T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SOSYAL AĞLARDA TOPLULUKLARI KEŞFETMEK İÇİN ÇOK AMAÇLI GENETİK ALGORİTMA KULLANIMI

YÜKSEK LİSANS TEZİ

Ertan BÜTÜN

(111129103)

Tezin Enstitüye Verildiği Tarih : 14 Haziran 2013
Tezin Savunulduğu Tarih : 7 Haziran 2013

Tez Danışmanı : Doç.Dr. Mehmet KAYA
Diğer Jüri Üyeleri : Doç.Dr. Arif GÜLTEN
Doç.Dr. A.Bedri ÖZER

HAZİRAN-2013

ÖNSÖZ

Yüksek lisans tez çalışmamda değerli hocam Sayın Doç. Dr. Mehmet KAYA'ya gerek yönlendirme gerekse motivasyon olarak katkılarından dolayı teşekkür ederim.

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

Sayfa No

ÖZET

Sosyal ağ analizi (SAA) 2000'li yıllardan itibaren gittikçe önem kazanan bir konudur. SAA sosyal bilimler, bilgisayar bilimleri, biyoloji vb. birçok bilim dalının ilgi alanına girmektedir. SAA canlı ya da cansız varlıklar arasında bulunan ilişkileri bilgi ağı şeklinde ele alıp karmaşık ağların daha iyi anlaşılması ile ilgilenir. Ağdaki aktörler arası bağlantı tahmini, kullanıcıların davranışlarından örüntü çıkarımı, topluluk keşfi SAA alanındaki çalışma konularından bazılarıdır.

Bu çalışmada sosyal ağlarda topluluk keşfi için çok amaçlı bir genetik algoritma öneriliyor. Çok amaçlı optimizasyon yaklaşımı birden çok kriteri olan problemlerin çözümünde iyi sonuçlar vermektedir. Sosyal ağlarda topluluk keşfi problemi de çok amaçlı optimizasyon problemi olarak ele alınabilir. Literatürde olan uygunluk fonksiyonları önerilen etkili bir seçim yöntemi ile kullanılarak sosyal ağlardaki topluluklar keşfediliyor. Benzer çalışmaların çoğu, başlangıç popülasyonun oluşturulmasında ve yeni bireyler elde etme için kullanılan mutasyon adımında klasik yöntemleri kullanmışlardır. Çalışmamızda topluluk keşfi probleminin yapısına uygun bir yöntemle bu işlemler yapılıyor. Önerilen yöntem gerçek ve yapay ağlar üzerinde test ediliyor, elde edilen sonuçlar benzer çalışmalarla karşılaştırılıp daha iyi sonuçlar elde edildiği gözlemleniyor.

Anahtar Kelimeler: Karmaşık ağlar, topluluk keşfi, çok amaçlı evrimsel algoritma.

SUMMARY

USING MULTIOBJECTIVE GENETIC ALGORITHM FOR THE COMMUNITY DISCOVERY IN SOCIAL NETWORKS

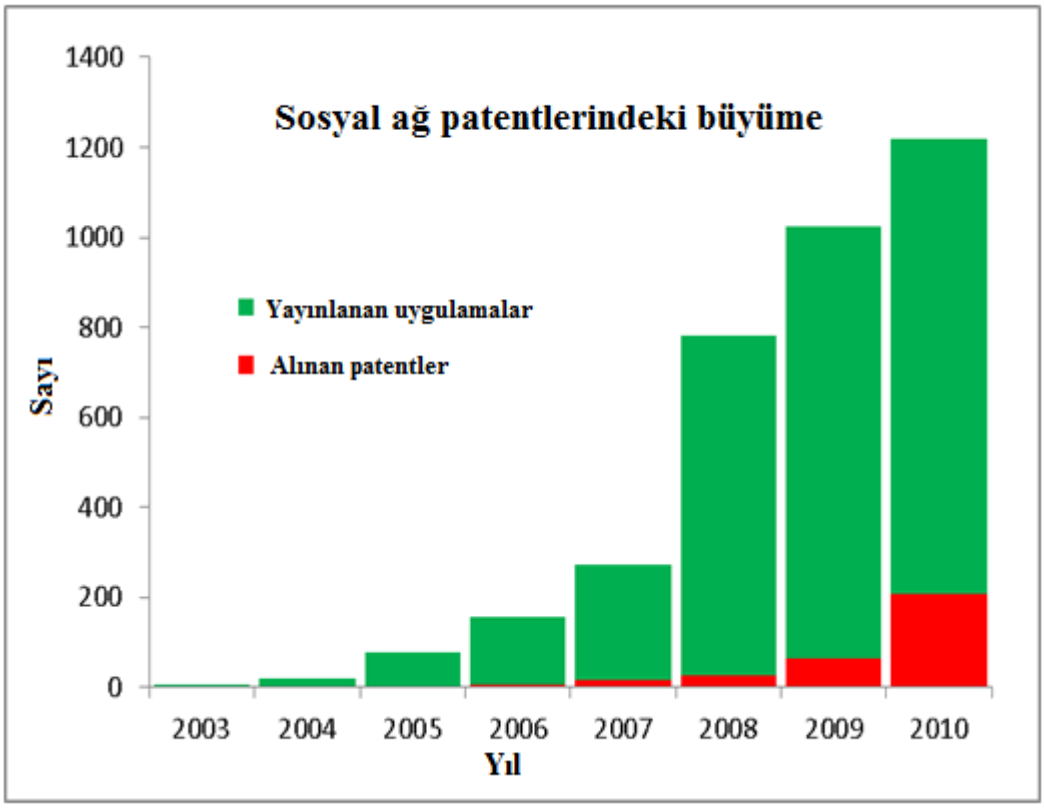
Social network analysis (SNA) is a topic of interest gaining more attention since 2000 years. SNA is in the scope of many different branches of science such as social science, computer science, and biology etc. SAA takes an interest in better understanding complex network by handling relations among animate and inanimate entities as information network. Link prediction among actors in the network, pattern discovery on users' behaviors and community discovery are some study topics in SAA.

In this study a multi-objective genetic algorithm is proposed for community discovery in social networks. Multi-objective optimization approaches give better results on problems having multiple criteria. The problem of community discovery in social networks can be considered as a multi-objective optimization problem. Communities are discovered in social networks by using fitness functions in the literature with proposed effective selection approach. Many of the similar studies use classic methods when creating initial population and mutation step which is used for creating new individuals. In our study these processes are made with respect to method which is suitable community discovery problem structure. The proposed method is tested on synthetic and real networks, it is observed that better results are obtained by compared to a similar study.

Keywords: Complex networks, community discovery, multi-objective evolutionary algorithm.

1. GİRİŞ

Sosyal ağlar bireyler (aktörler) ve bireyler arasındaki ilişkilerden oluşan yapılardır. Bireyler insanlar, organizasyonlar, canlılar, bilgiler olabilir. İlişkiler arkadaşlık, beğeni, iş birlikteliği, fonksiyonellik, kaynak transferi ya da bilgi akışı, satın alma vs. şeklinde olabilir. Sosyal ağ analizi (SAA) bu ilişkileri ve bireylerin davranışlarını inceler. Ağdaki verileri ilişkiler üzerinden irdelemeye çalışır. Sosyal ağda aktörler diğer aktörleri nasıl etkiliyor, bu aktörler bir grup oluşturuyorsa aktörlerin grup içi ve grup dışı ilişkileri nasıldır, bu grupların fonksiyonelliği nedir, grupların aralarındaki ilişkiler nasıldır şeklindeki soruları araştırır. Sosyal ağların yapısını daha iyi anlamayı, sosyal ağlarda saklı olan bilgileri ortaya çıkarmayı hedefler. SAA son yıllarda gittikçe ilgi çeken bir konudur. Şekil 1.1.'de [1] sosyal ağ ile ilgili patent ve uygulamalardaki sayının yıllara göre artışı dikkat çekicidir.



Şekil 1.1 Sosyal ağ ile ilgili patent ve uygulamaların yıllara göre dağılımı

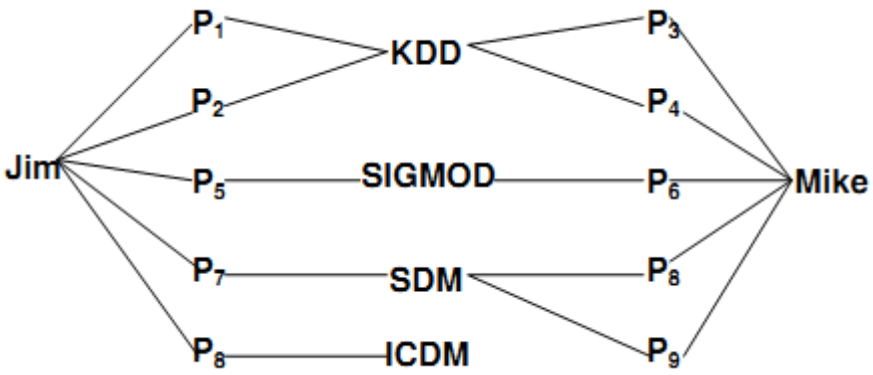
Stanley Milgram 1967'te Amerika'da dünya küçüktür fenomenini ortaya atmıştır. Bunun için şöyle bir deney yapılmıştır; Amerika'da bir grup insandan bir mesajı belirlenmiş bir kişiye yakınları arkadaşları vasıtasıyla ulaştırmaları istenmiştir. Deney sonucunda ortalama beş kişi ile bu mesajın bu insana ulaştığı görülmüştür. Kolombiya Üniversitesi'ndeki araştırmacılar bu deneyi ilerletmiş, dünya üzerinde herhangi iki kişinin internet ya da telefon üzerinden iletişime geçmesi ortalama 5-7 aracı ile gerçekleşmiştir. Dünyadaki insanların ya da başka canlıların dünyayı saran bir örümcek ağı gibi birbirlerine bağlı olduğu yapıları incelemek analiz etmek birçok noktada ufuk açıcı şeyler karşımıza çıkarmakta ve çıkaracaktır.

Sosyal ağ analizi biyologların, matematikçilerin, bilgisayar bilimcilerin, ekonomistlerin, sosyologların ve daha birçok bilim adamların ilgi alanına girmektedir. Internet kullanıcılarının ilgi alanlarının, alışverişlerinin, internet sayfalarındaki gezintilerinin analizi, ağdaki aktörler arasında olası bağlantı tahmini [2,3], kullanıcıların davranışlarından örüntü çıkarımı [4], topluluk keşfi [5,6,7], terörist ağının analizi ile terörist gruplarına karşı önlem alma SAA'nin uygulama alanlarından bazılarıdır. Örneğin ülkeler arası ekonomik ilişkiler bir sosyal ağda gösterilebilir. Ülkeler arası ticaretler yatırımlar bu ağda analiz edilerek ülkelerin ekonomik durumları başka ülkelere etkileri daha iyi anlaşılabilir.

1.1. Sosyal Ağ Çeşitleri

Sosyal ağların yapıları standart değildir. Bazı ağlarda düğümler arası bağlantılar yönlü bazılarında yönsüzdür. Bağlantıların yönsüz olduğu ağlarda bağlantılar, düğümler arasında sadece bir ilişki olduğunu gösterir, yönlü bağlantılar ise bu ilişkide etkileyen ve etkilenen tarafları da gösterir. Bağlantılar tek tip de olabilir ya da birden fazla bağlantı çeşidinin olduğu ağlar da olabilir. Aynı şey düğümler için de geçerlidir. Bu farklılıklar bazen ele alınan sosyal ağın doğal yapısından kaynaklanır, bazen de sosyal ağ için önerilen modelden kaynaklanır.

Örneğin akademi dünyasını ele aldığımızda yazar işbirliğini gösteren bir sosyal ağda düğümler yazarları, bağlantılar yazarlar arasındaki işbirliğini gösterir. Bu sosyal ağda düğümler ve bağlantılar tek tip ve yönsüzdür. Bu ağa yazarların atıf aldıkları ya da atıfta bulundukları yazarları da katarsak bağlantıların yönlü olduğu bir sosyal ağ oluşur.



Şekil 1.2 Yazar konferans makale ağı

Akademi dünyası için yazar-konferans-makale sosyal ağı da oluşturulabilir. Şekil 1.2'de [3] bunla ilgili bir model görülmektedir. Bu ağda yazarlar, konferanslar ve makaleler birer düğüm şeklindedir. Yazarların, makalelerin, makalelerin yayınlandığı konferanslar ve dergilerin düğüm olduğu, makalelerdeki referansların, makalelere yapılmış atıfların yönlü bağlantılar olduğu kapsamlı bir ağ da oluşturulabilir.

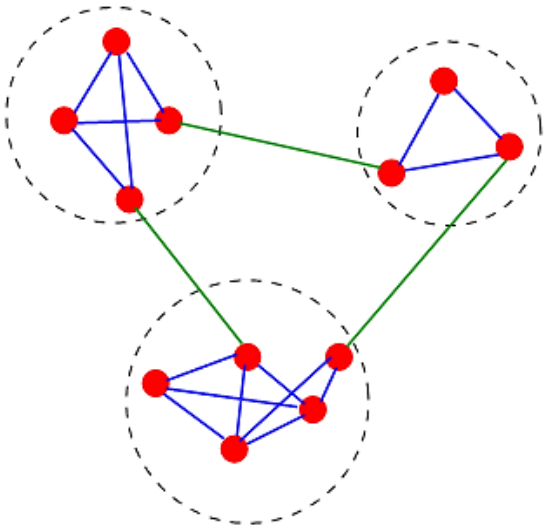
1.2. Topluluk Tanımı

Sosyal ağdaki bireyleri düğüm, bireyler arasındaki ilişkileri bağlantı olarak ifade edebiliriz. Newman ve Girvan'nın [8] topluluk tanımı şöyledir: Topluluklar kendi grubu içindeki düğümlerle çok sayıda bağlantıların olduğu, diğer gruplardaki düğümlerle ise az sayıda bağlantıların olduğu yapılardır. Şekil 1.3'te [9] örnek bir ağda topluluklar gösterilmektedir.

Bir sosyal ağı sembolize etmek için Pizzuti'nin [10] gösterimini kullanılacaktır. $G = (V, E)$ sosyal ağın graf olarak sembolüdür. Burada V düğümlerin kümesi, E ise bağlantıların kümesidir. Topluluk V'lerin oluşturduğu gruplardır. Burada her bir düğüm kendi grubu içinde yoğun, diğer gruplarla seyrek bağlantılara sahip olmalıdır. Bir düğümün derecesi o düğümün bağlantı sayılarının toplamıdır.

$$k_i = \sum_j Aij \tag{1.1}$$

burada A, G grafının komşuluk matrisidir.



Şekil 1.3 Toplulukları gösteren örnek bir ağ

G'nin alt kümesi olan S'nin düğümlerini şu denklemle gösterebiliriz:

$$k_i(S) = k_i^{in}(S) + k_i^{out}(S) \tag{1.2}$$

$$k_i^{in}(S) = \sum_{j \ni S} A_{ij} \tag{1.3}$$

$k_i^{in}(S)$, i düğümünün S topluluğu içindeki bağlantılarıdır.

$$k_i^{out}(S) = \sum_{j \not\ni S} Aij \tag{1.4}$$

$k_i^{out}(S)$ ise i düğümünün topluluk dışı bağlantı sayısıdır. Radicchi'nin [11] topluluk tanımına göre

$$k_i^{in}(S) > k_i^{out}(S), \forall i \ni S \tag{1.5}$$

koşulu sağlanıyorsa bu topluluk güçlü bir topluluktur. Ancak topluluktaki tüm düğümlerin topluluk içi bağlantıların sayısının toplamı, düğümlerin topluluk dışı bağlantıların sayısının toplamından büyükse bu zayıf bir topluluktur. Bunun formülü de şu şekildedir:

$$\sum_{j \ni S} k_i^{in}(S) > \sum_{j \ni S} k_i^{out}(S) \quad (1.6)$$

Ağdaki topluluk yapıları çeşitlilik gösterebilir. En basit şekli ayrık topluluklardır. Şekil 1.3'te bunun bir örneği gösterilmektedir. Bu topluluklarda her bir düğümün bir topluluğu vardır. Kesişen topluluklarda ise bazı düğümler birden fazla topluluğa dahil olabilirler. Bu yapılarda topluluklar iç içe geçmiştir. Örneğin bir facebook ağında bir kişinin hem arkadaş grubu hem aile grubu hem de iş arkadaşlarının olduğu bir grubu olabilir. Böyle bir ağda kişi üç faklı toplulukta görünüyordur. Başka bir topluluk tipi de hiyerarşik topluluklardır. Hiyerarşik topluluklarda topluluklar birbirleri içerisinde hiyerarşik bir yapı oluştururlar. Bir topluluğun içerisinde daha küçük topluluklar, yine bu topluluklar içerisinde başka topluluklar yer alır. İnsanın biyolojik yapısı bu yapıya uygun bir örnektir. İnsan organlardan, organlar dokulardan, dokular hücrelerden oluşur.

Toplulukları keşfetmek, ağdaki saklı bilgileri ortaya çıkarmamıza yarar. Bir sosyal ağda hangi bireylerin birbirleri ile yakın ilişki içinde olduğu, hangi bireylerin kendi grupları içerisinde aktif rol oynadığı, hangi bireylerin gruplar arasında geçiş noktasında durduğu gibi soruları ağdaki toplulukları keşfederek anlayabiliriz. Bir bilgisayar ağı topluluklar arası trafik minimum olacak şekilde topluluklara bölünebilir, böylece ağdaki bant genişliği verimli bir şekilde kullanılabilir. Bir protein ağında topluluklar hangi protein grubunun hangi biyolojik olayda önemli bir role sahip olduğunu anlamamıza yardımcı olurlar. Bir akademik yazar işbirliği ağında hangi yazarlar arasında ortak ilgi alanı var, hangi yazarlar birbirlerine daha yakın gibi soruların cevaplanmasında topluluk keşfi ışık tutar. İnternet ağında topluluklar kullanıcıların benzer ilgi alanlarından benzer alışveriş alışkanlıklarını görmeyi sağlar. Böylece kullanıcılara uygun ürün tavsiyesi yapılabilir.

Topluluk keşfi disiplinler arası çalışmaya da imkân tanır. Örneğin beyindeki birçok sinir hücreleri birbirleri ile bağlantılıdır ve bunlar bir ağ oluşturlar. Bilgisayar bilimi bu ağdaki toplulukları bulur, tıp bilimi ise bu toplulukların beyindeki hangi işi gerçekleştirdiğini araştırır.

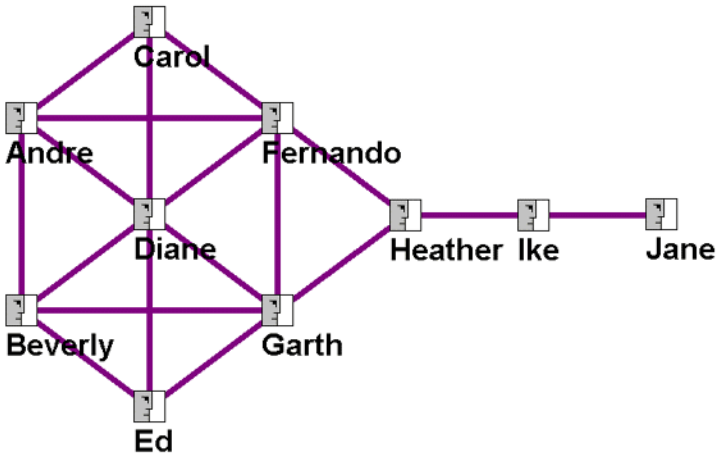
1.3. Topluluk Kavramları

1.3.1. Derece Merkeziliği

Derece merkeziliği (degree centrality) bir sosyal ağda hangi aktörlerin güçlü, etkili ve kritik olduğunu anlamak için kullanılan bir ölçüt birimidir. Derece merkeziliği yüksek olan bireyler popüler aktif bireylerdir. Bir düğümün derecesi o düğümün bağlantılı olduğu düğüm sayısıdır. Şekil 1.4'te [12] derece merkeziliği en yüksek olan düğüm *Diahe* düğümüdür.

1.3.2. Yakındalık Merkeziliği

Bir düğümün diğer tüm düğümlere olan en kısa yol uzunluklarının ortalaması alınarak hesaplanır. Bu ölçüt birimi düğümün diğer tüm düğümlere yakınlık değerini gösterir. Tüm düğümlere uzaklığı en kısa olan düğüm ağın yakındalık derecesi en yüksek olan düğümüdür. Bu düğümler ağdaki bilgi akışını en iyi gözlemleyebilen düğümlerdir. Şekil 1.4'te Fernando ve Garth düğümleri yakınlık merkeziliği (closeness centrality) en yüksek olan düğümlerdir.



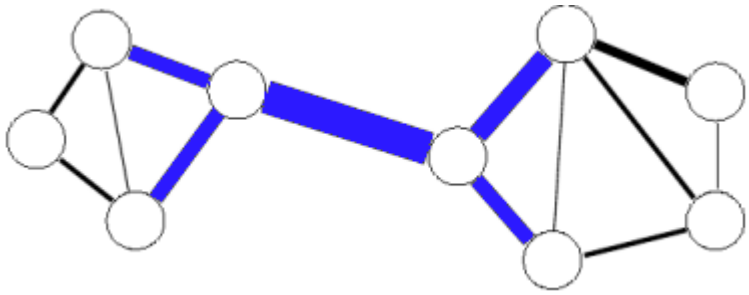
Şekil 1.4 Kite ağı

1.3.3. Arasındalık Merkeziliği

Freeman [13] tarafından bulunmuştur. Ağdaki grupların kesişim noktasındaki düğümleri belirlemek için kullanılır. Ağda her bir düğümün tüm düğümlere olan en kısa yol uzunlukları hesaplanır. Bu hesaplamalar sonucunda her bir düğümden kaç defa geçildiği bilgisi bize arasındalık değerini verir. En yüksek değere sahip olan düğüm arasındalık değeri en iyi olan düğümdür. Şekil 1.4'te Heather düğümü arasındalık merkeziliği (betweenness centrality) en yüksek olan düğümdür.

1.3.4. Kenar Arasındalık Merkeziliği

Girvan ve Newman [14] Freeman'nın [13] düğümler için önerdiği arasındalık kavramından esinlenerek bu kavramı kenarlar için geliştirmiştir. Arasındalık merkeziliğine benzer. Arasındalık merkeziliğinde düğümler esas alınırken kenar arasındalık merkeziliğinde kenarlar esas alınır. Her bir düğüm için diğer tüm düğümlere olan uzaklıklar hesaplanır. Bu hesaplamadan sonra kenarlardan kaç defa geçildiği bilgisi kenar arasındalık merkeziliği (edge betweenness centrality) değerini verir. Şekil 1.5'te [15] kenar arasındalık merkeziliğini gösteren örnek bir ağ gösterilmiştir. Bu ağda kenarların kalınlıkları kenar arasındalık değerinin simgeliyor. Topluluk yapılarının olduğu ağlarda toplulukları birbirlerine bağlayan kenarlar en kısa yollar üzerinde en çok geçilen kenarlardır. Bu ölçüt birimi sosyal ağlarda toplulukları keşfetmek için kullanılmıştır. En yüksek kenar arasındalık değerine sahip kenarlar toplulukları birbirlerine bağlayan kenarlardır. Kenar arasındalık değerinin hesaplanmasında başlangıç düğümünden itibaren enine öncelikli arama yapılıyor.



Şekil 1.5 Kenar arasındalık merkeziliğini gösteren örnek bir ağ

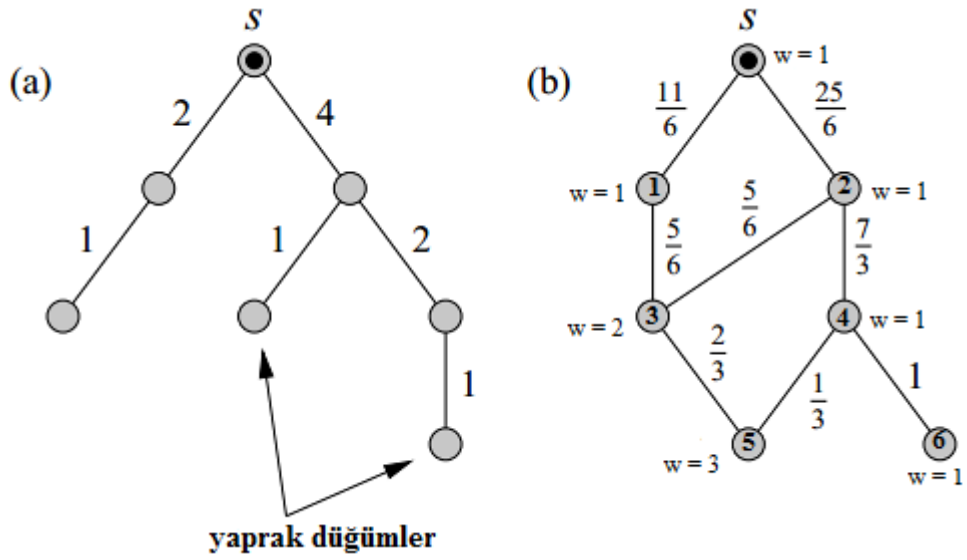
Algoritmanın [8] adımları şöyledir:

1) Başlangıç düğümü olan s düğümünün uzaklık değerine $d_s = 0$ ağırlığına $w_s = 1$ verilir.
2) s düğümünün komşusu olan her i düğümünün uzaklığı $d_i = d_s + 1 = 1$, ağırlığı ise $w_i = w_s = 1$ olarak verilir.
3) Bu i düğümlerinin komşuları olan her j düğümü için aşağıdaki üç adım uygulanır:
    a. Eğer j düğümü için bir uzaklık atanmamışsa uzaklık için $d_j = d_i + 1$ ağırlık için $w_j = w_i$ atanır.
    b. Eğer j için bir uzaklık atanmış ve $d_j = d_i + 1$ ise j düğümünün ağırlığı $w_j \leftarrow w_j + w_i$ olarak belirlenir.
    c. Eğer j düğümünün bir uzaklığı varsa ve bu da $d_j < d_i + 1$ koşulunu sağlıyorsa herhangi bir şey yapılmaz
4) Ağda uzaklığı atanmamış hiçbir düğüm kalmayana kadar 3. adım tekrarlanır.

Bu adımlar sonucunda bulunan her bir i düğümünün ağırlığı, s düğümüden i düğümüne kaç tane farklı yol olduğunu gösterir. Bu aşamadan sonra kenar ağırlık değerini hesaplamak için şu adımlar uygulanır:

1) Ağdaki her bir yaprak düğüm (t) bulunur. Yaprak düğümler bir önceki aşamada ağırlıklar hesaplanırken ağın en altında kalan düğümlerdir. s düğümünden başka düğümlere gidebilmek için t düğümlerinden geçilen hiçbir yol yoktur.
2) t düğümünün komşusu olan tüm i düğümlerinin kenar değeri t den i'ye doğru w_i / w_t'dir.
3) t düğümlerinin kenar değerleri hesaplandıktan sonra yukarıda kalan kenarların değerleri o kenara komşu olan aşağıdaki kenarların değerleri toplamına 1 ilave edilerek hesaplanır.
4) Başlangıç düğümü olan s düğümüne ulaşıncaya kadar 3. adım tekrarlanır.

Şekil 1.6'da [8] kenar arasındalık değerinin hesaplanması için Newman ve Girvan'ın kullandığı örnek iki ağ verilmiştir. Şekil 1.6 a)'da başlangıç düğümünden diğer tüm düğümlere olan en kısa yollar sadece bir tanedir. Alternatif kısa bir yol hiçbir düğüm için yoktur. Böyle bir ağda kenar arasındalık değerini kolay bir şekilde hesaplanıyor. Şekil 1.6 b)'de ise bazı düğümler için farklı kısa yolların olduğu bir ağ vardır. Yukarıda verilen



Şekil 1.6 Kenar arasındalık değerinin hesaplandığı örnek iki ağ

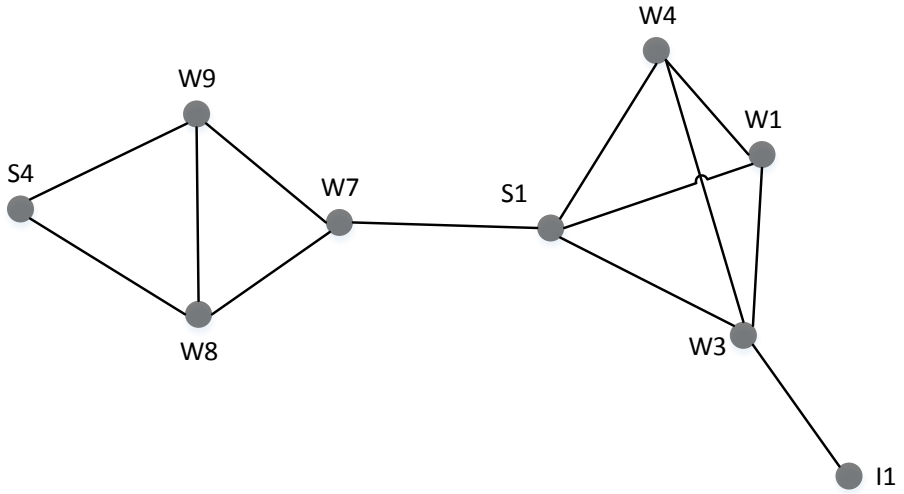
algoritma adımlarına göre önce ağırlıklar hesaplanmıştır. Daha sonra yaprak düğümlerden yukarıya doğru kenarların arasındalık değerleri bulunmuştur. 5 ve 6 numaralı düğümler yaparak düğümlerdir. Yukarıya doğru ilk kenarların değerleri komşu düğümünün ağırlığının yaprak düğümlerinin ağırlığına oranı ile bulunur. 4-5 kenarının değeri 1/3, 4-6 kenarının değeri ise 1'dir. 2-4 kenarı hesaplanırken 4-5 ve 4-6 kenar değerleri toplanır ve 1 ilave edilir o da bize 7/3 değerini verir.

1.3.5. Özvektör Merkeziliği

Bir düğümün bir ağdaki etki derecesini ölçmek için kullanılır. Derece merkeziliği yüksek olan düğümler için aktif düğümler demiştik. Düğümün özvektör derecesi bunu daha nitelikli bir şekilde ölçer. Bir düğümün ağ içinde etki derecesini gösterir. Şekil 1.7'de [16] örnek bir ağ için özvektör, derece, yakındalık ve arasındalık merkeziliği değerleri gösterilmiştir. En yüksek derece merkeziliğine (eigeanvector centrality) sahip düğümler W3 ve S1 iken, en yüksek özvektör merkeziliğine sahip düğüm S1'dir.

Necmi Gürsakal Sosyal Ağ Analizi kitabında özvektör merkeziliği için şöyle yazmıştır [17].

"… özvektör merkeziliği, bağlantıların sayısına olduğu kadar kalitesine de bağımlıdır. Eğer bir düğümün az sayıda yüksek kaliteli bağlantısı varsa bu düğümün çok sayıda ortalama sahip bir düğümü özvektör merkeziliği anlamında geçebilir. Arama motoru Google'ın web sayfalarını sıralarken kullandığı PageRank bu bağlamda çalışmaktadır ".



Özvektör merkeziliği		Derece merkeziliği	Yakındalık merkeziliği	Arasındalık merkeziliği
S1	(.498)	W3, S1	S1	S1
W3	(.472)	W9, W8, W7, W1, W4	W7	W7
W1, W4	(.438)	S4	W3	W3
W7	(.254)	I1	W1, W4	W8,W9
W8, W9	(.159)		W8, W9	W1, I1, W4, S4 (0)
I1	(.147)		I1	
S4	(.099)		S4	

Şekil 1.7 Özvektör, derece, yakındalık ve arasındalık merkeziliği gösterilen bir ağ

Geri kalan kısımda 2. bölümde topluluklar için kullanılan yaygın ölçüt birimlerinden bahsedilecek, 3. bölümde topluluk keşfi için önerilen bazı yaklaşımların algoritmaları birbirlerine göre avantajları ve dezavantajları anlatılacak, 4. bölümde çalışmamızda topluluk keşfi için önerilen çok amaçlı genetik algoritma tabanlı yöntemimizin detayları ve test sonuçları anlatılacak 5. bölümde ise çalışmamızın sonuçları değerlendirilecektir.

2. ÖLÇÜT BİRİMLERİ

Sosyal ağda toplulukların ölçüt birimi için bir standart yoktur. Bir sosyal ağda toplulukların keşfi için önerilen yaklaşımlardan hangisinin en iyi olduğu hakkında kesin bir şey ortaya konamamıştır. Bu bölümde en yaygın ölçüt birimlerinden *modülerlik* (*modularity*) ve *normalize edilmiş ortak bilgi'*den (*normalize mutual information*, NMI) bahsedilecektir.

2.1. Modülerlik

Ölçüt birimlerinden en yaygın olanı Newman ve Girvan tarafından önerilen *modülerlik*'tir [8]. Bu ölçüt biriminde ağdaki topluluk sayısına k denmiş, kxk şeklinde bir e matrisi düşünülmüştür. e matrisinin herhangi bir elamanı e_{ij}, i topluluğundaki düğümlerle j topluluğundaki düğümler arasındaki bağlantı sayısının ağdaki toplam bağlantı sayısına oranıdır.

$$Tr\ \mathrm{e} = \sum_i e_{ii} \tag{2.1}$$

$Tr\ e$, i topluluğunun düğümlerinin kendi aralarındaki bağlantı sayısının ağdaki tüm bağlantı sayısına oranıdır. Burada ağda sadece bir topluluğun olması $Tr\ e$ değerini maksimum değer olan 1 yapar, fakat tek topluluk ağ için bir anlam ifade etmez. *Modülerlik* formülü ise şöyledir:

$$Q = \sum_i (e_{ii} - \alpha_i^2) = Tr\ \mathrm{e} - \| \mathrm{e}^2 \| \tag{2.2}$$

Modülerlik hesaplanırken ele alınan ağ için rastgele bir ağ oluşturulur. Bu rastgele ağın düğüm sayısı, bağlantı sayısı ve toplulukları orijinal ağ ile aynıdır. Ancak düğümler arasındaki bağlantılar rastgele seçilir. Formülde α_i, rastgele ağdaki i topluluğunu ifade eder. i topluluğunun *modülerlik* değeri, bu topluluktaki bağlantı sayısının α_i topluluğundaki bağlantı sayından farkıdır. Ağın *modülerlik* değeri ise toplulukların *modülerlik* değerlerinin toplamıdır. Bu formül şu şekilde de gösterilmiştir [18]:

$$Q = \frac{1}{2m} \sum_{i,j \in V} \left[A_{ij} - \frac{k_i k_j}{2m}\right] \delta(c_i, c_j) \quad (2.3)$$

k_i, i düğümünün derecesi; m ağdaki toplam bağlantı sayısı; $k_i k_j / 2m$, i düğümü ile j düğümü arasında bağlantı olma ihtimalidir; A_{ij} ise şöyledir:

$$A_{ij} = \begin{cases} 1 & eğer\ i\ ve\ j\ düğümleri\ birbirlerine\ bağlı\ ise \\ 0 & değilse \end{cases} \quad (2.4)$$

$\delta(c_i, c_j)$ ise şu anlama gelmektedir:

$$\delta(c_i, c_j) = \begin{cases} 1 & i\ ve\ j\ düğümleri\ aynı\ topluluk\ içinde\ ise \\ 0 & değilse \end{cases} \quad (2.5)$$

Fortuna and Barthelemy [19] *modülerlik* ölçüt biriminin çözünürlük limiti olduğu söylemişlerdir. Buna göre ağın büyüklüğüne bağlı olarak toplulukların büyüklükleri belli bir değerin altında ise *modülerlik* başarısız olmaktadır. Büyük ağlar içinde küçük toplulukların olduğu durumda topluluklar çok bariz de olsa *modülerlik* iyi sonuç verememektedir.

Newman ve Girvan'nın önerdiği *modülerlik* yönsüz ağlar içindir, Leicht ve Newman [18] *modülerliği* yönlü ağlar için geliştirmişlerdir. Denklem 2.3'te iki düğüm arasında bağlantı olma ihtimali düğümlerin derecelerine bağlıdır. Dereceleri yüksek olan düğümlerin aralarında bağlantı olma ihtimali yüksektir. Yönlü ağlarda durum biraz değişiyor. Yönlü bir ağda bir düğümden diğer düğüme doğru bir bağlantının olma ihtimali düğümünün kendine doğru ve düğümden dışarı doğru olan bağlantıların durumuna göre değişir. Leicht ve Newman [18] bu durumu şöyle açıklıyor: A ve B diye iki düğüm düşünün. A düğümünün dışarıya doğru bağlantı sayısı yüksek, içeriye doğru bağlantı sayısı düşük olsun. B düğümü ise bu durumun tam tersine sahip olsun. Bu durumda bir bağlantının A'dan B'ye doğru olma ihtimali tam tersi yönde bağlantı olma ihtimalinden daha yüksektir. Ağda B'den A'ya bir bağlantının olmasının, A'dan B'ye bir bağlantının olmasından *Modülerlik* açısından katkısı daha fazladır. Bu denklem 2.6'da verilen yönlü ağlar için modülerlik formülünde görülmektedir.

$$Q = \frac{1}{m} \sum_{i,j \in V} \left[A_{ij} - \frac{k_i^{in} k_j^{out}}{m} \right] \delta(c_i, c_j) \quad (2.6)$$

k_i^{in}, i düğümünün içeri doğru olan bağlantı sayısı; k_j^{out}, j düğümünün dışarı doğru olan bağlantı sayısıdır. Bu formülde öncekinden farklı olarak paydadaki 2 katsayısı çıkarılmıştır ve düğümlerin dereceleri yönlere göre alınmıştır.

Yönlü ağlar ve kesişen topluluklar için de *modülerlik* geliştirilmiştir [20]. Kesişen düğümlerde bir düğümün birden fazla topluluğa ait olması söz konusudur. Düğümün topluluklara aitlik derecesi, düğümün topluluklara olan bağlantı sayılarının oranına göre değişir. Nicosia vd. [20] her bir düğümün topluluklara aitlik oranını *belonging factors* diye adlandırdıkları bir dizi ile göstermektedirler: $[\alpha_{i,1}, \alpha_{i,1} \dots \alpha_{i,|C|}]$.

$$\sum_{c=1}^{|C|} \alpha_{i,c} = 1 \quad (2.7)$$

i düğümünün topluluklara olan ait derecelerinin toplamı 1 olarak verilmiştir. i düğümünden j düğümüne doğru olan bir bağlantı $l = (i, j)$ ile gösterilmiş ve bu bağlantının topluluğa aitlik katsayısı da şöyle gösterilmiştir:

$$\beta_{l,c} = \mathcal{F}(\alpha_{i,c}, \alpha_{j,c}) \quad (2.8)$$

Denklem 2.6'da $\delta(c_i, c_j)$, i ve j düğümü aynı toplulukta ise 1 değilse 0 değerini alıyordu. Kesişen topluluklarda için bunun yerine $r_{i,j}$ ve $s_{i,j}$ kullanılmıştır. Yönlü ağlarda kesişen topluluklar için bir topluluğun *modülerlik* formülü şöyle gösterilmiştir:

$$Q_{ov} = \frac{1}{m} \sum_{i,j \in V} \left[r_{i,j} A_{ij} - s_{i,j} \frac{k_i^{in} k_j^{out}}{m} \right] \quad (2.9)$$

$r_{i,j}$, i düğümünden j düğümüne doğru olan bağlantının c topluluğunda olma katsayısı veya oranıdır.

$$r_{i,j} = \beta_{l(i,j),c} = \beta_{l,c} = \mathcal{F}(\alpha_{i,c}, \alpha_{j,c}) \quad (2.10)$$

Ağdaki tüm topluluklar için *modülerlik* değeri şöyledir:

$$Q_{ov} = \frac{1}{m} \sum_{c \ni C} \sum_{i,j \in V} \left[r_{i,j} A_{ij} - s_{i,j} \frac{k_i^{in} k_j^{out}}{m} \right] \quad (2.11)$$

$s_{i,j}$ için iki ayrı formül verilmiştir. Birincisi c topluluğu içindeki bir düğümden başlayan bir bağlantının aitlik katsayısı:

$$\beta_{l(i,j),c}^{out} = \frac{\sum_{j \ni V} \mathcal{F}(\alpha_{i,c}, \alpha_{j,c})}{|V|} \quad (2.12)$$

İkincisi bir düğümden başlayan bir bağlantının c topluluğu içine doğru gitmesinin aitlik katsayısı:

$$\beta_{l(i,j),c}^{in} = \frac{\sum_{i \ni V} \mathcal{F}(\alpha_{i,c}, \alpha_{j,c})}{|V|} \quad (2.13)$$

Yönlü ağlarda kesişen topluluklar için *modülerliğin* son hali şöyledir:

$$Q_{ov} = \frac{1}{m} \sum_{c \ni C} \sum_{i,j \in V} \left[r_{i,j} A_{ij} - \frac{\beta_{l(i,j),c}^{out} k_i^{in} \beta_{l(i,j),c}^{in} k_j^{out}}{m} \right] \quad (2.14)$$

2.2. NMI

Ölçüt birimi için başka bir yaklaşım da önerilen yöntemi, toplulukları bilinen ağlarla test etmektir. Bulunan sonuçların, gerçek topluluklara benzerlik oranı yöntemin doğruluk derecesini göstermektedir. Bunun için toplulukları bilinen gerçek ağlar [21,22] ya da yapay sosyal ağlar [14,23] kullanılmaktadır. Girvan ve Newman'ın (GN) [14] oluşturduğu yapay sosyal ağ 128 düğümden oluşmaktadır. Her bir düğümün ortalama derecesi 16'dır. Ağ 4 gruba bölünmüştür ve her grubun düğüm sayısı 32'dir. Ağdaki düğümlerin derecelerinin ve gruplardaki düğümlerin sayılarının eşit olması ağı gerçeklikten oldukça uzaklaştırmaktadır. Ancak önerilen algoritmanın, sınırları çok belli olan bu ağda test edilmesi doğruluğu hakkında yine de bilgi verir. Fakat gerçek sosyal ağlardaki performansı düşündürücü olur. Lancichinetti [23], GN'nin bu sınırları keskin olan yapay sosyal ağını geliştirmiş ve gerçeğe daha yakınlaştırmıştır. Lancichinetti'nin [23] oluşturduğu yapay ağda düğümlerin ortalama bir derecesi vardır, düğümler verilen parametre sayısı kadar

birden fazla topluluğa ait olabilir, ağın topluluk sayısı değişebilir, düğümlerin topluluk içi ve dışı bağlantı sayılarının oranları değiştirilebilir. Topluluk keşfi için önerilen algoritmaların çoğu GN'nın yapay sosyal ağında çok iyi performans gösterirken, Lancichinetti yaklaşımı ve benzeri yapay sosyal ağlarda algoritmaların gerçek limitleri ortaya daha iyi çıkmaktadır.

Ayrık toplulukların benzerliklerini ölçmek için Danon bir formül (NMI) önermiştir [24] . NMI şöyledir:

$$I(A,B) = \frac{-2\sum_{i=1}^{C_A}\sum_{j=1}^{C_B} N_{ij} log\left(\frac{N_{ij}N}{N_{i.}N_{.j}}\right)}{\sum_{i=1}^{C_A} N_{i.} log\left(\frac{N_{i.}}{N}\right) + \sum_{j=1}^{C_B} N_{.j} \log\left(\frac{N_{.j}}{N}\right)} \quad (2.15)$$

NMI'da gerçek toplulukların düğümleri ile bulunan toplulukların düğümleri bir N matrisinde gösterilir. N_{ij}'de i gerçek toplulukların tarafı, j buluna toplulukların tarafıdır. A gerçek topluluklar, B bulunan topluluklar; C_A ve C_B, A ve B'nin eleman sayıları; $N_{i.}$ i satırındaki elemanların toplamı; $N_{j.}$ j kolonundaki elemanların toplamıdır. Bu formül 0 ile 1 arasında bir sonuç verir. Eğer sonuç 1 ise gerçek topluluklar ile bulunan topluluklar birebir aynıdır.

Lancichinetti ise Danon'un yaklaşımını ilerleterek kesişen toplulukların da benzerliğini ölçebilen bir yaklaşım sunmuştur [25].

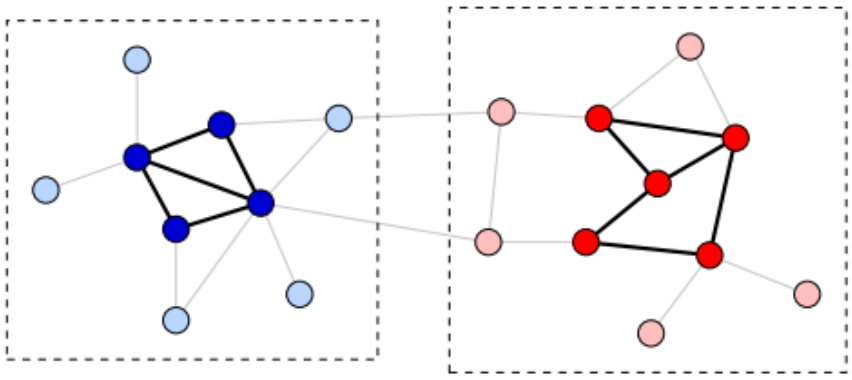
3. TOPLULUK KEŞFİ İÇİN ÖNERİLEN BAZI YAKLAŞIMLAR

Bu bölümde topluluk keşfi için önerilen bazı yöntemleri ana karakteristik özelliklerine göre gruplandırarak anlatılacaktır.

3.1. Bölmelendirme Algoritmaları

Bu algoritmalar toplulukları birbirine bağlayan bağlantıları bulup bunları ağdan kaldırarak toplulukları ortaya çıkarırlar.

Girvan ve Newman (GN) [8] tarafından yapılan çalışma topluluk keşfi problemi için önerilen yaklaşımların en popüler olanlarındandır. Bu yaklaşıma 26 Mart 2013 tarihi itibari ile yapılan atıf sayısı 3779'dur. GN sosyal ağlarda topluluk keşfi için önerilen yaygın yaklaşımlardan biri olan toplayıcı metotların kısıtlarından bahsetmiştir. Bu metotlar ağdaki düğümlerin benzerlik ya da daha başka ölçüt birime göre gruplanmasına dayalıdır. Ağda sıkı bir şekilde birbirlerine bağlı düğümleri topluluk olarak bulabilirken topluluğa daha seyrek bağlantılarla bağlı olan düğümleri bulmada yetersiz kalabilmektedirler. Şekil 3.1'deki [8] örnek ağda sıkı bir şekilde birbirlerine bağlı olan gruplar daha koyu renkle gösterilmiştir. Toplayıcı metotlar şekilde koyu renkle gösterilen aralarında sık bağlantıların olduğu grupları bulabilirken açık renkle gösterilen toplulukla seyrek bağlar içeren düğümleri bulmada iyi değildirler. Hâlbuki bu düğümler de gösterilen topluluklara aittir. Bunlar da doğru topluluklara koyulmalıdır.



Şekil 3.1 Topluluk içinde yoğun ve seyrek bağlantıları gösteren örnek bir ağ

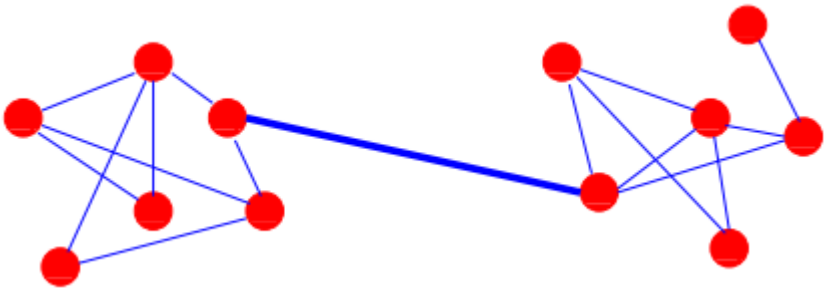
GN, bölmelendirme metotlarının birbirlerine fazla benzemeyen düğümleri bulup bunlar arasındaki bağlantıyı kaldırarak toplulukları bulduğundan bahsetmiştir. GN'nin önerdiği yaklaşım ise arasındalık değeri yüksek bağlantıları (kenarları) bulup bunları ağdan kaldırmaktır. Arasındalık değeri için daha önce topluluk kavramları bölümünde bahsedilen yine bu yazarların bulmuş olduğu kenar arasındalık değeri kullanılmıştır. Arasındalık değeri yüksek olan kenarlar, düğüm gruplarını birbirine bağlayan kenarlardır. Bu kenarlar ağdan kaldırılarak topluluklar keşfedilmektedir. Şekil 3.2'de [9] iki düğüm grubunu sadece bir kenar birbirine bağlamaktadır. Bu kenar, ağda arasındalık değeri en büyük olan kenardır. Bu kenarın kaldırılması ile topluluklar ortaya çıkmaktadır.

Bu algoritmanın adımları şöyledir:

1) Ağdaki tüm kenarlar için kenar arasındalık değeri hesaplanır.
2) Kenar arasındalık değeri en yüksek olan kenar ağdan çıkarılır.
3) Kalan kenarların kenar arasındalık değeri yeniden hesaplanır.
4) 2. adım tekrarlanır.

Bu algoritma daha önce bahsedilen *modülerlik* ölçüt birimi en yüksek değerine ulaştığında durur.

Radicchi vd. [11] topluluk keşfi için GN algoritmasına benzeyen bir bölmelendirme algoritması önermişlerdir. Burda da yine toplulukları birbirine bağlayan kenarların ağdan çıkarılması yaklaşımı vardır. Radicchi vd. [11], GN algoritmasının ağdaki güçlü toplulukları iyi bulabildiğini zayıf toplulukları ise bulmakta o kadar iyi olmadığını, aynı zamanda GN algoritmasının hesaplama zamanın da yüksek olduğunu belirtmişlerdir. GN algoritması tekrar tekrar, ağdaki tüm kenarların kenar arasındalık değerini hesaplamakta ve



Şekil 3.2 Kenar arasındalık kavramını resmeden örnek bir ağ

bu da hesaplama zamanını artırmaktadır. Radicchi vd. [11]'nin önerdiği yaklaşım bu kenar hesaplama işini lokal olarak yapmakta ve hesaplama zamanını düşürmektedir.

Radicchi vd. [11] toplulukları birbirlerine bağlayan kenarları farklı bir şekilde bulmaktadırlar. Topluluklar, içerisinde yoğun bağlantılar bulunduran düğüm gruplarıdır ve bu bağlantıların topluluk içinde bir döngü oluşturması muhtemeldir. Toplulukları birbirlerine bağlayan kenarların ise döngü oluşturması düşük ihtimaldir. Bu yaklaşım için Radicchi vd. [11] kenar kümelenme katsayısı (edge clustering coefficient) ölçüt birimini önermişlerdir. Kenar kümelenme katsayısı bir kenarın içinde bulunduğu üçgenlerin yine bu kenarı oluşturan düğümlerin komşularının oluşturabileceği üçgenlerin sayısına oranıdır. Formülü ise şöyledir:

$$C_{i,j}^{(3)} = \frac{z_{i,j}^{(3)}}{\min[(k_i-1),(k_j-1)]} \tag{3.1}$$

$z_{i,j}^{(3)}$, i ve j düğümleri arasındaki bağlantının içinde olduğu üçgen sayısı; $\min[(k_i - 1), (k_j - 1)]$, bu bağlantının içinde bulunabileceği maksimum üçgen sayısıdır. Bir bağlantının $C_{i,j}^{(3)}$ değeri ne kadar yüksekse bu bağlantının bir topluluk içinde olma olasılığı o kadar yüksektir. Aynı şekilde $C_{i,j}^{(3)}$ değeri ne kadar düşükse bu bağlantının toplulukları birbirine bağlayan bir bağlantı olma olasılığı o kadar yüksektir. $z_{i,j}^{(3)}$ değeri ve $\min[(k_i - 1), (k_j - 1)]$, 0 olduğunda tanımsız bir durum ortaya çıkacağından bunun önüne geçmek için $z_{i,j}^{(3)}$ değerine 1 ilave edilmiştir:

$$\tilde{C}_{i,j}^{(3)} = \frac{z_{i,j}^{(3)}+1}{\min[(k_i-1),(k_j-1)]} \tag{3.2}$$

Üçgen yerine bir döngü şeklinde olması için formülün parametrik yapısı şöyledir:

$$\tilde{C}_{i,j}^{(g)} = \frac{z_{i,j}^{(g)}+1}{s_{i,j}^{(g)}} \tag{3.3}$$

Burada $z_{i,j}^{(g)}$, i ve j düğümleri arasındaki bağlantının g uzunluğunda kaç tane çokgenin içinde bulunduğu sayısı, $s_{i,j}^{(g)}$ ise i ve j düğümlerinin komşularının oluşturabileceği maksimum g uzunluğundaki çokgen sayısıdır.

Kenar kümelenme katsayıları bulunduktan sonra en düşük değere sahip kenarın ağdan çıkarılıp çıkarılmayacağı, yine bu çalışmada tanımlanmış güçlü ve zayıf topluluk tanımlarına göre yapılmaktadır. Bu tanımlara topluluk tanımları bölümde değinmiştik.

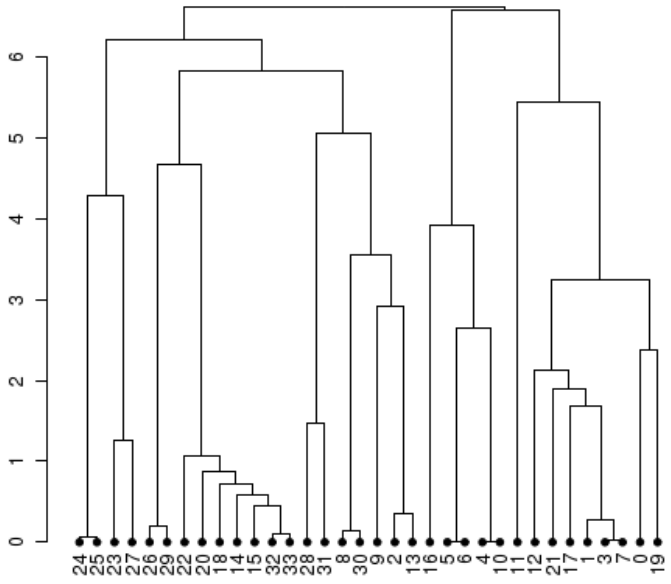
3.2. Modülerlik Tabanlı Optimizasyon Algoritmaları

GN'ın [8] bulmuş olduğu *modülerlik* ölçüt birimi, topluluk keşfi problemini ele alan yaklaşımlarda oldukça kullanılmıştır. Bu bölümde *modülerliği* ölçüt birimi olarak kullanan bazı algoritmalara değineceğiz.

GN algoritması yüksek hesaplama zamanına ihtiyaç duymaktadır. En kötü durumda $O = (m^2 n)$ ile çalışır. m, ağın kenar sayısı; n, ağın düğüm sayısıdır. Newman [26], GN algoritması ile aynı prensipte ve daha hızlı çalışan bir yaklaşım önermiştir. Bu algoritma en kötü durumda $O = ((m + n)n)$ ile çalışmaktadır. Çoğu gerçek ağlarda kenar sayısı yoğun değildir, kenar sayısı ile düğüm sayısı birbirine yakındır. Böyle ağlara seyrek ağlar(sparse graph) denir. Seyrek ağlarda çalışma zamanı $O = (n^2)$'dir. Başlangıçta ağdaki her bir düğüm bir topluluk olarak ele alınır. Daha sonra en büyük modülerlik artışını sağlayan topluluk çiftleri birleştirilir. Aralarında bağlantı bulunmayan toplulukların birleştirilmesinden modülerlik artışı olmayacağından bunlar birleştirilmez ve *modülerlik* hesaplanmaz.

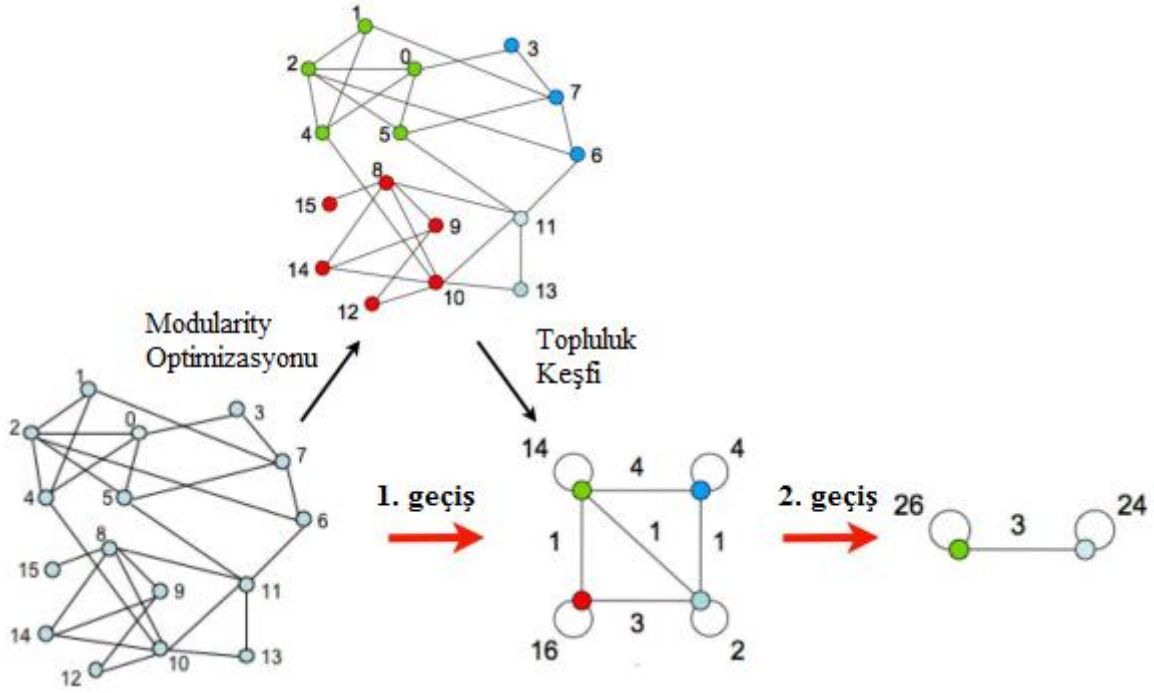
Clauset vd. [27], Newman algoritmasını hız açısından geliştirmişlerdir. Bu algoritma $O = (md \log n)$ zamanda çalışmaktadır. d, ağın dendogram derinliğidir. Dendogram bir ağın ağaç yapısı şeklinde hiyerarşik olarak gösterimidir. Şekil 3.3'te [28] Zachary'nin Karate kulübü ağı [21] için dendogram gösterilmiştir. Seyrek ağlarda $d \sim \log n$'dir. Seyrek ağlar için çalışma zamanı $O = (nlog^2 n)$'dir.

Genelde gerçek ağlarda seyrek bağlantılar olduğundan dolayı komşuluk matrisinde veriler, matrisi yoğun bir şekilde kaplamamaktadır. Bu yüzden Clauset vd. ağın komşuluk matrisini, performansı artıracak ve arama uzayını azaltacak etkili bir veri yapısı ile kullanmışlardır. Bu veri yapısında en büyük modülerlik artışını sağlayan topluluk çiftleri sabit zamanda bulunuyor.



Şekil 3.3 Örnek bir ağ (sağdaki) ve dendogramı (soldaki)

Blondel vd. [29], Clauset vd.'nin yaptığı çalışmaya göre daha iyi sonuç veren bir algoritma önermişlerdir. Blondel vd., Clauset vd. algoritmasının çok sayıda düğümün olduğu toplulukları bulmaya daha yatkın olduğunu bunun *moduarity* değeri için çok iyi sonuçlar vermediğini ve büyük ağlarda çok zamana ihtiyaç duyduğunu belirtmişlerdir. Blondel vd. önerdikleri metotla bu dezavantajlardan kaçınmışlardır. Bu metot da yine *modülerlik* tabanlı bir optimizasyon yaklaşımıdır. Algoritma temel olarak iki adımdan oluşmaktadır. En iyi sonuca ulaşıncaya kadar bu iki adım tekrarlanmaktadır. Birinci adımda tüm düğümler birer topluluk olarak ele alınır. Daha sonra her bir düğüm kendi topluluğundan çıkarılıp komşularının bulunduğu topluluklara sırayla dahil edilir ve *modülerlik*'teki artış değerine bakılır. En büyük artışı sağlan yer değiştirme uygulanır, diğer yer değiştirmeler uygulanmaz. İkinci adımda, yer değiştirmeden sonra oluşan topluluklar yeni birer düğüm şeklinde ele alınır. Bu topluluklardaki düğümlerin diğer topluluklarla olan bağlantılarının toplamı yeni düğümde ağırlıklı bağlantı olarak ele alınır. Bu iki adım modülerlik değerinde artış olduğu müddetçe tekrarlanır, artış durunca sonlanır. Şekil 3.4'te [29] algoritmanın adımlarını gösteren örnek bir ağ gösterilmiştir. Şekilden de görüldüğü gibi önce lokal olarak modülerlik artışına göre topluluklar bulunuyor. Daha sonra topluluklar birer düğüm şeklinde ele alınıyor. Yeni düğümler birbirlerine ağırlıklı bağlantılar şeklinde bağlanıyor. Ve tekrar *modülerlik* artışına göre topluluklar birleştiriliyor. Algoritma hiyerarşik toplulukları da bulabilmektedir. *Modülerliğin* bazı durumlarda iyi sonuç vermediğine değinmiştik.



Şekil 3.4 Blondel vd. önerdiği algoritmanın adımlarını gösteren örnek bir ağ

Büyük ağlarda toplulukların büyüklükleri belli bir ölçekten küçük olunca *modülerlik* iyi sonuç verememektedir [19]. Blondel vd. önermiş oldukları yaklaşımla ağdaki topluluk yapılarının farklı seviyelerde gözlemlenebileceğini ve böylece *modülerliğin* bu olumsuz durumdan kaçınılabileceğini ifade etmektedirler.

3.3. Evrimsel Algoritmalar

Evrimsel algoritmalar optimizasyon tabanlı algoritmalardır. Tabiattaki canlıların yaşamlarından esinlenerek bulunmuştur. Bu algoritmaların adımları canlıların üremesi, değişime uğraması ve güçlü olanların hayatta kalması esaslarına benzemektedir. Matematiksel hesaplamaların çok fazla zaman gerektiği ve arama uzayının çok büyük olduğu problemlerde kullanılmaktadırlar. Problemin arama uzayı içerisinde belirlenen uygunluk fonksiyonuna göre en iyi çözümü ararlar.

3.3.1. Tek amaçlı Evrimsel Algoritmalar

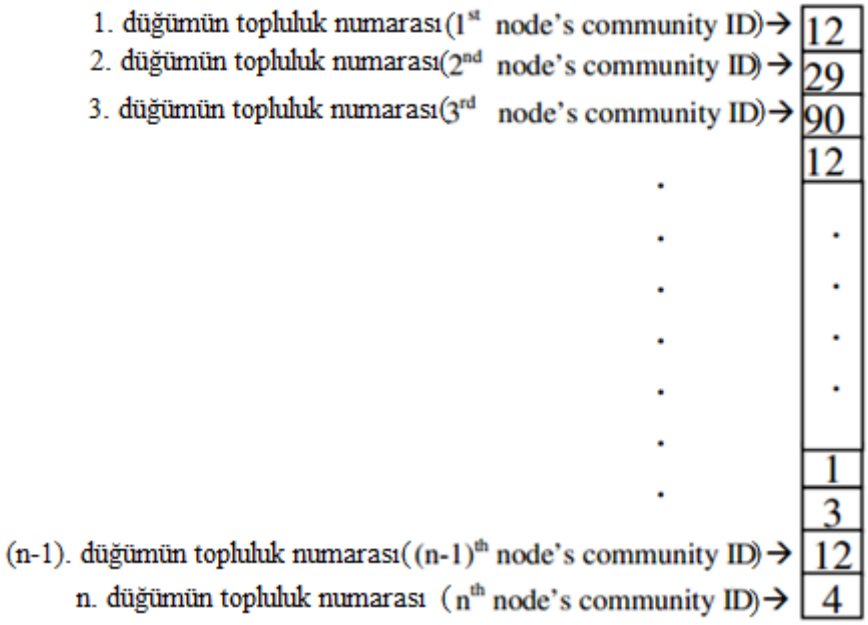
Genetik algoritmalar evrimsel algoritmalar sınıfına girer. Genetik algoritmalarda bir uygunluk fonksiyonu olur. Problem için olası çözümler bulunur. Bu çözümler bir kromozom yapısına sahip bireyler şeklinde düşünülür. Problemin verileri uygun bir şekilde

modellenir. Başlangıç olarak rastgele bireyler oluşturulur. Daha sonra mutasyon çaprazlama gibi adımlarla önceki bireylerden yeni bireyler oluşturularak daha iyi çözümler yakalanmaya çalışılır.

Tasgin ve Bingöl [30], topluluk keşfi için *modülerliği* optimize eden genetik bir algoritma önermişlerdir. Bu çalışmada *modülerlik* maksimize edilmeye çalışılmaktadır. Yapılan bazı çalışmalar topluluklar keşfetmek için topluluk sayısı ya da bir eşik değeri gibi ön bilgilere ihtiyaç duymuşlardır. Bu durum bir dezavantajdır. Tasgin ve Bingöl [30]'ün önerdiği çalışma herhangi bir ön bilgiye ihtiyaç duymamaktadır.

Problem tamsayılardan oluşan bir dizi şeklinde modellenmiştir. Bu dizi ağdaki düğümlerin ait oldukları topluluk numaralarını tutmaktadır. Dizi şeklindeki bu yapı kromozomu oluşturmaktadır. Kromozom modeli Şekil 3.5'te [30] görülmektedir. Başlangıçta her bir düğüm için 1 ile ağın düğüm sayısı (n) arasında rastgele olarak bir topluluk numarası atanır. Birbirleri ile komşu olan düğümler aynı topluluk içinde olacağından bu rastgele topluluk numarası ataması her düğümün komşuları arasından yapılır. Böylece gereksiz adımlardan kurtulmuş olunur.

Her bir kromozomun *modülerlik* değeri onun uygunluk değerini gösterir. Çaprazlama (cross-over) adımında uygunluk değerine göre kaynak ve hedef diye iki kromozom seçilir. Kaynak kromozundan bir topluluk belirlenir, hedef kromozomda karşılık gelen düğümlerin



Şekil 3.5 Algoritmanın kromozom gösterimi

topluluk numaraları bu belirlenen topluluk numarası ile değiştirilir. Böylece yeni bireyler elde edilmiş olur. Şekil 3.6'da [30] çaprazlama örneği verilmiştir.

Mutasyon adımında rastgele kromozomlar seçilir, seçilen bu kromozomlarda bir düğüm rastgele belirlenen bir topluluğa dâhil edilir.

Çalışmada, bulunan çözümün iyi olup olmadığını uygunluk fonksiyonuna bakılarak anlaşılabileceği ancak bunun bazen bazı olumsuz durumları gösteremeyebileceğinden bahsedilmiştir. Bulunan topluluk yapıları genel itibari ile iyi olabilir ancak bazı düğümlerin toplulukları doğru bulunmamış olabilir. Böyle durumların önüne geçmek için clean-up diye bir işlem de uygulanmıştır. Bu işlemde birbirine komşu olan düğümlerin aynı topluluk içerisinde olma ihtimalinin yüksek olduğundan yola çıkılmıştır. Düğümün ve komşularının ait oldukları farklı toplulukların varyansı alınmıştır. Bu değer ne kadar küçükse yukarıda bahsedilen durumun olma ihtimali o kadar düşüktür ve bulunan topluluk yapısı iyidir.

Clean-up işleminin formülü şöyledir:

$$CV(i) = \frac{\sum_{(i,j)\ni E} f_{(i,j)}}{\deg(i)}$$

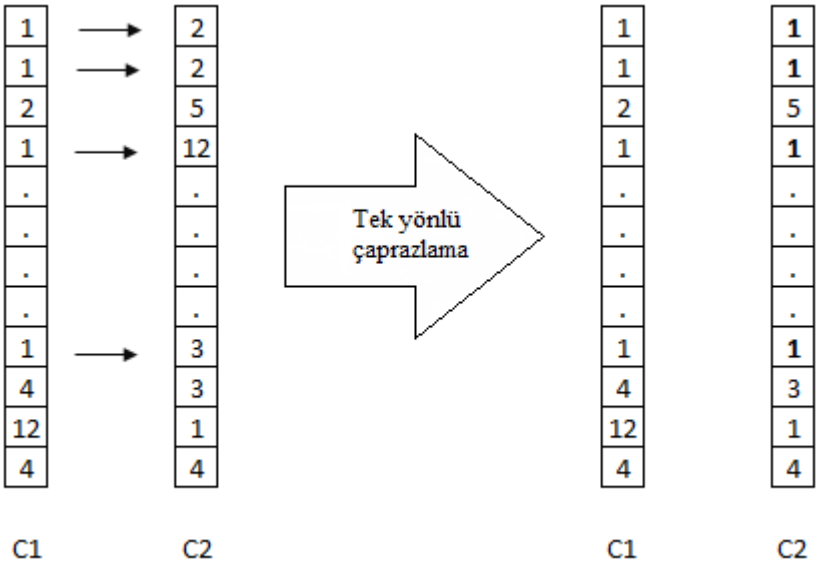
$$burada$$

$$f_{(i,j)} = \begin{cases} 1, & comm(i) \neq comm(j) \\ 0, & değilse \end{cases} \qquad (3.4)$$

$$\deg(i), i \text{ düğümünün } derecesidir$$

$$E \text{ } kenarların \text{ } kümesidir$$

$$comm(i), i \text{ düğümünün } topluluğudur$$



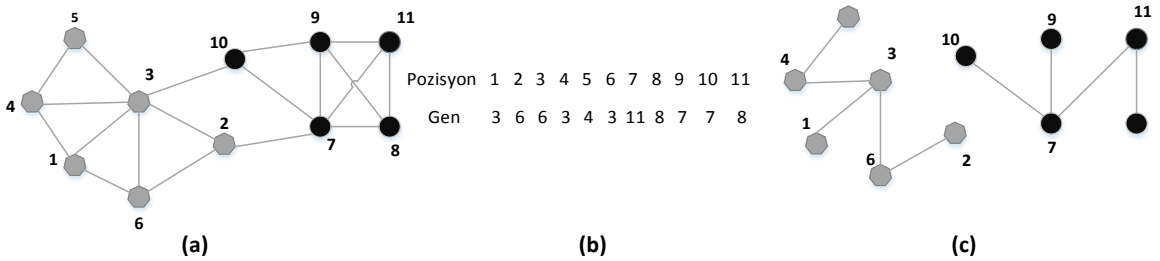
Şekil 3.6 Algoritmanın kromozon gösterimi

$CV(i)$, i düğümünün topluluk varyans değeridir. Bu hesaplanırken rastgele düğümler ele alınır ve topluluk varyans değeri hesaplanır. Bu değer belli bir eşik değerinin üstünde ise ele alınan düğüm ve komşuları aynı topluluğa dâhil edilir, değilse bir işlem yapılmaz.

Jin vd. [31] odak tabanlı komşuluk gösterimi (*locus-based adjacency representation,* LAR) modeli ile lokal arama tabanlı bir genetik algoritma önermişlerdir. Çalışmanın odak noktası mutasyon adımında önerilen lokal arama yaklaşımıdır. Var olan mutasyon yöntemlerinin topluluk keşfi için verimli olmadığı belirtilmiş, mutasyon için marjinal gen kavramı önerilmiş ve bu yaklaşım lokal arama yaklaşımı ile birleştirilerek *modülerliği* optimize eden bir yaklaşım sunulmuştur. LAR modelinin verimliliğini artırmak için başlangıç popülasyonu oluşturulurken Markov rasgele yürüyüş kullanılmıştır. Topluluk keşfinde LAR modelinin geleneksel çaprazlama yöntemleri için uygun olduğu ancak mutasyon için iyi sonuç vermediği belirtilmiştir.

LAR gösterimi Park ve Song [32] tarafından önerilmiştir. LAR gösterimi sınıflandırma problemleri için çok amaçlı yaklaşımlarda [33,34], topluluk keşfi problemi için tek amaçlı genetik algoritmalarda [35,36] ve çok amaçlı genetik algoritmalarda [10,37,38,39] kullanılmıştır. Bu gösterimde bir birey ya da kromozom n tane genden oluşmakta, her gen ağdaki bir düğüme karşılık gelmektedir. Her genin bir allele geni vardır, bu allele gen de yine ağdaki bir düğüme karşılık gelmektedir. Genle allele gen, ağdaki bir bağlantıyı temsil etmektedir. Yani i geninin allele geni j geni ise, ağda i düğümü ile j düğümü arasında bir bağlantı var demektir. LAR gösteriminin çözümlenmesi ile ağdaki topluluklar ortaya çıkar. Çözümleme sonucu topluluk sayısı otomatik olarak ortaya çıkmaktadır. Şekil 3.7'de [31] örnek bir ağ için LAR gösterimi ve çözümlenmesi gösterilmiştir.

Bir düğümün ait olduğu topluluk içinde illa ki o düğümün komşuları olmak zorundadır. Bu nedenden dolayı başlangıç popülasyonu oluşturulurken gereksiz arama uzayından kurtulmak amacıyla genlerin allele geni, genlerin komşuları arasından Markov rasgele yürüyüş ile belirlenmiştir. Böylece en iyi çözüme ulaşmak için doğru bir başlangıç yapılmıştır.



Şekil 3.7 Örnek bir ağ için LAR gösterimi ve çözümlenmesi a) 11 elemandan oluşan bir ağ; b) LAR kodlaması c) Çözümleme sonucu ortaya çıkan iki topluluk

Çaprazlama adımında tek biçimli çaprazlama kullanılmıştır. A ve B diye iki birey seçilir, bireylerin uzunluğunda 0 ve 1'lerden oluşan rasgele bir ikili vektör oluşturulur. Bu vektörde 1'in olduğu pozisyonlardaki genler A'dan, 0'ın olduğu pozisyonlardaki genler B'den alınarak yeni birey oluşturulur. Başlangıç popülasyonu oluşturulurken ağda aralarında bağlantı bulunan düğümlere göre Markov rasgele yürüyüş ile seçim yapıldığından bireylerin çaprazlanmasından oluşan yeni bireylerde de bu durum söz konusudur. Böylece gereksiz arama uzayından kaçınılmaktadır.

Mutasyon adımında var olan mutasyon yöntemlerinin dezavantajlarının önüne geçmek için marjinal gen kavramı önerilmiştir. LAR gösterimi kullanılan çalışmalardaki mutasyon yöntemleri topluluk keşfi için uygun değildir. Guimera and Amaral'a [40] göre topluluk keşfi probleminde aday çözümlerden yeni çözümler üretilirken şu işlemler yapılırsa daha iyi sonuçlar elde edilebilir: Bir düğümü bir topluluktan başka bir topluluğa taşıma, toplulukları birleştirme ya da bir topluluğu bölme. Genetik algoritmada çaprazlama işlemi bireyler üzerinde makro değişiklikler yaparken mutasyon işlemi mikro değişiklikler yapar. Topluluk keşfi probleminde de çaprazlama işlemi global düzeyde toplulukları birleştirebilirse ya da bir topluluğu bölebilirse mutasyon da lokal düzeyde bir düğümü topluluklar arasında yer değiştirebilir. Jin vd. [31], topluluk keşfi probleminde geleneksel mutasyon yöntemlerinin toplulukları birleştirdiği ya da böldüğünü bunun da iyi sonuç vermediğini belirtmişlerdir. Şöyle bir örnek verilmiştir: i ve j düğümlerinin farklı topluluklara ait olduğunu düşünelim. Mutasyon sonucunda i geni için allele gen olarak j geni seçilirse bunun sonucundan büyük bir ihtimalle iki topluluk birleşecektir. Bu ise istenilen bir durum değildir.

LAR gösteriminde bir kromozomda j geni allele genler arasında yer almıyorsa bu geni temsil eden düğüm marjinal düğümdür. Mutasyon işleminde marjinal düğüm bir topluluktan başka bir topluluğa aktarılırsa bu işlem sonucundan toplulukların birleşmesi ya da bölünmesi gerçekleşmez. n tane düğümden oluşan bir ağda i düğümünün j düğümünü

allele gen olarak alma ihtimali $p = 1/n$'dir. i düğümünün j düğümünü almama ihtimali ise $1 - p$'dir. Tüm düğümler için bu olasılık $\beta = (1 - 1/n)^n$'dir. n sonsuza gittiğinde bu değer $\lim_{n\to+\infty} \beta(n) = 1/e \approx 0.3679$ olur.10 tane düğümü olan bir ağda $p(10) = 0.3487$'dir. Bu oran 10'dan büyük değerler için $0.3487 \sim 0.3679$ aralığındadır. Kromozomun genlerine bu oranda mutasyon uygulanmıştır.

3.3.2. Çok Amaçlı Evrimsel Algoritmalar

Bazı problemlerin birden çok kriteri olabilir. Böyle problemler için sadece bir kriteri dikkate alan çözümler problemi bütünüyle ele alamazlar. Bu problemleri birden çok ama fonksiyonu ile tanımlamak daha doğru bir yaklaşım olur. Topluluk keşfinde de çok amaçlı durumlar söz konusudur. Örneğin bir ağda toplulukları belirlemede topluluk içi bağlantıların yoğun ve topluluk dışı bağlantıların zayıf olması gibi iki uygunluk fonksiyonu belirlenebilir. Yani ağda bir grup düğüme topluluk diyebilmemiz için hem düğümlerin grup içi bağlantılarının çokluğuna hem de düğümlerin gruplar arası bağlantılarının azlığına dikkat etmek gerekecektir.

Gerçek ağlarda topluluk içindeki bireyler diğer topluluklardaki bireylerden keskin bir şekilde ayrık olmayabilirler. Bazı bireyler birden çok toplulukta boy gösterme eğiliminde olabilir. Bu durum insan karakterlerine benzemektedir. Bazı insanların etkileşim içerisinde bulunduğu insanlar sınırlıdır, bazılarının ise çok geniştir. Asosyal olan insan sadece kendi topluluğu içinde etkileşim gösterme eğilimdeyken, sosyal olan insan farklı farklı topluluklarla etkileşimde bulunma eğilimdedir. Sosyal olan insanın bir sosyal ağdaki konumu toplulukların sınırlarının birbirine yakınlaştığı yerlerde iken, asosyal olan insanın konumu ait olduğu topluluğun iç kısımlarıdır. Bir insanın iş arkadaşları, ailesi, sosyal arkadaşları, spor kulübündeki arkadaşları vs. farklı sosyal grupların kesişmesine bir örnektir.

Bir ağın kesişen toplulukları da orta çıkaracak şekilde modellenmesi daha gerçekçi bir yaklaşım olur. Liu vd. [41] çalışmasında hem ayrık hem de kesişen toplulukları keşfeden çok amaçlı evrimsel bir algoritma sunmuştur. Liu vd. [41] evrimsel algoritmalarla kesişen topluluklar için bir model sunmanın zorluğundan, bu alanda yapılan çalışmaların çoğunun ayrık toplulukları keşfeden çalışmalar olduğundan bahsetmiştir.

Liu vd. [41] bu çalışmayı bağlantıların yönsüz ve ağırlıksız olduğu ağlar üzerinde gerçekleştirmişler, hem ayrık hem de kesişen toplulukları keşfedecek bir kodlama

yapmışlardır. Bu evrimsel algoritmada bir bireyin kodlanmasında iki bileşen vardır biri permutasyon bileşeni, diğeri topluluk bileşenidir. Bir birey $\boldsymbol{\mathcal{A}}$ ile ifade edilmektedir. $\boldsymbol{\mathcal{A}<P>}$, ağdaki düğümlerin permutasyonudur:

$$\boldsymbol{\mathcal{A}\langle P\rangle} = \{V_{\pi 1}, V_{\pi 2}, \dots, V_{\pi n}\} \tag{3.5}$$

Topluluk bileşeni ise $\boldsymbol{\mathcal{A}<C>}$ ile ifade edilmiştir. $\boldsymbol{C}$ ağdaki toplulukların kümesidir, $\boldsymbol{C} = \{C_1, C_2, \dots, C_n\}$. $\boldsymbol{\mathcal{A}<P>}$'yi $\boldsymbol{\mathcal{A}<C>}$'ye dönüştürmek için bir çözümleyici metot önerilmiştir.

Toplulukların niteliğini ölçmek için Lancichinetti vd.'nin [25] önerdiği *topluluk uygunluğu (community fitness)* fonksiyonu kullanılmıştır. Bu fonksiyon şöyledir:

$$f(C_i) = \frac{k_{in}^{C_i}}{(k_{in}^{C_i} + k_{out}^{C_i})^{\alpha}} \tag{3.6}$$

Bu formülde C_i, ağdaki herhangi bir topluluk; $k_{in}^{C_i}$, düğümlerin topluluk içi derecelerinin toplamı; $k_{out}^{C_i}$, düğümlerin topluluk dışı derecelerinin toplamı; α, ise toplulukların büyüklüğünü belirlemeye yarayan pozitif reel bir parametredir. $\boldsymbol{\mathcal{A}<P>}$'nin $\boldsymbol{\mathcal{A}<C>}$'ye çözümlenmesinde *topluluk uygunluğu* fonksiyonu temel alınmıştır. Bu algoritmanın sözde kodu şöyledir [41]:

```
Giriş: P = {V_π1, V_π2, ..., V_πn};
Çıkış: C;
|C|, C topluluğunun düğüm sayısını gösteriyor
begin
   C ← ∅;
   //P'deki düğümleri sırasıyla tüm topluluklara ekle
   for i = 1 to n do
   begin
     for i = 1 to |C| do
     begin
            if (f(C_j) < f(C_j ∪ v_i)) then
                  C_j ← C_j ∪ v_i;
            end;
     if(v_i hiçbir topluluğa atanmamışsa)
            C ← C ∪ {v_i};
     end;
```

$//Toplulukların\ birleştirilmesi$
$while\ (\boldsymbol{C'}de\ denklem\ 3.6'yısağlayan\ topluluklar\ olduğu\ müddetçe)$
$\quad toplulukları\ birleştir$
$end;$

$$\forall \boldsymbol{C_i} \neq \boldsymbol{C_j}, \frac{|\boldsymbol{C_i} \cap \boldsymbol{C_j}|}{\boldsymbol{C_i}} > \frac{1}{2}\ ya\ da\ \frac{|\boldsymbol{C_i} \cap \boldsymbol{C_j}|}{\boldsymbol{C_j}} > \frac{1}{2} \tag{3.7}$$

Çözümleme algoritması giriş olarak $\boldsymbol{\mathcal{A}<P>}$'yi alır, çıkış olarak $\boldsymbol{\mathcal{A}<C>}$'yi üretir. Algoritmanın adımlarına bakacak olursak düğümler permutasyon sırasına göre tek tek alınır, her bir düğümün o an var olan tüm topluluklara *topluluk uygunluğu* değeri olarak katkıda bulunup bulunmayacağı hesaplanır. Katkının sağlanacağı ilk topluluğa bu düğüm katılır. Burada akla, bir düğümün birden çok topluluğa katkıda bulunabilme ihtimali gelebilir. Zaten bu ihtimal de kesişen toplulukların ortaya çıkmasını sağlamaktadır. Bir düğümün birden çok topluluğa *topluluk uygunluğu* değeri açısından katkısı olacaksa o düğüm bu toplulukların hepsine dahil edilir. Akla gelebilecek diğer bir durum, toplulukların büyük bir oranda iç içe girmesidir. Bu istenmeyen bir durumdur. Bunun önüne geçmek için şöyle bir yol izlenmektedir: Kesişen iki topluluğun ortak bireyleri bu topluluklardan herhangi birinin toplam düğüm sayısının yarsından fazla ise bu iki topluluk birleştirilir. Bu durum yukarıdaki sözde kodun son kısmında yer almaktadır.

Çok amaçlı evrimsel algoritma için üç uygunluk fonksiyonu kullanılmıştır; $f_{kalite}(.)$, $f_{kesişim}(.)$, $f_{ayrık}(.)$.

$$f_{kalite}\ (\boldsymbol{\mathcal{A}}) = \frac{\sum_{i=1}^{m} f(C_i)}{m} \tag{3.8}$$

$f_{quality}(.)$ daha önce belirtilen *topluluk uygunluğu* [25] fonksiyonunun tüm topluklara oranıdır.

$$\boldsymbol{V}_{ayrık} = \begin{cases} v \mid v \in \boldsymbol{V}\ ve \\ |\{\boldsymbol{C'} \mid \boldsymbol{C'} \in \boldsymbol{\mathcal{A}\langle C\rangle}\ ve\ v \in \boldsymbol{\mathcal{A}\langle C\rangle}\}| > 1 \end{cases} \tag{3.9}$$

$\boldsymbol{V}_{kesişim}$ ele alınan ağda farklı topluluklara ait düğüm sayısıdır. $\boldsymbol{V}_{kesişim}$ değeri $f_{ayrık}(.)$, $f_{kesişim}(.)$ fonksiyonlarının hesaplanmasında kullanılmaktadır.

$$f_{ayrık}(\boldsymbol{\mathcal{A}}) = -|V_{kesişim}| \tag{3.10}$$

$f_{ayrık}(.)$ değeri, $\boldsymbol{V}_{kesişim}$ değerinin negatifidir. Birden fazla topluluğa ait olan düğüm sayısı ne kadar çoksa $f_{ayrık}(.)$ negatifi olması nedeniyle o kadar düşük bir değer olacaktır.

$$f_{kesişim}(\boldsymbol{\mathcal{A}}) = \sum_{v \in \boldsymbol{V}_{kesişim}} \underset{C \in \{\boldsymbol{C}' | C' \in \boldsymbol{\mathcal{A}}\langle \boldsymbol{C} \rangle \, ve \, v \in \boldsymbol{\mathcal{A}}\langle \boldsymbol{C} \rangle\}}{min} \left\{ \frac{k_v^c}{k_v} \right\} \tag{3.11}$$

$f_{kesişim}$ hesaplanırken kesişen tüm düğümler tek tek alınır. k_v^c düğümün c topluluğu ile olan bağlantı sayısı, k_v düğümün toplam bağlantı sayısıdır. Ele alınan düğümün ait olduğu tüm topluluklar için k_v^c/k_v değeri hesaplanır, bunlar arasından minimum olanı alınır. Tüm düğümler için bulunan bu minimum değerler toplanır, sonuç $f_{kesişim}$'i verir.

Tüm düğümlerin tek bir topluluğa ait olması veyahut her bir düğümün kendi başına bir topluluk olması istenmeyen durumlardır. Bu istenmeyen durumların uygunluk fonksiyonu değerleri yüksek olsa da bunlar cezalandırılmıştır, böyle durumlara en kötü değerler verilmiştir.

Hem topluluk niteliklerinin yüksek olması hem de aynı anda ayrık ve kesişen toplulukları yakalayabilmek için bireylerin fonksiyon değerlerinin maksimuma gitmeleri gerekir. Çok amaçlı yaklaşımın güzelliği burada ortaya çıkıyor; bulunan çözüm kümesinde bir bireyin $f_{ayrık}(.)$ değeri ikinci bireye baskın gelecek, ikinci bireyin de $f_{kesişim}(.)$ değeri birinci bireye baskın gelecek, $f_{kalite}(.)$ de toplulukların niteliklerini belirleyecek. Böylece çözüm kümesi aynı sosyal ağ için hem ayrık hem de kesişen topluluklar sunabilecek.

Evrimsel algoritmanın sözde kodu şöyledir [41]:

Rastgele $\boldsymbol{Pop_i}$<$\boldsymbol{P}$> *üret* $f_{ayrık}$<$\boldsymbol{Pop_i}$> , $f_{kesişim}$ <$\boldsymbol{Pop_i}$> *ve* f_{kalite}<$\boldsymbol{Pop_i}$> *değerlerini* $i = 1,2, \dots N$ *için hesapla;*
while(*durma şartı sağlanana kadar adımları tekrara*) *do*
begin
    $\boldsymbol{DPop_i} \leftarrow \boldsymbol{Pop_i}, i = 1,2, \dots, N;$
    $i \leftarrow N + 1;$
    repeat
        $\boldsymbol{DPop_k}$ *ve* $\boldsymbol{DPop_l}$ *adında rastgele iki farklı bireyi,* $\boldsymbol{DPop_1}, \boldsymbol{DPop_2}, \dots \boldsymbol{DPop_N}$ *içinden seç,*
        $if(Crowded - ComparisonOperator(\boldsymbol{DPop_k}, \boldsymbol{DPop_l})$*sonucu* $\boldsymbol{DPop'_k}$*nın* $\boldsymbol{DPop'_l}$*den iyi olduğu sınucunu sağlıyorsa*)
        then
            $\boldsymbol{DPop_i} \leftarrow \boldsymbol{DPop_k};$

$else$

$\boldsymbol{DPop_i \leftarrow DPop_l}$;

$i \leftarrow N + 1$;

$until(i > 2N)$;

$\boldsymbol{DPop_i}, i = N + 1, N + 2, \dots, 2N$ $içinde\ reverse\ operator\ işlemini\ uygula$
$Ve\ bunların\ amaç\ fonksiyon\ değerlerini\ güncelle$;

$Pop \leftarrow Fast - Non - Dominated - Sort(\boldsymbol{DPop})$;

end;

$\boldsymbol{C^i \leftarrow Pop_i} < C >, i = 1,2, \dots, N$;

Başlangıçta ağın düğümlerinin permutasyonundan rastgele bireyler oluşturularak popülasyon oluşturulmuş, bu popülasyondan yukarıda bahsedilen çözümleme metodu ile topluluklar bulunmuş ve bunların uygunluk fonksiyonları hesaplanmıştır. Sonra bu popülasyondaki düğümlerin dizilişine, düğümlerin farklı bir dizilişi eklenmiştir. Bu diziliş popülasyondan rastgele iki bireyin seçilip bunlardan uygunluk fonksiyonu değerleri yüksek olanın yeni dizilişe eklenmesi ile gerçekleştirilmiştir. Elde edilen bu yeni dizilişe tersine çevirme işlemi uygulanmıştır. Tersine çevirme işlemindeki amaç evrimsel algoritmanın doğası gereği çeşitliliği sağlamak ve en iyi bireylere ulaşmaya çalışmaktır. Tersine çevirme işleminin sözde kodu şöyledir [41]:

$\boldsymbol{Giriş}$: $\boldsymbol{P} = \{V_{\pi 1}, V_{\pi 2}, \dots, V_{\pi n}\}$;

$Çıkış$: $\boldsymbol{P'} = \{V_{\psi 1}, V_{\psi 2}, \dots, V_{\psi n}\}$;

$begin$

$1, 2, \dots, n\ arasından\ i < j\ olacak\ şekilde\ i\ ve\ j\ seç$

$V_{\psi k} \leftarrow V_{\pi k}, k = 1, 2, \dots, i - 1$;

$V_{\psi k} \leftarrow V_{\pi j - k + i}, k = i, i + 1, \dots, j$;

$V_{\psi k} \leftarrow V_{\pi k}, k = j, j + 1, \dots, n$;

end;

Tersine çevirme işleminde düğümlerin dizilişinden rastgele iki düğüm seçilir, bu iki düğüm arasında kalan düğümlerin yerleri sondakiler başa, baştakiler sona gelecek şekilde, yani ters bir dizilişle değiştirilir. Tersine çevirme işlemin uygulandıktan sonra popülasyondaki bireylerin uygunluk fonksiyonları NSGA-II [42] algoritmasına göre sıralanır. Bu adımların ne kadar uygulanacağı bir parametre ile belirlenir. Gelen parametre değeri kadar bu adımlar uygulanır. Bu adımların sonunda bir çözüm kümesi oluşur.

Gong vd.'nin [39] topluluk keşfi için önerdikleri algoritma topluluklar içindeki bağlantıları maksimum, topluluklar arasındaki bağlantıları ise minimum yapmaya çalışır. Algoritmada çok amaçlı genetik bir algoritma kullanılmıştır. Genetik kodlama olarak da tek amaçlı genetik algoritmalar bölümünde değinilen LAR gösterimi kullanılmıştır.

Uygunluk fonksiyonları için *modülerlik yoğunluğu* [43] ölçüt biriminden yola çıkılmıştır. Formülü şöyledir:

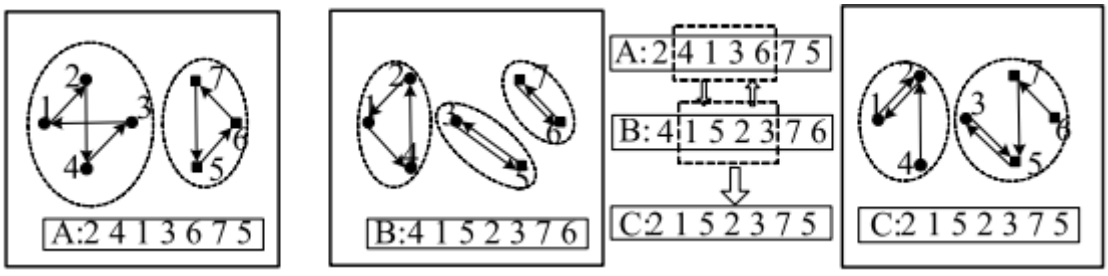
$$D = \sum_{i=1}^{m} \frac{L(V_i,V_i) - L(V_i,\overline{V_i})}{|V_i|} \tag{3.12}$$

Ağdaki düğüm grupları $S = (V_1, V_2, \dots, V_m)$ şeklinde gösterilmiştir. $L(V_1,\ V_2) = \sum_{i \ni V_1, j \ni V_2} A_{ij}$, V_1 grubundaki düğümlerle V_2 grubundaki düğümler arasındaki bağlantıların toplamıdır. . $L(V_1,\ \overline{V_2}) = \sum_{i \ni V_1, j \ni \overline{V_2}} A_{ij}$, V_1 grubundaki düğümlerin V_2 grubunda olmayan düğümlerle olan bağlantıların sayısının toplamıdır. 3.12 denklemi her grup için grup içi ve grup dışı bağlantıların farkıdır. Bu denklem ratio association [44] ve ratio cut [45] fonksiyonlarını içermektedir. Ratio association topluluk içindeki bağlantıların yoğun olması için kullanılırken ratio cut topluluklar arası bağlantıların minimum olması için kullanılmaktadır. Ratio association formülü şöyledir:

$$NRA = -\sum_{i=1}^{m} \frac{L(V_i,V_i)}{|V_i|} \tag{3.13}$$

Problem minimum optimizasyon olarak ele alındığından ratio association'ın negatifi alınmıştır. Ratio cut ise şöyledir:

$$RC = \sum_{i=1}^{m} \frac{L(V_i,\overline{V_i})}{|V_i|} \tag{3.14}$$



Şekil 3.8 iki nokta çaprazlama (two-point crossover) yöntemine bir örnek

Çaprazlama için iki nokta çaprazlama (two-point crossover) yöntemi kullanılmıştır. Bu yöntemde iki kromozom çaprazlama için kullanılır kromozomların uzunluğu dahilinde iki nokta rasgele seçilir. Bu iki nokta arasında kalan genler kromozomlar arasında yer değiştirilir. Şekil 3.8'de [39] çaprazlama için bir örnek gösterilmiştir.

Mutasyonda ise rasgele bir kromozom ve bu kromozom içinde rasgele bir gen seçilir. Seçilen bu genin allele gen değeri söz konusu genin komşuları içinden rasgele seçilir. Rasgele seçim işlemi komşular ile sınırlandığından gereksiz arama uzayından kaçınılmış olur.

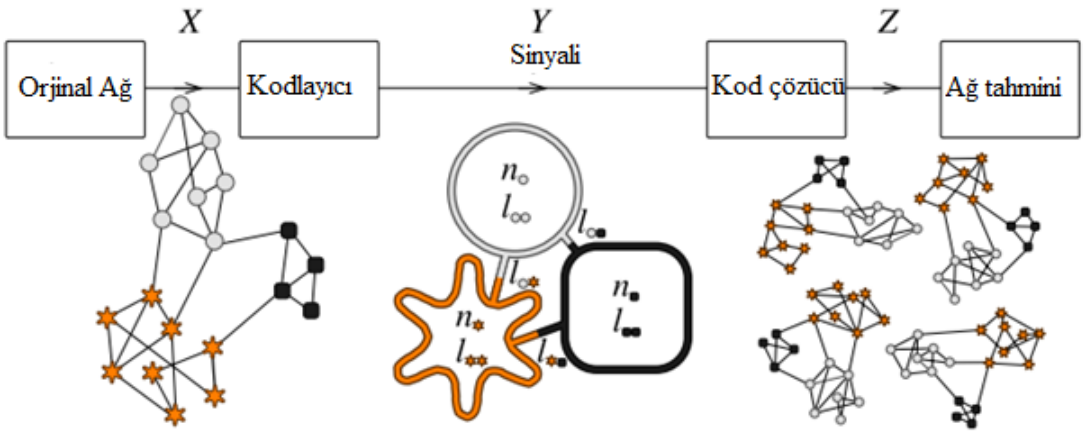
Çok amaçlı optimizasyon yaklaşımı bir çözüm kümesi sunmaktadır. Bu çözüm kümesi içinden de modülerlik ölçüt birimine göre en iyi olan seçilmektedir.

3.4. Bilgi Teorisi Tabanlı Algoritmalar

Rosvall ve Bergstrom [46], topluluk keşfi problemini ağdaki bilgiyi sıkıştırma problemi olarak ele almışlardır. Ağın komşuluk matrisi bilgisini olabildiğince az alan kapsayacak şekilde sıkıştırıp bu sıkıştırmanın çözümlenmesi ile de toplulukları ortaya çıkaran bir yaklaşım sunmuşlardır.

Bu algoritmanın çalışma prensibi Şekil 3.9'da [46] gösterilmiştir.X, ağın link yapısını göstermektedir. Bu X bilgisi, Y sinyali olarak kodlanmaktadır. Ağın yapısındaki bilgiyi olabildiğince barındıracak şekilde etkili bir sıkıştırma yapılmaya çalışılmaktadır. Y sinyali çözümlenmekte ve bunun sonucunda ağın orijinal X yapısı tahmin edilerek topluluklar ortaya çıkarılmaktadır.

Önerilen Y_i aday kodlamalardan hangisinin iyi olduğuna karar vermek için ölçüt



Şekil 3.9 Topluluk keşfi probleminin bilgi sıkıştırma ve iletişim modeli

birimleri bölümünde de bahsedilen NMI kullanılmıştır. Önerilen algoritma bağlantıların yönsüz ve ağırlıksız olduğu ağlar içindir. X ağının düğüm sayısı n ile bağlantı sayısı l ile gösterilmiştir. Y ise şöyle gösterilmiştir:

$$Y = \left\{ \boldsymbol{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}, \boldsymbol{M} = \begin{pmatrix} l_{11} & \cdots & l_{1m} \\ \vdots & \ddots & \vdots \\ l_{m1} & \cdots & l_{mm} \end{pmatrix} \right\} \tag{3.15}$$

$\boldsymbol{a}$ vektörü ağdaki n tane düğümün ait olduğu modülü (topluluğu) belirtmektedir. Modüllerin matrisi de $\boldsymbol{M}$ ile gösterilmiştir. $\boldsymbol{M}(X, \boldsymbol{a})$ ise m tane modülün orijinal ağa göre birbirlerine nasıl bağlı olduklarını göstermektedir. n_i tane düğüme sahip i modülü j modülüne l_{ij} sayıda bağlantı ile bağlıdır. a^* içinden en iyi modül ataması olası modül atamaları içinden en iyi NMI değerini veren modül ile belirlenmektedir. Şöyle formüllendirilmiştir:

$$a^* = \underset{a}{\arg\max}\, I(X;Y) \tag{3.16}$$

NMI formülü ise şöyledir:

$$I(X;Y) = H(X) - H(X|Y) = H(X) - H(Z) \tag{3.17}$$

$H(X)$, X'i tanımlamak için gereken bilgi; $H(Z)$, verilen Y'ye göre X'i tanımlamak için kullanılacak bilgidir.

Rosvall ve Bergstrom [47] yine bilgi tabanlı bir başka yaklaşım önererek bağlantıların yönlü ve ağırlıklı olduğu ağlarda toplulukları keşfeden bir algoritma sunmuşlardır. Ağda rasgele yürüyüş yapılırken bilgi akışının olasılığından yola çıkılarak topluluklar ortaya çıkarılmıştır. Rosvall ve Bergstrom [47] bir modül (topluluk) içindeki düğümlerin kendi aralarındaki bilgi akışının hızlı ve kolay olacağını modüller arasında ise geniş bir bilgi akışının olacağını belirtmişlerdir. Ağdaki bilgi akışı için proxy gibi rasgele yürüyüş kullanılmıştır. Ağda rasgele yürüyüş için etkili bir kodlama gerçekleştirilmiş topluluk keşfi problemi kodlama problemi olarak ele alınmıştır. Kodlama yapılırken hem etkili bir sıkıştırma, hem de kodlamanın ağın yapısını yansıtacak bir yapıda olması amaçlanmıştır.

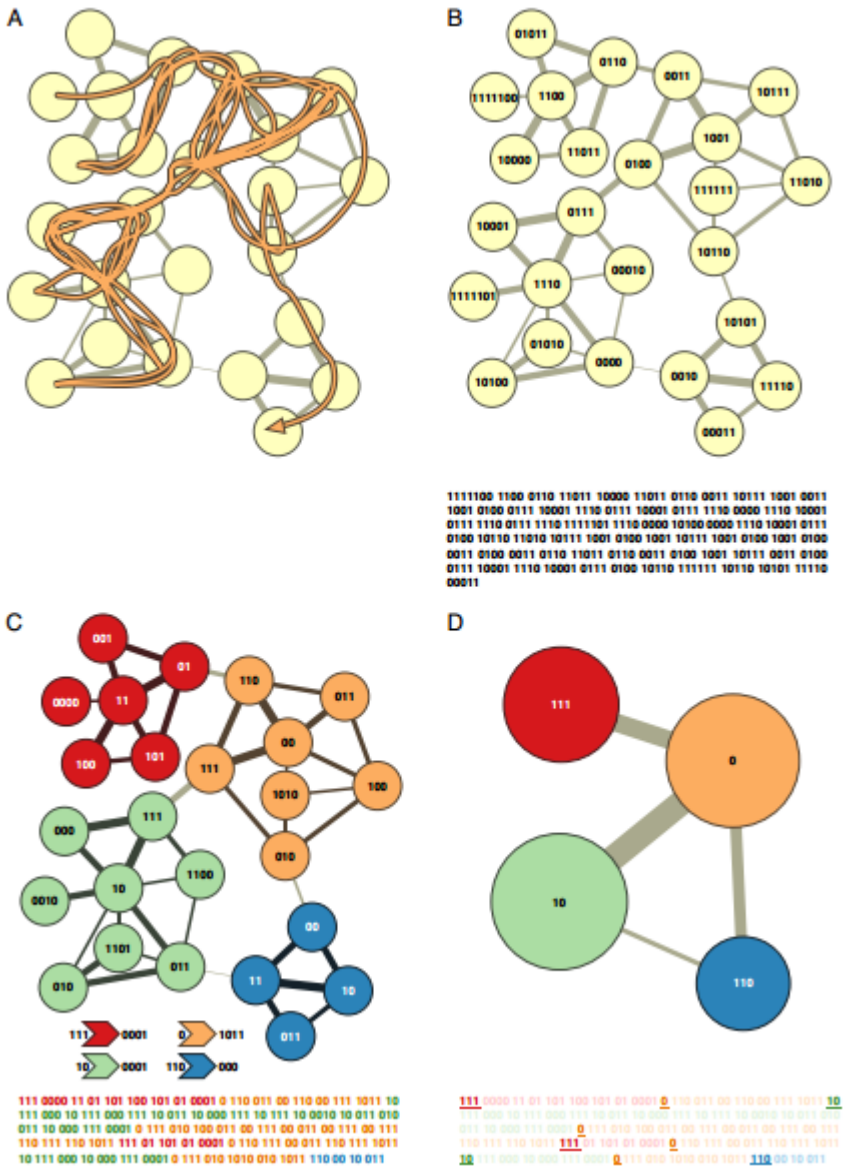
Ağ için iki düzey bir tanımlama yapılmıştır: Düğümler için kullanılan kodlama aynı zamanda toplulukları tanımlamak için de kullanılmıştır.

Önerilen algoritmanın Şekil 3.10'da [47] 25 tane düğümden oluşan ağırlıklı bir ağın topluluklarını nasıl ortaya çıkardığı gösterilmektedir. Bu ağda 71 adımdan oluşan rasgele bir yürüyüş gerçekleştirilmiştir. Şekil 3.10 A'da ağda gerçekleştirilen rasgele yürüyüş gösterilmektedir. Düğümlerin isimleri için Şekil 3.10 B'de görüldüğü gibi Huffman kodlama kullanılmıştır. Her düğümün eşsiz bir numarası vardır. Bu kodlama ile ortak olan şeyler olabildiğince en kısa şekilde kodlanarak alandan kazanma amaçlanmıştır. Kodların uzunluğu düğümlerin ziyaret edilme sıklığına göre belirlenmiştir. Şekil 3.10 B'de 71 adımdan oluşan yürüyüş 314 bit ile gösterilmiştir. Rasgele yürüyüşe 1111100 nolu düğümden başlanılmıştır. İkinci ziyaret edilen düğüm 1100 nolu düğümdür. Şekil 3.10 B'de ziyaret edilen düğümlerin sırası verilmiştir. Şekil 3.10 C'de ağın iki düzey tanımlaması ile hem düğümler hem de topluluklar gösterilmiştir. Topluluklar şu mantığa göre ortaya çıkarılmıştır: Rasgele yürüyüş sırasında topluluklar içinde harcanan süre uzundur. Buna göre rasgele yürüyüş esnasında uzun süre harcanan düğüm grupları birer topluluktur.

3.5. Bazı Algoritmaların Performanslarının Karşılaştırılması

Lancichinetti ve Fortunato (LF) [48] bu çalışmalarında popüler topluluk keşfi algoritmalarını karşılaştırmıştır. Doğruluk ölçümü için yapılan çalışmaların birçoğunda küçük ve topluluk yapıları basit olan ağlar kullanılmıştır.

Böyle ağlarda algoritmaların doğruluklarını iyi bir şekilde ölçmek güçtür. Bu çalışma bu alanda yapılan testlerin en kapsamlı olanlarından biridir. Testler yapay sosyal ağlarda [14,23] yapılmıştır. Kullanılan yapay sosyal ağlar düğümlerin dereceleri ve toplulukların büyüklükleri bakımından heterojen bir dağılım göstermektedirler. Bu sayede önerilen algoritmaların doğrulukları tutarlı bir şekilde ölçülebilmiştir. Topluluk keşfi için önerilen algoritmaların hangisinin daha iyi olduğu konusunda bir netliğin olmadığı belirtilmiş, bunun da topluluk için herkesçe kabul edilen bir tanımın ortaya konmamasından kaynaklandığı vurgulanmıştır. Literatürde en yaygın olarak kabul edilen tanımın yerleştirilmiş ℓ bölmelendirme modeli (*planted ℓ- partition model)* [49] olduğundan bahsedilmiştir. Bu model toplulukları gösteren basit bir ağ modelidir. Bu modelde "plants", düğüm gruplarıdır. Her bir düğümün grup içi ve grup dışı bağlantılarının olasılığı



Şekil 3.10 Rasgele yürüyüş ve kodlama ile topluluklar ortaya çıkarılıyor

p_{in} ve p_{out} şeklindedir. $p_{in} > p_{out}$ olduğu gruplar topluluk, $p_{in} \leq p_{out}$ olduğunda ise ağ rasgele bir ağdır. Girvan ve Newman [14] bu modele göre yapay sosyal ağ üretmişlerdir. Bu yapay sosyal ağ 128 düğüm ve 4 topluluktan oluşmaktadır, her düğümün derecesi 16'dır. Bu ağ topluluk keşfi yapan algoritmaların performans testi için yaygın bir şekilde kullanılmıştır. Ancak basit bir yapıda olduğundan bu ağda iyi sonuç veren bir algoritmanın iyi olduğu tartışılır. LF [48] her düğümün derecesinin ve toplulukların büyüklüklerinin eşit olmasından dolayı bu yapay sosyal ağın gerçek ağlara pek benzemediğini, gerçek ağların heterojen bir yapıda olduğunu belirtmişlerdir. LF gerçek ağlara daha yakın yapay sosyal ağ modeli üretmişlerdir. Bu yapay sosyal ağ modelinde düğümlerin dereceleri ve toplulukların

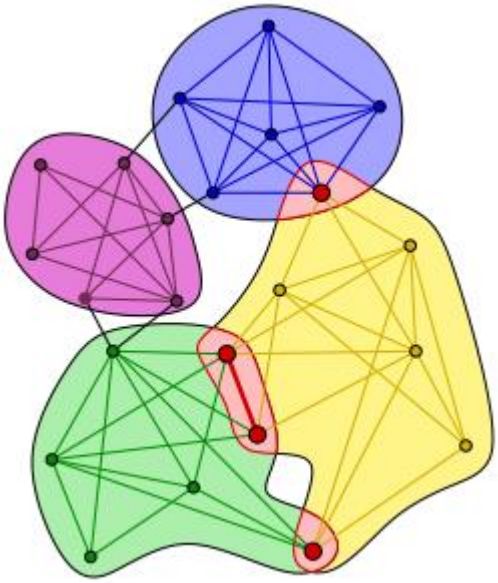
sayıları ve büyüklüklükleri heterojen dağılım göstermektedir. Kesişen topluluklar içeren ağlar üretilebilmekte, düğümlerin ait olacağı topluluk sayısı belirtilebilmektedir. Bağlantıların yönlü/yönsüz veya ağırlıklı/ağırlıksız olduğu ağlar da üretilebilmektedir. Bu özellikleriyle LF'nin önerdikleri bu yapay sosyal ağ modeli algoritmaları karşılaştırmak için oldukça iyi bir modeldir.

Ölçüt birimi olarak NMI [24] kullanılmıştır. İstenilen büyüklükte ve topluluk yapısında ağlar üretilmiş algoritmaların bu ağlarda bulduğu topluluklar önceden bilinen topluluklarla NMI kullanılarak karşılaştırılmıştır.

Bu çalışmada test edilen Girvan ve Newman (GN) [8], Clauset vd. [27], Blondel vd. [29], Radicchi vd. [11], Rosvall ve Bergstrom (Infomod) [46], Rosvall ve Bergstrom (Infomap) [47] algoritmalarından 3. bölümde bahsedilmiştir. Bahsedilmeyen algoritmalara kısaca değinilecektir:

- **Benzetilmiş tavlama yaklaşımı ile kapsamlı bir modülerlik optimizasyonu (Exhaustive modularity optimization via simulated annealing, Simm. ann.)** [50,40]**:** *Modülerliği* optimize eden bir yaklaşım önerilmiştir. Bu yöntemin amacı maksimum *modülerliği* yakalamaktır. Kapsamlı bir optimizasyon yapıldığı için sonuçların doğruluğu yüksektir. Ancak hesaplama zamanının yüksek olması en büyük dezavantajıdır. Ayrıca bazı optimizasyon parametrelerine ihtiyaç duymaktadır.
- **Cfinder** [51]**:** Bu algoritma toplulukları lokal düzeyde arama yaparak bulur. Kesişen toplulukları da bulabilmektedir. Topluluk olabilecek maksimum $k\text{-}clique$ grupların birleşimidir. $k\text{-}clique$, k tane düğümden oluşan ve her bir düğümün bu gruptaki diğer düğümlerle bağlantısının olduğu gruptur. Topluluk komşu $k\text{-}clique$'lerden oluşur. İki $k\text{-}clique$'nun komşu olması için k-1 tane düğümün ortak olması lazımdır. Bu algoritmanın tüm $k\text{-}clique$'leri bulması oldukça zaman alıcıdır, bu yüzden çok büyük ağlarda hesaplama zamanı yüksektir. Şekil 3.11'de [51] $k = 4$ değeri için $k\text{-}clique$ toplulukların gösterildiği örnek bir ağ gösterilmiştir.
- **Markov küme algortiması (MCL)** [52]**:** Toplulukları bulmak için ağ üzerinde rasgele yürüyüş yaklaşımı kullanılmıştır. Bu yaklaşıma göre ağda bir düğümden başlanılarak rasgele bir yürüyüş yapıldığında bu yürüyüşün büyük çoğunluğunun topluluk içinde olma ihtimali yüksektir. Ağın komşuluk matrisinden transfer matrisi elde edilir. Bu matris ağ üzerindeki rasgele yürüyüşün olasılığını gösterir. Transfer

matrisinde her kolonun toplamı bire eşittir. Bu transfer matrisi üzerinde *genişletme* (*expansion)* ve *artırma* (*inflation)* işlemleri uygulanmıştır. *Genişletme* işleminde matrisin tamsayı olan bir parametre ile kuvveti alınır, *artırma* işleminde de α parametresi ile matrisin her elemanının kuvveti alınır. *Genişletme* işlemi sonucunda yoğun bölgeler genişler, *artırma* işlemi sonucunda seyrek olan kısımlar azalır ve rasgele yürüyüş esnasında yürüyüşün topluluk içinde kalma ihtimali artar. Bu metotla elde edilen sonuçlarda α parametresinin değeri belirleyicidir. Bu iki işlemin tekrarlı bir şekilde uygulanmasıyla matriste birbirine bağlı olmayan eleman grupları şeklinde topluluklar ortaya çıkmaktadır.



Şekil 3.11 $k = 4$ değeri için $k\text{-}clique$ toplulukların gösterildiği örnek bir ağ

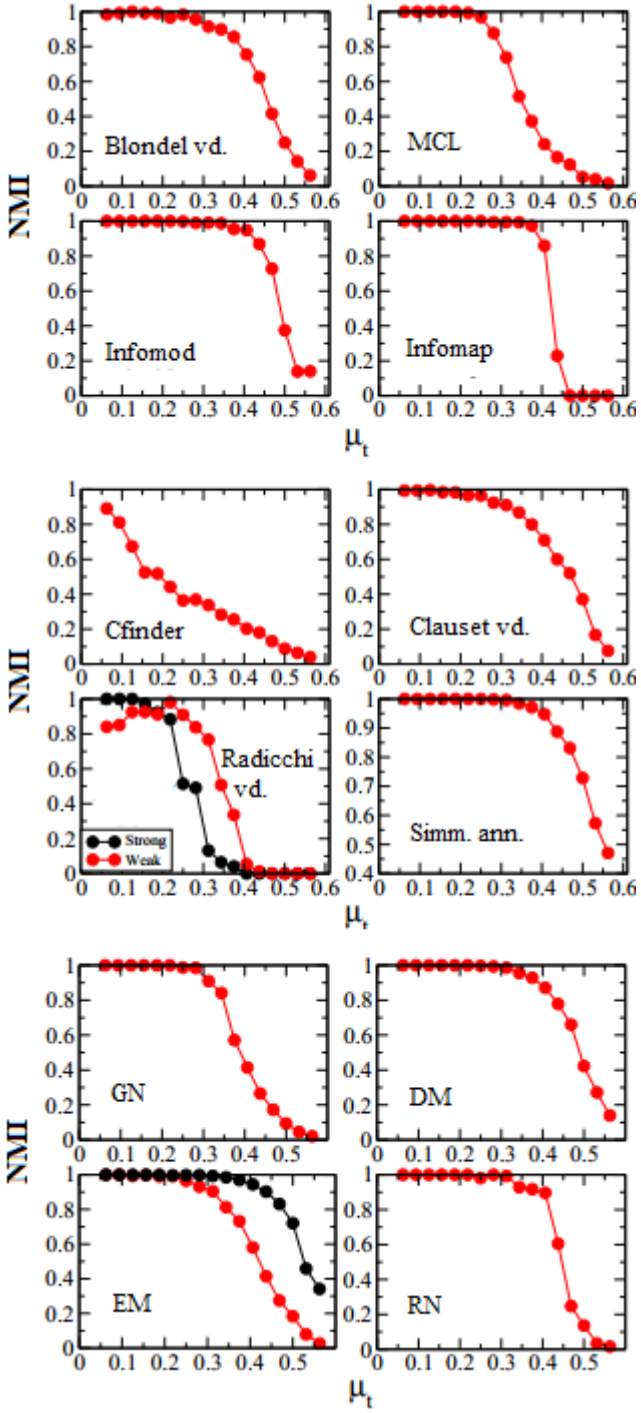
- **Spektral bir algoritma (DM)** [53]**:** Aynı topluluktaki düğümlerin özvektör değerlerinin benzer olması esasına dayanmaktadır. Laplacian matris ile özvektör değerleri gösterilmiştir. g sayısı kadar özvektör kullanılmış, ağdaki her düğüm g boyutlu öklid uzayında bir nokta olarak ele alınmıştır. Klasik hiyerarşik kümeleme kullanılarak noktalar sınıflandırılmıştır. Bulunan gruplardan modülerlik değeri yüksek olan çözüm olarak sunulmuştur.
- **Maksimum beklenti algoritması (Expectation-maximization algorithm, EM)** [54]**:** Düğümlerin kendi grupları içindeki düğümlerle bağlantı içerme olasılığına dayanmaktadır. Bu algoritma ağdaki düğümleri belli sayıda sınıflara bölmektedir.

Daha sonra maksimum beklenti algoritması ile bağlantı açısından birbirine benzer örüntüler içeren düğüm gruplarını topluluk olarak bulmaktadır. Başlangıç olarak düğüm grupları c sayıda sınıflara ayrılmaktadır. Algoritmanın c parametresine ihtiyaç duyması bir dezavantajıdır.

- **Potts model (RN)** [55]**:** Bu çalışmada kullanılan modelde aynı ağ için oluşturulan farklı kopyalar arasındaki benzerliklerden yararlanılarak topluluklar keşfedilmektedir. Kullanılan modelde döngüler vardır. Ağda bulunan her bir döngü ancak kendi topluluğu içindeki döngülerle etkileşim içinde bulunabilir.

Yukarıda bahsedilen algoritmaların GN yapay sosyal ağında karşılaştırma sonuçları Şekil 3.12'de [48] gösterilmiştir. μ_t, düğümlerin topluluk dışı bağlantı sayısının düğümün toplam bağlantı sayısına oranıdır. μ_t değerinin düşük olması düğümlerin bağlantıların çoğunun topluluk içinde olduğu anlamına gelir.

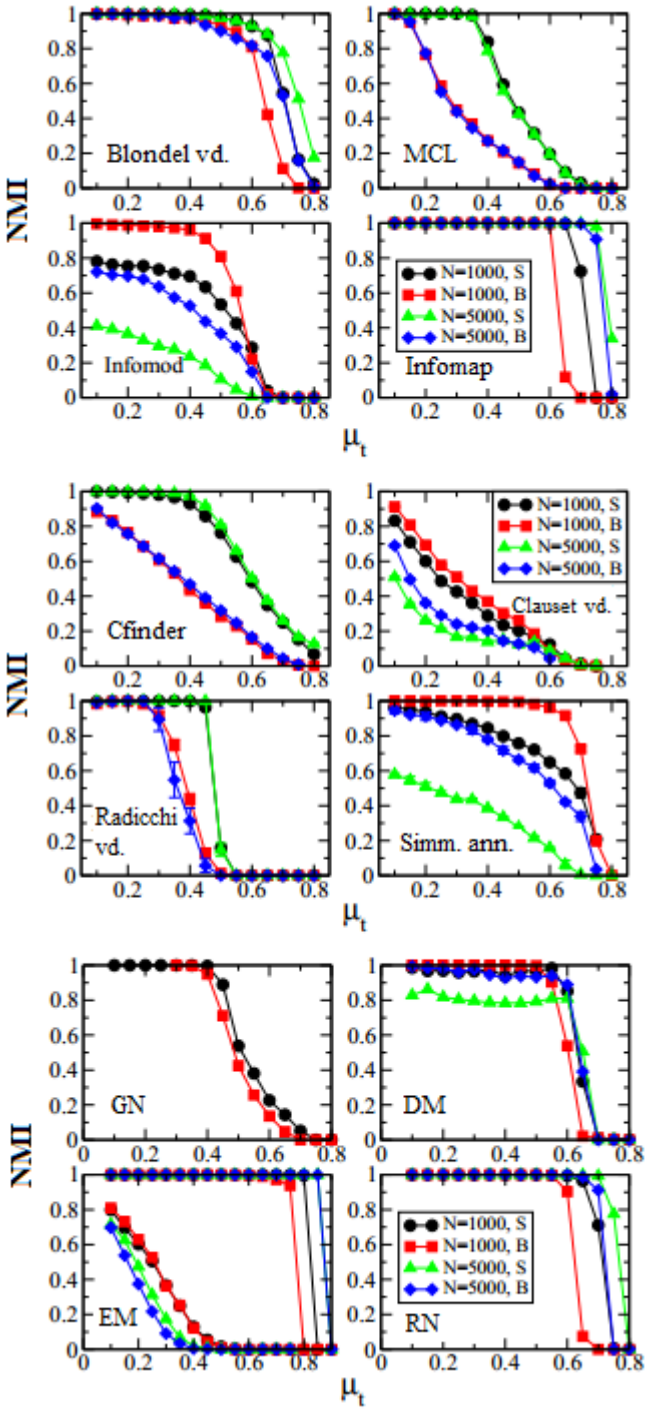
Cfinder, μ_t sıfır civarlarında olduğunda diğerlerine göre daha başarısız olmuştur. Aslında μ_t'nin küçük değerde olması ağdaki toplulukların sınırlarının daha net olması demektir, zaten diğer metotların hepsi bu aralıkta maksimum değer olan 1'e ulaşmıştır. Radicchi de diğerlerine göre μ_t'nin küçük değerlerinde başarısız olsa da Cfinder'a göre daha başarılıdır. MCL, Radicchi'den daha başarılı olurken modülerlik ölçüt birimini referans alan yaklaşımların (Simm. ann., Clauset et al, Blondel et al) gerisinde kalmıştır. Infomap en iyi performansı göstermiştir, μ_t'nin 0.4 değerine kadar toplulukları tam bulmuştur.



Şekil 3.12 Algoritmaların GN ağında karşılaştırma sonuçları

Diğer testler Lancichinetti ile Fortunato'nun [23] yapay sosyal ağında (LFR) yapılmıştır. Lancichinetti ve Fortunato'nun yapay sosyal ağı gerçek ağlara daha yakın ağlar oluşturmaktadır. Bu teste 1000 ve 5000 düğümden oluşan ağlar oluşturulmuş, düğümlerin ortalama dereceleri 20, maksimum dereceleri 50, düğüm derecelerinin dağılımı -2, toplulukların büyüklüklerinin dağılımı -1 olarak belirlenmiştir.İki farklı topluluk

büyüklükleri belirlenmiş, küçük topluluklar (S) 10 ile 50 arasında düğümden, büyükler (B) 20 ile 100 düğümden oluşturulmuşlardır. Sonuçlar Şekil 3.13'te [48] gösterilmiştir.

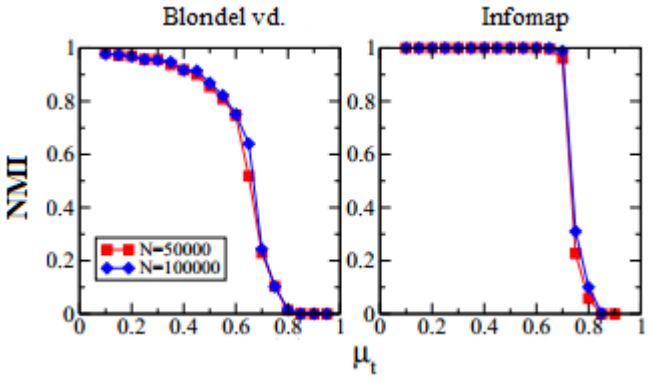


Şekil 3.13 LFR yapay sosyal ağında karşılaştırma sonuçları

Siyah noktalar 1000 düğümden oluşan ve küçük topluluklara sahip ağları, kırmızı noktalar 1000 düğümden oluşan ve büyük topluluklara sahip ağları, yeşil noktalar 5000 düğümden oluşan ve küçük topluluklara sahip ağları, mavi noktalar 5000 düğümden oluşan ve büyük topluluklara sahip ağları göstermektedir. GN'ın hesaplama zamanı uzun olduğu için sadece 1000 düğümlü ağların sonucu gösterilmiştir. Şekil 3.13'teki [48] sonuçlardan da görüldüğü gibi metotların performanslarını ayırt etmede LFR yapay sosyal ağı, GN'nin yapay sosyal ağından daha belirleyicidir

Modülerlik tabanlı metotlardan Blondel vd. hariç diğerleri için büyük ağ küçük topluluk karşılaştırmasında düşüş görülmüştür. Cfinder, MCL ve Radicch vd.'nde topluluk büyüklüklerindeki değişiklik performans üzerinde belirleyici bir etki göstermemiştir. DM ciddi bir performans göstermiştir. Infomap ve RN en iyi performansı göstermişler, μ_t için 0.5 değeri civarına kadar toplulukları tam keşfetmişlerdir. Blondel vd. de bunlara çok yakın bir performans göstermişlerdir. Infomap ile Blondel metotları hızlı oldukları için bunların performansı daha büyük ağlarda 50000 ile 100000 düğümlü ağlarda test edilmiştir. Sonuçlar Şekil 3.14'te [48] görülmektedir. Topluluk keşfi için literatürdeki çalışmalar testlerini küçük ağlarda yapmışlardır, bunların büyük ağlarda göstereceği performans önemlidir.

Bu testte topluluk büyüklükleri 20 ile 1000 düğüm arasında değişmektedir. Bu aralıkla toplulukların büyüklük olarak heterojen yapıda olması sağlanmaktadır. Düğümlerin maksimum derecesi 200'dür. Blondel vd. metodunun bir önceki teste göre performansının azaldığı, Infomap'ın ise büyük ağlarda da tutarlı bir sonuç verdiği görülmektedir.



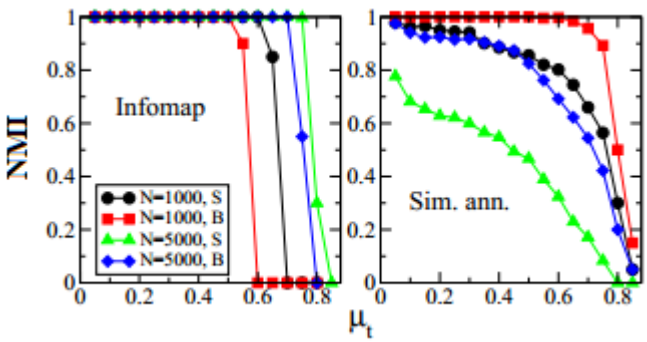
Şekil 3.14 Büyük ağlarda Blondel ve Infomap metotlarının sonuçları

Şu ana kadar yapılan testlerde ağlar yönsüz, ağırlıksız ve ayrıktı. Bu özellikler yapay

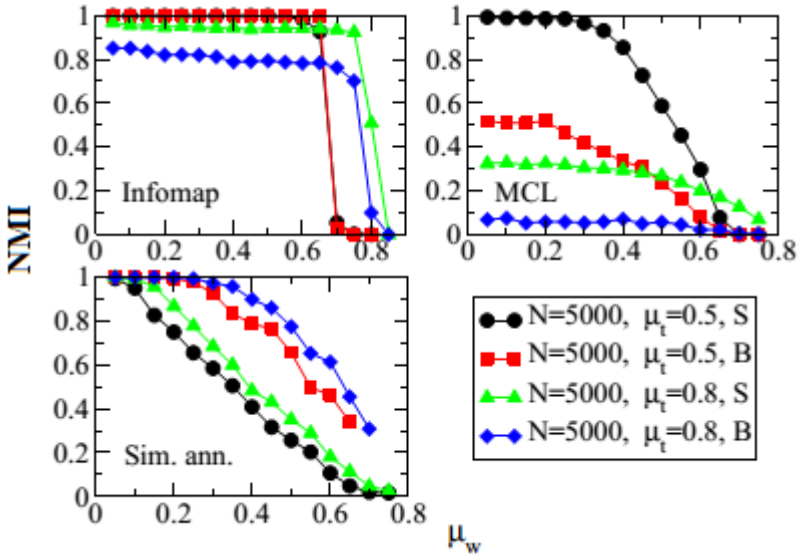
sosyal ağlara ayrı ayrı eklenip bunlar da test edilmiştir. İlk olarak yönlü ve ağırlıksız yapay ağlarda testler yapılmıştır. Ele alınan metotların hepsi yönlü ağlar için çalışmamaktadır. Yönlü ağlar için çalışan metotlar; Clauset vd., Simm. ann., Cfinder, Infomap ve EM'dir. Yönlü ve ağırlıksız ağlar için Infomap ve Simm. ann. algoritmaları test edilmiştir. Şekil 3.15'deki [48] sonuçlardan da görüldüğü gibi Infomap daha iyi sonuç vermiştir.

Diğer bir test yönsüz ve ağırlıklı ağlarda yapılmıştır. Bu ağ, bilgisayar ağlarına benzemektedir. Düğümler bilgisayarları, ağırlıklı bağlantılar da bant genişliğini temsil edebilir. Dolayısıyla ağdaki topolojileri ve bant genişlikleri en verimli şekilde kullanabilmek için önerilecek metodun ağırlıklı ağları analiz edebilecek bir metot olması gerekir. Bu testte μ_w diye bir parametre de kullanılmıştır, bu parametre μ_t'nin ağırlıklı versiyonudur.

Testte μ_t'ye sabit değerler verilmiş, μ_w'ya ise farklı değerler verilmiştir. Amaç ağırlığın sonuç üzerindeki etkisini incelemektir. Ağırlıkların ağ üzerindeki dağılımı bir parametre ile belirlenmişt, değeri 1.5 olarak verilmiştir. Ağırlıklı ağları inceleyebilen metotlar; MCL, Infomap ve Sim. ann.'dir. Sonuçlar Şekil 3.16'da [48] görülmektedir. Test ağları 5000 düğümden oluşmaktadır. Infomap yine iyi bir performans göstermiştir. Sadece toplulukların büyük ve μ_t değerinin yüksek olduğu durumda performans biraz kötüye gitmiştir. MCL sadece μ_t'nin 0.5 ve toplulukların küçük olduğu ağlarda iyi sonuç vermiş diğer durumlarda kötü sonuç vermiştir. Sim. ann. ise topluluk büyüklüklerine göre farklı sonuçlar vermiştir.



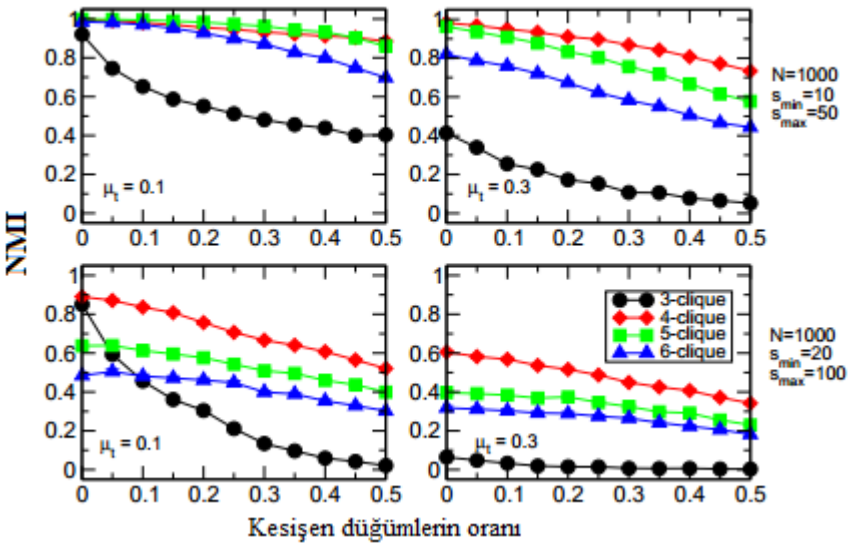
Şekil 3.15 Yönlü ve ağırlıksız ağda Infomap ve Sim. ann. karşılaştırması



Şekil 3.16 Ağırlıklı ağda bazı metotların performans sonuçları

Bir başka test yönsüz, ağırlıksız kesişen topluluklara sahip ağda yapılmıştır. Ele alınan metotlardan sadece Cfinder kesişen toplulukları bulabilmektedir. Şekil 3.17'de [48] 1000 düğümlü bir ağ için sonuçlar görülmektedir. X ekseni kesişen düğümlerin topluluklar arası oranıdır. Cfinder'ın k parametresi için 3,4,5 ve 6 değerleri verilmiştir. k=3 testi en kötü sonucu vermiştir.

Tüm test denemeleri düşünüldüğünde en iyi sonucu Infomap metodu vermiştir. Metotların performansını ölçmede LFR yapay sosyal ağının GN yapay sosyal ağına göre daha belirleyici olduğu görülmüştür.



Şekil 3.17 Cfinder algoritmasının sonuçları

4. TOPLULUK KEŞFİ İÇİN ÖNERİLEN ÇOK AMAÇLI GENETİK ALGORİTMA YAKLAŞIMI

Bu bölümde sosyal ağlarda topluluk keşfi için önerilen çok amaçlı genetik algoritma [56] anlatılacaktır. Önce genetik algoritmalardan ve çok amaçlı optimizasyon yaklaşımlarından genel olarak bahsedilecek, daha sonra yöntemin detayları ve test sonuçları açıklanacaktır.

4.1. Yöntem

4.1.1. Genetik algoritmalar

Genetik Algoritma doğa kanunlarından esinlenerek oluşturulmuş bir yaklaşımdır. Güçlü bireylerin hayatta kalması, zayıfların ise yok olması esasına dayanır. Genetik algoritmaların temel ilkeleri ilk kez Michigan Üniversitesi'nde John Holland tarafından ortaya atılmıştır. Holland 1975 yılında yaptığı çalışmaları "Adaptation in Natural and Artificial Systems" adlı kitabında bir araya getirmiştir. İlk olarak Holland evrim yasalarını genetik algoritmalar içinde eniyileme problemleri için kullanmıştır. Genetik algoritmalar problemlere tek bir çözüm üretmek yerine farklı çözümlerden oluşan bir çözüm kümesi üretir. Böylelikle, arama uzayında aynı anda birçok nokta değerlendirilmekte ve sonuçta bütünsel çözüme ulaşma olasılığı yükselmektedir [57].

Genetik algoritmada bireylerden ya da kromozomlardan oluşan bir popülasyon vardır. Bireyler genlerden oluşur. Bireyin gen yapısının daha doğrusu modelinin nasıl olacağını problemin yapısı belirler. Buna kodlama da denilir. Bir problemi genetik algoritma ile çözerken probleme uygun bir kodlama oldukça önemlidir.

Popülasyon ilk olarak arama uzayında rastgele bireyler oluşturulur. Problemin yapısına göre arama uzayını küçültmek için tamamen rastgele bireylerden ziyade belli kısıtlamalar dâhilinde rastgele bireyler oluşturulabilir. Böylelikle ilk popülasyon oluşturulurken olası kötü bireyler baştan elenmiş olur.

Popülasyondaki bireylerden hangilerinin çözüm olacağına uygunluk fonksiyonu ile karar verilir. Popülasyon oluşturulduktan sonra tüm bireylerin uygunluk değerleri

hesaplanır. Daha sonra bu bireylerden çaprazlama ve mutasyon yöntemleri ile yeni bireyler üretilir. Amaç daha iyi bireylere ulaşmaktır.

4.1.2. Çok Amaçlı Optimizasyon Yaklaşımları

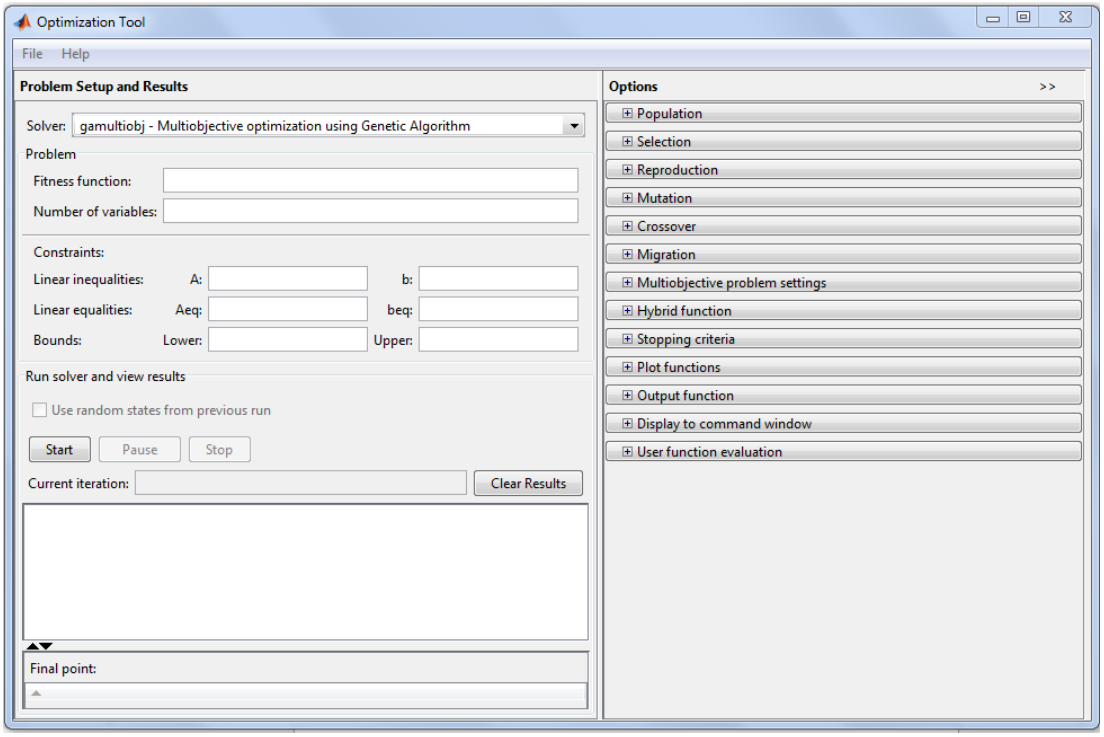
Bazı problemleri sadece bir uygunluk fonksiyonu ile tanımlamak güçtür. Böyle problemleri birden çok uygunluk fonksiyonu ile tanımlamak daha doğru bir yaklaşım olur. Topluluk keşfinde de çok amaçlı durumlar söz konusudur. Örneğin bir ağda toplulukları belirlemede topluluk içi bağlantıların yoğun ve topluluk dışı bağlantıların zayıf olması gibi iki amaç belirlenebilir. Yani ağda bir grup düğüme topluluk denebilmesi için hem düğümlerin grup içi bağlantıların çokluğuna hem de düğümlerin gruplar arası bağlantılarının azlığına dikkat edilmesi gerekir.

Çok amaçlı yaklaşım bize bir çözümden ziyade pareto çözüm kümesi diye adlandırılan bir çözüm kümesi sunmaktadır. Çözüm kümesi içerisinde farklı sayıda topluluk içeren çözümler bulunabilir. Hedef en iyi puana sahip bireyleri, arama uzayının oluşturduğu bireyler arasından bulmaktır. Örneğin iki uygunluk fonksiyonumuzun olduğunu varsayalım. Popülasyonda her bir birey için iki sonuç olacaktır. Bu iki sonuç için birbirlerine galip gelemeyenler çözüm kümesini oluştururlar. Bir bireyin birinci sonucu diğer bireyden üstün olabilir, diğer bireyin de ikinci sonucu öteki bireyden üstün olabilir, ya da tam tersi. Bu durumda bireyler birbirlerine galip gelememişlerdir, her iki birey de çözüm kümesine dâhil edilirler.

4.1.3. Çok Amaçlı Genetik Algoritma için Matlab Kullanımı

Çok Amaçlı Genetik Algoritma için Matlab'ın optimizasyon aracı kullanıldı. Kullanılan Matlab'ın versiyonu 7.11.0.584 (R2010b), optimiazsyon aracının vervisyonu 5.1'dir. Bu bölümde Matlab'ın optimizasyon aracına değinilecektir. Yararlandığımız kaynak [58] Matlab'ın genetik algoritmalar ve direkt arama aracının (Genetik Algorithm and Direct Search Toolbax) kullanım klavuzudur. Matlab açıldığında komut satırına *gatool* komutu verildiğinde optimizasyon aracı açılmaktadır. Karşımıza gelen ekran Şekil 4.1'te gösterilmektedir.

Ekranın sol kısmında uygunluk fonksiyonu, değişken sayısı, kısıtlamalar belirlenmektedir. Uygunluk fonksiyonu bir vektör döndürmektedir.



Şekil 4.1 Matlab optimizasyon aracı

Bu vektörün elemanları problem için belirlenen uygunluk fonksiyonlarının hesaplanan değerleridir. Sağ tarafta ise optimizasyon konfigürasyonu ile ilgili kısımlar bulunmaktadır.

Popülasyon bölümünde popülasyondaki verinin türü, popülasyonun büyüklüğü, popülasyonun oluşturulma yöntemi, başlangıç skoru ve başlangıç aralığı belirlenmektedir.

Seçim kısmında yeni jenerasyon üretimi için kullanılacak bireylerin nasıl seçileceği belirlenir. Turnuva *(tournament)* ve özel diye iki seçenek vardır. Turnuva ile belirtilen sayı kadar birey popülasyon içinden rasgele seçilir ve seçilenler arasından uygunluk değeri en iyi olanlar yeni jenerasyon için kullanılır. Özel seçeneğinde ise tanımlanan özel bir yöntem ile seçim yapılır.

Çoğalma bölümünde yeni popülasyonun yüzde kaçının çaprazlama ile yüzde kaçının mutasyonla üretileceği belirlenmektedir. Çaprazlama için [0,1] aralığında bir değer verilir. Mutasyon oranı da bu değerin birden çıkarılması ile bulunur. Ayrıca mutasyon oranı verilmez.

Mutasyon bölümünde uygulanacak mutasyon yöntemi seçilir. Seçeneklerde kısıtlı bağımlılık, gaussian, tek biçimli, adaptif uygulanabilir ve özel yöntemleri vardır. Kısıtlı bağımlılık seçilmişse ve bir kısıt belirtilmişse gaussian yöntemi, belirtilmemişse adaptif uygulanabilir yöntem uygulanır. Gaussian'da her bir gen için rasgele sayı oluşturulur. Bu rasgele sayı Gauss dağılımı ile elde edilir. Bu dağılımın standart sapması iki parametre ile

kontrol edilir. Biri ölçek, diğeri küçültme parametresidir. Ölçek parametresi ile ilk jenerasyonun üretimi için standart sapma belirlenir. Küçültme parametresi ile de jenerasyon üretimi devam ederken standart sapmanın nasıl küçüleceği kontrol edilir. Eğer küçültme için 0 verilmişse standart sapma sabit kalır, 1 verilmişse standart sapma doğrusal bir şekilde son jenerasyonda sıfıra ulaşacak şekilde azalır. Tek biçimli mutasyonda bireyin genlerinden belli bir kısmı verilen parametreye göre mutasyon için seçilir. Her genin seçilme olasılığı eşittir. Seçilen genlerin değerleri genin alabileceği değerler aralığında rasgele değiştirilir. Adaptif uygulanabilir yönteminde kısıtlara ve sınırlara göre mutasyon yapılır. Özel seçeneğinde belirlenen özel bir yöntemle mutasyon yapılır.

Çaprazlama için seçilebilecek yöntemler şunlardır: Dağınık, tek nokta, iki nokta, ortalama, sezgisel, aritmetik ve özel. Dağınık çaprazlamada birey uzunluğunda 0 ve 1'lerden oluşan rasgele bir vektör oluşturulur. Çaprazlama için iki birey alınmakta ve vektör üzerinde 1 değerinin bulunduğu pozisyonlar için 1. bireyin genleri, 0 değerinin bulunduğu pozisyonlar için 2. bireyin genleri alınıp yeni bir birey oluşturulur. Tek nokta yönteminde n tane genden oluşan kromozom için 1 ile n arasında rasgele bir sayı oluşturulur. n. gene kadar olan kısım 1. Kromozomdan n. genden sona kadar olan kısım ikinci genden alınarak yeni bir birey oluşturulur. İki nokta çaprazlamada 1 ile n arasında x ve y diye rasgele iki sayı üretilir. 1. genden x. gene kadar olan kısım 1. kromozomdan x. genle y. gen arasında olan kısım 2. kromozomdan y. genden n. gene kadar olan kısım yine 1. kromozomdan alınarak yeni bir birey oluşturulur. Ortalama yöntemde yeni bireyler, bireylerin rasgele ağırlıklandırılmasından elde edilir. Oran diye bir parametre kullanılır. Bu parametre [0,1] aralığında ise yeni bireyler bir küpün içinde oluşturulur. Bireyler küpün zıt köşelerine koyulur. Eğer oran değeri birden büyükse yeni bireyler küpün dışında oluşturulur. Sezgisel çaprazlamada iki birey bir çizgiye koyulur. Üretilecek yeni birey bu çizgideki pozisyona göre üretilir. Pozisyon iki bireyin uygunluk değerine göre belirlenir. Üretilecek yeni birey uygunluk değeri yüksek olan bireye daha yakın uygunluk değeri düşük olan bireye daha uzak olarak konumlandırılır. Aritmetik çaprazlama iki bireyin rasgele aritmetik bir ortalaması alınarak yapılır. Özel çaprazlamada belirlenen özel bir yöntem kullanılır.

Göç bölümünde alt popülasyonlar arasında iyi bireyler kötülerle yer değiştirilir. Bu bölümde üç parametre vardır: Yön, oran, aralık. Yön ile göçün hangi yönde olacağı belirlenir. İleri seçeneği seçildiğinde göç sonuncu alt popülasyona doğru yapılır. n. alt popülasyondan (n+1). alt popülasyona doğru göç gerçekleşir. İkisi seçilince göç iki yönde

de yapılır. Oran ile alt popülasyonların yüzde kaçının göç yapacağı belirlenir. Aralık parametresi ile de kaç jenerasyonda bir göçün gerçekleşeceği belirtilir.

Çok amaçlı problem ayarları bölümünde mesafe ölçüm fonksiyonu popülasyonun konsantrasyonunu ölçmek için kullanılır. Popülasyonun pareto sınır oranı en fit durumda olan popülasyonu biraz genişleterek çeşitliliği artırmaya yarar.

Hibrid fonksiyon bölümü genetik algoritma bittikten sonra başka bir minimizasyon yapılmasına imkân tanır.

Durma kriteri bölümünde nelerin optimizasyonu sonlandıracağı belirtilir. Sonlandırma kriteri için jenerasyon sayısı, zaman limiti, uyguluk fonksiyon limiti ve daha özel kriterler vardır. Jenerasyon sayısı ile optimizasyon belirtilen sayı kadar jenerasyon ürettikten sonra, zaman limiti ile belli bir zaman geçtikten sonra, uygunluk değeri ile belirtilen uygunluk değerine ulaşıldıktan sonra sonlanır.

Kullanıcı fonksiyonu değerlendirme bölümünde uygunluk fonksiyonunun nasıl değerlendirileceği belirlenir. Seçeneklerde seri, paralel ve vektörize vardır. Seri olunca popülasyondaki tüm bireylerin uygunluk değerleri tek tek hesaplanır. Paralel olunca hesaplama bir grup işlemci ile paralel bir şekilde yapılır. Vektörize olunca tüm popülasyonun uygunluk değerleri bir fonksiyonun çağrılması ile hesaplanır.

Kalan bölümler de adımların ve sonuçların görsel olarak nasıl gösterileceği ile ilgilidir.

4.1.4. Uygunluk Fonksiyonları

Uygunluk fonksiyonları için MOGA-Net'te [10] kullanılan fonksiyonlar kullanıldı. Bunlardan biri *topluluk skoru* (*community scrore)* [10] diğeri *topluluk uygunluğudur* [25]. *topluluk skoru* topluluk içindeki bağlantıları maksimize etmeye çalışırken *topluluk uygunluğu* topluluklar arası bağlantıları minimize etmeye çalışır. *Topluluk skoruna* bakacak olursak: Topluluklar $\{S_i, \dots, S_k\}$ şeklinde ifade edilmiştir. μ_i, i düğümünün kendi topluluğu içindeki bağlantı sayısının topluluğun düğüm sayısına oranıdır.

$$\mu i = \frac{1}{|S|} k_i^{in}(S) \tag{4.1}$$

$M(S)$, S topluluğu içindeki tüm düğümler için μi değerlerinin r. dereceden kuvvetlerinin S'in eleman sayısına oranıdır. Formülü şöyledir:

$$M(S) = \frac{\sum_{i \ni S}(\mu_i)^r}{|S|} \quad (4.2)$$

r. dereceden kuvvetin alınması topluluk içi bağlantı sayısı fazla olan düğümlerin ağırlığını artırırken topluluk içi bağlantı sayısı az olan düğümlerin ağırlığını azaltmaktadır. Böylece topluluk tanımına uygun olan düğüm grupları öncelenmiş olmaktadır. S topluluğu için *topluluk skoru* şöyle tanımlanmıştır:

$$score(S) = M(S) \times v_s \quad (4.3)$$

v_s, topluluk içindeki bağlantı sayısıdır. Tüm ağın *topluluk skoru* ise şöyledir:

$$CS = \sum_{i=1}^{k} score(S_i) \quad (4.4)$$

Topluluk uygunluğu formülü ise düğümlerin topluluk içi bağlantı sayılarının düğümlerin toplam bağlantı sayısına oranlarının toplamıdır. Formülü şöyledir:

$$P(S) = \sum_{i \in S} \frac{k_i^{in}(S)}{(k_i^{in}(S) + k_i^{out}(S))^{\alpha}} \quad (4.5)$$

α, ise toplulukların büyüklüğünü belirlemeye yarayan pozitif reel bir parametredir. $P(S)$ değeri maksimum değerine ulaştığında topluluklar arası bağlantılar da minimum olacaktır.

4.1.5. Topluluk Keşfi için Genetik Kodlama

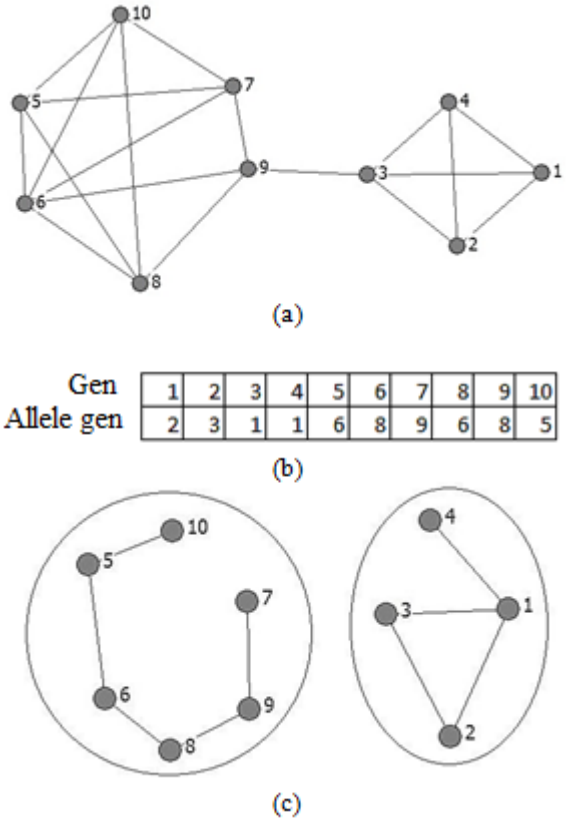
Genetik kodlama için Park ve Song tarafından önerilen odak tabanlı komşuluk gösterimi (LAR) [32] kullanıldı. LAR, topluluk keşfi için önerilen genetik algoritmalarda kullanılmıştır. Bu gösterimde her bir bireyin ya da kromozomun uzunluğu ağın düğüm sayısı kadardır. Kromozomun genleri ağın düğümleridir. Her genin bir allele geni vardır, bu allele gen de yine ağdaki bir düğüme karşılık gelmektedir. Genle allele gen, ağdaki bir bağlantıyı temsil etmektedir. Yani i geninin allele geni j geni ise, ağda i düğümü ile j düğümü arasında bir bağlantı var demektir. LAR gösteriminin çözümlenmesi ile ağdaki topluluklar ortaya çıkar. Çözümleme sonucu topluluk sayısı otomatik olarak ortaya çıkmaktadır. Şekil 4.2'de örnek bir ağ için LAR modeli gösterimi ve çözümle sonucu

oluşan topluluklar gösterilmektedir. Çözümlemede düğümlerin karşılarına allele gen olarak gelen düğümlerden ulaşılabilen tüm düğümler aynı topluluğa dahil olurlar. Örneğin Şekil 4.2'de 1. düğümün karşısına 2., 2. düğümün karşısına 3., 3. ve 4. düğümlerin karşılarına 1. düğüm gelmiştir. Bu şekilde ulaşılabilen düğümler aynı topluluğa dahil edilirler.

4.1.6. Birey Seçimi

MOGA-Net başlangıç popülasyonu oluştururken arama uzayını düğümün komşuları ile sınırlamıştır. Böylece gereksiz aram uzayından kaçınılmıştır. Şekil 4.2'deki örnek ağa bakacak olursak 4. düğüm için seçilecek düğüme 10. düğüm gelirse çözüm adayı kümemize kötü bir aday koymuş oluruz. Çünkü 4. ve 10. düğümlerin toplulukları farklıdır. Aynı topluluk içinde olan düğümler arasında bağlantıların yoğun olması beklenir. Bir düğümün dahil olacağı topluluk yine bu düğümün komşularından bir çoğunu barındırmalıdır. Ancak bu şekilde topluluk için yoğun bağlantılar elde edilir. O yüzden MOGA-Net'in başlangıç popülasyonu oluştururken birey seçimini düğümlerin komşuları ile sınırlaması arama uzayını daraltmakta ve iyi çözümler bulunma ihtimalini artırmaktadır. Çalışmamızda bu seçim topluluk tanımına daha uygun olacak şekilde geliştirildi. Aynı topluluk içindeki düğümler arasında ortak komşuların fazla olma ihtimali yüksektir. Başlangıç popülasyonu oluştururken seçim hem düğümün komşuları ile sınırlandırıldı hem de ortak komşuları fazla olan düğümlerin seçilme olasılığı artırıldı. Böylece daha iyi çözümler bulmak için başlangıç koşulları güçlendirildi. Bu yaklaşım rulet seçim yöntemi ile gerçekleştirildi. Rulet seçim yöntemi rulet tekerleğinin simülasyonunu yapar. Rulet tekerleğinde dilimler vardır. Seçimin olasılığı dilimlerin büyüklüklerine bağlıdır. Büyük dilimlerin seçilme olasılığı daha yüksektir. Allele gen seçilirken düğümlerin ortak komşuluklarının ağırlığı dilimler olarak belirlendi. Böylece ortak komşuları fazla olan düğümlerin aynı toplulukta olma ihtimalleri artırıldı, aynı zamanda ortak komşuları az olan düğümlerin aynı toplulukta olma ihtimalleri azaltıldı.

Şekil 4.3'de Şekil 4.2'deki 9. düğüm için ortak komşuluklara göre rulet seçim grafiği gösterilmiştir. 9. düğümün komşuları 3, 6, 7 ve 8. düğümlerdir. 9. Düğümün 3, 6, 7 ve 8. düğümlerle olan ortak komşularının sayısı sırasıyla 0, 2, 1 ve 1'dir. 9. düğüm için allele gen belirlenirken 6. düğümün seçilme olasılığı %50, 7 ve 8. düğümlerin seçilme olasılığı %25'tir. Görüldüğü gibi 3. düğüm 9. düğümün komşusu olmasına rağmen seçime girememiştir. Zaten bu düğümlerin toplulukları farklıdır. Böylece gereksiz arama

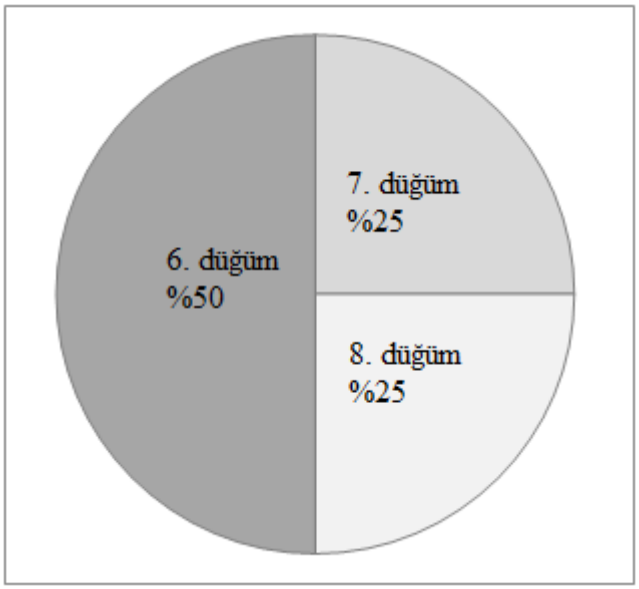


Şekil 4.2 a) 10 düğümden oluşan örnek bir ağ
b) LAR modeli c) Çözümleme sonucu

uzayından kaçınılarak daha iyi sonuçlar elde edildi. Bu seçim yaklaşımı hem başlangıç popülasyonu oluşturulurken hem de mutasyon yapılırken uygulandı.

4.1.7. Mutasyon

Mutasyon kromozomun genlerinde değişiklikler yapar. Bu adımda tek biçimli mutasyon kullanıldı. Tek biçimli mutasyonda her gen için 0 ve 1 aralığında rasgele bir sayı üretilir. Ve yine 0 ve 1 aralığında mutasyon parametresi kullanılır. Rasgele üretilen sayı mutasyon parametresinden büyük ise gen değiştirilir küçük ise bir değişiklik yapılmaz. Standart tek biçimli mutasyonda gen değişikliği rasgele yapılırken çalışmamızda bu değişiklik birey seçimi bölümünde bahsedilen seçim yaklaşımına göre yapıldı. Bu sayede değişikliğe uğrayan bireylerin daha iyi bir çözüm adayı olma ihtimali artırıldı.



Şekil 4.3 9. düğümün ortak komşuluklara göre rulet seçim grafiği

4.1.8. Çaprazlama

Çaprazlamada bireylerin birleşiminden yeni bireyler oluşturulur. Bunun için dağınık çaprazlama yöntemini kullanıldı. Dağınık çaprazlamaya bölüm 4.1.3'te değinilmişti. Başlangıç popülasyonu önerilen birey seçim yaklaşımı ile oluşturulduğundan dolayı çaprazlama sonucu oluşacak bireyler de çözüm için iyi bir potansiyele sahip olacaktır.

4.1.9. Parametreler

Matlab'ta tüm çözüm kümesi içinden baskın olmayan çözüm kümesini bulmak için Deb vd. tarafından önerilen genetik algoritmada baskın olmayan sıralama yöntemi (nondominated sorting genetic algorithm, NSGA-II) kullanılmaktadır. Matlab içerisinde çok amaçlı genetik algoritma için kullandığımız parametreler şöyledir: Popülasyon büyüklüğü 200, jenerasyon sayısı 30, çaprazlama oranı 0.4, mutasyon oranı 0.3, tournament sayısı 100'dür. Çaprazlama oranı yeni popülasyon üretilirken popülasyonun yüzde kaçının çaprazlama ile üretileceğini göstermektedir. Uygulamamızda her jenerasyonda 80 birey çaprazlama ile üretilmiştir. Kalan 120 birey de mutasyon ile üretilmiştir. Mutasyon oranı mutasyon yapılırken parametre olarak gelmekte üretilen rasgele sayı bu mutasyon oranından büyükse söz konusu gen değiştirilmektedir. Turnuva sayısı ise bir sonraki jenerasyon için kullanılacak bireylerin sayısını göstermektedir. Bu

sayı kadar birey popülasyon içinden rasgele seçilmekte, seçilen bu bireyler arasından uygunluk değeri en iyi olan bireyler yeni jenerasyonun üretimi için kullanılmaktadır.

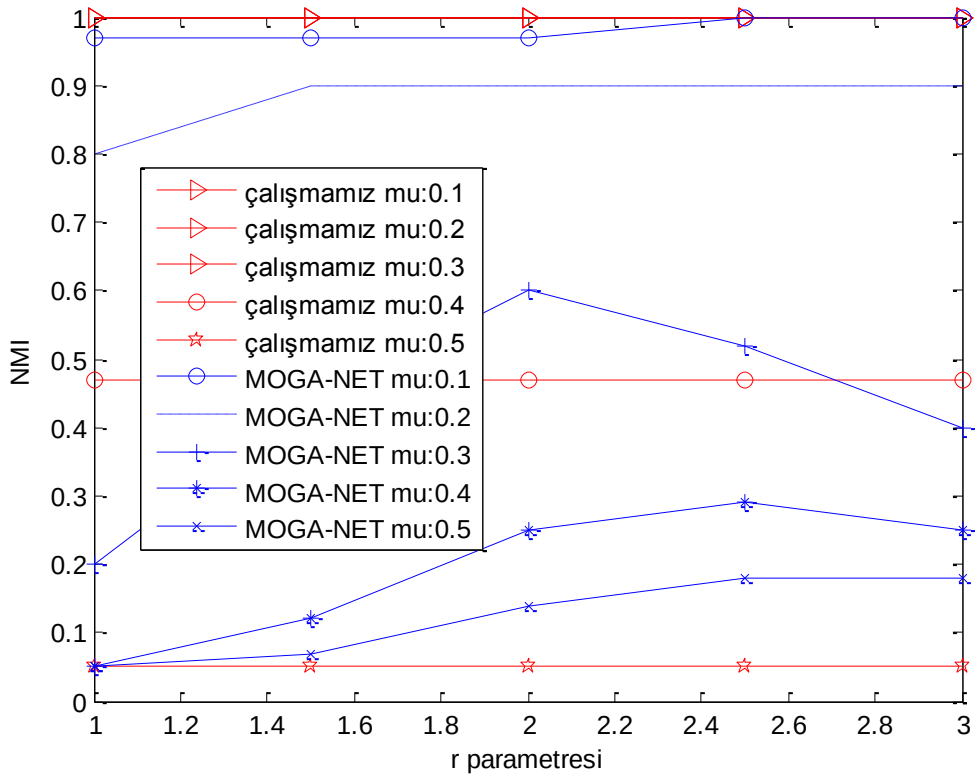
4.2. Sonuçlar

4.2.1. Yapay Sosyal Ağlar

Topluluk keşfi için yapılan çalışmaların doğruluğunu test etmek için yapay sosyal ağlar önerilmiştir. Topluluk yapıları bilinen yapay sosyal ağlar algoritmada çalıştırılıp algoritmanın bulduğu topluluklar karşılaştırılarak benzerlik oranına göre doğruluk derecesi belirlenmektedir. Benzerlik ölçüt birimi için NMI [24] oldukça kullanılan bir yöntemdir. Çalışmamızda Lancichinetti ve Fortunato'nun (LF) [23] önerdiği yapay sosyal ağlar ve ölçüt birimi olarak NMI kullanıldı. Gerçek ağlarda toplulukların ve düğümlerin yapıları heterojen bir dağılım göstermektedir. LF yapay sosyal ağları oldukça zengin parametrelere sahiptir. Böylece gerçek ağlara yakın ağlar üretilebilmektedir. LF yapay sosyal ağ oluştururken kullanılan parametreler şunlardır:

- **N** : düğüm sayısı.
- **k** : düğümlerin ortalama bağlantı sayısı.
- **maxk** : maksimum bağlantı sayısı.
- **mu** : karıştırma oranı, düğümlerin topluluk içi bağlantı sayısı ile topluluk dışı bağlantı sayısı.
- **-t1 :** düğümlerin bağlantılarının dağılımı için üs parametresi.
- **-t2 :** toplulukların büyüklüklerinin dağılımı için üs parametresi.
- **minc** : bir toplulukta olabilecek minimum düğüm sayısı.
- **maxc** : bir toplulukta olabilecek maksimum düğüm sayısı.
- **on** : kesişen düğümlerin sayısı.
- **om** : kesişen düğümlerin kaç tane toplulukla kesişeceği sayısı.
- **C** : ortalama kümeleme katsayısı.

Test için 128 düğümden, 4 topluluktan oluşan ve düğümlerinin ortalama dereceleri 16 olan ağlar üretildi. Karıştırma parametresi (mu) [0.1, 0.5] aralığında, *topluluk skoru* fonksiyonunda kullanılan r parametresi [1,3] aralığında verildi. mu parametresi 0.1 olan bir ağda bir düğümün bağlantılarının %90'ı topluluk içinde %10'u topluluk dışındadır.

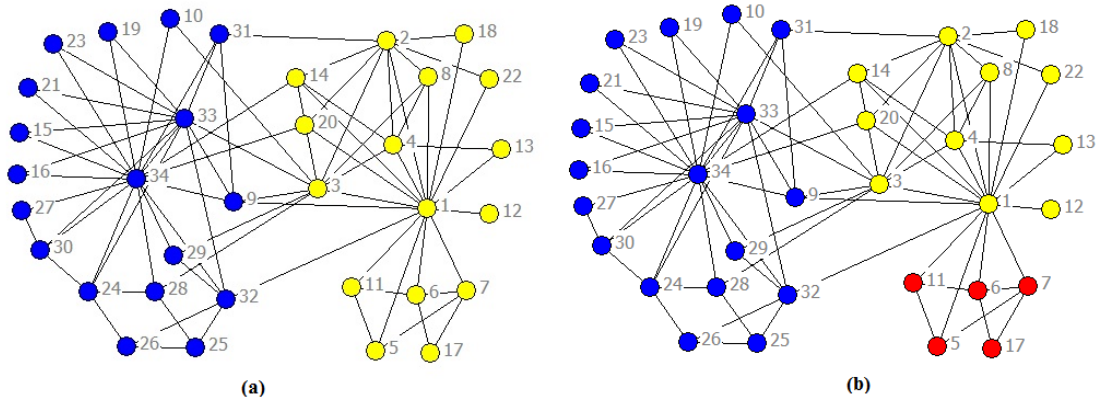


Şekil 4.4 MOGA-Net ve bizim çalışmamızın yapay sosyal ağlarda sonuçları

Elde edilen sonuçlar MOGA-Net algoritması ile karşılaştırıldı. Sonuçlar Şekil 4.4'te görülmektedir. mu parametresinin [0.1, 0.3] aralığında olduğu ağlarda en iyi sonuçlar elde edildi. Önerdiğimiz yöntemle bu ağlarda, önceden bilinen toplulukların aynısı elde edildi, NMI için 1 değerine ulaşıldı. MOGA-Net ise 1 değerine sadece mu'nun 0.1 ve r'nin 2.5'ten büyük olduğu durumda ulaşabilmiştir. Bu aralıkta MOGA-Net farklı r değerleri için farklı sonuçlar elde etmiştir. Yöntemimiz ile farklı r değerleri için en iyi sonuç elde edilmiştir. Önerdiğimiz yaklaşımla hem daha iyi sonuçlar elde edildi hem de parametre bağımlılığından kurtulmuş olundu. mu'nun 0.4 olduğu ağda da tüm r değerleri için MOGA-Net'ten daha iyi sonuç elde edildi. NMI için 0.5'e yakın bir değer elde edildi. MOGA-Net ise r'nin 2.5 olduğu durumda NMI için 0.3'e yaklaşabilmiştir. MOGA-Net sadece mu'nun 0.5 olduğu durumda çalışmamızdan daha iyi sonuç elde etmiştir.

4.2.2. Gerçek Ağlar

Önerdiğimiz algoritmayı gerçek ağlarda da test ettik. Test ettiğimiz ağlar Zachary'nin Karate kulübü, Bottlenose yunusları ve Amerikan kolej futbol takımı ağlarıdır. Karate kulübü, Zachary [21] tarafından oluşturulmuştur. 34 üyeden 78 tane bağlantıdan oluşan bir

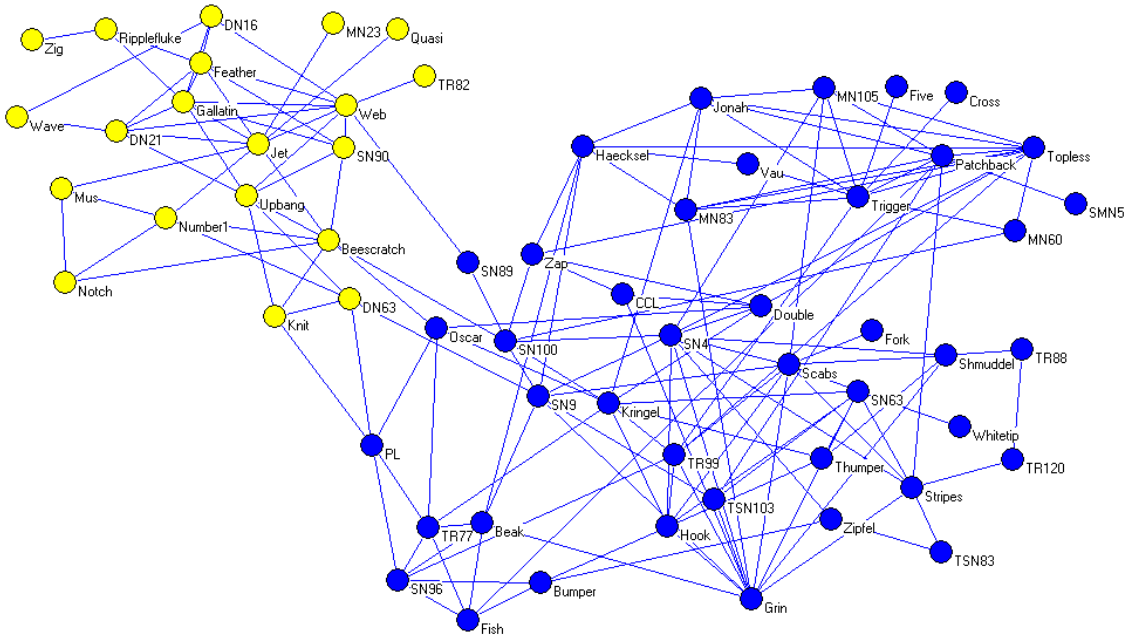


Şekil 4.5 Yaklaşımımızın Karate kulübü ağında bulduğu topluluklar

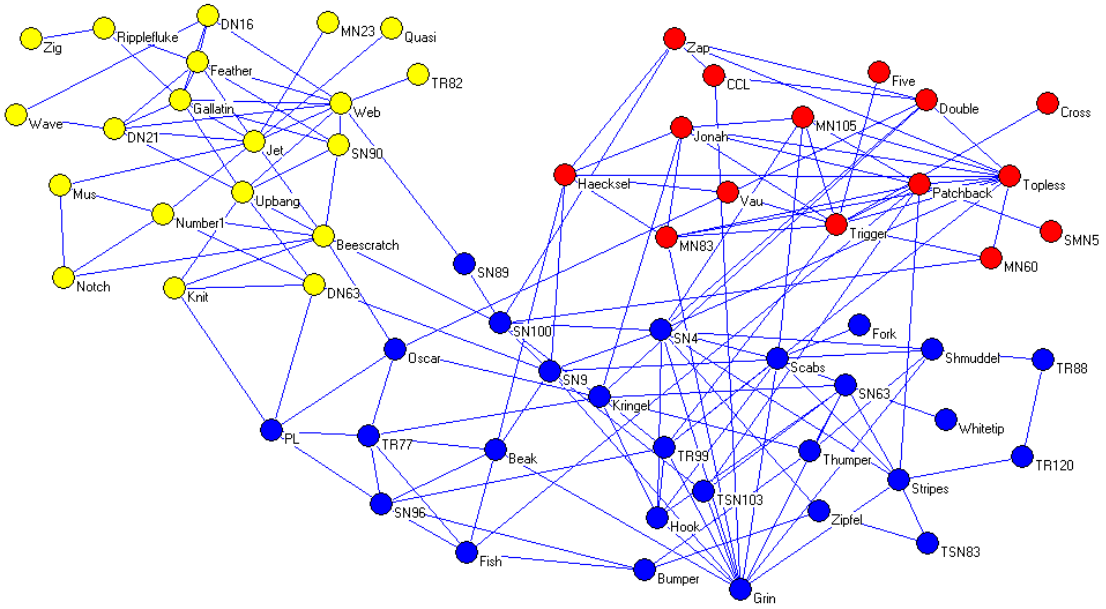
karate kulübündeki arkadaşlık ağıdır. İki yıllık bir süre içerisinde bu kulüpteki üyeler bazı anlaşmazlıklardan dolayı ikiye bölünmüşlerdir. Grupların sayıları birbirlerine yakındır. Karate kulübü ağı ve toplulukları Şekil 4.5'te görülmektedir. Şekil 4.5 a) ağın orijinal topluluğunu göstermektedir. Modülerlik değeri 0.3715'tir. Çalışmamız bu toplulukları bulmuştur. Şekil 4.5 b) çok amaçlı optimizasyon yaklaşımı sayesinde elde ettiğimiz farklı bir sonuçtur. Modülerlik değeri 0,3991'dir. Bulduğumuz bu sonucun modülerlik değeri orijinal toplulukların modülerlik değerinden büyüktür.

Bottlenose yunusları Yeni Zellanda'nın Doubtful denizyolunda yaşayan 62 tane yunustan oluşan bir sosyal ağdır. Lusseau [22] tarafından bu yunusların davranışları 7 yıl boyunca incelenmiştir. İncelemeler sonucunda bu ağda iki topluluğun ve 159 bağlantının olduğu görülmüştür. Şekil 4.6'da Bottlenose yunuslarının topluluk yapıları, Şekil 4.7'de ise çalışmamızın bu ağ için bulduğu topluluklar gösterilmiştir. Orijinal topluluklara ulaşılamamışsa da bulunan topluluklar topluluk tanımına uygundur, topluluk içi bağlantılar yoğun topluluk dışı bağlantılar zayıftır. Ayrıca orijinal ağın modülerlik değeri = 0.3735 iken Şekil 4.7 a)'daki ağın modülerlik değeri 0.4886, Şekil 4.7 a)'daki ağın modülerlik değeri 0.4802'dir. Orijinal ağın modülerlik değerinden daha iyi sonuçlar elde edilmiştir.

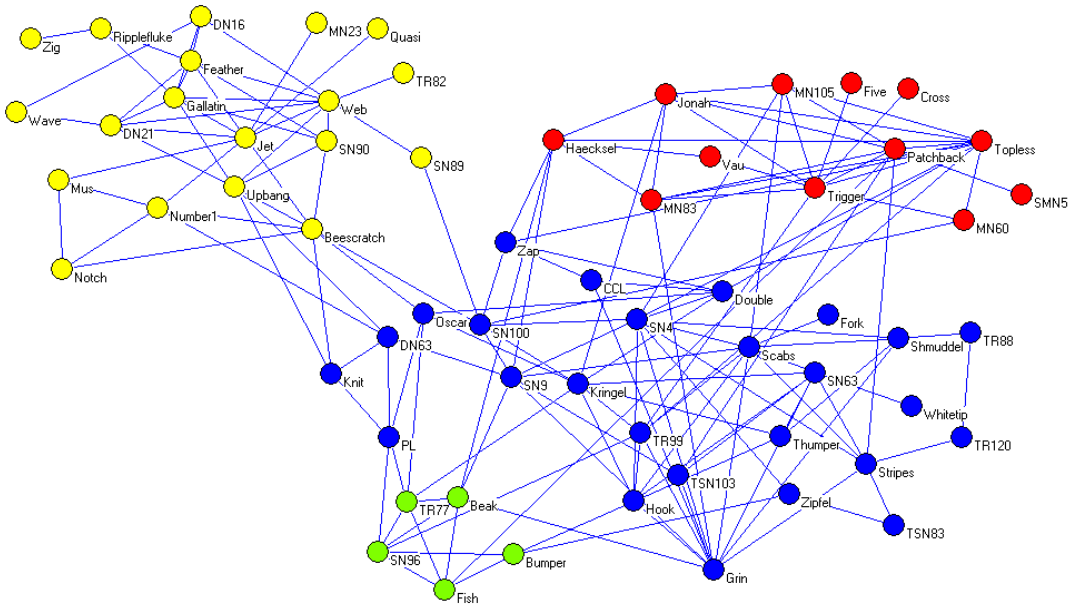
Amerikan kolej futbol takımı ağı 2000 yılında Amerikan kolejleri arasında yapılan futbol maçlarını göstermektedir. Girvan ve Newman tarafından oluşturulmuştur [59]. Düğümler takımları, bağlantılar takımlar arasında olan maçları göstermektedir. Takımlar gruplara bölünmüştür. Grup içi ortalama maç sayısı 7, grup dışı ortalama maç sayısı 4'tür. Şekil 4.8'de orijinal topluluklar, Şekil 4.9'da bizim bulduğumuz topluluklar gösterilmektedir. Bu ağ 115 düğümden, 616 bağlantıdan ve 12 gruptan oluşmaktadır. Orijinal ağın modülerlik değeri 0.597, bizim bulduğumuz çözümün modülerlik değeri 0,592'dir.



Şekil 4.6 Bottlenose yunusları ağının orijinal topluluk yapısı

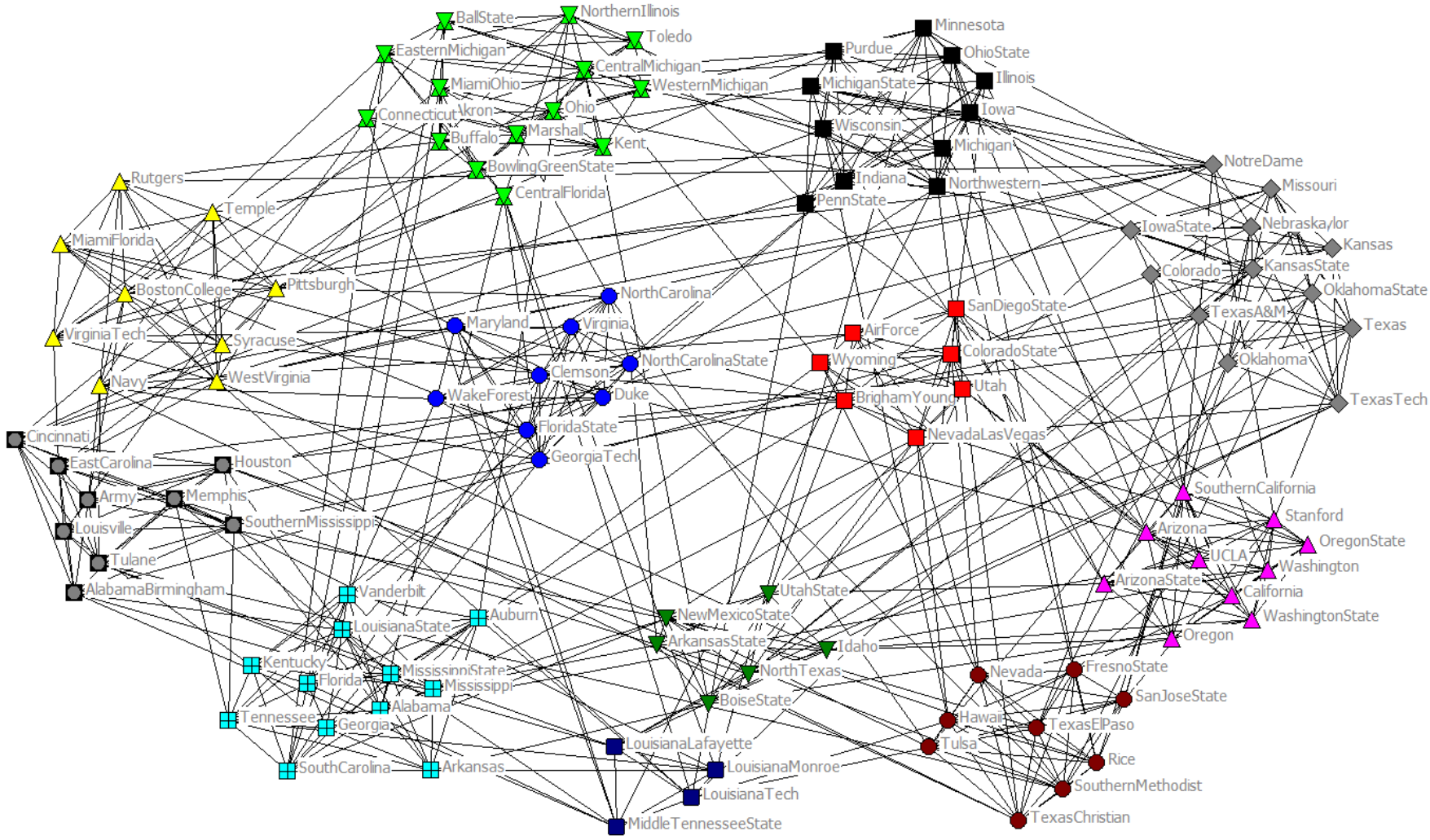


(a)

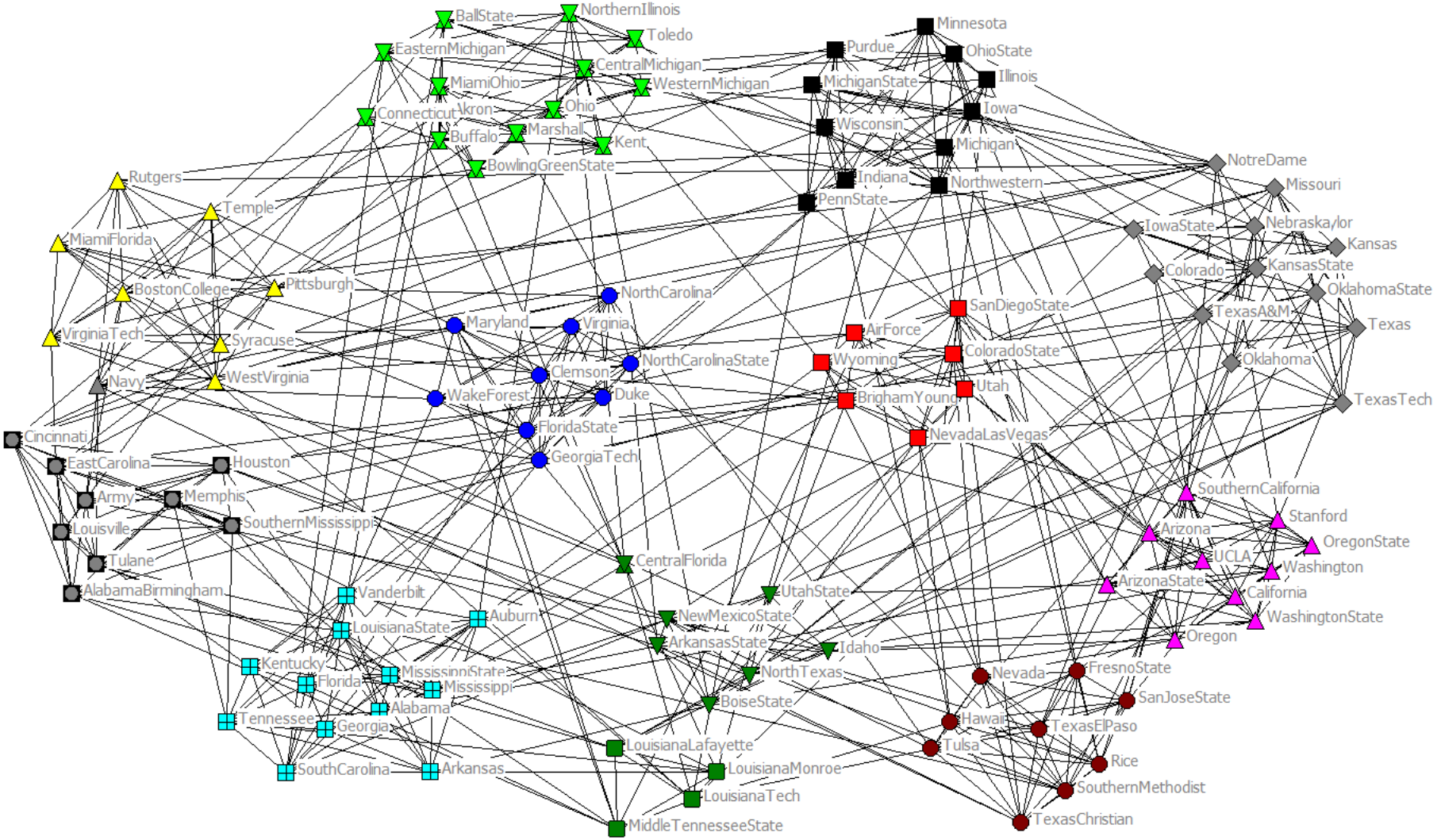


(b)

Şekil 4.7 Çalışmamızın Bottlenose yunusları ağı için bulduğu topluluklar



Şekil 4.8 Amerikan kolej futbol takımı ağı ve toplulukları



Şekil 4.9 Amerikan kolej futbol takımı ağı için bulduğumuz topluluklar

5. SONUÇ

Sosyal ağ analizi son yıllarda gittikçe ilgi gören bir alandır. Canlı ve cansız varlıklar arasında birçok yönden ilişkiler mevcuttur. Sosyal ağ analizi bu yapıları bilgi ağı şeklinde ele alıp bu yapıların daha iyi anlaşılması ile ilgilenir. Ağdaki aktörler arasında olası bağlantı tahmini, kullanıcıların davranışlarından örüntü çıkarımı, topluluk keşfi sosyal ağ analizi alanındaki çalışma konularından bazılarıdır.

Sosyal ağlardaki topluluklar, ağı daha iyi anlamamıza yardımcı olurlar; içeresindeki aktörlerin ve bağlantıların bütün ağla nasıl bir anlam ilişkisi içerisinde olduğu hakkında ipuçları verirler. Çalışmamızda [56] sosyal ağlarda topluluk keşfi problemi ele alındı ve sosyal ağlarda toplulukları keşfetmek için bir yöntem önerildi. Yöntemimizde çok amaçlı genetik algoritma kullanıldı. Uygunluk fonksiyonu olarak *topluluk skoru* [10] ve *topluluk uygunluğu* [25] fonksiyonları kullanıldı. Ağdaki düğümler aralarında ortak komşuları fazla olan düğümlerle aynı toplulukta olma eğilimi gösterirler. Zaten topluluk tanımı da bunu gerektirmektedir. Bu durum dikkate alınarak genetik algoritmanın popülasyon oluşturma ve mutasyon işlemleri önerilen seçim yöntemi ile yapıldı. Önerdiğimiz yöntemin daha iyi sonuçlar elde edilmesinde katkısı olduğu test sonuçlarında görüldü.

Yaklaşımımızı yapay ve gerçek sosyal ağlarda test ettik. Yapay sosyal ağlar için Lancichinetti ve Fortunato'nun [23] önerdiği yapay sosyal ağlar kullanıldı. Bu yapay sosyal ağlar verilen parametrelere göre üretilen ve toplulukları bilinen ağlardır. Test için topluluk yapılarının bulunma güçlüğü kolaydan zora doğru farklı ağlar üretildi. Ölçüt birimi olarak NMI'yı kullandık. Benzer bir çalışma (MOGA-Net [10]) ile sonuçlar karşılaştırıldı. Çalışmamız test ağlarının birçoğunda MOGA-Net'e üstünlük gösterdi. Önerimiz topluluk keşfi çalışmalarında yaygın olarak kullanılan gerçek ağlardan Zachary'nin Karate kulübü, Bottlenose yunusları ve Amerikan kolej futbol takımı ağları üzerinde test edildi. Toplulukları bilinen gerçek ağlarda başarı kıstasını belirlemek güçtür. Çünkü var olan topluluk yapısı ideal topluluk yapısına uymayabilir. Bu nedenle ölçüt birimi olarak *modülerlik* kullanıldı. Zachary'nin Karate kulübü ağı için hem gerçek topluluklara ulaşıldı hem de farklı bir topluluk çözümü sunuldu. Bulduğumuz farklı çözümün modülerlik değeri (0,3991) gerçek toplulukların modülerlik değerine (0.3715) üstünlük gösterdi. Bottlenose yunusları ağı için gerçek topluluklara ulaşılamadı. Ancak gerçek toplulukların modülerlik değerinden (0.3735) daha iyi değerler (0.4802, 0.4886)

elde edildi. Amerikan kolej futbol takımı ağı için de gerçek toplulukların modülerlik değerine (0.597) çok yakın bir değer (0.592) elde edildi.

KAYNAKLAR

[1] **Nowotarski M.,** 2011. Challenges of Social Network Patents, *IP Watchdog, January 23, 2011*.

[2] **Nowell D. L. and Jon K.,** 2003. The link prediction problem for social networks, *In CIKM '03, pages 556–559, New York, NY, USA, ACM*.

[3] **Sun Y., Barber R., Gupta M., Aggarwal C., and Han J.,** 2011. Co-Author Relationship Prediction in Heterogeneous Bibliographic Networks, *2011 International Conference on Advances in Social Networks Analysis and Mining*.

[4] **Benevenuto F., Rodrigues T., Cha M., and Almeida V.A.F.,** 2009. Characterizing user behavior in online social networks, *In Internet Measurement Conference, pages 49–62*.

[5] **Fortunato S.,** 2010. Community detection in graphs, *Physics Reports,486(3-5):75 – 174.*

[6] **Gregory S.,** 2009. Finding overlapping communities using disjoint community detection algorithms, *In Complex Networks: CompleNet ,pages 47–61. Springer-Verlag*.

[7] **Leskovec J., Lang K.J., and Mahoney M. W.,** 2010. Empirical comparison of algorithms for network community detection, *In WWW,pages 631–640*.

[8] **Newman M. E. J. and Girvan M.,** 2004. Finding and evaluating community structure in networks, *Phys. Rev. E* , vol. 69, no. 026113.

[9] **Fortunato S. and Castellano C.,** 2009. Community structure in graphs, *in:R.A.Meyers(Ed.), Encyclopedia of Complexity and Systems Science ,vol.1, Springer, Berlin, Germany, eprintarXiv:0712.2716.*

[10] **Pizzuti C.,** 2012. A Multiobjective Genetic Algorithm to Find Communities in Complex Networks, *IEEE Trans. Evolutionary Computation*, vol. 16, no. 3, pp. 418-430.

[11] **Radicchi F., Castellano C., Cecconi F., Loreto V., and Parisi D.,** 2004. Defining and Identifying Communities in Networks, *PNAS*, vol. 101, no. 9, pp. 2658-2663.

[12] http://www.orgnet.com/sna.html Social Network Analysis, A Brief Introduction, 19 Mart 2013.

[13] **Freeman L.,** 1977. A set of measures of centrality based upon, *Sociometry 40, 35–41*.

[14] **Girvan M. and Newman M. E. J.,** 2002. Community structure in social and biological networks, *Proc. Natl. Acad. Sci.*, vol. 99, no. 12, pp. 7821–7826.

[15] http://discopal.ispras.ru/SocialGraphs/Community_Detection, SocialGraphs/Community Detection, 21 Mart 2013.

[16] http://www.soc.ucsb.edu/faculty/friedkin/Syllabi/Soc148MA/Eigenvector Centrality.pdf, EIGENVECTOR CENTRALITY, 21 Mart 2013.

[17] Gürsakal N., *Sosyal Ağ Analizi*. Bursa, Türkiye: Dora, 2009.

[18] **Leicht E. A. and Newman M. E.J.,** 2008. Community structure in directed networks, *Physical Review Letters*, vol. 100, no. 11, p. 118703.

[19] **Fortunato S. and Barthelemy M.,** 2007. Resolution limit in community detection, *Proc. Natl. Acad. Sci. USA*, vol. 104, no. 1, pp. 36-41.

[20] **Nicosia V., Mangioni G., Carchiolo V., and Malgeri M.,** 2009. Extending the defnition of modularity to directed graphs with overlapping communities, *J. Stat. Mech. P03024*.

[21] **Zachary W.W.,** 1977. An information flow model for conflict and fission in small groups, *Journal of Anthropological Research, 33:452–473*.

[22] **Lusseau D.,** 2003. The emergent properties of dolphin social network, *Biology Letters, Proc. R. Soc. London B (suppl.)*, vol. 270, no. 186-188.

[23] **Lancichinetti A. and Fortunato S.,** 2009. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities, *Phys. Rev. E 80, 016118*.

[24] **Danon L., Daz-Guilera A., Duch J., and Arenas A.,** 2005. Comparing community structure identification, *Journal of Statistical Mechanics, P09008*.

[25] **Lancichinetti A., Fortunato S., and Kertesz J.,** Mar. 2009. Detecting the overlapping and hierarchical community structure of complex networks, *New J. Phys.*, vol. 11, no. 033015.

[26] **Newman M. E.J.,** 2004. Fast algorithm for detecting community structure in networks, *Physical review E*, vol. 69, no. 6, p. 066133.

[27] **Clauset A., Newman M. E. J., and Moore C.,** 2004. Finding community structure in very large networks, *Physical Review E,70:066111*.

[28] http://igraph.sourceforge.net/screenshots2.html, The igraph library, 22 Mayıs 2013.

[29] **Blondel V.D., Guillaume J., Lambiotte R., and Lefebvre E.,** 2008. Fast unfolding of communities in large networks, *J. Stat. Mech. P10008*.

[30] **Tasgin M. and Bingol H.,** 2006. Community detection in complex networks using genetic algorithm, *cond-mat/0604419*.

[31] **Jin D., He D., Liu D., and Baquero C.,** 2010. Genetic algorithm with local search for community mining in complex networks, *Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on*, pp. 105-112.

[32] **Park Y. J. and S. Song M.**, 1989. A genetic algorithm for clustering problems, *in Proc. 3rd Annu. Conf. Genet. Algorithms*, pp. 2-9.

[33] **Handle J. and Knowles J.,** 2007. An Evolutionary Approach to Multiobjective Clustering, *IEEE Transactions on Evolutionary Computation*, pp. 56-76.

[34] **Matake N., Hiroyasu T., Miki M., and Senda T.,** 2007. Multiobjective clustering with automatic k-determination for large-scale data, *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pp. 861-868.

[35] **Pizzuti C.,** 2008. Ga-net: a genetic algorithm for commu-nity detection in social networks, *In Proc. of the 10th Intenational Conference on Parallel Problem Solving from Nature (PPSN 2008), pages 1081–1090*.

[36] **Guoqiang C., Yuping W., and Yifang Y.,** 2011. Community Detection inComplex Networks Using Immune Clone Selection Algorithm, *International Journal of Digital Content Technology and its Applications*.

[37] **Pizzuti C.,** 2009. A Multi-Objective Genetic Algorithm for Community Detection in Network, *Proc. Of the 21th International Conference on Tools with Artificial Intelligence ICTAI 2009*, Newark, New Jersey, USA, pp. 379-386.

[38] **Shi C., Yan Z., Pan X., Cai Y., and Bin W.,** 2011. Multi-objective decisionmaking in the detection of comprehensive community structures, *IEEE Congress on Evolutionary Computation*, pp. 1489-1495.

[39] **Gong M., Ma L., Zhang Q., and Jiao L.,** 2012. Community detection in networks by using multiobjective evolutionary, *Physica A*, vol. 391, no. 15, pp. 4050–4060.

[40] **Guimera R. and Amaral L.A.N ,** 2005. Functional cartography of complex metabolic networks, *Nature 433, 895-900*.

[41] **Liu J., Zhong W., Abbass H. A., and Green D. G.,** 2010. Separated and overlapping community detection in complex networks using multiobjective Evolutionary Algorithms, *IEEE Congress on Evolutionary Computation*, pp. 1-7.

[42] **Deb K., Pratap A., Agarwal S., and Meyarivan T.,** 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182-197.

[43] **Li Z., Zhang S., Wang R., Zhang X., and Chen L.,** 2008. Quantitative function for community detection, *Physical Review E*, vol. 77, no. 3, p. 036109.

[44] **Angelini L., Boccaletti S., Marinazzo D., Pellicoro M., and Stramaglia (2007) S.,** 2007. Identification of network modules by optimization of ratio association, *arXiv:cond-mat/0610182*.

[45] **Wei Y. C. and Cheng C. K.,** 1991. Ratio cut partitioning for hierarchical designs, *IEEE Trans. Comput. Aid. D*, vol. 10 (7), pp. 911- 921.

[46] **Rosvall M. and Bergstrom C. T.,** 2007. An information-theoretic framework for resolving community structure in complex networks, *Proc. Natl. Acad. Sci. USA 104 , 7327*.

[47] **Rosvall M. and Bergstrom C. T.,** 2008. Maps of random walks on complex networks reveal community structure, *Proc. Natl. Acad. Sci.USA 105 , 1118*.

[48] **Lancichinetti A. and Fortunato S.,** 2009. Community detection algorithms: a comparative analysis, *Phys. Rev. E 80 056117*.

[49] **Condon A. and Karp R. M.,** 2001. Algorithms for graph partitioning on the planted partition model, *Random Structures and Algorithms*, vol. 18, no. 2, pp. 116-140.

[50] **Guimera R., Sales-Pardo M., and Amaral L. A. N.,** 2004. Modularity from fluctuations in random graphs and complex networks, *Phys. Rev. E 70, art. no. 025101*.

[51] **Palla G., Derenyi I., Farkas I., and Vicsek T.,** 2005. Uncovering the overlapping community structure of complex networks in nature and society, *Nature, 435(7043):814–818*.

[52] **Dongen S. Van,**. Ph.D. thesis, Dutch National Research Institute for Mathematics and Computer Science University of Utrecht, Netherlands, *2000*.

[53] **Donetti L. and Munoz M. A.,** 2004. Detecting Network Communities: a new systematic and efficient algorithm, *J. Stat. Mech. P10012*.

[54] **Newman M.E.J. and Leicht E.A.,** 2007. Mixture models and exploratory analysis in

networks, *Proc. Natl. Acad. Sci. USA 104, 9564–9569*.

[55] **Ronhovde P. and Nussinov Z.,** 2009. Multiresolution community detection for megascale networks by information-based replica correlations, *Phys. Rev. E*, vol. 80, no. 1, p. 016109.

[56] **Bütün E. and Kaya M.,** 2013. A Multi-Objective Genetic Algorithm For Community Discovery, *Intelligent Data Acquisition and Advanced Computing Systems (IDAACS) 2013 7th IEEE International Conference on*.

[57] http://tr.wikipedia.org/wiki/Genetik_algoritma, 3 Mayıs 2012.

[58] **Matlab_User's_Guide,** 2004. Genetik algorithm and direct search toolbox.

[59] **Girvan M. and Newman M. E. J.,** 2002. Community structure in social and biological networks, *Proc. Natl. Acad. Sci. USA 99, 7821–7826*.

[60] http://www.geovista.psu.edu/GeoSocialApp/howitworks.html, Spatial-Social Network Visualization for Exploratory Data Analysis, 28 Mart 2013.

ÖZGEÇMİŞ

Ertan BÜTÜN
Elazığ – Türkiye
ertanbutun@gmail.com

KİŞİSEL BİLGİLER:

Doğum tarihi : 01.01.1984
Doğum Yeri : ELAZIĞ

EĞİTİM:

2003–2008	Dokuz Eylül Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği, İzmir
2002–2003	Dokuz Eylül Üniversitesi Yabancı Diller Yüksek Okulu (İngilizce Hazırlık), İzmir
1997–2000	Balak Gazi Lisesi, Elazığ

İŞ TECRUBELERİ:

Kurum adı:	Fırat Üniversitesi Bilgisayar Mühendisliği Araştırma Görevlisi (2011-Halen çalışıyorum)
Şirket adı:	Proklab Bilişim Sistemleri (İzmir, http://proklab.com 2008-2010)