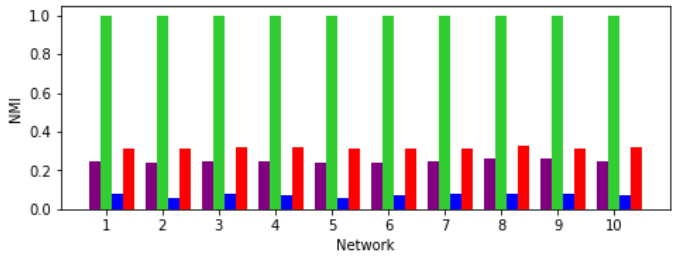
(b) F-Skor



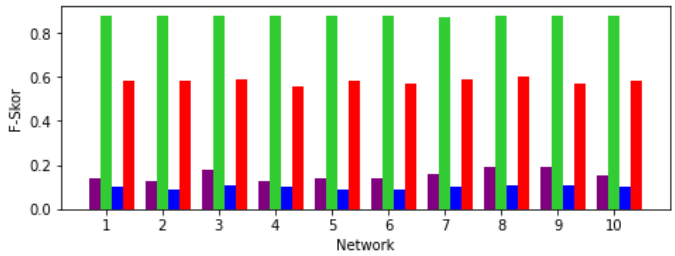
(c) Çalışma Zamanı (sn)

Şekil 3.15 Örtüşen topluluk tespiti algoritmasının sentetik veri kümeleri üzerindeki sonuçları (1000 köşe)

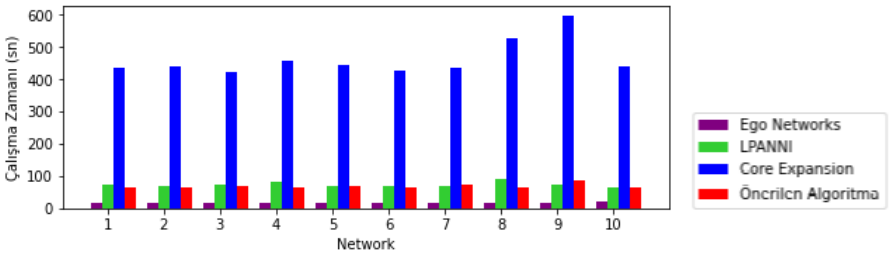
Ayrıca topluluktaki köşe sayısı 10 kat artırıldığında elde edilen sonuçlar şekil 3.16'da verilmiştir. Bu veri kümesinde en uzun işlem zamanına sahip algoritma Ego Network algoritmasıdır. Önerilen algoritma ise en kısa sürede çalışmakta ve başarılı sonuçlar vermektedir.



(a) NMI



(b) F-Skor



(c) Çalışma Zamanı (sn)

Şekil 3.16 Örtüşen topluluk tespiti algoritmasının sentetik veri kümeleri üzerindeki sonuçları (10000 köşe)

4. TARTIŞMA ve SONUÇ

Literatürde çeşitli topluluk tanımları ve bu tanımlara uygun topluluk tespiti yapan algoritmalar mevcuttur. Ancak matematiksel tanımlar çok katıdır ve bütün koşullar sağlansa bile tek bir koşulun sağlanmaması yapının topluluk tanımı dışında kalmasına neden olmaktadır. Bu nedenle sezgisel yöntemler önerilmiştir. Sezgisel yöntemlerde aşağıdan yukarı ve yukarıdan aşağı olmak üzere iki farklı yaklaşım izlenmiştir. Yukarıdan aşağı yaklaşımlar çizgeyi parçalarına ayırarak toplulukları tespit eder. Ancak bu yaklaşımlarda bölme işleminin ne zamana kadar sürdürüleceği problemi bulunmaktadır. Bu nedenle bu algoritmalar daha çok topluluk sayısının bilindiği durumlarda kullanılmaktadır. Aşağıdan yukarı yaklaşımlarda ise köşeler birleştirilerek topluluklar oluşturulmaktadır. Bu algoritmalarda hangi köşelerin birleştirileceğini belirlemek ve birleştirme işleminin ne zamana kadar sürdürüleceği konusunda bazı metriklere ihtiyaç duyulmaktadır.

Bu çalışmada ağırlıksız ve yönsüz çizgeler üzerinde topluluk tespiti yapan iki yeni verimli algoritma önerilmiştir. İlk olarak ayrık topluluk tespiti problemi üzerinde durulmuş ve benzerlik ölçütleri incelenmiştir. Yapılan deneyler sonucunda köşenin kendisinin de komşuluk listesine dahil edilmesi durumunda kosinüs benzerliğinin başarılı sonuç verdiği gözlemlenmiştir. Ancak farklı veri kümelerinde farklı benzerlik metrikleri daha başarılı olabilir. Bu durumda algoritmalar benzerlik metriğini dışarıdan parametre olarak alabilecek şekilde tasarlanmıştır. Bunun için Algoritma 1'de verilen ayrık topluluk tespiti algoritmasında 2. satırın, Algoritma 3'te verilen örtüşen topluluk tespiti algoritmasında 4. satırın değiştirilmesi yeterlidir. Birleştirme işleminin ne zaman sonlandırılacağını belirlemek için yine literatürde mevcut olan modülarite formülünden faydalanılmıştır. Önerilen algoritma gerçek veri kümeleri üzerinde test edilmiş ve *karate* veri kümesinde en başarılı algoritmalardan biri olan Girvan-Newman algoritmasından daha yüksek performans ve modülarite elde edilmiştir. Girvan-Newman algoritması ön bilgi olarak topluluk sayısını kullanmaktadır ve *karate* veri kümesinde iki topluluk bulunmamaktadır. Ancak önerilen algoritma iki alt topluluk daha tespit etmiştir bu nedenle modülarite ve performans kriterinden daha yüksek sonuçlar alınmıştır. Ancak gerçek topluluklarla kıyaslandığında NMI değerinin düşük olduğu görülmüştür. Önerilen algoritmaya topluluk sayısı bilgisi veril-

diğinde *karate* verisi için Girvan-Newman algoritmasında olduğu gibi sadece bir köşeyi hatalı kümelemesine rağmen Girvan-Newman'dan daha yüksek modülarite, performans ve NMI değerleri elde edilmiştir.

Ayrık topluluk tespiti deneyinde kosinüs benzerliği ve modülarite hesaplamanın işlem zamanını çok artırdığı gözlemlenmiştir. Bu nedenle algoritmayı hızlandırmak için başlangıç toplulukları belirlenmiştir. Sırasıyla derecesi en yüksek köşeler komşularıyla beraber topluluklara ayrılarak başlangıç toplulukları oluşturulmuştur. Birden fazla topluluğa dahil edilen köşeler topluluklarından çıkarılarak toplulukları Algoritma 1 ile belirlenmiştir. Bu algoritmada toplulukları birleştirme işlemine verilen topluluk sayısına ulaşıncaya kadar devam edilmiştir. Bu deney sonucunda *karate* ve *dolphins* veri kümelerinde orijinal toplulukların hatasız bir şekilde tespit edildiği görülmüştür. Önerilen algoritma farklı gerçek veri kümeleri üzerinde test edildiğinde topluluk sayısının belli olmadığı durumlarda başlangıç topluluklarını kullanmanın tespit edilen topluluk sayısını artırdığı gözlemlenmiştir. Başlangıç toplulukları belirlenirken sırasıyla en yüksek dereceye sahip köşelerin komşularıyla beraber bir topluluk oluşturması, başlangıç toplulukları sayısıyla köşelerin dereceleri arasında ters orantıya neden olmaktadır. Bu nedenle yüksek dereceli köşelerin az sayıda bulunduğu büyük ağlarda çok sayıda topluluk tespit edilmektedir. Elde edilen sonuçlar incelendiğinde *PGP* veri kümesinde yaklaşık 10 kat fazla topluluk bulunmasına rağmen çok yakın modülarite değerlerinin elde edildiği görülmüştür. Bu nedenle topluluk sayısının bilinmediği durumlarda başlangıç topluluklarının kullanılması önerilmemektedir. Ayrıca bu sonuçlar modülaritenin çok sağlıklı bir metrik olmadığını göstermektedir.

Yapılan deneylerden sonra topluluk tanımı üzerinde düşünülmüş ve çizge üzerinde tespit edilmesi beklenen topluluklar klik, yıldız ve çember olarak üç grupta modellenerek test ağları oluşturulmuştur. Daha sonra genel topluluk tanımına bağlı kalınarak bu üç topluluğu da başarılı bir şekilde tespit edebilecek örtüşen topluluk tespiti algoritması önerilmiştir. Bu sayede farklı çözümleri ve gizli yapıları görmemizi sağlayan sonuçlar elde edilmiştir. Önerilen algoritma gerçek ve sentetik veriler üzerinde modülarite, NMI ve F-Skor değerleri ile test edilmiştir. Elde edilen sonuçlar önerilen algoritmanın düşük çalışma zamanında başarılı sonuçlar verdiğini göstermektedir.

Örtüşen topluluk tespitinde LPANNI, LFR ile üretilen ağlarda başarılı olmasına rağmen tanımlanan örnek ağlarda başarısız olmuştur. PercoMVC ise tanımlanan ilk üç ağda başarılı olmasına rağmen LFR ile üretilen ağlarda başarısız olmuştur. Core Expansion sentetik ağlarda en kötü sonucu vermiştir ancak tanımlanan örnek ağlarda başarılı sonuçlar vermiştir. Ego Network ise olası bütün toplulukları bulmasına rağmen bu topluluklar içinden ana toplulukları ayırt etmek zordur ve çok fazla işlem zamanı gerektirmektedir. Yapılan çalışmalar bütün verilerde iyi sonuç veren bir algoritma olmadığını ve bu nedenle önerilen her algoritmanın değerli olduğunu göstermiştir.

Bu tezde önerilen ayrık ve örtüşen topluluk tespiti algoritmaları ile başarılı sonuçlar elde edilmiştir. Benzerlik metriği, topluluk sayısı, başlangıç toplulukları kullanıp kullanmama algoritmaya parametre olarak sağlanabilmektedir. Ayrıca algoritmanın yönlü çizgelerde kullanılması ve ağırlıklı çizgelerde benzerlik metriği kullanmadan doğrudan ağırlıklarla çalıştırılması mümkündür. Ancak yine de bütün algoritmalarda olduğu gibi önerilen algoritmalar da geliştirilmeye açıktır. Örneğin, önerilen iki algoritma birleştirilerek hem ayrık hem de örtüşen toplulukların aynı algoritma ile bulunması sağlanabilir. Ayrıca önerilen algoritmalar paralelleştirilerek işlem zamanının daha da kısaltılması mümkündür.

KAYNAKLAR

Aghaalizadeh, S., Afshord, S. T., Bouyer, A. ve Anari, B. 2021. A three-stage algorithm for local community detection based on the high node importance ranking in social networks. Physica A: Statistical Mechanics and its Applications, 563, 125420.

Ahn, Y. Y., Bagrow, J. P. ve Lehmann, S. 2010. Link communities reveal multiscale complexity in networks. Nature, 466 (7307), 761–764.

Alba, R. D. 1973. A graph-theoretic definition of a sociometric clique. Journal of Mathematical Sociology, 3 (1), 113–126.

Arasteh, M. ve Alizadeh, S. 2019a. A fast divisive community detection algorithm based on edge degree betweenness centrality. Applied Intelligence, 49 (2), 689–702.

Arasteh, M. ve Alizadeh, S. 2019b. Community detection in complex networks using a new agglomerative approach. Turkish Journal of Electrical Engineering and Computer Sciences, 27 (5), 3356–3367.

Bai, L., Liang, J., Du, H. ve Guo, Y. 2018. A novel community detection algorithm based on simplification of complex networks. Knowledge-Based Systems, 143, 58–64.

Batagelj, V. ve Zaversnik, M. 2011. Fast algorithms for determining (generalized) core groups in social networks. Advances in Data Analysis and Classification, 5, 129–145.

Blondel, V. D., Guillaume, J.-l., Lambiotte, R. ve Lefebvre, E. 2008. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 10, 10008.

Bouyer, A. ve Roghani, H. 2020. LSMD: A fast and robust local community detection starting from low degree nodes in social networks. Future Generation Computer Systems, 113, 41–57.

CDlib - Community Discovery Library. Web Sitesi: https://cdlib.readthedocs.io/en/latest/, Erişim Tarihi: 31.07.2021

Chen, D., Shang, M., Lv, Z. ve Fu, Y. 2010. Detecting overlapping communities of weighted networks via a local algorithm. Physica A: Statistical Mechanics and its Applications, 389 (19), 4177–4187.

Chen, Z., Jia, M., Yang, B. ve Li, X. 2015. Detecting overlapping community in complex network based on node similarity. Computer Science and Information Systems, 12 (2), 843–855.

Choumane, A., Awada, A. ve Harkous, A. 2020. Core expansion: A new community detection algorithm based on neighborhood overlap. Social Network Analysis and Mining, 10 (30).

Clauset, A., Newman, M. E. J. ve Moore, C. 2004. Finding community structure in very large networks. Physical Review E, 70 (6), 1–6.

Cordasco, G. ve Gargano, L. 2010. Community detection via semi-synchronous label propagation algorithms, 2010 IEEE International Workshop on: Business Applications of Social Network Analysis (BASNA), 15 Aralık, Bangalore, India.

Coscia, M., Rossetti, G., Giannotti, F. ve Pedreschi, D. 2012. DEMON: A local-first discovery method for overlapping communities, Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 12 Ağustos, Beijing, China.

Everett, M. G. ve Borgatti, S. P. 1998. Analyzing clique overlap. Connections, 21 (1), 49–61.

Fang-ju, A. I. 2017. Research on a large-scale community detection algorithm based on non-weighted graph. Cluster Computing, 22 (2), 2555–2562.

Farkas, I., Ábel, D., Palla, G. ve Vicsek, T. 2007. Weighted network modules. New Journal of Physics, 9 (6), 180.

Fortunato, S. 2010. Community detection in graphs. Physics Reports, 486 (3), 75–174.

Freeman, L. C. 1977. A set of measures of centrality based on betweenness. Sociometry, 40 (1), 35–41.

Gao, Y., Yu, X. ve Zhang, H. 2021. Overlapping community detection by constrained personalized PageRank. Expert Systems with Applications, 173, 114682.

Girvan, M. ve Newman, M. E. J. 2002. Community structure in social and biological networks. National Academy of Sciences, 99 (12), 7821–7826.

Gopalan, P. K. ve Blei, D. M. 2013. Efficient discovery of overlapping communities in massive networks. Proceedings of the National Academy of Sciences, 110 (36), 14534–14539.

Gregory, S. 2007. An algorithm to find overlapping community structure in networks, Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, 17 Eylül, Warsaw, Poland.

Gregory, S. 2008. A fast algorithm to find overlapping communities in networks, Machine Learning and Knowledge Discovery in Databases (ECML PKDD), 15 Eylül, Antwerp, Belgium.

Gregory, S. 2010. Finding overlapping communities in networks by label propagation. New Journal of Physics, 12, 103018.

Hollocou, A., Bonald, T. ve Lelarge, M. 2016. Improving PageRank for Local Community Detection. ArXiv,

Hosseini, R. ve Rezvanian, A. 2020. AntLP : ant-based label propagation algorithm for community detection in social networks. CAAI Transactions on Intelligence Technology, 5 (1), 34–41.

Hu, X., He, W., Li, L., Lin, Y., Li, H. ve Pan, J. 2017. An efficient and fast algorithm for community detection based on node role analysis. International Journal of Machine Learning and Cybernetics, 10 (4), 641–654.

Huang, L., Wang, G., Wang, Y., Pang, W. ve Ma, Q. 2016. A link density clustering algorithm based on automatically selecting density peaks for overlapping community detection. International Journal of Modern Physics B, 30 (24).

Jin, D., Gabrys, B. ve Dang, J. 2015. Combined node and link partitions method for finding overlapping communities in complex networks. Scientific Reports, 5, 8600.

Joghan, H. S., Bagheri, A. ve Azad, M. 2019. Weighted label propagation based on local edge betweenness. The Journal of Supercomputing, 75, 8094–8114.

Jokar, E. ve Mosleh, M. 2018. Community detection in social networks based on improved label propagation algorithm and balanced link density. Physics Letters A, 383 (8), 718–727.

Kasoro, N., Kasereka, S., Mayogha, E., Vinh, H. T. ve Kinganga, J. 2019. PercoMCV: A hybrid approach of community detection in social networks, The 10th International Conference on Ambient Systems, Networks and Technologies (ANT), 29 Nisan, Leuven, Belgium.

Kim, J., Lim, S., Lee, J.-G. ve Lee, B. S. 2018. LinkBlackHole* : Robust overlapping community detection using link embedding. IEEE Transactions on Knowledge and Data Engineering, 31 (11), 2138–2150.

Kleinberg, J. 2002. An impossibility theorem for clustering, Proceedings of the 15th International Conference on Neural Information Processing Systems (NIPS'02), 1 Ocak, Cambridge, MA, USA.

Kouni, I. B. E., Karoui, W. ve Ben Romdhane, L. 2020. Node Importance based Label Propagation Algorithm for overlapping community detection in networks. Expert Systems with Applications, 162, 113020.

Lancichinetti, A., Fortunato, S. ve Kertesz, J. 2009. Detecting the overlapping and hierarchical community structure in complex networks. New Journal of Physics, 11, 033015.

Lee, C., Reid, F., McDaid, A. ve Hurley, N. 2010. Detecting highly overlapping community structure by greedy clique expansion. ArXiv,

Lierde, H. V., Member, G. S. ve Chow, T. W. S. 2019. Scalable spectral clustering for overlapping community detection in large-scale networks. IEEE Transactions on Knowledge and Data Engineering, 32 (4), 754–767.

Lim, S., Ryu, S., Kwon, S., Jung, K. ve Lee, J.-G. 2014. LinkSCAN: Overlapping community detection using the link-space transformation, IEEE 30th International Conference on Data Engineering (ICDE), 4 Nisan, Chicago, IL, USA.

Long, H. 2018. Overlapping community detection with least replicas in complex networks. Information Sciences, 453, 216–226.

Lu, M., Zhang, Z., Qu, Z. ve Kang, Y. 2019. LPANNI : Overlapping community detection using label propagation in large-scale complex networks. IEEE Transactions on Knowledge and Data Engineering, 31 (9), 1736–1749.

Luce, R. D. 1950. Connectivity and generalized cliques in sociometric group structure. Psychometrica, 15 (2), 169–190.

Ma, T., Wang, Y., Tang, M., Cao, J., Tian, Y., Al-Dhelaan, A. ve Al-Rodhaan, M. 2016. LED: A fast overlapping communities detection algorithm based on structural clustering. Neurocomputing, 207, 488–500.

Mancoridis, S., Mitchell, B. S., C., R., Chen, Y. ve Gansner, E. R. 1998. Using automatic clustering to produce high-level system organizations of source code, 6th Inter-

national Workshop on Program Comprehension (IWPC'98), 24 Haziran, Ischia, Italy.

McDaid, A., Greene, D. ve Hurley, N. 2011. Normalized mutual information to evaluate overlapping community finding algorithms. ArXiv,

Meghanathan, N. 2016. A greedy algorithm for neighborhood overlap-based community detection. Algorithms, 9 (8).

Mohammadi, M., Fazlali, M. ve Hosseinzadeh, M. 2020. Accelerating Louvain community detection algorithm on graphic processing unit. The Journal of Supercomputing, 77 (6), 6056–6077.

Mokken, R. J. 1979. Cliques, clubs and clans. Quality and Quantity, 13, 161–173.

Molloy, M. ve Reed, B. 1995. A critical point for random graphs with a given degree sequence. Random Structures and Algorithms, 6 (2), 161–180.

Nepusz, T., Petróczi, A., Négyessy, L. ve Bazsó, F. 2008. Fuzzy communities and the concept of bridgeness in complex networks. Physical Review E, 77 (1), 1–12.

Newman, M. E. J. 2006. Modularity and community structure in networks. Proceedings Of The National Academy Of Sciences Of The United States Of America, 103 (23), 8577–8582.

Newman, M. E. J. 2010. Networks an Introduction. Oxford University Press, 784, Oxford.

Newman, M. E. J. ve Girvan, M. 2004. Finding and evaluating community structure in networks. Physical Review E, 69 (2), 1–15.

Okuda, M., Satoh, S., Sato, Y. ve Kidawara, Y. 2021. Community detection using restrained random-walk similarity. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43 (1), 89–100.

Palla, G., Derényi, I., Farkas, I. ve Vicsek, T. 2005. Uncovering the overlapping community structure of complex networks in nature and society. Nature, 435, 814–818.

Pares, F., Garcia-Gasulla, D., Vilalta, A., Moreno, J., Ayguade, E., Labarta, J., Cortes, U. ve Suzumura, T. 2017. Fluid communities: A competitive, scalable and diverse community detection algorithm. Complex Networks and Their Applications, (689), 229–240.

Pattabiraman, B., Patwary, M. A., Gebremedhin, A. H., Liao, W.-K. ve Choudhary, A. 2015. Fast algorithms for the maximum clique problem on massive graphs with applications to overlapping community detection. Internet Mathematics, 11 (4-5), 421–448.

Pons, P. ve Latapy, M. 2005. Computing communities in large networks using random walks, Computer and Information Sciences (ISCIS), 26 Ekim, Istanbul, Turkey.

Psorakis, I., Roberts, S., Ebden, M. ve Sheldon, B. 2011. Overlapping community detection using Bayesian non-negative matrix factorization. Physical Review E, 83, 066114.

Raghavan, U. N., Albert, R. ve Kumara, S. 2007. Near linear time algorithm to detect community structures in large-scale networks. Physical Review E, 76 (3), 036106.

Reichardt, J. ve Bornholdt, S. 2006. Statistical mechanics of community detection. Physical Review E, 74 (1), 016110.

Shen, H., Cheng, X., Cai, K. ve Hu, M.-b. 2009. Detect overlapping and hierarchical community structure in networks. Physica A: Statistical Mechanics and its Applications, 388 (8), 1706–1712.

Wagenseller, P., Wang, F. ve Wu, W. 2018. Size matters: A comparative analysis of community detection algorithms. IEEE Transactions on Computational Social Systems, 5 (4), 951–960.

Wang, T., Wang, H. ve Wang, X. 2015. A novel cosine distance for detecting communities in complex networks. Physica A: Statistical Mechanics and Its Applications, 437, 21–35.

Wang, X., Liu, G. ve Li, J. 2017. Overlapping community detection based on structural centrality in complex networks. IEEE Access, 5, 25258–25269.

Wen, X., Chen, W.-N., Lin, Y., Gu, T., Zhang, H., Li, Y., Yin, Y. ve Zhang, J. 2019. A maximal clique based multiobjective evolutionary algorithm for overlapping community detection. IEEE Transactions on Evolutionary Computation, 21 (3), 363–377.

Xie, J., Szymanski, B. K. ve Liu, X. 2011. SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process, IEEE 11th International Conference on Data Mining Workshops, 11 Aralık, Vancouver, Canada.

Yang, G., Zheng, W., Che, C. ve Wang, W. 2020. Graph-based label propagation algorithm for community detection. International Journal of Machine Learning and Cybernetics, 11 (6), 1319–1329.

You, X., Ma, Y. ve Liu, Z. 2020. A three-stage algorithm on community detection in social networks. Knowledge-Based Systems, 187, 104822.

Zhang, Y. ve Yeung, D. Y. 2012. Overlapping community detection via bounded nonnegative matrix tri-factorization, Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 12 Ağustos, Beijing, China.

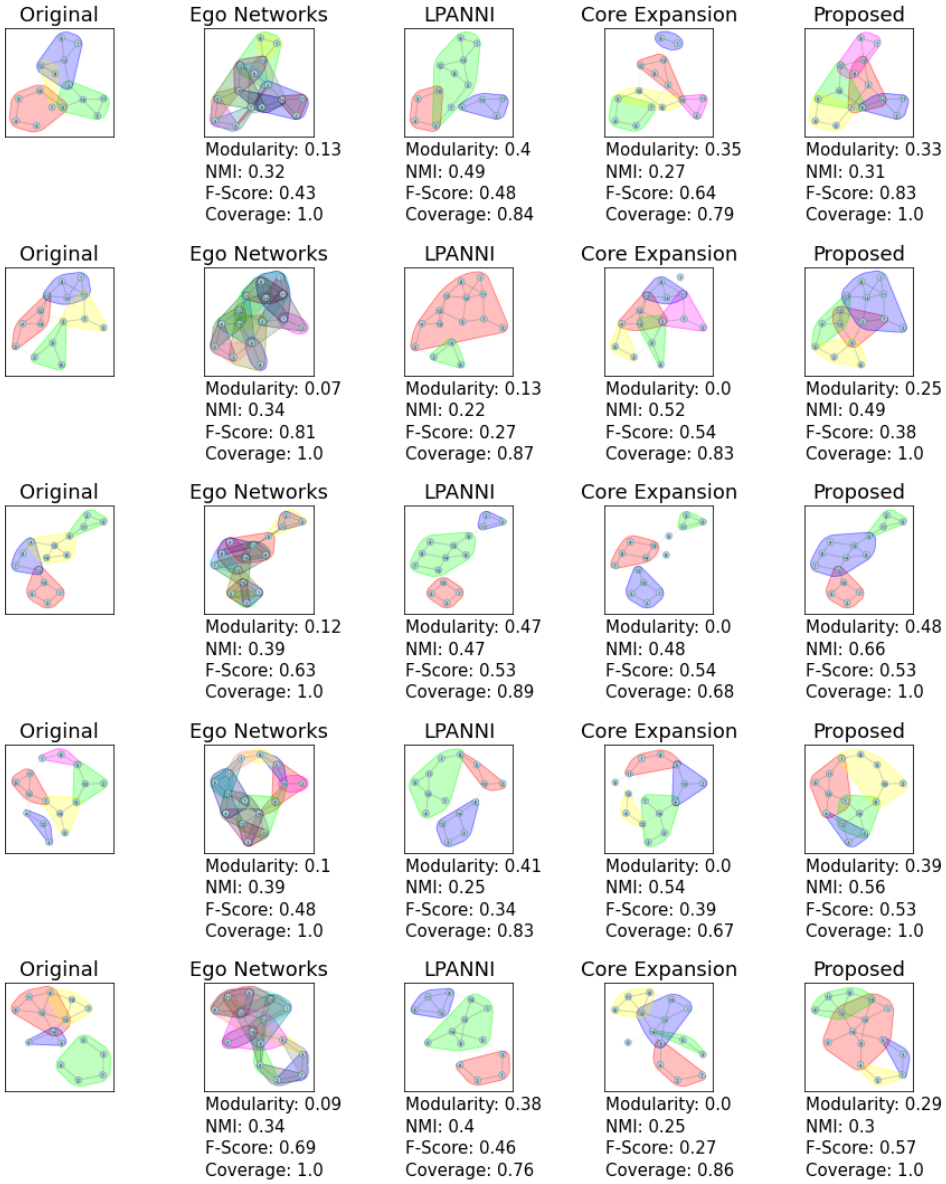
Zhang, Y., Liu, Y., Jin, R., Tao, J., Chen, L. ve Wu, X. 2020. GLLPA: A graph layout based label propagation algorithm for community detection. Knowledge-Based Systems, 206, 106363.

Zhi-Xiao, W., Ze-Chao, L., Xiao-fang, D. ve Jin-hui, T. 2016. Overlapping community detection based on node location analysis. Knowledge-Based Systems, 105, 225–235.

Zhou, Q., Cai, S. ve Zhang, Y. 2019. Detecting overlapping community structure with node influence. IEEE Access, 7, 171223–171234.

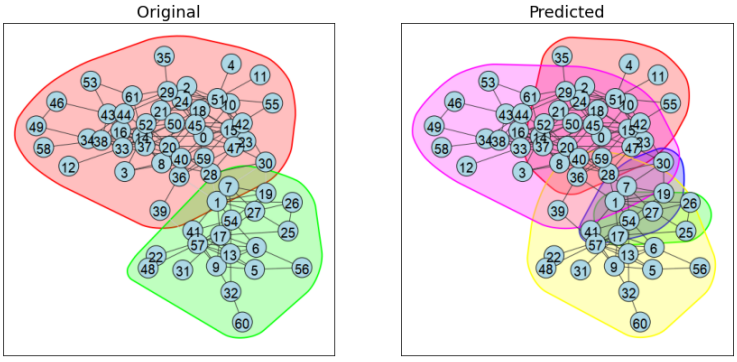
EKLER

EK 1

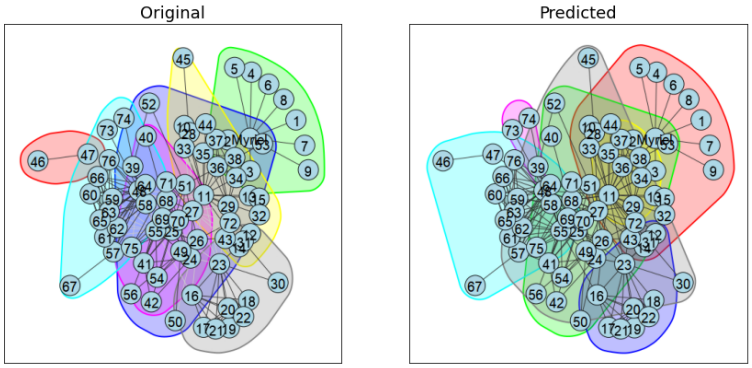


Örtüşen topluluk tespiti algoritmasının sentetik veri kümesi üzerindeki sonuçları (15 köşe)

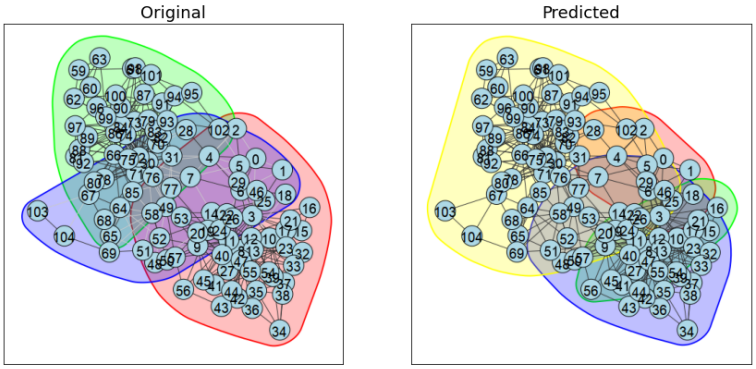
EK 2



(a) Dolphins



(b) Lesmis



(c) Polbooks

Örtüşen topluluk tespiti algoritmasının gerçek veri kümeleri üzerindeki sonuçları