GENETİK ALGORİTMA İLE İLETİŞİM AĞLARINDA YÖNLENDİRME OPTİMİZASYONU

Alper ÖZBİLEN

YÜKSEK LİSANS TEZİ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HAZİRAN 2006
ANKARA

Alper ÖZBİLEN tarafından hazırlanan GENETİK ALGORİTMA İLE İLETİŞİM AĞLARINDA YÖNLENDİRME OPTİMİZASYONU adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Yrd. Doç. Dr. Erkan Afacan
Tez Yöneticisi

Bu çalışma, jürimiz tarafından Elektrik-Elektronik Mühendisliği Anabilim Dalında Yüksek lisans tezi olarak kabul edilmiştir.

Başkan: : Prof. Dr Erdem YAZGAN

Üye : Yrd. Doç. Dr. Erkan AFACAN

Üye : Dr. Nursel AKÇAM

Tarih : 22/06/2006

Bu tez, Gazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygundur.

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Alper Özbilen

GENETİK ALGORİTMA İLE İLETİŞİM AĞLARINDA YÖNLENDİRME OPTİMİZASYONU

(Yüksek Lisans Tezi)

Alper ÖZBİLEN

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
Haziran 2006

ÖZET

İletişim ağları için trafik yönlendirme hayati önem taşımaktadır. Yeterli kapasiteye sahip olsalar bile, yanlış yönlendirme iletişim ağlarında darboğazlar yaşanmasına neden olmaktadır. Doğru yönlendirme meselesi, karmaşık ve çok yönlü değerlendirilmesi gereken bir konudur. Bir iletişim ağı için majör şartları uzun vadeli kapasite planlaması, minör şartları ise periyodik operasyonlar belirler. Periyodik operasyonların başarısı ağ üzerindeki trafiğin tüm ağ kaynakları kullanılarak aktarılmasına bağlıdır. Dengeli trafik dağılımı için, ağın taşıma birimlerinin hesaplanmış ağırlık değerlerine göre ayarlanması gerekir. Ulusal veya uluslararası düzeyde hizmet veren ağlar için linklerin ağırlık değerini belirlemek oldukça zordur. Link ağırlık değerlerinin analitik veya nümerik yöntemlerle belirlenmesi neredeyse imkansızdır. Bu tezde linklerin ağırlık değerlerinin genetik algoritma kullanılarak belirlenmesi üzerinde çalışılmıştır. Yapılan simülasyonlarda, nispeten büyük ağlarda genetik algoritmanın daha başarılı sonuçlar verdiği görülmüştür.

Bilim Kodu : 905.1.034
Anahtar Kelimeler : Grafik Teorisi, Genetik Algoritma, Yönlendirme
Sayfa Adedi : 108
Tez Yöneticisi : Yrd. Doç. Dr. Erkan Afacan

ROUTING OPTIMIZATION IN COMMUNICATION NETWORKS WITH GENETIC ALGORITHM

(M.Sc. Thesis)

Alper ÖZBİLEN

GAZİ UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY

June 2006

ABSTRACT

Traffic routing is a vital issue for communication networks. Misrouting may be a reason of bottlenecks in communication networks even if they have enough capacity. Proper routing is a complex issue and needs to be evaluated in many ways. For a communication network, major conditions are determined by capacity planning and minor conditions are determinde by periodical operations. The success of periodical operations depend on the traffic transfer by using all network resources. In order to provide balanced traffic diversity, network carrier units should be set with precalculated weight values. For networks offering national and international services, link setting is a difficult issue. Link weight setting is nearly imposible to be obtained by analytical or numerical methods. In this study, genetic algorithm is used to obtain link weight values. It is shown that genetic algorithm offers more successful results for relatively large networks.

Science Code : 905.1.034
Key Words : Graph Theory, Genetic Algorithm, Routing
Page Number : 108
Adviser : Asist. Prof. Dr. Erkan Afacan

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarından ötürü Hocam Yrd. Doç. Dr. Erkan Afacan'a, ayrıca lisans ve yüksek lisans eğitimim boyunca manevi destek ve teşviklerinden ötürü Gazi Üniversitesi Elektrik Elektronik Mühendisliği Bölüm Başkanı Prof. Dr. Cengiz Taplamacıoğlu'na, her türlü konuda yardımını esirgemeyen meslektaşım Arif Arısoy'a ve varlıklarını hep arkamda hissettiğim sevgili aileme teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış bazı kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	**Açıklama**
IETF	İnternet Mühendisliği Görev Takımı
RFC	Yorum Talep Dokümantasyonu
IGP	İç Ağ-geçitleri Protokolü
EGP	Dış Ağ-geçitleri Protokolü
BGP	Sınır Ağ-geçitleri Protokolü
RIP	Yönlendirme Bilgi Protokolü
IGRP	İç Ağ-geçitleri için Yönlendirme Protokolü
EIGRP	İyileştirilmiş İç Ağ-geçitleri Yönlendirmesi
OSPF	Açık En Kısa Yol Protokolü
STP	En Kısa Yol Ağacı
IP	İnternet Protokolü
SNMP	Basit Ağ Yönetim Protokolü
MRTG	Çok Yönlendiricili Trafik Grafikçisi
GA	Genetik Algoritma
PHP	Kişisel Web Programla Dili
CPU	Merkezi İşlem Birim
RAM	Rasgele Erişilir Bellek

1. GİRİŞ

İletişim ağlarının en önemli konusu geleceği de öngören kapasite planlaması ve periyodik operasyonların amacına uygun yapılmasıdır. Günümüzde iletişim ağlarının geçmiş yıllara göre çok daha hızlı büyüdüğü ve üzerlerindeki trafik yükünün her geçen gün artığı düşünülürse planlama ve yönetim işinin daha da zorlaştığı yorumu kolayca yapılabilir.

Bir iletişim ağının düzgün işletilmesi, üzerindeki trafik yükünün tüm ağ kaynaklarına dengeli olarak paylaştırılmasıyla mümkündür. Yük ve kapasitenin böyle bir denge üzerine oturtturulabilmesi ise ağın ve trafiğin matematiksel olarak modellenebilmesiyle gerçekleştirilebilir. Gerek ses gerekse veri taşıyan ağların dengeli trafik yüküne kavuşturulabilmesi, ağ üzerinde düzgün yönlendirmeler yapılmasına bağlıdır.

İkinci bölümde, graf teorisi ve herhangi bir iletişim ağının graf teorisi kullanılarak sıklık ve komşuluk matrisleri cinsinden nasıl modellenebileceği açıklanmıştır. Ayrıca, üçgen eşitsizliği yöntemini temel alan algoritmalarla uçlar arası en kısa yolun nasıl hesabı gösterilmiştir [1].

Üçüncü bölümde, yönlendirme protokolleri, yönlendirme matrisi, tahmini trafik matrisi gibi trafik mühendisliğinin temel argümanları açıklanarak, ağ üzerindeki trafik dağılımının nasıl hesaplandığı gösterilmiştir.

Dördüncü bölümde, genetik algoritmanın temel çalışma prensipleri anlatılmış ve bu çalışma kapsamında tasarlanan algoritmanın, en iyi trafik dağılımı için en uygun link ağırlık değerlerini ararken kullandığı uygunluk fonksiyonu tanımlanmıştır.

Beşinci bölümde ise, iki farklı iletişim ağı topolojisi için, farklı trafik matrisleri ve farklı genetik algoritma parametreleri kullanılarak yürütülen deney çalışmalarına yer verilmiştir.

2. GRAF TEORİSİ VE EN KISA YOL ALGORİTMALARI

2.1. Graf Teorisi

Graf Teorisi'nin öyküsü Almanya topraklarındaki Königsberg kasabasını dört parçaya bölen Pregel nehri üzerindeki yedi köprüyle başlar. Bu şehir üzerinden seyahat eden yolcular, aynı köprü üzerinden ikinci bir defa geçmeden bir an önce gitmek istedikleri tarafa geçme çabasında olmuşlardır. 1736 yılında ünlü matematikçi Leonhard Euler, böyle bir durumun matematiksel analizinin sadece graf terimleri kullanılarak izah edilmesinin mümkün olmadığını ispatlamıştır. Euler'in bu konuda yayınladığı makale Graf Teorisini işaret eden bir başlangıç olmuştur [1, 2] .

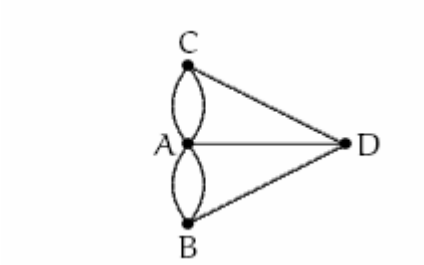
Graf Teorisi günümüzde bilgisayar bilimleri, kimya, ekonomi ve hatta sosyal bilimleri de içeren birçok alanda kullanım sahası bulmuştur. Bilgisayar ağlarının tasarımı, ulaşım ağlarının modellenmesi, elektronik baskı kartları üzerindeki devrelerin dizilişi, veritabanı mimarilerine, sınav ve ders programlarının çizelgelenmesi, sosyal ilişkilerinin resmedilmesi Graf Teorisi'nin tipik kullanım alanları arasına girmiştir.

2.2. Graf Terminolojisi

Graf Teorisini kullanarak, en kısa yol algoritmalarına başlamadan önce düğüm, kenar, yol, basit yol, yönlülük, ağırlık gibi kavramların neyi ifade ettiğinin tanımlaması gerekir.

2.2.1. Düğüm ve kenarlar

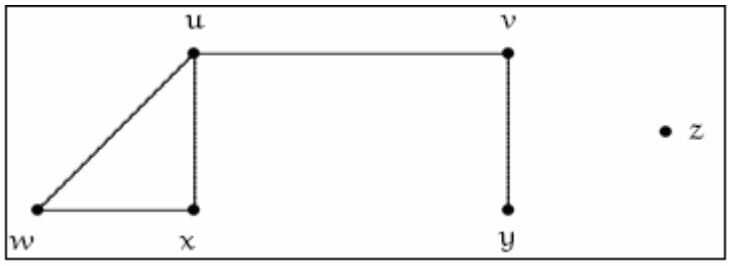
Bölüm 2.1'de bahsettiğimiz dört parçadan oluşan ve yedi köprü sayesinde birbirine bağlanan kasabayı, Şekil 2.1 'deki gibi graf olarak modelleyelim.



Şekil 2.1. Yedi köprülü geçiş problemi diyagramı [1]

Şekil 2.1'deki grafımızda gösterilen A, B, C, D noktaları düğüm (vertex) olarak isimlendirilir. Bu noktalar arasına çekilen çizgiler ise kenar (edge) olarak isimlendirilir.

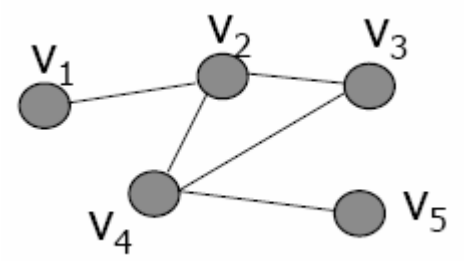
Bir grafın, düğümlerini ifade eden kümesi V(G); kenarlarını ifade eden kümesi ise E(G) şeklinde gösterilir. Örneğin Şekil 2.2'deki graf V(G)={u,v,w,x,y,z} ve E(G)={uv,uw,ux,vy,wx} olarak ifade edebiliriz.



Şekil 2.2. Basit graf örneği [1]

2.2.2. Yol ve basit yol

İki düğümü, bir veya birden fazla kenar üzerinden birbirine bağlayan güzergaha yol adı verilir. Herhangi iki düğüm arasında, grafın topolojisine bağlı olarak birden fazla yol tanımlanabileceği gibi, hiçbir yol tanımlanamaya da bilir. Bir düğümden diğerine giderken güzergah üzerindeki düğümden sadece bir defa geçilmesi şartı konulmuşsa, bu yol, basit yol olarak tanımlanır.



Şekil 2.3. Yol ve basit yol gösterimi

Şekil 2.3'deki graf üzerinde tanımlanabilecek V2, V3, V4, V2, V1 güzergahı bir yoldur. Diğer yandan, V2, V3, V4, V5 ise basit bir yoldur.

2.2.3. Yönlü ve yönsüz graflar

Bir grafta tanımlanan kenarlar yönlü ya da yönsüz olarak ifade edilebilir. Herhangi bir graftaki, kenarların yönlü ya da yönsüz olması, graf olarak resmedilen sistemin karakteristiğiyle ilgilidir.

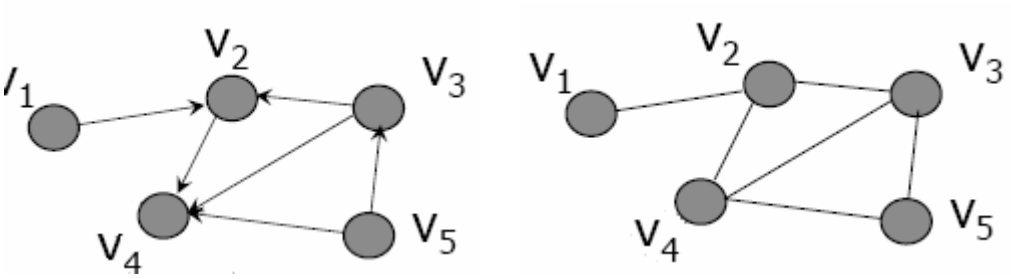
U ve V düğümleri arasında çizili bir kenar düşünelim. Şayet U ile V arasındaki kenar yönlü olarak verilmişse (U, V) ile (V, U) farklı şeyi ifade etmezler. Şayet kenar (U, V) olarak tanımlanmışsa U kaynak V ise hedef noktasını gösterir [1,2] .

Bir graftaki iki düğüm arasında yön belirtilmemişse, bu kenar yönsüz kenar olarak adlandırılır. Bu durumda, U ile V düğümleri arasındaki kenarın (U, V) veya (V, U) olarak gösteriminde herhangi bir fark yoktur [1] .

Bir grafin yönlü olarak adlandırılabilmesi için tüm kenarlarının yönlü olması gerekir [1,2] .

Bir grafın yönsüz olarak adlandırılabilmesi için tüm kenarlarının yönsüz olması gerekir [1] .

Şekil 2.4.a' da yönlü, Şekil 2.4.b' de ise yönsüz grafa örnek verilmiştir.



(a) (b)

Şekil 2.4. Yönlü ve yönsüz basit graf a) yönlü b) yönsüz

2.2.4. Düğümlerin komşuluğu

Graf Teorisi ile yapılan analiz ve tasarımlarında en temel işlem düğümlerin komşuluk ilişkilerini incelenmesidir.

Komşuluk, herhangi bir düğümün diğer düğümlerle olan doğrudan bağlantısını ifade eder. Ele aldığımız graftaki, U ve V olarak adlandırdığımız iki düğümü birbirine bağlayan bir kenar varsa bu iki düğüm birbirinin komşusudur.

Yönlü graflarda, komşuluk ilişkisinin belirlenmesi, daha fazla dikkat gerektirmektedir. Zira bir U ile V düğümü arasında (U, V) yönlü kenarı tanımlı ancak (V, U) tanımlı değilse, U düğümü V düğümünün komşusuyken V düğümü U düğümünün komşusu değildir.

2.2.5. Komşuluk ve sıklık matrisi

Grafları matematiksel olarak modellemek için matrisler kullanılır. En yaygın kullanılan yöntem ise komşuluk ve sıklık matrisleridir.

Düğümlerin birbirleriyle olan komşuluğunu gösteren matrise komşuluk matrisi adı verilir. Ele aldığımız grafın V1, V2, V3,……,Vn düğümlerinden oluşan n sayıda düğümü olduğunu düşünürsek, bu grafiğin komşuluk matrisi nxn boyutunda kare matris olacaktır [1,2] .

En genel ifadeyle komşuluk matrisi Çizelge 2.1'deki gibi ifade edilir.

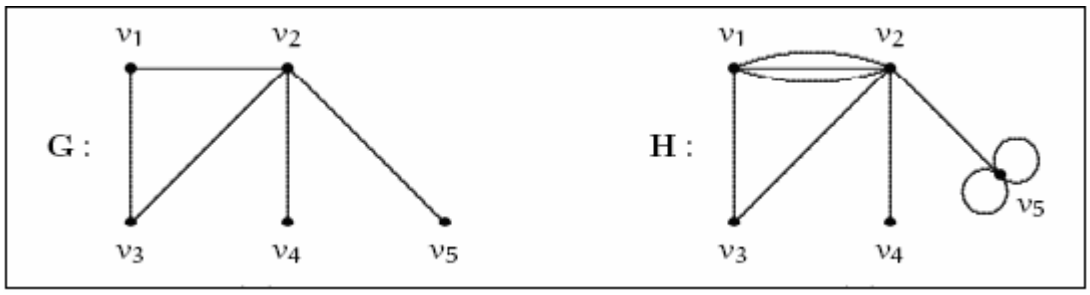
Çizelge 2.1. Komşuluk matrisinin genel ifadesi

	V1	V2	…	…	…	…	Vn
V1	A1,1	A1,2	…	…	…	…	A1,n
V2	A2,1	A2,2	…	…	…	…	A2,n
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
...	…	…	…	…	…	…	…
Vn	An,1	An,2	…	…	…	…	An,n

V1 düğümü, V2 düğümü ile komşu ise (V1, V2) komşuluk değeri 1'dir ve A12=1 olarak matris üzerinde işaretlenir. Şayet, V1 düğümü V2 düğümü ile komşu değil ise (V1, V2) komşuluk değeri 0'dır ve A12=0 olarak matris üzerinde işaretlenir.

Yönsüz graflarda (Vi, Vj) komşuluk değeri (Vj,Vi) komşuluk değeriyle aynı olduğundan, matrisimizde yer alan tüm Aij değerleri Aji değerleriyle aynıdır. Dolayısıyla, yönsüz graflarda komşuluk matrisimiz köşegenine göre simetriktir.

Şekil 2.5' de verilen G yönsüz grafıyla ile H yönsüz grafının komşuluk matrisleri, Şekil 2.6'da verilmiştir.



Şekil 2.5. G yönlü grafıyla ile H yönsüz grafı

$$M_{adj}(G) = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \qquad M_{adj}(H) = \begin{bmatrix} 0 & 3 & 1 & 0 & 0 \\ 3 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 \end{bmatrix}$$

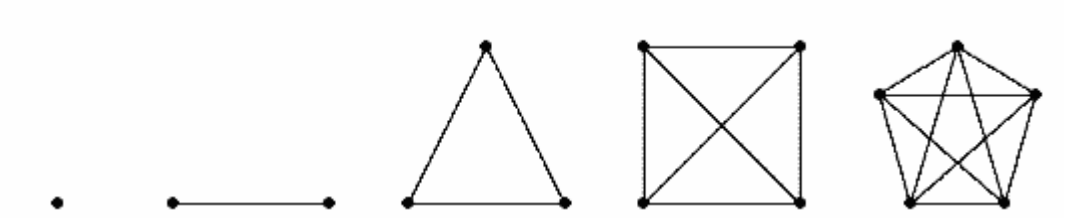
Şekil 2.6. G yönlü grafıyla ile H yönsüz grafının komşuluk matrisleri

Şekil 2.6'de yer alan G matrisi ile H matrisinin ortak özelliği, her iki matrisin de köşegenlerine göre simetrik olmasıdır. Diğer yandan, her iki matriste de 5 tane düğüm olmasına rağmen, G ve H matrisi arasında temel farklılıklar bulunmaktadır. Bu farklılığın nedeni, H matrisinin daha fazla kenara sahip olmasıdır. H matrisindeki her bir düğümün sadece tek bir komşuluğu varken G matrisindeki düğümlerin, kendisiyle ya da diğer düğümlerle birden fazla komşuluğu bulunmaktadır. Dolayısıyla H matrisi sadece 1 ve 0 değerlerinden oluşurken, G matrisi 2, 3 gibi farklı değerler de almıştır.

Bir graftaki tüm düğümlerin birbirleriyle komşuluk ilişkisi içerisinde olabilmesi için, N sayıda düğüm içeren bir grafta olması gereken minimum kenar sayısı Eş. 2.1'deki gibi tanımlanır [1,2] .

Kenar Sayısı= N.(N-1)/2 (2.1)

Şekil 2.7'de verilen üç farklı graftaki tüm düğümlerin, birbirleriyle komşuluk içerisinde olduğu ve toplam kenar sayısının Eş 2.1'i sağladığı görülmektedir.



Şekil 2.7. Graflarda düğüm kenar ilişkisi [1]

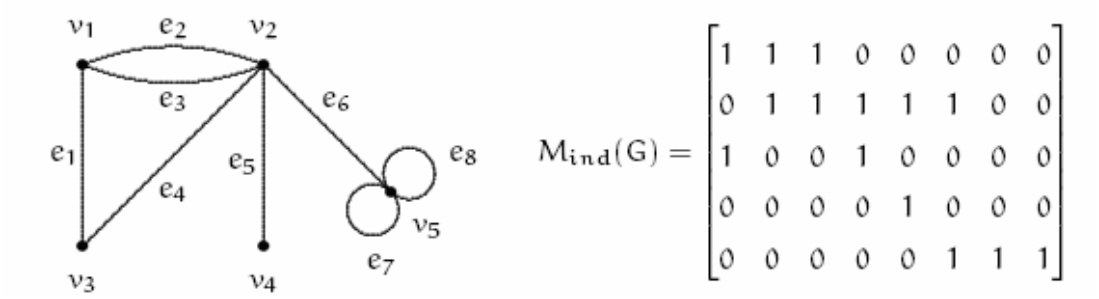
Grafları matematiksel olarak tanımlamada yaygın olarak kullanılan diğer bir yöntemse sıklık matrisidir. Komşuluk matrisi, düğümlerin birbirleriyle olan ilişkisini gösterirken, sıklık matrisi düğümlerin kenarlarla olan ilişkisini gösterir.

Ele aldığımız grafiğimizin N sayıda düğüm ve M sayıda kenarı olduğunu düşünürsek, sıklık matrisimiz NxM boyutunda olacaktır. En genel ifadeyle sıklık matrisi Çizelge 2.2.'daki gibi ifade edilir.

Çizelge 2.2. Sıklık matrisinin genel ifadesi

	E1	E2	…	…	…	…	…	Em
V1	A1,1	A1,2	…	…	…	…	…	A1,m
V2	A2,1	A2,2	…	…	…	…	…	A2,m
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…
Vn	An,1	An,2	…	…	…	…	…	An,m

Şekil 2.8' de, 5 adet düğüm 8 adet kenara sahip bir graf sıklık matrisiyle birlikte verilmiştir.



$$M_{ind}(G) = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Şekil 2.8. Bir grafın sıklık matrisinin ifadesi [1]

Şekil 2.8'de de görüldüğü gibi, sıklık matrisi komşuluk matrisinden çok temel farklar taşımaktadır. Bu farklar:

i) Graftaki kenar sayısından bağımsız olarak, sıklık matrisi sadece 1 ve 0 değerlerinden oluşur.
ii) Yönlü veya yönsüz graflar için, sıklık matrisinin köşegene göre simetrik olması söz konusu değildir.
iii) Düğüm sayısı kenar sayısına eşit olmadığı sürece kare matris değildir.

2.2.6. Ağırlık değeri taşıyan kenarlar

Graf olarak modellediğimiz bazı sistemlerde, komşuluk ilişkisini sadece 1 ve 0 ile ifade etmek bizi varmak istediğimiz sonuca taşımayabilir. Örneğin bir ulaşım ağı modellenirken, iki farklı nokta arasındaki mesafe, yoğunluk gibi şartların mutlaka dikkate alınması gerekir. Benzer şekilde bir iletişim ağı modellenirken bant genişliği, terminaller arasındaki trafik geçişi temel kriterler arasındadır. Dolayısıyla bu tür ağları modellenirken, grafta yer alacak düğümler arasındaki ilişki sadece 1 ve 0'lardan oluşan komşuluk ilişkisi içerisinde incelenmelidir.

Ağırlık değeri, düğümler arasındaki kenarlara atadığımız bir değer olup, ele aldığımız sisteme ilişkin temel veriler ışığında seçilir. Farklı kaynaklarda uygunluk, uygunsuzluk ya da maliyet değeri olarak da adlandırılmaktadır.

Ağırlık değerinin temel işlevi, düğümler arasında tanımlanabilecek alternatif yollar arasında uygunluk sınıflandırmasına imkan tanımasıdır. Örneğin komşu iki şehri birbirine bağlayan üç alternatif yol olduğunu düşünelim. Yol mesafelerinin sırasıyla 100 km, 200 km, 300 km olduğu varsayılırsa, yolların ağırlık değerlerini sırasıyla 1, 2 ve 3 olarak atamamız gerekir.

Ağırlık değerini hesaplamak her zaman yukarıdaki örnekte verdiğimiz kadar kolay değildir. Zira bir çok durumda ele alınması gereken bir çok kriter bulunmaktadır. Örneğin, veri iletim ağlarında trafik yönlendirme protokolü olarak kullanılan IGRP (Interior Gaterway Routing Protocol), bağlantı hatlarının ağırlık değerini hesaplarken Eş 2.2 'deki denklemi kullanmaktadır [4] .

$$A.\ D = [K1 * b + (K2 * b/(256 - yük) + K3 * d] * (K5/(güvenilirlik + K4)] \quad (2.2)$$

$b = 256 * 10^7/\min(\text{bant genişliği})$, $d = 256 * \text{gecikme}$

Eş 2.2'de hattın bant genişliği, yükü, gecikmesi, güvenilirliği ağırlık değerini belirlemekte kullanılmaktadır. K sabitleri ise hangi parametrenin hangi ölçüde etki edeceğini belirler.

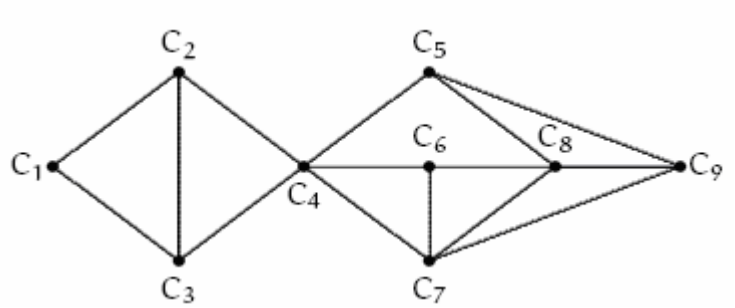
Kullanım temelli ücretlendirilen kiralık veri hatlarında, hatların kiralama maliyetine bağlı olarak ağırlık değerlerinin hesaplanması da sıklıkla kullanılan yöntemler arasındadır.

2.2.7. Düğümler arası uzaklık ve en kısa yol

İki düğüm arasında geçilen kenar sayısı ve varsa bu kenarların ağırlık değeri düğümler arasındaki uzaklığı gösterir.

Şayet iki düğüm arasında tanımlanabilecek tek bir geçerli yol varsa, bu düğümler birbirlerine sabit bir uzaklıktadırlar. Diğer yandan, düğümler arasında tanımlanabilecek birden çok yol varsa, seçilen yola bağlı olarak düğümler birbirinden

farklı uzaklıklarda olabilirler. Ancak genel olarak, iki düğüm arasındaki uzaklık, iki düğüm arasındaki en kısa yolun uzaklığı olarak ifade edilir [1,2].



Şekil 2.9 Graf olarak modellenmiş bir bilgisayar ağı [1]

Şekil 2.9'da verilen bilgisayar ağındaki düğümler arasındaki kenarların ağırlıksız ve yönsüz olduğu düşünüldüğünde, C_1 ve C_9 arasında tanımlanabilecek en kısa yolları bulmak için kriter, üzerinden geçilen düğüm sayısının sayılmasıdır. C_1'den C_9'a ya da C_9'dan ve C_1'e gidebilmek için kullanabilecek en kısa yol sayısının 4 olduğu görülmektedir. Bu yollar:

i) C1 – C2 – C4 – C5 – C9
ii) C1 – C2 – C4 – C7 – C9
iii) C1 – C3 – C4 – C7 – C9
iv) C1 – C3 – C4 – C5 – C9

şeklindedirler.

Yukarıda dört yol haricindeki yollar tanımlanırken daha fazla düğüm üzerinden geçileceğinden, en kısa yol tanımına uymaz ve bu nedenle tanımlanmaları anlamlı değildir.

Şayet ele alınan graf yönlü ve ağırlıklı olsaydı, en kısa yolu bulurken atlanan düğüm sayısı yerine uçtan uca toplam ağırlık değerine bakılması gerekirdi. Bu durumda uçtan uca en küçük ağırlık toplamın veren yol en kısa yol olur.

Graf üzerindeki düğümler arasındaki en kısa yolları hesaplamak için çeşitli algoritmalar geliştirilmiş olup, tüm bu algoritmaların temel özelliği uçlar arasındaki mesafenin adım adım hesabına dayanır. Her adımda mesafe yeniden hesaplanarak, alternatif adımlar ile karşılaştırılır.

Üzerinde en az iki düğümü olan bir graf düşünelim. Bu graf üzerinde U ve V olarak adlandırdığımız iki düğüm olsun. Şayet, grafımızın topolojisi U ile V arasında bir yol tanımlamaya olanak veriyorsa U ile V arasındaki en kısa mesafeyi d(U, V) olarak adlandıralım. Bu durumda d(U, V) Eş. 2.3 - Eş. 2.6 sağlamalıdır, [1, 3-4] .

$$d(U, V) \geq 0 \quad (2.3)$$

$$d(U, V) = 0 \text{ ancak ve ancak } U=V \text{ ise} \quad (2.4)$$

$$d(U, V) = d(V, U) \text{ grafımız yönsüz ise} \quad (2.5)$$

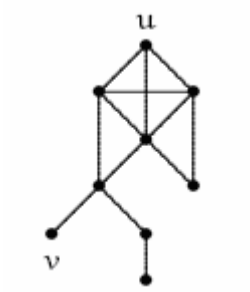
$$d(U, V) \leq d(U, W) + d(W, V) \quad (2.6)$$

Diğer yandan ele aldığımız grafiğimizin topolojisi U ile V arasında yol tanımlanmasına imkan vermiyorsa Eş 2.7 geçerlidir.

$$d(U, V) = \infty \quad (2.7)$$

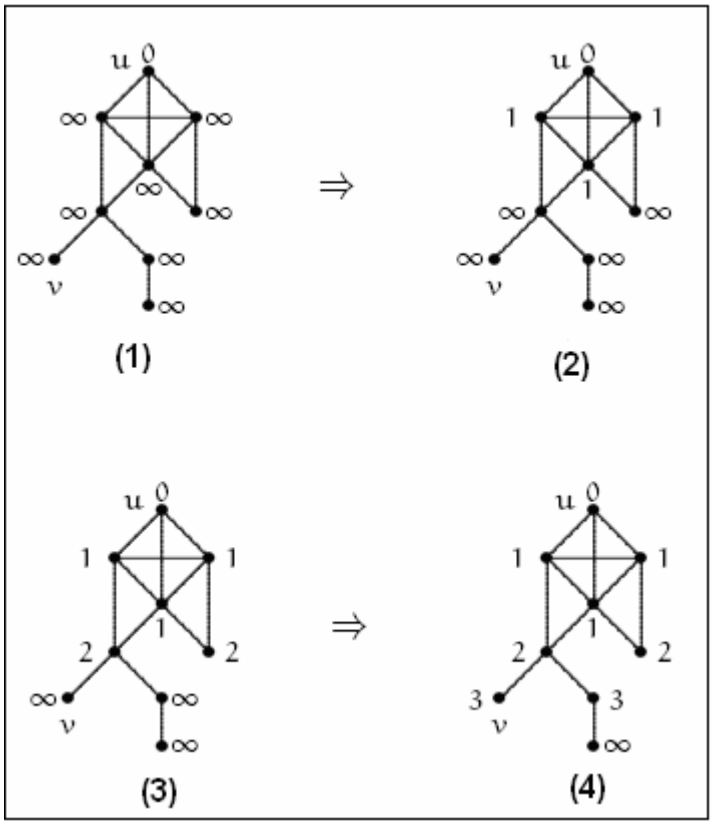
Eş. 2.6 üçgen eşitsizliği olarak adlandırılır ve en kısa yol algoritmalarında kullanılan anahtar denklemdir. Eş 2.6'da yer alan W grafı üzerindeki herhangi bir noktayı temsil eder. En kısa yol algoritmalarında W, arama sırasında elimizdeki d(U, V) değerinin hala geçerli olup olmadığını sınamakta kullanılır.

En kısa yol algoritmalarında, bir uçtan başlayarak diğer uca giderken kullanılan en etkili yollardan biri graftaki tüm düğümlerin etiketlenmesi yöntemidir [1-3]. Algoritma, başlangıç ve varış noktası da dahil olmak üzere düğümleri etiketler ve bu etiketlerin değerini her adımda güncelleyerek mesafe hesabını yeniden yapar. Bu işlem varış noktasına varıncaya kadar devam eder.



Şekil 2.10. U ve V düğümlerine sahip bir graf [1]

Şekil 2.10'daki grafta yer alan U ve V düğümleri arasında topolojik olarak birden fazla yol tanımlamanın mümkün olduğu görülmektedir. Yönsüz ve ağırlıksız olan grafımızda yer alan U ve V arasındaki en kısa yol, Eş 2.6'de tanımlanan üçgen eşitsizliği ve etiketleme yöntemi kullanılarak hesaplanır. Hesaplamada kullanılan dört adım Şekil 2.11' de verilmiştir.



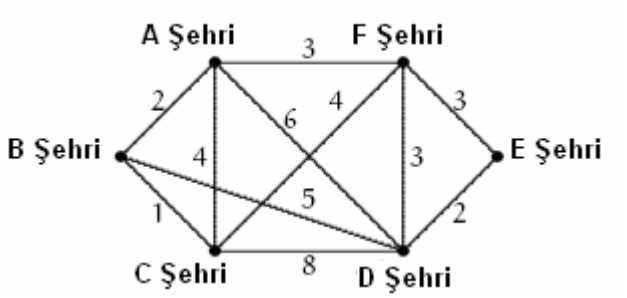
Şekil 2.11. U düğümünden V düğümüne dört adımda varış [1]

Şekil 2.11'deki birinci adımda başlangıç düğümümüz olan U, sıfır olarak etiketlenmiş, varış düğümümüz olan V'de dahil olmak üzere diğer tüm düğümler sonsuz olarak işaretlenmiştir. Birinci adımda algoritma, U düğümü haricindeki tüm düğümleri kayan noktamız olan W olarak görmüştür. İkinci adımda algoritmanın taradığı düğümlerden U düğümünün komşusu olan düğümler, komşuluk matrisi sayesinde tespit edilmiş ve U düğümüne komşu düğümlerin etiketi sonsuzdan 1 değerine çekilmiştir. Üçüncü adımda, etiket değeri 1 olan düğümlerin komşuları taranmış ve değerleri sonsuzdan 2'ye çekilmiştir. Son adımda V düğümüne ulaşılmıştır.

2.2.8. Ağırlıklı graflarda uzaklık

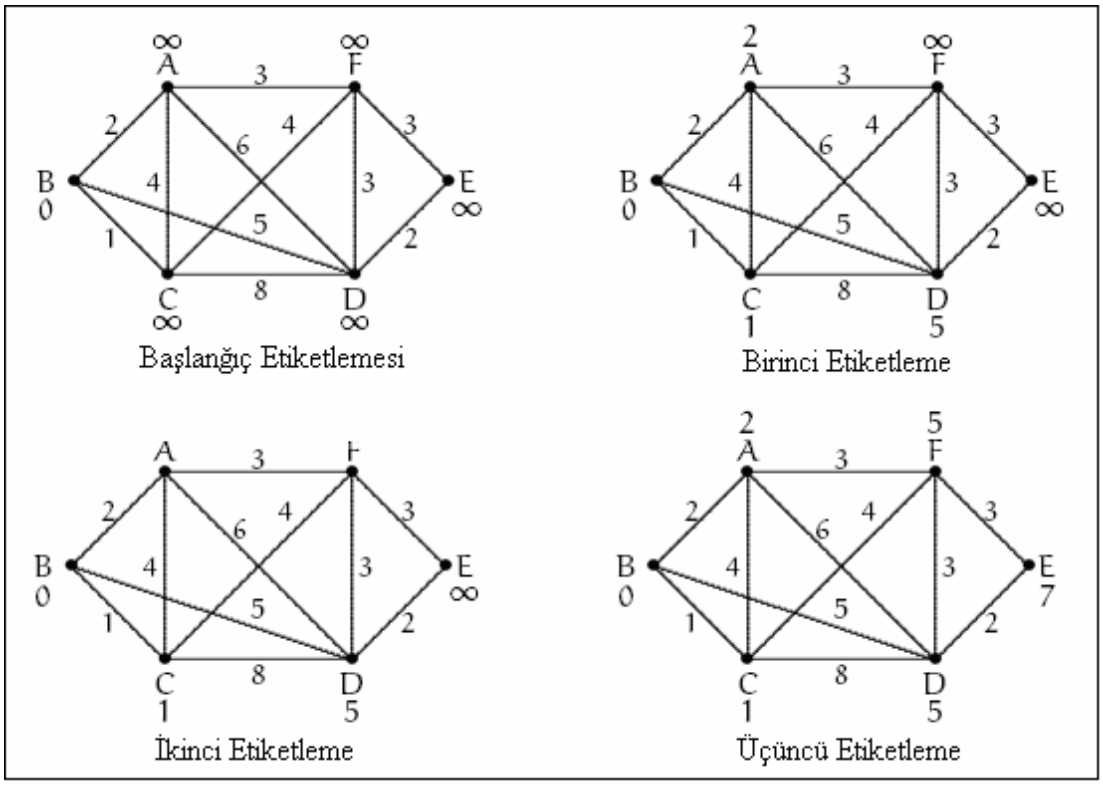
Bölüm 2.2.5'de, Ulaşım ve İletişim ağları gibi tipik graf teorisi uygulamalarında, düğümlerin komşuluk ilişkisinin, kenarların ağırlık değerleriyle birlikte değerlendirilmesi gerektiği belirtilmiştir. Ağırlıksız graflarda en kısa yol aranırken sadece yol boyunca geçilen düğüm sayısına bakılırken, ağırlıklı graflarda en kısa yol hesabı tamamıyla kenarların ağırlık değerine bağlıdır. İki uç arasındaki en kısa yol, toplam ağırlık değeri en küçük olan yoldur.

Şekil 2.12'de 6 şehri birbirine bağlayan yollar ve bu yolların ağırlık değeri verilmiştir. Ağırlık değeri olarak bir şehirden diğerine gitmek için harcanan ortalama süreler atanmıştır.



Şekil 2.12. Altı şehrin ağırlıklı graf ile ulaşım zamanı modeli

Ağırlıklı graflarda en kısa yol bulma algoritması, temel anlamda ağırlıksız graflarda olduğu gibi Eş 2.6'da tanımlanan, üçgen eşitsizliğine ve komşuluk ilişkilerine göre her adımda yeniden etiketleme yöntemine dayanır. Ancak kaynaktan hedefe doğru ilerlerken etiket birer birer artırılmak yerine, komşu düğümler arasındaki kenar ağırlıklarını kadar arttırılırlar.



Şekil 2.13. Düğümlerin etiketlenmesi

Şekil 2.12'de yer alan B şehrinden E şehrine giden en kısa yolu bulma süreci, Şekil 2.13'de dört adımda gösterilmiştir. Birinci adımda, başlangıç şehri olan B sıfır, diğer tüm şehirler ise sonsuz değerini almıştır. İkinci adımda B şehrinin komşusu olan A, C ve D şehirlerine B ile aralarındaki kenar ağırlık değerleri kadar uzaklık değeri atanmıştır. Üçüncü adımda, A, C ve D şehirlerinin, kendilerine komşu olup, B'ye komşu olmayan F şehrine uzaklığı bir önceki adımdaki değerler de dikkate alınarak hesaplanmıştır. Son olarak B şehrinden F şehrine olan en kısa uzaklık hesaplanmıştır. Her bir adımda hesaplanan değerler Çizelge 2.3'de verilmiştir.

Çizelge 2.3. Etiketleme sonrası hesaplanan değerler

		A	B	C	D	E	F
Etiketleme 1	-	∞	0	∞	∞	∞	∞
Adım 1	B	0+2=2	0	0+1=1	0+5=5	∞	∞
Adım 2	C	2	0	1	5	∞	1+4=5
Etiketleme 2	A	2	0	1	5	∞	5
Adım 3	D	2	0	1	5	5+2=7	5
Etiketleme 1	F	2	0	1	5	7	5
Adım 4	E	2	0	1	5	7	5

Bellman Ford Algoritması, Johnson Algoritması, Floyd-Warshall Algoritması ve Dijkstra Algoritması ağırlıklı grafların en kısa yol hesaplamalarında kullanılan başlıca algoritmalardandır [3]. Bu algoritmalar içerisinde en çok kullanım alanı bulan, Dijkstra Algoritması'dır.

2.3. Dijkstra Algoritması

Adını Alman bilgisayar bilimcisi Edsger Dijkstra'dan alan Dijkstra Algoritması, yönlü ve ağırlıklı graflarda, düğümler arasındaki en kısa yolu hesaplamakta kullanılır [3] .

Dijkstra Algoritması, internet ağ trafiği yönlendirme protokolü olan ve kullanımı oldukça yağın olan OSPF (Open Shortest Path First) protokolünün temelini teşkil eden algoritmadır [3] .

Dijktsra Algoritması'nı benzer uygulamalarda kullanılan Bellman Ford Algoritmasından ayıran temel özellik negatif ağırlıklı kenar içeren graflarda uygulanamayışıdır [3] .

2.3.1. Dijkstra algoritması için komşuluk matrisinin hazırlanması

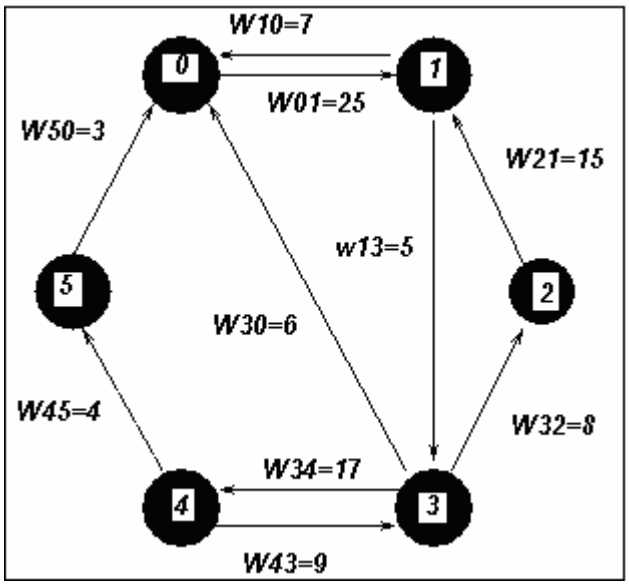
Dijktsra Algoritması'nın işletilebilmesi için, program girdilerinin, düğüm ve düğümler arası kenarların ağırlık değerlerini içerecek biçimde düzenlenmesi gerekir. N sayıda düğüm içeren grafımızdaki düğümler, 0'dan başlayarak N-1'e kadar sırayla etiketlenmelidir. Düğümler arasındaki kenarlar ise hangi düğümlerin arasında yer alıyorsa o düğümlere ait etiketlerle ifade edilmelidir. Başka bir ifadeyle düğümlerimizi $V=(V_0, V_1, V_2, \ldots\ldots, V_{N-1})$ biçiminde gösteriyorsak, kenarlarımız $W=(W_{00}, W_{01}, W_{02}, \ldots\ldots, W_{N-2\ N-1}, W_{N-1\ N-1})$ biçiminde gösterilmelidir.

Genel haliyle program girdisi, hem düğüm etiketlerini ve düğümler arası ağırlık değerini içeren Eş 2.8' deki gibi tanımlanır.

$$G=(V_i, V_j, W_{ij}) \tag{2.8}$$

Eş. 2.8, komşuluk listesi olarak da isimlendirilir. Komşuluk listesine tüm düğümlerin ve tüm düğümler arasındaki kenarların ağırlık değerlerinin girilmesi gerekmez. Programın, girilen komşuluk listesini kullanarak, komşuluk matrisini elde edebilmesi için graftaki toplam düğüm sayısının da ayrıca belirtilmesi yeterlidir. Ayrıca, algoritmanın kodlanması sırasında, bir düğümün kendisine olan uzaklığı sıfır kabul edilmeli; program girdisinde iki düğüm arasında bir kenar bildirilmemişse, bu iki düğümün birbirine doğrudan uzaklığı sonsuz olarak atanmalıdır.
Komşuluk listesi verilen bir graf için komşuluk matrisini üreten sözde kod (pseudocode) EK-1' de verilmiştir.

EK-1'de verilen ve komşuluk matrisini elde etmek için kullanılan sözde kodun (pseudocode), PHP yazım kurallarına göre yazılmış hali EK-2'de gösterilmiştir.



Şekil 2.14. Altı düğüm ve on kenardan oluşan ağırlıklı ve yönlü graf

Şekil 2.14'deki gibi 6 düğüm ve 10 kenardan oluşan ağırlıklı ve yönlü olarak verilen bir grafın Eş 2.8'deki ifadeye uygun olarak düzenlenen komşuluk listesi aşağıdaki gibidir.

G_1=(0, 1, 25)
G_2=(1, 0, 7)
G_3=(2, 1, 15)
G_4=(1, 3, 5)
G_5=(3, 2, 8)
G_6=(3, 0, 6)
G_7=(3, 4, 17)
G_8=(4, 3, 9)
G_9= (4, 5, 4)
G_{10}=(5, 0, 3)

Komşuluk listesinden komşuluk matrisini elde etmek için, EK-2'deki yordam işletildiğinde elde edilen matris Çizelge 2.4' deki gibi olacaktır.

Çizelge 2.4. Komşuluk listesinden elde edilen komşuluk matrisi

	0	1	2	3	4	5
0	0	25	∞	∞	∞	∞
1	7	0	∞	5	∞	∞
2	∞	15	0	∞	∞	∞
3	6	∞	8	0	17	∞
4	∞	∞	∞	9	0	4
5	3	∞	∞	∞	∞	0

2.3.2. Dijkstra algoritmasının çalışma şekli

Dijkstra Algoritması, düğümlerin etiketlenerek uçtan uca toplam ağırlık değerinin Eş 2.6'da verilen üçgen eşitsizliğine göre her adımda sınanmasına dayanır. Bu sınama işlemi "kenar rahatlama operasyonu" olarak da adlandırılır [3].

Dijkstra Algoritması çalışırken bellekte iki farklı düğüm kümesi tutar. Bunlardan birincisi, hedef düğüme olan en kısa uzaklığın hesaplandığı düğüm kümesidir. Diğeri ise, henüz hedef düğüme olan en kısa uzaklığı hesaplanmayan düğüm kümesidir. EK-3'de verilen Dijkstra Algoritması sözde kodunda birinci küme S, ikinci küme ise Q kümesi olarak isimlendirilmiştir [1, 3, 4].

Q kümesi sınanması gereken düğümleri içeren kümedir. Her düğüm sınandıkça, Q kümesinden çıkarılıp S kümesine yerleştirilir. Bu işlem Q kümesinde eleman kalmayıncaya kadar devam eder.

S kümesine geçen düğümlerin tümünün hedef düğüme olan d(v) değeri hesaplanmıştır. Program içerindeki işaretçi her döngüde, hedefe uzaklığı hesaplanan en son düğümü bir sonra ki adımda hesaplanacak düğüme hedefmiş gibi göstermektedir. Başka bir ifadeyle kayan bir sınama noktası oluşturulmuştur.

EK-4'de, PHP dili yazım kurallarına göre hazırlanmış olan Dijkstra fonksiyonu gösterilmektedir. Bu fonksiyon Dijkstra Algoritması kullanarak, bir veri iletişim ağındaki tüm düğümler arasındaki en kısa yolu hesaplamakta ve bu ağda kullanılan yönlendirme matrisi bu sayede oluşturulabilmektedir.

2.3.3. Dijkstra algoritmasının çalışma zamanı

Arama ve sıralama algoritmaları birbirleriyle karşılaştırılırken en temel kriterler arasında zaman karmaşıklığı (time complexity) ve uzay karmaşıklığı (space complexity) kavramları yer almaktadır.

Uzay karmaşıklığı, problemin girdi sayısına bağlı olarak bellekte sarf edilecek alan miktarını ifade eder. Diğer yandan, problemin zaman karmaşıklığı, o problemin kaç adımda çözülebileceğini girdi boyutunun fonksiyonu olarak gösteren ifadedir. Örneğin N bit girdisi olan bir problemin çözümü için N^2 adet adım gerekiyorsa bu problemin Zaman Karmaşıklığı O (N^2) olarak gösterilir [12].

Bir graf probleminin en temel anlamda girdisinin düğüm ve kenar olduğunu düşünülürse, problemin zaman karmaşıklığı düğüm ve kenar sayısının fonksiyonu olarak ifade edilebilir. Örneğin, grafımızdaki düğüm sayısını V, kenar sayısını ise N olarak ifade edersek Bellman-Ford Algoritması ile yapılacak çözülmemenin zaman karmaşıklığı, O(V.E) olarak tanımlanmaktadır. Diğer bir ifadeyle Bellman-Ford Algoritması ile problemin çözüm zamanı, düğüm ve kenar sayılarının çarpım değerinin büyüklüğüne bağlıdır.

Dijkstra Algoritması'nın zaman karmaşıklığı, kenar rahatlaması sürecinin nasıl bir yöntemle işletildiğiyle doğrudan ilişkilidir. Şayet kenar rahatlaması süreci, basit bir döngü içerisinde işletilirse, Dijkstra Algoritması'nın zaman karmaşıklığı düğüm sayısının karesinin fonksiyonu olarak ifade edilir. Daha seyrek graflar için zaman karmaşıklığı İkili Yığın (Binary Heap) veya Fibonacci Yığın (Fibonacci Heap) yöntemleriyle daha da iyileştirilebilir.

Çizelge 2.5'de Dijkstra Algoritması ve diğer en kısa yol algoritmaları için düğüm sayısı V ve kenar sayısı E cinsinden zaman karmaşıklığı tablosu verilmiştir.

Çizelge 2.5. En kısa yol hesaplama algoritmalarının zaman karmaşıklıkları

Dijkstra Algoritması	Basit Döngü İle	İkili Yığın İle	Fibonacci Yığın İle
	$O(V^2)$	$O((V+E)\log V)$	$O(V+E\log V)$
Bellman-Ford Algoritması	$O(V.E)$		
Johnson Algoritması	$O(V^2.\log V + V.E)$		
Floyd-Warshall Algoritması	$O(V^3)$		

3. YÖNLENDİRME PROTOKOLLERİ VE TRAFİK MÜHENDİSLİĞİ

3.1. İnternet ve Trafik Yönlendirme

Geride bıraktığımız son on senede, internet tüm dünyada yayınlaşmasına tanık olunmaktadır. Bugün yaşantımızın vazgeçilmez bir parçası haline dönüşen internet, son kullanıcılarına servis sağlayıcılarının veri iletişim omurga ağları üzerinden sunulmaktadır. Artan ihtiyaçları karşılayabilmek için bu omurgalar sürekli genişletilmekte ve yenilenmektedir. Birçok servis sağlayıcı, omurgaları üzerinden kullanıcılarına web, e-mail gibi temel internet servislerinin yanı sıra, telefon, video konferans, radyo, TV gibi tüm telekomünikasyon hizmetlerini vermek istemektedirler. Servis sağlayıcıların farklı karakteristiklere sahip tüm bu servisleri tek bir omurga üzerinden verebilmek için, ciddi bir trafik mühendisliğine gereksinim duymaktadırlar.

Trafik mühendisliği (TE), farklı önceliklere sahip, ses, video ve veri trafiğinin ortak bir taşıma platformu üzerinden sağlıklı olarak nasıl transfer edileceğiyle ilgilenmektedir.

İnternet omurgası, birçok servis sağlayıcının ara-bağlantılar ile birbirine bağlanmasından oluşan global bir omurgadır. Dolayısıyla böyle bir ağ, tek bir otorite tarafından yönetilmediğinden anarşik bir yapıya sahiptir.

Aralarında doğrudan bağlantı olan servis sağlayıcılar komşuluk ilişkisi içerisindedirler. Her servis sağlayıcı, sadece kendi komşuları arasında kararlı bir trafik akış ilişkisi kurmayı amaçlar. Bu nedenle, servis sağlayıcılar, hem kendi iç omurgalarında hem de doğrudan komşularıyla olan bağlantıları üzerinde trafik mühendisliğine ihtiyaç duyarlar.

Servis sağlayıcılar, kendi içindeki ve sınırlarındaki trafiği yönlendirebilmek için, çoğu IETF (Internet Engineering Task Force) tarafından standartlaştırılmış olan

yönlendirme protokollerini (routing protocol) kullanılırlar. Bu protokoller, ağ operatörlerinin girdikleri çeşitli parametre değerlerine göre trafiğin akışını belirler.

3.2. Yönlendirme Protokolleri

Yönlendirme protokolleri, bir noktadan kaynaklanan trafiğin hedefine hangi yolla gideceğini belirlemekte kullanılır. Bu anlamda yönlendirme protokollerinin temel işlevlerini şöyle sıralanabilir :

i) Ağın durumunu ve linklerdeki değişimi izleyip anons etmek.
ii) Muhtemel her yön için ağ üzerindeki en kısa yolu hesaplamak.
iii) Veri paketlerini kaynaktan hedefe doğru yönlendirmek.

Tüm yönlendirme protokolleri, aktaracağı trafik için en uygun yolu bulmaya çalışır. Başka bir ifadeyle alternatif yolları elindeki uygunluk fonksiyonuna göre eler ve kendince en iyi yolu hesaplar. Her protokolün kendisine ait yol bulma kriterleri bulunmaktadır. Genel olarak bu kriterler, atlanan düğüm sayısı (hop count), linklerin bant genişliği, linklerdeki gecikme ve operatörlerin filtrelerle belirlediği kendine özel politikaları olarak sıralanabilir.

Yönlendirme protokollerinin en uygun yolu nasıl hesapladıkları RFC (Request for Comments) olarak adlandırılan standart metinlerinde ayrıntılarıyla tanımlanmıştır.

Herhangi bir yönlendirme protokolünce seçilen yol veya yollar ağdaki bir çok trafik kaynağı için uygun olarak seçilmişse ve yolun kapasitesi bu yolu kullananların trafik taleplerinin toplamından az olmaya başlarsa, o yol artık çoğu trafik kaynağı için uygun bir yol olmaktan çıkar. Kaynakların trafik talebi önceden kesin olarak bilinmediğinden, omurgadaki bazı linklerde aşırı yoğunluklar yaşanırken alternatif güzergahlarda ise düşük link kullanımları görülebilir. Bu durum yönlendirme protokollerinin etkin kullanılamamasından kaynaklanır. Bu nedenle yönetilen ağın durumuna göre hangi yönlendirme protokolünün veya protokollerinin kullanılması gerektiğine doğru karar vermek gerekir. Ayrıca kullanılan protokolün

parametrelerinin, ağ elemanlarının kapasitelerine ve trafik akışlarına göre hesaplanarak atanması dengeli trafik dağılımı açısından hayati önem taşır.

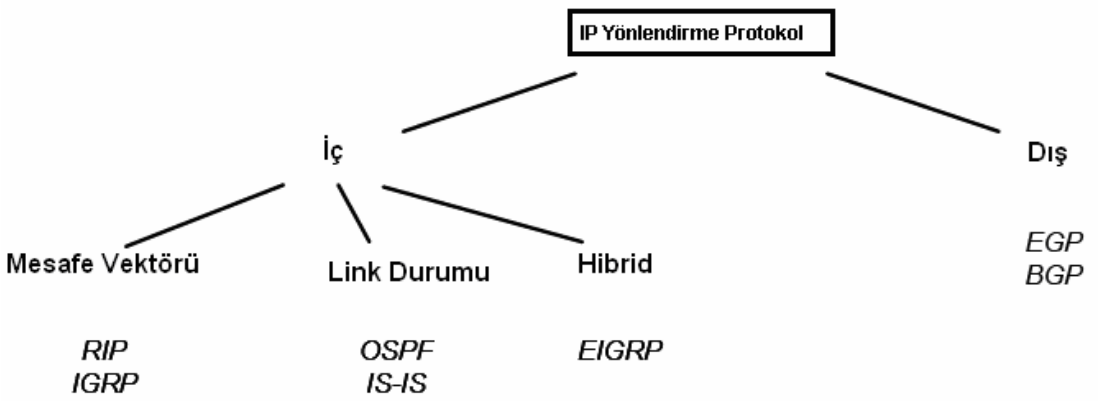
3.2.1. Yönlendirme protokollerinin sınıflandırılması

Yönlendirme protokolleri, genel anlamda omurga içi ve dışı olmak üzere iki kategoriye ayrılırlar. Omurga içinde kullanılan protokoller IGP (Interior Gateway Protocol) olarak adlandırılır. IGP protokolleri :

i) RIP (Routing Information Protocol) ,
ii) IGRP (Interior Gateway Routing Protocol),
iii) EIGRP (Enhanced Interior Gateway Routing Protocol) ,
iv) OSPF (Open Shortest Routing First) 'dir [5, 6] .

Omurgalar arası sınırda kullanılan yönlendirme protokolleri ise EGP (Exterior Gateway Protocol) olarak adlandırılır ve bugün BGP (Border Gateway Protocol) neredeyse kullanılan tek EGP protokolüdür [4].

İnternet yönlendirme protokollerini sınıflandıran ağaç Şekil 3.1'de gösterilmiştir.



Şekil 3.1 Yönlendirme protokollerinin sınıflandırılması [4]

Bir ağda yönlendirmeden sorumlu olan elemanlara Yönlendirici (Router) adı verilir. Yönlendiriciler, ağ üzerindeki istemci, sunucu gibi terminallerden kaynaklanan internet trafiğinin hedeflerine iletilmesinden sorumludurlar.

Yönlendiriciler üzerinden bir ya da birden fazla yönlendirme protokolü koşabilir. Örneğin omurganın sınırında olan bir yönlendirici, omurga içindeki yönlendiricilerle OSPF protokolü ile, omurga dışındakilerle ise BGP protokolü ile konuşabilir.

3.2.2. Yönlendirme bilgi protokolü (RIP)

RIP protokolü, Bellman Algoritması'na dayanan ve 1988'de IETF tarafından standartlaştırılan bir yönlendirme protokolüdür.

RIP, en kısa yol bulma işlemini atlama sayılarına (hop count) bakarak yerine getirir. Kaynaktan hedefe giden yollar arasında en az kaç yönlendirici atlanacaksa o yol RIP için tercih edilen yoldur.

RIP protokolün en büyük dezavantajı alternatif yollar arasındaki bant genişliği, yoğunluk, gecikme gibi önemli parametreleri dikkate almamasıdır.

RIP, ağ içerisindeki değişiklikleri yönlendiricilerin birbirlerine periyodik olarak gönderdiği "yönlendirme durum paketlerinden" anlar. Ağda bir değişiklik olsun ya da olmasın bu bilgiler belirlenmiş bir süre zarfında dağıtılır. Büyük ağlarda sürekli gönderilen durum bilgileri ciddi bant genişliği israfına neden olabilir [5, 6].

3.2.3. İç kapı yönlendirme protokolü (IGRP)

Yönlendirici cihaz pazarının öncüsü olan Cisco Firmasının kendi laboratuarlarında geliştirmiş olduğu bir yönlendirme protokolüdür. IGRP'de uygun yol hesabında linklerin bant genişliği, gecikme, güvenilirlik ve yük değerlerini göz önüne alınır.

Kaynak ile hedef arasındaki yolun ağırlık değeri Eş. 3.1'deki ifadeden hesaplanır [4].

$$A\ D = [K1 * b + (K2 * b/(256 - yük) + K3 * d] * (K5/(güvenilirlik + K4)] \quad (3.1)$$

$b = 256 * 10^7/\text{Min(bant genişliği)}$, $d = 256 * \text{delay}$

Yolun yük ve güvenilirlik değerleri trafiğin aktarımı sırasında yönlendirici tarafından hesaplanır. Min(bant genişliği) ifadesi ise, yol boyunca bulunan linkler arasında, bant genişliği en küçük olan linkin sahip olduğu bant genişliği değeridir. Başka bir ifadeyle zincirin en zayıf halkasını temsil eder.

Cisco yönlendiricilerde, K1, K2 ve K3 için 1 ; K2, K4, ve K5 için ise 0 ön tanımlı değerler olarak atanmıştır. Bu durumda, ön tanımlı ağırlık değeri Eş. 3.2'deki ifadeden hesaplanır.

$$M_{default} = [10^7/\text{Min(Bant Genişliği)} + \text{gecikme}] * 256 \quad (3.2)$$

Eş. 3.2, minimum bant genişliği büyük ve gecikmesi yüksek olan yolun ağırlık değerinin küçük olacağını söyler. Kaynak ile hedef arasındaki alternatif güzergahlar arasında ağırlık değeri en küçük olan yol, trafik akışı için tercih edilen yol olacaktır.

3.2.4. İlk önce açık en kısa yol (OSPF)

OSPF protokolü, geniş omurgalar içindeki dinamik yönlendirme ihtiyacına çözüm olarak 1998 yılında IETF tarafından standartlaştırılan bir veri trafiği yönlendirme protokoldür [5, 6].

Aralarında OSPF protokolü ile konuşan tüm yönlendiriciler, birbirlerine kendi linklerine ait durum bilgisini aktarırlar. Her yönlendirici cihaz kendisine ait link-durum bilgilerini komşuluk yaptığı diğer tüm yönlendiricilere bildirir. Daha sonra, bu komşular da diğer komşularına aynı bilgileri iletir, kademe kademe tüm omurga

aynı ortak bilgilere sahip olur. Böylece tüm yönlendiriciler omurga içindeki tüm linklerin durumunu öğrenmiş olur.

Komşuluk ilişkisiyle, link bilgilerini tüm omurgaya yayma metoduna Link-Durum Algoritması (Link-State Algortithm) adı verilir. Şayet OSPF ağının topolojisinde bir değişiklik olursa, sadece değişiklik bilgisi komşulara bildirilerek, hesaplamalarını güncellemeleri sağlanır. Oluşan değişikliklere göre tüm yönlendiriciler yönlendirme veritabanlarını günceller. Böylece, en kısa yol ağacında değişiklikler meydana gelir.

OSPF protokolü, nispeten geniş omurgaların yönlendirme ihtiyacını karşılamak maksadıyla tasarlanmasına rağmen, çok büyük omurgalarda gerekli tedibirler alınmazsa ağı ciddi şekilde hantallaştırabilir. Zira omurga büyüdükçe o omurgaya mensup link sayısı, dolayısıyla da topolojiye ilişkin değişiklik olasılığı artacaktır. Her değişiklik sonrası tüm yönlendiricilerin yönlendirme veritabanında değişiklikler olacak ve bu değişikliklere göre yeniden hesaplamalar yapılarak en kısa yol ağacı elde edilecektir. Dolayısıyla ağda sıklaşan değişiklikler, hem yönlendiricilerin CPU, RAM gibi sistem kaynaklarının hem de bant genişliklerinin israf olmasına neden olacaktır. Böyle bir durumu kontrol altına alabilmek için omurga içinde alt OSPF ağları tanımlamak en çok başvurulan yöntemler arasındadır. OSPF protokolünü diğer IGP protokollerinden ayıran güçlü özelliklerinin başında, omurga içinde farklı çalışma alanlarının yaratılabilmesine imkan tanıması gelir [4] .

Her yönlendirici, ağdaki diğer tüm yönlendiricilerin link-durum bilgilerini içeren ortak veritabanına sahip olduğundan, hangi trafiğin hangi yönde gitmesi gerektiğine dair tüm verilere sahiptir. Her bir yönlendirici öncelikle kendisine ulaşan trafiğin hangi hedefe gitmek istediğine bakar. Daha sonra her bir hedef için veritabanındaki link durum bilgilerine göre en kısa yolu hesaplamaya başlar. Yönlendiricilerin hesapladıkları en kısa yol bilgisi STP (Shortest Path Tree) olarak isimlendirilir. Yönlendirme veritabanındaki her değişiklik sonrası STP yeniden oluşturulur. STP hesaplamasında Dijkstra Algoritması kullanılır [3].

Yönlendiriciler ağdaki diğer yönlendiricilerin link durumları hakında sahip oldukları bilgileri kullanarak yönlendirme veritabanlarını (routing database) oluştururlar. Her yönlendirici elindeki bu bilgileri kullanarak her yön için STP (Shortest Path Tree) olarak adlandırılan en kısa yol ağacını oluşturur.

SPT oluşturma süreci aşağıdaki gibi işler.

i) Ağ operatörü her bir yönlendiricinin her linkine ağırlık değeri atar. Bu değer 1 ile 65535 (2^{16}-1) arasında değişen bir tamsayı olabilir. Linklerin ağırlık değerleri belirlenirken, omurganın topolojisi ve kaynakların oluşturacağı tahmini trafik talebi göz önünde tutulur [4].
ii) Şayet operatör, linklerin ağırlık değerini girmez ise yönlendiriciler her linkin ağırlığını o linkin bant genişliğiyle ters orantılı olacak şekilde kendileri belirlerler. Farklı üreticilerin ürettikleri yönlendiricilerde ön tanımlı ağırlık değeri belirleme yöntemi farklı da olabilir [4].
iii) OSPF ağındaki tüm yönlendiricilerin link durum veri tabanı aynıdır. Ancak her yönlendirici kendi STP hesabını link ağırlık değerlerine bakarak kendisi yapar [4].
iv) Link durum veritabanındaki her değişiklik sonrasında STP yeniden hesaplanır.

3.2.5. Yönlendirme tablosunun oluşturulması

Trafik mühendisliğinin en önemli konusu, kaynak ve hedef noktaları arasındaki ağ trafiğinin her bir link üzerinde dengeli olarak taşınmasıdır. Ağ içerisindeki bazı linkler hiç kullanılmazken bazıları sürekli aşırı doluluk yaşıyorsa bu ağın trafik mühendisliği çok kötü yapılmış veya hiç yapılmamıştır.

Bir ağın trafik mühendisliğinin yapılması, mevcut ağ ve trafik kaynaklarını göz önüne alarak en iyi yönlendirme tablosunun oluşturulması anlamına gelir. Ağdaki trafik ne kadar iyi yönlendiriliyorsa, kaynak israfının da o kadar önüne geçilmiş olur.

OSPF protokolü gibi ağa ilişkin ortak bir veri tabanı kullanan ağlarda, trafik yönlendirme tablosunu belirleyen iki temel faktör vardır. Bunlar, linklerin durumu ve linklere atanan ağırlık değerleridir.

Ağın link durum tablosu, aşağıdaki olaylardan herhangi birinin gerçekleşmesi sonucunda değişir. Bunlar:

i) Herhangi bir link veya birden fazla linkin devre dışı kalması,
ii) Ağdaki herhangi bir yönlendiricinin devre dışı kalması,
iii) Ağa yeni bir link ilavesi edilmesi,
iv) Mevcut linklerden birinin iptal edilmesidir .

Bu olaylardan bir ya da bir kaçı meydana geldiğinde, bundan doğrudan etkilenen yönlendiriciler tüm ağı durum değişikliğinden haberdar edecek ve her yönlendirici yönlendirme tablosunu yeniden oluşturacaktır. Meydana gelen durum değişikliği, ağın topolojisine ve değişimin meydana geldiği yerin önemine bağlı olarak yönlendirme tablolarında ciddi veya önemsiz değişikliklere neden olabilir.

Ağda yapılan bakım çalışması, kapasite artırımı veya indirimi gibi durumlar dışındaki olaylar genel olarak ağ otoritesinin kontrolü dışındaki olaylardır. Bu nedenle yönlendirme tablosunun olumsuz olarak değişmesine neden olurlar. Bu kontrol dışı değişikliği minimize etmenin yolu, ağ için önem taşıyan linklerin yedeklenmesidir. Bu tür yedeklemelerde ciddi maliyet arttırıcı tedbirler olduğundan çoğu zaman kısmen yerine getirilir.

Trafik yönlendirme tablosunun planlı olarak oluşturulabilmesi, link kapasiteleri ve tahmini trafik değerlerinin göz önüne alınarak linklere doğru ağırlık değerleri atanmalıdır.

Linklere atanacak ağırlık değerlerinin belirlenmesi çeşitli yöntemlerle yapılabilir. Bu yöntemler arasında en çok kullanılanları aşağıda sıralanmıştır:

i) Birim Ağırlık Değeri: Her bir linkin ağırlık değeri 1 olarak atanır. Böylece ağımız ağırlıksız bir graf problemi olarak modellenmiş olur. En kısa yolu ifade eden yönlendirme tablomuz, en az atlanan düğüm sayısı değerine göre oluşturulmuş olur [7] . RIP yönlendirme protokolü tipik karakteristik olarak her link için birim ağırlık değeri kullanır. Birim Ağırlık Değeri yönteminde tahmini trafik değerleri ve link kapasiteleri göz önüne alınmadığından ciddi bir trafik mühendisliği vaat etmez.

ii) Ters Orantılı Kapasite Değeri: Her linke atanan ağırlık değeri o linkin bant genişliği değeriyle ters orantılıdır. Başka bir ifadeyle, kapasitesi yüksek linklerin ağırlık değeri düşük, kapasitesi düşük olan linklerin ağırlık değeri yüksek olur [7]. Böylece yüksek kapasiteli linklerin trafik taşıma ihtimalleri yükseltilmiş olur. Birim Ağırlık Değeri yönteminden daha iyi sonuçlar verse de, sadece linklerin bant genişliği değerlerine bakılıp tahmini trafik değerleri ihmal edildiğinde çok ciddi bir başarı vaat etmez.

iii) Rasgele Ağırlık Değeri Üretme : Tipik deneme yanılma yöntemiyle ağırlık değeri belirleme işlemidir [7]. Her bir link için birkaç defa rasgele ağırlık değeri atanır ve en iyi sonuç veren değerler linklere atanır. Daha çok, ağ yönetimindeki günlük operasyonlarda geçici çözüm olarak kullanılan bir yöntemdir.

iv) Sezgisel Taramalar : Her link için en uygun ağırlık değerini bulmak için sezgisel tarama yöntemlerine başvurulur. En iyi sonuçların değerlendirilmesinde ağ üzerindeki trafiğin akışı göz önünde tutulduğundan başarılı sonuçlar elde etmemizi mümkün kılar. Genetik Algoritma, Fortz ve Thorup lokal sezgisel arama bu yöntemler arasında yer alır [7].

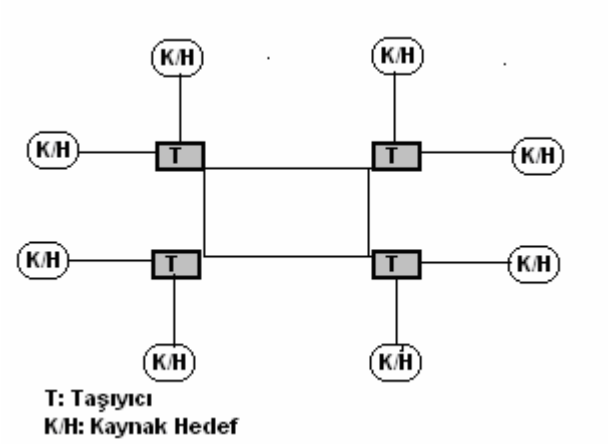
3.3 Trafik Akışı ve Link Kullanımı

3.3.1. Trafik matrisi

Bir iletişim ağındaki kaynak ve düğüm noktaları arsındaki trafik hacmini ifade eden matrise Trafik Matrisi (TM) adı verilir. Bir internet ağında trafik yönlendirmekte kullanılan cihazlar, aktif ağ cihazları olarak isimlendirilir ve modellemelerde düğüm olarak gösterilirler [8].

Trafik açısından, bir ağda temel olarak iki farklı aktif ağ cihazı bulunmaktadır. Bunlardan birincisi, trafiğin kaynaklandığı veya sonlandırıldığı cihazlardır. Bu kaynak veya hedef düğümleri uç düğümler veya terminaller olarak adlandırılır. İkincisi ise, trafiği uçlar arasında taşıyan aktif cihazlardır. Bunları ise taşıyıcı düğümler veya taşıyıcılar olarak adlandırılır.

Ağ içerisinde taşıyıcı düğümlerin görevi kendisine gelen trafiği, yönlendirme tablosuna bakarak bir sonraki düğüme iletmektir. Şekil 3.2'de gösterilen örnek bir ağın kaynak/hedef düğümleriyle taşıyıcı düğümleri işaretlenmiştir. Şekil 3.2'den kolayca görüleceği gibi kaynak/hedef düğümler ağın dış kenarında, taşıyıcı düğümler ise merkezinde yer almaktadırlar. Ağ topolojisinin karmaşıklığına bağlı olarak daha katmanlı bir taşıyıcı yapısı da oluşturulabilir.



Şekil 3.2. Taşıyıcı ve kaynak/hedef düğümler topolojisi

Ağdaki toplam düğüm sayısından taşıyıcı düğüm sayısını çıkarıldığımda uç düğüm sayısını elde edilir. Bir iletişim ağında uç düğüm sayısını N ile gösterirsek, tüm uçlar arasında tanımlanabilecek maksimum trafik akışı sayısı Eş 3.3 'den elde edilir [8].

$$\text{Maksimum Trafik Akış Sayısı} = N^2 - N \qquad (3.3)$$

Trafik matrisi her bir uç çiftinin arasında akan trafik değerini gösteren tek boyutlu bir matristir. Uç çiftlerini P_{ij} olarak gösterirsek trafik matrisimiz Şekil 3.3'deki gibi olacaktır.

TM =

$P_{0\,1}$
$P_{0\,2}$
……..
……..
$P_{i-1\,j}$
P_{ij}

Şekil 3.3. Trafik matrisinin genel ifadesinin görünümü

Yüklü bir ağın trafik matrisinin elde edilmesi çoğu zaman problemli bir mesele olmakla birlikte, ağın kapasite planlaması, yükün dengelenmesi ve doğru yönlendirme tablolarının oluşturulabilmesi için mutlaka gereklidir.

Trafik matrisini elde edebilmek için tüm ağı dinleyen ve her uç birimden aldığı örneklerle trafik ölçümü yapan araçlara ihtiyaç vardır. Piyasada açık kaynak kodlu veya lisanslı olarak temin edilebilecek bu ölçüm yazılımları, giden veri paketlerinden çeşitli oranlarda örnekler alarak kaynak ve hedef adreslerine bakarlar. Böylece ağ içerisindeki hangi uçlar arasında ne miktarda trafik akışı olduğunu tespit ederler. Özellikle büyük ağlarda, bu tür bir ölçüm için bir çok uçbirimden veri toplamak, depolamak ve işlemek masraflı ve zaman alıcı bir iştir. Bu ölçümleri daha uygulanabilir kılmak için yeni örnekleme ve istatistiksel değerlendirme konularında .alışmalar devam etmektedir [8] .

3.3.2. Yönlendirme matrisi

Bölüm 3.2.5'de yönlendiricilerin yönlendirme tablolarını nasıl oluşturdukları ayrıntılarıyla incelemişti. Yönlendirme matrisi (RM), terminaller arasındaki trafik akışının hangi linkler üzerinden yapılacağını gösteren matristir. Yönlendirme matrisi

yönlendirme tablolarından elde ediliyor olup, yönlendirme tablosunda oluşan herhangi bir değişiklik yönlendirme matrisinin de anında değişmesine neden olur.

Yönlendirme matrisinin boyutu ağdaki link sayısına ve tüm uçlar arasında tanımlanabilecek maksimum trafik akışı sayısına bağlıdır. Tüm terminal çiftleri arasında tanımlanabilecek trafik akışı sayısının ağdaki terminal düğüm sayısının Eş 3.3 'den elde edildiği görülmüştü. Ağdaki link sayısına L, çiftler arası maksimum trafik akışı sayısına da P denilirse, yönlendirme matrisi LxP boyutunda bir matris olacaktır. Yönlendirme matrisi sadece 1 ve 0 değerlerinden oluşur. Herhangi bir terminal çiftinin akış yolunda kullanılan linklerin değeri 1, kullanılmayan linklerin değeri ise 0 olarak işaretlenir. Şekil 3.4'de düğüm numaralarına göre indekslenmiş örnek bir yönlendirme matrisi verilmiştir.

	$P_{0\,1}$	P_{02}	. . . . .	$P_{i-1\,j}$	$P_{i\,j}$
$L_{0\,1}$	1	1	. . . . .	1	1
$L_{0\,2}$	0	1	. . . . .	1	0
.	. . . . .		. . . . .	. . . . .	
.	. . . . .		. . . . .	. . . . .	
.	. . . . .		. . . . .	. . . . .	
.	. . . . .		. . . . .	. . . . .	
			. . . . .		
$L_{k-1\,m}$	0	1	. . . . .	0	0
$L_{k\,m}$	1	0	. . . . .	0	1

Şekil 3.4. Yönlendirme matrisi

3.3.3. Link kullanımı matrisi

Link kullanım matrisi, ağdaki tüm linklerin kullanım değerlerini gösteren matristir.

Ağdaki yönlendirme matrisinin veya trafik matrisinin değişimi sonucu link matrisi de değişir. Yönlendirme matrisi RM, trafik matrisi ise TM olarak gösterilirse, link kullanım matrisi Eş. 3.4 'de görüldüğü gibi bu iki matrisini çarpımından elde edilir [8].

$$LM = RM \times TM \quad (3.4)$$

İnternet Protokolü (IP) kullanan bir iletişim ağında, linklerin kullanım değerleri IETF (Internet Engineering Task Force) ağ izleme ve görüntüleme standardı olan SNMP (Simple Network Management Protocol) protokolü ile kolayca geçekleştirilir [8].

SNMP, ağdaki aktif cihazların her türlü durumunu izlemek için kullanılan çok maksatlı bir protokol olup, MRTG (Multi Router Traffic Grapher) gibi açık kaynak kodlu uygulamalar SNMP protokolü sayesinde aktif ağ cihazlarının link yoğunluğunu gösterirler.

Trafik matrisinin ölçümü, veri paketlerinin kaynak ve hedef adres bilgilerine bakmak zorunda olduğundan ve bu nedenle daha özel örnekleme ve istatistiksel tekniklere ihtiyaç duyduğundan daha zor ve maliyetli bir iştir. Diğer yandan link yoğunluğunun ölçümü SNMP protokolü sayesinde, veri paketleri üzerinde özel incelemeler yapmaya gerek duymaksızın, kolayca yapılabilmektedir.

Yönlendirme matrisinin ve link kullanım matrisinin elde edilmesi trafik matrisinin elde edilmesinden çok daha kolaydır. Bu nedenle, trafik matrisi, yönlendirme matrisinin tersi RM^{-1} ve link kullanım matrisi ile Eş.3.5'de gösterildiği gibi hesaplanır.

$$TM = RM^{-1} \times LM \quad (3.5)$$

3.3.4. Link kullanım değerlerinin hesaplanması

Servis sağlayıcılar, kapasite planlaması, ağdaki aktif cihazların veya linklerin devre dışı kalması ve linklerin ağırlık değerlerinin belirlenmesi maksatlarıyla tahmini trafik matrislerini kullanırlar. Daha gelnel bir ifadeyle tahmini trafik matrisi, ağda yaşanabilecek farklı durumlar için trafik akışının simüle edilmesinde kullanılır.

Gerçek trafik taşıyan bir ağda, sabit bir trafik matrisinden bahsedilemez. Örneğin gece gündüz, hafta sonu gibi farklı saat ve tarihlerde terminallerin trafik talepleri değişeceğinden farklı zamanlarda farklı trafik matrisleri elde edilir.

İdeal olarak yönetilen bir ağda, tahmini trafik matrisi, en yoğun saatlerde ölçülen trafik kullanım değerlerine göre belirlenmeli ve ağda yaşanabilecek çeşitli sorunlar da dikkate alınarak trafik akışının düzgün olup olmadığı simüle edilmelidir. Bu tür simülasyonlar sonucunda, bazı linklerde tıkanmalar yaşanırken, alternatif güzergahlarında boş kapasite olması linklerin ağırlık değerlerinin düzgün belirlenemediğini gösterir.

EK-5'de PHP dili yazım kurallarına göre hazırlanmış fonksiyon, kullanıcı tarafından girilen tahmini trafik matrisini kullanarak bir OSPF ağının tüm linklerinin kullanım değerlerini hesaplamaktadır.

EK-6'da ise hesaplanan link değerlerini kullanarak ağdaki tüm linklerin kullanım değerlerini gösteren ve link kullanımları hakkında gelen bir istatistik sunan fonksiyon yer almaktadır.

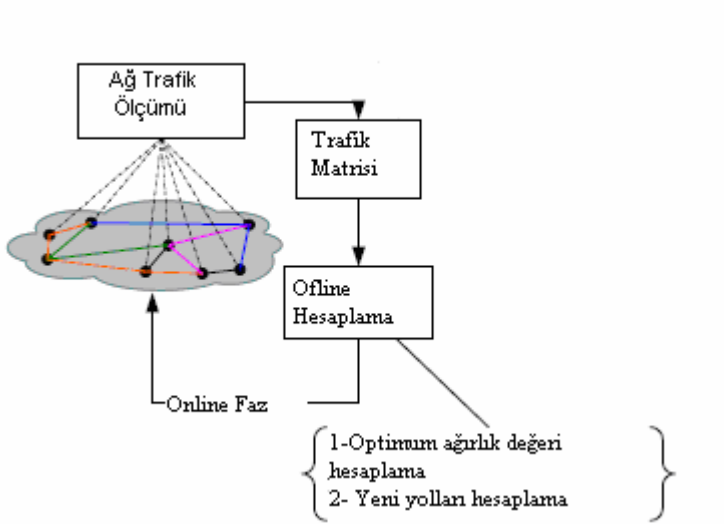
3.3.5 Tahmini trafik matrisi ile ağırlık değerlerinin belirlenmesi

Bir ağdaki linklerin ağırlık değerinin değiştirilmesi, ağın yönlendirme matrisinin yeniden hesaplanmasına neden olur. Bu nedenle linklerin ağırlık değerleri dolaylı olarak link kullanım değerlerini etkiler.

Bir ağda kullanılması gereken optimum link ağırlık değerlerini ararken, dikkat edilmesi gereken husus, link kullanım değerlerinin arzu edildiği gibi dengeli olup olmadığıdır. Ancak canlı trafik taşıyan bir ağda tahmini trafik matrisi zamanın fonksiyonu olarak değişeceğinden, link ağırlık değerleri sabit kalsa bile yük kullanım matrisi değişecektir. Bu nedenle, tek bir tahmini trafik matrisine göre elde edilen ağırlık değerleri mutlak iyi değerler olarak düşünülemez. Bu nedenle ağda kullanılacak link ağırlık değerlerinin, çeşitli senaryolar için üretilen tahmini trafik

matrisine göre bir çok defa hesaplanması gerekir. Böylece ağda yaşanması olası her türlü trafik değişikliğine karşı önceden belirlenmiş ağırlık değerine sahip olunacaktır.

Link ağırlık değerlerinin belirlenmesinde, uygulaması yukarıda anlatılan yöntemden daha zor olan ikinci bir yöntem ise, Şekil 3.5'de gösterildiği gibi trafik matrisinin bir faz geriden online olarak takip edilip, ağırlık değerlerinin hesaplanmasıdır.

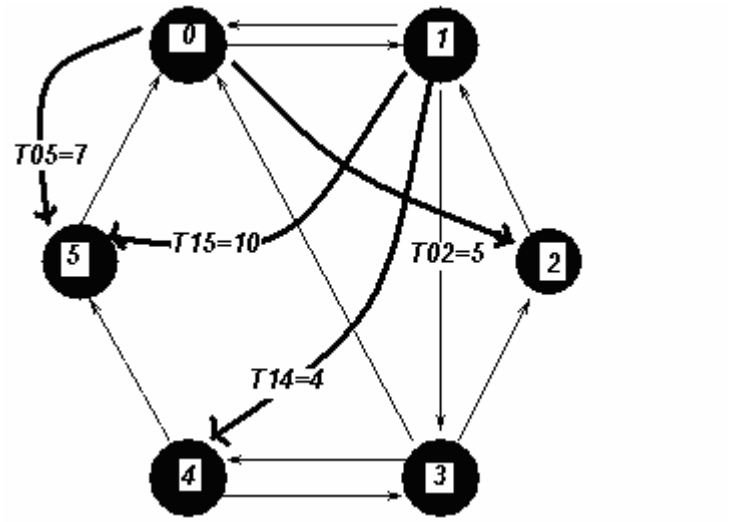


Şekil 3.5. Online trafik takibi süreci [9]

Tahmini trafik matrisi yerine Şekil 3.5'deki yöntemin kullanılmasının avantajı, ağırlık değerlerinin hesaplanmasında birebir gerçek trafik değerlerinin kullanılmasıdır. Ancak, ağın büyüklüğü, offline link ağırlık değeri hesaplama yönteminin karmaşıklığına bağlı olarak faz farkı artacağından, ciddi trafik değişimlerinin birebir takibi her zaman mümkün olmayabilir.

3.3.6 Tahmini trafik matrisinin hazırlanması

Bir ağda tanımlanabilecek maksimum trafik akış sayısı Eş 3.3'de verilmiştir. Bir ağda tanımlı maksimum trafik akış sayısının P olduğu düşünülürse, hesaplamalarda kullanılacak tahmini trafik matrisinin boyutu 1xP olmalıdır.



Şekil 3.6 Yönlü trafik akışı olan bir ağ

Şekil 3.6'da verilen örnek ağ için trafik matrisi elemanları:

$P_{02}=5$,
$P_{05}=7$,
$P_{15}=10$,
$P_{25}=4$,
olarak bulunur.

Şekil 3.6 için tanımlanabilecek çift sayının 30 olduğu görülmektedir (Bkz Eş 3.3) . Bu durumda belirtilmeyen tüm çiftlerin değeri otomatik olarak 0 atanarak link kullanım matrisinin doğru hesaplanabilmesi sağlanmalıdır.

P_{02} olarak indekslenmiş terimimiz, 0. düğüm ile 2. düğüm arasındaki trafik akışını temsil eder. Aynı temsil biçimi diğer trafik akış elementleri için de geçerlidir.

Herhangi bir düğümün herhangi bir linki üzerinden birden fazla trafik akışı tanımlanabiliyorsa, o link üzerinde tahmini trafik akışlarının toplam değeri, linkin bant genişliğinden büyük olmamalıdır.

Tahmini trafik matrisimiz ne kadar fazla element içeriyorsa, gerçek bir ağı o kadar fazla modelliyor demektir. Aksi takdirde, kullanacağımız trafik matrisimiz, optimizasyon için uygun değildir. Bu nedenle çalışmalarda trafik matrisleri tüm düğümleri kapsayacak biçimde tasarlanır.

4. GENETİK ALGORİTMA İLE OPTİMUM AĞIRLIK DEĞERİ BULMA

4.1. Genetik Algoritma

Genetik Algoritma (GA), evrimsel hesaplama yöntemi kullanan yapay zeka uygulamalarından biri olarak nitelendirilebilir. Adından da anlaşılacağı üzere Darwin'nin evrim teorisinden ilham alınarak modellenmiş olup, doğal seleksiyon ve doğal genetik mekanizmaya dayanan araştırma metodu kullanır. En iyi çözümü elde edebilmek için, zayıf çözümleri evrimsel bir işleyişe göre eleme yoluna gider. En iyi çözüm, yapılan çevrimler (iteration/yineleme) sonucunda hala hayatta kalabilmeyi başaran çözümdür [9, 10].

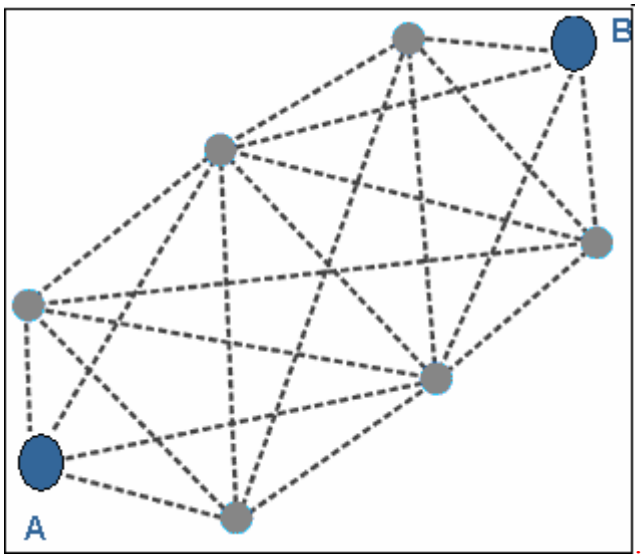
Evrimsel hesaplama metodu ilk olarak I. Rechenberg'in 'Evrim Stratejileri' eserinde tanıtılmıştır. Ardından Jonh Holland, evrim sürecini bilgisayar ortamında kullanarak genetik algoritmaları oluşturmuştur. Holland'ın ögrencisi olan D.E Goldberg 1989 yılında GA'nın çeşitli konularda kullanılabileceğini gösteren bir kitap yayınlamıştır. 1992 yılında ise John Koza genetik algoritmayı kullanarak genetik programlamayı geliştirmiştir [9, 10].

GA'nın bir çok mühendislik probleminin çözümüne ilişkin uygulamaları mevcuttur. GA, geleneksel yöntemlerle çözümü zor veya imkansız olan bir çok problem üzerinde uygulanabilir. Deneysel çalışmalardan, endüstriyel uygulamalara kadar birçok alanda GA çalışmaları yapılmaktadır [10]. Mühendislikte, çözüm uzayı geniş olan problemlerin, muhtemel çözümlerinin taranarak en iyinin bulunmasının çok zaman alıcı olduğu durumlarda, zaman kazandırıcı bir yöntem olarak da karşımıza çıkar.

Genetik Algoritma, problemin ele alındığı ortamda yer alan uygun ve güçlü çözümlerin yaşatılarak, iyiler arasından daha da iyileştirilmiş çözümler üretmeyi amaç edinen bir yöntemdir. Ancak her problem için en optimum sonucu bulunduğunu garanti etmez.

GA' nın işleyişinde en temel unsur "uygunluğun" ne olduğunun tarifinde yatar. Matematiksel anlamda uygunluk, modellenmesi ve muhtemel her çözüm üzerinde uygulanması gereken bir fonksiyondur.

Çok temel bir örnek olarak iki şehir arasında onlarca yol olduğunu ve bu yolların çoğunun Şekil 4.1'deki gibi birbiriyle kesiştiğini düşünelim. Yollar arasındaki bu kesişim noktalarından yeni güzergahlar elde edilir. Böylece alternatif güzergah sayısı artacaktır. Şimdi alternatiflerin en uzundan en kısaya kadar sıralandığını ve uzun olanların elendiğini düşünelim. Şayet mevcut yollardan yeni güzergahlar türetilmeden en uzun yollar hemen elenseydi belki de daha kısa güzergahlar elde edilemeyecekti. Zira uzun yollar diğer yollarla kesiştiğinde, daha kısa yolların oluşmasına imkan tanıyabilir.



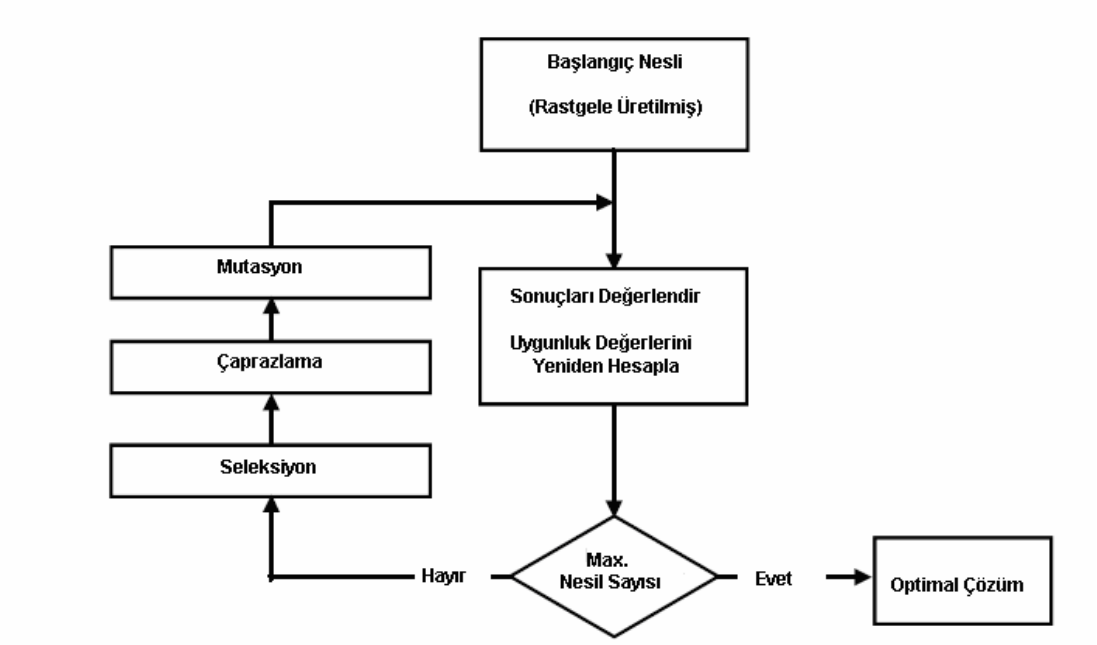
Şekil 4.1. İki nokta arasındaki olası güzergahlar

İlk bakışta, en uygun yol en kısa yol olarak tarif edilebilir. O zaman uygunluk fonksiyonu mesafe ile ters orantılı olmalıdır. İki nokta arasındaki trafik çok yoğunsa ve sadece tek bir güzergah ile taşınamıyorsa uygunluk fonksiyonu, mesafe ve trafik yoğunluğu değişkenlerinin her ikisini de kapsayacak şekilde yeniden tasarlanmalıdır. Bazı yolların paralı olduğunu ve fiyatlarının da değişik olduğunu düşünelim. Bu durumda ise en iyi yol, mesafe, yoğunluk ve maliyet değişkenleri dikkate alınarak

tasarlanmış bir fonksiyonla bulunabilir. Özetle, amaca götüren bir çok çözüm olabilir ama hangi çözümümün en iyi olduğunu bulabilmek için kısıtlamaların iyi tarif edilmesi ve bu tarife göre mevcut çözümler arasından en iyilerin seçilmesi gerekir.

4.2. Genetik Algoritmanın Çalışma Prensibi

GA' nın akış şeması en genel haliyle Şekil 4.2 gösterilmiştir.



Şekil 4.2. GA 'nın genel akış şeması [14].

Şekil 4.2' de görüldüğü gibi GA' nın temel süreçleri, nesillerin uygunluğunun sınanması ve her nesilde yer alan bireylerin eleme, çaprazlama ve mutasyon işlemlerine tabi tutulmasıdır.

4.2.1. Kromozomların kodlanması ve başlangıç nesli

GA, belirli problemlerin muhtemel çözümlerini kromozom benzeri veri yapıları şeklinde kodlar. Böylece her çözüm belirli kriterlere göre kodlanmış diziler haline getirilir ve bu diziler kromozom olarak adlandırılır [9, 10].

Kromozomların kodlanması farklı şekillerde olabilir. Kodlamalar arasında en kullanışlı olanlar ikili kodlama, permütasyon kodlaması ve değer kodlamasıdır.

i) İkili Kodlama: GA' da ilk kullanılan ve en yaygın olan kodlama şeklidir. Her kromozom 1 ve 0'lardan oluşan katarlar halinde ifade edilir [9].
ii) Permutasyon Kodlaması: Daha çok sıralama problemlerinde kullanılan kodlama yöntemidir. Görev ya da güzergah sıralaması problemlerinde kullanılabilir [9].
iii) Değer Kodlaması: Bünyesinde gerçek sayısal değerler barındıran problemlerde kullanılan kodlama şeklidir. Bu tür problemleri ikili kodlama ile çözmeye çalışmak daha zor bir yol olabilir. Böyle durumlarda, kromozomlar gerçek değerlerle doğrudan kodlanır [9].

Başlangıçta, elimizde var olan çözümler başlangıç neslini ifade eder. Şayet elimizde var olan bir başlangıç nesli yoksa başlangıç kromozomları rasgele üretilebilir. Daha sonra, doğal seleksiyon sürecini başlatabilmek için çaprazlama ve mutasyon operatörleri kullanılarak yeni nesiller türetilir.

Ağırlık değerlerinin GA ile arandığı bu çalışmada, her bir linkin ağırlık değeri gen, tüm ağın gen değerleri ise kromozom olarak ifade edilmiştir. Böylece, her bir kromozom ağı için olası bir çözümü ifade ederken, nesil çözüm uzayını ifade etmektedir.

Bu çalışmada, sadece tek bir kromozomdan oluşacak başlangıç neslinin program operatörü tarafından bir defaya mahsus olarak girilmesi istenmiştir. Ayrıca sonraki nesillerin kaç kromozomlu olacağı da operatörün arzusuna bırakılmıştır. Böylece, operatörün farklı kromozom sayılarına göre yaptığı araştırmalar arasındaki farkı görebilmesi hedeflenmiştir.

EK-7'de PHP yazım diline uygun olarak hazırlanan nesil oluşturma kodları gösterilmiştir. Kodlamanın 3. satırında, operatörün girdiği ağırlık değeri kümesi W'nın başlangıç nesli olarak alındığı görülmektedir. 7 ile 19. satırlar arasında ise

operatörün seçtiği kromozom sayısı kadar rasgele kromozom üretilerek birinci nesil oluşturulmuştur.

4.2.2. Popülasyon büyüklüğü ve kromozom uzunluğu

Popülasyon büyüklüğü, bir nesilde bulunan birey sayısı ile ifade edilir. Bir problemin çözümünde büyük bir popülasyonun kullanılması, problemin çözüm uzayının iyi örneklendiği anlamına gelir. İyi örneklenmiş bir çözüm uzayı ise en iyi çözümün bulunma olasılığını artıracaktır [10]. Diğer taraftan, bir problemin çözüm zamanı, popülasyonun büyüklüğüne ve çevrim sayısına doğrudan bağlıdır. Bu nedenle, popülasyon genişliğinin gereğinden büyük seçilmesi, çözüm sürecini geciktireceği gibi iyi çözüme ulaşmamıza katkıda da bulunmayabilir. Sonuç olarak, popülasyon büyüklüğünün üzerinde uğraşılan probleme bakılarak belirlenmesi en uygun yoldur.

Bir probleme GA ile çözüm arama sürecinde kromozom uzunluğu da oldukça önemli bir parametredir. Ancak kromozom büyüklüğü, popülasyon büyüklüğünden farklı olarak çözümleyici tarafından belirlenebilir bir parametre olmayabilir. Burada ele alınan örnekte, ağdaki tüm linklerin ağırlık değerleri tek bir kromozom olarak ifade edildiğinden, kromozom büyüklüğü ele alınan ağın topolojisine bağlıdır.

4.2.3. Çaprazlama ve mutasyon

Mevcut neslin seçilmiş kromozomlarından yeni bireyler oluşturmak üzere çaprazlama işlevi yürütülür. Bu operatörün temel fonksiyonu, yeni nesillerin, atalarının birebir kopyası olmalarını engellemektir.

Çaprazlama operatöründen kromozomların iyi özelliklerini birleştirerek daha iyi kromozomlar oluşturması beklenir. Böylece arzu edilen en iyi bireye yani çözüme ulaşılır. Çaprazlama sonucunda iyi olmayan bireyler de türeyebilir ancak onlar uygunluk değerleri düşük olacağından seleksiyon gereği elenecektir.

Programlama sürecinde farklı çaprazlama operatörleri kullanılabilir. Bu operatörlerden hangisinin seçileceğine ise problemin çözümü öncesi yapılacak değerlendirmelere bakılarak karar verilir.

Çaprazlama operatörleri:

- Tek Nokta Çaprazlama: Kullanılabilecek en basit çaprazlamadır. Kromozomlar üzerinde tek bir nokta seçilir ve o nokta referans alınarak ebeveyn kromozomlar çaprazlanır. Şayet popülasyonda yeterince birey yoksa, en iyi çözüme ulaşmak güçleşir [9].
- İki Nokta Çaprazlama: Tek noktadan farkı iki referans nokta seçilmesidir [9].
- Çok Nokta Çaprazlama: İki noktanın gelişmiş halidir. Bir çok referans nokta seçilir ve bu referanslara göre ebeveynlerin kromozomları çaprazlanır [9].
- Tek Biçimli Çaprazlama: Kromozom üzerinde referans noktalar kullanmak yerine, çaprazlama maskesi kullanılır. Çaprazlama maskesinin uzunluğu her bir kromozomun uzunluğu kadardır. Maskedeki 1 ve 0' lar türetilen kromozomun hangi geni hangi ebeveynden alacağını belirler [9] .

İki kromozomun farklı şekillerde çaprazlanması Çizelge 4.1'de verilmiştir.

Çizelge 4.1 İki kromozomun farklı şekillerde çaprazlanması

	Tek Nokta	İki Nokta	Çok Nokta	Üniform
Kromozom-1	11010|0101	110|100|101	11|0|1001|01	110100101
Kromozom-2	01101|0011	011|010|011	01|1|0100|11	011010011
Kromozom Maskesi	..- -			101010101
Yeni Kromozom	110100011	110010101	111100111	110000111

Tüm bu çaprazlama yöntemlerinden farklı olarak, M. Ericsson ve arkadaşlarının GA ile OSPF ağırlık değeri bulmak için yaptıkları çalışmada kullandıkları rasgele

anahtarlar (random keys) yöntemi ile, ebeveynler arasındaki kromozom alışverişi olasılık matrislerine dayandırılmıştır [7].

Rasgele anahtarlar yönteminde, yavru kromozomu oluşturacak her iki ata bireyin de seçilme olasılığını belirleyen tek bir matris, 0 ile 1 arasında değerler alacak şekilde oluşturulmuştur. Ayrıca, 0 ile 1 arasında bir kesim değeri kullanarak, herhangi bir genin yavruya aktarımında hangi ata bireyin hangi seçilme olasılığına sahip olacağı belirlenmiştir. Rasgele anahtar matrisi çaprazlama yönteminin sözde kodu EK-8'de gösterilmiştir.

Tüm çaprazlama yöntemlerinin temel amacı, bir öncekinden farklı nesiller üreterek çeşitliliği arttırmak ve böylece daha geniş bir çözüm uzayında çalışarak arzu edilen sonuca ulaşma olasılığını güçlendirmektir.

Çaprazlamanın yanı sıra, ele alınan problemin çözüm uzayını genişletebilmek için mutasyon yöntemi de bir seçenek olarak düşünülebilir. Mutasyon, bir kromozomun bir veya birden fazla geni üzerinde rasgele değişiklik yapmak üzere kullanılan bir olasılık fonksiyonudur.

Ele alınan problemin karakteristiğine ve seçilen mutasyon olasılığı değerine bağlı olarak, mutasyonun problemin çözüm uzayı üzerinde olumlu veya olumsuz etkileri olabilir. Bu nedenle, GA sürecinde genel olarak mutasyon olasılık değeri düşük tutulur.

M. Ericsson ve arkadaşlarının yaptıkları çalışmada mutasyon olasılığı da rasgele anahtarlar (random keys) yöntemi ile modellenmiştir [7] .

EK-9'daki PHP kodu ile biri elit (elite) diğeri de elit-olmayan (non-elite) diye sınıflandırılan iki ata bireyin çaprazlanarak yavru bireyin elde edilmesi sağlanmaktadır. Hem çaprazlama hem de mutasyon için rasgele anahtarlar yöntemi ile oluşturulan olasılık matrisleri kullanılmıştır.

4.2.4. Seçim

GA ile amacımıza hitap eden çözümlere ulaşılabilmesi, bir önceki nesilden sonrakilere aktarılacak bireylerin uygun kriterlere göre belirlenmesiyle mümkündür. Böylece uygunluk kriterlerimize uyan çözümleri saklama, uymayanları ise eleme imkanına kavuşuruz.

Seçim kriterinin gücü, uygunluk kriterlerinin iyi saptanmasında yatmaktadır. Bu nedenle bireylerden beklenen uygunluğun ne olduğu iyi tanımlanmalı ve matematiksel olarak formüle edilmelidir. Tanımlanacak bu fonksiyon problemin uygunluk fonksiyonu olarak adlandırılır [10, 14].

GA'nın her çevriminde elde edilen kromozomların uygunluk değerleri, uygunluk fonksiyonuna tabi tutularak hesaplanır. Daha sonra yüksek uyuma sahip olanlar seçilir.

Uygunluk fonksiyonu, çözümlerinin uygunluk değeri en yüksek olandan düşük olana doğru sıralanmasını sağlar ancak sıralanan çözümler arasında çeşitliliği de koruyacak şekilde bir seçim yapılması gerekir. Bu yöntemler arasında en çok bilinenleri Rulet Çemberi, Sıralı Seçim ve Elitist seçimdir .

Rulet Çemberi

Bu seçim yöntemine, orantılı seçim mekanizması da denilebilir. Rulet çemberi yönteminde her bireyin seçilme şansı, uyumluluk değeriyle orantılıdır. Uyumluluk değeri yüksel olan bireyin seçilerek yeni popülasyona eklenme olasılığı yüksektir. Ancak uyumluluk değeri düşük olanların da seçilme olasılığı vardır [9].
Örneğin elimizde 4 kromozom olduğunu düşünelim. Bu kromozomların uyumluluk değerleri aşağıdaki gibi olsun.

Kromozom 1 : 45
Kromozom 2 : 30

Kromozom 3 : 15
Kromozom 4 : 10

Tüm bu kromozomların uyumlulukları toplamı 100 olup kromozomların seçilme olasılıkları Çizelge 4.2'de verildiği gibidir.

Çizelge 4.2 Kromozomların seçilme olasılığı

Kromozom	Seçilme Olasılığı
Kromozom 1	0.45
Kromozom 2	0.3
Kromozom 3	0.15
Kromozom 4	0.1

Programa, 0 ile toplam değer olan 100 arasında rasgele bir sayı ürettirildiğinde; eğer sayı, 0 ile 45 arasında ise Kromozom 1; 45 ile 75 arasında ise Kromozom 2 ; 75 ile 90 arasında ise Kromozom 3 ; 90 ile 100 arasında ise Kromozom 4 seçilsin. Böylece her kromozom, uyumluluk değeri ölçüsünde seçilme şansı kazanmış olacaktır.

Şayet kromozomların uyumluluk değeri arasında çok ciddi farklılıklar varsa, sonraki nesillerde çeşitlilik azalacaktır. Bu durum Rulet Çemberinin dezavantajı olarak düşünülebilir.

Sıralı Seçim

Her kromozom, uyumluluk fonksiyonuyla hesaplanmış uyum değerine göre sıralanır. Daha sonra kromozomlar en kötü uyum değerine sahip olandan en iyi değere sahip olana doğru sıralanır. En uyumsuz kromozomun değeri 0, en uyumlu kromozomun değeri is N (toplam kromozom sayısı) kadardır. Böylece kromozomların seçilme olasılığı, doğrusallaştırılmış artan fonksiyon haline getirilir [9, 10, 14] .

Elitist Seçim

Bu yöntemin diğerlerinden farkı, en iyi bireyleri korumasıdır. En iyi uygunluk değerine sahip olan birey veya belirli bir yüzdeye giren en iyi bireyler korunarak yeni topluma doğrudan eklenir. Burada iyi bireylerin çaprazlama ve mutasyon sonucu kaybolmasını engellemek için böyle bir yol izlenmiştir [9]. Deney çalışmalarında Elitist Seçim yöntemi kullanılmış ve böylece çaprazlama ve mutasyon faktörlerinin en uyumlu çözümleri elemesinin önüne geçilmiştir.

EK-10'da, PHP yazım kurallarına göre hazırlanmış Elitist Seçim yöntemi gösterilmektedir. Burada yer alana *$elite_gen_number_per_population* değişkeni program kullanıcısı tarafından girilecek elit oranı olup, popülasyonun yüzde kaçının elit kabul edilip saklanacağını belirlemekte kullanılmaktadır. Deney çalışmalarında bu oran 0,1 değeri ile 0,5 değeri arasında değiştirilebilmektedir.

4.2.5. Uygunluk fonksiyonunun tanımlanması

Genetik algoritmadan başarılı bir sonuç elde edebilmek için uygunluk fonksiyonun amaca uygun olarak tanımlanması gerekmektedir. Bu çalışma kapsamında amaç, ele alınan iletişim ağlarında yük dağılımının en iyi şekilde sağlanması ve mevcut kapasitenin optimum şekilde kullanılmasıdır. Genetik algoritmadan beklenen bu amaca ulaşılabilmesi için tanımlan uygunluk fonksiyonu kullanılarak her link için en uygun ağırlık değerinin hesaplanmasıdır.

Üzerinde çalışılan iletişim ağının yük dağılımın nasıl olduğunu anlamanın en kolay ve etkin yolu tüm linkleri kapsayan bir çalışma uzayında standart sapma değerlerinin hesaplanmasıdır.

Ağ üzerindeki, toplam link sayısı N, herhangi bir linkin trafik değeri x_i ve tüm linklerin trafik yükünün aritmetik ortalaması $\bar{x}$ olarak gösterilirse, linkler üzerindeki trafik yüküne ilişkin standart sapma değeri Eş.4.1'deki gibi ifade edilir:

$$S_N = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2} \qquad (4.1)$$

5. DENEYSEL ÇALIŞMALAR

Deneylerimizi gerçekleştirmek üzere hazırlanan program, C tabanlı WEB programlama dili olan PHP dili ile hazırlanmıştır. Programın kullanıcı birimi web tabanlı olduğundan her yerden erişilebilir bir ara yüze sahiptir.

Deneylerde temel olarak üç farklı parametre girilecektir. Bunlar sırasıyla, ağ topolojisi, trafik ve GA parametreleridir.

5.1. Ağ Topolojisinin Programa Parametre Olarak Girilmesi

Bir iletişim ağı genel olarak, aktif, pasif ağ cihazları ve bu cihazlar arasındaki aktarım ortamlarından oluşur. Böyle bir ağ üzerinde optimizasyon çalışması yapılabilmesi için ağın matematiksel olarak modellenebilmesi gerekmektedir. Bir iletişim ağına ait linkleri matematiksel olarak ifade edebilmek için aşağıdaki işlemler yapılmalıdır.

i) Üzerinde çalışılan ağa ait tüm aktif ağ cihazları düğüm olarak ve düğümler arasındaki linkler kenar olarak sınıflandırılmalı ve teker teker numaralandırılmalıdır. Böylece ağın graf olarak resmedilebilmesi olanaklı hala gelecektir .
ii) N düğüm ve W düğümler arasındaki kenar olmak üzere her bir link G(Ni, Nj, Wij) biçimde satır satır listelenmelidir.

Bundan sonra G(Ni, Nj, Wij) ifadesi graf elementi olarak adlandırılacaktır.

Graf elementlerini deney programına girerken kullanılacak sentaks ve dikkate alınacak kurallar aşağıda tanımlandığı gibi olmalıdır.

1. İlk element 0 ile başlamalıdır.
2. Girilen element sayısı en az ağdaki toplam düğüm sayısı kadar olabilir. Graf elementler sisteme girilmeden önce toplam düğüm sayısı sisteme girilmelidir.

3. Daha önce girilen graf elementinin aynısının tekrar girilmesi herhangi bir hataya neden olmaz. Ancak aynı kaynak ve hedef düğüm için farklı ağırlık değeri girilmesi halinde en son girilen graf elementin ağırlık değeri geçerli olacaktır. Örneğin, önce G(0,1,5) girildiğini, sonraki herhangi bir satıra ise G(0,1,7) girildiğini varsayalım. Bu durumda 0. düğüm ile 1. düğüm arasındaki ağırlık değeri 7 olacak ve 5 geçersiz hale gelecektir.
4. Linkler yönlü olduğundan, geliş ve gidiş yönündeki linklerin ağırlık değerlerinin de aynı olma zorunluluğu yoktur. Dolayısıyla, Ni -Nj arasındaki link ile Nj-Ni arasındaki link aynı değildir. Ancak aksi belirtilmedikçe, program Nj -Ni için atanan ağırlık değerinin Ni -Nj ile aynı olduğunu kabul edecektir . Örneğin, 0. düğüm ile 1. düğüm arasındaki ağırlık değeri 5 olsun , şayet programa 1. düğüm ile 0. düğüm arasındaki ağırlık değeri girilmemişse, program bu değeri 5 olarak kabul eder. Şayet 1. düğüm ile 0. düğüm arasındaki ağırlık değeri 7 olarak girilirse, ağırlık değeri 5 yerine 7 olur.
5. Her bir graf elementin arasında noktalı-virgül (**;**) kullanılmalıdır. Sadece en son elementin ardından noktalı-virgül (**;**) kullanılmamalıdır.

Sentaks'ın yanlış girilmesi durumunda program çalışmayabilir ya da öngörülemeyen hatalar üretebilir. Program virgül veya noktalı-virgüller arasında kullanacağınız boşluklara karşı toleranslı ve korumalıdır.

Programımıza, graf elementleri girmekte kullanılan Web tabanlı kullanıcı ara birimin görünüşü EK-11 de verilmiştir.

Deneyde kullanılabilecek maksimum düğüm sayısı 50 ile sınırlandırılmıştır. N adet düğüm kullanılarak gerçekleştirilen bir denemede, program NxN'lik bir komşu matrisi belleğe yazar. Dolayısıyla N sayısı ne kadar yüksek ise bellekte kaplanan yer de o kadar yüksek olacaktır.

5.2. Trafik Matrisinin Programa Parametre Olarak Girilmesi

İletişim ağındaki düğümler arasındaki trafik akışını gösteren trafik matrisi, gerçek uygulamalarda örnekleme teknikleri kullanarak yapılan ölçümlerle veya doğrudan hesaplama yoluyla (Bkz. Eş. 3.4) elde edilir. Trafik matrisi, bir önceki ölçümlere dayanarak bulunduğundan değişkendir. Bu nedenle hesaplanan trafik matrisi tahmini trafik matrisi olarak isimlendirilir.

Deney çalışmasında, N düğüm ve T düğümler arasındaki kenar olmak üzere, herhangi bir düğüm çiftinin yönlü trafiği TM (Ni, Nj, Tij) biçiminde satır satır listelenecektir.

Bundan sonra TM(Ni, Nj, Tij) ifadesi trafik elementi olarak adlandırılacaktır.

Trafik elementi tıpkı graf elementi gibi yönlüdür. Bu nedenle TM(Ni, Nj, Tij) ile TM(Nj, Ni, Tji) farklı değerleri ifade eder.

Trafik elementi graf elementinden temel farkı, graf elementi aralarında doğrudan link olan düğümler arasında tanımlanırken trafik elementinin herhangi iki düğüm arasında tanımlanabilir olmasıdır. Ayrıca, programın linklerdeki yük değerlerinin hesaplanabilmesi için en az bir tane trafik elementinin girilmesi yeterli olacaktır.

Trafik elementinin programa parametre olarak girilirken kullanılacak sentaks graf elementinin girişinde kullanılan sentaks ile aynıdır.

Programa, trafik elementlerini girmekte kullanılan web tabanlı kullanıcı ara biriminin görünüşü EK-12' de verilmiştir .

5.3. GA Parametreleri

Çalışmalarda 5 farklı GA parametresi kullanılmıştır. Bu değerler belirli aralıklarda değiştirilerek faklı sonuçlar elde edilmiş ve sonuca olan etkileri incelenmiştir.

5.3.1. Maksimum nesil

Programın kaç çevrim yapacağını belirleyen parametredir. Bu nedenle sonlandırma parametresi olarak da isimlendirilir. Bu parametre, üzerinde çalışılacak ağın sahip olduğu düğüm sayısına göre belirlenir. Başka bir ifadeyle, komşuluk matrisi ne kadar büyük ise çevrim sayısının da büyütülmesi gerekmektedir. Örneğin M.Ericsson ve arkadaşlarının yaptığı çalışmada [13], 100 düğümlük (100x100'lük komşuluk matrisi olan) AT&T ağı için 500 çevrim yapılmıştır. Yine aynı çalışmada, GA'nın iyi bir sonuç vermesi için 100 civarı çevrim yapılması gerektiği belirtilmiştir.

Programının kullanıcı arabiriminde, çevrim sayısı 10, 25, 50, 100, 200, 300, 400 ve 500 olmak üzere 8 seçenekli olarak tasarlanmıştır.

5.3.2. Popülasyonda kullanılacak gen sayısı

Her bir kromozom yerel bir çözüm uzayını ifade eder. Örneğin bünyesinde 5 gen bulunduran bir kromozom 5 farklı çözüm taşır.

Çalışmamızda, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100 olmak üzere 15 farklı gen sayısı önerilmiştir. Kromozomların gen sayısının yüksek tutulması, yeni nesil kromozomların çeşitliliği üzerinde doğrudan etkilidir.

5.3.3. Çaprazlama kesim parametresi

Rasgele anahtarlar yöntemiyle elde edilen ve olasılık fonksiyonu olarak kullanılan matrisin aldığı değerlerle karşılaştırılan olasılık değeridir. Kesim değeri, ata bireylerin yavru bireyin oluşturulma sürecindeki etkilerini belirler. Deney çalışmalarında bu değer 0,4 ile 0,8 arasında değiştirilmiştir. M. Ericsson ve arkadaşlarının yaptığı çalışmada [13], iyi bir yakınsama ile kromozom çeşitliliği dengesi gözetildiğinde en uygun kesim değerinin 0,7 olduğu belirtilmiştir.

5.3.4. Popülasyondaki elit oranı

Her neslin bireyleri, uyumluluk değeri en yüksel olandan en düşük olana doğru sıralandıktan sonra en tepedeki kaç bireyin saklanarak sonraki nesle aktarılacağını belirleyen parametredir. Elit oranının yüksek tutulması çeşitliliği artırırken iyi sonuca varış zamanını uzatıcı etki yaratabilir.

Programın kullanıcı arabiriminde, elit oranı 0,1'den 0,5'e kadar değiştirilebilmektedir.

5.3.5. Mutasyon değeri

Çaprazlama sırasında mutasyon yöntemi ile herhangi bir genin rasgele değiştirilmesi olasılığını belirler. Mutasyon etkisi, çeşitliliği arttırıcı bir rol oynar. Ancak rasgele bir değişiklik olduğundan makul ölçüler içerisinde sınırlandırılmalıdır.

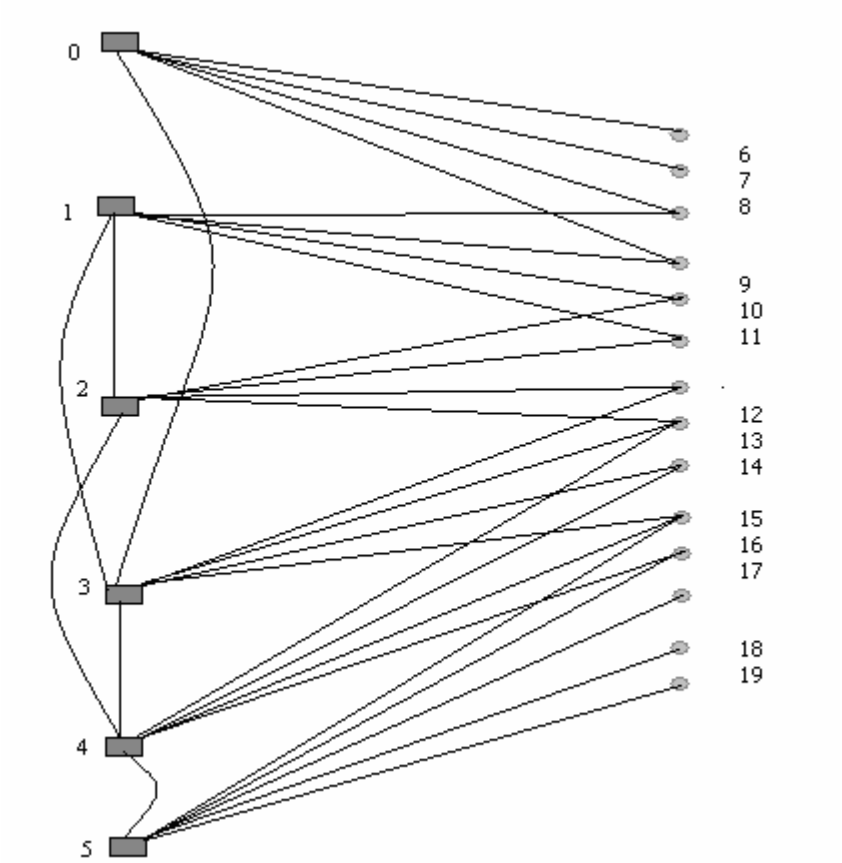
Programın kullanıcı arabiriminde, mutasyon olasılığı 0,01'den 0,5'e kadar değiştirilebilmektedir. Mutasyon değeri, diğer parametrelerle arzu edilen çeşitliliğin sağlanamaması durumda etkin bir parametre olarak kullanılabilir. Çeşitliliğin yeterince iyi olduğu durumlarda ise oldukça düşük bir değer verilerek etkinliği sınırlanabilir.

5.4. Deney 1

Deney 1 boyunca yapılan denemelerde Şekil 5.1'de verilen ağ topolojisi üzerinde çalışılmıştır. Şekil 5.1'de verilen ağ, toplam 20 düğüm ve 62 kenar (31 gidiş ve 31 geliş yönünde) bileşenine sahiptir.

Şekil 5.1'deki düğümlerden, 6'dan 19'a kadar olanlar trafiğin kaynaklandığı ve sonlandığı uç birimleri temsil etmektedir. Diğer yandan, 0 'dan 5'e kadarki düğümler ise taşıyıcı ve dağıtıcı merkez birimleri temsil etmektedir.

Şekil 5.1'deki topoloji hiyerarşik bir merkezi yapıya sahip değildir. Pratikte bu tür bir topoloji, son kullanıcıdan kaynaklanan trafiğin uç birimlerden yedekli olarak toplanmasında kullanılabilir.



Şekil 5.1 Deney 1 çalışmalarında kullanılan ağ topolojisi

Çizelge 5.1'de tüm deneylerin tüm denemelerinde kullanılacak ön tanımlı GA parametreleri verilmiştir.

Çizelge 5.1 Deneylerde kullanılacak ön tanımlı GA parametreleri

Maksimum Nesil Sayısı	25
Popülasyonda Kullanılacak Gen Sayısı	15
Çaprazlama Kesim Parametresi	0,7
Popülasyondaki Elit Oranı	0,3
Mutasyon Değeri	0,05

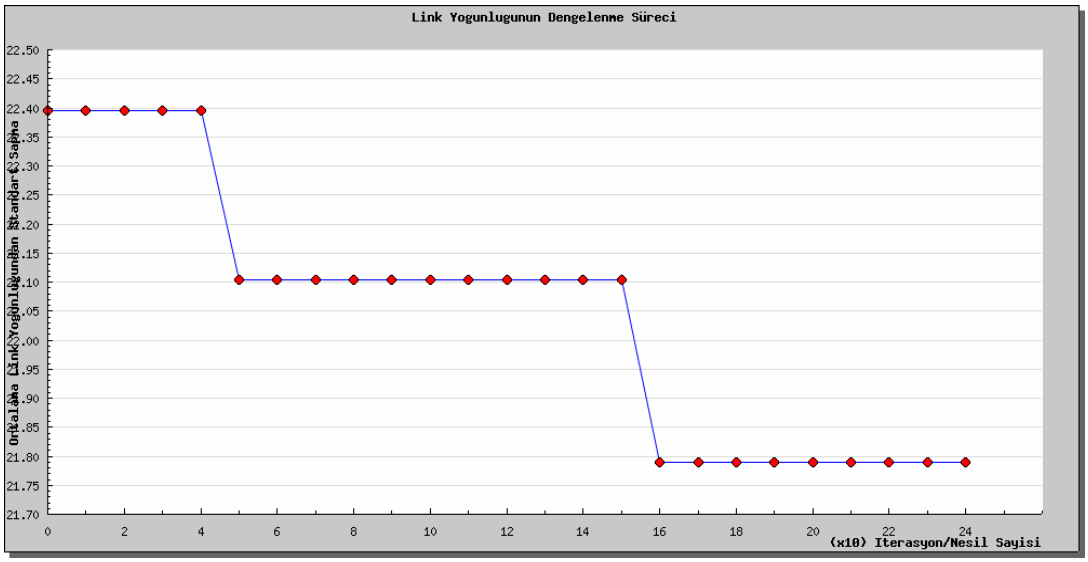
5.4.1. Deney 1 - deneme 1

Şekil 5.1'deki ağa uygulanmak üzere hazırlanan trafik matrisi Çizelge 5.2'de gösterilmiştir. Bu trafik matrisinin program sentaksına uygun formatıyla hazırlanmış hali EK-13'de verilmiştir.

Çizelge 5.2 Deney 1'de kullanılan 1. trafik matrisi

0 →19	120
18→1	132
2→17	232
4→9	67
1→4	23
5→6	41

Bu denemede, ön tanımlı GA parametreleri değiştirilmeden kullanılmıştır. Çizelge 5.2'deki trafik matrisi ve ön tanımlı GA parametreleri girildiğinde, programın hesapladığı, her bir çevrim için link kullanımına ilişkin standart sapma değerleri Şekil 5.2'de gösterildiği gibidir.



Şekil 5.2 1. Deneyin 1. denemesinin grafik sonuç değerleri

5.4.2. Deney1 - deneme 2

1. Deneyin ilk denemesinde trafik matrisi olarak kullanılan Çizelge 5.2'i ve ön tanımlı GA parametrelerini hiç değiştirmeden program yeniden koşturulmuştur. Ancak bu defa, sonuç olarak her bir nesil için uygunluk değerleri ve en son nesil için ağırlık değerleri istenmiştir. Programın döndürdüğü tablo EK-14'de verilmiştir.

5.4.3. Deney1 - deneme 3

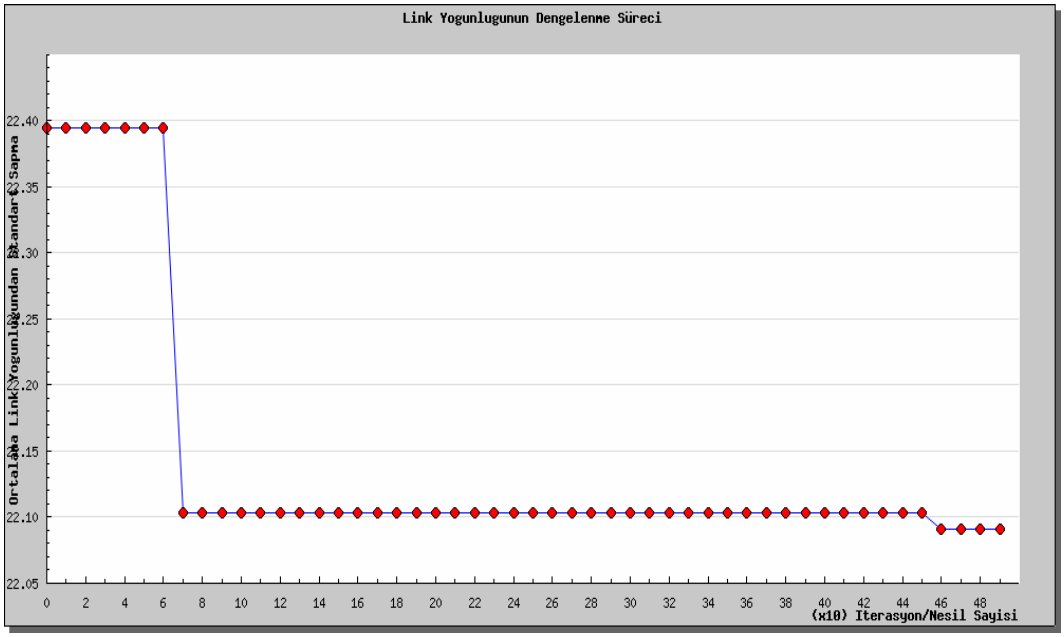
3. denemede trafik matrisi olarak Çizelge 5.2, GA parametreleri olarak da Çizelge 5.3 uygulanmıştır.

Çizelge 5.3.1. Deneyin 3. denemesinde kullanılan GA parametreleri

Maksimum Nesil Sayısı	50
Popülasyonda Kullanılacak Gen Sayısı	15
Çaprazlama Kesim Parametresi	0,7
Popülasyondaki Elit Oranı	0,3
Mutasyon Değeri	0,05

Çizelge 5.3'de görüldüğü üzere, ilk iki denemeden farklı olarak 3. denemede Maksimum Nesil Sayısı parametresi 25'ten 50'ye çıkarılmıştır.

3. denemede, her bir çevrim için link kullanımın standart sapma değerleri Şekil 5.3'de gösterildiği gibi olmuştur.



Şekil 5.3 1. Deneyin 3. denemesinin grafik sonuç değerleri

5.4.4. Deney1 - deneme 4

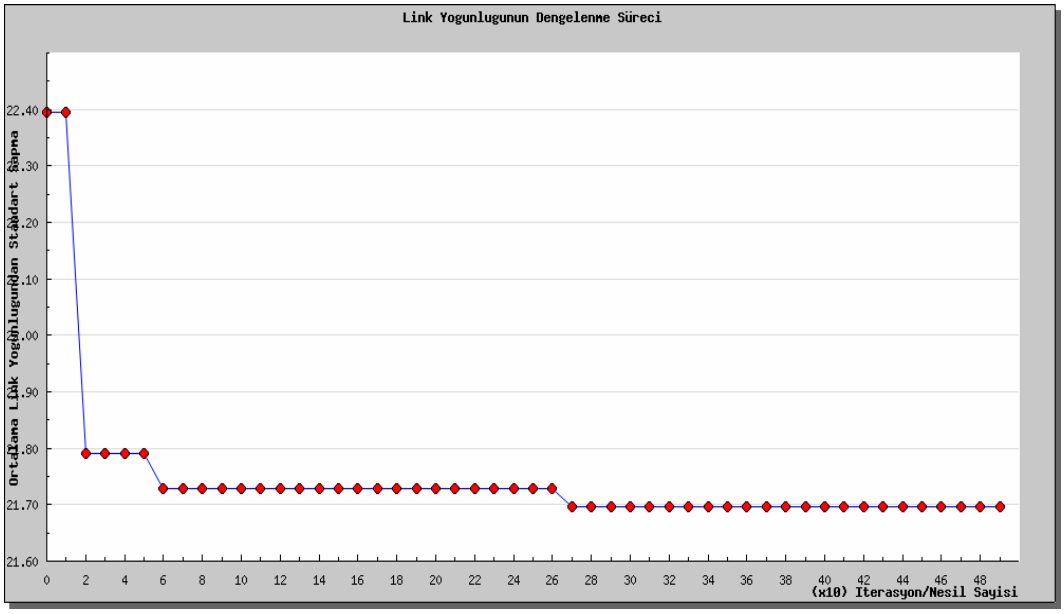
4. denemede trafik matrisi olarak Çizelge 5.2, GA parametreleri olarak da Çizelge 5.4 uygulanmıştır.

Çizelge 5.4. 1. Deneyin 4. denemesinde kullanılan GA parametreleri

Maksimum Nesil Sayısı	50
Popülasyonda Kullanılacak Gen Sayısı	10
Çaprazlama Kesim Parametresi	0,7
Popülasyondaki Elit Oranı	0,3
Mutasyon Değeri	0,05

Bu denemede ise Popülasyonda Kullanılacak Gen Sayısı parametresi 15'den 10'a çekilmiştir.

4 denemede, her bir çevrim için link kullanımın standart sapma değerleri Şekil 5.4'de gösterildiği gibi olmuştur.



Şekil 5.4 1. Deneyin 4. denemesinin grafik sonuç değerleri

5.4.5. Deney1 - deneme 5

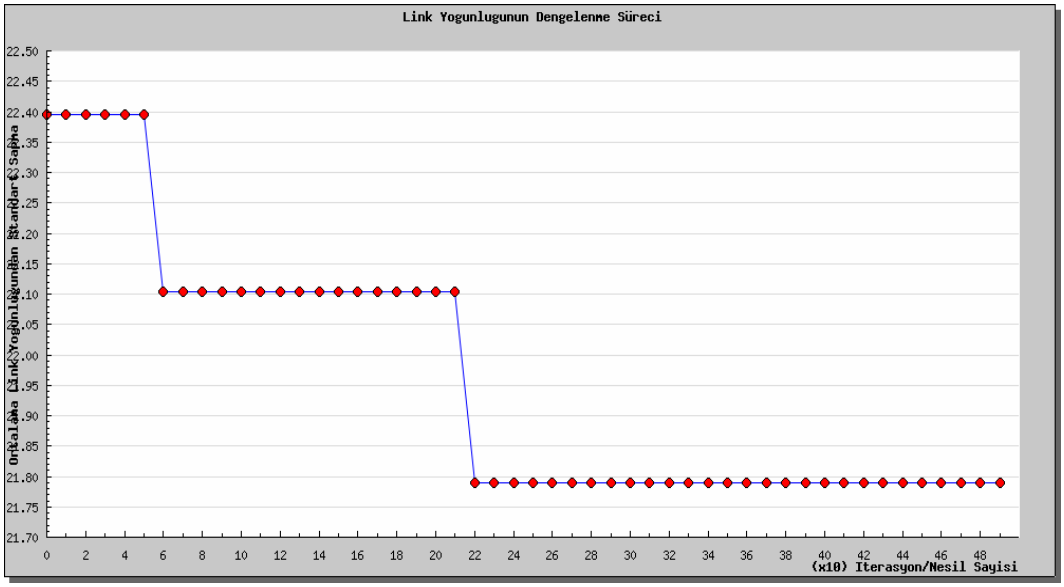
5. denemede trafik matrisi olarak Çizelge 5.2, GA parametreleri olarak da Çizelge 5.5 uygulanmıştır.

Çizelge 5.5 1. Deneyin 5. denemesinde kullanılan GA parametreleri

Maksimum Nesil Sayısı	50
Popülasyonda Kullanılacak Gen Sayısı	15
Çaprazlama Kesim Parametresi	0,7
Popülasyondaki Elit Oranı	0,3
Mutasyon Değeri	0,2

Bu denemede ise Mutasyon Değeri parametresi 0,05'den 0,2 değerine çıkarılmıştır.

5. denemede, her bir çevrim için link kullanımın standart sapma değerleri Şekil 5.5'de gösterildiği gibi olmuştur.



Şekil 5.5 1. Deneyin 5. denemesinin grafik sonuç değerleri

5.4.6. Deney1 - deneme 6

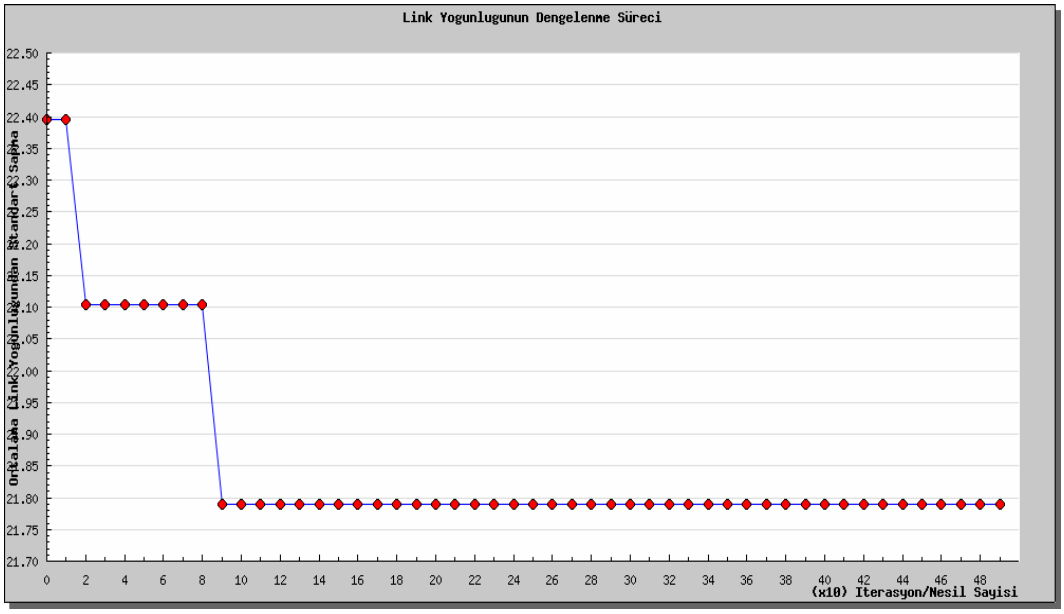
6. denemede trafik matrisi olarak Çizelge 5.2, GA parametreleri olarak da Çizelge 5.6 uygulanmıştır.

Çizelge 5.6 1. Deneyin 6. denemesinde kullanılan GA parametreleri

Maksimum Nesil Sayısı	50
Popülasyonda Kullanılacak Gen Sayısı	15
Çaprazlama Kesim Parametresi	0,4
Popülasyondaki Elit Oranı	0,3
Mutasyon Değeri	0,2

Bu denemede ise Çaprazlama Kesim parametresinin değeri 0,7 den 0,4 değerine çekilmiştir.

6 denemede, her bir çevrim için link kullanımının standart sapma değerleri Şekil 5.6'da gösterildiği gibi olmuştur.



Şekil 5.6 1. Deneyin 6. denemesinin grafik sonuç değerleri

5.4.7. Deney1 - deneme 7

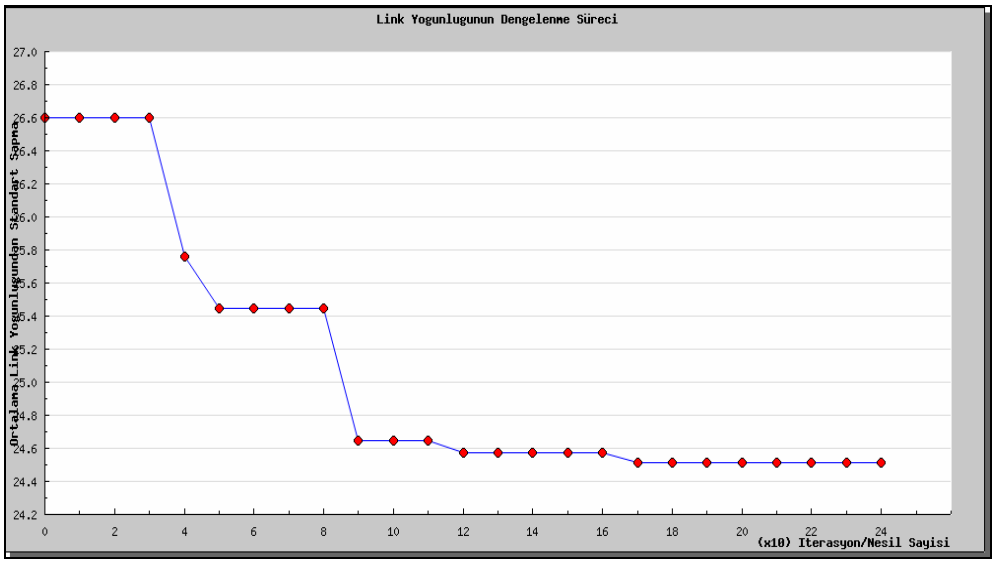
Bu denemede, Şekil 5.1'deki ağa uygulanmak üzere trafik matrisi zenginleştirilerek, Çizelge 5.7'deki hale getirilmiştir. Zenginleştirme matristeki eleman sayısının artması anlamına gelir. Çizelge 5.7'deki trafik matrisinin program sentaksına uygun formatıyla hazırlanmış hali EK-15'de gösterilmiştir.

Çizelge 5.7. Deney 1'de kullanılan 2. trafik matrisi

6→19	113
19→6	43
7→18	87
18→7	17
8→11	96
10→19	65
12→18	91
16→7	135
13→14	45
14→13	171

Bu denemede, ön tanımlı GA parametreleri hiç değiştirilmeden kullanılmıştır.

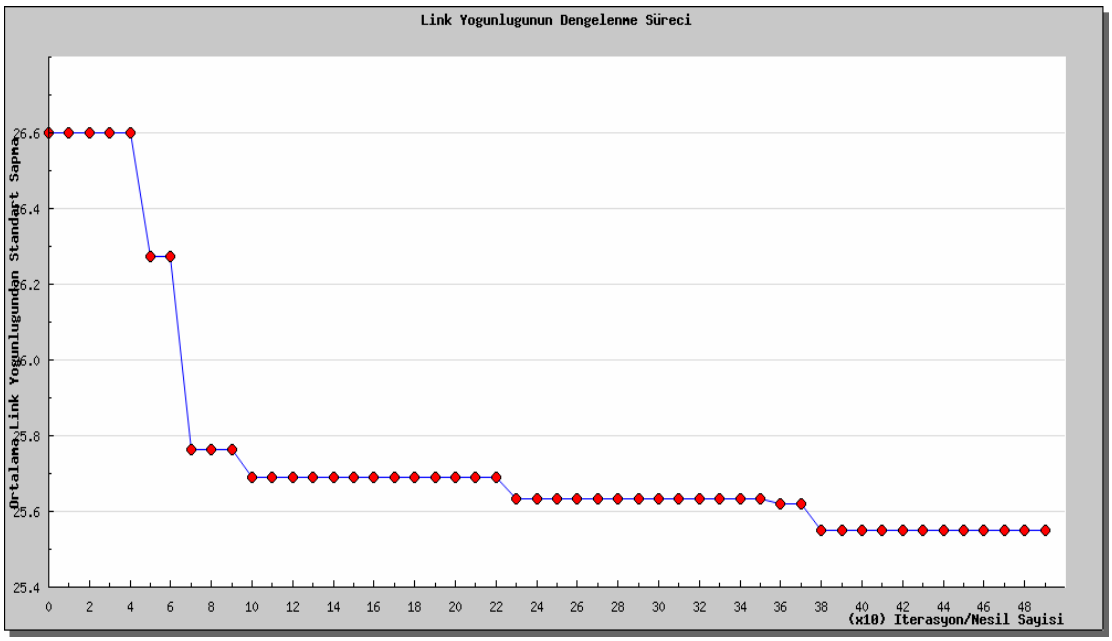
Çizelge 5.7'deki trafik matrisi ve ön tanımlı GA parametreleri girildiğinde, programın, her bir çevrim için link kullanımına ilişkin hesapladığı standart sapma değerleri Şekil 5.7'de verildiği gibi şekillenmiştir.



Şekil 5.7 1. Deneyin 7. denemesinin grafik sonuç değerleri

5.4.8. Deney1 - deneme 8

8. denemede, Çizelge 5.7'de verilen zengin trafik matrisi kullanılmıştır. Bu denemede, ön tanımlı GA parametrelerinden sadece Maksimum Nesil Sayısı değiştirilerek 50'ye çıkarılmıştır. Deneme sonucunda, her çevrim için link kullanımın standart sapma değerleri Şekil 5.8'deki gibi oluşmuştur.



Şekil 5.8 1. Deneyin 8. denemesinin grafik sonuç değerleri

5.4.9. Diğer denemeler

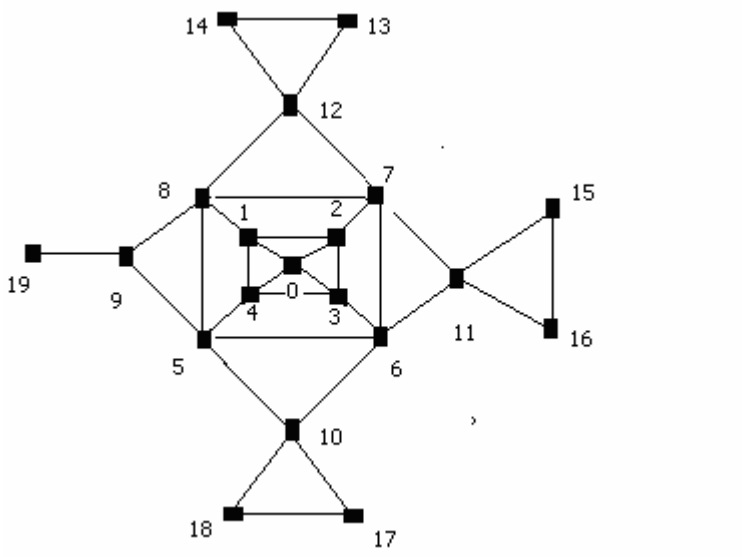
Şekil 5.1'de verilen iletişim ağı için, Çizelge 5.7'de verilen zengin trafik matrisi kullanılarak son dört deneme daha yapılmıştır. Bu denemelerde, ön tanımlı GA parametrelerinden bir ya da iki tanesinin değiştirilmesiyle link kullanım değerlerinin nasıl değiştiği gözlenmeye çalışılmıştır. Yapılan son dört denemede hangi G.A parametrelerinin değiştirildiği ve nasıl bir sonuç elde edildiği EK-16'da verilmiştir.

5.5. Deney 2

Deney 2 boyunca yapılan denemelerde Şekil 5.9'da verilen ağ topolojisi üzerinde çalışılmıştır. Şekil 5.9'da verilen ağ, toplam 20 düğüm ve 68 kenar (34 gidiş ve 34 geliş yönünde) bileşenine sahiptir.

Şekil 5.9'daki düğümlerden, 13'den 19'a kadar olanlar trafiğin kaynaklandığı ve sonlandığı uç birimleri temsil etmektedir. Diğer yandan, 0 ile 13 arasındaki düğümler ise taşıyıcı ve dağıtıcı merkez birimleri temsil etmektedir.

Şekil 5.9'daki iletişim ağı, hiyerarşik bir merkezi yapıya sahiptir. Pratikte bu tür bir topoloji, uç birimlerden toplanan trafiğin sağlam yedekliliğe sahip bir omurga üzerinden taşınmasını sağlamak için kullanılabilir.



Şekil 5.9 Deney 2 çalışmalarında kullanılan ağ topolojisi

Deney 1'de olduğu gibi, Deney 2'nin tüm denemelerinde ön tanımlı GA parametresi olarak Çizelge 5.1'deki değerler kullanılmıştır.

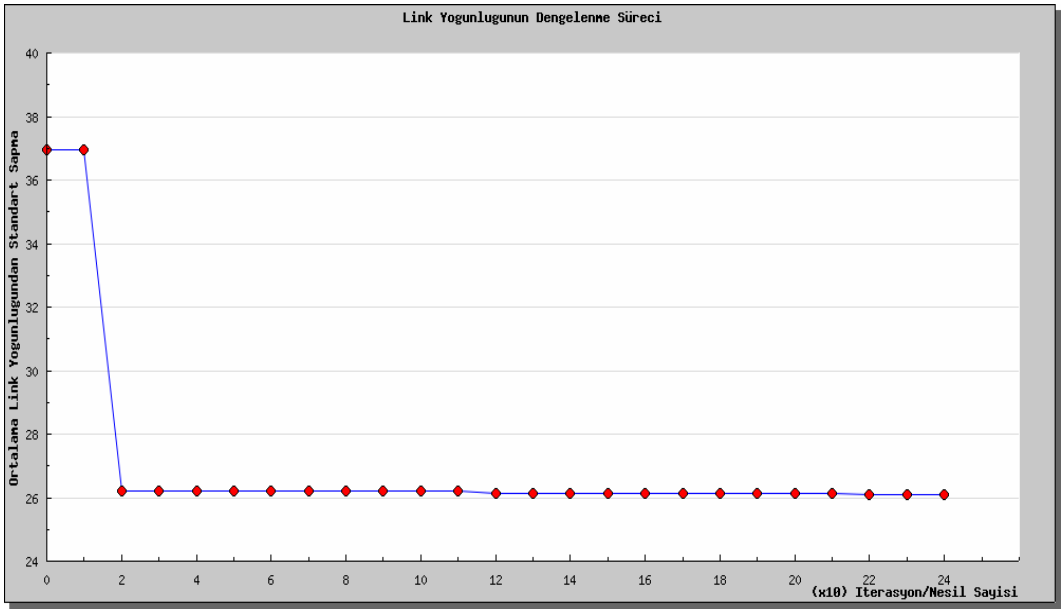
5.5.1. Deney 2 - deneme 1

Şekil 5.9'deki ağa uygulanmak üzere hazırlanan trafik matrisi Çizelge 5.8'de gösterilmiştir.

Çizelge 5.8 Deney 2'de kullanılan 1. trafik matrisi

14 →17	120
15→19	132
19→16	232
18→15	67
13→16	23

Bu denemede, ön tanımlı GA parametreleri değiştirilmeden kullanılmıştır. Çizelge 5.8'deki trafik matrisi ve Çizelge 5.1'deki GA parametreleri girildiğinde, programın her bir çevrim için hesapladığı link kullanımın değerlerine ilişkin standart sapma değerleri Şekil 5.10'de verildiği gibidir.



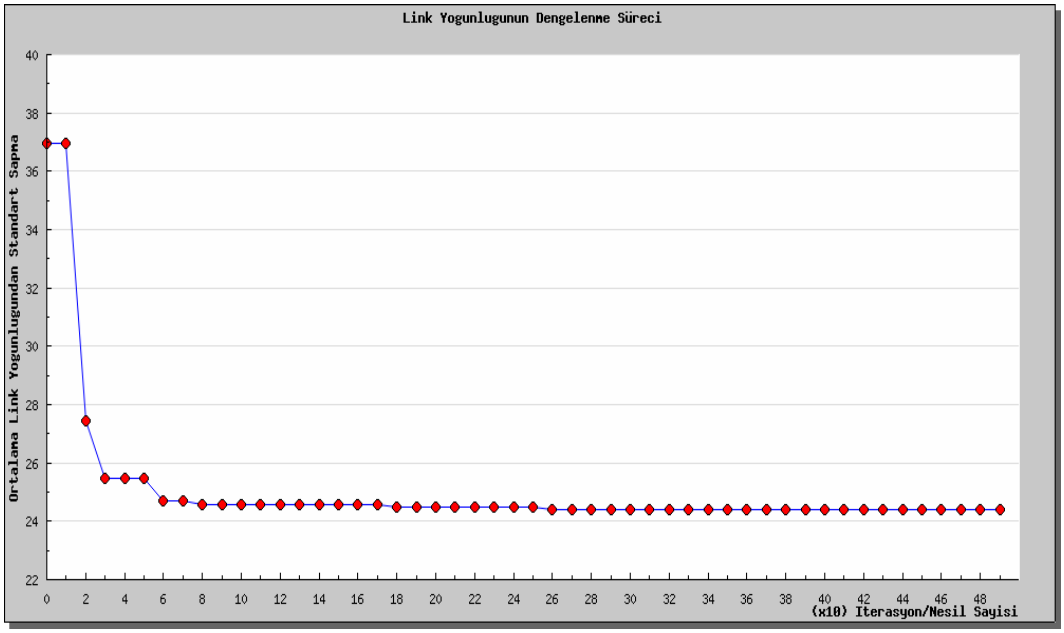
Şekil 5.10 2. Deneyin 1. denemesinin grafik sonuç değerleri

5.5.2. Deney 2 - deneme 2

2. Deneyin ilk denemesinde trafik matrisi olarak kullanılan Çizelge 5.8'i ve ön tanımlı GA parametreleri hiç değiştirilmeden program yeniden koşturulmuştur. Program çıktısı olarak her bir nesil için uygunluk değerleri ve en son nesil için ağırlık değerleri istenmiştir. Programın döndürdüğü tablo EK-17'de verilmiştir.

5.5.3. Deney 2 – deneme 3

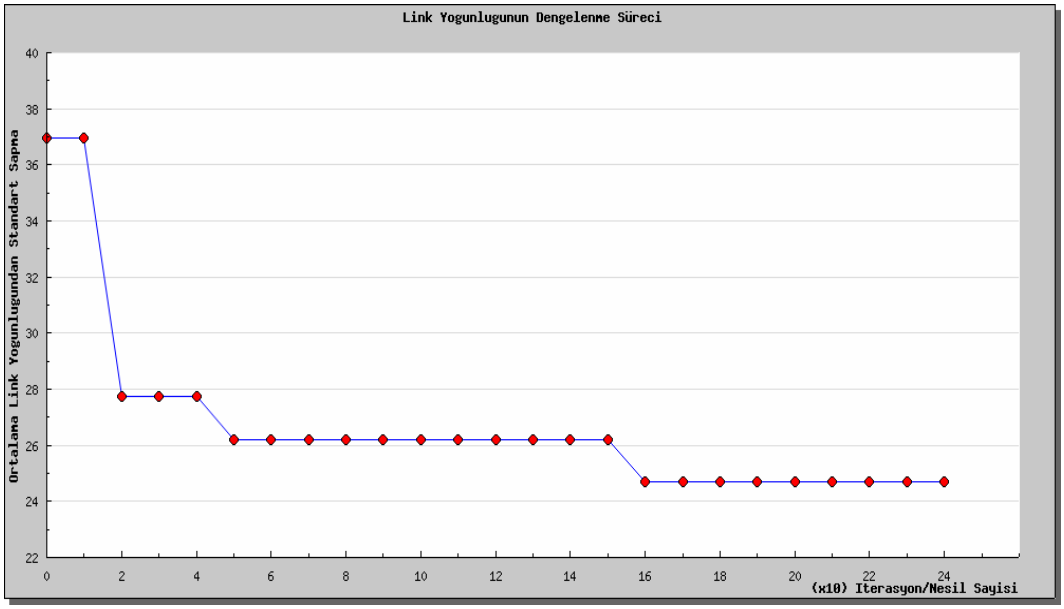
3. denemede, Çizelge 5.8'deki trafik matrisi değiştirilmeden kullanılmıştır. Diğer yandan, ön tanımlı GA parametrelerinden sadece Maksimum Nesil Sayısı değiştirilerek 50'ye çıkarılmıştır. Deneme sonucunda, her çevrim için link kullanımın standart sapma değerleri Şekil 5.11'deki gibi oluşmuştur.



Şekil 5.11 2. Deneyin 3. denemesinin grafik sonuç değerleri

5.5.4. Deney 2 - deneme 4

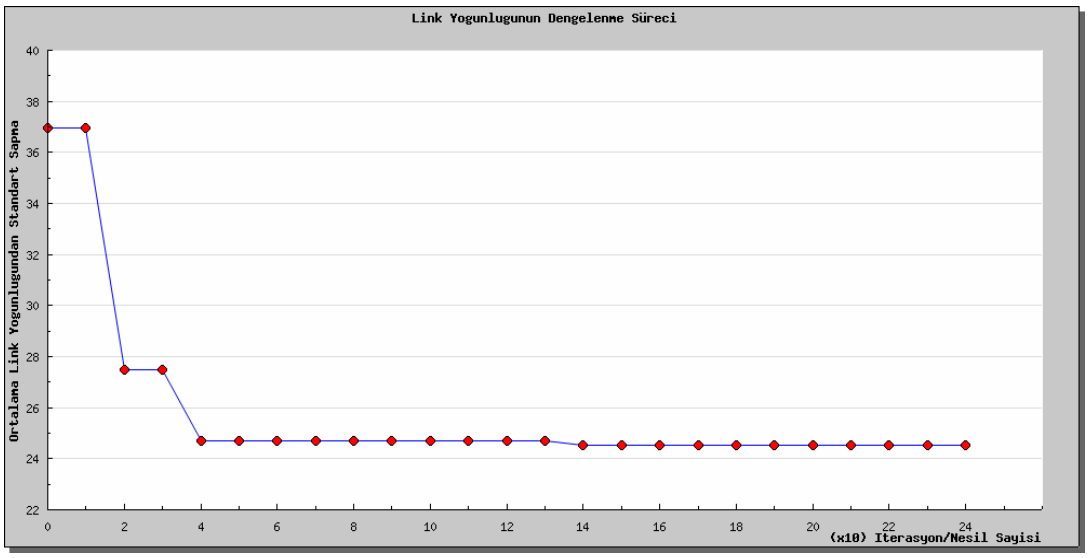
4. denemede, Çizelge 5.8'deki trafik matrisi değiştirilmeden kullanılmıştır. Çizelge 1.1'deki ön tanımlı GA parametrelerinden sadece Gen Sayısı parametresi 15'den 10'a çekilmiştir. Deneme sonucunda, her çevrim için link kullanımın standart sapma değerleri Şekil 5.12'deki gibi oluşmuştur.



Şekil 5.12 2. Deneyin 4. denemesinin grafik sonuç değerleri

5.5.5. Deney 2 - deneme 5

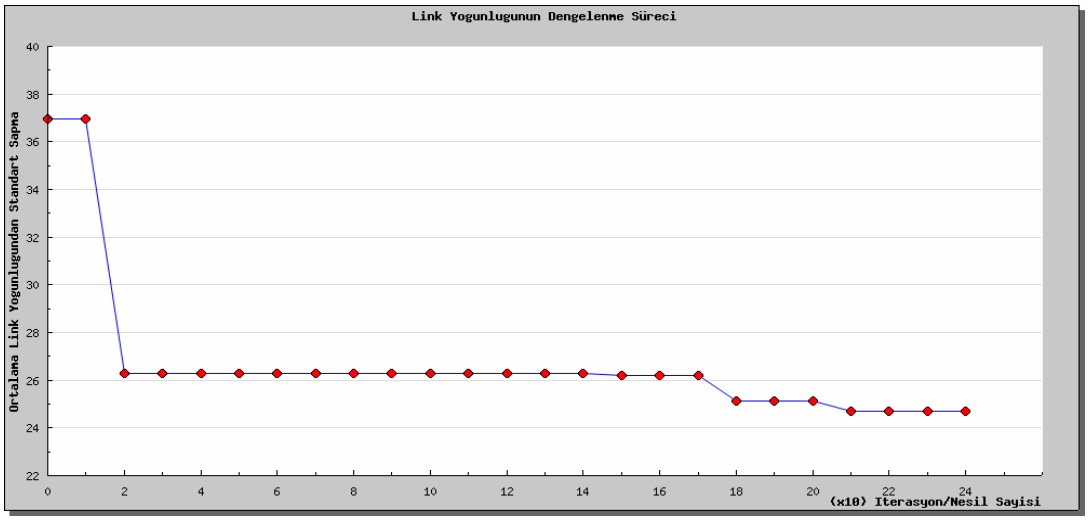
5. denemede, Çizelge 5.8'deki trafik matrisi değiştirilmeden kullanılmıştır. Ön tanımlı GA parametrelerinden ise sadece Mutasyon Değeri parametresi 0,05'den 0,2'ye çıkarılmıştır. Deneme sonucunda, her çevrim için link kullanımına ait standart sapma değerleri, Şekil 5.13'deki gibi oluşmuştur.



Şekil 5.13 2. Deneyin 5. denemesinin grafik sonuç değerleri

5.5.6. Deney 2 - deneme 6

Ön tanımlı GA parametrelerinden sadece Çaprazlama Kesim parametresi 0,4'den 0,7'ye çekilerek 6. deneme tamamlanmıştır. Deneme sonucunda, her çevrim için link kullanımına ait standart sapma değerleri, Şekil 5.14'deki gibi oluşmuştur.



Şekil 5.14 2. Deneyin 6. denemesinin grafik sonuç değerleri

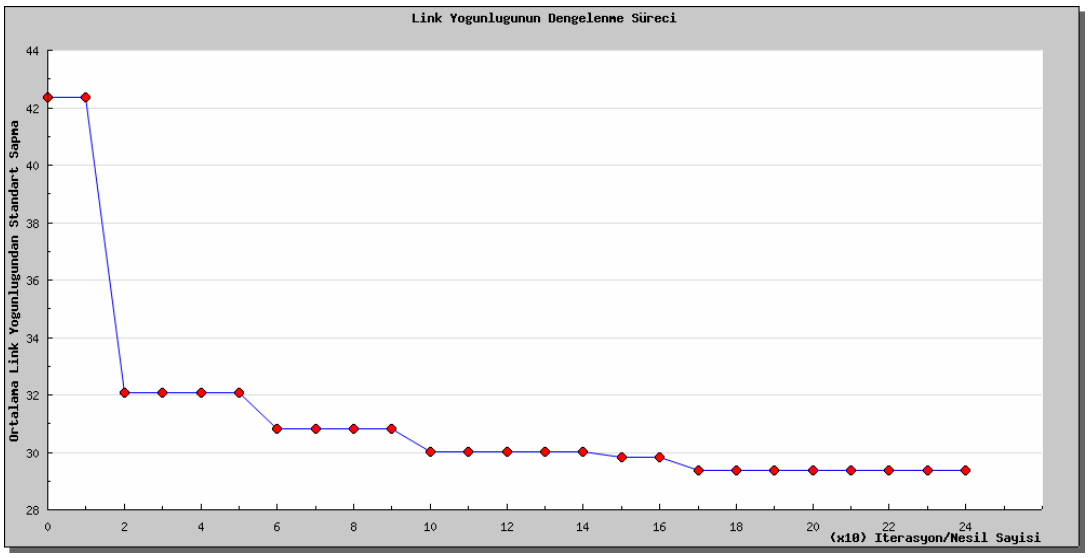
5.5.7. Deney 2- Deneme 7

Bu denemede, Şekil 5.9'daki ağa uygulanmak üzere trafik matrisiz zenginleştirilerek Çizelge 5.9'daki hale getirilmiştir.

Çizelge 5.9 Deney 2'de kullanılan 2. trafik matrisi

14→17	120
15→19	45
19→16	145
18→15	33
17→19	86
18→14	169
13→16	138

Diğer yandan ön tanımlı GA parametreleri değiştirilmeden aynen kullanılmıştır. Çizelge 5.9'deki trafik matrisi ve ön tanımlı GA parametreleri girildiğinde, programın her bir çevrim sonunda link kullanımına ilişkin hesapladığı standart sapma değerleri Şekil 5.15'de verilmiştir.



Şekil 5.15 2. Deneyin 7. denemesinin grafik sonuç değerleri

5.5.8. Diğer denemeler

Şekil 5.9'da verilen iletişim ağı için, Çizelge 5.7'de verilen zengin trafik matrisi kullanılarak son 5 deneme daha yapılmıştır. Bu denemelerde, ön tanımlı GA parametrelerinden bir ya da iki tanesinin değiştirilmesinden link kullanım değerlerinin nasıl değiştiği gözlenmeye çalışılmıştır. Yapılan son 5 denemede hangi G.A parametrelerinin değiştirildiği ve nasıl bir sonuç elde edildiği EK-18'de verilmiştir.

5.6. Deneysel Çalışmalardan Elde Edilen Sonuçlar

Birinci deney çerçevesinde, trafik matrisi ve GA parametreleri değiştirilerek ağ üzerindeki link yoğunluklarının standart sapma değerlerine ilişkin 12 farklı sonuç elde edilmiştir. Bu sonuçlar incelendiğinde, ağ üzerindeki kısıtlı bir trafik akışı için,

iletim ortamının dengeli kullanımına ilişkin iyileştirmenin % 2,6 ile sınırlı kaldığı, ancak nispeten zengin bir trafik matrisi için iyileştirme değerinin %10'a ulaştığı görülmüştür. Diğer yandan belirli aralıklarda değiştirilen GA parametrelerinin iyileştirme değerleri üzerinde kısıtlı etkiler yarattığı görülmüştür.

İkinci deney çerçevesinde, trafik matrisi ve GA parametreleri değiştirilerek ağ üzerindeki link yoğunluklarının standart sapma değerlerine ilişkin 12 farklı sonuç elde edilmiştir. Bu sonuçlar incelendiğinde, ilk deneyden oldukça farklı bir tablo ortaya çıkmıştır. Bu farkın temel nedeni, merkezi hiyerarşiye ve yedekliliğe sahip ağ üzerindeki düğümler arasında tanımlanabilecek güzergah sayısının, ilk deneyde kullanılan ağınkinden çok daha zengin oluşudur.

İkinci deneyde kullanılan ilk trafik matrisi için iletim ortamının dengeli kullanıma ilişkin iyileştirme %27 civarında olmuştur. Nispeten daha zengin olan ikinci trafik matrisi kullanıldığında ise bu değer % 31,3'e kadar çıkmıştır

6. SONUÇ VE TARTIŞMALAR

Deney çalışmalarımızda, aktif-pasif ağ cihazları ve bu cihazlar arasındaki iletim ortamından oluşan ve topolojik olarak birbirinden oldukça farklı iki iletişim ağı kullanılmıştır. Bu iletişim ağlarında yer alan ağ cihazları düğüm olarak, iletim ortamları ise link olarak sınıflandırılmış ve graf teorisi kullanılarak matematiksel olarak modellenmesi sağlanmıştır.

İlk deneyde konu edilen iletişim ağı, hiyerarşik bir merkezi yapıya sahip değildir. Bu yapı pratikte terminal birimlerden trafik toplamak maksadıyla kullanılan bir topolojiye sahip olup, iletim ortamı kısmi yedekliliğe sahiptir. Dolayısıyla oluşturulabilecek alternatif güzergah sayısı kısıtlıdır. Alternatif yolların kısıtlılığı nedeniyle böyle bir topoloji üzerinde ciddi bir trafik akış optimizasyonunun sağlanamamış olması öngörülen bir durumdur.

İkinci deneyde konu edilen iletişim ağı hiyerarşik bir merkezi yapıya sahiptir. Pratikte, daha geniş bir coğrafyada yer alan ve kenarda yer alan ağların trafiğini taşıyan ağ omurgasını temsil eder. İlk deneyde kullanılan topolojiden farklı olarak daha sahip olduğu linkler yedekli bir yapıya sahiptir. Dolayısıyla oluşturulabilecek alternatif güzergah sayısı nispeten fazladır. Alternatif yolların bolluğu nedeniyle, böyle bir topoloji üzerinde anlamlı bir trafik akış optimizasyonunun sağlanabilmesi öngörülen bir durumdur.

İki deney karşılaştırıldığında, link trafik değerlerinin dengeli kullanımı için yapılacak iyileştirme çalışmalarında ağ topolojisinin oldukça önem taşıdığı görülmektedir. Zira, bir ağ üzerinde yapılacak trafik mühendisliği çalışmasının başarısı, ağ topolojisinin alternatif güzergah üretmeye müsait oluşuna bağlıdır.

Genetik algoritmadan başarılı bir sonuç elde edebilmek için uygunluk fonksiyonunun amaca uygun olarak tanımlanması temel kriterdir. Bu çalışma kapsamında amaç, ele alınan iletişim ağlarında yük dağılımının en iyi şekilde sağlanması ve mevcut kapasitenin optimum şekilde kullanılmasıdır. Genetik algoritmadan beklenen bu

amaca ulaşılabilmesi için tanımlanan uygunluk fonksiyonu kullanılarak her link için en uygun ağırlık değerinin hesaplanmasıdır.

Üzerinde çalışılan iletişim ağının yük dağılımının nasıl olduğunu anlamanın en kolay ve etkin yolu tüm linkleri kapsayan bir çalışma uzayında standart sapma değerlerinin hesaplanmasıdır. Bu çalışma kapsamında yer alan tüm denemelerde standart sapma değeri her çevrimde yeniden hesaplanmış ve uygun çözümler en düşük standart sapma değerine göre sıralanmıştır. En iyi çözümü sunduğu varsayılan en son çevrimde ise elde edilen ağırlık değerleri ise yük dengesini optimum hala getiren nihai ağırlık değerleri olarak tespit edilmiştir.

GA parametrelerinden maksimum nesil sayısının (iterasyon değeri) arttırılması, en iyi iyileştirme değerinin bulunması şansını artırır. Diğer yandan maksimum nesil sayısının programın koşma zamanını belirleyen temel parametre olduğu görülmüştür.

Popülasyonda kullanılan gen sayısının arttırılması, link ağırlık değeri çeşitliliğini zenginleştirdiğinden çözüm uzayını da genişletmektedir. Diğer yandan bu parametrenin aşırı arttırılmasının da programın koşma zamanını oldukça yükselttiği gözlenmiştir. Optimizasyon/zaman değeri göz önüne alındığında, bu parametre için en uygun değerin 15 olduğu gözlenmiştir.

Çaprazlama kesim değeri, yeni nesil için üretilen yavru bireylerin, ata nesillerindeki elitlerden alacağı gen sayısının olasılığını belirlemektedir. Yakınsama ve çeşitlilik kriterleri dikkate alındığında, bu değerinin 0,6 ile 0,8 arasında tutulması, deneylerin performansı açısından önem taşımaktadır.

Popülasyondaki elit oranı, popülasyonun yüzde kaçlık diliminin elit olarak kabul edileceğini belirleyen göreceli bir parametredir. Tıpkı çaprazlama kesim değeri gibi, yakınsama ve çeşitlilik kriterleri dikkate alınarak belirlenir. Popülasyonun %50'den azının elit olarak kabul edilmesi performans açısından önemlidir.

Mutasyon değeri, çözüm uzayında rasgele değişiklikler yapmakta kullanılan bir parametre olup, joker fonksiyonu görür. Bu parametre kısmi çeşitlilik yaratmakta kullanılır. Ancak, bu parametrenin zararlı etkilerinden korunmak için önceki nesillerin elitleri saklanarak, bir sonraki nesille karşılaştırılmalıdır.

KAYNAKLAR

1. İnternet: "Course Notes of Xingde Jia, Ph.D. Professor of Mathematics, Texas State University". http://erdos.math.txstate.edu/resources /lecture_notes/ adm_ch11.pdf (2005).

2. Diestel, R. , "Graph Theory 3th Ed. ", ***Springer-Verlag Heidelberg***, New York, 1-37, 52-75 (2005) .

3. İnternet: "Free Online Encyclopedia, Wikimedia Foundation, Inc, USA" . http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm (2005).

4. İnternet: "Course Notes of Dekai Wu, Associate Professor of Department of Computer Science in Hong Kong University of Science and Technology ". http://www.cs.ust.hk/~dekai/271/notes/L10/L10.pdf (2005)

5. Riedl, A. , "Routing Optimization and Capacity Assignment in Multi-Service IP Networks", Doktora Tezi, ***The Institute of Communication Networks***, ***München Technical University ,*** München Germany, 40-56 (2003).

6. Lewis, C. , "Cisco TCP/IP Routing Professional Reference", 2nd edition, ***Mcgraw Hill*** , New York USA, 325-402 (1998).

7. Bolat, B., Erol, K.O., İmrak, C.E., " Mühendislik Uygulamalarında Genetik Algoritmalar ve Operatörlerin İşlevleri", ***Sigma Mühendislik ve Fen Bilimleri Dergisi***, 4: 264-281 (2004).

8. Soule, A., Nucci, A., Cruz, R. ,Leonardi, E., Taft, N. , "How to Identify and Estimate the Largest Traffic Matrix Elements in a Dynamic Environment", ***ACM Press***, New York USA, 73 – 84 (2004).

9. Fafali, P., Patrikakis, C., Michalas, A., Loumos, V., "Internet Traffic Engineering: History Monitoring Information Featuring Routing Algorithms", ***Automatics and Informatics Conference***, Sofia Bulgaria, 46-51 (2003).

10. Mitchell, M., "Introduction To Genetic Algorithms", ***Bradford Press***, London, 10-48 (1998).

11. İnternet: "Department of Computer Science and Engineering, FEE CTU Prague". http://cs.felk.cvut.cz/%7Exobitko/ga/main.html (2005).

12. İnternet: "Free Online Encyclopedia, Wikimedia Foundation, Inc, USA" . http://en.wikipedia.org/wiki/Computational_complexity_theory (2006).

13. Ericsson, M., Resende M., Pardalos, P., "A genetic algorithm for the weight setting problem in OSPF routing", ***Journal of Combinatorial Optimization***, 6: 299-333 (2002).

14. Mulyana, M., Killat, U., "An Alternative Genetic Algorithm to Optimize OSPF Weights", Internet Traffic Engineering and Traffic Management, ***15-th ITC Specialist Seminar***, Wuerzburg Germany, 186-192 (2002).

EKLER

EK-1 Komşuluk listesinden komşuluk matrisi üreten sözde kod

```
Prosedure:  Komşuluk Listesinden Komşuluk Matrisini Üret
Let  G=( Vi, Vj, Wij )    //Komşulu listesi verilsin
Let  N    // Düğüm sayısı verilsin

//Girilmeyen (olmayan) kenarların ağırlık değeri  sonsuzdur
For  i=1  to N
For  j=1 to N
 IF Wij  Girilmemişse
          Wij := Sonsuz
 Endif

// Bir düğümün kendisine olan uzaklığı sıfırdır
For  i=1  to N
For  j=1 to N
IF i=j
          Wij := 0
Endif
```

EK-2 Komşuluk listesinden komşuluk matrisi üreten PHP kodları çıktısı

```
//Komşuluk listesinde girilmemiş kenarların değerini sonsuz yap!
for ($counter1=0; $counter1 <$number_of_nodes; $counter1++) {
for ($counter2=0; $counter2 <$number_of_nodes ; $counter2++){
 if (!isset($w[$counter1][$counter2]))     $w[$counter1][$counter2]=INFINITY;
}
}

//Bir düğümün kendisine olan ağırlığını sıfırla!
//komşuluk  matrisinin köşegeni tamamen sıfır olmalı.
 for ($counter1=0; $counter1 <$number_of_nodes; $counter1++) {
 for ($counter2=0; $counter2 <$number_of_nodes ; $counter2++) {
    if ($counter1 == $counter2)    $w[$counter1][$counter2] = 0;
  }
  }
```

EK-3 Dijkstra algoritmasının sözde kodu

```
Prosedure  Dijkstra Algoritması
Let  G=( Vi, Vj, Wij )
Let  N    // Dügüm sayısı
Let  d( V0 ) : = 0
Let  S : = Ø
Let  Q : = V[G]  //düğüm kümesi

For  i=1  to N
          d(Vi ) : = ∞
           previous [Vi] =undefined     //işaretci
While  Q  ≠  Ø
           U : = testing point   //sınama düğümü
           S : =  S + U
           Q: = Q – U
            For  her bir (U,V) kenarı için
                  IF  d[U]+ w(U,V ) < d(V)
                        d(V) := d[U] + w(U,V )  // kenar rahatlaması
                         previous [V] :=U
```

EK-4 Dijkstra Algoritması ile veri iletişim ağındaki tüm düğümler arasındaki en kısa yolu hesaplayan PHP kodları

```
class Node {
var $predecessor;
var $link_cost;    // iki düğüm arasındaki en kısa yolun ağırlık değeri
var $label;
function  Node($k){
}
function baslangic_degeri_ata(){
$this->predecessor= -1;
$this->link_cost=INFINITY;
$this->label=TENT;
}
function hedef_nodu_duzenle(){
$this->link_cost=0;   //hedef düğümün ağırlık değeri sıfıra çekiliyor
$this->label=PERM;    // hedef  düğümün etiket değeri  sabitleniyor
}

} //Node isimli sınıf tanımı bitti

function Dijkstra (&$w,&$node_object,$number_of_nodes,$link_numbers){

for ($source_index=0;$source_index<$number_of_nodes;$source_index++){
for ($destination_index=0;$destination_index<$number_of_nodes;$destination_index++) {
if (!($destination_index==$source_index)) {
for ($counter=0;$counter<$link_numbers;$counter++)
    {
 $node_object[$counter]->baslangic_degeri_ata();                                    }
 $node_object[$destination_index]->hedef_nodu_duzenle();
 $k=$destination_index;

 do {
 for($i=0;$i<$number_of_nodes;$i++)
{
    if($w[$k][$i]!=0 && $node_object[$i]->label==TENT) {
        if(($node_object[$k]->link_cost + $w[$k][$i]) < $node_object[$i]->link_cost)
```

EK-4 (Devam) Dijkstra Algoritması ile veri iletişim ağındaki tüm düğümler arasındaki en kısa yolu hesaplayan PHP kodları

```
{
                    $node_object[$i]->predecessor=$k;
                     $node_object[$i]->link_cost=$node_object[$k]->link_cost + $w[$k][$i];
                  }       }    }
$k=0;
$min=INFINITY;
for($i=0;$i<$number_of_nodes;$i++)
    {       if($node_object[$i]->label==TENT &&  $node_object[$i]->link_cost <$min)
              {
               $min=$node_object[$i]->link_cost;
              $k=$i;
              }  }

$node_object[$k]->label=PERM;   }
while($k!=$source_index);  // k bizim kayan noktamiz, hedeften kaynaga kadar yayarak gidiyor
   //Routing Matrix Element Fonksiyonu Çağrılıyor
   $k=$source_index;
   for ($i=0;$i<$number_of_nodes; $i++)
   $transfer[$i]=$node_object[$i]->predecessor;
   $RM_element[$source_index][$destination_index]=
                                    Routing_Matrix_Element  ($number_of_nodes,$k,$transfer);
} else  {
$RM_element[$source_index][$destination_index]=
Routing_Matrix_Element_For_Same_DS_index($number_of_nodes);
}

} // source_index için for bitti
} // destination_index için for bitti
return $RM_element;
}//Dijktra fonksiyonu tanımlandı.
```

EK-5 OSPF ağının tüm linklerinin kullanım değerlerini hesaplayan PHP kod çıktısı

```
Function
Link_Utilization_Hesapla (&$w, &$node_object, $number_of_nodes, $link_numbers, &$TM)
{
 $RM_element=Dijkstra (&$w,&$node_object,$number_of_nodes,$link_numbers);
 $source_index=0;
foreach ($RM_element as $dimension1) {
            $destination_index=0;
 foreach ($dimension1 as $dimension2 => $value1) {
 foreach ($value1 as $internal_array1=>$value2) {
 foreach ($value2 as $internal_array2=>$value3){
 if (($internal_array2 >= 0)){
$Link_Capacity_Usage[$internal_array1][$internal_array2]=
$Link_Capacity_Usage[$internal_array1][$internal_array2]
+ $value3*$TM[$source_index][$destination_index];
 }
 }
 }
  $destination_index++;
 }
  $source_index++;
}
return $Link_Capacity_Usage;
}
```

EK-6 Link kullanımları istatistiği sunan fonksiyonun PHP kodu çıktısı

```
Function
Link_Statistic_Report_Function(&$w,&$node_object,$number_of_nodes,$link_numbers, $TM)
{
$Link_Capacity_Usage=
Link_Utilization_Hesapla(&$w,&$node_object,$number_of_nodes,$link_numbers,&$TM);
 $total=0;
 echo"</BR></BR>Link Kullanim Raporu:</BR></BR>";
 $used_link_counter=0;
 foreach($Link_Capacity_Usage as $dimension1=>$value1){
   foreach($value1 as $dimension2=>$value2 ){
              if ($value2 > 0 ){
               echo"</BR>Link_Capacity_Usage[$dimension1][$dimension2] : $value2 </BR>";
               $total=$total+$value2;
                $used_link_counter++;
                }//if kapandi
   }
 }
$ready_link_for_loading_counter=0; //halihazirdaki linkler
for ($counter1=0; $counter1 <$number_of_nodes; $counter1++) {
for ($counter2=0; $counter2 <$number_of_nodes ; $counter2++){
     if (($w[$counter1][$counter2]>0)&&($w[$counter1][$counter2]<INFINITY))
    $ready_link_for_loading_counter++;
}
}
$maksimum_total_link_numbers=$number_of_nodes*($number_of_nodes-1);
$unused_link=$ready_link_for_loading_counter-$used_link_counter;
$average_usage_for_one_direction=$total/($ready_link_for_loading_counter*0.5+0.0000000001);
echo"</BR></BR>Toplam $used_link_counter adet link üzerinde yük bulunmakta</BR></BR>";
echo"Toplam $unused_link link üzerinde hic yük bulunmamakta</BR></BR>";
echo"Yüklü ve yüksüz tüm linklerin kulanim ortalamasi (Tek bir yön
için):$average_usage_for_one_direction </BR></BR>";
 echo"</BR></BR>.....Link Kullanim Raporu Bitti.....</BR></BR>";
}
```

EK-7 Nesillerin oluşturulması sürecinin tanımlandığı PHP kodlarının çıktısı

```
1   /*-------------------------0. nesil yani baslangıç nesli için -----------------*/
2   if ($counter_of_population==0) {
3   $gene[$counter_of_population][$counter_of_gene]=Gene_Pickup_Function(&$w);
4   }
5   /*-------------------------------------------------------------------------------*/
6
7   /*--------------------------1. nesil için-----------------------------------*/
8   elseif($counter_of_population==1) {
9   for($counter1=0;$counter1<$number_of_kromozom;$counter1++){
10  for($counter2=0;$counter2<$gene_size;$counter2++){
11  $element_of_gene[$counter2]=rand(1,INFINITY);
12  }//for
13  $gene[$counter_of_population][$counter_of_gene]=$element_of_gene;
14  $counter_of_gene++;
15  }//for
    //0. neslin genini kullanmak için
16  $gene[$counter_of_population][0]=$gene[$counter_of_population-1][0]; 17

18  }
19  /*-------------------------------------------------------------------------------*/
```

EK-8 Rasgele Anahtarlar yöntemiyle çaprazlama yönteminin sözde kodu

```
Prosedure  Rasgele Matris Yöntemiyle Çaprazlama
Let N                        // N kromozom sayısı verilsin
Let Ata1 Ata2                // Ebeveyn kromozomlar verilsin
Let K                        // 0 ile 1 arasında olacak kesim değeri verilsin
For i = 0 to N
      RK[i]= (Random (0,10))/10      // 0 ile 1 arasında rasgele ondalık değer üret
For  i = 0 to N
If  RK[i] < K
       Yavru[i]=Ata1[i]
Else
        Yavru[i]=Ata2[i]
End
```

EK-9 Rasgele anahtarlar matrisi kullanarak çaprazlama ve mutasyon işlemlerinin yürütüldüğü PHP kodlamasının çıktısı

```
for($counter1=0;$counter1<$elite_gen_number_per_population;$counter1++){
//mutasyon sınaması için rasgele  matris üretiyoruz
for ($counterA=0;$counterA<$gene_size;$counterA++){
$random_matrix_for_mutation[$counterA]=rand(0,1000)/1000;
}
// elite-nonelite ayrımın ı yapabilmek için  rasgele anahtarlar matrisi üretiyoruz.
for ($counterB=0;$counterB<$gene_size;$counterB++){
$random_matrix_for_elite_portion[$counterB]=rand(0,10)/10;
}
//yeni nesil için yeni  yavrular üretiyoruz.
for ($counter2=0;$counter2<$gene_size;$counter2++){
 if($random_matrix_for_mutation[$counter2]<$mutation_factor)
            $generated_gene[$counter1][$counter2]=rand(1,INFINITY);
 elseif($random_matrix_for_elite_portion[$counter]<$crossover_cutoff)

          $generated_gene[$counter1][$counter2]=$keep_elite_gene[$counter1][$counter2];
   else
$generated_gene[$counter1][$counter2]=
            $keep_nonelite_gene[($counter1+$elite_gen_number_per_population)][$counter2];
}
}
```

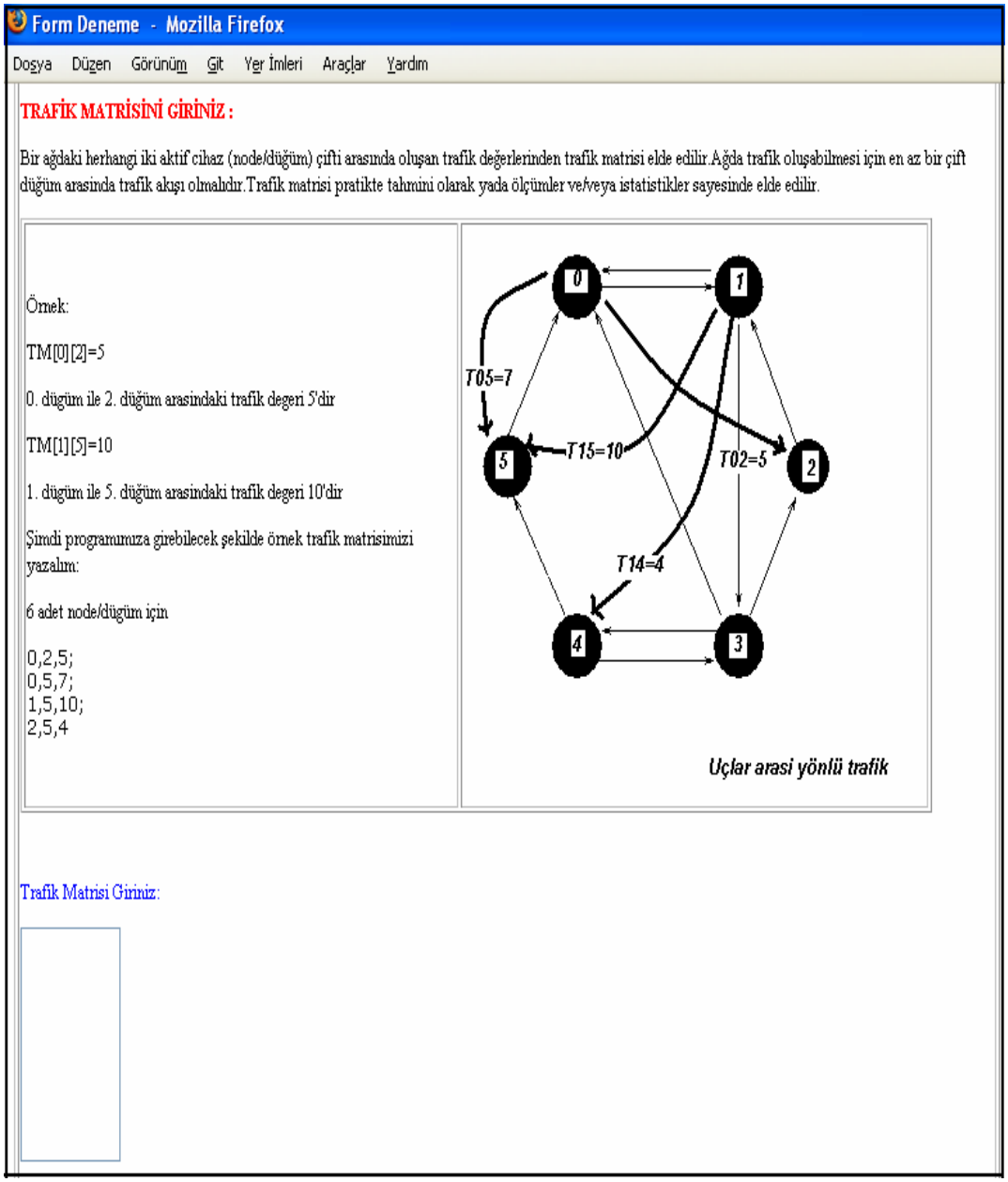
EK-10 Elitist Seçim yöntemi ile seleksiyonun yürütüldüğü PHP kod çıktısı

```
for ($counter=0;$counter<$elite_gen_number_per_population;$counter++)
{   $gene[$counter_of_population][$counter]=$keep_elite_gene[$counter]; }
$counter_generated_gene=0;
for
($counter=$elite_gen_number_per_population;
$counter<(2*$elite_gen_number_per_population) ; $counter++)
{
 $gene[$counter_of_population][$counter]=$generated_gene[$counter_generated_gene];
$counter_generated_gene++;
}

for($counter=(2*$elite_gen_number_per_population);$counter<$number_of_gene;$counter++)
{
 $gene[$counter_of_population][$counter]=
                $gene[$counter_of_population-1][$counter-$elite_gen_number_per_population];
}
```

EK-11 Programa graf elementlerinin girildiği web tabanlı kullanıcı arabiriminin görünüşü

EK-12 Programa trafik elementlerin girildiği web tabanlı kullanıcı arabiriminin görünüşü



EK-13 Deney 1'de kullanılan 1. trafik matrisinin program sentaksına uygun formatıyla hazırlanmış hali

0, 19, 120;

18, 1 ,132;

2, 17, 232;

4, 9, 67;

1, 4, 23;

5, 6, 41

EK-14 Deney 1-Deneme 2'nin uygunluk değerleri ve en son nesil için ağırlık değerleri çizelgesi

Nesil(Generation)	Uygunluk Değerleri
0 .Nesil	22.395159713
1 .Nesil	22.395159713
2 .Nesil	22.1028957379
3 .Nesil	22.1028957379
4 .Nesil	22.1028957379
5 .Nesil	22.1028957379
6 .Nesil	22.1028957379
7 .Nesil	22.1028957379
8 .Nesil	22.1028957379
9 .Nesil	22.1028957379
10 .Nesil	22.1028957379
11 .Nesil	22.1028957379
12 .Nesil	22.1028957379
13 .Nesil	22.1028957379
14 .Nesil	22.1028957379
15 .Nesil	22.1028957379
16 .Nesil	22.1028957379
17 .Nesil	22.1028957379
18 .Nesil	22.1028957379
19 .Nesil	21.789974759
20 .Nesil	21.789974759
21 .Nesil	21.789974759
22 .Nesil	21.789974759
23 .Nesil	21.789974759
24 .Nesil	21.789974759

EK-14 (Devam) Deney 1-Deneme 2'nin uygunluk değerleri ve en son nesil için ağırlık değerleri çizelgesi

25.Nesil (Generation)	En Uygun Ağırlık Değerleri
w[0][3]	6
w[0][6]	28
w[0][7]	21
w[0][8]	187
w[0][9]	183
w[0][10]	4
w[1][2]	236
w[1][4]	201
w[1][8]	181
w[1][9]	250
w[1][10]	221
w[1][11]	121
w[1][12]	187
w[2][1]	242
w[2][4]	188
w[2][10]	32
w[2][11]	155
w[2][12]	204
w[2][13]	23
w[2][14]	150
w[3][0]	100
w[3][4]	101
w[3][12]	211
w[3][13]	229
w[3][14]	224
w[3][15]	7
w[3][16]	157
w[4][3]	20
w[4][5]	231
w[4][14]	231
w[4][15]	222
w[4][16]	97
w[4][17]	91
w[4][18]	232
w[4][1]	227
w[4][2]	163
w[5][16]	137
w[5][17]	15
w[5][18]	221
w[5][19]	16

EK- 14 (Devam) Deney 1-Deneme 2'nin uygunluk değerleri ve en son nesil için ağırlık değerleri çizelgesi

w[5][4]	109
w[6][0]	163
w[7][0]	96
w[8][0]	6
w[8][1]	243
w[9][0]	233
w[9][1]	87
w[10][0]	231
w[10][1]	216
w[10][2]	140
w[11][1]	79
w[11][2]	85
w[13][2]	175
w[13][3]	183
w[14][2]	217
w[14][3]	52
w[14][4]	122
w[15][3]	122
w[15][4]	244
w[15][15]	0
w[16][3]	136
w[16][4]	12
w[16][5]	83
w[17][4]	128
w[17][5]	130
w[17][17]	0
w[18][4]	56
w[18][5]	60
w[19][5]	163

EK-15 Deney 1'de kullanılan 2. trafik matrisinin program sentaksına uygun formatıyla hazırlanmış hali

6, 19, 113;

19, 6, 43;

7, 18, 87;

18, 7, 17;

8, 11, 96;

10, 19, 65;

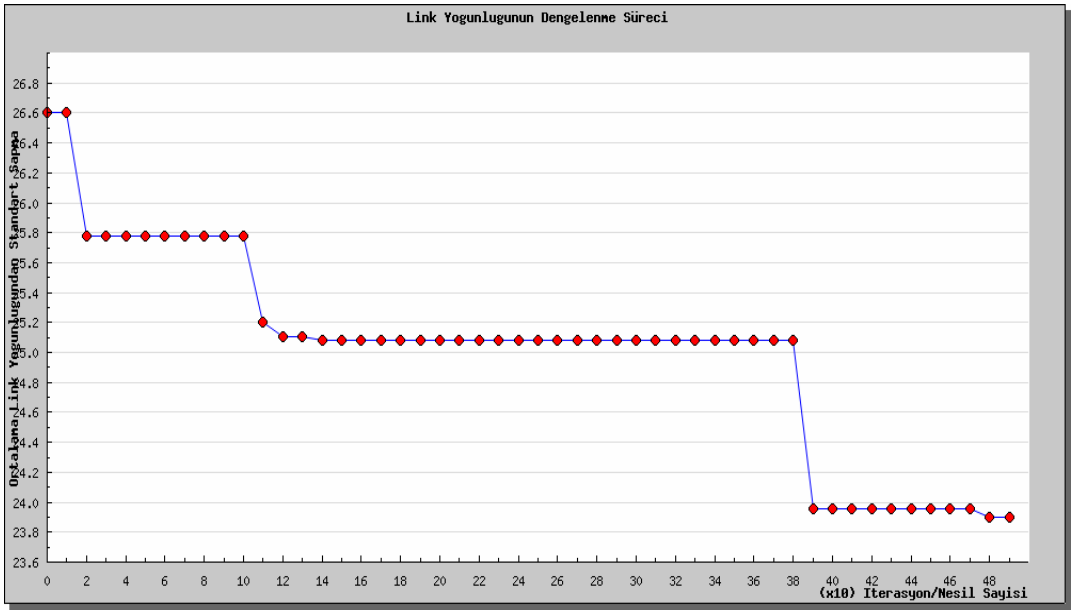
12, 18, 91;

16, 7, 135;

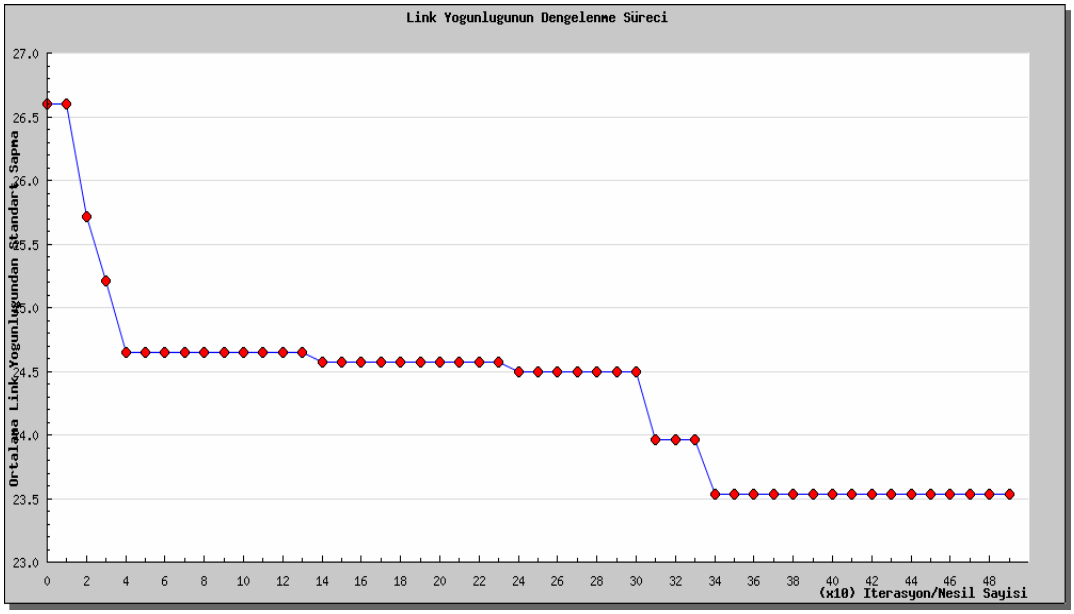
13, 14, 45;

14, 13, 171

EK-16 Deney 1'in son dört denemesinden elde edilen sonuçlar

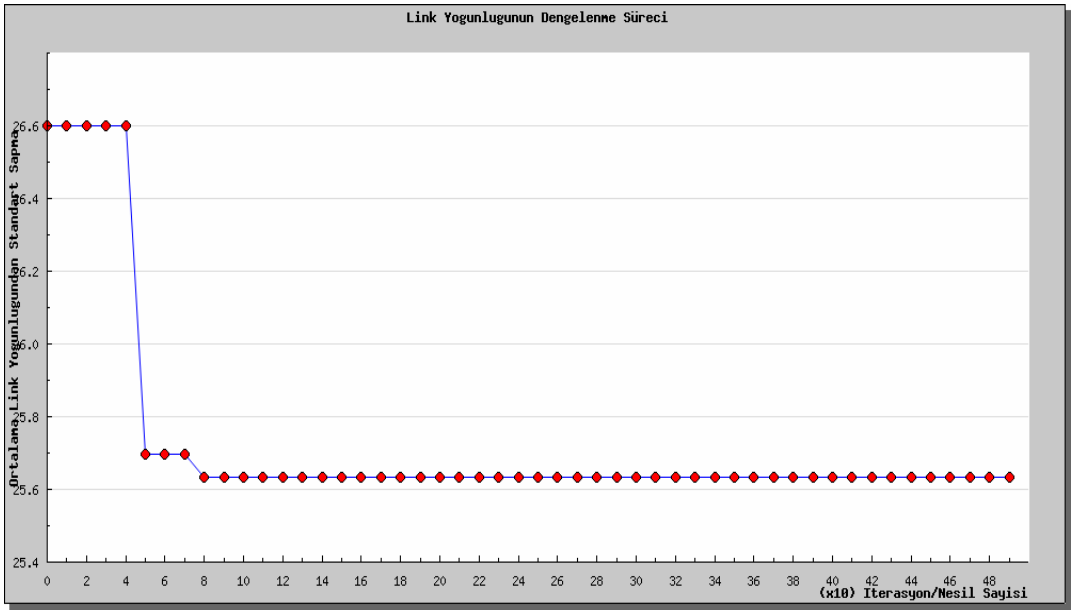


Şekil 16.1. Maksimum nesil parametresi 50 ve gen sayısı parametresi 10 iken yapılan denemenin sonuç grafiği

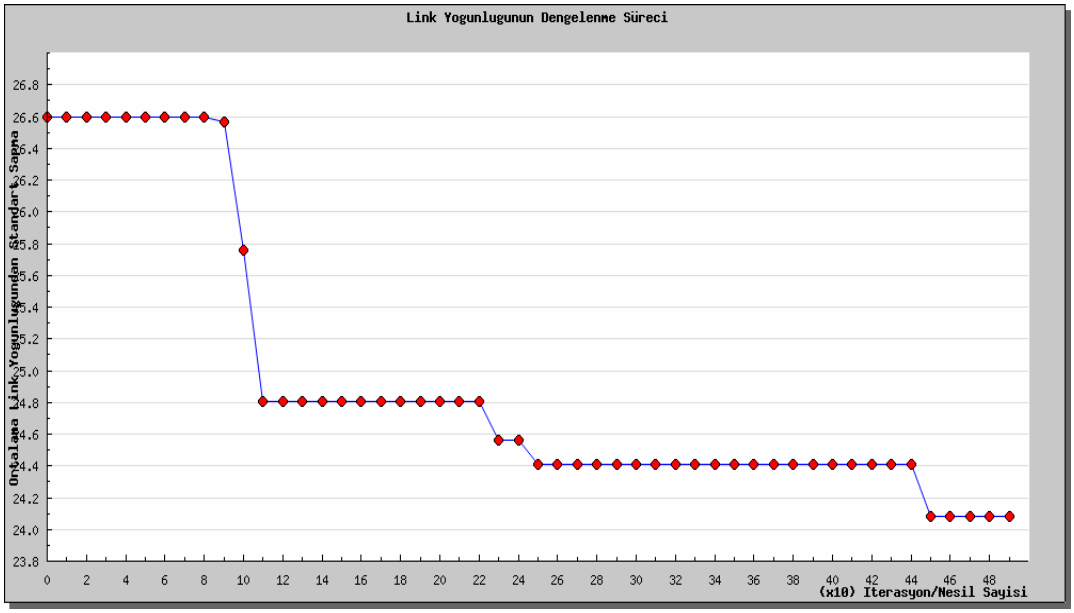


Şekil 16.2. Maksimum nesil parametresi 50 ve mutasyon parametresi 0,2 iken yapılan denemenin sonuç grafiği

EK-16 (Devam) Deney 1'in son dört denemesinden elde edilen sonuçlar



Şekil 16.3. Maksimum nesil parametresi 50 ve çaprazlama kesim parametresi 0,5 iken yapılan denemenin sonuç grafiği



Şekil 16.4. Maksimum nesil parametresi 50 ve çaprazlama kesim parametresi 0,8 iken yapılan denemenin sonuç grafiği

EK-17 Deney 2-Deneme 2'nin uygunluk değerleri ve en son nesil için ağırlık değerleri çizelgesi

Nesil (Generation)	Uygunluk Değerleri
0 .Nesil	36.9587543359
1 .Nesil	36.9587543359
2 .Nesil	26.2201258578
3 .Nesil	26.1406398443
4 .Nesil	26.1406398443
5 .Nesil	26.1406398443
6 .Nesil	24.6970913982
7 .Nesil	24.6970913982
8 .Nesil	24.6970913982
9 .Nesil	24.6970913982
10 .Nesil	24.6970913982
11 .Nesil	24.6970913982
12 .Nesil	24.6970913982
13 .Nesil	24.6970913982
14 .Nesil	24.6970913982
15 .Nesil	24.6970913982
16 .Nesil	24.6970913982
17 .Nesil	24.6970913982
18 .Nesil	24.6970913982
19 .Nesil	24.6970913982
20 .Nesil	24.6970913982
21 .Nesil	24.6970913982
22 .Nesil	24.6970913982
23 .Nesil	24.6970913982
24 .Nesil	24.6970913982

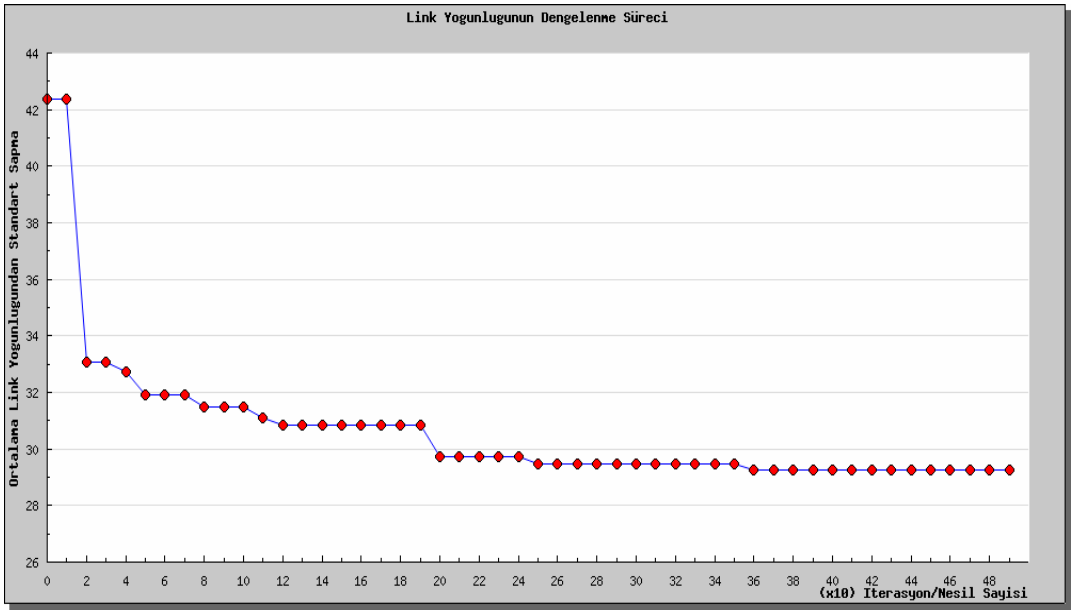
EK-17 (Devam) Deney 2-Deneme 2'nin uygunluk değerleri ve en son nesil için ağırlık değerleri çizelgesi

25.Nesil(Generation)	En Uygun Agirlik Degerleri
w[0][1]	133
w[0][2]	59
w[0][3]	185
w[0][4]	248
w[1][2]	221
w[1][4]	38
w[1][8]	217
w[1][0]	69
w[2][1]	28
w[2][0]	80
w[2][3]	234
w[2][7]	210
w[3][0]	120
w[3][2]	234
w[3][4]	51
w[3][6]	48
w[4][0]	157
w[4][1]	93
w[4][3]	224
w[4][5]	108
w[5][4]	70
w[5][8]	115
w[5][6]	82
w[5][9]	62
w[5][10]	9
w[6][7]	129
w[6][5]	42
w[6][3]	163
w[6][11]	124
w[6][10]	62
w[7][2]	216
w[7][8]	206
w[7][6]	135
w[7][11]	88
w[7][12]	224
w[8][1]	210
w[8][5]	45

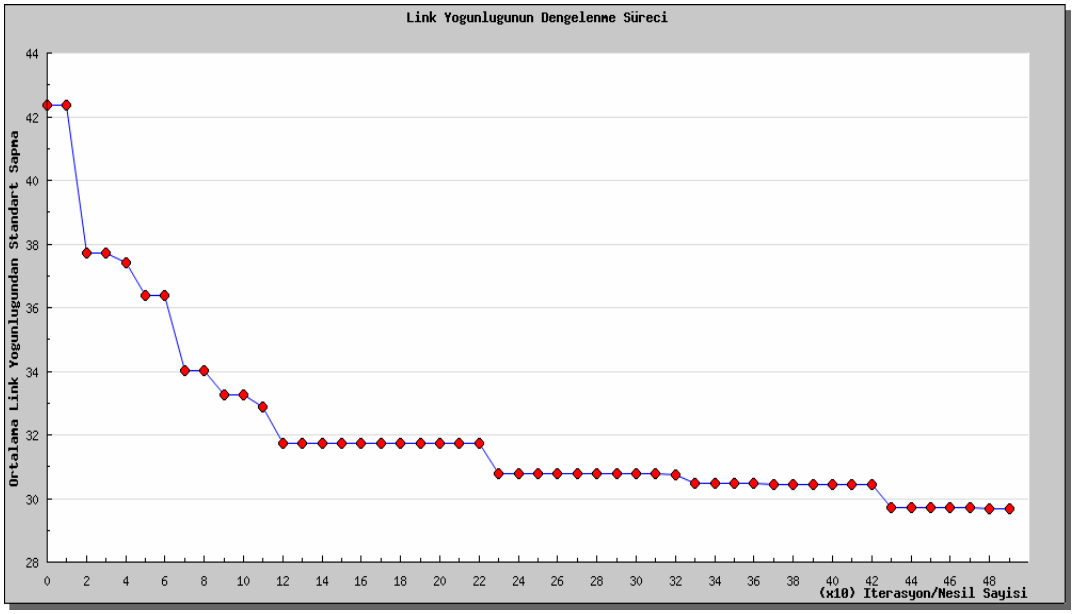
EK-17 (Devam) Deney 2-Deneme 2'nin uygunluk değerleri ve en son nesil için ağırlık değerleri çizelgesi

w[8][7]	23
w[8][9]	155
w[8][12]	145
w[9][8]	183
w[9][5]	51
w[9][19]	210
w[10][5]	106
w[10][6]	156
w[10][17]	217
w[10][18]	234
w[11][6]	138
w[11][7]	27
w[11][15]	15
w[11][16]	186
w[12][7]	55
w[12][8]	249
w[12][13]	144
w[12][14]	67
w[13][12]	34
w[13][14]	247
w[14][12]	34
w[14][13]	89
W[15][11]	218
W[15][16]	210
W[17][10]	23
xw[17][18]	193
w[18][10]	192
w[18][17]	108
w[19][9]	222

EK-18 Deney 2'in son 5 denemesinden elde edilen sonuçlar

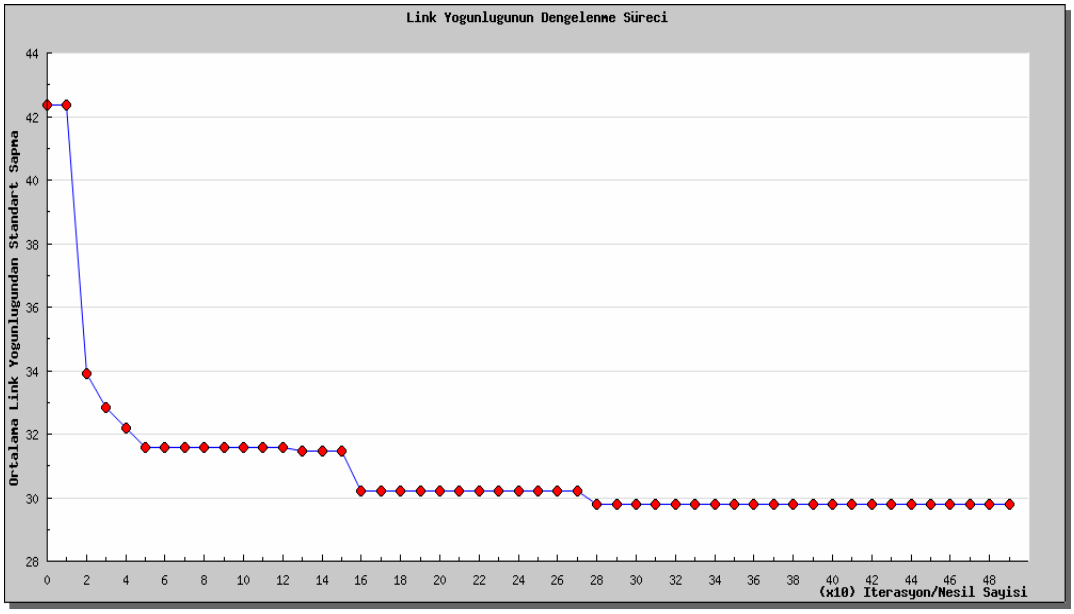


Şekil 18.1. Maksimum nesil parametresi 50 iken yapılan denemenin sonuç grafiği

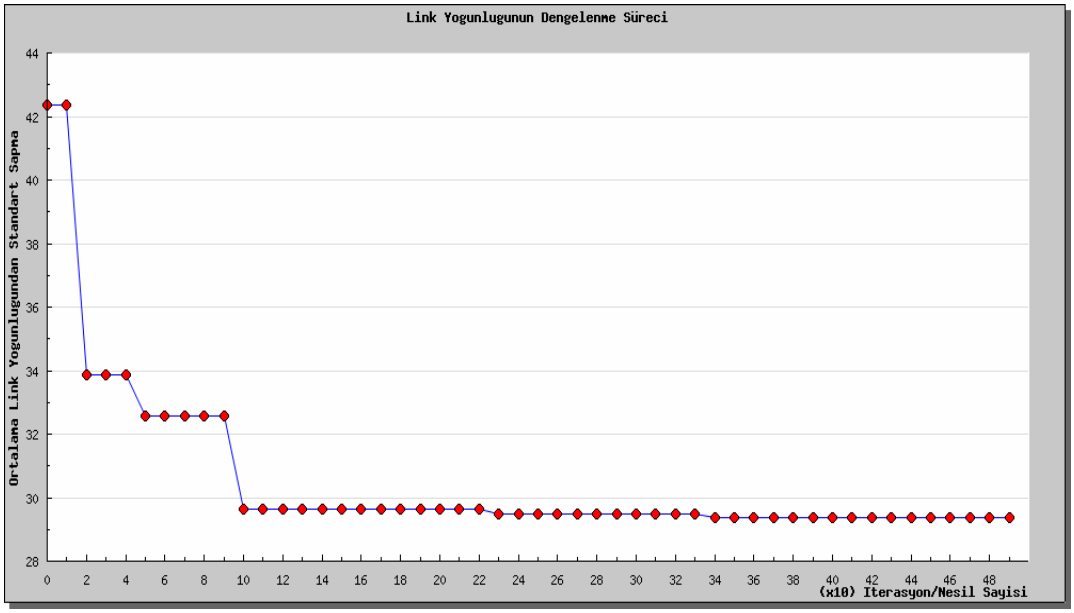


Şekil 18.2. Maksimum nesil parametresi 50 ve gen sayısı parametresi 10 iken yapılan denemenin sonuç grafiği

EK-18 (Devam) Deney 2'in son 5 denemesinden elde edilen sonuçlar

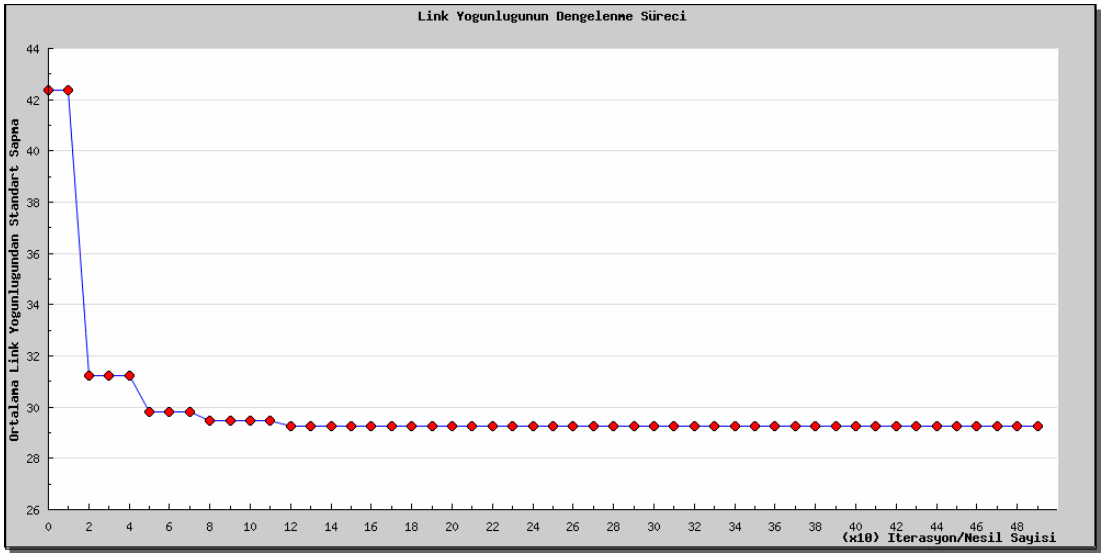


Şekil 18.3. Maksimum nesil parametresi 50 ve mutasyon değeri parametresi 0,2 iken yapılan denemenin sonuç grafiği



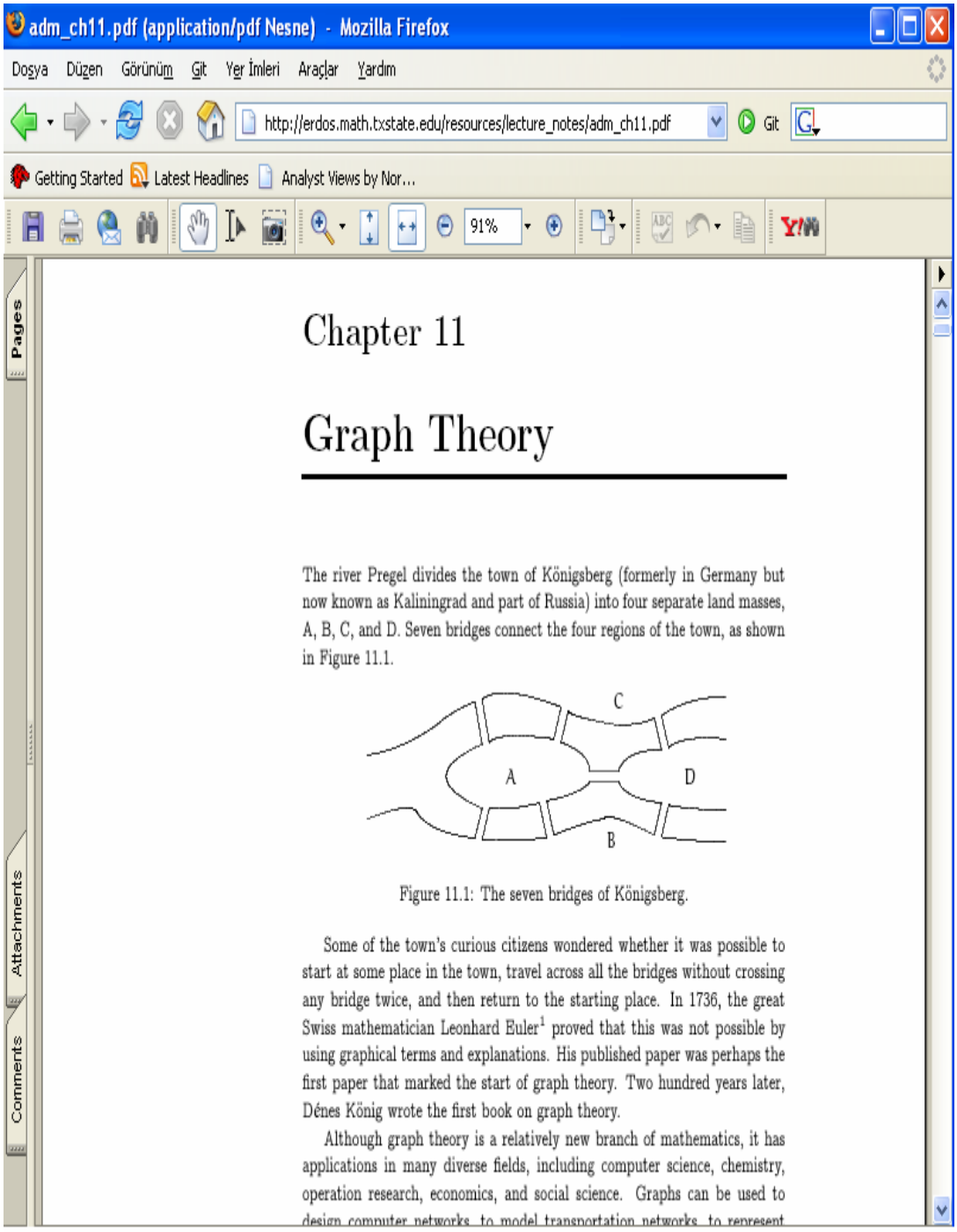
Şekil 18.4. Maksimum nesil parametresi 50 ve çaprazlama kesim parametresi 0,5 iken yapılan denemenin sonuç grafiği

EK-18 (Devam) Deney 2’ nin son 5 denemesinden elde edilen sonuçlar



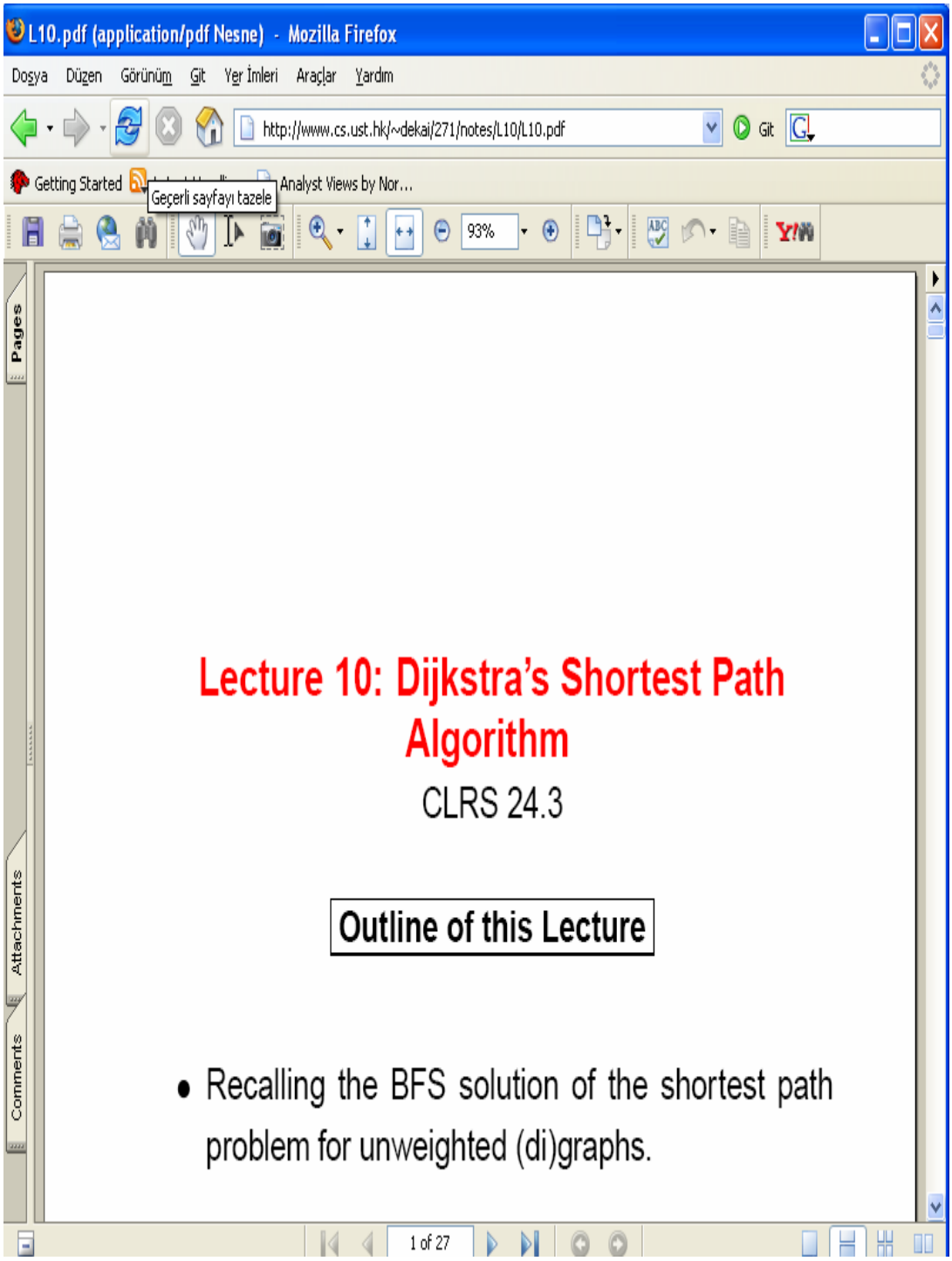
Şekil 18.5. Maksimum nesil parametresi 50 ve çaprazlama kesim parametresi 0,8 iken yapılan denemenin sonuç grafiği

EK-19 Kaynak olarak kullanılan internet adreslerine ait ilk sayfalar

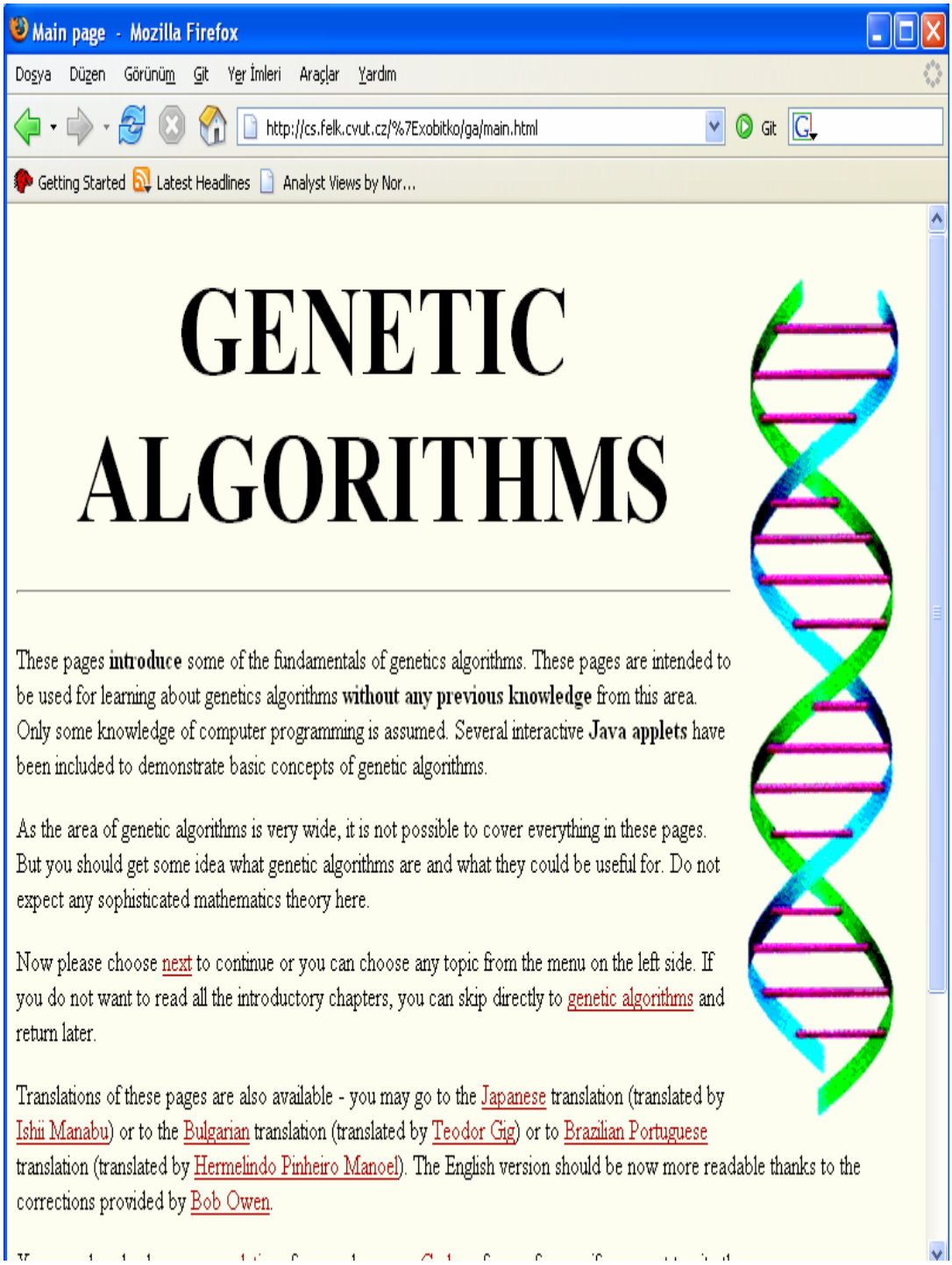


EK-19 (Devam) Kaynak olarak kullanılan internet adreslerine ait ilk sayfalar

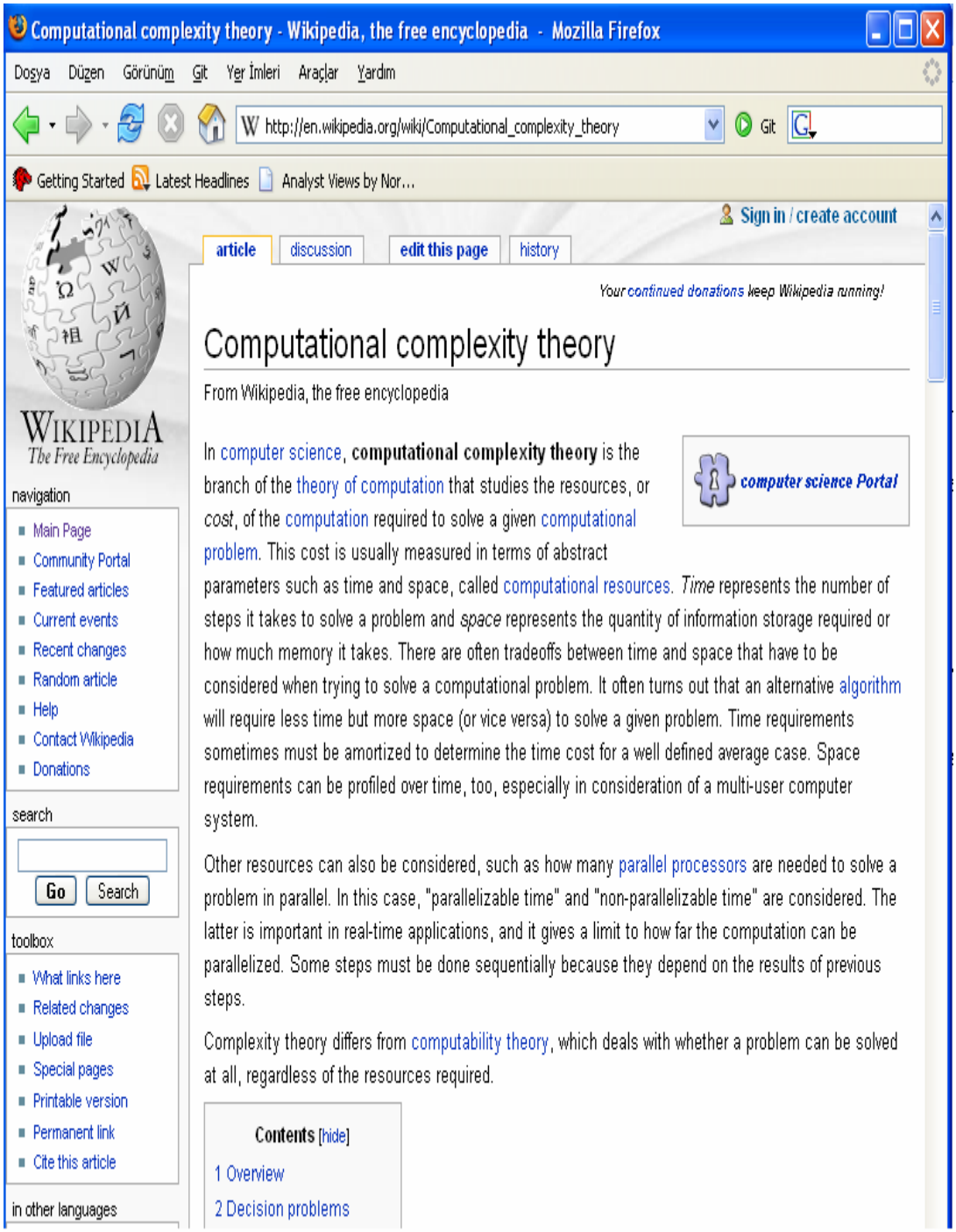
EK-19 (Devam) Kaynak olarak kullanılan internet adreslerine ait ilk sayfalar



EK-19 (Devam) Kaynak olarak kullanılan internet adreslerine ait ilk sayfalar



EK-19 (Devam) Kaynak olarak kullanılan internet adreslerine ait ilk sayfalar



ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ÖZBİLEN, Alper

Uyruğu : T.C.

Doğum tarihi ve yeri : 08.01.1980 ANTAKYA

Medeni hali : Bekar

Telefon : 0 (312) 215 74 80

e-mail : alper@ozbilen.net.

Eğitim

Derece	**Eğitim Birimi**	**Mezuniyet tarihi**
Lisans	Gazi Ünv. / Elek.-Elektronik Müh.	2003
Lise	Kırıkhan Lisesi	1996

İş Deneyimi

Yıl	**Yer**	**Görev**
2002-2003	Emre Bil. A.Ş	Ağ Mühendisi
2003-Devam	Türk Telekom	Telekom Uzmanı

Yabancı Dil

İngilizce

Yayınlar

1. Bilgisayar Ağları ve Güvenliği, ***Pusula Yayıncılık Ltd,*** (2005).

Hobiler

Bilgisayar teknolojileri, Yüzme, Seyahat