

**WEB TABANLI KOMPLEKS
AĐ SİMÜLATÖRÜ TASARIMI**

**2016
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĐİ**

Gökhan KUTLUANA

**WEB TABANLI KOMPLEKS
AĐ SİMULATÖRÜ TASARIMI**

Gökhan KUTLUANA

Karabük Üniversitesi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalında

Yüksek Lisans Tezi

Olarak Hazırlanmıştır.

KARABÜK

Haziran 2016

Gökhan KUTLUANA tarafından hazırlanan “WEB TABANLI KOMPLEKS AĞ SİMÜLATÖRÜ TASARIMI” başlıklı bu tezin Yüksek Lisans Tezi olarak uygun olduğunu onaylarım.

Yrd. Doç. Dr. İlker TÜRKER

Tez Danışmanı, Bilgisayar Mühendisliği Anabilim Dalı



Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir. 14/ 06/ 2016


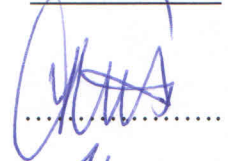
Ünvanı, Adı SOYADI (Kurumu)

Başkan : Yrd. Doç. Dr. Yüksel ÇELİK (KBÜ)

Üye : Yrd. Doç. Dr. İlker TÜRKER (KBÜ)

Üye : Doç. Dr. Ergin YILMAZ (BEÜN)

İmzası



...../...../2016

KBÜ Fen Bilimleri Enstitüsü Yönetim Kurulu, bu tez ile, Yüksek Lisans derecesini onamıştır.

Prof. Dr. Nevin AYTEMİZ

Fen Bilimleri Enstitüsü Müdürü





“Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim.”

Gökhan KUTLUANA

ÖZET

Yüksek Lisans Tezi

WEB TABANLI KOMPLEKS AĞ SİMULATÖRÜ TASARIMI

Gökhan KUTLUANA

Karabük Üniversitesi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:

Yrd. Doç. Dr. İlker TÜRKER

Haziran 2016, 112 Sayfa

Günümüzün popüler çalışma alanlarından olan kompleks ağlar, gerçek dünyada var olan birçok kompleks sistemi ifade etmek için kullanılırlar. Kompleks ağlar üzerine yapılan çalışmalar; ağların analiz edilmesi, modellenmesi ve görselleştirilmesi üzerine yoğunlaşmıştır. Bu çalışmada, bilinen temel kompleks ağ modellerinin ve bu modeller arasında gerçekleşen dönüşümler sonucunda oluşan ağların analizi yapılmış ve bu modeller geliştirilen web tabanlı uygulamayla görselleştirilmiştir. Barabási-Albert, Erdős-Rényi ve Düzenli Ağ modelleri arasında dönüşümler gerçekleştirilmiştir. Bu dönüşümlerin sonucunda, kümelenme katsayısı, ortalama yol uzunlukları ve derece dağılımı özelliklerinin değişimi çıktı olarak verilmiştir. Ayrıca kurulum gerektirmeyen web tabanlı uygulama ile bu modeller ve aralarındaki dönüşümler kolay erişilebilir ve eğitilebilir hale getirilmiştir.

Anahtar Sözcükler : Kompleks ağlar, ağ modelleme, görselleştirme, düzenli ağlar,
ölçek-bağımsız ağlar, rassal ağlar.

Bilim Kodu : 902.1.014



ABSTRACT

M. Sc. Thesis

DESIGN OF THE WEB-BASED COMPLEX NETWORK SIMULATOR

Gökhan KUTLUANA

**Karabük University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering**

**Thesis Advisor:
Assist. Prof. Dr. İlker TÜRKER**

June 2016, 112 pages

Complex networks, as a popular field of study, are used for explaining many complex systems in the real world. Studies on complex networks have mostly focused on the modeling, analysis and visualization of the networks. In this study, the analyses of well-known complex network models and the networks resulting from transformations defined among these models are performed. These networks are also visualized by the web-based app we have developed. Transformations among Barabási-Albert, Erdős-Rényi and Regular Network models are carried out. As a result of these transformations, the clustering coefficient, the mean path lengths and degree distribution properties are given as output. Furthermore, these models and the transformations among them have been easily accessible and educatable by the developed web-based app that does not require installation.

Key Words : Complex networks, network modelling, visualization, regular networks, scale-free networks, random networks.

Science Code : 902.1.014



TEŐEKKÜR

Bu tez alıőmasının planlanmasında, araőtırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteęini esirgemeyen, engin bilgi ve tecrübelerinden yararlandıęım, yönlendirme ve bilgilendirmeleriyle alıőmamı bilimsel temeller ışığında őekillendiren sayın hocam Yrd. Do. Dr. İlker TÜRKER'e sonsuz teőekkürlerimi sunarım.

Baőta eőim Özlem KUTLUANA olmak üzere tüm aileme manevi hiçbir yardımı esirgemedен yanımda oldukları için tüm kalbimle teőekkür ederim.

İÇİNDEKİLER

	<u>Sayfa</u>
KABUL.....	ii
ÖZET	iv
ABSTRACT.....	vi
TEŞEKKÜR.....	viii
İÇİNDEKİLER	ix
ŞEKİLLER DİZİNİ	xii
ÇİZELGELER DİZİNİ	xv
SİMGELER VE KISALTMALAR DİZİNİ	xvi
BÖLÜM 1	1
GİRİŞ	1
BÖLÜM 2	3
KOMPLEKS AĞLAR	3
2.1. KOMPLEKS AĞ NEDİR?	3
2.1.1. Küçük Dünya Özelliği	4
2.1.2. Kümelenme.....	5
2.1.3. Derece Dağılımı.....	6
2.2. GERÇEL AĞ ÇALIŞMALARI	7
2.3. KOMPLEKS AĞ MODELLERİ	8
2.3.1. Erdős-Rényi Model	9
2.3.1.1. Derece Dağılımı.....	10
2.3.1.2. Bağlantılılık Durumu Ve Çap	12
2.3.1.3. Kümelenme Katsayısı	14
2.3.2. Düzenli (Regular) Ağ Modeli.....	15
2.3.3. Watts-Strogatz Model.....	15
2.3.4. Barabási-Albert Model	18

	<u>Sayfa</u>
BÖLÜM 3	21
KOMPLEKS AĞ GÖRSELLEŞTİRME ARAÇLARI.....	21
3.1. KOMPLEKS AĞ GÖRSELLEŞTİRME UYGULAMALARI.....	22
3.1.1. Gephi	22
3.1.2. NetworkX	23
3.1.3. Pajek	24
3.1.4. IGraph.....	24
3.1.5. Diğer Masaüstü Görselleştirme Uygulamaları	25
3.1.6. Web Tabanlı Görselleştirme Araçları.....	25
3.1.4.1. Cytoscape.js JS Kütüphanesi	27
3.1.4.2. Cytoscape Web	28
3.1.4.3. Sigma.js JS Kütüphanesi	28
3.1.5. Görselleştirme Araçlarına Genel Bakış	30
BÖLÜM 4	31
KOMPLEKS AĞ MODEL DÖNÜŞÜM ÇALIŞMALARI	31
4.1. ER MODEL – DÜZENLİ AĞ MODELİ ARASI DÖNÜŞÜM.....	33
4.2. DÜZENLİ AĞ MODELİ – BA MODEL ARASI DÖNÜŞÜM	38
4.3. BA MODEL – DÜZENLİ AĞ MODELİ ARASI DÖNÜŞÜM	41
4.4. ER MODEL – BA MODEL ARASI DÖNÜŞÜM	44
4.5. BA MODEL – ER MODEL ARASI DÖNÜŞÜM	48
4.6. ER MODEL – DÜZENLİ AĞ VE BA MODELİ ARASI DÖNÜŞÜM.....	52
4.7. DÖNÜŞÜM MODELLERİNE GENEL BAKIŞ	56
BÖLÜM 5	59
WEB TABANLI KOMPLEKS AĞ SİMÜLATÖRÜ.....	59
5.1. YAZILIM BİLEŞENLERİ.....	60
5.1.1. jQuery ve jQuery UI Kütüphaneleri	60
5.1.2. jqPlot Kütüphanesi	61
5.1.3. Three.js Kütüphanesi.....	62
5.1.4. FileSaver.js Kütüphanesi.....	63
5.2. YAZILIM ÖZELLİKLERİ	64

	<u>Sayfa</u>
5.2.1. Model Seçimi İçin Sekmeli Ayar Paneli	64
5.2.2. 2B Görselleştirme Ve 3B Dönüşümler	65
5.2.3. 3B Görselleştirme Ve 3B Dönüşümler	66
5.2.4. Fare Olayları	66
5.2.5. Yerleşim Düzeni Değişimi	66
5.2.6. Derece Dağılımı, Kümelenme Katsayısı Ve Ortalama Yol Uzunluğu...	70
5.2.7. Ağ Çıktıları	70
5.3. UYGULAMA ÖRNEKLERİ	71
BÖLÜM 6	74
SONUÇLAR	74
EK AÇIKLAMALAR A. MODELLER ARASI DÖNÜŞÜMLER MATLAB KODLARI	80
EK AÇIKLAMALAR B. WEB TABANLI UYGULAMA YAZILIM KOD ÖRNEKLERİ	93
ÖZGEÇMİŞ	112

ŞEKİLLER DİZİNİ

Sayfa

Şekil 2.1.	Mary Pickford ve Kevin Bacon arasındaki bağlantı.	4
Şekil 2.2.	Rassal , Ölçek-bağımsız ağlar için derece dağılımları.	7
Şekil 2.3.	$P=5$ ve $E=4$ olan bir graf gösterimi.	8
Şekil 2.4.	Erdős-Rényi model ile oluşturulan grafın gelişimi.	10
Şekil 2.5.	Bir rassal grafın derece dağılımının sayısal gösterimi.	12
Şekil 2.6.	Bazı gerçel ağların ortalama yol uzunluklarının kıyaslanması ve rassal graf için kesikli çizgilerle 2.10'daki formülün tahmin edilmesi.	14
Şekil 2.7.	Bazı gerçel ağlarda oluşan kümelenme katsayısı oranıyla, 2.11'de olan formülün değeri kıyaslanmıştır.	15
Şekil 2.8.	Watts-Strogatz Model ile düğüm ve bağlantı sayısı değişmeden düzenli bir lattice grafi ile rassal graf arasında olan tekrar bağlanma işlemi.	16
Şekil 2.9.	Watts-Strogatz Model için $\ell(p)$ düğümler arası uzaklık ve $C(p)$ kümelenme katsayısı özellikleri.	17
Şekil 2.10.	Ortalama derecesi $k = 4$ olan BA model ile oluşturulmuş ağın kümelenme katsayısı ile $C_{rand} \approx \langle k \rangle / N$ olan rassal grafın kümelenme katsayısının karşılaştırılması.	20
Şekil 3.1.	Gephi'de oluşturulmuş bir ağ.	23
Şekil 3.2.	Pajek uygulama ara yüzü.	24
Şekil 3.3.	Cytoscape.js tarafından görselleştirilen bir gen etkileşim ağı.	27
Şekil 3.4.	$t=500$, $m_0 = 2$, $m=2$ olan bir Barabási-Albert model ağının Sigma.js ile görselleştirilmesi.	29
Şekil 4.1.	Poisson dağılımı olasılık kütle fonksiyonu.	34
Şekil 4.2.	Yerel minimum ve maksimum değerlere göre normalize edilmiş, kümelenme katsayısı ($CC=\square$), ortalama yol uzunluğu ($\langle d \rangle = \circ$) değişim grafiği, Global minimum ve maksimum değerlere göre normalize edilmiş kümelenme katsayısı ($CC=\blacksquare$), ortalama yol uzunluğu ($\langle d \rangle = \bullet$)	36
Şekil 4.3.	$p_{rewire}=0,9857$ olasılıkta oluşan derece dağılım grafiği.	37
Şekil 4.4.	Yerel minimum ve maksimum değerler için, kümelenme katsayısı ($CC=\square$), ortalama yol uzunluğu ($\langle d \rangle = \circ$) değişim grafiği, Global minimum ve maksimum değerler için kümelenme katsayısı ($CC=\blacksquare$), ortalama yol uzunluğu ($\langle d \rangle = \bullet$)	39
Şekil 4.5.	$p_{rewire}=0,1194$ olasılıkta oluşan derece dağılım grafikleri.	40

Şekil 4.6.	Yerel minimum ve maksimum değerlere göre normalize edilmiş, kümelenme katsayısı ($CC=\square$), ortalama yol uzunluğu ($\langle d \rangle = \circ$) değişim grafiği, Global minimum ve maksimum değerlere göre normalize edilmiş kümelenme katsayısı ($CC=\blacksquare$), ortalama yol uzunluğu ($\langle d \rangle = \bullet$)	43
Şekil 4.7.	$p_{rewire}=0,9930$ olasılıkta oluşan derece dağılım grafikleri	44
Şekil 4.8.	Genel minimum ve maksimum değerler için kümelenme katsayısı ($CC=\blacksquare$), ortalama yol uzunluğu ($\langle d \rangle = \bullet$)	46
Şekil 4.9.	$p_{rewire}=0,0588$ olasılıkta oluşan derece dağılım grafikleri	47
Şekil 4.10.	$p_{rewire}=1,0$ olasılıkta oluşan derece dağılım grafikleri	48
Şekil 4.11.	Global minimum ve maksimum değerlere göre normalize edilmiş, kümelenme katsayısı ($CC=\blacksquare$), ortalama yol uzunluğu ($\langle d \rangle = \bullet$)	50
Şekil 4.12.	$p_{rewire}=0,1194$ olasılıkta oluşan log-log derece dağılım grafiği, $p_{rewire}=0,4924$ olasılıkta oluşan log-log derece dağılım grafiği.	51
Şekil 4.13.	Yeniden bağlanma prosedürü örneği.....	53
Şekil 4.14.	Yerel minimum ve maksimum değerlere göre normalize edilmiş, kümelenme katsayısı ($CC=\square$), ortalama yol uzunluğu ($\langle d \rangle = \circ$) değişim grafiği, Global minimum ve maksimum değerlere göre normalize edilmiş, kümelenme katsayısı ($CC=\blacksquare$), ortalama yol uzunluğu ($\langle d \rangle = \bullet$)	55
Şekil 4.15.	$p_{rewire}=0,9857$ olasılıkta oluşan derece dağılım grafikleri	56
Şekil 5.1.	jQuery UI “Tabs” bileşeni kullanımı, “Dialog” bileşeni kullanımı	61
Şekil 5.2.	Başlangıçta $N=200$ ve $k = 6$ olan Düzenli bir ağın, $p_{rewire} = 0,4924$ yeniden bağlanma olasılığı ile BA modele dönüşümü sonucu oluşan derece dağılımı.	62
Şekil 5.3.	three.js kütüphanesi kullanılarak Watts-Strogatz model ağının simüle edilmesi sonucu oluşan grafik görüntü.....	63
Şekil 5.4.	Model seçimi için sekmeli ayar paneli	65
Şekil 5.5.	Geliştirilen uygulama kullanıcı ara yüzü.	65
Şekil 5.6.	Bir kompleks ağın 3B görselleştirilmesi, Fruchterman-Reingold yerleşim düzeni uygulanması ve fare ile yakınlaştırılması.	66
Şekil 5.7.	Rassal koordinatlarla oluşturulmuş grafın (solda), daire yerleşim düzeni (sağda) ile görselleştirilmesi.....	67
Şekil 5.8.	Daire yerleşim düzeni ile oluşturulmuş 3B ağ, bu ağa Fruchterman-Reingold yerleşim düzeni algoritması uygulanması.	69
Şekil 5.9.	Ağ çıktıları paneli.....	69
Şekil 5.10.	Watt-Strogatz modelde oluşturulmuş ve analiz edilmiş bir 2B kompleks ağ ve analiz sonuçları.....	71

Sayfa

Şekil 5.11. ER model-BA model arası dönüşüm sonucu oluşan 2B kompleks ağ ve analiz sonuçları.....	72
Şekil 5.12. BA model – Düzenli Ağ modeli arası dönüşüm sonucunda oluşan 3B bir kompleks ağ ve analiz sonuçları.	72
Şekil 5.13. ER model- Düzenli Ağ ve BA model arası dönüşüm sonucunda oluşan 3B bir kompleks ağ ve analiz sonuçları.	73



ÇİZELGELER DİZİNİ

Sayfa

Çizelge 2.1. Küçük dünya ağ örnekleri.....	18
Çizelge 4.1. Model dönüşümleri sonucu oluşan minimum-maksimum kümelenme katsayısı (CC) ve ortalama yol uzunluğu (<d>) değerleri..	32
Çizelge 4.2. ER Model – Düzenli Ağ Modeli Arası Dönüşüm denemeleri sonucu elde edilen kümelenme katsayısı (CC), ortalama yol uzunluğu (<d>) değerleri.	35
Çizelge 4.3. Düzenli Ağ Modeli- BA model arası dönüşüm denemeleri sonucu elde edilen kümelenme katsayısı (CC), ortalama yol uzunluğu (<d>) değerleri.	38
Çizelge 4.4. BA Model - Düzenli Ağ Modeli arası dönüşüm denemeleri sonucu elde edilen kümelenme katsayısı (CC), ortalama yol uzunluğu (<d>) değerleri.	42
Çizelge 4.5. ER Model - BA Model arası dönüşüm denemeleri sonucu elde edilen kümelenme katsayısı (CC), ortalama yol uzunluğu (<d>) değerleri.	45
Çizelge 4.6. BA Model - ER Model arası dönüşüm denemeleri sonucu elde edilen kümelenme katsayısı (CC), ortalama yol uzunluğu (<d>) değerleri.	49
Çizelge 4.7. ER Model – Düzenli Ağ ve BA Model arası dönüşüm denemeleri sonucu elde edilen kümelenme katsayısı (CC), ortalama yol uzunluğu (<d>) değerleri.	54

SİMGELER VE KISALTMALAR DİZİNİ

SİMGELER

CC	: Kümelenme Katsayısı
C	: Global Kümelenme Katsayısı
C_i	: Yerel Kümelenme Katsayısı
$\langle k \rangle$: Ortalama Derece
P(k)	: Dağılım Fonksiyonu
$\langle d \rangle$: Ortalama Yol Uzunluğu
P	: Düğüm Dizisi
E	: Bağlantı Dizisi
N	: Düğüm Sayısı
L	: Bağlantı Sayısı
λ	: Poisson Dağılımı Ortalama Parametresi

KISALTMALAR

API	: Application Programming Interface (Uygulama Programlama Arayüzü)
2B	: İki Boyutlu
3B	: Üç Boyutlu
BA	: Barabási-Albert
DA	: Düzenli Ağ
DA_BA	: Düzenli Ağ – Barabasi Albert
ER	: Erdős-Rényi
GEXF	: Graph Exchange XML Format
JS	: Javascript
JSON	: JavaScript Object Notation
WS	: Watts-Strogatz
WWW	: World Wide Web

BÖLÜM 1

GİRİŞ

Günümüzün popüler konularından olan kompleks ağlar, gerçek dünyada var olan birçok sistemi ifade etmek için kullanılmaktadır. Kompleks ağlar; biyolojik sistemler, sinir ağları, mekânsal oyunlar, genetik ağlar, bilimsel işbirliği ağları, sosyal ağlar, aktör ağları, bilgisayar ağları, besin ağları, elektrik şebekesi ağları ve linguistik ağlar ve daha birçok kendi kendini düzenleyen sistemlerin analizini kapsar [1-8].

Kompleks ağlar, Graf Teorisi ilkelerine göre tanımlanmaktadır. Örnekleri yukarıda belirtilen sistemler düğümler ve düğümler arasında kurulan bağlantılardan oluşmaktadır. Bu sistemler kompleks ağ yaklaşımı ile analiz edilmekte ve görsel hale getirilmektedir. Sistemler analiz edilerek sistemler hakkında daha detaylı bilgiler ortaya çıkarılmakta, sistem özellikleri ve sistemi meydana getiren öğeler arasında ilişkiler hakkında fikir sahibi olunmaktadır. Görselleştirme işlemleri, sistemlerin daha kolay anlaşılmasını ve insanların görsel yetileriyle sistemlerden daha kolay bilgi üretmelerini ve çok karmaşık verilerle uğraşılmadan sistemin temel topolojisini anlamalarında yardımcı olur.

Günümüzde yapılan kompleks ağ çalışmaları; modelleme, analiz ve görselleştirme üzerine yoğunlaşmaktadır. Modelleme çalışmalarında gerçek dünya ağlarının özellikleri dikkate alınarak bu özelliklere uygun modeller geliştirilmeye çalışılmaktadır. Bu modellerden en çok bilinenleri; Barabási-Albert, Erdős-Rényi (Rassal Ağ) ve Watts-Strogatz modelleridir.

Analiz işlemleri, kompleks ağların iç dinamiklerini daha detaylı biçimde anlamak amacıyla yapılmaktadır. Düğüm derece dağılımı, ağların yerel ve genel

kümelenmeleri, düğümler arası uzaklıklar vs. özellikler ortaya çıkarılarak ağlar daha anlaşılır ve yorumlanabilir hale gelmektedir.

Kompleks ağların görselleştirilmesi için birçok uygulama geliştirilmiştir. Bu uygulamalar 2B ve 3B olarak bu ağların görselleştirilmesini ve görselleştirilen graflar üzerinde kullanıcı etkileşimini sağlamaktadır. Görselleştirme amacıyla kullanılan uygulamalar genellikle bilgisayarda kurulum gerektirmekle birlikte günümüzde platform bağımsız ve herhangi bir kurulum gereksinim duymayan web tabanlı görselleştirme araçları da mevcuttur.

Gerçekleştirilen bu tez çalışmasında, en çok bilinen 3 temel model ve Düzenli Ağ modeli arasındaki dönüşümler gerçekleştirilmiş ve dönüşüm sonucunda oluşan ağlar incelenmiş ve özellikleri çıkarılmıştır. Daha sonra elde edilen bu dönüşümlerle birlikte 3 temel model için, platform bağımsız olan ve web tarayıcılarda çalışabilecek bir kompleks ağ simülatörü tasarlanmıştır.

Tez çalışmasının ikinci bölümünde kompleks ağlar ve özellikleri, gerçel ağlar ve özellikleri ile kompleks ağ modelleri hakkında daha detaylı bilgiler verilmiştir. Çalışmanın üçüncü bölümünde, kullanımı yaygın olan görselleştirme araçlarından detaylı bir şekilde bahsedilmiştir. Çalışmanın dördüncü bölümünde bilinen ağ modelleri arasında yapılan dönüşüm prosedürleri hakkında bilgi verilmiş, bu dönüşümler sonucunda ortaya çıkan sonuçlar paylaşılmıştır. Çalışmanın beşinci bölümünde, ikinci bölümde bahsedilen kompleks ağ modelleri ile dördüncü bölümde gerçekleştirilen model dönüşümlerinin, web tabanlı geliştirilen uygulamayla 2B ve 3B olarak görselleştirilmesi yapılmıştır. Altıncı bölümde ise tüm bu çalışmalar değerlendirilmiş olup yapılan çalışmanın sağladığı faydalardan bahsedilmiştir. İleride tez çalışmasının devamı niteliğinde neler yapılabileceğinden bahsedilmiştir.

BÖLÜM 2

KOMPLEKS AĞLAR

2.1. KOMPLEKS AĞ NEDİR?

Kompleks ağlar doğa ve toplumda bulunan geniş bir yelpazede yer alan sistemleri tanımlar. Örnek olarak hücre, birbirine kimyasal bağlarla bağlanan en iyi kompleks ağ olarak ifade edilebilir. İnternet de çeşitli fiziksel ve kablosuz bağlantılarla bilgisayar ve yönlendiricilerin birbirine bağlı olduğu bir kompleks ağdır. Sosyal ağlar üzerinde insanların çeşitli sosyal ilişkiler kurması yoluyla oluşan fikirler ve web sayfalarının bağlantılarla birbirine bağlı olduğu büyük bir sanal ağ olan WWW de kompleks ağlara örnek olarak verilebilir [1].

Geleneksel Kompleks ağ çalışmaları matematik alanında özellikle de graf teorisi üzerinde yoğunlaşmıştır. Graf Teorisi başlangıçta düzenli graflar üzerine odaklanmıştır. 1950'lerden itibaren geniş ölçekli ağlar, rassal graflarla ifade edilmiş ve kompleks ağın basit ve anlaşılır gerçekleşmesi için bu graflar önerilmiştir [1]. Günümüzün geniş çaplı gerçel ağları, Erdős-Rényi (ER) modelle üretilen rassal graflardan farklı olduğu için bu ağların benzetiminde yeni modellere ihtiyaç duyulmuştur [9].

Bilgisayarların veri toplama işlemi için her alanda kullanılması ile gerçel ağlarda büyük veri tabanları ortaya çıkmıştır. Bilgisayarların işlem güçlerinin artması, büyük düğüm sayılarına sahip ağları araştırmamıza ve topolojik detaylarına erişmemize olanak sağlamıştır. Araştırmacıların disiplinler arası farklı veri tabanlarına erişimleri, kompleks ağların genel özelliklerinin çıkarılmasına imkan vermiştir. Son olarak sistemlerin davranışlarının bir bütün olarak incelenmesine ihtiyaç duyulmuştur. Ortaya çıkan bu gelişmeler, kompleks ağ bileşenleri arasındaki etkileşimlerin daha detaylı anlaşılması gereğini kaçınılmaz hale getirmiştir [1].

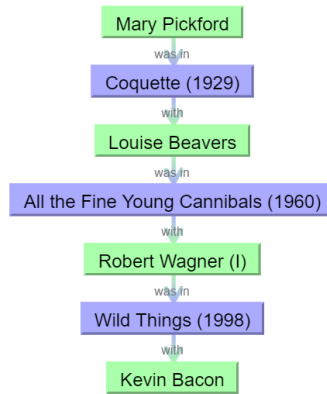
Günümüzde gerçel ağların 3 temel özelliği vardır [9]:

1. Küçük Dünya (Small World) Özelliği
2. Kümelenme (Clustering)
3. Power-Law Derece Dağılımı (Degree Distribution)

2.1.1. Küçük Dünya Özelliği

Dünyada rastgele seçilmiş herhangi iki kişi, araların da yer alan kişilerin oluşturduğu kısa zincirlerle birbirine bağlanır ve zincir uzunluğu yaklaşık olarak altıdır [8]. Bu konuda en bilinen çalışmanın sonucu olan, “six degrees of separation” (altı adımda ulaşım) kavramı bir sosyal psikolog olan Stanley Milgram tarafından ortaya çıkarılmış, Amerika Birleşik Devletlerinde çoğu insan çiftleri arasındaki yolun uzaklığının yaklaşık altı olduğunu ortaya koymuştur [1]. Bu çalışmada Nebraska- Boston şehirleri arasında rastgele seçilen kişiler vasıtasıyla mektuplar gönderilmiş ve alıcılara mektupların ortalama altı adımda ulaştığı gözlemlenmiştir.

Küçük Dünya etkisi çoğu kompleks ağda görülmektedir. Örnek olarak “The Six Degrees of Kevin Bacon” oyunu verilebilir. Oyunda aktör ve aktrislerin yer aldığı IMDB veri tabanı kullanılmış, bu veri tabanındaki oyuncuların birlikte rol aldıkları filmler, diziler vb. göz önünde bulundurularak Kevin Bacon’a olan uzaklıkları hesaplanmıştır. Ve uzaklıklarının hesaplanmasıyla Bacon Sayısı tanımlanmıştır. Şekil 2.1’de bu oyuna örnek olarak verilmiştir. Bacon Sayısı 3’tür.



Şekil 2.1. Mary Pickford ve Kevin Bacon arasındaki bağlantı [10].

Benzer bir şekilde Tom Remes'te aynı takımda oynamış beyzbol oyuncuları için, Bacon oyununda olduğu gibi, en fazla altı adımda beyzbol oyuncularının birbirlerine ulaşabildiğini belirlemiştir [11].

2.1.2. Kümelenme

Birçok gerçel ağ ve graf, birbirlerine tam bağlı alt graflardan meydana gelir. Bu tür yapılar klik olarak adlandırılır. En bilinen örnek bir sosyal ağda bir klik içerisinde yer alan arkadaşlardan herhangi birinin, diğer kişileri bilmesi olarak ifade edilebilir. Bu doğal eğilim, kümelenme katsayısı ile ölçülür [8]. Kümelenme katsayısı, geleneksel olarak iki yöntemle bulunur. Global kümelenme katsayısı [12] ve yerel kümelenme katsayısı'dır [8]. Global Kümelenme Katsayısı, ağ kümelenmesi hakkında genel bilgi verirken, yerel kümelenme katsayısı ise tekil düğümlerin kümelenmelerinin ortalaması hakkında bilgi verir [13].

Global Kümelenme Katsayısı bir ağ içerisinde düğümlerin oluşturduğu üçlülere dayanır. Üçlü, üç düğümün iki veya üç bağlantı kurmasıyla oluşur. İki bağlantılı olan açık (triplet), üç bağlantılı olan kapalı (triangle) üçlü olarak yönsüz graflarda ifade edilir. Bu parametreyi ölçmek için ilk çalışma Luce ve Perry tarafından yapılmıştır [14]. Global kümelenme katsayısı eşitlik 2.1 ile ifade edilir:

$$C = \frac{3xKapalı\ Üçlü\ Sayısı}{Toplam\ Üçlü\ Sayısı} \quad (2.1)$$

Yerel kümelenme katsayısı Global kümelenme katsayısına alternatif olarak kullanılmaktadır. Bir düğümün yerel kümelenme katsayısı eşitlik 2.2 ile ifade edilir:

$$C_i = \frac{2E_i}{k_i(k_i - 1)} \quad (2.2)$$

Formülde yer alan E_i değeri; $[1, N]$ arasındaki i . düğümün bağlı olduğu diğer düğümlerin birbirleri arasındaki bağlantı sayısıdır. i . düğüm ve ona bağlı diğer düğümlerin oluşturabileceği maksimum bağlantı sayısı $k_i(k_i - 1)/2$ 'dir. E_i

değerinin, $k_i(k_i - 1)/2$ değerine bölünmesiyle C_i her bir düğümün yerel kümelenme katsayısı bulunur. Ağda bulunan düğümler için bu değerlerin ortalaması alındığında ağın kümelenme katsayısı bulunmaktadır:

$$C = \frac{1}{N} \sum_{i=1}^n C_i \quad (2.3)$$

Bulunan kümelenme katsayısı ortalama yerel kümelenme katsayısı olarak da ifade edilir.

2.1.3. Derece Dağılımı

Bir düğümün bağlı olduğu komşu sayısı, o düğümün derecesi olarak adlandırılır. Bir kompleks ağda bulunan düğümlerin dereceleri birbirinden farklıdır. Bir ağda düğümlerin derecelerinin çeşitliliği, dağılım fonksiyonu $P(k)$ ile ifade edilir. Bu fonksiyon rastgele seçilen k adet bağlantıya sahip düğümlerin olasılığını ifade eder [1]. Rassal graflarda bağlantılar rastgele yerleştirildiğinden düğümlerin çoğunluğu benzer dereceye sahip olup, bu değer ağın ortalama derecesi $\langle k \rangle$ 'ya yakındır. Rassal grafların derece dağılımı, tepe noktası $P(\langle k \rangle)$, olan Poisson dağılımı özelliği gösterir.

$$P(k) = \frac{e^{-\langle k \rangle} \langle k \rangle^k}{k!} \quad (2.4)$$

Diğer yandan büyük gerçel ağlar üzerinde yapılan deneysel çalışmalarda dağılımın, Poisson dağılımından farklı olduğu görülmüştür. Bu oluşan dağılım power-law (güç-yasası)'dır. Ve bu ağlar scale-free (ölçek-bağımsız) ağ olarak ifade edilmektedir. Ölçek-bağımsız ağların derece dağılımı, eşitlik 2.5'te verilmiştir.

$$P(k) \sim k^{-\gamma} \quad (2.5)$$

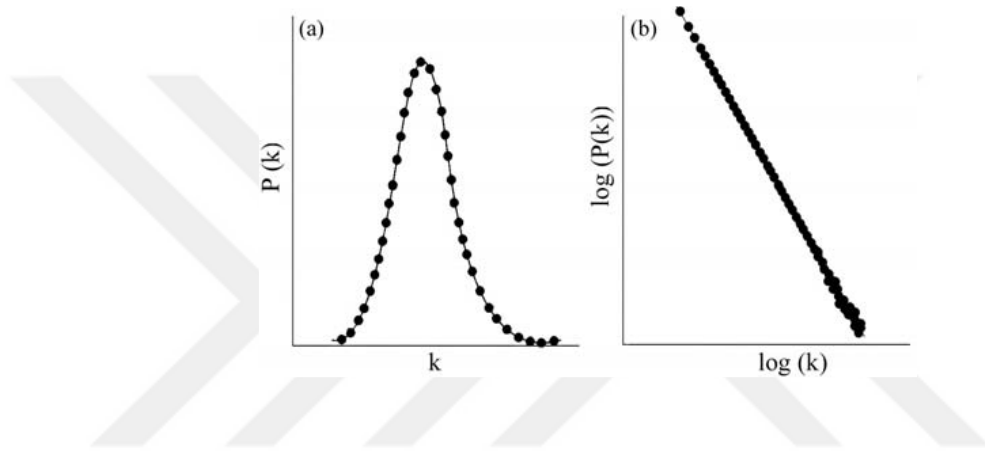
Bazı ağlar ise eşitlik 2.6'daki gibi exponential (üstel) derece dağılımı gösterirler.

$$P(k) \sim e^{-\lambda k} \quad (2.6)$$

Ya da eşitlik 2.7'deki gibi power-law derece dağılımının sağ tarafında exponential bir cut-off bölgesi barındırırlar.

$$P(k) \sim k^{-\gamma} e^{-\lambda k} \quad (2.7)$$

Bu cut-off bölgesi, ağıın scale-free olarak nitelendirilmesini engellemez.



Şekil 2.2. a) Rassal , b) Ölçek-bağımsız ağlar için derece dağılımları [15].

2.2. GERÇEL AĞ ÇALIŞMALARI

Kompleks ağ araştırmaları, son zamanlarda biyoloji, ekonomi, dilbilim, tıp, sosyal bilimler, teknoloji ve ulaşım gibi alanlarda çeşitli uygulamaları konu almıştır. Kompleks ağ çalışmaları birçok elemanı ve bağlantıyı içeren kompleks sistemlerin tanımlanması, simülasyonu, analiz edilmesi ve modellenmesi üzerine odaklanır. Örnek olarak internet, gen düzenleyici ağlar, protein-protein etkileşim ağları, sosyal ilişkiler ve WWW verilebilir [15].

Yapılan araştırmalarda graf içerisindeki düğümlerin birbirlerine olan ortalama uzaklıkları, kümelenme katsayısı ve derece dağılımı gibi özellikleri aynı özelliklere sahip rassal grafla kıyaslanmıştır. Ölçülen bu değerler sonucunda gerçel ağların küçük ortalama uzaklıklara, yüksek kümelenme katsayısı, birçoğunun güç yasası derece dağılımına sahip olduğu görülmüştür. İnternet, hücresel ağlar, WWW, bazı sosyal

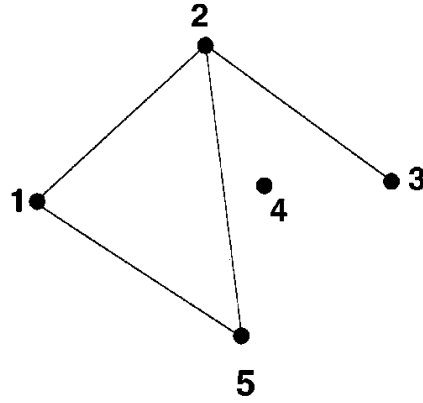
ağlar ve atıf ağları ölçek-bağımsızdır. Ancak Sinir ağları ve Elektrik şebekelerinde üssel dağılım ya da ölçek-bağımsız ve üssel dağılımların uyumlu bir şekilde harmanlandığı özellikler görülür.

Bu yapılan çalışmalar gösteriyor ki bu ağlar rassal ağ tanımlamasından çok uzaktır. Bu ağlar yeni düğüm ve noktaların sürekli eklenmesi ile gelişen ağlar olarak tanımlanabilir ve karakteristik olarak derece dağılımlarında power-law uyumlu bölgeler barındırırlar [9].

2.3. KOMPLEKS AĞ MODELLERİ

Matematiksel olarak bir ağ, graf olarak ifade edilir. Bir graf P ve E çiftinin kümesinden $G = \{P, E\}$ oluşur. P grafı oluşturan N adet düğümden oluşan düğümler dizisi $P_1, P_2, P_3, \dots, P_N$; E ise grafta yer alan iki düğüm arasında yer alan bağlantılar (link kenar) kümesidir.

Aşağıda örnek olarak bir graf verilmiştir.



Şekil 2.3. $P=5$ ve $E=4$ olan bir graf gösterimi, $P=\{1,2,3,4,5\}$ düğüm dizisi ve $E=\{\{1,2\},\{1,5\},\{2,3\},\{2,5\}\}$ bağlantı dizisi [1].

Graflar genellikle her bir düğümün bir noktaya karşılık geldiği noktalar dizisi olarak ifade edilen ve Şekil 2.3'te görüldüğü gibi noktaların birbirine çizgilerle bağlanmasıyla oluşan yapılardır [1].

Gerçek dünyadaki ağ yapılarının benzetimini yapmak için birçok ağ modeli öne sürülmüştür. Bunlar arasında en çok bilinen modeller şu şekildedir:

1. Erdős-Rényi Model
2. Düzenli Ağ Modeli
3. Watts-Strogatz Model
4. Barabási-Albert Model'dir

2.3.1. Erdős-Rényi Model

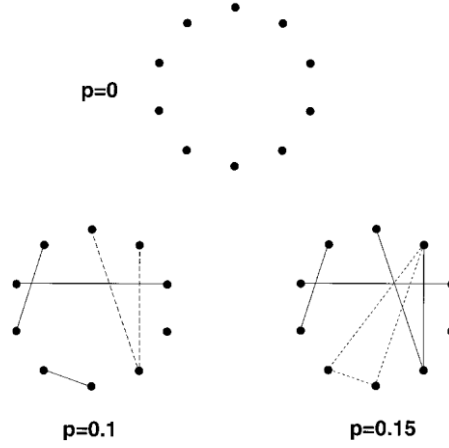
Erdős ve Rényi, rassal graflar üzerine yapılan ilk makalelerinde, N adet etiketli düğüm ve $N(N - 1)/2$ adet olası bağlantı arasından rastgele seçilen n bağlantı sayısından oluşan bir rassal graf tanımlar. N düğüm ve n bağlantıdan oluşan eşit olasılıklı gerçekleşme ihtimali olan toplamda $C_{[N(N-1)/2]}^n$ adet graftan oluşan bir olasılık uzayından oluşur [1].

Rassal grafi tanımlayan alternatif ve eş değer model binom modeldir. N düğümle başlar ve bir p olasılığıyla bütün düğüm çiftlerinin bağlanmasından oluşur (Şekil 2.4). Sonuç olarak toplam bağlantı sayısının beklenen değeri eşitlik 2.8'deki gibi ifade edilir.

$$E(n) = p[N(N - 1)/2] \quad (2.8)$$

N düğümlü bir ağda düğümler arasında oluşabilecek maksimum bağlantı sayısı $N(N - 1)/2$ 'dir. (2.8) eşitliği, bu bağlantıların p olasılığı ile tanımlanma prosedürünü ifade eder. Olasılık 1'e yaklaştıkça graf için bağlanılabilirlik artmaktadır. Tam bağlantılı graf için, $p \rightarrow 1 \Rightarrow n = N(N - 1)/2$ 'dir [9].

Eğer $G_0, P_1, P_2, P_3, \dots, P_N$ düğüm ve n bağlantıdan oluşan bir graf ise, bu grafın elde edilme olasılığı $P(G_0) = p^n (1 - p)^{[N(N-1)/2]-n}$ ile ifade edilir [1].



Şekil 2.4. Erdős-Rényi model ile oluşturulan grafın gelişimi [1].

Şekil 2.4'te $N=10$ düğüm başlangıçta $p=0$, için $E=0$ 'dır. $p=0.1$ için $E \approx 5$ 'dir ve $p=0.15$ için $E \approx 7$ 'dir. Düğümler arası oluşan bağlantılardan sonra üçlüler kesikli çizgilerle gösterilmiştir.

Rassal graflar, N adet düğüm arasında eşitlik 2.8 ile hesaplanan sayıdaki linkin, tamamen rassal kaynak ve hedef düğümler arasında tanımlanması ile oluşturulurlar. Linklerin tanımlama prosedüründe, aynı düğüm çiftleri arasındaki linklerin tekrarı ve kendi kendine linkler (self-link) engellenmelidir.

Rassal graflarda ortalama derece $\langle k \rangle = 2n/N = p(N - 1) \cong p.N$ eşitliği ile hesaplanır. Ortalama derecenin yanında düğümlerin dereceleri ile ilgili detaylı bilgiye ulaşmak, derece dağılımı ile mümkündür.

2.3.1.1. Derece Dağılımı

" p " bağlantı olasılığına sahip bir rassal grafta bir düğüm olan i . düğümün derecesi k_i 'nin p ve N 'e bağlı olarak değeri, eşitlik 2.9'daki olasılık ile tahmin edilebilir.

$$P(k_i = k) = C_{N-1}^k p^k (1 - p)^{N-1-k} \quad (2.9)$$

Bu olasılık, bir düğümden çizilebilir k farklı bağlantı yolunun sayısını ifade eder: k bağlantılarının olasılığı p^k , eklenen kenarların olmama olasılığı $(1-p)^{N-1-k}$ ve C_{N-1}^k seçilebilecek k bitiş noktası için eşdeğer yolların sayısını ifade eder [1].

Derece dağılımının incelemek için derecesi k olan düğümlerin sayısı X_k 'yi incelemek gerekir. Derecesi k olan düğümlerin beklenen sayısı:

$$E(X_k) = NP(k_i = k) = NC_{N-1}^k p^k (1-p)^{N-1-k} = \lambda_k \quad (2.10)$$

X_k değerlerinin dağılımı $P(X_k = r)$ Poisson dağılımına yaklaşıp.

$$P(X_k = r) = e^{-\lambda_k} \frac{\lambda_k^r}{r!} \quad (2.11)$$

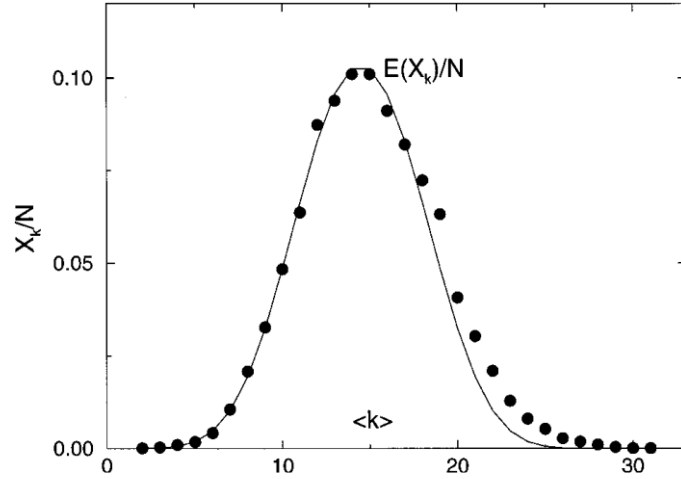
Böylece derecesi k olan düğümlerin sayısı ortalama λ_k değeriyle Poisson dağılımına uyar. Dağılımın beklenen değeri λ_k bir sabit değildir ve (2.10)'da yer alan fonksiyonla elde edilir. Poisson dağılımı çok büyük r değerleri için hızlı bir şekilde düşer, standart sapması $\sigma_k = \sqrt{\lambda_k}$ formülü ile bulunur. Basit sadeleştirmelerle elde edilen (2.11)'de elde edilen X_k değeri, $X_k = NP(k_i = k)$ değerinden çok da farklı değildir [1].

Böylece güzel bir yaklaşımla rassal grafın derece dağılımının (2.12) deki gibi binom olduğu,

$$P(k) = C_{N-1}^k p^k (1-p)^{N-1-k} \quad (2.12)$$

Çok büyük N değerleri için ise (2.13) deki gibi Poisson dağılımına dönüştüğü söylenebilir [9].

$$P(k) \simeq e^{-pN} \frac{(pN)^k}{k!} = e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!} \quad (2.13)$$



Şekil 2.5. Bir rassal grafın derece dağılımının sayısal gösterimi. 10 000 düğüm ve $p=0,0015$ değeriyle oluşturulmuştur. Grafikte Poisson dağılımına göre beklenen X_k/N değerleri kıyaslama amacıyla düz çizgi ile ifade edilmiştir. Standart Sapma oldukça küçüktür [1].

2.3.1.2. Bağlantılılık Durumu Ve Çap

Bir grafın çapı herhangi iki düğüm çifti arasındaki en kısa mesafelerin en uzunudur. Birbirinden izole birçok parçadan oluşan bağlantısız grafın çapı sonsuzdur. Bazen çap, grafi oluşturan parçaların en büyük çapa sahip olanın çapı olarak tanımlanabilir [9].

Rassal graflar p bağlantı olasılığının oldukça küçük olmadığı durumlarda küçük çaplara sahip olma eğilimindedir. Rassal bir grafın bağlantılılığının artıyor olması muhtemeldir. Büyük olasılıkla verilen bir düğümden ℓ mesafesi kadar uzak olan düğümlerin sayısı $\langle k \rangle^\ell$ değerinden daha küçük değildir. $\langle k \rangle^\ell$ ve N eşitlikleriyle çapı bulmak için $\ln(N)/\ln(\langle k \rangle)$ formülü kullanılır: Böylelikle çap düğüm sayısının logaritmasına bağlıdır [1].

Birçok araştırmacı tarafından rassal grafın çapı incelenmiştir. Birçok p değeri için genel kanı, neredeyse bütün grafların aynı N ve p değerleri için kesinlikle aynı çap değerine sahip olduğudur. Bütün N düğüm sayısı ve p bağlantı olasılığı olan graflar göz önüne alındığında, bu grafların çap değerlerinin aralığı çok küçük değişimler göstermekte, genellikle de belli bir değer aralığında yoğunlaşmıştır [1].

Bir rassal grafın çapı eşitlik 2.14 ile ifade edilir:

$$d = \frac{\ln(N)}{\ln(pN)} = \frac{\ln N}{\ln\langle k \rangle} \quad (2.14)$$

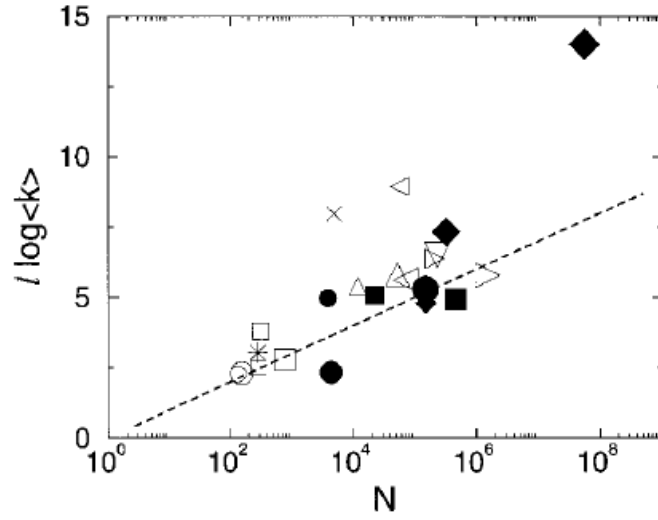
Rassal ağlarla ilgili birkaç önemli sonucu özetlemek gerekirse [9]:

1. Eğer $\langle k \rangle = pN < 1$ olursa graf izole ağaçlardan oluşur ve grafın çapı ağaçların içerisinde en büyük çapa sahip olana eşittir.
2. Eğer $\langle k \rangle = pN > 1$ olursa dev bir küme oluşur ve grafın çapı bu kümenin çapına eşittir.
3. Eğer $\langle k \rangle = pN \geq \ln(N)$ olursa graf tam bağlantılıdır. Ve çap $\ln(N)/\ln\langle k \rangle$ formülüyle bulunur.

Rassal bir grafın bağlılığını tanımlamak için bir diğer yol bütün düğüm çiftleri arasındaki ortalama uzaklıkları ya da ortalama yol uzunluğunu hesaplamaktır. Ortalama yol uzunluğu hesabı, yaklaşık olarak ağın çapının hesaplandığı eşitliğe (Bkz. Eşitlik 2.15) yakınsar [9],

$$\ell_{rand} \sim \frac{\ln N}{\ln\langle k \rangle} \quad (2.15)$$

Aşağıdaki şekilde bazı gerçel ağların ortalama yol uzunlukları verilmiştir. Şekle bakıldığında elde edilen gerçel ağların ortalama yol uzunluklarının rassal ağlardaki gibi oldukça kısa oldukları görülmektedir [1]. Yani gerçel ağlar, ortalama yol uzunlukları açısından rassal ağlar ile benzerlik göstermektedir.



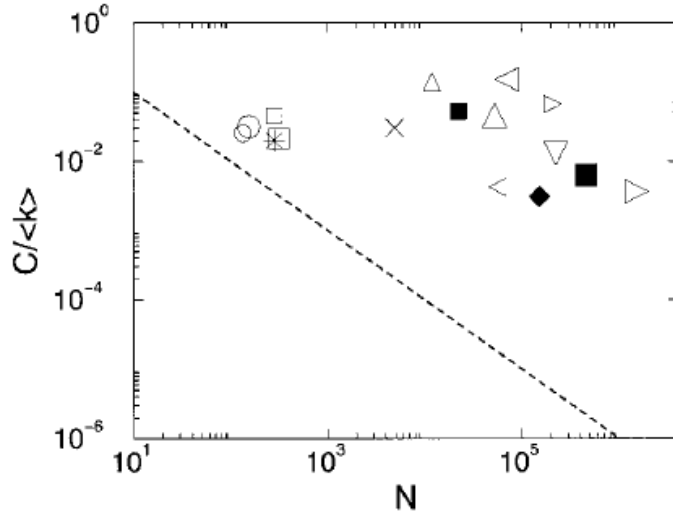
Şekil 2.6. Bazı gerçel ağların ortalama yol uzunluklarının kıyaslanması ve rassal graf için kesikli çizgilerle 2.10'daki formülün tahmin edilmesi [1].

2.3.1.3. Kümelenme Katsayısı

Kompleks ağlar yüksek kümelenme sergilerler. Rassal grafta bir düğümü en yakın komşularını düşünürsek, bu komşuların bağlanma olasılıkları, rastgele seçilen iki düğümün birbirine bağlanma olasılığına eşittir. Sonuç olarak rassal grafın kümelenme katsayısı eşitlik 2.16 ile ifade edilir [1].

$$C_{rand} = p = \frac{\langle k \rangle}{N} \quad (2.16)$$

Farklı boyutlardaki rassal graflar için düğüm sayısı N 'in fonksiyonuyla $C_{rand}/\langle k \rangle$ oranlarını çizdiğimizde, log-log grafik üzerinde düz bir çizgi eğimi (-1) boyunca sıralanacaktır. Gerçel ağların kümelenme katsayısı oranı ile rassal grafların tahmini kümelenme katsayısı birlikte çizildiğinde, gerçel ağlar rassal grafın kümelenme katsayısına uymamaktadır. $C/\langle k \rangle$ kesir değeri N^{-1} ile azalmaz, bu değerden bağımsızdır [1, 9]. Yani rassal ağlar, kümelenme katsayısı açısından gerçel ağlardan ayrılmaktadır.



Şekil 2.7. Bazı gerçel ağlarda oluşan kümelenme katsayısı oranıyla, 2.11.'de olan formülün değeri kıyaslanmıştır [1].

2.3.2. Düzenli (Regular) Ağ Modeli

Düzenli bir ağda, Şekil 2.8'de sol tarafta olduğu gibi bütün düğümler sıralı k komşuya bağlanmış bir örgü veya ağ düzeni içerisinde sıralanmıştır. Bu ağ aynı düğüm dereceleri ve düğümler arasında nispeten uzun uzaklıklar olduğundan küçük dünya ve scale-free özelliklerinden yoksun iken, gerçel ağlardaki gibi yerel bağlantılarla birlikte karakteristik olarak yüksek kümelenme gösterir. Düzenli ağın oluşumu [8]'de tanımlanmıştır.

2.3.3. Watts-Strogatz Model

Watts ve Strogatz sonlu boyutlu bir düzenli graf ile rassal graf arasında olan bir model önermişlerdir [1]. Önerdikleri bu modelle tıpkı düzenli graflarda olduğu gibi yüksek kümelenme ve rassal graflardaki gibi düğümler arası uzaklıkların kısa olduğu bir ağ tanımlanabileceğini göstermişlerdir [8].

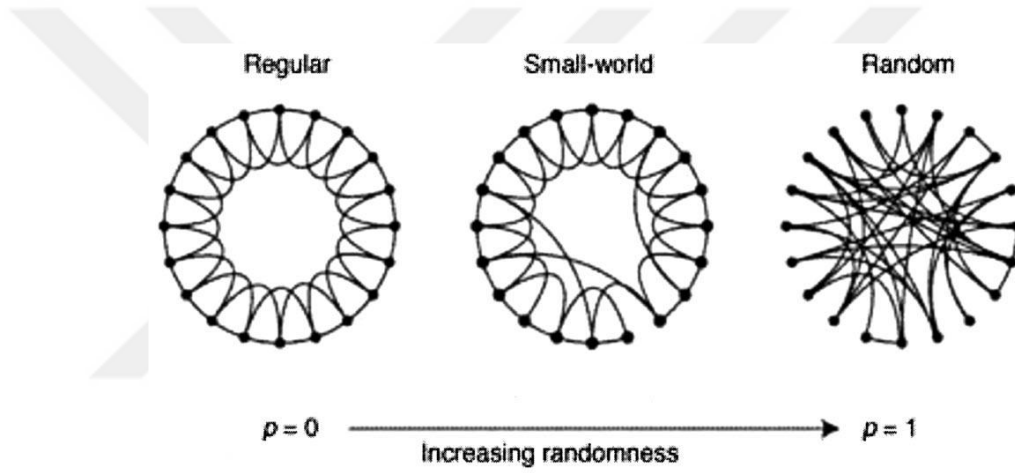
Modeli oluşturan algoritma aşağıdaki gibidir [1]:

1. Düzenli bir graf ile başlama: N düğüm sayısına sahip ve her bir düğümün k sayıda bağlantıya sahip olduğu ($k/2$ kadarı kendinden önceki düğümlere bağlı,

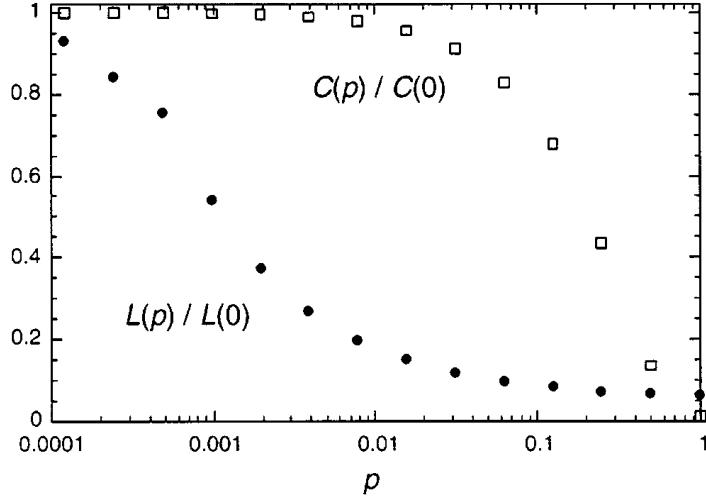
$k/2$ kadarı ise kendinden sonraki düğümlere bağlı) düzenli bir çember graf ile başlanır.

2. Yeniden Bağlanma: Düzenli çember grafı oluşturan tüm bağlantılar, bir p_{rewire} olasılığı ile koparılarak kaynak düğüm sabit kalacak şekilde hedef düğüm tamamen rassal olarak yeniden belirlenir.

Şekil 2.8'de $N=20$ düğüm sayısı ve $k=4$ bağlantı sayısından oluşan düzenli kafes (lattice) grafı kullanılmıştır. Tekrar bağlanma olasılığı $p=0$ iken grafta hiçbir değişiklik olmamış; p değerinin artışıyla graf düzensizleşmeye başlamış ve $p=1$ iken bütün bağlantılar rassal hale gelmiştir [8].



Şekil 2.8. Watts-Strogatz Model ile düğüm ve bağlantı sayısı değişmeden düzenli bir lattice grafı ile rassal graf arasında olan tekrar bağlanma işlemi [1].



Şekil 2.9. Watts-Strogatz Model için $\ell(p)$ düğümler arası uzaklık ve $C(p)$ kümelenme katsayısı özellikleri. Veriler $\ell(0)$ ve $C(0)$ çember lattice değerlerine göre normalize edilmiştir. Küçük dünya olgusuna karşılık gelen $\ell(p)$ değerinin hızlı düşüş gösterdiği bölge, yatay eksenin logaritmik ölçeklenmesi ile daha açık bir biçimde gösterilmiştir [1].

Yüksek kümelenme ve düğümler arası kısa uzaklığı birlikte anlamak için, p tekrar bağlanma olasılık değerine bağlı olarak $\ell(p)$ düğümler arası uzaklık ve $C(p)$ kümelenme katsayısı fonksiyonlarının değişimlerinin incelenmesi gerekir[1]. Çember lattice için $\ell(0) \sim n/2k \gg 1$ ve $C(0) \sim 3/4$ 'dür; böylece ℓ sistemin boyutuyla birlikte doğrusal ölçeklenir ve kümelenme katsayısı büyüktür. Diğer uç durumda $p \rightarrow 1$ için model rassal grafa $\ell(0) \sim \ln(N)/\ln(\langle k \rangle)$ yakınsar ve kümelenme $C(1) \sim K/N$ olur. Böylece ℓ , N ile birlikte logaritmik ölçeklenir ve kümelenme katsayısı N arttıkça azalır [1]. Böyle sınırlayıcı durumlar, yüksek kümelenme (C), düğümler arası uzun uzaklıklarla (ℓ), düşük kümelenme (C) düğümler arası kısa uzaklıklarla (ℓ) her zaman birleşir olgusunu destekleyebilir [8].

Bunun aksine, Şekil 2.9'da $\ell(p)$ değerinin ℓ_{random} kadar küçük olduğu ve $C(p) \gg C_{random}$ olduğu bir aralık vardır [8]. Küçük p değerlerinde, kümelenmenin genellikle değişmediği ve düğümler arası uzaklıkların hızla azaldığı görülmektedir. Düğümler arası kısa uzaklıklarının ve yüksek kümelenmenin görüldüğü durum küçük dünya olarak adlandırılan gerçel ağların özellikleriyle mükemmel bir uyum içindedir [1].

Çizelge 2.1. Küçük dünya ağ örnekleri [8].

	L_{actual}	L_{random}	C_{actual}	C_{random}
<i>Film Actors</i>	3,65	2,99	0,79	0,00027
<i>Power Grid</i>	18,7	12,4	0,080	0,005
<i>C. Elegans</i>	2,65	2,25	0,28	0,05

Yukarıdaki tabloda 3 farklı gerçel ağın kümelenme katsayısı ve düğümler arası uzaklıkları, her bir düğümün ortalama bağlantı sayısı (k) ve düğüm sayısı (n) aynı olan bir rassal graf ile karşılaştırılmıştır. Aktör ağı için $n=225.226$ ve $k=61$, güç şebekesi için $n=4.941$ ve $k=2,67$, *C. Elegans* sinir ağı için $n=282$ ve $k=14$ 'dür. Bu üç ağda da küçük dünya etkisi görülmektedir: $L \gtrsim L_{random}$, $C \gg C_{random}$ [8].

Küçük dünya modelde gerçel yol uzunluklarının dağılımı ve ortalama yol uzunluğu ℓ tam anlamıyla henüz hesaplanamamıştır; analitik hesaplamalar yapmanın bu model için zor olduğu kanıtlanmıştır [11].

Watts-Strogatz model için kümelenme hesaplanırken 2.1'de yer alan global kümelenme katsayısından yararlanır. Bu modelde derece dağılımı rassal grafın derece dağılımına benzer. Ağın topolojisi nispeten homojen ve bütün düğümler yaklaşık olarak aynı bağlantı sayısına sahiptir [1].

Watts-Strogatz modelin kümelenme katsayısı ve ortalama yol uzunluğu özellikleri gerçel ağlara benzemektedir. Fakat derece dağılımı, rassal graf özelliği gösterdiğinden birçok gerçel ağ derece dağılımından farklıdır.

2.3.4. Barabási-Albert Model

Gerçel ağların iki önemli özelliği ER model ve WS modele dâhil edilmemiştir. Birincisi, her iki modelde de sabit sayıda düğüm sayısı (N) ile başlanmış ve ER modelde rassal bağlantılar, WS modelde ise yeniden bağlantılar tanımlanmış ve yeni bir düğüm eklenmemiştir. Bu durumun aksine gerçel ağların birçoğu sisteme düğümlerin sürekli eklenmesiyle genişlemektedir. Böylelikle düğüm sayısı N ağın yaşam zamanı boyunca artmaktadır. Örnek olarak aktör ağına yeni aktörlerin

eklenmesiyle büyümesi, WWW ağının yeni web sayfalarının zamanla eklenmesiyle katlanarak büyümesi ve araştırma literatürünün sürekli yeni yayınlarla büyümesi verilebilir. Bu sistemler sürekli yeni düğümlerin mevcut olan ağa bağlanması nedeniyle “gelişen ağlar” olarak adlandırılır [16].

Gerçek ağların ikinci önemli özelliği, “Seçimli Bağlanma”dır. ER modelde iki düğümün bağlanma olasılığının rastgele ve tekdüze olduğu varsayılır. Aksine birçok gerçek ağ seçimli bağlanma gösterir. Örnek olarak yeni bir aktörün çok tanınan bir aktörle rol alma olasılığı, daha az tanınan aktörle birlikte rol alması olasılığından daha yüksektir. Benzer şekilde yeni oluşturulan bir web sayfasının bilinen ve çok ziyaret edilen web sayfalarına link verme olasılığı da yüksektir. Bu örnekler yeni eklenen bir düğümün var olan düğümlere bağlanma olasılığının tek düze olmadığını göstermektedir. Büyük bağlantı sayısına sahip bir düğüme, yeni bir düğümün bağlanma olasılığı daha yüksektir [16].

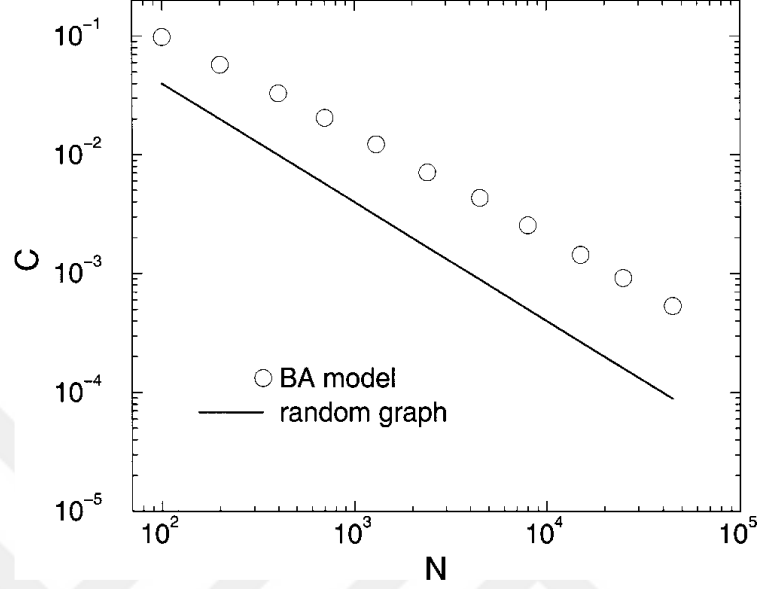
Barabási-Albert Modeli, büyüme ve seçimli bağlanmadan ilham almış ve ilk defa güç yasası (power-law) derece dağılımına sahip bir ağ, bu modelle gerçekleşmiştir. Barabási-Albert Modelinin algoritması şu şekildedir [1]:

1. Büyüme: Küçük sayıda düğümlerle (m_0) başlanır, her bir zaman diliminde $m \leq m_0$ olacak şekilde m farklı düğüm mevcut olan ağa eklenir.
2. Seçimli bağlanma: Yeni düğüm bağlanacağı düğümü seçeceği zaman, yeni düğümünün bağlanacağı düğüme bağlanma olasılığı Π , bağlanacağı düğüm i 'nin derecesi k_i 'ye bağlıdır ve eşitlik 2.12'deki ifade edilmektedir.

$$\Pi(k_i) = \frac{k_i}{\sum_j k_j} \quad (2.12)$$

Bu modelde t zaman adımından sonra, $N = t + m_0$ düğüm ve $L = mt$ bağlantı oluşur. Sayısal simülasyonlar, bu ağdaki düğümlerin derece dağılımı $\gamma_{BA} = 3$ katsayısına sahip bir power-law eğrisi ile uyumlu olduğunu göstermektedir. γ_{BA} katsayısı m değerinden bağımsızdır [1].

Bu model için ortalama yol uzunluklarının, rassal graftan daha düşük olduđu ve ortalama yol uzunluğunun yaklaşık olarak $\log(N)$ ile arttığı gösterilmiştir [1].



Şekil 2.10. Ortalama derecesi $\langle k \rangle = 4$ olan BA model ile oluşturulmuş ađın kümelenme katsayısı ile $C_{rand} \approx \langle k \rangle / N$ olan rassal grafin kümelenme katsayısının karşılaştırılması [1].

Şekil 2.10'daki grafikte görüldüğü gibi, ölçek bağımsız ağların kümelenme katsayıları, rassal grafin kümelenme katsayısından yaklaşık 5 kat kadar büyüktür ve bu fark düğüm sayısının artışıyla yavaş yavaş yükselmektedir. BA model kümelenme katsayısı ađın boyutu büyüdükçe azalmaktadır [1].

Tez Çalışması kapsamında, bu bölümde detaylarına değindiğimiz modeller ile birlikte bu modeller arasında tanımlanan ara modellerin analizlerine de yer verilmiştir. Bu analizler 4. Bölümde sunulacak olup ana ve ara modellerin web tabanlı olarak görselletirilmesine olanak tanıyan uygulamamız ise 5. Bölümde detaylanacaktır.

BÖLÜM 3

KOMPLEKS AĞ GÖRSELLEŞTİRME ARAÇLARI

Görselleştirme, insanların ağ yapıları ve verileri içerisinde ağ özelliklerini bulmaları noktasında algısal yeteneklerini artırmaya yardımcı olur. Görselleştirme oldukça zor ve araştırma gerektiren bir süreçtir. Kompleks ağ görselleştirme ve analiz araçları, teknik olarak doğru ve görselleştirmede etkileyici olmanın yanı sıra, kullanıcının araştırma sürecini iyileştirmek için gerçek zamanlı görselleştirme ve analiz işlemlerini yerine getirmelidir. Kullanıcı etkileşimli ağ görselleştirme araçları, kompleks ağlar üzerine yapılan araştırmalara başarıyla rehberlik etmektedir [17].

Kompleks ağları anlamak amacıyla, büyük ağların görselleştirilmesi için birçok başarılı yazılım geliştirilmiştir [20]. Bu yazılımların bazıları bütün kompleks ağları görselleştirilebilirken, bazıları ise sadece o yazılım için özel olanları görselleştirilebilmektedir. Bütün ağları görselleştirebilen uygulamalara Gephi, Cytoscape, Pajek gibi uygulamalar verilebilir [17, 18, 19]. Özel ağ uygulamalarında ise çoklu-omik verilerin görselleştirmesini ve analizini sağlayan MONGKIE yazılımı, sosyal ağlar için görselleştirme sağlayan ADraw uygulaması ve gen setlerinin analizini ve görselleştirmesini sağlayan Kiwi uygulaması verilebilir [20, 21, 22].

Kompleks ağ görselleştirme ve analiz araçları için bazı temel gereksinimler vardır: Bunlar yüksek kalitede yerleşim algoritmaları, veri filtreleme, kümelenme, istatistikler ve notlardır. Pratikte bu gereksinimler esnek, ölçeklenebilir ve kullanıcı dostu bir yazılıma dâhil edilmelidir [20]. Bu gereksinimleri sağlayan birçok masaüstü ve web tabanlı uygulama mevcuttur. Ancak bu görselleştirme ve analiz araçları incelendiğinde masaüstü uygulamaların daha ağırlıklı olduğu göze çarpmaktadır [17, 18, 19, 23]. Son yıllarda internet ve mobil teknolojilerindeki ilerlemelere paralel olarak JS tabanlı bazı uygulamalar da geliştirilmiştir [24, 25, 26].

3.1. KOMPLEKS AĞ GÖRSELLEŞTİRME UYGULAMALARI

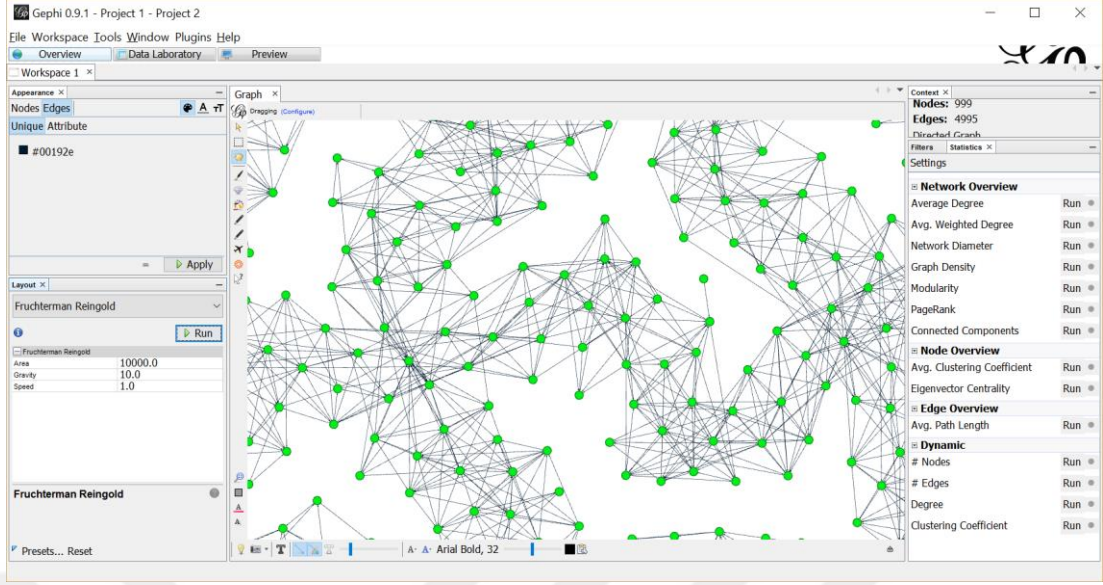
3.1.1. Gephi

Günümüz popüler masaüstü ağ görselleştirme araçlarından olan, Gephi açık kaynak kodlu bir ağ araştırma ve görselleştirme yazılımıdır. Gelişmiş modülleri ile her türlü ağı, analiz edebilme ve görselleştirme yeteneğine sahiptir. Gephi grafların görsel hale gerçek zamanlı dönüştürülmesi için özel bir 3B dönüştürme teknolojisi kullanır. Bu teknoloji ile bilgisayar işlemcisinden bağımsız olarak tıpkı video oyunları gibi bilgisayarın ekran kartını kullanır. Gephi, çok çekirdekli işlemcilerin avantajını kullanan, çoklu işlem modeli üzerine inşa edilmiştir ve 20 000 ve üzeri düğüme sahip büyük ağları analiz edebilmektedir [20].

Gephi uygulamasının dikkat çeken özelliklerinden biri, gerçek zamanlı olarak konfigüre edilebilen yerleşim algoritmalarına sahiptir. Bunlardan en önemlisi kuvvet yönelimli yerleşim algoritması olan ve Gephi proje ekibi tarafından geliştirilen Force Atlas 2'dir [20]. Kuvvet yönelimli yerleşim algoritmalarının iyi bilinen özelliklerinden esinlenerek geliştirilen bu algoritma ile düğüm konumları sürekli olarak yenilenebilmekte, 100 binden az düğüm sayısına sahip ağlar için iyi bir performans sağlamaktadır [27].

Gephi uygulaması, GEXF dosya uzantısı ile dış veri olarak bir ağ uygulama içerisine aktarabilmektedir. GEXF, kompleks ağları veri ve dinamiklerini birleştiren yapıları tanımlamak için kullanılan XML tabanlı bir dildir. Aynı zamanda görselleştirilen ağları SVG, PNG, PDF gibi formatlara dönüştürebilmektedir. Analiz noktasında da ağla ilgili birçok istatistik ve ölçümü yapabilmektedir. Bunlara örnek olarak kümelenme katsayısı, ortalama yol uzunluğu, derece dağılımı, aradalık merkeziliği, yakınlık merkeziliği, ağ genişliği (çapı) vs. verilebilir [21].

Şekil 3.1'de Gephi'de oluşturulmuş ve Fruchterman-Reingold yerleşim algoritması uygulanmış bir ağ örneği verilmiştir.



Şekil 3.1. Gephi’de oluşturulmuş bir ağ.

Gephi uygulaması ile iş dünyasından sosyal ağlara bir çok alanda kompleks ağlar üzerinde görselleştirme ve analiz çalışmaları yapılmıştır. Heymann ve Le Grand [28]’in iş zekası veri modelinin analiz edildiği ve araştırıldığı çalışması, Muhongya ve Maharaj [29]’ın çevrimiçi sosyal ağların görselleştirilmesi ve analiz edilmesi çalışması, Bravo, Del Valle ve Gavidia [30]’nın Katalan parlamenterlerin twitter ağı üzerinde liderlik ve politik kutuplaşmaları çok katmanlı olarak analiz edilmesi çalışması bu çalışmalara örnek olarak verilebilir.

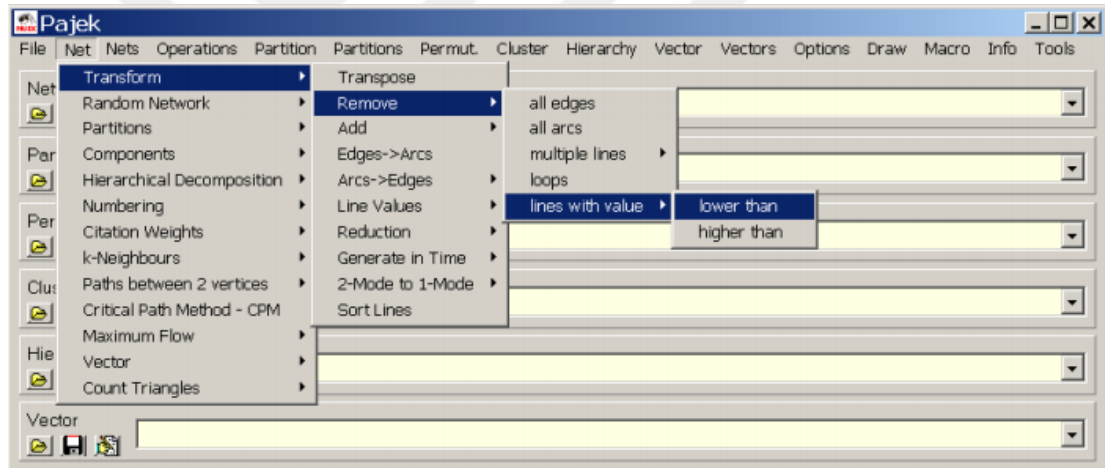
3.1.2. NetworkX

NetworkX, kompleks ağların oluşturulduğu, işlendiği ve ağ yapıları ve işlevlerinin incelenebildiği, Python dilinde yazılmış bir yazılım kütüphanesidir. NetworkX uygulaması ile standart veri formatları ile ağlar uygulamaya yüklenebilmekte ve kaydedilebilmektedir. Ayrıca bu uygulama ile klasik graflar ve popüler ağ modelleri olan ER, WS ve BA modellerinin örnekleri oluşturulabilmektedir. NetworkX Python programlama dili için geliştirilmiş bir kütüphane olduğu için kullanım için grafiksel bir ara yüze sahip değildir. Analiz ve görselleştirme için gerçek zamanlı kullanıcı etkileşimi noktasında zayıftır. Ancak yukarıda bahsedilen Gephi uygulamasının aksine bir milyondan fazla düğüm sayısına sahip ağları işleyebilmektedir [31]. NetworkX

GML, GEXF, JSON gibi çeşitli graf formatları da desteklemektedir. Python tabanlı bir kütüphane olmasından dolayı eğitim amaçlı olarak da kullanılabilir [32].

3.1.3. Pajek

Pajek büyük kompleks ağları görselleştirmek ve analiz etmek için geliştirilmiş, Windows tabanlı sistemlerde çalışan ücretsiz bir uygulamadır. 1996 yılından itibaren Delphi programlama dilinde geliştirilmeye başlanmıştır [33]. Uygulamanın dikkat çekici özelliklerinden biri çoklu-ilişkisel ağ tipini desteklemesidir. Uygulama bir milyonun üzerinde düğüme sahip olan ağları da işleyebilmektedir. Ayrıca popüler olan birçok ağ yerleşim algoritmasını desteklemektedir [31]. Şekil 3.2’de Pajek uygulama ara yüzü verilmiştir.



Şekil 3.2. Pajek uygulama ara yüzü [33].

3.1.4. IGraph

IGraph R, C/C++ ve Python’da programlanabilen, tanımlayıcı ağ analizi ve görselleştirmesi için çok yönlü seçenekler sunan bir kütüphanedir. Bu uygulama milyonu aşkın düğüm ve bağlantısı olan grafları işlemeye izin vermektedir. IGraph, Kamada Kawai ve Fruchterman Reingold gibi popüler yerleşim algoritmalarını desteklemektedir. Akhtar’ın bazı ağ araçları üzerinde yapmış olduğu karşılaştırma araştırmasına göre IGraph uygulaması diğer yazılımlara oranla daha başarılı çalışma zamanlarını ortaya koymuştur [31].

IGraph uygulama örneklerini incelediğimizde; analiz noktasında etkileyici özelliklere sahip olduğu görülmektedir. 3 milyon düğüm ve 15 milyon bağlantıya sahip olan çok büyük bir ağın ortalama yol uzunluğu hesabı için dağıtık bilgi işlemi yöntemi kullanılmıştır. Uygulama; ağ oluşturma, merkezilik ölçümlerini yapabilme, ortalama yol uzunluğu hesaplama, farklı dosya formatlarını destekleme gibi özelliklere sahiptir [23]. Uygulamanın ağlar üzerinde analiz ve görselleştirilme özellikleri, Ognyanova tarafından yapılan çalışmalarda detaylı olarak incelenmiştir [34, 35].

3.1.5. Diğer Masaüstü Görselleştirme Uygulamaları

Yukarıda bahsedilen görselleştirme ve analiz araçları dışında; açık kaynak kodlu olan, Gython (Python ve Jython eklentileri) adında kendine özgü sözdizimi ve operatörleri içeren dili olan, graf ve ağlar için bir analiz ve görselleştirme aracı olan GUESS; ağ verilerini görsel ve interaktif olarak analiz edilmesine yardımcı olan NetMiner; büyük ağlar içerisinde düğüm toplulukları gibi önemli özellikleri ve onların değişimini odaklanan ve görselleştirebilen MapEquation gibi uygulamalar vardır. Bunların dışında daha birçok görselleştirme ve analiz yapabilen uygulama mevcuttur [36, 37, 38].

3.1.6. Web Tabanlı Görselleştirme Araçları

Yukarıda belirtilen ağ analiz ve görselleştirme araçlarının dışında birçok web tabanlı ve masaüstü uygulama vardır [39]. Web tabanlı yazılımlar kurulum gereksinimi olmadan kolay adaptasyon sağlamaları açısından tercih sebebidirler. Bu web tabanlı yazılımların büyük bir çoğunluğu, coğrafik birimler, proje takımları ve iş birimleri arasındaki grupların işbirliğini görselleştirmeyi amaçlar. Bu yazılımlar nispeten küçük ve orta boyutlu ağlarda örgütsel hiyerarşi görselleştirmesi ve kalıplaşmış örgütsel sorunların çözümünde, kritik birimler arasındaki kopukluğu ortaya çıkarmada, birimleri ve ürünleri birbirine bağlayan kişileri ortaya çıkarmaya yardımcı olur. Bu türdeki bazı yazılımlar Maven7, Teamscom, KeyNetiQ, Sociliyzer, Polinode, SocioViz and Graph Commons'tır [39].

Shi tarafından temel modelleme ve analiz özellikleri olan bir web tabanlı model önerilmiştir. Geliştirilen uygulama temel ağ modellerinin 2B olarak japon dilinde oluşturulmasını sağlamaktadır. Yazılımda temel ağ modellerinin öğreniminin kolaylaşması amaçlanmıştır. Ayrıca geliştirilen yazılım, kullanıcı etkileşimli bir arayüze sahiptir. Bu yazılımda Java Applet teknolojisi kullanılmıştır [40].

Son yıllarda internet teknolojilerinin gelişmesiyle, Web, HTML, CSS ve JavaScript (JS) gibi standart teknolojileri kullanan kompleks ağ görselleştirme ve analiz uygulamaları geliştirilmiştir [26]. Bu geliştirilen uygulamalar farklı şekillerde karşımıza çıkmaktadır. Masaüstü uygulamaların web platformu için geliştirmiş oldukları program paketleri ya da bu uygulamalardan bağımsız olarak web teknolojileri kullanılarak geliştirilmiş sadece web tabanlı yazılım paketleri veya kütüphaneleri şeklindedir. Cytoscape Web, Cytoscape.js gibi uygulamalar Cytoscape uygulamasının web tabanlı parçalarıdır. Sigma.js gibi uygulamalar ise herhangi bir programa bağlı olmayan bir grafik kütüphanesidir [24, 25, 26].

Bağımsız olarak geliştirilen bu uygulamaların birçoğu bazı görselleştirme ve analiz araçlarına entegre olabilmektedir. İstemci-sunucu mimarisiyle çalışan linkurious uygulaması web tabanlı görselleştirme için sigma.js kütüphanesini kullanmaktadır [41].

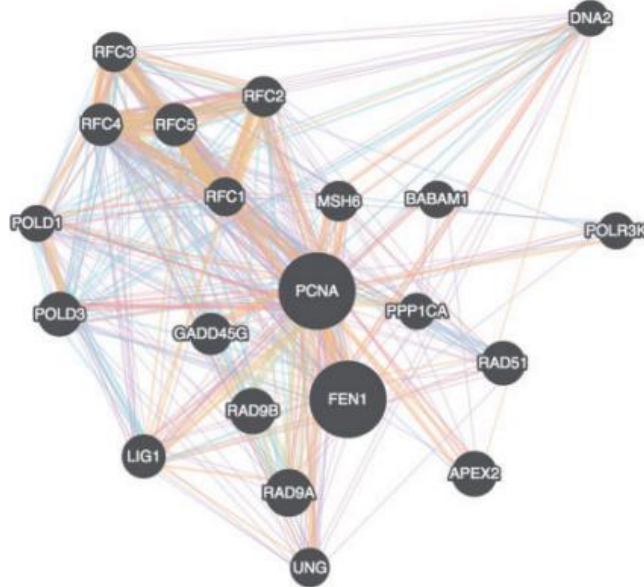
Görselleştirme özelliği olarak web tabanlı uygulamalar genellikle 2B’te görselleştirme yapmaktadır. Özel olarak geliştirilmiş bazı uygulamalar da 3B olarak görselleştirme yapılabilmektedir. Bu uygulamalara Piegza tarafından geliştirilen, grafların WebGL kullanarak oluşturulduğu graf görselleştirme projesi “graphit“ verilebilir [42].

Ağların görselleştirilmesi ve analiz edilmesinde kullanılan, web tabanlı yazılım ve kütüphane örnekleri aşağıda verilmiştir.

3.1.4.1. Cytoscape.js JS Kütüphanesi

Cytoscape.js yazılım geliştiricilerinin kendi veri modelleri ve web kullanıcı ara yüzlerine grafları entegre etmek için geliştirilmiş bir JS API'dir. Cytoscape.js biyolojik ağlar ve sosyal ağlar gibi birçok alanda kullanılabilir. Cytoscape.js'in mimarisi tek başına herhangi bir grafik arayüze ihtiyaç duymadan çalışmaya müsaittir. HTML5 Canvas kullanımı yoluyla bir görselleştirme bileşeni olarak da kullanılabilir. Uygulama hem istemci-sunucu tabanlı olarak hem de sadece istemci bilgisayarında çalışabilir [26].

Cytoscape.js'te ağlar, 2B olarak görselleştirilmektedir. Ayrıca uygulama ile çeşitli graf yapıları da çeşitli yerleşim özellikleri ile görselleştirebilir. JSON, PNG ve JPG gibi formatlarla ağları uygulama içine ve dışına aktarabilir. Performans açısından bakıldığında, ortalama bir bilgisayar donanımında yaklaşık 1 000 graf ögesini işleyebilir. Uygulama ile ilgili yapılan bazı çalışmalara Franz vd.'lerinin yapmış olduğu çalışmada yer verilmiştir [26]. Şekil 3.4'te Cytoscape.js'te oluşturulmuş bir ağ örneği olarak verilmiştir.



Şekil 3.3. Cytoscape.js tarafından görselleştirilen bir gen etkileşim ağı [26].

3.1.4.2. Cytoscape Web

Bir diđer web tabanlı uygulamada, Cytoscape Web'dir. Bu uygulama Cytoscape uygulamasından sonra modellenmiş web tabanlı bir interaktif ağ görselleştirme aracıdır. Cytoscape Web ağları 2B olarak görselleştirmektedir. Cytoscape Web uygulaması istemci tarafında çalışmaktadır. Ağların görselleştirilmesinde Flex/ActionScript'i kullanmasına rağmen, kullanıcı ile etkileşim ve ağın konfigüre edilmesi için JS API'leri kullanmaktadır. Bu JS API ile Flash kodların yeniden derlenmesine gerek kalmamaktadır. Uygulama ile HTML sayfasına oluşturulan ağ gömülmektedir. İnternet tarayıcılarda görüntülemek için Flash Player'a ihtiyaç olmaktadır [25].

Uygulama performans açısından küçük ve orta çaplı ağlarda başarılı çalışmaktadır. Ancak 2 000 öğeye sahip bir büyük ağa için kullanıcı etkileşimi etkisizdir. Ağları PNG ve PDF gibi formatlara dönüştürebilmektedir [25]. Flash bağımlılığı ise diđer tüm günümüz uygulamasında olduğu gibi erişilebilirliği olumsuz etkilemektedir.

3.1.4.3. Sigma.js JS Kütüphanesi

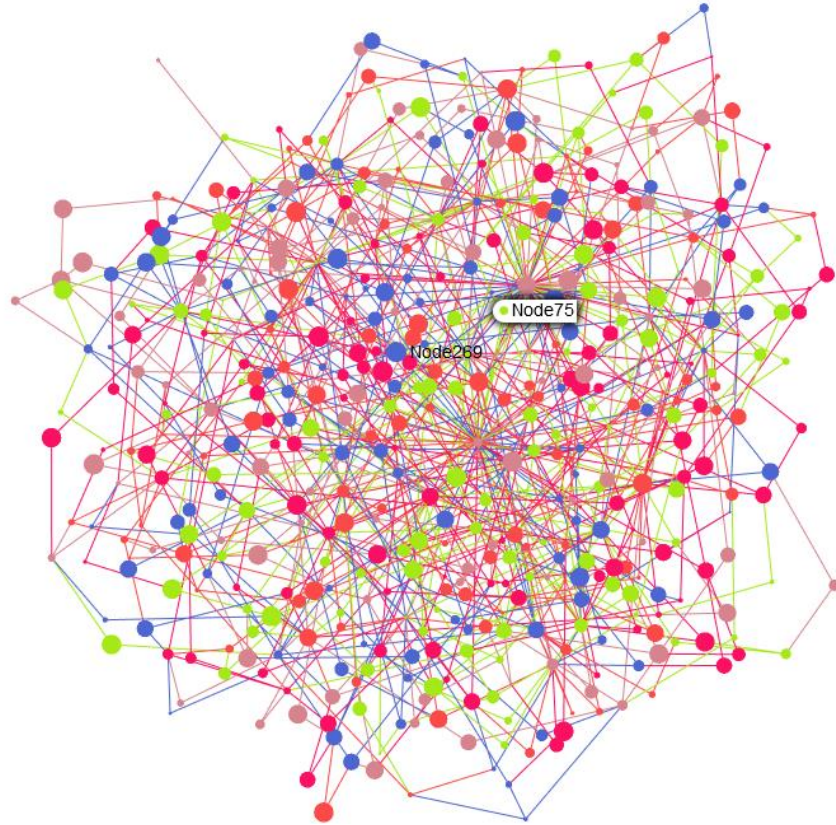
Sigma.js graf çizdirmek için tasarlanmış bir JS kütüphanesidir. Graf görselleştirmeleri ve için yüksek etkileşimli web uygulamaları geliştirmek için kullanılan ve konfigüre edilebilen bir araç olarak tasarlanmıştır. Sigma.js web sayfaları üzerinde ağların görselleştirilmesi için Canvas ve WebGL görüntü giydirici (renderer)'leri kullanmaktadır. Canvas desteđi olan modern tarayıcılarda çalışabildiđi gibi, WebGL desteđi olan modern tarayıcılarda daha hızlı çalışmaktadır [27].

Sigma.js, gephi'de oluşturulmuş GEXF dosya uzantısı ile kaydedilmiş ağları görselleştirebilmektedir. Ayrıca JSON formatında tanımlanmış ağları da görselleştirebilmektedir [43].

Sigma.js, 2B ve kullanıcı etkileşimine açık ağ görselleştirilmesine olanak sağlamaktadır. Ağ üzerinde düğümler seçilebilmekte ve oluşturulan graf

taşınabilmektedir. Gephi uygulamasında ağların yerleşimini sağlamak için kullanılan Force Atlas 2 gibi güçlü yerleşim algoritmalarını da desteklemektedir [27].

Sigma.js web tabanlı çalışan bazı görselleştirme ve analiz araçları için de ağların gösterimi noktasında yardımcı olmaktadır. Linkurious.js kütüphanesinde ağların çizdirilmesi için Sigma.js'ten yararlanılmıştır. Uygulama için performans kriterleri dikkate alındığında, Sigma.js, desteklediği 3 farklı görüntü giydirmesi (renderer) ile öne çıkmaktadır. WebGL renderer için maksimum 20 000 düğümü olan ağı görüntülemeyi ve etkileşmeyi desteklemektedir. Canvas renderer kullanıldığında ise maksimum 5 000 düğümü olan bir ağı görüntüleme ve etkileşmeyi desteklemektedir. Vektör-tabanlı renderer (SVG) için ise maksimum 1 000 düğümlü bir ağı görüntülemeyi ve etkileşmeyi desteklemektedir [41]. Şekil 3.4'de BA model oluşturulmuş bir ağın Sigma.js ile görselleştirilmesi verilmiştir.



Şekil 3.4. $t=500$, $m_0 = 2$, $m=2$ olan bir Barabási-Albert model ağının Sigma.js ile görselleştirilmesi.

3.1.5. Görselleştirme Araçlarına Genel Bakış

Yukarıda belirtilen görselleştirme ve analiz araçları incelendiğinde masaüstü araçların daha büyük ağları daha iyi görselleştirebildiği ve analiz edebildiği görülmüştür. Ancak internetin ve web teknolojilerinin gelişmesi ile web tabanlı yazılımlar da kullanımı artmaktadır. Genel itibariyle görselleştirme araçlarının birçoğu kullanıcı ile etkileşim sağlamaktadır. Bu araçlar aynı zamanda farklı dosya formatlarında hazırlanmış ağları görselleştirebilmekte ve analiz edebilmektedir. Web tabanlı olarak kullanılan araçlar ise genellikle bir kütüphane şeklinde diğer uygulamalara yardımcı olarak kullanılmaktadır.

Web tabanlı görselleştirme araçları genel olarak 2B görselleştirme yapmaktadır. Bunun yanında WebGL gibi kütüphaneleri kullanarak 3B görselleştirme yapan uygulamalarda mevcuttur. Tez çalışmamız kapsamında hazırlanmış olduğumuz web tabanlı uygulama, hem 3B desteği, hem zengin kullanıcı etkileşimi hem de bilinen ağ modelleri arasında tanımlanan geçişler ile literatürde benzeri olmayan bir uygulamadır.

BÖLÜM 4

KOMPLEKS AĞ MODEL DÖNÜŞÜM ÇALIŞMALARI

Tez çalışmamızın, görselleştirme dışındaki en önemli amaçlarından birisi, WS modelde olduğu gibi, bilinen modeller arasında geçişler tanımlamak ve bu geçişlerin ağ parametreleri üzerindeki etkilerini incelemektir. Bu amaçla özellikle Düzenli ağ modeli, ER model ile BA model arasında geçişler planlanmış, bu geçiş prosedürleri MATLAB yazılımı ile gerçekleştirilmiştir.

WS modeli, düzenli ağların rassallaştırılması esasına dayanmaktadır. Rassallaştırma prosedürü aslında linklerin yeniden bağlanması açısından en kolay gerçekleştirilebilen prosedürdür. Ancak rassal bir ağın düzenlenebilirliği nispeten daha komplike yordamlar gerektirmektedir. Çünkü koparılıp yeniden bağlanacak olan linkin, kaynak düğümünün kendisine yakın düğümlere bağlanması beklenir. Öte yandan yeniden bağlanma prosedüründe “seçimli bağlanma” olgusunun kullanılması da görece komplike yordamlar gerektirir. Tezimiz kapsamında gerçekleştirdiğimiz bu geçiş prosedürleri ve ağ parametrelerine etkileri bu bölümde incelenecektir.

Modeller arası gerçekleştirilen dönüşüm çalışmaları aşağıda verilmiştir:

1. ER Model - Düzenli Ağ Modeli
2. Düzenli Ağ modeli- BA Model
3. BA Model - Düzenli Ağ Modeli
4. ER Model - BA Model
5. BA Model - ER Model
6. ER Model - Düzenli Ağ Modeli ve BA Model

Yukarıda yer alan maddelerde sol taraf kompleks ağların başlangıçta oluşturulduğu modeli, sağ tarafta yer alan ise yeniden bağlantıların oluşturulmasıyla kompleks ağın dönüştüğü modeli ifade etmektedir. Uygulamada her bir dönüşüm denemesi için benzer düğüm sayısı ve bağlantı sayısı değerleri belirlenmiştir ($N \sim 1000$ ve $L \sim 5000$). Bu değerler Watts ve Strogatz'ın [8] ağ modelinde kullandığı rakamlar ile uyumludur. Bu şekilde modellerin birbiriyle kıyaslanması amaçlanmıştır. Yapılan dönüşüm çalışmalarından sonra ağlarda oluşan kümelenme katsayısı, derece dağılımı ve ortalama yol uzunlukları bulunmuş ve karşılaştırılmıştır.

Her bir deneme için gerçekleştirilen dönüşüm işlemlerinin ardından dönüşüm işlemlerinin ortak değerlendirilebilmesi için, kümelenme katsayısı ve ortalama yol uzunluklarının her bir dönüşüm işlemi almış oldukları maksimum ve minimum değerler göz önünde bulundurularak normalizasyon uygulanmıştır.

Çizelge 4.1. Model dönüşümleri sonucu oluşan minimum-maksimum kümelenme katsayısı (CC) ve ortalama yol uzunluğu(<d>) değerleri.

	CC _{Min}	CC _{Maks}	<d> _{Min}	<d> _{Maks}
1(ER-DA)	0,0113	0,2044	3,2579	12,6017
2(DA-BA)	0,0090	0,6667	3,2503	50,4505
3(BA-DA)	0,0409	0,2019	2,9754	12,1137
4(ER-BA)	0,0091	0,0123	3,2206	3,2622
5(BA-ER)	0,0097	0,5595	3,2579	30,3117
6(ER-DA_BA)	0,0178	0,0456	2,9565	3,1581
Min-Maks Değerler	0,009	0,6667	2,9565	50,4505

Her dönüşüme ait yerel maksimum ve minimumlara ek olarak, bütün denemelerin sonucunda kümelenme katsayısı ve ortalama yol uzunluğu değerlerinin global minimum ve maksimum değerleri bulunmuş, her bir dönüşüme ait kümelenme katsayısı ve ortalama yol uzunluğu değerleri bu global minimum ve maksimuma göre ayrıca normalize edilerek değişimleri çizdirilmiştir. Böylece her dönüşüme ait kümelenme katsayısı ve ortalama yol uzunluğu değerlerinin p_{rewire} yeniden bağlanma olasılığına göre dağılımları, hem yerel hem de global min/maks değerlerine göre ayrı ayrı normalize edilerek görselleştirilmiştir.

Yapılan çalışmalar sonucunda oluşan maksimum ve minimum değerleri içeren çizelge sırasıyla her bir deneme için Çizelge 4.1’de verilmiştir.

4.1. ER MODEL – DÜZENLİ AĞ MODELİ ARASI DÖNÜŞÜM

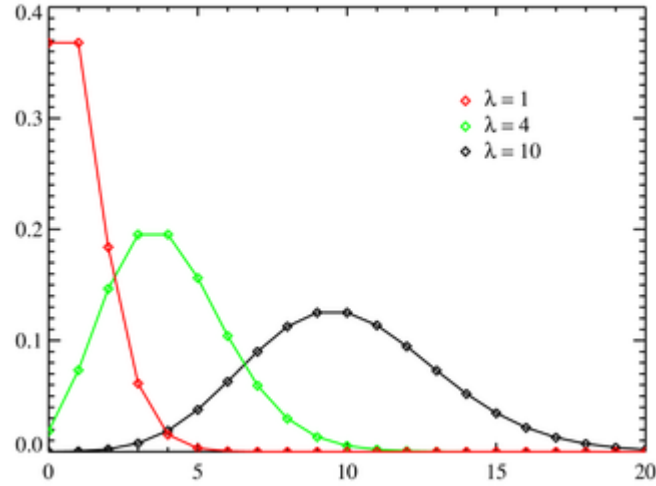
ER modelde ağ, N düğüm sayısı ve L bağlantısı sayısı ile ifade edilir. Ağda yer alan bağlantılar $[1-N]$ arası düğümlerin birbirine tekrar bağlanmadan ve kendi kendine bağlanmadan rastgele kaynak ve hedef düğümlerin seçilmesiyle oluşur [5]. Bu modelde düğümlerin ortalama yol uzunlukları gerçel ağlarla uyumlu fakat kümelenme katsayısı ve derece dağılımı uyumlu değildir. Bu bilgiden yola çıkılarak ER modelde oluşturulan ağ, düzenli ağ modeline dönüştürmeye çalışılarak gerçel ağ özellikleri ile kıyaslama yapılmıştır.

Ağlar arası dönüşüm işleminde, ER modelde oluşan ağda bağlantılar yeniden bağlanma işlemine tabi tutulmuştur. Yeniden bağlanma işlemi düğümler arasındaki tüm bağlantıların hedef düğümlerinin, tanımlanan p_{rewire} olasılığı ile yeniden bir düğüme bağlanması sağlanmıştır. N düğüm sayısı ve L bağlantı sayısına sahip bir ağ için beklenen yeniden bağlanma sayısı $(p_{rewire} \cdot L)$ ’dir. Ağı düzenli bir yapıya kavuşturmak için, yeniden bağlanma için yeniden seçilen bağlantıların kaynak düğümlerinin komşu düğümlerle bağlanması sağlanmıştır. Bunun için Poisson dağılımıyla kaynak düğümlere yakın rassal sayı üretilmiştir.

Poisson dağılımı olasılık fonksiyonu eşitlik 4.1’de verilmiştir.

$$f(k, \lambda) = \frac{\lambda^k e^{-\lambda}}{k!} \quad (4.1)$$

Poisson dağılımında λ ’nın sıfıra yakın olduğu değerlerde dağılımın ve üretilen rastgele sayıların düzensiz olmasından dolayı (Bkz. Şekil 4.1) λ değeri $N/2$ olarak belirlenmiş, daha sonra üretilen sayı, kaynak düğümü dağılımın merkezine oturtacak şekilde öteleme işlemine tabi tutulmuştur.



Şekil 4.1. Poisson dağılımı olasılık kütle fonksiyonu [44].

Ayrıca ağı daha düzenli bir hale getirmek için bir daraltma faktörü (narrow factor) kullanılarak kaynak düğüme daha yakın hedef düğümler seçilmesi amaçlanmıştır. Simülasyonlarımızda 0,5 olarak aldığımız narrow factor değeri, Poisson dağılımı ile kaynak düğüme yakın olarak seçilen yeni hedef düğümü 2 kat daha yakın hale getirmekte, yerleşmeyi desteklemektedir.

Gerçekleştirilen uygulamada 14 farklı p_{rewire} değeri için deneme yapılmıştır. Ve bu denemeler sonucunda kümelenme katsayısı (CC), ortalama yol uzunlukları ($\langle d \rangle$)nın değişimleri dikkate alınarak gerçel ağlara en yakın değişimin sağlandığı bölgede ağın derece dağılımı da çizdirilmiştir.

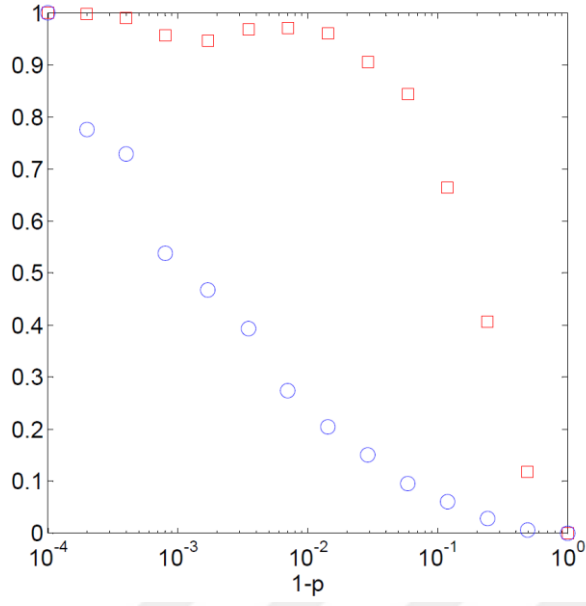
Önceki bölümde anlatılan min-maks normalizasyonları uygulanarak CC ve $\langle d \rangle$ parametrelerinin değişimlerinin Watts ve Strogatz'ın Şekil 2.9'da bulmuş oldukları değişim grafiğine benzer şekilde sunulması amaçlanmıştır. Bu sayede geçiş modellerinin karşılaştırılması daha kolay ve anlaşılır hale gelmiştir.

Dönüşüm sonucunda elde edilen sonuçlar ve grafikler Çizelge 4.2 ve Şekil 4.2 - 4.3'te verilmiştir.

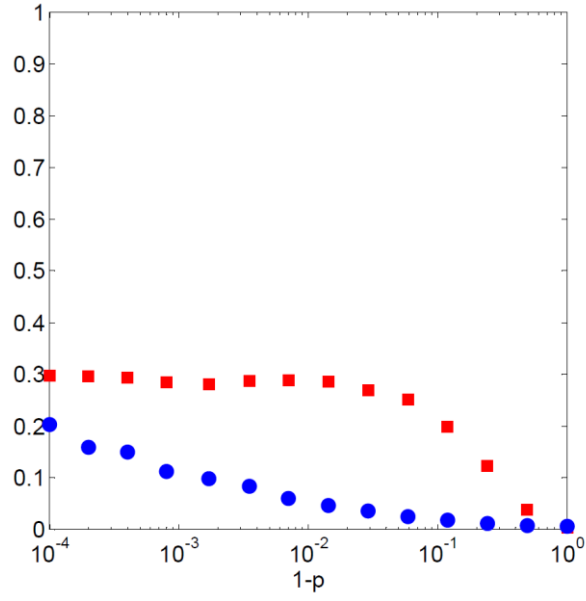
Çizelge 4.2. ER Model – Düzenli Ağ Modeli Arası Dönüşüm denemeleri sonucu elde edilen kümelenme katsayısı (CC), ortalama yol uzunluğu ($\langle d \rangle$) değerleri.

N	RN_{Prob}	$Narrow\ Factor$	P_{rewire}	CC	$\langle d \rangle$
1005	0,01	0,5	0,9999	0,2044	12,6017
1005	0,01	0,5	0,9998	0,2040	10,5078
1005	0,01	0,5	0,9996	0,2024	10,0671
1005	0,01	0,5	0,9992	0,1961	8,2843
1005	0,01	0,5	0,9983	0,1941	7,6244
1005	0,01	0,5	0,9965	0,1984	6,9322
1005	0,01	0,5	0,9930	0,1987	5,8164
1005	0,01	0,5	0,9857	0,1967	5,1683
1005	0,01	0,5	0,9711	0,1862	4,6642
1005	0,01	0,5	0,9412	0,1742	4,1485
1005	0,01	0,5	0,8806	0,1395	3,8244
1005	0,01	0,5	0,7576	0,0897	3,5213
1005	0,01	0,5	0,5076	0,0341	3,3145
1005	0,01	0,5	0	0,0113	3,2579

Dönüşüm çalışmasında önce p_{rewire} , [0-1] aralığında eşit aralıklarla artırılarak CC ve $\langle d \rangle$ parametrelerinin değişimleri incelenmiş; değişimin $p_{rewire} > 0,9$ bölgesinde yoğunlaştığı gözlenmiştir. Bu nedenle Watts ve Strogatz'ın da bu bölgeyi daha anlaşılır hale getirmek için uyguladığı logaritmik p_{rewire} eksenini uygulanmış, $p_{rewire} > 0,9$ bölgesini daha detaylı inceleyebilmek adına yatay ekseninde (1-p) değerleri baz alınmıştır. Yani bu grafikte aslında sağdan sola ilerledikçe p_{rewire} artmakta, ağ ER modelden düzenli modele doğru evrilmektedir.

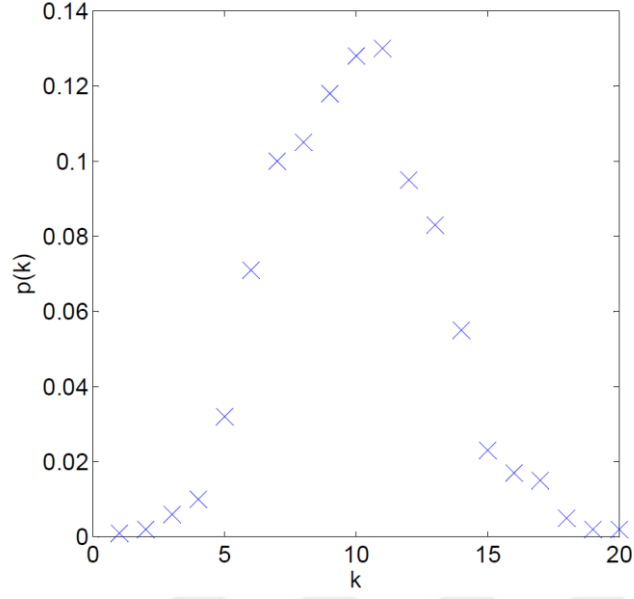


(a)



(b)

Şekil 4.2. a) Yerel minimum ve maksimum değerlere göre normalize edilmiş, kümelenme katsayısı (CC=□), ortalama yol uzunluğu (<d>=○) değişim grafiği, b) Global minimum ve maksimum değerlere göre normalize edilmiş kümelenme katsayısı (CC=■), ortalama yol uzunluğu (<d>=●)



Şekil 4.3. $p_{rewire}=0,9857$ olasılığında oluşan derece dağılım grafiği.

Yukarıda elde edilen sonuçlar ve grafikler incelendiğinde; ER modelde oluşturulan ağın p_{rewire} yeniden bağlanma olasılığı arttıkça daha düzenli hale geldiği, kümelenme katsayısında ve ortalama yol uzunluğunda artış olduğu gözlemlenmiştir. Bu artış, WS modeldekine benzer biçimde, CC ve $\langle d \rangle$ için farklı p_{rewire} değerlerinde gerçekleşmiş, yatay eksenin orta noktalarında small-world özelliği gösteren (düşük $\langle d \rangle$, yüksek CC) bir bölgenin varlığını göstermiştir. Aslında bu bölümde uygulamış olduğumuz dönüşüm prosedürü ters WS modeli olarak düşünülebilir. Bu bakış açısıyla WS model çıktılarının tersten ispatı niteliği de taşımaktadır.

Dönüştürülmüş ağın small-world özellikleri gösterdiği $p_{rewire} = 0,9857$ değeri için ağın derece dağılımı Şekil 4.3'te çizdirilmiş olup başlangıçtaki gibi Poisson dağılımı gözlenmekte, düğüm derecelerinin hala rassal özellik sergilediği anlaşılmaktadır. Dolayısıyla bu dönüşüm sonucunda CC ve $\langle d \rangle$ için gerçel ağlarda olduğu gibi bir small-world bölgesine ulaşılmış olsa da derece dağılımında gerçel ağ benzerliği yakalanamamıştır.

4.2. DÜZENLİ AĞ MODELİ – BA MODEL ARASI DÖNÜŞÜM

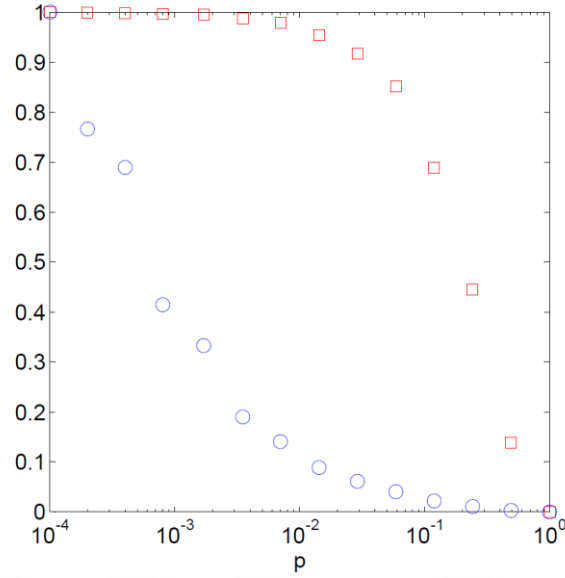
Düzenli ağ modeli Bölüm 2.3.2’de ifade edilmişti. Bu dönüşüm çalışmasında düzenli ağ modeliyle oluşturulan kompleks ağ üzerindeki her bir bağlantının bir p_{rewire} olasılığıyla hedef düğümlerinin yeniden bağlanması işlemi gerçekleştirilmiştir. Hedef düğüm seçim işlemi için Bölüm 2.3.4’de belirtilen “seçimli bağlanma” prosedürü kullanılmıştır. Bu prosedürün işlemsel olarak gerçekleştirilmesi, derecesi fazla olan düğümlere daha yüksek ihtimalle bağlanmanın sağlanmasını gerektirmektedir. Bu da link dizisindeki kaynak-hedef düğümlerinden herhangi birinin rassal seçilmesi ile mümkün olmaktadır. Çünkü her bir düğüm ne kadar çok bağlantıya sahipse, link dizisinde o kadar fazla tekrar edilmektedir. Dolayısıyla link dizisinden rassal bir düğüm seçmek, seçimli bağlanma gereksinimlerinin kolay ve eksiksiz olarak karşılanmaktadır.

Yapılan dönüşüm çalışması sonucunda elde edilen sonuçlar ve grafikler Çizelge 4.3 ve Şekil 4.4 - 4.5’te gösterilmiştir.

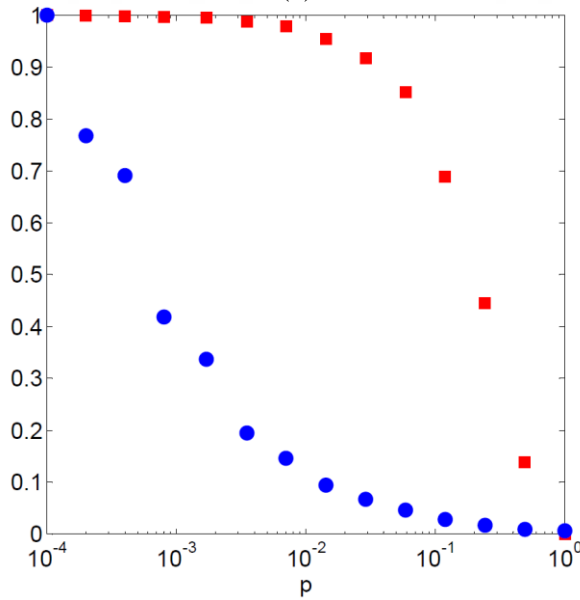
Çizelge 4.3. Düzenli Ağ Modeli- BA model arası dönüşüm denemeleri sonucu elde edilen kümelenme katsayısı (CC), ortalama yol uzunluğu ($\langle d \rangle$) değerleri.

N	<i>Komşu Sayısı</i>	P_{rewire}	CC	$\langle d \rangle$
1000	10	0,0001	0,6667	50,4505
1000	10	0,0002	0,6659	39,4199
1000	10	0,0004	0,6654	35,7793
1000	10	0,0008	0,6645	22,8244
1000	10	0,0017	0,6633	18,9649
1000	10	0,0035	0,6584	12,2150
1000	10	0,0070	0,6526	9,888
1000	10	0,0143	0,6366	7,4382
1000	10	0,0289	0,6124	6,1401
1000	10	0,0588	0,5691	5,1532
1000	10	0,1194	0,4622	4,2861
1000	10	0,2424	0,3019	3,7521
1000	10	0,4924	0,0996	3,3860
1000	10	1,0000	0,0090	3,2503

Bu dönüşüm modelinde başlangıçta CC ve $\langle d \rangle$ yüksek olduğundan, CC hala yüksek iken $\langle d \rangle$ 'nin gerçel ağlardaki gibi düştüğü bir bölgenin bulunması hedeflenmektedir. Bu bölgenin $p_{rewire} \approx [0.01 - 0.1]$ civarında var olduğu gözlemlendiğinden, bu bölgenin daha detaylı incelenebilmesi için yatay eksen p_{rewire} 'a göre logaritmik olarak çizdirilmiş, logaritmik eş-aralıklı p_{rewire} değerleri baz alınmıştır.

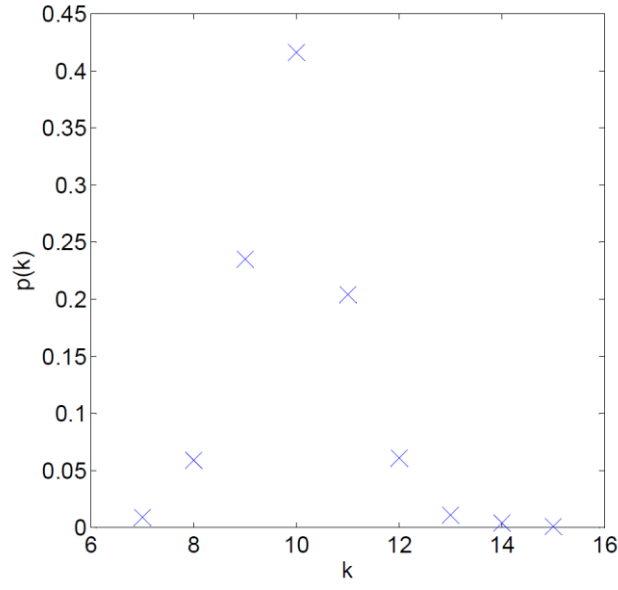


(a)

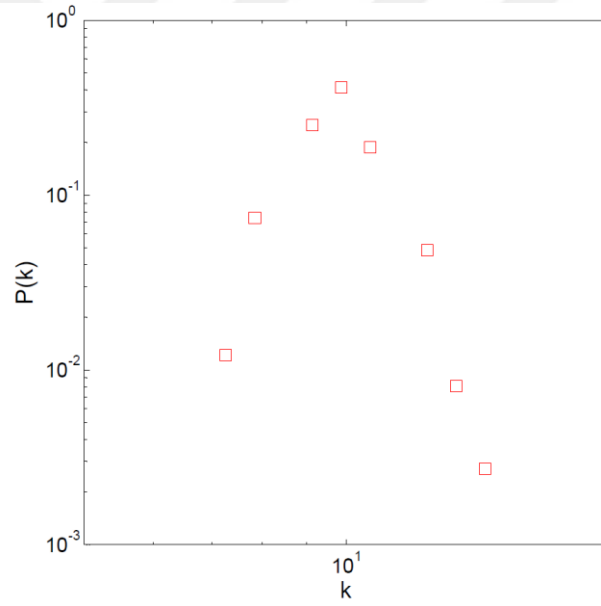


(b)

Şekil 4.4. a) Yerel minimum ve maksimum değerler için, kümelenme katsayısı (CC= \square), ortalama yol uzunluğu ($\langle d \rangle$ = \circ) değişim grafiği, b) Global minimum ve maksimum değerler için kümelenme katsayısı (CC= \blacksquare), ortalama yol uzunluğu ($\langle d \rangle$ = \bullet)



(a)



(b)

Şekil 4.5. $p_{rewire}=0,1194$ olasılıkta oluşan derece dağılım grafikleri a) Normal eksen dağılım grafiği, b) log-log ekseninde oluşan derece dağılım grafiği.

Bu dönüşüm prosedürü sonucunda, düzenli ağ yapısı, p_{rewire} arttıkça BA modele yaklaşmakta, CC'nin yüksek olduğu ve $\langle d \rangle$ 'nin düşük olduğu bir "small-world" bölgesine ulaşmaktadır ($p_{rewire}, 0,01 - 0,1$ civarında). Bu iki parametrenin en ideal değerleri $p_{rewire} \approx 0,01$ civarında gözlemlenmekte, fakat bu bölgede derece dağılımı hala düzenli ağla çok benzerlik taşımakta olduğundan CC, $\langle d \rangle$ ve derece dağılımının

gerçel ağlara en yakın olduğunu gözlemlediğimiz $p_{rewire} = 0,1194$ değeri için derece dağılımı çizdirilerek Şekil 4.5'te sunulmuştur. Bu dönüşüm modelinde gerçel ağların "small-world" özelliğine ulaşılmış, derece dağılımında ise power-law kuyruk bölgesi edilebilmiştir. Ancak gerçel ağlarda da gözlenebilen düşük k değeri (2,3,4) satürasyon bölgesi, bu bölgede nispeten yüksek k değerleri için ($k < 10$) gözlenmekte ve bu bölgenin gereğinden başka olması gerçel ağlar ile önemli bir farklılık arz etmektedir. Bununla birlikte düzenli ağın BA'ya dönüştürülmesi ile BA modelde olmayan görece yüksek kümelenme değerine ulaşılması önemli bir kazanımdır.

4.3. BA MODEL – DÜZENLİ AĞ MODELİ ARASI DÖNÜŞÜM

BA model Bölüm 2.3.4'de ifade edilmiştir. Bu model özellikleri itibariyle gerçel ağların kümelenme katsayısı dışında kalan diğer özelliklerini karşılamaktadır. Düzenli ağ modeline dönüşümde gerçel ağların yüksek kümelenme özelliğinin diğer özellikler ile aynı anda elde edilmesi amaçlanmaktadır.

Öncelikle oluşturduğumuz BA model için, başlangıçta $m_0 = 5$ düğüm, $L_{başlangıç} = 5$ bağlantı ve her bir yeni düğüm eklemede yeni eklenen düğümün bağlantı sayısı $L_{yenidüğüm} = 5$ 'dir. BA modelle oluşturulan kompleks ağın, düzenli ağ modeline dönüşümü için her bir bağlantının hedef düğümleri p_{rewire} olasılığıyla yeniden seçilerek bağlanmıştır. Yeni hedef düğümler seçilirken Poisson dağılımıyla rastgele sayı üretiminden yararlanılmıştır. Bu yerleştirme ve düzenleştirme prosedürü Bölüm 4.1'deki ile aynıdır.

Yeniden bağlanma işlemlerinin ardından p_{rewire} olasılığı arttıkça kümelenme katsayısının arttığı ve ortalama yol uzunluklarının artış gösterdiği gözlemlenmiştir.

Yapılan çalışma sonucunda elde edilen sonuçlar Çizelge 4.4, Şekil 4.6 - 4.7'de verilmiştir.

Çizelge 4.4. BA Model - Düzenli Ağ Modeli arası dönüşüm denemeleri sonucu elde edilen kümelenme katsayısı (CC), ortalama yol uzunluğu ($\langle d \rangle$) değerleri.

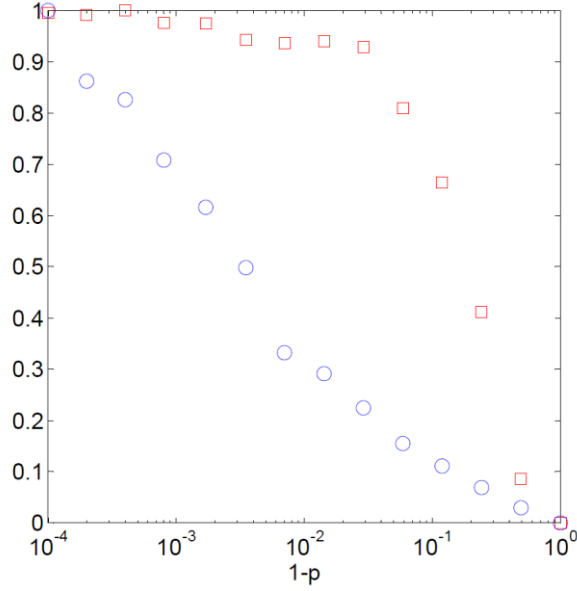
N	m_0	$L_{başlangıç}$	$L_{yenidüğüml}$	<i>Narrow Factor</i>	P_{rewire}	CC	$\langle d \rangle$
1005	5	5	5	0,5	0,9999	0,2019	12,1137
1005	5	5	5	0,5	0,9998	0,2012	10,8523
1005	5	5	5	0,5	0,9996	0,2026	10,5223
1005	5	5	5	0,5	0,9992	0,1987	9,4448
1005	5	5	5	0,5	0,9983	0,1985	8,6058
1005	5	5	5	0,5	0,9965	0,1933	7,5288
1005	5	5	5	0,5	0,9930	0,1923	6,0093
1005	5	5	5	0,5	0,9857	0,1929	5,6379
1005	5	5	5	0,5	0,9711	0,1911	5,0262
1005	5	5	5	0,5	0,9412	0,1718	4,3919
1005	5	5	5	0,5	0,8806	0,1483	3,9886
1005	5	5	5	0,5	0,7576	0,1075	3,6066
1005	5	5	5	0,5	0,5076	0,0548	3,2461
1005	5	5	5	0,5	0	0,0409	2,9754

Bu bölümde gerçekleştirdiğimiz BA-DA dönüşümü, önceki bölümdeki dönüşümün tersi niteliğindedir. Dolayısıyla CC ve $\langle d \rangle$ parametrelerinin değişim karakteristikleri tersten benzerlik göstermektedir.

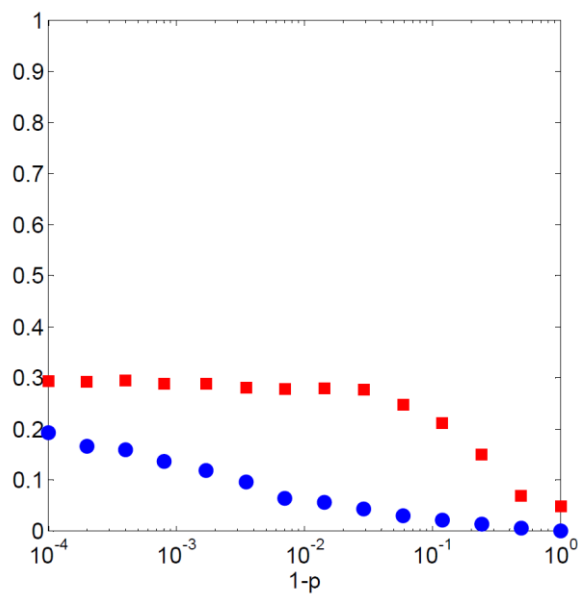
Yüksek CC ve düşük $\langle d \rangle$ değerlerinin elde edildiği p_{rewire} aralığı $p_{rewire} > 0.9$ civarına tekabül ettiği için bu bölgenin daha anlaşılır hale getirilmesi amacı ile $(1-p)$ 'ye göre eşit aralıklı logaritmik ölçekleme tercih edilmiştir.

Şekil 4.6'da görüldüğü gibi ağın small-world özelliği sergilediği bir yüksek CC, düşük $\langle d \rangle$ aralığı mevcuttur. Bununla birlikte gerçel ağlara yakın bir derece dağılımı elde etmek için bu "small-world" bölgesinin mümkün olduğu kadar düşük p_{rewire} bölgesine yaklaşması gerekmektedir. CC, $\langle d \rangle$ ve derece dağılımı açısından gerçel ağlara en yakın bulduğumuz p_{rewire} değeri 0,9930'dur. Nispeten yüksek kümelenme, düşük düğümler arası uzaklığa sahip bu değer için derece dağılımı Şekil 4.6'da sunulmuştur. Derece dağılımı sınırlı bir düşük k saturasyon bölgesinin ardından uzun bir power-law uyumlu kuyruk sergilemekte, gerçel ağlara son derece yakın bir

karakteristik sunmaktadır. Dolayısıyla BA-DA model dönüşümü, BA-DA model dönüşümüne göre daha başarılı bir gerçel ağ benzetimi sağlamaktadır.

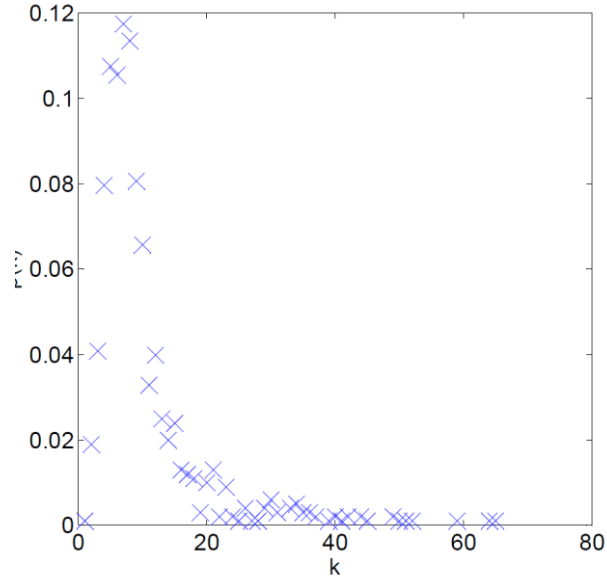


(a)

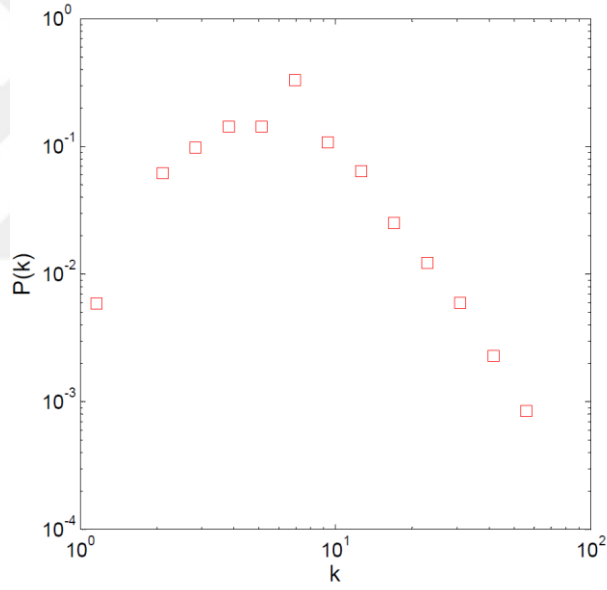


(b)

Şekil 4.6. a) Yerel minimum ve maksimum değerlere göre normalize edilmiş, kümelenme katsayısı (CC=□), ortalama yol uzunluğu (<d>=○) değişim grafiği, b) Global minimum ve maksimum değerlere göre normalize edilmiş kümelenme katsayısı (CC=■), ortalama yol uzunluğu (<d>=●)



(a)



(b)

Şekil 4.7. $p_{rewire}=0,9930$ olasılıkla oluşan derece dağılım grafikleri a) Normal eksen dağılım grafiği, b) log-log eksenle oluşan derece dağılım grafiği.

4.4. ER MODEL – BA MODEL ARASI DÖNÜŞÜM

ER model Bölüm 2.3.1’de ifade edilmiştir. Bu dönüşüm çalışmasında başlangıç ağında yer alan bağlantıların p_{rewire} olasılığı ile yeniden bağlanması amaçlanmıştır, yeniden

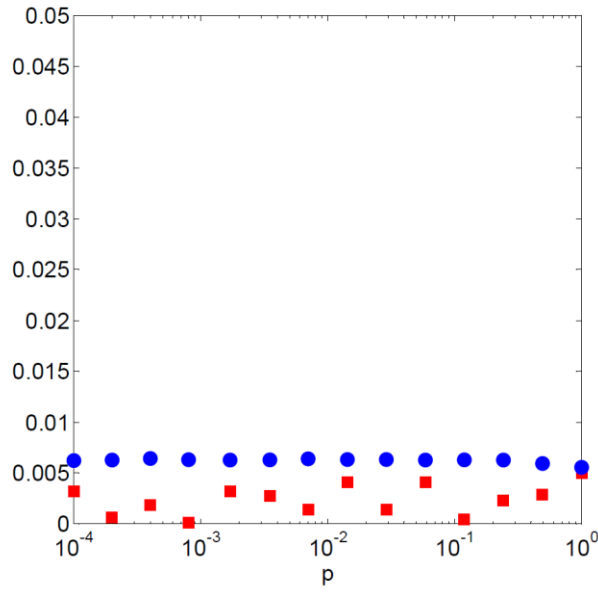
bağlanma işleminde BA modelin “seçimli bağlanma” özelliği kullanılmıştır. Yeniden bağlanma olasılığı p_{rewire} arttıkça rassal ağ özelliği BA modele yaklaşmaktadır.

Yapılan dönüşüm çalışması sonucunda elde edilen sonuçlar ve grafikler Çizelge 4.5 ve Şekil 4.8 - 4.9 - 4.10’da gösterilmiştir.

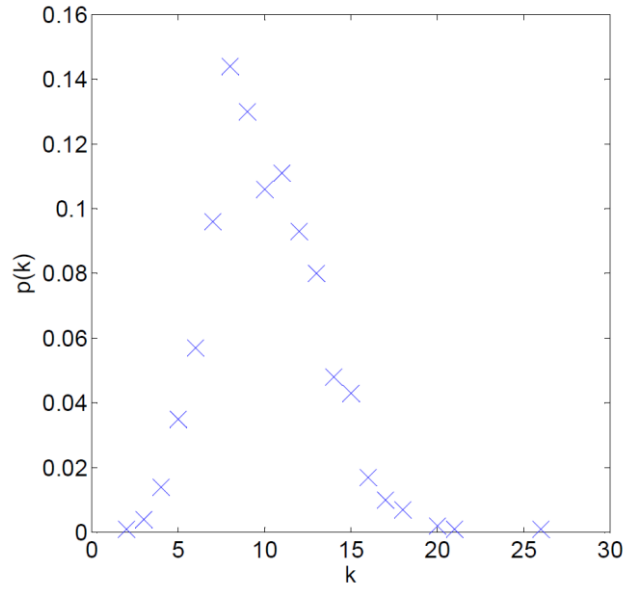
Çizelge 4.5. ER Model - BA Model arası dönüşüm denemeleri sonucu elde edilen kümelenme katsayısı(CC), ortalama yol uzunluğu(<d>) değerleri.

N	RN_{prob}	P_{rewire}	CC	$\langle d \rangle$
1000	0,01	0,0001	0,0111	3,2524
1000	0,01	0,0002	0,0094	3,2548
1000	0,01	0,0004	0,0102	3,2622
1000	0,01	0,0008	0,0091	3,2564
1000	0,01	0,0017	0,0111	3,2550
1000	0,01	0,0035	0,0108	3,2558
1000	0,01	0,0070	0,0099	3,2606
1000	0,01	0,0143	0,0117	3,2574
1000	0,01	0,0289	0,0099	3,2572
1000	0,01	0,0588	0,0117	3,2551
1000	0,01	0,1194	0,0093	3,2552
1000	0,01	0,2424	0,0105	3,2544
1000	0,01	0,4924	0,0109	3,2386
1000	0,01	1,0000	0,0123	3,2206

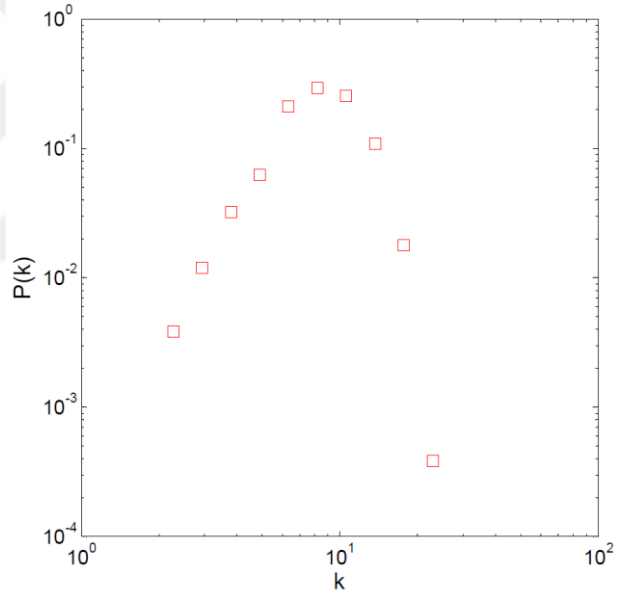
ER-BA modelleri arası tanımladığımız dönüşüm sonucunda CC ve $\langle d \rangle$ değerlerinin fazla değişim göstermediği görülmektedir. Bunun nedeni her iki modelin de zaten düşük CC ve $\langle d \rangle$ karakteristiğine sahip olmasıdır. Dolayısıyla bu dönüşüm, gerçel ağların yüksek CC değerlerini doğuracak bir potansiyele sahip değildir. Ancak p_{rewire} 1'e yaklaştıkça derece dağılımı Poisson'dan power-law'a doğru sınırlı bir değişim göstermektedir. Sonuç olarak bu dönüşüm modeli gerçel ağların sadece düşük $\langle d \rangle$ özelliğini karşılamakta, CC ve derece dağılımı açısından istenen değişimi sağlayamamaktadır.



Şekil 4.8. Genel minimum ve maksimum değerler için kümelenme katsayısı (CC=■), ortalama yol uzunluğu ($\langle d \rangle$ =●)

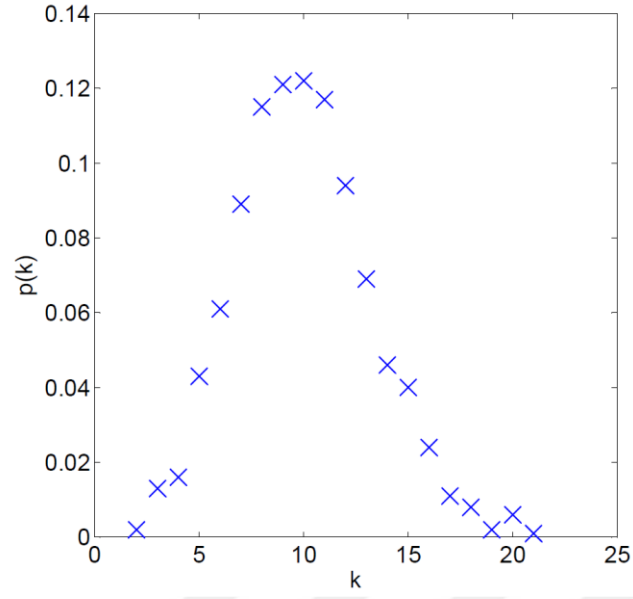


(a)

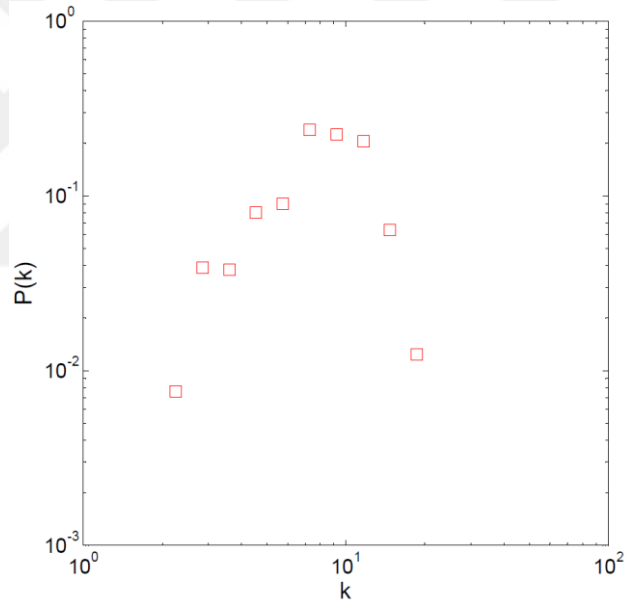


(b)

Şekil 4.9. $p_{\text{rewire}}=0,0588$ olasılıkta oluşan derece dağılım grafikleri a) Normal eksen dağılım grafiği, b) log-log ekseninde oluşan derece dağılım grafiği.



(a)



(b)

Şekil 4.10. $p_{rewire}=1,0$ olasılıkta oluşan derece dağılım grafikleri a) Normal eksen dağılım grafiği, b) log-log eksenle oluşan derece dağılım grafiği.

4.5. BA MODEL - ER MODEL ARASI DÖNÜŞÜM

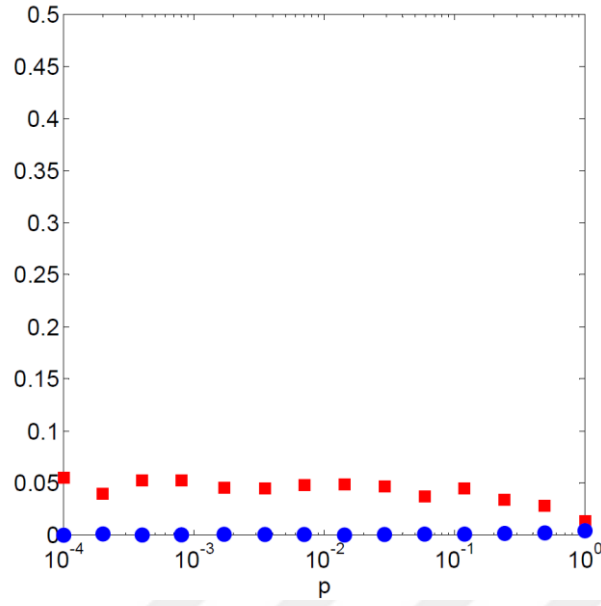
BA model Bölüm 2.3.4 ve ER model de Bölüm 2.3.1’de ifade edilmiştir. Bu dönüşüm çalışmasında BA modelde oluşturulan ağıın bağlantılarının hedef düğümlerinin p_{rewire} olasılığı ile yeniden bağlanması amaçlanmıştır. Yeniden bağlanma işlemi için hedef

düğüm ER model de olduğu gibi $[1-N]$ arasında rastgele seçilmiş ve yeni bağlantılar oluşturulmuştur.

Yeniden bağlanma olasılığı arttıkça başlangıçtaki ağ, ER model ağ özellikleri göstermektedir. p_{rewire} değerine bağlı olarak CC ve $\langle d \rangle$ değerlerinin değişimi Çizelge 4.6 ve Şekil 4.11- 4.12’de verilmiştir.

Çizelge 4.6. BA Model - ER Model arası dönüşüm denemeleri sonucu elde edilen kümelene katsayısı (CC), ortalama yol uzunluğu ($\langle d \rangle$) değerleri.

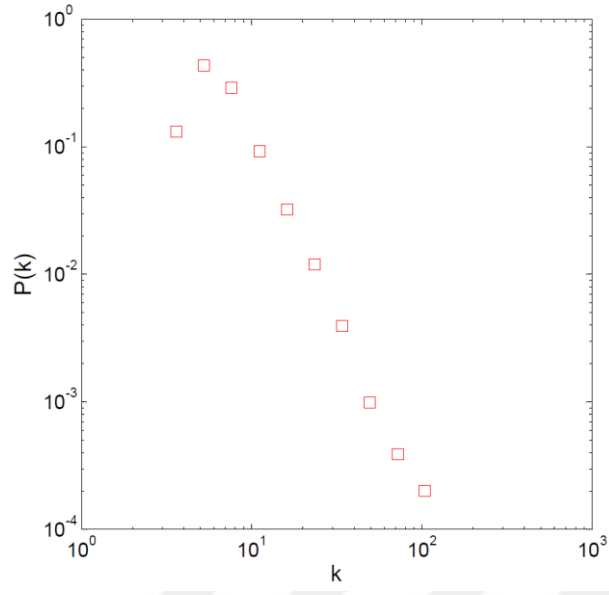
N	m_0	$L_{başlangıç}$	$L_{yenidüğüm}$	P_{rewire}	CC	$\langle d \rangle$
1005	5	5	5	0,0001	0,0453	2,9565
1005	5	5	5	0,0002	0,0352	3,0107
1005	5	5	5	0,0004	0,0435	2,9626
1005	5	5	5	0,0008	0,0436	2,9725
1005	5	5	5	0,0017	0,0388	2,9930
1005	5	5	5	0,0035	0,0383	2,9864
1005	5	5	5	0,0070	0,0406	2,9846
1005	5	5	5	0,0143	0,0410	2,9623
1005	5	5	5	0,0289	0,0396	2,9863
1005	5	5	5	0,0588	0,0336	2,9976
1005	5	5	5	0,1194	0,0383	2,9990
1005	5	5	5	0,2424	0,0314	3,0304
1005	5	5	5	0,4924	0,0273	3,0582
1005	5	5	5	1,0000	0,0178	3,1581



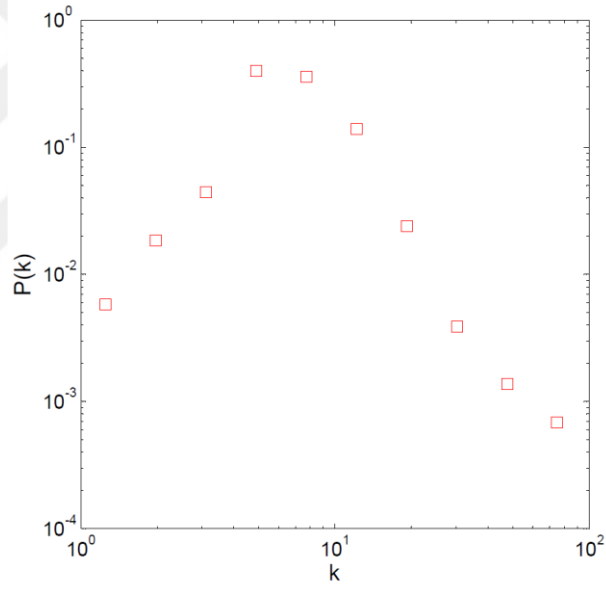
Şekil 4.11. Global minimum ve maksimum değerlere göre normalize edilmiş, kümelenme katsayısı (CC=■), ortalama yol uzunluğu (<d>=●)

Bu bölümde uygulanan dönüşüm prosedürü, önceki bölümdeki ER-BA model dönüşümünün tersi niteliğindedir. Dolayısıyla her iki modelde düşük olan <d> ve CC parametreleri önemli değişim göstermemiştir. Ancak parametreler ER-BA model dönüşümündeki kadar da durağan kalmamış, ER modele yaklaştıkça 0,04 civarından önceki dönüşümdeki ~0,01 değerine yakınsama göstermiştir.

Artan p_{rewire} değeri ile ER modele yakınlaştıkça derece dağılımının saf power-law uyumundan uzaklaştığı, Şekil 4.11'de gözlenmektedir. Şekil 4.10 ve 4.11 birlikte incelendiğinde bu dönüşümün de gerçel ağ benzetiminde başarılı olmadığı anlaşılmaktadır. Ancak önemli bir çıktı olarak, düşük p_{rewire} (0,1194) değeri için BA modelin saf power-law derece dağılımına, bir düşük k satürasyon bölgesinin eklendiği, bununla gerçel ağ dağılımına benzerliği artırdığı görülmektedir (Şekil 4.12 a).



(a)



(b)

Şekil 4.12. (a) $p_{\text{rewire}}=0,1194$ olasılıkta oluşan log-log derece dağılım grafiği, (b) $p_{\text{rewire}}=0,4924$ olasılıkta oluşan log-log derece dağılım grafiği.

4.6. ER MODEL – DÜZENLİ AĞ VE BA MODELİ ARASI DÖNÜŞÜM

ER Model – Düzenli Ağ ve BA Modeli arası yapılan dönüşüm çalışmasında, diğer modeller arası gerçekleştirilen dönüşüm işlemlerine nispeten benzeyen, ancak biraz daha karmaşık bir bağlanma işlemi uygulanmıştır. Yeniden bağlanma işleminde kaynak düğüm yeni bir düğüme bağlanırken hem fiziksel yakınlık hem de derece değerleri dikkate alınmıştır. ER model ağı daha düzenli hale gelmesiyle komşuluk artarken aynı zamanda tanımlanan seçimli yeniden bağlanma prosedürü ile “zengin” komşulara bağlanma teşvik edilmektedir.

Yeniden bağlanma işlemi için, Bölüm 4.1’de kullanılan Poisson dağılımlı komşu seçim prosedürü esas alınmıştır. Ancak yeni belirlenen hedef düğümler sıralı düğüm numaralarından seçilmemiş, sıralı ve link sayısınca tekrarlayan düğüm numaralarından seçilmiştir.

Tekrarlama sayısı düğümlerin derecelerinden oluşturulmuştur. Bunun sonucunda yeniden bağlanma işleminde, hem yeni seçilen hedef düğümler kaynak düğüme daha yakın olanlardan seçilmiş hem de “seçimli bağlanma” uygulanmıştır. Bununla birlikte yeniden bağlanma olasılığı p_{rewire} ile daha düzenli ve daha seçimli bağlanma içeren bir ağ elde edilmiştir.

Şekil 4.13’te düğüm sayısı 9 ve dereceler toplamı 26 olan bir ağ için, düğümler numaralandırılmış ve her bir düğüm, derecesi kadar tekrar edilecek şekilde bir dizi tanımlanmıştır. Yeniden bağlanma prosedürü başladığında; p_{rewire} olasılığına bağlı olarak her bir bağlantının kaynak düğümün, düğüm tekrar dizisindeki başlangıç ve bitiş indisleri bulunur. Ardından Poisson dağılımına göre kaynak düğüme ait indis merkez kabul edilerek bir rassal sayı üretilir. Poisson dağılımı yüksek λ için simetrik olduğundan rassal sayı, sıralanmış düğüm tekrar dizisinde kaynak düğüm indisinden küçük ya da büyük olabilir. Kaynak düğüm indisinden küçük rassal sayı için sola, büyük rassal sayı için ise sağa doğru ilerleyerek bulunan yeni indis, içerisinde hangi düğüm numarasını taşıyorsa o düğüm yeni hedef düğüm olarak kabul edilir. Böylece hem koşulunun hem de düğüm tekrar sayısının (derece) etkili olduğu bir yeniden bağlanma prosedürü gerçekleştirilmiş olur.

Şekil 4.13'te görülen, kaynak düğümü 7 olan bir bağlantının yeniden bir hedef düğüme bağlanmasını ele alalım. 7 numaralı düğüm bağlantı dizisinde 5 kez tekrar etmiş ve düğüm tekrar dizisinde 15-19 dizi indisleri arasındadır. Poisson dağılımıyla rassal bir sayı olan 9'un üretildiğini farz edelim. Üretilen bu sayıdan Poisson merkez değeri olan $\lambda = 26/2 = 13$ değerini çıkaralım: $9-13=-4$. Bu fark değeri negatif olduğunda dizinin soluna, pozitif olduğunda ise dizinin sağına doğru ilerleyerek yeni düğüm bulunur. Sonuçta 7 numaralı düğümün başlangıç dizi indisi 15'in 4 indis değeri kadar solunda yer alan 11. indisteki dizi değeri 5 numaralı düğüm hedef düğüm olarak belirlenir. Üretilen rassal sayı -4 değil +4 çıksa idi, 7. Düğümün tekrar edileceği son indis olan 19'a +4 rakamı eklenerek $19+4=23$. sırada bulunan 8 numaralı düğüm hedef düğüm olacaktı.

Sıralanmış Düğüm Tekrar Dizisi	1	1	1	2	3	3	3	4	4	5	5	5	6	6	7	7	7	7	7	8	8	8	8	8	9	9
Düğüm Sayısı N=9												↑														
Dereceler Toplamı L=26												←														
Poisson Rassal Sayı Parametre Değeri $\lambda=L/2=13$																										
Poissonmd(13)=9																										

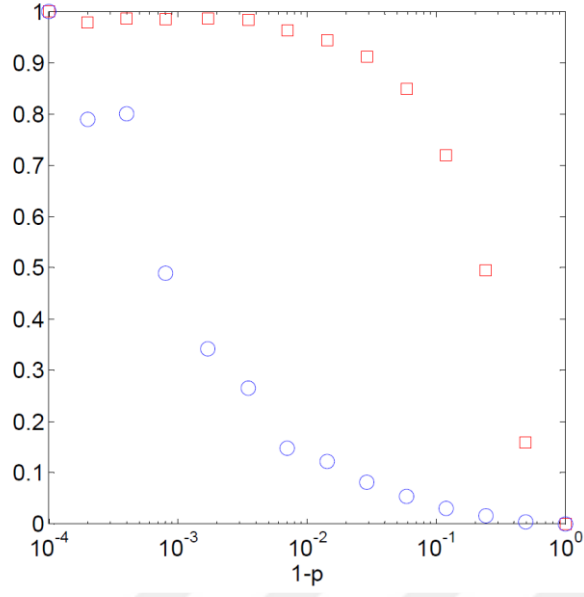
Şekil 4.13 Yeniden bağlanma prosedürü örneği.

Yapılan çalışma sonucunda elde edilen sonuçlar Çizelge 4.7'de sunulmuş, Şekil 4.14 - 4.15'de görselleştirilmiştir.

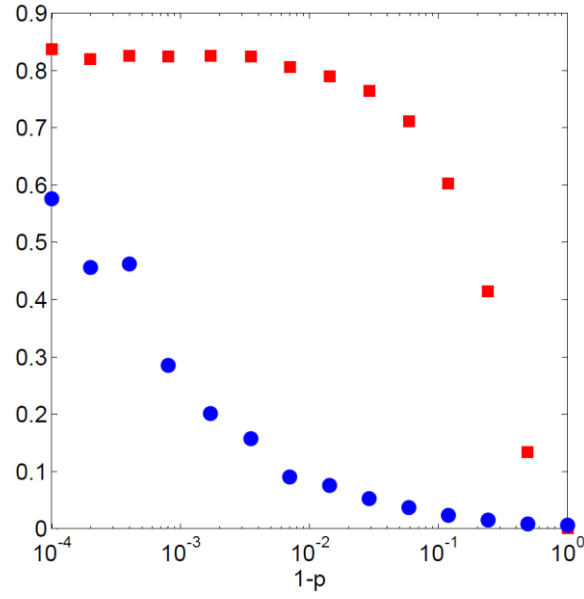
Çizelge 4.7. ER Model – Düzenli Ağ ve BA Model arası dönüşüm denemeleri sonucu elde edilen kümelenme katsayısı (CC), ortalama yol uzunluğu ($\langle d \rangle$) değerleri.

N	RN_{Prob}	<i>Narrow Factor</i>	P_{rewire}	CC	$\langle d \rangle$
1000	0,01	0,50	0,9999	0,5595	30,3117
1000	0,01	0,50	0,9998	0,5478	24,6199
1000	0,01	0,50	0,9996	0,5523	24,9090
1000	0,01	0,50	0,9992	0,5514	16,4920
1000	0,01	0,50	0,9983	0,5520	12,5052
1000	0,01	0,50	0,9965	0,5508	10,4297
1000	0,01	0,50	0,9930	0,5393	7,2525
1000	0,01	0,50	0,9857	0,5285	6,5501
1000	0,01	0,50	0,9711	0,5114	5,4597
1000	0,01	0,50	0,9412	0,4764	4,7143
1000	0,01	0,50	0,8806	0,4050	4,0710
1000	0,01	0,50	0,7576	0,2817	3,6845
1000	0,01	0,50	0,5076	0,0968	3,3613
1000	0,01	0,50	0	0,0097	3,2579

Bu bölümde uygulanan ER-DA_BA model hibrit dönüşümü sonucunda belli bir p_{rewire} aralığında düşük $\langle d \rangle$ ile birlikte yüksek CC değerlerinin edildiği bir “small-world” bölgesinin varlığı gözlenmiştir. Ancak bu bölgede derece dağılımları incelendiğinde gerçel ağlar ile benzemediği, Poisson dağılımında fazla uzaklaşmadığı görülmektedir. Dolayısıyla seçimli bağlanma olgusu yerel olarak uygulandığında, ağın genel derece dağılımını yeterince etkileyemediği sonucuna varılabilir.

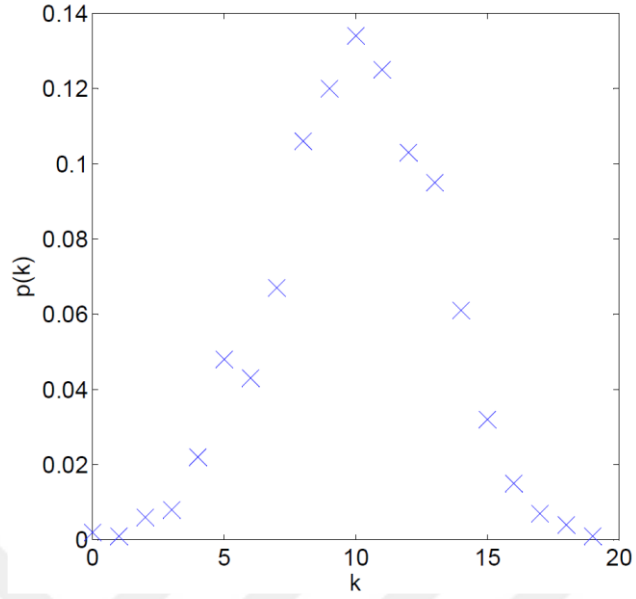


(a)

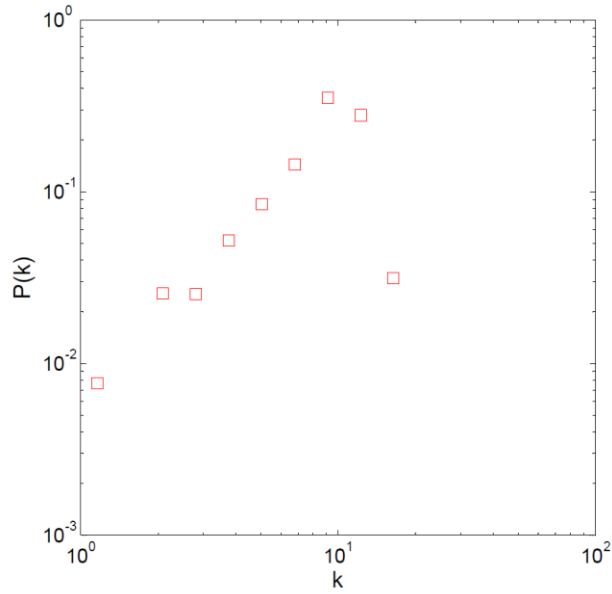


(b)

Şekil 4.14. a) Yerel minimum ve maksimum değerlere göre normalize edilmiş, kümelenme katsayısı (CC= \square), ortalama yol uzunluğu ($\langle d \rangle = \circ$) değişim grafiği, b) Global minimum ve maksimum değerlere göre normalize edilmiş, kümelenme katsayısı (CC= \blacksquare), ortalama yol uzunluğu ($\langle d \rangle = \bullet$)



(a)



(b)

Şekil 4.15. $p_{rewire}=0,9857$ olasılıkta oluşan derece dağılım grafikleri a) Normal eksen dağılım grafiği, b) log-log ekseninde oluşan derece dağılım grafiği.

4.7. DÖNÜŞÜM MODELLERİNE GENEL BAKIŞ

Modeller arası dönüşüm çalışmalarında başarılı yöntem ve metotlar kullanılmıştır. Rassal ağları ve kümelenmenin düşük olduğu diğer ağ modellerini, düzenleştirme işlemi için tanımlanan yeniden bağlanma prosedürlerinde, Poisson dağılımıyla kaynak

düğüme yakın rassal sayılar üretilmiş ve bu yakınlığı da kontrol edebilmek için narrow factor (daraltma faktörü) kullanılmıştır. Yeniden bağlanma olasılığı p_{rewire} değerinin [0-1] aralığında 14 farklı değeri için dönüşüm denemesi yapılmıştır. Ve bu yeniden bağlanma değerleri için ağlarda düşük $\langle d \rangle$ ve yüksek CC (small-world) özellikleri araştırılmıştır. Bazı dönüşümler için bu “small-world” özelliği, $p_{rewire} \sim 0,01$, bazılarında ise $\sim 0,9$ 'a yakın olduğu değerlerde gözlenmiştir. Bu nedenle CC ve $\langle d \rangle$ özelliklerinin değişimini logaritmik ölçekte iyi gözlemleyebilmek için p_{rewire} eksenleri gerektiğinde $(1 - p_{rewire})$ olarak da ölçeklenmiştir.

Dönüşümlerde kullanılan ağlara ait düğüm ve link sayıları, Watts ve Strogatz'ın [8] ağ modelinde kullandığı değerler ile uyumludur. Ayrıca bu dönüşümler sonucunda oluşan CC ve $\langle d \rangle$ değerleri içinden global minimum ve maksimum değerleri seçilerek; CC ve $\langle d \rangle$ değerlerinin, p_{rewire} olasılığı ile olan değişimleri hem yerel hem de global değerlere göre normalize edilmiş ve birbirleriyle kıyaslanmıştır. Yapılan dönüşüm çalışmalarında, her deneme sonucunda oluşan derece dağılımı hem normal eksen, hem de log-log ekseninde çizdirilmiştir.

Bu gerçekleştirilen modeller arası dönüşümler sonucunda aşağıdaki çıkarımlar elde edilmiştir:

1. Dönüşümlerde, gerçel ağ özellikleri kısmen gözlenmiştir. Kümelenme ve ortalama yol uzunlukları açısından bazı modellerde “small-world” (düşük $\langle d \rangle$, yüksek kümelenme CC) özelliği gözlenmiştir. ER-DA modelleri arası dönüşüm için $p_{rewire} \sim 0,9$ civarında, DA-BA modelleri arası dönüşümde $p_{rewire} \sim 0,1$ civarında, BA-DA modeller arası dönüşümde $p_{rewire} \sim 0,9$ civarında, ER-DA_BA modelleri arası dönüşümde, $p_{rewire} \sim 0,9$ civarında small-world aralık yakalanmıştır.
2. ER-BA, BA-ER modelleri arası dönüşümlerde “small-world” özelliğine rastlanmamıştır.
3. Dönüşümler arasında diğer bir dikkat çekici gerçel ağ özelliği de derece dağılımıdır. Bazı model dönüşümleri sonucunda, gerçel ağların derece dağılımı özelliği kısmen sağlanmıştır. DA-BA model dönüşümü, (Şekil 4.19. b) gerçel ağlarda da bulunan bir düşük k saturasyon bölgesine sahip bir power-law

derece dağılımı sergilemektedir. Ancak düşük k saturasyon bölgesinin gerçel ağlara oranla daha geniş ($1 < k < 10$) olması nedeniyle önemli bir farklılık göstermektedir. Bir diğer modeller arası dönüşüm olan ER-BA'da ise p_{rewire} değerleri 1'e yaklaştıkça derece dağılımı Poisson'dan power-law'a sınırlı bir değişim göstermektedir.

4. Bazı modeller arası dönüşümlerde, gerçel ağların karakteristik özelliklerinden olan "small-world" özelliği açısından olumlu sonuçlara ulaşılmıştır. Ancak bir diğer karakteristik özellik olan power-law derece dağılımı noktasında istenilen sonuçlara tam olarak ulaşamamıştır.

Ağ modelleri arası yapılan dönüşüm çalışmalarının MATLAB uygulama kodları EK AÇIKLAMALAR A'da verilmiştir.

BÖLÜM 5

WEB TABANLI KOMPLEKS AĞ SİMÜLATÖRÜ

Tez çalışmamızın önemli çıktılarında birisi Web tabanlı kompleks ağ simülatörüdür. Tamamen web tabanlı ve web tarayıcılarının tümünde çalışabilen bir uygulama geliştirilmesi amaçlanmıştır. Bu amaçla gerçekleştirilecek uygulama için hangi web tabanlı teknolojilerin ve API'lerin kullanılabileceği araştırılmıştır.

Modern web tarayıcılar, sahip oldukları JavaScript desteğinin ötesinde, gittikçe daha güçlü özellikler kazanmaya başlamıştır. HTML5 etiketleri ve interaktif HTML5 Canvas nesnelerinin kullanılmasıyla ses, video ve görsel desteği üst düzeye çıkmıştır. Bu özelliklere, WebGL desteği de eklenmiştir. WebGL ile birlikte ekran kartının işlemi gücü doğrudan kullanılabilmekte ve yüksek performanslı 2B ve 3B grafikler oluşturulabilmektedir [45].

Bu güçlü özelliklerin yanında WebGL ile doğrudan JavaScript kodlarıyla 2B ve 3B grafikler oluşturmak ve hareketlendirmek oldukça karmaşıktır ve hata ile karşılaşma ihtimali yüksektir. Three.js bu durumun zorluğunu ortadan kaldıran bir JS kütüphanesidir [45].

Gerçekleştirilen web tabanlı uygulamada WebGL için Three.js ile birlikte birkaç JavaScript kütüphanesinden yararlanılmış ve kompleks ağ gereksinimleri için özel kodlamalar JavaScript dilinde yapılmıştır. Graf özellikleri de dikkate alınarak grafların gösterimi için özel veri tanımlamaları, temel özelliklerinin analiz edilmesi için JS fonksiyonlar, graflarda düğüm ve bağlantıların görsel olarak değişimi ve daha düzenli gösterimi için yerleşim algoritmaları kullanılmıştır. Yazılımla ilgili detay bilgiler bu bölümün devamında verilecektir.

5.1. YAZILIM BİLEŞENLERİ

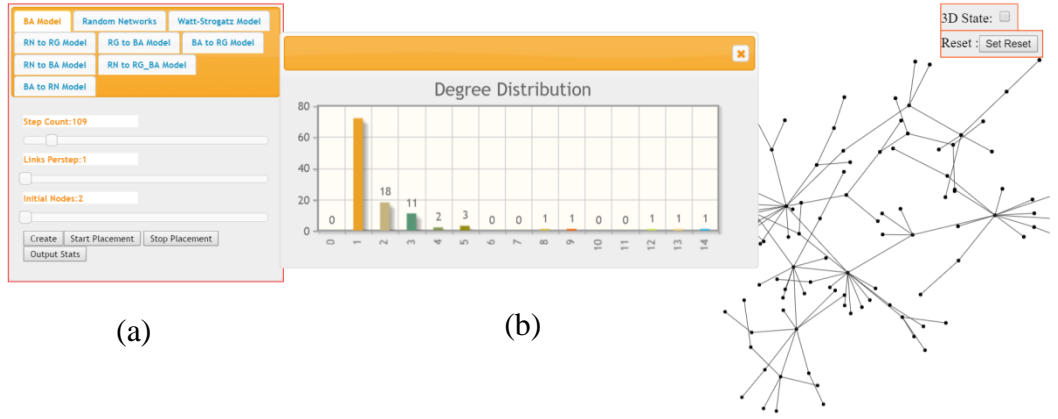
Geliştirilen bu yazılım ile web tarayıcıların tümünde herhangi bir eklentiye gerek duymadan çalışan bir uygulama geliştirilmesi amaçlanmıştır. Daha iyi bir kullanıcı deneyimi sağlamak için, kompleks ağların 3B sunumu, taşınması, yakınlaştırılması ve çizimi için çeşitli kütüphaneler kullanılmıştır. Ayrıca temel ağ modelleri ve bu modeller arasındaki dönüşümlerin sonucunda oluşan ağların çıktıları farklı formatlarda (text, png, richtext) alınabilmesi için JS kütüphanelerinden yararlanılmıştır. Uygulamayı destekleyen kütüphaneler ve bileşenler aşağıda verilmiştir.

5.1.1. jQuery Ve jQuery UI Kütüphaneleri

jQuery hızlı, küçük boyutlu ve zengin özellikli bir JS kütüphanedir. HTML dokümanlarının işlenmesini ve oluşturulmasını, olayların işlenmesini, animasyonların ve AJAX işlemlerinin çok daha kolay gerçekleştirilmesini sağlar. jQuery UI, jQuery JS kütüphanesi üzerine inşa edilmiş kullanıcı etkileşimleri, efektler, widgetlar ve temalardan oluşan yardımcı bir kütüphanedir. Yüksek etkileşimli web uygulamaları oluşturmak için, yaygın kullanılan bileşenlerin web sayfalarına eklenmesiyle çok basit ve etkili bir yol sunmaktadır [46, 47].

Geliştirilen uygulamada bu iki kütüphane, temel kompleks ağ modelleri ve bu modeller arasındaki dönüşümler için veri giriş alanı oluşturulmasında kullanılmıştır. jQuery UI'nin "Tabs" bileşeniyle 3 temel model ve 6 dönüşüm için veri girişlerinin, sayfa kaydırma olmaksızın aynı ekranda gerçekleştirilmesi sağlanmaktadır. Aynı zamanda görselleştirme aşamasında derece dağılımlarının grafiksel ara yüzde gösterilmesi için "Dialog" bileşeni kullanılmıştır. Dialog bileşeni fare ile taşınabilmekte ve kapatılabilme özelliğine sahiptir.

Şekil 5.1'de bu kütüphanelerin kullanıldığı HTML bileşenleri gösterilmiştir.

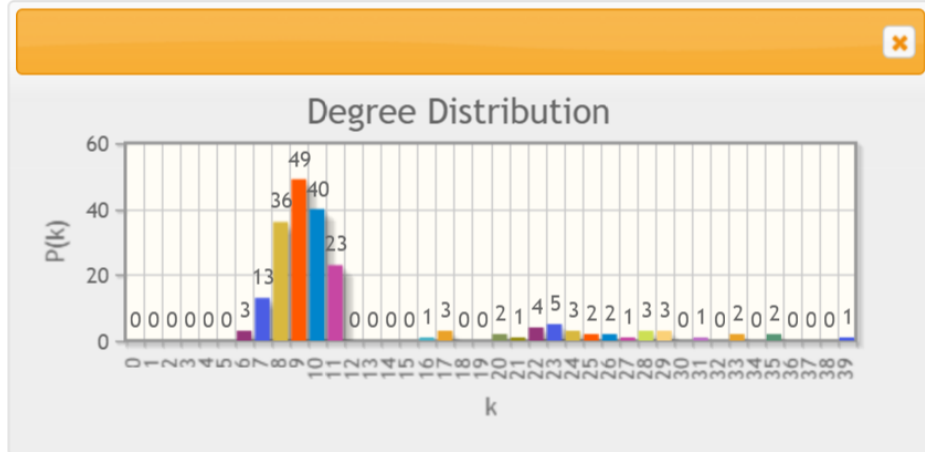


Şekil 5.1. a) jQuery UI “Tabs” bileşeni kullanımı, b) “Dialog” bileşeni kullanımı

5.1.2. jqPlot Kütüphanesi

jqPlot, sadece istemci tarafında grafikler oluşturan bir jQuery eklentisidir. jqPlot çubuk, çizgi ve pasta grafiklerini birçok özelliği ile birlikte üretir. Çizgiler, eksenler, gölgeler, hatta çizim alanındaki bölümlenmeyi sağlayan ızgara yapısı bile jqPlot’ın kendi içerisinde eklenip çıkarılabilen “renderers” (görüntü giydirci) vasıtasıyla hesaplanıp çizdirilebilmektedir. Çalışabilmesi için jQuery 1.4 ve üstü versiyonlara gereksinim duymaktadır [48].

jqPlot kütüphanesi uygulamamızda modellerin görselleştirilmesi aşamasında derece dağılımlarının grafiklerini oluşturmak için kullanıldı. Grafik çizim işlemleri, jqPlot eklentisinin çeşitli görüntü giydircileri; barRenderer, axisRenderer, logAxisRenderers vs. kullanılarak gerçekleştirilmiştir. Şekil 5.2.’de jqPlot ile çizdirilen bir derece dağılımı grafiği gösterilmiştir.



Şekil 5.2. Başlangıçta $N=200$ ve $k = 6$ olan Düzenli bir ağın, $p_{rewire} = 0,4924$ yeniden bağlanma olasılığı ile BA modele dönüşümü sonucu oluşan derece dağılımı.

5.1.3. Three.js Kütüphanesi

Three.js kütüphanesi; WebGL'in detaylarını öğrenmeye ihtiyaç duymadan güzel 3B grafikler oluşturabileceğimiz, WebGL özellikleri üzerine kurulmuş, kullanımı kolay bir JavaScript API'sidir. Three.js; 3B sahneleri direkt olarak web tarayıcıda oluşturan birçok özellik ve API'lere sahiptir [45].

Three.js ile yapılabilecek bazı şeylere örnek olarak: basit ve kompleks 3B geometrik nesnelere oluşturma, 3B sahnelerdeki nesnelere taşıma ve hareketlendirme, nesnelere doku ve materyal uygulama, 3B modelleme yazılımlarında nesne yükleme, 2B hareket tabanlı grafik oluşturma verilebilir [45].

Three.js tüm WebGL yeteneğine sahip grafik kartları ile Google Chrome Versiyon 9.0+, Mozilla Firefox Versiyon 4.0+, Safari Versiyon 5.1+, Opera Versiyon 12.0+, Internet Explorer Versiyon 11+ ve Microsoft Edge Versiyon 0.95+ web tarayıcılarla uyumludur.

Gerçekleştirilen uygulamada three.js, ağ modellerinin 3B görselleştirilmesinin yanı sıra 3B animasyonlar ve hareketler için de kullanılmıştır. 3B taşımalar, kamera açıları, yakınlaştırma ve uzaklaştırma eylemleri için three.js kütüphanesinin "trackballcontrols" eylemleri kullanılmıştır.

Şekil 5.3'te three.js kullanımı ile ilgili bir örnek verilmiştir.



Şekil 5.3. three.js kütüphanesi kullanılarak Watts-Strogatz model ağının simüle edilmesi sonucu oluşan grafik görüntü.

Bu kullanılan bileşenler dışında JavaScript'in veri tanımlama özellikleri kullanılarak kompleks ağı oluşturan graflarla ilgili düğüm, bağlantı, düğüm dereceleri, derece dağılımları, yerel kümelenme katsayısı, ortalama yol uzunluk tanımlamaları yapılmıştır. Bu bağlamda düğüm ve bağlantı, düğüm derecelerini oluşturmak ve bağlantılar oluşturulurken düğümlerin daha önceden bağlanıp bağlanmadığının kontrolünün yapılması için Graph.js dosyası oluşturulmuştur. Uygulamaya ait html sayfasında script etiketler kullanılarak diğer veri tanımlamaları yapılmıştır. Three.js, jqplot, jQuery tanımlamaları ve işlevleri, temel modeller ve modeller arası dönüşümlere ait JS kodlamaları bu sayfada yapılmıştır.

5.1.4. FileSaver.js Kütüphanesi

FileSaver.js hassas bilgileri kaydetmek ve dosya oluşturmak için bir sunucuya ihtiyaç duymayan, istemci tarafında çalışan ideal bir web tabanlı kütüphanedir. FileSaver.js farklı formatlardaki (text, canvas, richtext) dosya verilerini kaydedebilmektedir. Ayrıca dosya sistemlerindeki büyük nesne dosyalarının veritabanlarında saklanması

sağlayan Blobs (Binary Large Objects) formatındaki verileri de kaydedebilmektedir [49, 50].

5.2. YAZILIM ÖZELLİKLERİ

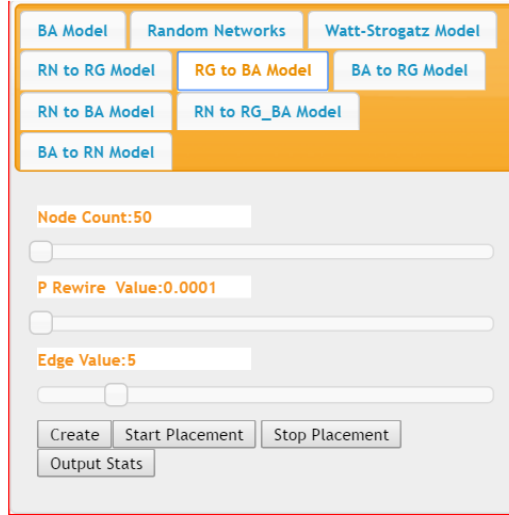
Geliştirilen uygulamada, öncelikli hedeflenen nokta yüksek erişimli bir yazılım gerçekleştirilmesidir. Bu nedenle platform bağımsız, tamamen tarayıcı üzerinde ve eklenti gerektirmeden çalışan bir uygulama geliştirilmiştir. Uygulamada, Bölüm 2’de verilen ağ modelleri ve bu modeller arasındaki dönüşümlerin gerçekleştirilmesi ve bu ağ modellerinin 2B ve 3B olarak görselleştirilmesi sağlanmıştır. Ayrıca temel ağ özellikleri olan kümelenme katsayısı, ortalama yol uzunlukları ve derece dağılımı her bir model ve modeller arası dönüşümler için analiz edilmiştir.

Yazılım ile ilgili görselleştirme ve analiz özellikleri bu konu başlığı altında verilmiştir.

5.2.1. Model Seçimi İçin Sekmeli Ayar Paneli

Ağ modelleri arasındaki dönüşüm parametreleri ve temel ağ modellerinin başlangıç değer atamalarının yapıldığı bir veri giriş alanına ihtiyaç duyulmuştur. Bunun için jQuery UI kütüphanesinin “Tabs” bileşeni kullanılmıştır. Sekmede 3 temel model ve bu temel modellerle birlikte düzenli ağ modeline dönüşümlerin oluşturduğu 6 ara model tanımlanmıştır. İlk olarak 3 temel model; “BA Model”, “Random Networks” ve “Watts-Strogatz Model”, daha sonra Bölüm 3’te tanımlanan modeller arası dönüşümler; “ER to RG Model”, “RG to BA Model”, “BA to RG Model”, “ER to BA Model”, “BA to ER Model”, “ER to RG_BA Model” sekmeleri tanımlanmıştır. Her bir ağ modeli sekmesi seçildiği zaman veri girişi için bir HTML “<div>” alanı açılmaktadır. Bu alanda seçili ağ modelleri için başlangıç parametreleri girilebilmektedir. Aynı zamanda bu alanda ağ oluşturma, yerleşim ve istatistiksel sonuç alma işlevleri için de butonlar tanımlanmıştır.

Şekil 5.4’te, oluşturulan panel ile ilgili görsel verilmiştir.

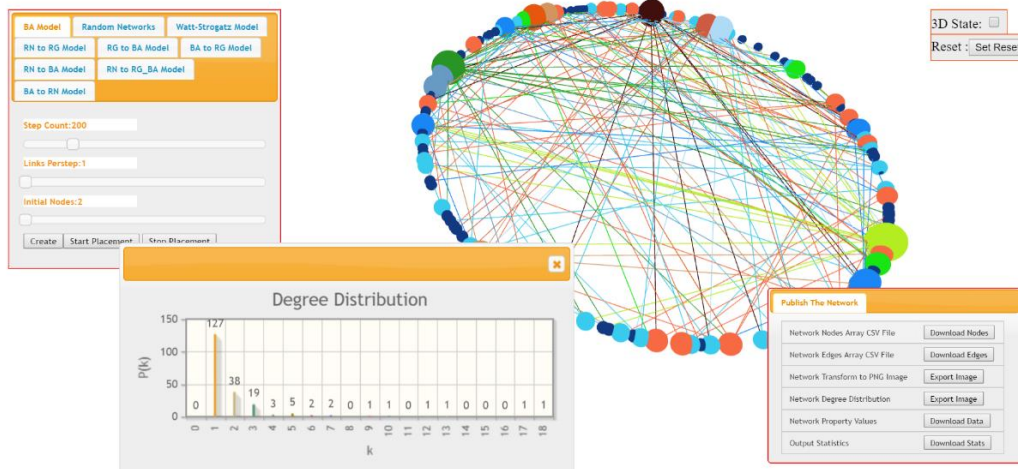


Şekil 5.4. Model seçimi için sekmeli ayar paneli

5.2.2. 2B Görselleştirme Ve 3B Dönüşümler

Uygulamanın kullanıcı ara yüzünde “3D State” seçim kutusu ile 3B görselleştirme aktif hale getirmektedir. Bu alan seçilmediği takdirde ağ 2B olarak ve çember yerleşim düzeni ile görselleştirilmektedir. Çizdirilen ağ grafiği için, döndürme, yakınlaştırma ve uzaklaştırma, taşıma gibi 3B dönüşümler, fare ile sağ-sol tıklanması, sürüklenmesi ve kaydırılması olayları kullanılarak yapılmaktadır.

Fare ile kamera açısı değiştirilmiş, sola taşınmış bir ağ Şekil 5.5’de gösterilmiştir.



Şekil 5.5. Geliştirilen uygulama kullanıcı ara yüzü.

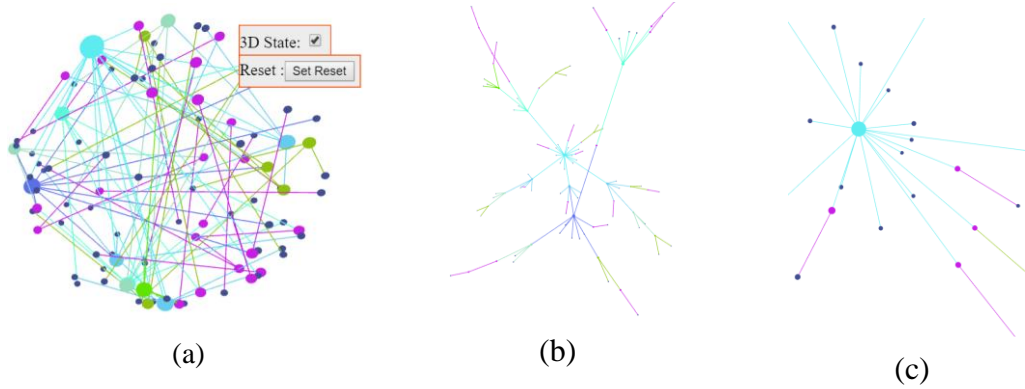
5.2.3. 3B Görselleştirme Ve 3B Dönüşümler

Uygulamada “3D State” seçim kutusu işaretlendiğinde ağ, 3B olarak ve küresel yerleşim düzeni ile görselleştirilmektedir. Döndürme, yakınlaştırma ve uzaklaştırma, taşıma gibi 3B dönüşümler 2B çizimlerdeki gibi gerçekleştirilmektedir.

Şekil 5.6’da 3B görselleştirilen, yerleşim düzeni ve kamera görüş açıları değiştirilen bir kompleks ağ gösterilmiştir.

5.2.4. Fare Olayları

Uygulamada fare olayları, görselleştirilen kompleks ağın kamera bakış açılarının değiştirilmesini sağlamaktadır. Three.js’te kamera açılarının fare ile değiştirilmesi kütüphane içerisinde yer alan TrackBallControls.js ile gerçekleştirilmektedir. Sol fare tuşuna basılı tutulup sürüklendiğinde nesne kendi etrafında döndürülebilmektedir. Fare tekerleği ile yakınlaştırma uzaklaştırma yapılabilmekte ve sağ fare tuşuyla nesne aşağı, yukarı, sağa ve sola doğru taşınabilmektedir [51].

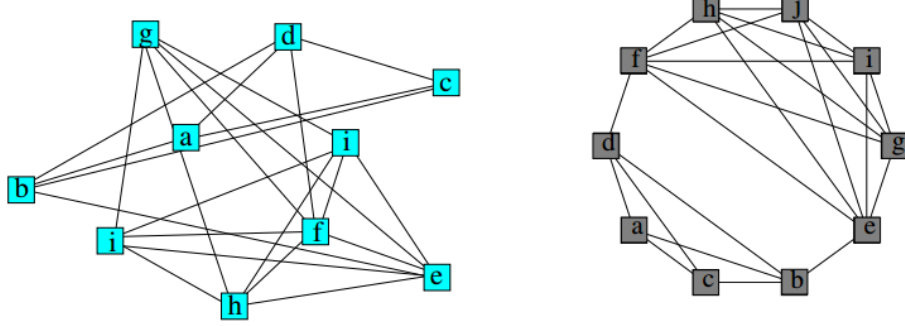


Şekil 5.6. a) Bir kompleks ağın 3B görselleştirilmesi, b) Fruchterman-Reingold yerleşim düzeni uygulanması ve c) fare ile yakınlaştırılması.

5.2.5. Yerleşim Düzeni Değişimi

Web tabanlı uygulamada her bir modelin görselleştirilmesi için başlangıçta çember yerleşim düzeni uygulanmıştır. Bir grafın çember yerleşim düzeninde görselleştirilmesi için graf düğümlerinin bir daire üzerine eşit uzaklıklarla

yerleştirilmesi ve düğümler arasındaki bağlantıların düz çizgilerle çizilmesi gerekmektedir [52].



Şekil 5.7. Rassal koordinatlarla oluşturulmuş grafin (solda), daire yerleşim düzeni (sağda) ile görselleştirilmesi [52].

Uygulamada varsayılan özellik olarak 2B görselleştirme için daire yerleşim düzeni, 3B görselleştirme için küresel yerleşim düzeni kullanılmıştır.

Ancak düğüm sayısı arttığında dairesel ve küresel yerleşimle görselleştirilen ağ karmaşık hale gelmekte, iç yapısının daha iyi anlaşılması zorlaşmaktadır. Özellikle gerçel ağların ağaç benzeri yapıları, kuvvet yönelimli algoritmalar ile daha anlaşılır biçimde görselleştirilebilmektedir. Çalışmamızda da varayılan dairesel ve küresel yerleşim düzeni istenildiği takdirde kuvvet yönelimli yerleşim düzene çevrilebilmektedir.

Kuvvet-yönelimli yerleşim düzen algoritmaları basit yönsüz grafik yerleşim düzenleri hesaplamak için en esnek yöntemler arasındadır. Bu tür algoritmalar grafin etki alanına özgü bilgileri kullanmak yerine, sadece grafin kendi bünyesinde yer alan bilgileri kullanarak yerleşim düzenlerini hesaplar. Bu yerleşim düzeni ile çizilen graflar, estetik olma eğilimindedir, simetri gösterirler ve düzlemsel graflar için geçiş-serbest düzenleri üretme eğilimindedirler [53].

Eades'in 1984 yılında oluşturduğu yerleşim algoritması, 30 düğümlü estetik görünümlü bir grafi, 2B yerleşim ve CRT monitörler için mekanik bir modelle

üretmiştir. Estetik olma özelliği; tüm kenar uzunluklarının eşit olduğu ve yerleşimin olabildiğince simetrik olması kriterleri ile sağlanmıştır [53].

Yukarıdaki bu kıstaslara, Fruchterman ve Reingold 1991 yılında “düzenli düğüm dağılımı” kriterini eklemiş, graf içerisindeki düğümlere “atom parçacıkları veya gök cisimlerinin birbirlerine uyguladıkları itme çekme kuvvetlerini” uygulamıştır [53].

Bu kuvvetler eşitlik 5.1’de tanımlanmıştır:

$$f_a(d) = d^2/k, \quad f_r(d) = -k^2/d \quad (5.1)$$

İki düğüm arasındaki mesafe d ile gösterilir. İki düğüm arasındaki en uygun mesafe k eşitlik 5.2’de tanımlanmıştır:

$$k = C \sqrt{\frac{\text{area}}{\text{number of vertices}}} \quad (5.2)$$

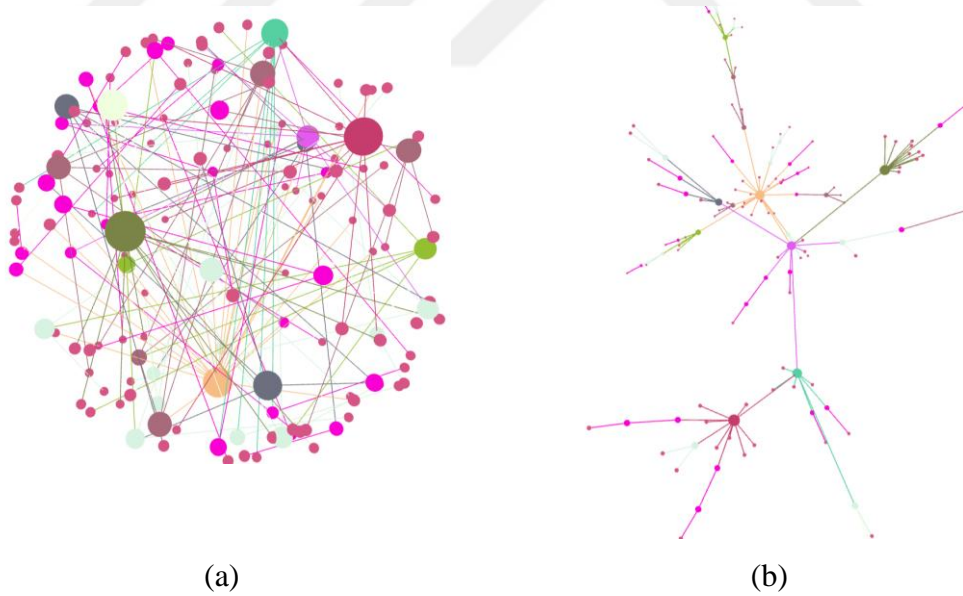
Bu yerleşim düzeni algoritmasında komşu düğümler için çekme kuvvetleri (f_a), tüm düğüm çiftleri içinde itme kuvvetleri (f_r) hesaplanır. Aynı zamanda yerleşimin gerçekleştirileceği alanın yükseklik ve genişlik değeriyle bir alan (area) hesaplanır ve düğüm sayısına bölünerek k değeri hesaplanır. Burada C değeri sabittir. Algoritmaya eklenen bir “temperature” değişkeni düğümlerin yer değişimlerini kontrol etmektedir, bu sayede yerleşimi daha düzenli hale getirmekte ve çizim alanına bir sınırlama getirerek düğümlerin yer değişimlerini kısıltmaktadır [53].

Algoritmanın çalışması özetle şu şekildedir: Öncelikle başlangıç parametreleri olan graf (düğüm ve bağlantılarla birlikte), çizim alanı değerleri ve algoritma iteratif bir yaklaşımla çalıştığı için iterasyon sayısı değeri algoritmaya verilmektedir. Her bir iterasyonda düğüm dizisi ve komşuluk dizisi kullanılarak düğümlerin, komşu oldukları düğümler kullanılarak itme kuvvetleri hesaplanır. Hesaplanan bu kuvvet ile mevcut düğümlerin yeni yerleşim pozisyonları hesaplanır. Ardından bağlantı dizisinden yararlanılarak her bir bağlantıdaki kaynak ve hedef düğümlerin birbirlerine uyguladıkları çekme kuvvetleri hesaplanır. Hesaplanan bu kuvvet ile yerleşim

pozisyonları her iki düğüm için tekrar güncellenir. İtme ve çekme kuvvet hesaplamaları bittiğinde son olarak “temperature” değişkeninden yararlanılarak yerleşim pozisyonunun çizim alanı içerisinde kalması sağlanır. Bu işlemler bittikten sonra “temperature” değeri azaltılarak daha iyi bir yerleşim elde edilmesi sağlanır.

Çalışmamızda Fruchterman-Reingold yerleşim düzeni algoritmasının 3B kullanımı için Bjorge'nin CUDA platformu için yayınlamış olduğu içerikten uyarlama yapılmıştır [54].

Geliştirdiğimiz uygulamada veri giriş panelinin alt kısmında Fruchterman-Reingold yerleşimi algoritmasına göre ağı yeniden düzenlemek için “Start Placement” ve “Stop Placement” düğmeleri kullanılmıştır. İlk düğmeyle çember yerleşim düzeninden, Fruchterman-Reingold yerleşim düzenine geçiş için yukarıda anlatılmış olan kurallara uygun olarak gerçek zamanlı yerleşim değişimi gerçekleştirilmektedir. İkinci düğme ile bu yerleşim döngüsü durdurulmakta ve kompleks ağ son görsel halini almaktadır.



Şekil 5.8. a) Daire yerleşim düzeni ile oluşturulmuş 3B ağ, b) bu ağa Fruchterman-Reingold yerleşim düzeni algoritması uygulanması.

5.2.6. Derece Dağılımı, Kümelenme Katsayısı Ve Ortalama Yol Uzunluğu

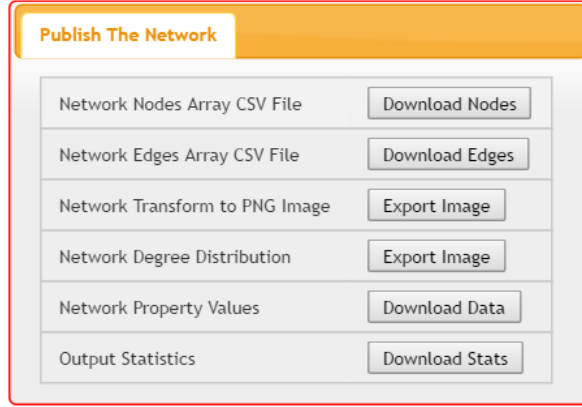
Uygulamada oluşturulan her bir ağın derece dağılımı jqPlot eklentisi yardımıyla bir pop-up pencere içerisinde çizdirilmektedir. Derece dağılımı gerçek zamanlı ağın oluşumunda bir veri dizisinde depolanmaktadır. Bu veri dizisi jqPlot bileşenine parametre olarak aktarılır. Kompleks ağın kümelenme katsayısı ve düğümler arası ortalama yol uzunlukları “Publish The Network” paneli içerisinde yer alan “Output Statistics” kısa yoluyla text dosya olarak kaydedilebilmektedir.

Kümelenme katsayısı hesaplanırken, yerel kümelenme katsayılarının ortalamalarının alınması yönteminden yararlanılmaktadır. Hesaplanma yöntemi Bölüm 2’de verilmiştir. Bu hesaplama için ağda bulunan her bir düğüm ve bağlantıdan oluşan bir komşuluk dizisi tanımlanmıştır. Bu dizide yer alan değerler yardımı ile her bir düğümün yerel kümelenme katsayısı yine bir kümelenme dizisinde tutulmaktadır. Bu dizide yer alan değerler ile de yerel kümelenme katsayısı ortalaması hesaplanmaktadır.

Ortalama yol uzunluğu hesabı için ise, gephi uygulamasında “aradalık merkezliliği” hesaplamak için geliştirilmiş algoritmadan yararlanılmıştır [55]. Düğümler arası mesafeler derinlemesine arama algoritması (BFS) ile bulunarak, tanımlanan komşuluk matrisine girilmekte ve algoritmanın çalışması bittiğinde ortalama yol uzunluğu değeri bu matris yolu ile bulunmaktadır.

5.2.7. Ağ Çıktıları

Web tabanlı uygulamada temel ağ modellerinin ve model dönüşümlerinin gerçekleştirilmesinin ardından oluşan ağın çıktıları, oluşturulan “Publish The Network” paneliyle alınabilmektedir. Oluşan ağın düğümleri, bağlantıları “csv” formatında alınabilmekte ve her bir ağ için tanımlanmış başlangıç parametreleri ve ağ istatistikleri “txt” formatında kaydedilebilmektedir. Oluşturulan ağın görseli ve derece dağılımı “png” resim formatında alınabilmektedir. Şekil 5.9’da ağ çıktılarının alınabilmesi için oluşturulmuş panel verilmiştir.

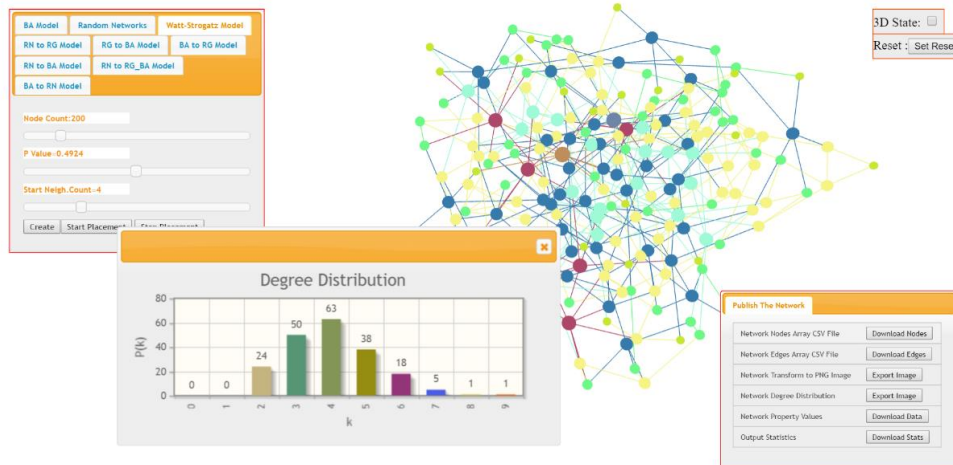


Şekil 5.9. Ağ çıktıları paneli.

5.3. UYGULAMA ÖRNEKLERİ

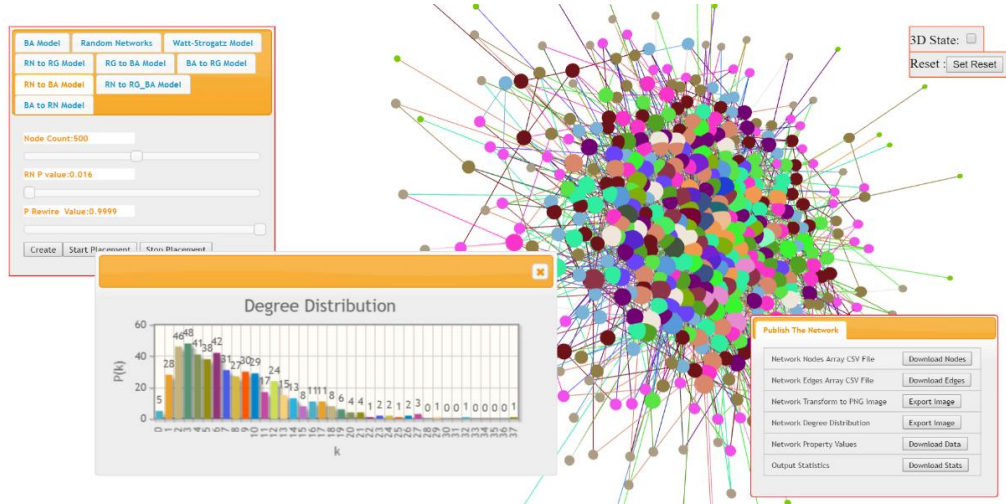
Yukarıda 3 temel model ve modeller arası dönüşüm ile elde edilen 6 ağ yapısından bahsedilmişti. Bu konu başlığında bu modeller için yapılan bazı görselleştirme örnekleri verilmiştir.

Şekil 5.10'da başlangıçta 200 düğüm ve her bir düğümün 4 komşusu olduğu bir düzenli ağ, $p_{rewire} = 0,4924$ yeniden bağlanma prosedürüne tabi tutulmuştur. Bunun sonucunda oluşan ağ 2B olarak görselleştirilmiş ve temel ağ özellikleri analiz edilmiştir.



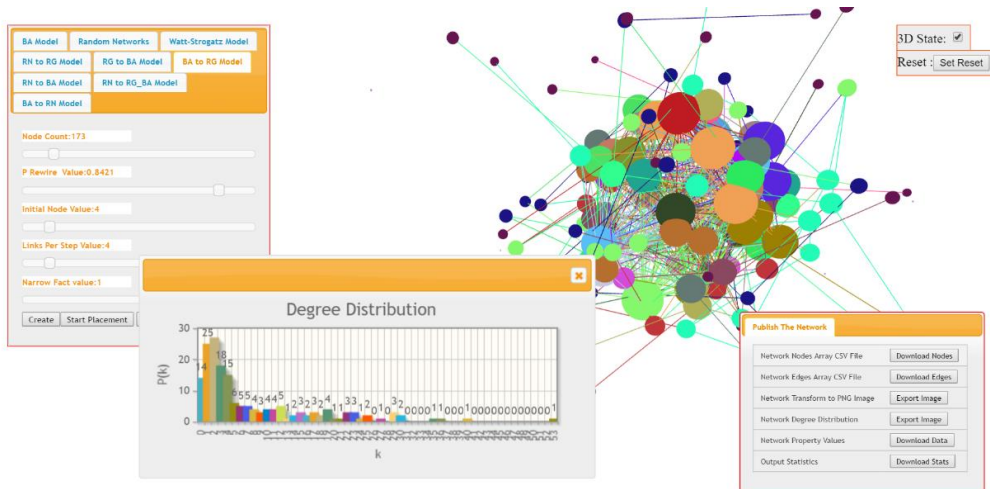
Şekil 5.10. Watt-Strogatz modelde oluşturulmuş ve analiz edilmiş bir 2B kompleks ağ ve analiz sonuçları.

Şekil 5.11’de düğüm sayısı 500 ve $p=0,036$ bağlantı olasılığı ile oluşturulmuş bir rassal ağ, $p_{rewire} = 0,9999$ yeniden bağlanma olasılığı ile BA modele dönüşümü sağlanmış ve 2B olarak görselleştirilmiştir.



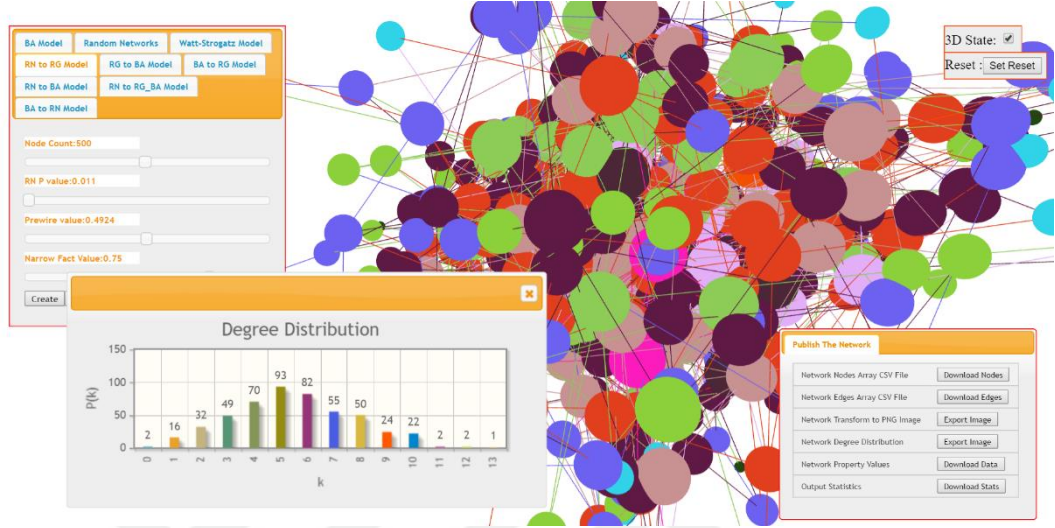
Şekil 5.11. ER model-BA model arası dönüşüm sonucu oluşan 2B kompleks ağ ve analiz sonuçları.

Şekil 5.12’de başlangıç düğüm sayısı=4, her bir iterasyonda 4 bağlantı oluşturacak olan bir BA model ağı, $p_{rewire} = 0,8421$ yeniden bağlanma olasılığı ve daraltma faktörü 1 değeri ile Düzenli Ağ modeline dönüşümü sağlanmış ve 3B olarak görselleştirilmiştir.



Şekil 5.12. BA model – Düzenli Ağ modeli arası dönüşüm sonucunda oluşan 3B bir kompleks ağ ve analiz sonuçları.

Şekil 5.13’de düğüm sayısı 500 ve $p=0,011$ bağlantı olasılığı ile oluşturulmuş bir rassal ağ, $p_{rewire} = 0,4924$ yeniden bağlanma olasılığı ve 0,75 daraltma faktörü ile Düzenli Ağ ve BA modele dönüşümü sağlanmış ve 3B olarak görselleştirilmiştir.



Şekil 5.13. ER model- Düzenli Ağ ve BA model arası dönüşüm sonucunda oluşan 3B bir kompleks ağ ve analiz sonuçları.

Web tabanlı gerçekleştirilen yazılımın örnek kodları EK AÇIKLAMALAR B’de verilmiştir.

BÖLÜM 6

SONUÇLAR

Doğada yer alan birçok ağ tabanlı sistemin, topolojilerinin altında yatan bazı genel organizasyon ilkeleri vardır [1]. Bir bütün olarak bu sistemleri anlamak, kompleks ağ yaklaşımının hem analiz hem de modelleme çalışmalarıyla desteklenmesiyle mümkündür. Gerçek ağların analiz edilmesi ile elde edilen veriler, modelleme çalışmalarında yol göstermektedir.

Literatürde en çok tanınan modeller BA model, ER Model ve WS modeldir. WS model, Düzenli Ağ modeli-ER model arasında yeniden bağlanma prosedürü ile elde edilmiş bir ara modeldir. Tez çalışmamızda bu yeniden bağlanma prosedürünün diğer modeller arasında uygulanması ile gerçek ağlara ne derece benzer yapıların elde edilebileceği araştırılmıştır.

Bu amaçla, tanınmış modeller ve Düzenli Ağ modeli kullanılarak gerçekleştirilen dönüşüm denemelerinde; ER-DA, DA-BA, BA-DA, ER-DA_BA modelleri arasındaki dönüşümlerde elde edilen sonuçlar incelendiğinde kısmen başarılı ara modeller elde edileceği görülmüştür. Kümelenme katsayısı ve ortalama yol uzunlukları değişimi dikkate alındığında, bu ara modellerin bazılarında WS modeldekine yakın (Şekil 2.9) sonuçlar gözlemlenmiştir. Ancak gerçek dünyada yer alan sistemlerin birçoğunun derece dağılım özelliği güç yasasıdır. Dönüşüm çalışmalarında her bir p_{rewire} olasılığı sonucunda oluşan derece dağılımları dikkatli incelenmiş, genel olarak power-law uyumlu derece dağılımlarının kusursuz olarak yakalanmadığı, ancak bazı ara modellerde belli oranda benzerlik elde edildiği gözlemlenmiştir.

Bu anlamda kümelenme katsayısı, ortalama yol uzunlukları ve derece dağılımlarının gerçel ağlara en başarılı benzetiminin BA-DA ve DA-BA dönüşümünde elde edildiği ortaya koyulmuştur.

Tez çalışmasının bir diğer önemli çıktısı ise web tabanlı kompleks ağ simülatörüdür. Bu uygulama tarayıcı tabanlı olup kurulum gerektirmeden, internet tabanlı her bilgisayar ve mobil cihazda çalıştırılabilmektedir. Kolay kurulum ve erişilebilirlik anlamında kompleks ağ modelleme ve eğitim çalışmalarına önemli katkı sağlayacağını düşündüğümüz bu yazılım, aynı zamanda temel ağ parametrelerini de çıktı olarak vermektedir. Tezimiz kapsamında dinamiklerini incelemiş olduğumuz dönüşüm modellerini de içermesi bakımından literatürde benzersiz olan bu uygulama ileride geliştirme potansiyeli ile masaüstü programlara güzel bir alternatif oluşturacak yapıdadır.

Web tabanlı simülatör uygulaması, <http://www.ilkerturker.com/cnWebProject> URL adresinden erişime açıktır. Aynı zamanda açık kaynak kodlu olan bu uygulama, geliştiricilerin de kolaylıkla katkı sağlayabileceği, yeni uygulamalar için baz alabileceği bir kaynaktır.

Web tabanlı kompleks ağ simülatörünün önemli bir hedefi de kompleks ağ eğitiminde kolay erişilebilir ve kullanıcı dostu bir ortam sağlamaktır. Farenin sol, sağ tıklama sürükleme ve scroll özelliklerinin etkin kullanımı ile 3B deneyimi üst düzeye taşıyan uygulama, ağın içinde adeta görsel olarak gezinme imkânı sağlamakta, ağın topolojik yapısının anlaşılmasını kolay hale getirmektedir. Bu yönüyle kompleks ağ eğitiminde kendisine yer bulacağını değerlendirmekteyiz.

KAYNAKLAR

1. Albert, R. and Barabasi, A. L., "Statistical mechanics of complex networks", *Reviews of Modern Physics*, 74 (1): 47-97 (2002).
2. Barabasi, A. L., Jeong, H., Neda, Z., Ravasz, E., Schubert, A. and Vicsek, T., "Evolution of the social network of scientific collaborations", *Physica A: Statistical Mechanics and its Applications*, 311 (3-4): 590-614 (2002).
3. Cavusoglu, A. and Turker, I., "Patterns of collaboration in four scientific disciplines of the Turkish collaboration network", *Physica A: Statistical Mechanics and its Applications*, 413: 220-229 (2014).
4. Cavusoglu, A. and Turker, I., "Scientific collaboration network of Turkey.", *Chaos Solitons & Fractals*, 57: 9-18 (2013).
5. Barabasi, A. L., "Network Science", *Cambridge University Press*, Cambridge, (2016).
6. Kim, J. and Diesner, J., "Coauthorship networks: A directed network approach considering the order and number of coauthors", *Journal of the Association for Information Science and Technology*, 66 (12): 2685-2696 (2015).
7. Strogatz, S. H., "Exploring Complex Networks", *Nature*, 410 (6825): 268-276 (2001).
8. Watts, D. J. and Strogatz, S. H., "Collective dynamics of 'small-world' networks", *Nature*, 393 (6684): 440-442 (1998).
9. Porekar, J., "Random Networks", *Ljubljana* (2002).
10. İnternet: The Oracle of Bacon Home Page, "About the Oracle of Bacon", <https://oracleofbacon.org/> (1999-2016).
11. Newman, M. E. J., "Small Worlds: The Structure of Social Networks", *Complexity* (2000).
12. Doreian, P., "A note on the detection of cliques in valued graphs", *Sociometry*, 32 (2): 237-242 (1969).
13. Opsahl, T. and Panzarasa, P., "Clustering in weighted networks", *Social Networks*, 31 (2): 155-163 (2009).

14. Luce, R. D. and Perry, A. D., "A method of matrix analysis of group structure", *Psychometrika*, 14 (1): 95-116 (1949).
15. Costa, L., Rodrigues, F., and Cristino, A., "Complex networks: the key to systems biology", *Genetics and Molecular Biology*, 31 (3): 591-601 (2008).
16. Barabasi, A. L. and Albert, R., "Emergence of Scaling in Random Networks", *Science*, 286: 509-512 (1999).
17. Bastian, M., Heymann, S. and Jacomy, M., "Gephi: An Open Source Software for Exploring and Manipulating Networks", *International AAAI Conference on Web and Social Media*, North America (2009).
18. Batagelj, V. and Mrvar, A., "Pajek - A Program for Large Network Analysis", *Connections*, 21 (2): 47-57 (1998).
19. Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B. and Ideker, T., "Cytoscape: a software environment for integrated models of biomolecular interaction networks", *Genome Res*, 13 (11): 2498-2504 (2003).
20. Jang, Y., Yu, N., Seo, J., Lee, S. and Kim, S., "MONGKIE: An integrated tool for network analysis and visualization for multi-omics data", *Biology Direct*, 11: 10 (2016).
21. Wang, Z., Xiao, W., Ge, B. and Xu, H., "ADraw: A novel social network visualization tool with attribute-based layout and coloring", *2013 IEEE International Conference on Big Data*, Santa Clara, 25-32 (2013).
22. Varemo, L., Gatto, F. and Nielsen, J., "Kiwi: a tool for integration and visualization of network topology and gene-set analysis", *Bmc Bioinformatics*, 15: 408 (2014).
23. Csardi, G. and Nepusz, T., "The igraph software package for complex network research", *InterJournal*, Complex Systems: 1695 (2006).
24. Internet: Sigma.js Project Page, "Sigma is a JavaScript library", <http://sigmajs.org> (2016).
25. Lopes, C. T., Franz, M., Kazi, F., Donaldson, S. L., Morris, Q. and Bader, G. D., "Cytoscape Web: an interactive web-based network browser", *Bioinformatics*, 26 (18): 2347-2348 (2010).
26. Franz, M., Lopes, C. T., Huck, G., Dong, Y., Sümer, S. O. and Bader, G. D., "Cytoscape.js: a graph theory library for visualisation and analysis.", *Bioinformatics*, 32 (2): 309-311 (2016)

27. Jacomy, M., Venturini, T., Heymann, S. and Bastian, M., "ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software.", *PLoS ONE*, 9 (6): 1-12 (2014).
28. Heymann, S. and Grand, B. L., "Visual Analysis of Complex Networks for Business Intelligence with Gephi", *17th International Conference on Information Visualisation*, London, 307-312 (2013).
29. Muhongya, K. V. and Maharaj, M. S., "Visualising and analysing online social networks", *Computing, Communication and Security (ICCCS), 2015 International Conference*, Pamplermousses, 1-6 (2015).
30. Bravo, R. B., Del Valle, M. E. and Gavidia, A. R., "A multilayered analysis of polarization and leaderships in the Catalan Parliamentarians' Twitter Network", *Advances in ICT for Emerging Regions (ICTer), 2015 Fifteenth International Conference*, Colombo, 200-206 (2015).
31. Akhtar, N., "Social Network Analysis Tools", *Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference*, Bhopal, 389-392 (2014).
32. Hagberg, A. A., Schult, D. A. and Swart, P. J., "Exploring network structure, dynamics, and function using NetworkX", *7th Python in Science conference (SciPy 2008)*, Pasedena CA USA, 11-15 (2008).
33. Batagelj, V. and Mrvar, A., "Program for Analysis and Visualization of Large Networks", *Reference Manual*, Ljubljana, (2011).
34. Ognyanova, K., "Network visualization with R", *POLNET 2015 Workshop*, Portland OR (2015).
35. Ognyanova, K., "Network Analysis and Visualization with R and igraph", *NetSciX 2016 School of Code Workshop*, Wroclaw (2016).
36. Internet: Cambridge Networks Network, "List of Resources for Complex Network Analysis", <http://www.cnn.group.cam.ac.uk/Resources> (2015).
37. Internet: Netminer Software Tool, "Netminer product overview", <http://www.netminer.com/product/overview.do> (2000).
38. Adar, E., "Guess: a language and interface for graph exploration", *In CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, Montreal (2006).
39. Internet: Wikipedi, "Social Networks Analysis Software" https://en.wikipedia.org/wiki/Social_network_analysis_software (2016).

40. Min, S., Kaburagi, M., Aoyama, Y., and Min, K., "WEB Based Learning System for Complex Networks", *2006 7th International Conference on Information Technology Based Higher Education and Training*, Sydney, 557-562 (2006).
41. İnternet: Linkurious Visualize Data Easily, "Linkrious.js", <http://linkurio.us/> (2013)
42. İnternet: Graphit Project Page, "3D graph visualization with WebGL / Three.js", <http://graphit-live.herokuapp.com/> (2016).
43. İnternet: Sigma.js Project Collaboration Page, "Sigma.js User Manual ", <https://github.com/jacomyal/sigma.js/wiki> (2016).
44. İnternet: Wikipedia, "Poisson Dağılımı", https://tr.wikipedia.org/wiki/Poisson_dağılımı (2016).
45. Dirksen, J., "Learning Three.js: The JavaScript 3D Library for WebGL", *Packt Publishing*, Birmingham, 7-35 (2013).
46. İnternet: jQuery Project Page, "jQuery, a JavaScript library", <https://jquery.com/> (2016).
47. İnternet: jQuery UI Project Page, "jQuery UI, a JavaScript library", <https://jqueryui.com/> (2016).
48. İnternet: jqPlot Project Page, "jqPlot is a Javascript library for plotting", <http://www.jqplot.com/> (2016).
49. İnternet: FileSaver.js Project Collaboration Page, "FileSaver.js", <https://github.com/eligrey/FileSaver.js> (2016).
50. Sears, R., Van Ingen, C., and Gray, J., "To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?.", *CoRR*, abs/cs/0701168, (2007).
51. İnternet: Ben Chung Site, "Trackball Controls in three.js", <http://benchung.com/trackball-controls-three-js/> (2015).
52. Six, J. M. and Tollis, I. G., "Circular Drawing Algorithms.", "Handbook of graph drawing and visualization", Roberto Tamassia (Ed.), *CRC Press*, London, 285-315 (2007).
53. Kobourov, S., "Spring embedders and force directed graph drawing algorithms", *CoRR*, abs/1201.3011, (2012).
54. İnternet: Philip Bjorge Site, "3d Fruchterman Reingold Algorithm?.". <http://philipbjorge.com/2011/11/11/3d-fruchterman-reingold/> (2016).
55. Brandes, U., "A Faster Algorithm for Betweenness Centrality", *Journal of Mathematical Sociology*, 25 (2): 63-177 (2001).



EK AÇIKLAMALAR A.

MODELLER ARASI DÖNÜŞÜMLER MATLAB KODLARI

```

%1 Nolu Deneme (ER to RG Model)
nodes=zeros(1,1000);
edges=[];
degrees=zeros(1,1000);
for i=1:1000;
nodes(i)=i;
degrees(i)=0;
end

%initialize
link=1;
nodecount=length(nodes);
%wtncount=4;
rnprob=0.01;
disp(nodecount);
while link~=round((nodecount*(nodecount-1)/2)*rnprob)+1
x=round(1+(nodecount-1)*rand());
y=round(1+(nodecount-1)*rand());
if x~=y && x~=0 && y~=0 && controlnode(x,y,edges)~=1
edges(link,1)=x;
edges(link,2)=y;
degrees(x)=degrees(x)+1;
degrees(y)=degrees(y)+1;
link=link+1;
end

end

t=0;
x=0.0;
prob=0.9857;

narrowfact=0.5;
for r=1:link-1;
x=rand();
if x<=prob;

    alfa=1;
    alfa2=1;
    shift=round(nodecount/2);
    while 1
    alfa=poissrnd(shift);
    delta=alfa-shift;
    delta=delta*narrowfact;
    alfa2=edges(r,1)+round(delta);

    t=mod(alfa2,nodecount);
    if(t==0)
        t=nodecount;
    end

    if controlnode(edges(r,1),t,edges)~=1 && edges(r,1)~=t
        disp([num2str(r),'-',num2str(edges(r,1)),'-',num2str(t)]);
        degrees(edges(r,2))=degrees(edges(r,2))-1;
        edges(r,2)=t;
        degrees(t)=degrees(t)+1;
        break;

```

```
        end
    end

end
end

o=adjacent_sort(edges,nodecount,link);

k=BFS(nodecount,o);
b=APL(nodecount,k);

che=CHeap(nodecount,o);
averagecoeff=sum(che)/nodecount;

figure;
hist(degrees);
```



```

%2 Nolu Deneme (RG to BA Model)
nodes=zeros(1,1000);
edges=[];
degrees=zeros(1,1000);
for i=1:1000;
nodes(i)=i;
degrees(i)=0;
end

%initialize
link=1;
nodecount=length(nodes);
wtncount=10;
disp(nodecount);
for g=1:nodecount;
    for j1=1:wtncount/2;
        dene=0;

        if mod(g+j1,1000)==0
            dene=1000;
        else
            dene=mod(g+j1,1000);
        end

        edges(link,1)=g;
        edges(link,2)=dene;
        degrees(g)=degrees(g)+1;
        degrees(dene)=degrees(dene)+1;
        link=link+1;
    end
end

work=edges;

t=0;
a=0.0;
prewire=0.1194;
for i=1:1;
for r=1:link-1;
a=rand();
if a<=prewire;

    while 1
        t=round(1+(link-2)*rand());
        c=round(1+rand());

        if controlnode(edges(r,1),edges(t,c),edges)~=1 &&
edges(r,1)~=edges(t,c)
            disp([num2str(r),'-',num2str(edges(r,1)),'-',
num2str(edges(t,c))]);
            degrees(edges(r,2))=degrees(edges(r,2))-1;
            edges(r,2)=edges(t,c);
            degrees(edges(t,c))=degrees(edges(t,c))+1;
            break;
        end
    end
end
end

```

```
end

end
end
disp('burada');
o=adjacent_sort(edges,nodecount,link);

k=BFS(nodecount,o);
b=APL(nodecount,k);

che=CHear(nodecount,o);
averagecoeff=sum(che)/nodecount;

figure;
hist(degrees);
```



```

%3 Nolu Deneme (BA to RG Model)
nodes=[];
initialnodes=5;
linkperstep=5;
iterationsize=1000;
edges=[];
degrees=[];
degrees=zeros(1,5);

%initialize
link=0;

%starting setup
for j=1:initialnodes;
    while 1
        x=round(1+((5-1)*rand()));
        y=round(1+((5-1)*rand()));
        z=round(1+rand());
        if controlnode(x,y,edges)~=1 && x~=y
            edges(j,z)=x;
            edges(j,3-z)=y;
            nodes(j)=j;
            degrees(x)=degrees(x)+1;
            degrees(y)=degrees(y)+1;
            link=link+1;
            break;
        end
    end
end
for k=6:iterationsize+5;
    degrees(k)=0;
    nodes(k)=k;
    for o=1:linkperstep;
        while 1
            x=round(1+((link-1)*rand()));
            y=round(1+((2-1)*rand()));
            z=round(1+rand());
            if controlnode(k,edges(x,y),edges)~=1 && edges(x,y)~=k
                g=size(edges,1)+1;
                edges(g,z)=k;
                edges(g,3-z)=edges(x,y);
                degrees(k)=degrees(k)+1;
                degrees(edges(x,y))=degrees(edges(x,y))+1;
                link=link+1;
                break;
            end
        end
    end
end

t=0;
x=0.0;
prob=0.9996;
nodecount=length(nodes);
narrowfact=0.5;
for tekrar=1:1;

```

```

for r=1:link;
x=rand();
if x<=prob;

    alfa=1;
    alfa2=1;
    shift=round(nodecount/2);
    while 1
    alfa=poissrnd(shift);
    delta=alfa-shift;
    delta=delta*narrowfact;
    alfa2=edges(r,1)+round(delta);

    t=mod(alfa2,nodecount);
    if(t==0)
        t=nodecount;
    end

    if controlnode(edges(r,1),t,edges)~=1 && edges(r,1)~=t
        disp([num2str(r),'-',num2str(edges(r,1)),'-', num2str(t)]);
        degrees(edges(r,2))=degrees(edges(r,2))-1;
        edges(r,2)=t;
        degrees(t)=degrees(t)+1;

        break;
    end

end

end

end

end

o=adjacent_sort(edges,nodecount,link);

k=BFS(nodecount,o);
b=APL(nodecount,k);

che=CHearp(nodecount,o);
averagecoeff=sum(che)/nodecount;

figure;
hist(degrees);

```

```

%4 nolu deneme(ER to BA Model)
nodes=zeros(1,1000);
edges=[];
degrees=zeros(1,1000);
for i=1:1000;
nodes(i)=i;
degrees(i)=0;
end

%initialize
link=1;
nodecount=length(nodes);
%wtncount=4;
rnprob=0.01;
disp(nodecount);
while link~=round((nodecount*(nodecount-1)/2)*rnprob)+1
x=round(1+(nodecount-1)*rand());
y=round(1+(nodecount-1)*rand());
if x~=y && x~=0 && y~=0 && controlnode(x,y,edges)~=1
edges(link,1)=x;
edges(link,2)=y;
degrees(x)=degrees(x)+1;
degrees(y)=degrees(y)+1;
link=link+1;
end

end

t=0;
a=0.0;
prewire=1.0;
for j=1:1;

for r=1:link-1;
a=rand();
if a<=prewire;

while 1
t=round(1+(link-2)*rand());
c=round(1+rand());

if controlnode(edges(r,1),edges(t,c),edges)~=1 &&
edges(r,1)~=edges(t,c)
disp([num2str(r),'-',num2str(edges(r,1)),'-',
num2str(edges(t,c))]);
degrees(edges(r,2))=degrees(edges(r,2))-1;
edges(r,2)=edges(t,c);
degrees(edges(t,c))=degrees(edges(t,c))+1;
break;
end
end

end

end

end
end
end

```



```
o=adjacent_sort(edges,nodecount,link);  
  
k=BFS(nodecount,o);  
b=APL(nodecount,k);  
  
che=CHeap(nodecount,o);  
averagecoeff=sum(che)/nodecount;  
  
figure;  
hist(degrees);
```



```

%5 Nolu Deneme (BA to ER Model)
nodes=[];
initialnodes=5;
linkperstep=5;
iterationsize=1000;
edges=[];
degrees=[];
degrees=zeros(1,5);

%initialize
link=0;

%starting setup
for j=1:initialnodes;
    while 1
        x=round(1+(5-1)*rand());
        y=round(1+(5-1)*rand());
        z=round(1+rand());
        if controlnode(x,y,edges)~=1 && x~=y
            edges(j,z)=x;
            edges(j,3-z)=y;
            nodes(j)=j;
            degrees(x)=degrees(x)+1;
            degrees(y)=degrees(y)+1;
            link=link+1;
            break;
        end
    end
end
for k=6:iterationsize+5;
    degrees(k)=0;
    nodes(k)=k;
    for o=1:linkperstep;
        while 1
            x=round(1+(link-1)*rand());
            y=round(1+(2-1)*rand());
            z=round(1+rand());
            if controlnode(k,edges(x,y),edges)~=1 && edges(x,y)~=k
                g=size(edges,1)+1;
                edges(g,z)=k;
                edges(g,3-z)=edges(x,y);
                degrees(k)=degrees(k)+1;
                degrees(edges(x,y))=degrees(edges(x,y))+1;
                link=link+1;
                break;
            end
        end
    end
end

t=0;
c=0;
x=0.0;
prob=0.4924;
nodecount=length(nodes);

```

```

for r=1:link;
x=rand();
if x<=prob;

    while 1
    t=round(1+((nodecount-1)*rand()));

    if controlnode(t,edges(r,1),edges)~=1 && edges(r,1)~=t
        disp([num2str(r), '-', num2str(edges(r,1)), '-', num2str(t)]);
        degrees(edges(r,2))= degrees(edges(r,2))-1;
        edges(r,2)=t;
        degrees(t)=degrees(t)+1;
        break;
    end

end

end

end
disp('burada');
o=adjacent_sort(edges,nodecount,link);

k=BFS(nodecount,o);
b=APL(nodecount,k);

che=CHear(nodecount,o);
averagecoeff=sum(che)/nodecount;

figure;
hist(degrees);

```

```

%6 Nolu Deneme (ER to RG_BA Model)
N=1000;
nodes=zeros(1,N);
edges=[];
degrees=zeros(1,N);
for i=1:N;
nodes(i)=i;
degrees(i)=0;
end

%initialize
link=1;
rnprob=0.01;
prob=0.9999;
narrowfact=0.5;

disp(N);
while link~=round((N*(N-1)/2)*rnprob)+1
x=round(1+(N-1)*rand());
y=round(1+(N-1)*rand());
if x~=y && x~=0 && y~=0 && controlnode(x,y,edges)~=1
edges(link,1)=x;
edges(link,2)=y;
degrees(x)=degrees(x)+1;
degrees(y)=degrees(y)+1;
link=link+1;
end
end

for tekrar=1:1;
t=0;
x=0.0;

for r=1:link-1; %for r=1:link-1;
x=rand();
if x<=prob;
Serit=[edges(:,1); edges(:,2)];
Serit=sort(Serit);
L=length(Serit);

shift=round(L/2);
while 1
alfa=poissrnd(shift);
indis1=sum(degrees(1:edges(r,1)-1))+1; % örn. 5. düğümün
şeritteki başlangıç konumu, sum(degrees(1:4)+1)
indis2=sum(degrees(1:edges(r,1))); % örn. 5. düğümün
şeritteki bitiş konumu, sum(degrees(1:5))
delta=alfa-shift;
delta=round(delta*narrowfact);

if mod(indis1+delta,L)==0 || mod(indis2+delta,L)==0;
delta=delta-1; end;

if delta<=0;

```

```

        newconnect=Serit(mod(L+indis1+delta,L)); % L+ nin
sebebi negatif sayılardan kurtulmak // zaten modu alınıyor
    else
        newconnect=Serit(mod(L+indis2+delta,L));
    end

    t=mod(newconnect,N);
    if(t==0);t=N;end;

    if controlnode(edges(r,1),t,edges)~=1 && edges(r,1)~=t
        disp([num2str(r),'-',num2str(edges(r,1)),'-',
num2str(t)]);
        degrees(edges(r,2))=degrees(edges(r,2))-1;
        edges(r,2)=t;
        degrees(t)=degrees(t)+1;
        break;
    end
end
end
end
end

o=adjacent_sort(edges,N,link);

k=BFS(N,o);
b=APL(N,k);

che=CHearp(N,o);
averagecoeff=sum(che)/N;

figure;
hist(degrees,15);

```



EK AÇIKLAMALAR B.

WEB TABANLI UYGULAMA YAZILIM KOD ÖRNEKLERİ

```

/**
 * Created by gokhan on 09.04.2015.
 * Kompleks Ağ İçin Veri Tipi Tanımlamaları ve Fonksiyonlar
 */

var graph={
  nodes:[],
  edges:[],
  degree:[],
  degreedist:[],
  fr:[]
};

function addnode(i,xcoordinate,ycoordinate,zcoordinate){
  graph.nodes.push({
    id:i,
    label:'Node'+i,
    x:xcoordinate,
    y:ycoordinate,
    z:zcoordinate
  });
  graph.fr.push({id:i,
    x:0,
    y:0,
    z:0}
  );
}
function addedge(z,Esource,Etargert)
{
  graph.edges.push({
    id:z,
    source:Esource,
    target:Etargert
  });
}
function degreeformat(n)
{
  for (var f=0;f<n;f++)
  {
    graph.degree[f]=0;
    graph.degreedist[f]=0;
  }
}
function control_node(g,f)
{
  var ct=0;
  for (var t=0;t<graph.edges.length;t++)
  {
    if((graph.edges[t].source==g && graph.edges[t].target==f) ||
(graph.edges[t].source==f && graph.edges[t].target==g))
    {
      ct=1;
    }
  }
  return ct;
}

```

```

/* Derece Dağılımının Çizdirilme Fonksiyonu*/
function degreedist_apply()
{
    var ticks=[];

    for(var k=0;k<graph.nodes.length;k++)
    {
        graph.degreedist[graph.degree[k]]++;

    }

    var frd=0;
    for (var gt=0;gt<=maxdegree;gt++)
    {
        if(gt==0)
        {
            ticks[gt]='0';
        }
        else{
            ticks[gt]=gt;
        }
    }

    $.jqplot.config.enablePlugins = true;
    var plot1 = $.jqplot('graphdist',
[graph.degreedist], {
    // Only animate if we're not using excanvas (not
in IE 7 or IE 8)..
    animate: !$jqplot.use_excanvas,
    title:'Degree Distribution',
    seriesDefaults:{
        renderer:$.jqplot.BarRenderer,
        pointLabels: { show: true },
        rendererOptions: {
            // Set varyBarColor to true to use the
custom colors on the bars.
            varyBarColor: true
        }
    }
},
    axes: {
        xaxis: {
            renderer: $.jqplot.CategoryAxisRenderer,
            ticks: ticks,
            tickRenderer:
$.jqplot.CanvasAxisTickRenderer,
            tickOptions: {
                angle: -90,
                fontSize: '10pt',
                min:0
            }
        }
    }
},
    },
    },

```



```
        highlighter: { show: false }  
    });  
    $("#graph-properties").dialog({  
        height: 275,  
        width: 550  
  
    });  
}
```



```

/* Fruchterman-Reingold Yerleşim Düzeni Fonksiyonu*/
function yerlestir()
{

    var wth=60;
    var ht=30;
    var area=wth*ht;
    var k=Math.sqrt(area/graph.nodes.length);
    var temperature;
    var Epsilon=0.00000001;
    var iterationsize=25;
    var key=0;

    for (var t=0;t<iterationsize;t++){

        temperature=Math.pow(0.9,t);

        //calculate repulsion forces
        for (var j=0;j<graph.nodes.length;j++)
        {
            graph.fr[j].x=0;
            graph.fr[j].y=0;
            if(threedstate==1){
                graph.fr[j].z=0;}
            for (var hy=0;hy<graph.nodes.length;hy++)
            {
                if(j!=hy)
                {
                    var delta_x=graph.nodes[j].x-
graph.nodes[hy].x;
                    var delta_y=graph.nodes[j].y-
graph.nodes[hy].y;

                    var
deltar=Math.max(Epsilon,Math.sqrt((delta_x*delta_x)+(delta_y*delta_y
)));
                    if(threedstate==1) {
                        var delta_z = graph.nodes[j].z -
graph.nodes[hy].z;
                        var deltar_z =Math.max(Epsilon,
Math.sqrt((delta_y * delta_y) + (delta_z * delta_z)));
                        graph.fr[j].z+=(delta_z/deltar_z)*repulsionforce(deltar_z,k);
                    }

                    graph.fr[j].x+=(delta_x/deltar)*repulsionforce(deltar,k);
                    graph.fr[j].y+=(delta_y/deltar)*repulsionforce(deltar,k);

                }
            }
        }
    }
}

```

```

//calculate attractionforce
for(var h=0;h<graph.edges.length;h++)
{
    var
delta_x=graph.nodes[graph.edges[h].source].x-
graph.nodes[graph.edges[h].target].x;
    var
delta_y=graph.nodes[graph.edges[h].source].y-
graph.nodes[graph.edges[h].target].y;
    if(threedstate==1)
    {
        var
delta_z=graph.nodes[graph.edges[h].source].z-
graph.nodes[graph.edges[h].target].z;
        var
deltaz=Math.max(Epsilon,Math.sqrt((delta_z*delta_z)+(delta_y*delta_y
)));
        graph.fr[graph.edges[h].source].z-
=(delta_z/deltaz)*attractionforce(deltaz,k);

graph.fr[graph.edges[h].target].z+=(delta_z/deltaz)*attractionforce(
deltaz,k);
    }
    var
delta=Math.max(Epsilon,Math.sqrt((delta_x*delta_x)+(delta_y*delta_y
)));

    graph.fr[graph.edges[h].source].x-
=(delta_x/delta)*attractionforce(delta,k);
    graph.fr[graph.edges[h].source].y-
=(delta_y/delta)*attractionforce(delta,k);

graph.fr[graph.edges[h].target].x+=(delta_x/delta)*attractionforce(d
elta,k);

graph.fr[graph.edges[h].target].y+=(delta_y/delta)*attractionforce(d
elta,k);

}
//calculate position
for (var g=0;g<graph.nodes.length;g++)
{
    var
delta_tmp=Math.max(Epsilon,Math.sqrt((graph.fr[g].x*graph.fr[g].x)+(
graph.fr[g].y*graph.fr[g].y)));
    if(threedstate==1)
    {
        var
delta_tmp_z=Math.max(Epsilon,Math.sqrt((graph.fr[g].y*graph.fr[g].y
)+(graph.fr[g].z*graph.fr[g].z)));

graph.nodes[g].z+=(graph.fr[g].z/delta_tmp_z)*Math.min(temperature,d
elta_tmp_z);

geometry.vertices[graph.nodes[g].id].z=graph.nodes[g].z;

//textgeometry[graph.nodes[g].id].position.z=graph.nodes[g].z;

```

```

    }

    graph.nodes[g].x+=(graph.fr[g].x/delta_tmp)*Math.min(temperature,delta_tmp);

    graph.nodes[g].y+=(graph.fr[g].y/delta_tmp)*Math.min(temperature,delta_tmp);

    geometry.vertices[graph.nodes[g].id].x=graph.nodes[g].x;
    geometry.vertices[graph.nodes[g].id].y=graph.nodes[g].y;
}

}

    var i=0;
    for (var tq=0;tq<graph.edges.length;tq++)
    {

    geometries.vertices[i].set(geometry.vertices[graph.edges[tq]["source"]].x,geometry.vertices[graph.edges[tq]["source"]].y,geometry.vertices[graph.edges[tq]["source"]].z);
        i++;

    geometries.vertices[i].set(geometry.vertices[graph.edges[tq]["target"]].x,geometry.vertices[graph.edges[tq]["target"]].y,geometry.vertices[graph.edges[tq]["target"]].z);
        i++;

    }

}
}

```

```

/* Ortalama Uzaklıkları Hesaplayan Fonksiyon*/
function Average_Distance() {
    var AD=0;
    //var
count=1/((Node_level.length*(Node_level.length-1))/2);
    for (var g=0;g<Node_level.length;g++)
    {
        for (var t=g+1;t<Node_level.length;t++)
        {
            if (Node_level[g][t]>0)
            {
                AD+=Node_level[g][t];
                this.rg=rg+1;
            }
        }
    }
}

return AD/this.rg;
}

//Local Clustering, Yerel Kümelenmeler
var CLocal=[];
function Local_Clustering(Node)
{
    var ei=0;
    var ki=adjacent_array[Node].length;
    if (ki>1)
    {
        var f="";
        for (var tr=0;tr<ki;tr++)
        {
            var hy=tr+1;
            while(hy<ki) {

                //alert(adjacent_triple[Node][tr]+"
                "+adjacent_triple[Node][hy]+"
                "+Node_level[adjacent_triple[Node][tr]][adjacent_triple[Node][hy]]);

                if(Node_level[adjacent_array[Node][tr]][adjacent_array[Node][hy]]==1
                )
                {
                    ei++;
                }
                hy++;
            }
        }
    }
    CLocal[Node]=2*ei/(ki*(ki-1));
}
else{
    CLocal[Node]=0;
}
}

```

```

    }
}

/*Breadth First Search Algoritması ile D ğ mler Arası
Uzaklıkların Bulunması*/
function BFS ()
{
    Node_level=Create2DArray(graph.nodes.length);
    var kuyruk=[];

    for (var y=0;y<graph.nodes.length;y++)
    {
        for (var z=0;z<graph.nodes.length;z++)
        {
            Node_level[y][z]=-1;
        }
    }

    for (var
Noodle=0;Noodle<graph.nodes.length;Noodle++)
    {
        Node_level[Noodle][Noodle]=0;
        kuyruk.push(Noodle);
        while(kuyruk.length!=0)
        {
            var v=kuyruk.shift();
            for (var r=0;r<adjacent_array[v].length;r++)
            {
                if(Node_level[Noodle][adjacent_array[v][r]]==-1)
                {
                    kuyruk.push(adjacent_array[v][r]);

Node_level[Noodle][adjacent_array[v][r]]=Node_level[Noodle][v]+1;

//Node_level[adjacent_array[v][r]][Noodle]=Node_level[Noodle][v]+1;
                }
            }
        }
    }
}

}

}

}

}

var triple_count=0;
function find_triple ()
{
    for(var s=0;s<graph.edges.length;s++)
    {
        for(var t=s+1;t<graph.edges.length;t++) {

if(graph.edges[s]["source"]==graph.edges[t]["source"]

```

```
||graph.edges[s]["source"]==graph.edges[t]["target"] ||  
graph.edges[s]["target"]==graph.edges[t]["source"]  
||graph.edges[s]["target"]==graph.edges[t]["target"])  
    {  
        triple_count++;  
    }
```

```
    }
```

```
    }
```

```
    }
```



```

/* Poisson Dağılımı Rastsal Sayı Üreten Fonksiyon*/
function GetPoisson(lambda)
{
    if(lambda<15)
    {
        //small Lambda Count
        var p = 1.0;
        var L = Math.exp(-lambda);

        var k = 0;
        do
        {
            k++;

            p *= Math.random();
        }
        while (p > L);
        return k - 1;
    }
    else{

        //big lambda count
        var c = 0.767 - 3.36 / lambda;
        var beta = Math.PI / Math.sqrt(3.0 * lambda);
        var alpha = beta * lambda;
        var k = Math.log(c) - lambda - Math.log(beta);

        while(true)
        {
            var u = Math.random();
            var x = (alpha - Math.log((1.0 - u) / u)) /
beta;

            var n = Math.floor(x + 0.5);
            if (n < 0)
                continue;
            var v = Math.random();
            var y = alpha - beta * x;
            var temp = 1.0 + Math.exp(y);
            var lhs = y + Math.log(v / (temp * temp));
            var rhs = k + n * Math.log(lambda) -
LogFactorial(n);

            if (lhs <= rhs)
            {

                return n;
            }
        }
    }
}
}
}
}

```



```

/*5 Nolu Deneme (ER to RG_BA Model Transition and
Visualization*/

function init_rn_rg_ba(){

    var container, separation = 100, amountX = 50,
amountY = 50, particles, particle;

    container = document.createElement('div');
document.body.appendChild(container);

    camera = new THREE.PerspectiveCamera( 40,
window.innerWidth / window.innerHeight, 1, 100000 );
/*camera = new THREE.OrthographicCamera( width / -
2, width / 2, height / 2, height / - 2, 1, 2000 );*/
camera.position.z = 125;
controls = new THREE.TrackballControls(camera);

controls.rotateSpeed = 0.5;
controls.zoomSpeed = 5.2;
controls.panSpeed = 1;
controls.noZoom = false;
controls.noPan = false;
controls.staticMoving = false;
controls.dynamicDampingFactor = 0.3;
controls.keys = [ 65, 83, 68 ];
controls.addEventListener('change', render);

scene = new THREE.Scene();

    renderer = new THREE.WebGLRenderer({ alpha: true,
preserveDrawingBuffer: true, antialias: true });
renderer.setPixelRatio( window.devicePixelRatio );
renderer.setSize( window.innerWidth,
window.innerHeight );
container.appendChild( renderer.domElement );

// particles
degreeformat(this.NodeCount);

    var i=0;
do
{

    var geo=new THREE.Geometry();
var x=(Math.random() * this.NodeCount) | 0);
var y=(Math.random() * this.NodeCount) | 0);
if(x!=y && control_node(x,y)!=1)
{

        addedge(i,x,y);
graph.degree[x]++;
graph.degree[y]++;
i++;
}
}

```

```

    }
    while(i!=Math.floor(this.RNProb*(this.NodeCount*(this.NodeCount-
1)/2)))

        var a,t,c,L;
        var Strip=[];
        var indis1,indis2;
        var alfa,delta,shift;

        for (var k=0;k<i;k++)
        {
            a=Math.random();
            if(a<=this.PRewire){
                Strip=SortStripArray();
                L=Strip.length;
                shift=Math.floor(L/2);
                var newconnect;
                while(1)
                {
                    alfa=GetPoisson(shift);

indis1=Strip.indexOf(graph.edges[k].source);

indis2=indis1+graph.degree[graph.edges[k].source]-1;
                    delta=alfa-shift;
                    delta=Math.floor(delta*this.NarrowFactor);
                    if((delta+indis1)%L==0 ||
(delta+indis2)%L==0)
                    {
                        delta=delta-1;
                    }
                    if(delta<=0)
                    {
                        newconnect=Strip[(L+indis1+delta)%L];
                    }
                    else{
                        newconnect=Strip[(L+indis2+delta)%L];
                    }
                }
                t=Math.abs(newconnect%this.NodeCount);

if(control_node(graph.edges[k].source,t)!=1 &&
graph.edges[k].source!=t)
                {
                    graph.degree[graph.edges[k].target]-
-;
                    graph.edges[k].target=t;
                    graph.degree[t]++;
                    break;
                }
            }
        }
    }
}

```

```

    }

    var material;
    var geom;
    for (var h=0;h<nodescolor.length;h++)
    {
        material = new THREE.MeshBasicMaterial( {color:
nodescolor[h]} );
        //geom = new
THREE.SphereGeometry(nodesradius[h],32,32 );
        //nodesgeometry.push(geom);
        nodesmaterial.push(material);
    }
    geometry = new THREE.Geometry();

    area=2000;
    for ( var j = 0; j < this.NodeCount; j ++ ) {
        geom = new
THREE.SphereGeometry(0.2+Math.log(graph.degree[j]+1),32,32 );

        particle = new THREE.Mesh(
geom,nodesmaterial[graph.degree[j]%50] );
        particle.position.x = Math.floor(Math.random() *
(area + area + 1) - area);
        particle.position.y = Math.floor(Math.random() *
(area + area + 1) - area);
        if(threedstate==1)
        {
            particle.position.z =
Math.floor(Math.random() * (area + area + 1) - area);
        }
        else {
            particle.position.z = 0;
        }
        particle.position.normalize();
        particle.position.multiplyScalar(40);
        particle.scale.x = particle.scale.y = 1;
        scene.add( particle );

        addnode(j,particle.position.x,particle.position.y,particle.position.
z);

        geometry.vertices.push( particle.position );
    }
    geometries = new THREE.Geometry();

    for (var u=0;u<i;u++) {

        var geo=new THREE.Geometry();
        var vertex = new
THREE.Vector3(geometry.vertices[graph.edges[u]["source"]].x,geometry
.vertices[graph.edges[u]["source"]].y,geometry.vertices[graph.edges[
u]["source"]].z);

```

```

        var vertex2 = new
THREE.Vector3(geometry.vertices[graph.edges[u]["target"]].x,geometry
.vertices[graph.edges[u]["target"]].y,geometry.vertices[graph.edges[
u]["target"]].z);
        geo.vertices.push(vertex);
        geo.vertices.push(vertex2);

        THREE.GeometryUtils.merge( geometries, geo);

    }
    var ju=0;
    for (var i = 0; i < geometries.vertices.length; i+=2)
{
        var tu=0;

    if(graph.degree[graph.edges[ju]["source"]]>=graph.degree[graph.edges
[ju]["target"]])
        {
            tu=graph.edges[ju]["source"];
        }
        else{
            tu=graph.edges[ju]["target"];
        }
        geometries.colors[i] = new
THREE.Color(nodescolor[graph.degree[tu]%50]);
        geometries.colors[i+1] = geometries.colors[i];
        ju++;
    }

        var material = new THREE.LineBasicMaterial( {color:
0xffffffff,vertexColors: THREE.VertexColors} );
        var edgesGeo = new THREE.Line(geometries,material,
THREE.LinePieces);
        edgesGeo.frustumCulled=false;
        scene.add(edgesGeo);
        maxdegree=0;
        for (var gy=0;gy<graph.nodes.length;gy++)
        {
            if(graph.degree[gy]>maxdegree)
            {
                maxdegree=graph.degree[gy];
            }
        }
    }

    StartingValues[0]="Random Network to Regular & Barabasi
Albert Model Transition";
    StartingValues[1]="Networks Starting Parameter";
    StartingValues[2]="Node Count:"+NodeCount.toString();
    StartingValues[3]="RN Prob:"+RNProb.toString();
    StartingValues[4]="Prewire:"+Prewire.toString();
    StartingValues[5]="Narrow
Factor:"+NarrowFactor.toString();
    this.created=1;

}

```

```

/* Watt-Strogatz Model Visualization*/
function init_ws(){
    var container, separation = 100, amountX = 50,
amountY = 50, particles, particle;

    container = document.createElement('div');
    document.body.appendChild(container);

    camera = new THREE.PerspectiveCamera( 40,
window.innerWidth / window.innerHeight, 1, 100000);
    /*camera = new THREE.OrthographicCamera( width / -
2, width / 2, height / 2, height / - 2, 1, 2000 );*/
    camera.position.z = 125;
    controls = new THREE.TrackballControls(camera);

    controls.rotateSpeed = 0.5;
    controls.zoomSpeed = 5.2;
    controls.panSpeed = 1;
    controls.noZoom = false;
    controls.noPan = false;
    controls.staticMoving = false;
    controls.dynamicDampingFactor = 0.3;
    controls.keys = [ 65, 83, 68 ];
    controls.addEventListener('change', render);

    scene = new THREE.Scene();

    renderer = new THREE.WebGLRenderer({ alpha:
true, antialias: true, preserveDrawingBuffer: true });
    renderer.setPixelRatio( window.devicePixelRatio );
    renderer.setSize( window.innerWidth,
window.innerHeight );
    container.appendChild( renderer.domElement );

    degreeformat(wtcount);

    var link=0;
    for (var i=0;i<wtcount;i++)
    {
        var target=-1;
        for (var j=1;j<=Wtnccount/2;j++)
        {
            target=(j+i)%wtcount;
            addedge(link,i,target);
            graph.degree[i]++;
            graph.degree[target]++;
            link++;
        }
    }

    var t,x;
    for (var r=0;r<link;r++)
    {
        x=Math.random();
        if(x<=Probwt){
            do{

```

```

        t=Math.floor((Math.random() * wtcount));

}while(control_node(graph.edges[r].source,t)==1 ||
graph.edges[r].source==t)

        graph.degree[graph.edges[r].target]--;
        graph.edges[r].target=t;
        graph.degree[t]++;

    }

}

//particles

        geometry = new THREE.Geometry();
        var material;
        var geom;
        //var ematerial;
        for (var h=0;h<nodescolor.length;h++)
        {
            material = new THREE.MeshBasicMaterial( {color:
nodescolor[h]} );
            //geom = new
THREE.SphereGeometry(nodesradius[h],32,32 );
            //nodesgeometry.push(geom);
            nodesmaterial.push(material);

        }

        for (var i=0;i<wtcount;i++)
        {
            geom = new
THREE.SphereGeometry(0.2+Math.log(graph.degree[i]+1),32,32 );
            particle = new THREE.Mesh(
geom,nodesmaterial[graph.degree[i]%50] );
            particle.position.x = Math.cos(2 * i * Math.PI /
wtcount);
            particle.position.y = Math.sin(2 * i * Math.PI /
wtcount);
            if(threedstate==1)
            {
                particle.position.z = Math.tan(2 * i * Math.PI /
wtcount);
            }
            else{
                particle.position.z =0;
            }
            particle.position.normalize();
            particle.position.multiplyScalar(40);
            particle.scale.x = particle.scale.y = 1;
            scene.add( particle );

addnode(i,particle.position.x,particle.position.y,particle.position.
z);
            geometry.vertices.push( particle.position );

```

```

    }

    geometries = new THREE.Geometry();

    for (var u=0;u<graph.edges.length;u++) {

        var geo=new THREE.Geometry();
        var vertex = new
THREE.Vector3(geometry.vertices[graph.edges[u]["source"]].x,geometry
.vertices[graph.edges[u]["source"]].y,geometry.vertices[graph.edges[
u]["source"]].z);
        var vertex2 = new
THREE.Vector3(geometry.vertices[graph.edges[u]["target"]].x,geometry
.vertices[graph.edges[u]["target"]].y,geometry.vertices[graph.edges[
u]["target"]].z);
        geo.vertices.push(vertex);
        geo.vertices.push(vertex2);
        //var line = new THREE.Line( geo,
edgesmaterial[graph.edges[u]["source"]%18]);
        //scene.add(line);
        //geometries.push(geo);
        THREE.GeometryUtils.merge( geometries, geo);

    }
    var ju=0;
    for (var i = 0; i < geometries.vertices.length; i+=2)
{

        var tu=0;

        if(graph.degree[graph.edges[ju]["source"]]>=graph.degree[graph.edges
[ju]["target"]])
        {
            tu=graph.edges[ju]["source"];
        }
        else{
            tu=graph.edges[ju]["target"];
        }
        geometries.colors[i] = new
THREE.Color(nodescolor[graph.degree[tu]%50]);
        geometries.colors[i+1] = geometries.colors[i];
        ju++;
    }

        var material = new THREE.LineBasicMaterial( {color:
0xffffffff,vertexColors: THREE.VertexColors} );
        var edgesGeo = new THREE.Line(geometries,material,
THREE.LinePieces);
        edgesGeo.frustumCulled=false;
        scene.add(edgesGeo);

    maxdegree=0;
    for (var gy=0;gy<graph.nodes.length;gy++)
    {

```

```

        if(graph.degree[gy]>maxdegree)
        {
            maxdegree=graph.degree[gy];
        }
    }

    StartingValues[0]="Watts-Strogatz Model";
    StartingValues[1]="Networks Starting Parameter";
    StartingValues[2]="Node Count:"+wtcount.toString();
    StartingValues[3]="Start Neighbour
Count:"+Wtncount.toString();
    StartingValues[4]="Prewire:"+Probwt.toString();
    this.created=1;
}

```



ÖZGEÇMİŞ

Gökhan KUTLUANA, 30 Mart 1986 tarihinde Konya’da doğdu; ilk, orta öğrenimini Akşehir ilçesinde, yükseköğrenimini Konya ilinde tamamladı. 2004 yılında Akşehir Anadolu Lisesinden, 2008 yılında Selçuk Üniversitesi Bilgisayar Mühendisliği Bölümünden mezun oldu. 2009 yılında Bartın Üniversitesi Bilgi İşlem Dairesi Başkanlığında Programcı olarak çalışmaya başladı. 2011 yılında aynı üniversitenin Uzaktan Eğitim Araştırma ve Uygulama Merkezinde uzman olarak çalışmaya devam etti. 2013 yılından itibaren Bartın Üniversitesi Sağlık Hizmetleri Meslek Yüksekokulunda Öğretim Görevlisi olarak çalışmaya devam etmektedir. Karabük Üniversitesi, Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans Eğitimine devam etmektedir.

ADRES BİLGİLERİ

Adres : Bartın Üniversitesi
Sağlık Hizmetleri Meslek Yüksekokulu, Tıbbi Hizmetler ve
Teknikleri Bölümü
Türbe yolu Caddesi/BARTIN

Tel : (553) 547 48 40

E-posta : gkutluana@bartin.edu.tr