T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

;ENETİK ALGORİTMA VE GEZGİN SATICI PROBLEMİNİN ÇÖZÜMÜ

Mustafa KAYA

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

ELAZIĞ

1999

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GENETİK ALGORİTMA VE GEZGİN SATICI PROBLEMİNİN ÇÖZÜMÜ

Mustafa KAYA

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

Bu Tez, ... Tarihinde Aşağıda Belirtilen Jüri Tarafından Obirliği/Oyçokluğu ile Başarılı/Başarısız Olarak Değerlendirilmiştir.

(imza)	(imza)	(imza)
------	------	------

Danışman
Yrd. Doç. Dr. Ahmet ARSLAN

ÖZET

Yüksek Lisans Tezi

GENETİK ALGORİTMA VE GEZGİN SATICI PROBLEMİ

Mustafa KAYA

Fırat Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Ana Bilim Dalı

1999, sayfa 85

Bu çalışmada Genetik Algoritma ve Genetik Programlama ele alınıp, operatörleri ve algoritmaya hazırlanışı anlatılmıştır. Genetik programlamaya geçilirken karşılaşılan güçlükler ve bunlardan kurtulmanın yolları üzerinde çalışılmıştır.

Gezgin Satıcı Problemi (Travelling Salesman Problem) olarak bilinen konu, tanımlaması ve bağıntıları verilerek ele alınmıştır. Genetik Algoritma kullanılarak Gezgin Satıcı Problemi çözülmüş olmasına rağmen çözümün kalitesi üzerinde oldukça fazla tartışmalar yapılmıştır.

Türkiye' deki şehirler arası karayolları göz önüne alınarak, en kısa rota mesafesi yapılan genetik programlama ile hesaplanmaya çalışılmıştır. Bazı yardımcı algoritmalar ve veriyi daha iyi kodlama ile Genetik Aramanın çok iyi sonuçlar vereceği araştırmacılar tarafından savunulmakta ve üzerinde sıkça çalışılmaktadır.

Kullanılan iyileştirme algoritmaları verilmiş ve bu algoritmaların, problemi çözmede sağladığı avantajlar ve eksik tarafları vurgulanmıştır. Programın sonuçları, program kodu ve çıktısı yorumlanarak verilmiştir.

ANAHTAR KELİMELER: Genetik Algoritma, Gezgin Satıcı Problemi, Genetik Programlama, Rota Problemi.

ABSTRACT

Masters Thesis

GENETIC ALGORITHM AND TRAVELLING SALESMAN PROBLEM

Mustafa KAYA

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

1999, Page 85

In this research Genetic Algorithm and Genetic Programming are taken up. The operators and their preparing for algorithm are discussed. The difficulties of Genetic Programming and the technics to pass over are studied.

The definition and formulas of Travelling Salesman Problem are handled. TSP have been solved many times with Genetic Algorithms, but, the quality of these solutions have still been discussed.

The shortest route, with cties of Turkey, found by genetic programming. The results are given comperative. Many researchers defence and study that, if you use other subsidiary technics and better coding data, you can get better solutions.

Some heuristics were given, and their advantages and disadvantages on solutions of TSP are discussed. The results, program code and output of program are given by interpretation.

KEYWORDS: Genetic Algorithm,Traveller Salesman Problem, Genetic Programming, Route Problem.

TEŞEKKÜR

Tez çalışması süresince bilgi ve tecrübelerinden yararlandığım sayın Yrd. Doç. Dr. Ahmet ARSLAN beye, yardımlarını esirgemeyen Bilgisayar Müh. Ar.Gör. Ali KARCI ve Bilgisayar Yük. Müh. Ar. Gör. İbrahim KOLCU' ya teşekkür ederim.

İÇİNDEKİLER

ÖZET i

SUMMARY ii

TEŞEKKÜR iii

İÇİNDEKİLER iv

ŞEKİLLER v

TABLOLAR vi

SİMGELER ve KISALTMALAR vii

1. GİRİŞ 1

2. GENETİK ALGORİTMA 4

2.1. Başlangıç Topluluğu Oluşturma 4

2.2. Uygunluk Fonksiyonu 5

2.3. Yeniden Üretme 6

2.4. Çaprazlama 7

2.5. Mutasyon 8

2.6. Basit Genetik Algoritma Yaklaşımına Bir Örnek 9

3. GENETİK ALGORİTMALARIN FARKLI UYGULAMALARI 13

3.1. Rota Problemi 14

3.1.1. Ardışıl Temelli Çaprazlama 15

3.1.2. Rota Temelli Çaprazlama 17

3.2. Gezgin Satıcı Problemi Ve Hopfield Ağı İle Çözümünün Değerlendirilmesi 18

3.3. GSP' nin Bağıntıları 20

3.4. GSP' nin Çözümünde Kullanılan Diğer Yöntemler 23

3.4.1. Lin – Kernighan Yerel Arama Yöntemi 24

3.4.2. Isı Düşüşü Taklidi Yöntemi 25

3.4.3. Çizgelere Ayırma ve Çizge Üzerinden Çözüm Yöntemleri 25

3.4.4. Lin – Kernighan Yerel Arama ve Genişletilmeleri 26

3.4.5. Çizge Üzerinden Isı Düşüşü Taklidi ile İyileştirilmesi 27

3.4.6. Genetik Algoritma, Hızlı Tırmanma Tekniği ve Lokal Yapı 28

3.4.6.1. Hızlı Tırmanma Tekniği Değerlendirilmesi 29

3.4.6.2. Genetik Algoritmalar ve Lokal Yapı 30

3.4.7.Dallan ve Sınırla Yöntemi 31

3.4.8.Çok İhtimalli Optimum bulmada Genetik Aramanın Etkileri Üstüne Kihong Park Ve Bob Carter' ın Çalışmaları 32

3.4.8.1. Çözüm Kalitesi 35

3.4.8.2. Hazırlama 36

3.4.8.3. Problemi Kodlama 36

3.4.8.4. Çaprazlama 36

3.4.8.5. Sonuç Fonksiyonu 37

3.4.8.6. Mutasyon Oranı 37

4. YAPILAN ÇALIŞMA 38

4.1. Çizge Üzerinde Yardımcı Algoritma Geliştirme 38

4.2. Birey Kırarak Çaprazlama 41

5. SONUÇLAR 47

KAYNAKLAR 49

EKLER 51

ŞEKİLLER

Şekil 2.1. 15 Bit Uzunluğunda Birey Hazırlanması 4

Şekil 2.2. Rulet Tekerleği ile Seçim Yöntemi 7

Şekil 2.3. GA Akış Diyagramı 10

Şekil 2.4 L Uzunluklu Çubuk Momenti 10

Şekil 3.1. Ardışıl Temelli Çaprazlama 16

Şekil 3.2. Rota Temelli Çaprazlama 17

Şekil 4.1 Örnek Bir Rota İçin Oluşturulan Çizge 39

Şekil 4.2. Programda Bilgi Giriş Menüsü 45

Şekil 4.3. Önerilen Rotanın Grafik Ekranda Çizge Olarak Çıktısı 46

TABLOLAR

Tablo 2.1. Başlangıç Topluluğu 7

Tablo 2.2. Biyolojik Çaprazlama Örneği 8

Tablo 2.3. İki Bitlik Çaprazlama Örneği 8

Tablo 2.4. Mutasyon Operatörü 9

Tablo 2.5. GA' nın El ile Uygulaması I 11

Tablo 2.6. GA' nın El ile Uygulaması II 12

Tablo 3.1. Platolar ve yerel yapı ile fonksiyonda GA ve HTT. 31

Tablo 4.1. Kırarak Çaprazlama 43

SİMGELER

$C=[c_{i,j}]$	:	Çizge matrisi.
$D=[d_{i,j}]$	:	Derece matrisi.
$X_{i,j}$	:	i. ve j. Şehirler arasındaki mesafe.
E	:	Rpi ve Rdi arasındaki mutlak hatanın toplamı.
$E_{i,j}$	:	Çizge üzerinde V_i ve V_j vektörlerinin birleştiği köşe.
F	:	Bir bireyin uygunluk değeri.
N	:	Bir topluluktaki veri sayısı.
O_i	:	i. elemanı seçme olasılığı.
Rpi	:	Birey tarafından oluşturulan sonuç.
Rdi	:	Ulaşılmak istenen veri.
V_i	:	Çizge üzerinde yönlendirilmemiş vektör.

KISALTMALAR

BOH	:	Blok Oluşturma Hipotezi (Building Block Hipotesis, BBH).
GA	:	Genetik Algoritma.
GSP	:	Gezgin Satıcı Problemi.
HTT	:	Hızlı Tırmanma Tekniği.
IDT	:	Isı Düşüşü Taklidi (Simulated Annealing, S-A).
İBÇ	:	İkili Bölünmüş Çizge.
L-K	:	Lin – Kernighan iyileştirme yöntemi.
TSP	:	Travelling Salesman Problem.
ÇMS	:	Çaprazlama Mutasyon Seçme ile arama yöntemi.
ÇM	:	Çaprazlama Mutasyon ile arama yöntemi.
ÇS	:	Çaprazlama Seçme ile arama yöntemi.
MS	:	Mutasyon Seçme ile arama yöntemi.

1. GİRİŞ

Modern bilimde, veri kümeleri arasındaki ilişkileri, tecrübelerden de faydalanarak belirlemek, üzerinde çokça çalışılan yeni bir araştırma konusudur. Günümüzdeki araştırma problemleri eskiye nazaran çok daha karmaşık hal almıştır. Bu karmaşıklık problemi etkileyen parametre sayısının fazlalığından ve problemin çözüm kümesinin boyutunun büyümesinden kaynaklanmaktadır. Bundan dolayı mevcut verilerin analizi ve problemin çözümünü, bu verilerden tahmin etme yöntemlerinin önemi gittikçe artmaktadır. Günümüzde bir veri analiz yöntemi genellikle şu kriterlere göre değerlendirilmektedir; iyi tahmin veya sonucu tahmin etmeye yönelik olmalı, sistemin içindeki her bir mekanizma analiz edilebilmeli ve sonuçları mümkün olabilecek çözüm uzayı kümesinde olmalıdır. Bu tür problemlerdeki çözüm kümesinin büyüklüğü bir taraftan elde edilen çözümün değerlendirilmesinde zorluk çıkarırken diğer taraftan lineer yöntemlerin uygulanmasını imkansız kılacaktır.

Geçmişte araştırmacılar tarafından çalışılan, parametreler arasındaki ilişkiler, genelde deneme yoluyla, zor olan örneklerde karmaşık veya sabit olmayan ilişkiler için yapılmış; fakat parametre sayısı artınca çözümsüzlük veya elde edilen çözümü değerlendirememe problemini ortaya çıkarmıştır. İstatistiksel yöntemler, araştırmacılara ilişkileri bulmada faydalı olan ilk araçlardandır. İstatistiksel yöntemlerde: (1) verinin normal toplandığı, (2) verinin eşitlik ilişkisinin belirli bir formda olması (ör: lineer, quadratik, veya polinomsal), (3) değişkenlerin bağımsız olması gerekir (Hays 1988). Eğer problem bu kriterleri sağlarsa, istatistiksel yöntem ilişkileri bulmada faydalı olabilir. Oysa gerçek hayatta problemler bu kriterleri nadiren sağlarlar.

Modern sonuç tahmin etme veya sonuç geliştirme algoritmaları bu kriterlerle sınırlandırılamazlar. Yapay sinir ağları veya yapay zeka teknikleri karmaşık ilişkileri kapsamayabilir; fakat mekanizmanın önemli ilişkilerini tanımlayabilen güçlü tahmin modelleridir. Buna rağmen, diğer bir teknik olan genetik algoritma ve genetik programlama teknikleri çok daha güçlüdürler ve karışık çözüm uzayını daha da genişleterek sonucu bulabilirler. Bağımsız olan veri ve parametreler ile mekanizmanın ilişkilerini bulmada genetik algoritmanın başarılı örnekleri vardır.

Genetik Algoritma (GA), biyolojik bir sistemin, çevresine adaptasyonunda kullandığı yöntemin örneklenmesidir. Bilgisayarda, bu tür çok parametreli optimum bulma problemlerine ve makine öğrenme problemlerine çözüm modeli olarak alınabilir. Doğal adaptasyondan esinlenen GA' nın basit olarak iskeleti:

a) Bireyin bulunduğu ortamda hayatta kalmak için, kendi kendisini değiştirerek ortama uygun hale gelmesi,
b) Bu adaptasyon boyunca, yeni üretilecek nesillere, bu özellikler ile birlikte mümkün olabilecek daha çok değişim aktarılarak, bireylerin daha çok uyumlu hale getirilmesi olarak özetlenebilir.

Genetik algoritma, tam olarak rasgele arama tekniğidir. İlk defa Michigan Üniversitesi' nde John Holland tarafından geliştirilmiştir. Araştırmalarını, arama ve optimumu bulma için, doğal seçme ve genetik evrimden yola çıkarak yapmıştır. İşlem boyunca, biyolojik sistemde bireyin bulunduğu çevreye uyum sağlayıp daha uygun hale gelmesi örnek alınarak, optimum bulma ve makine öğrenme problemlerinde, bilgisayar yazılımları geliştirilmiştir.

Mühendislikte, bilimde, ekonomide, finansmanda v.s. problemleri çözmede kullanılan arama teknikleri, hesap-temelli ve direkt-arama teknikleri olarak sınıflandırılabilir. Eğer problemler sayısal veya analitik olarak iyi tanımlanabiliyorsa veya çözüm uzayı küçük ve tek ise, hesap temelli arama tekniği daha iyi sonuç verir. Buna rağmen hesap-temelli teknik, mühendislikte gittikçe artan optimum bulma fonksiyonlarında oldukça zayıf kalmaktadır. Sadece fonksiyon bilgisi gerekli olan Doğrudan arama tekniği, hesap-temelli teknikten daha kısa sürede işler ve daha etkilidir. Doğrudan arama tekniğinin esas problemi, ulaşılabilen bilgisayar zamanı ile optimal çözümün kesinliği arasındaki bağıntıdır (Goldberg, 1989).

GA, arama problemlerinde oldukça etkilidir, çözüm kümesinden optimum veya optimuma yakın çözümleri bulabilir. GA, adım-adım çözüm topluluğuna genetik operatörler uygulayarak, uygun topluluktan arama yoluyla yeni nesiller üreterek en iyi çözümlere ulaşmasını sağlar. GA' lar geçmiş yıllarda, robotlarda optimum bulma problemlerin çeşitliliği ve çözüm kümesinin doğrudan bulunamadığı durumlarda ve değişik alanlarda karşılaşılan benzer problemlerdeki çözümlerden dolayı mühendislerin daha çok kullandığı algoritmalar olmuştur. İlk olarak Hollanda' da makine öğrenme

sistemleri de yardımcı olarak kullanılmış, daha sonra GA' nın çok sayıda kollara ayrılmış gaz borularında, gaz akışını düzenlemek ve kontrol etmek için uygulanması tanımlanmıştır (Goldberg, 1989). Ayrıca kendisinin kullandığı makine öğrenmesi, nesne tanıma, görüntü işleme ve işlemsel arama gibi alanlarda kullanıldığı vurgulanmıştır.
Genetik Algoritma normal arama işleminden 4 yönden farklıdır.

1) GA parametrelerin kendisiyle değil, oluşturulabilir bir koduyla çalışır.
2) GA çözüm kümesine doğrudan tek nokta ile değil, noktalar topluluğuyla yaklaşır.
3) GA sadece fonksiyon bilgisini kullanır. Matematik bilgisine ihtiyaç duymaz.
4) GA sebep-sonuç ilişkisi içinde rasgele çalışan operatörler kullanır. Sebep-sonuç ilişkisi kurallarını değil ihtimal kurallarını kullanır.

GA, sonlu uzunlukta tanımlanan bit katarı bir kod nesnenin fonksiyonunun parametrelerini ihtimal kurallar kümesini kullanarak işletir. GA kördür, problem hakkında uygunluk fonksiyonundan başka hiç bir şey bilmez. Yeni nesil oluşturmayı ve bireysel katarlar topluluğunu test etmeyi adım-adım işletir. Topluluktaki bireylerin uygunluk değerlerini yükseltmeye çalışırlar.

Bu tezde 2. bölümde genetik algoritma anlatılarak, operatörleri tanımlanacaktır, 3. bölümde farklı uygulamalarından söz edilecektir. 4. Bölümde, bu algoritma kullanılarak GSP (Gezgin Satıcı Problemi) için Türkiye' deki şehirler arası karayolları baz alınarak çözüm aranacaktır. Probleme GA' nın uygulanması üzerinde çalışılırken, hızlı bir şekilde optimum çözümü bulmak için GA' nın operatörleri üzerinde değişiklik yapma, farklı tanımlama da hedeflenmektedir.

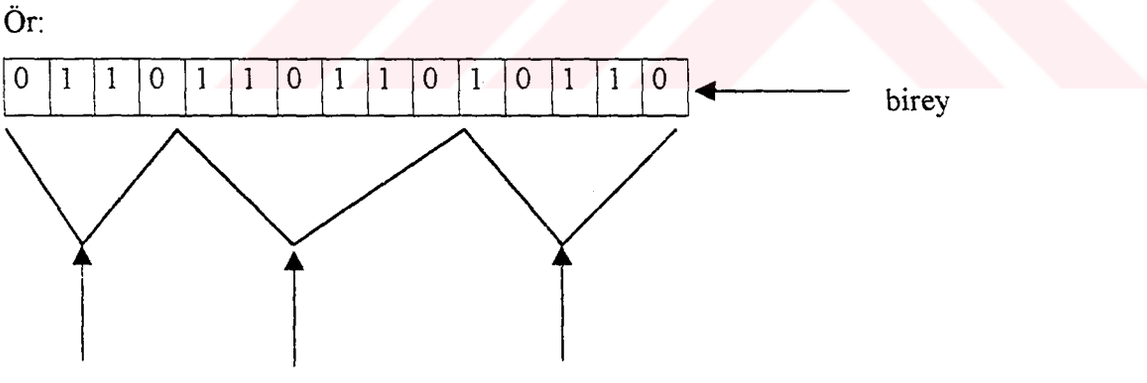
Son bölümde, operatörler üzerinde yapılan değişiklikler ile sonucu bulmada nasıl bir avantaj sağladığı ve daha neler yapılabileceği vurgulanmıştır. Yapılan programın kodu ve bilgisayar çıktısı ekler bölümünde verilmiştir.

2. GENETİK ALGORİTMA

Genetik algoritmada bulunması gereken yeniden üretme, çaprazlama ve mutasyon operatörleri tanımlanacak, genetik algoritmanın akış diyagramı ve el ile uygulanması verilecektir.

2.1. Başlangıç Topluluğunu Oluşturma

GA' yı bir probleme uygulamadan önce, verinin nasıl kodlanacağı ve mümkün olabilen çözümler kümesinden örnek çözümlerin bilinmesi gerekir. Genel olarak en yaygın kodlama uzunluğu belirlenmiş bit katarı şeklindedir. Klasik GA' da, topluluktaki her bir birey, problemdeki parametreler kümesinin kodlandığı belirli uzunlukta ikili sayı sistemindeki (binary) katardan oluşur. Bu kodlanan katarın yeniden kodu çözülerek parametrelerin onluk sayı sistemindeki değeri verilebilir.



Şekil 2.1. 15 bit uzunluğunda bir bireyin hazırlanışı

Başlangıç topluluğu, rasgele veya tahmin etme ile genelleştirilmiş potansiyel çözümler kümesidir. Rasgele olarak seçilen ilk topluluk, çözümler hakkında hiç bilgi yoksa kullanılır. İkinci yöntemde yaklaşıma bilinen bir çözüm kümesinden başlanılır. Böylece

çözüme ulaşma daha kısa sürede olacaktır. Fakat genelde çözüm topluluğu program içinde birey oluşturma işlemi kullanılarak rasgele seçilir.

2.2. Uygunluk Fonksiyonu

Başlangıç topluluğu bir kez oluşturulduktan sonra evrim başlar. GA bireylerin uygunluk ve iyi özelliklerine göre ayrılıp fark edilmesine gerek duyar. Uygunluk, topluluktaki bir kısım bireyin problemi nasıl çözeceği için iyi bir ölçüdür. Problem parametrelerini kodlamayla ölçülür ve uygunluk fonksiyonuna giriş olarak kullanılır. Yüksek ihtimalle uygun olan bu üyeler yeniden üretme, çaprazlama ve mutasyon operatörleriyle seçilirler.

Bazı problemler için bireyin uygunluğu, bireyden elde edilen sonuç ile tahmin edilen sonuç arasındaki hatadan bulunabilir. Daha uygun bireylerde bu hata sıfıra yakın olur. Bu hata genellikle, girişin yeniden sunulacak kombinezonlarının ortalaması veya toplamıyla hesaplanır (değerler değişkenlerden bağımsızdır). Beklenen ve üretilen değer arasındaki koralasyon etkeni, uygunluk değerini hesaplamak için kullanılabilir (Koza, 1994).

Formül 2.1.' de uygunluk değerinin hesaplanması görülmektedir. Burada, F uygunluk değerini, Rpi birey tarafından oluşturulan sonucu ve Rdi ise ulaşılmak istenen sonucu temsil etmektedir. Burada uygunluk değeri 0 ile 1 arasında olacaktır.

$$F=1/(1+\sum_{i=1}^{N}|Rpi-Rdi|) \qquad (2.1.)$$

2.3. Yeniden Üretme

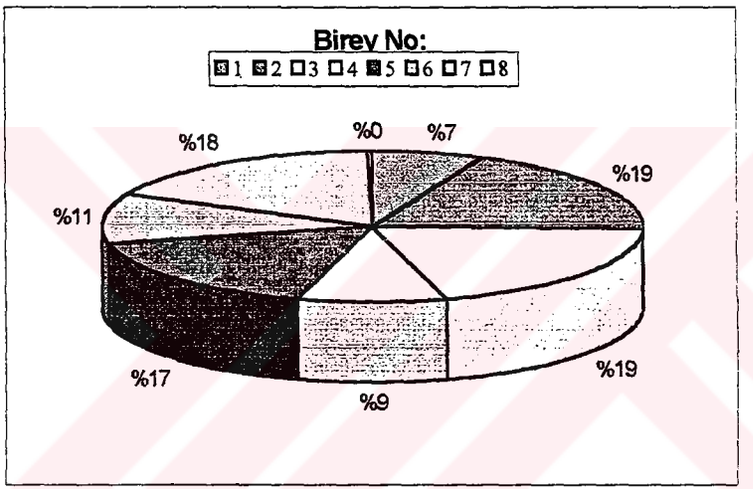
Yeniden üretme operatörü, hazır topluluktan uygun olan bireylerin seçilmesi ve bunların sonraki topluluğa kopyalanarak hayatta kalmalarıyla ilgilidir. Seçim modeli, tabiatın hayatta kalabilmek için uygunluk mekanizması modelidir. Yeniden üretme işleminde, bireyler onların uygunluk fonksiyonlarına göre kopya edilirler. Uygunluk fonksiyonu, mümkün olduğu kadar yükseltilmesi gereken faydalı bir ölçüdür. Topluluk uzayındaki her bir bireyin uygunlukları baz alınarak ne kadar sayıda kopyasının olacağına karar verilir. En iyi bireylerden daha fazla kopya döl alınır, en kötü bireylerden kopya alınmaz. Bu hayatta kalmak için uygunluk stratejisinin GA ya sağladığı avantajdır.

GA tarafından üretilen döllerin sayısını belirlemede birkaç yol vardır. Birbirine yakın parametrelerden kaçınmak için uygun bir seçim yöntemi kullanılmalıdır. Yeniden üretme başlangıcında basit bir yöntem olan rulet tekerleğiyle seçme' ye (Şekil 2.2.) göre, bireylerin uygunluk değerlerini bir rulet tekerleğinde hazırlar. Rasgele tekerleğin döndürülmesinden sonra, bireyin bir sonraki nesil için seçilmesi, tekerlek üzerinde kapladığı alanla doğrudan bağlantılıdır. Bu yöntem düşük uygunluğa sahip bireylere de seçilme hakkı verir. Rulet tekerleğiyle seçim yönteminde seçilen değer formül 2.2. ile bulunabilir. F_i, i. Eleman için uygunluk değerini ve N birey sayısını göstermektedir.

$$P_{seçilen} = F_i / \sum_{i=1}^{N} F_i \qquad (2.2.)$$

Tablo 2.1. Başlangıç Topluluğu

x	f(x)	uygunluk
56	143	0,533333
104	47	1,517949
108	39	1,6
191	127	0,697436
96	63	1,353846
72	111	0,861538
156	57	1,415385
224	193	0,020513



Şekil 2.2. Rulet Tekerleği İle Seçim

2.4. Çaprazlama

Biyolojik kromozomlar üzerinde yapılan çaprazlamada amaç daha iyi, daha dayanıklı birey oluşturup ait olduğu bireyler topluluğunun neslini dışarıdan etkiyle daha mükemmel hale getirmektir. Bu işlem yapılırken her zaman sonuçlar önceden tahmin edilemez. Bu yüzden gelişigüzel yapılan değişikliklerde sonucun mükemmelliğe doğru gitmesi için belirli kriterler bulmak için çalışılır. Kromozomlardaki genlerin yapısı ve

etkileri araştırılarak, bu genlere yapılan müdahalelerle bireye bazı iyi özellikler kazandırılabilir. Tablo 2.2. de biyolojik çaprazlamaya bir örnek verilmiştir.

Tablo 2.2. Biyolojik Çaprazlama Örneği

1.ebeveyn	AA BB	2. ebeveyn	aa bb
1. döl	AAbb	2. Döl	AaBB

Benzer şekilde GA, çaprazlama işlemini uygunluk değerlerine göre seçilmiş iki ebeveyn bireyden, iyi özellikte yeni bireyler elde etmek için kullanır. Çaprazlama rasgele seçilmiş iki çift katarın içindeki alt küme bilgilerin değiştirilmesi işlemdir. Kendi içindeki bilgilerini 1. Pozisyondan itibaren, katarın uzunluğunun bir eksik pozisyonuna kadar, aradaki bilgi kısmen karşılıklı bireyler arasında yer değiştirilir. Eğer iki bireyin problemin çözümünde bazı etkileri var ise onların bir parçaları faydalı, iyi veya uygun nitelenebilecek bilgi taşımaktadır. Çaprazlama belki problemin çözümünde, bu faydalı bilgileri birleştirerek, daha çok etkili yeni bireyler üretecektir. Tablo 2.3.' de ikili kodda verilmiş bir katarda, 2 bitlik bir çaprazlama işlemi verilmiştir.

Tablo 2.3. İki Bitlik Çaprazlama Örneği

1. ebeveyn	**100110**|**00**	1. Döl	**10011001**
2. ebeveyn	110111|01	2. Döl	11011**100**

2.5. Mutasyon

GA' nın ikinci mekanizması olan mutasyon katardaki rasgele bir bitlik bilginin eşleniğinin alınmasıyla ilgilidir. Mutasyon aramada kısır döngüye girilmemesini sağlamak, toplulukta çözüm olmayan birbirine benzer bireylerden kurtulmak ve yeni alt optimal çözümler bulunmasını sağlamak için kullanılır. İkili kodda bu ilişki 0' ın 1' e

ve 1' in 0' a değiştirilmesidir. Mutasyon ihtimalinin değeri, genellikle 0.001 ile 0.01 sınırında çok küçük tutulur. Topluluk büyüklüğü arttıkça bu ihtimal değerinin de küçülmesi gerekmektedir. Tablo 2.4. bit katarında hazırlanmış bir mutasyon operatörünü göstermektedir.

Tablo 2.4. Mutasyon Operatörü

	Mutasyondan önce	Mutasyondan sonra
1. döl	10011001	1|1|011001
2. döl	11011100	1101|0|100

2.6. Basit Genetik Algoritma Yaklaşımına Bir Örnek

Basit bir genetik algoritma akış şeması aşağıdaki adımlardan oluşur. Şekil 2.3. de şema olarak ta verilmiştir.

a) Başlangıç anını belirle

b) Toplulukta yer alacak bireyleri oluştur.

c) Bireylerin yaşam koşullarına uygunluğunu değerlendir.

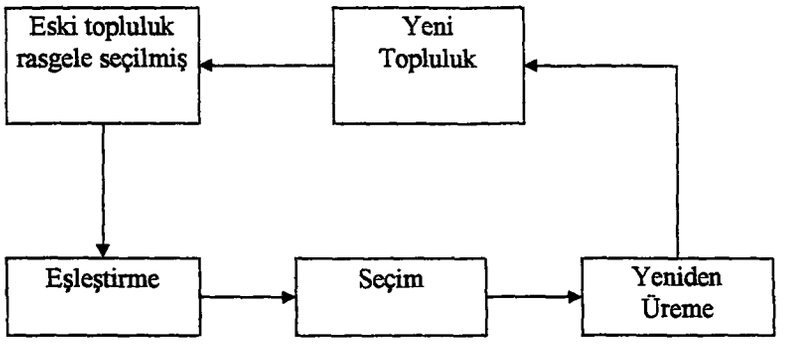
d) İstenilen düzeye ulaşıncaya kadar başarılı bireyleri oluştur.

e) Yeni bir nesil için başlangıç anını bir arttır.

f) Bir sonraki döl verecek bireyleri seç ve geçici topluluğa kopyala.

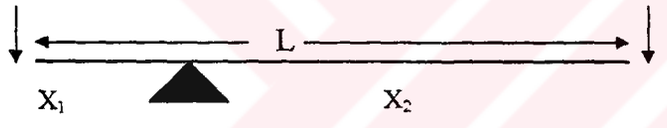
g) İterasyonun sonuna gelinmedi ise c) adımına geri dön.

h) Sonucu ver.



Şekil 2.3. GA Akış Diyagramı

Basit bir genetik algoritma, L uzunluğundaki homojen bir çubuğun momentinin 0 (sıfır) olduğu noktayı bulmak için kurulabilir (Şekil 2.4.).



Şekil 2.4. L Uzunluğundaki Çubuğun Momenti

L uzunluğuna bağlı olarak ara uzunluk x_1 ve x_2 0 ile 255 cm arasında işaretsiz bir tamsayı alınacaktır. $F(x)=x_1.m1-x_2.m2$ minimum değeri, aynı zamanda uygunluk fonksiyonudur. Dolayısıyla özgül kütle lineer dağıldığından, doğrudan uzunluklara bağlı olacaktır. Yani uygunluk fonksiyonu daha basit bir şekilde $F(x)=x_1-x_2=0$ olarak verilebilir. Burada $x_2=L-x_1$ olduğundan $F(x_1)=2.x_1-L=0$ alınabilir. Örnek basit seçildiğinden sonucun matematik bilgisine göre 127.5 cm olacağı açıkça görülmektedir. 0 ile 255 arasındaki sayıları gösterebilmek için 8 bit uzunluğunda bir bilgiye ihtiyaç vardır (0 00000000 , ..., 255 11111111).

İlk olarak başlangıç topluluğu bu sayılar arasından rasgele olarak seçilebilir. Bunun için 8 birey oluşturulursa (N=8), bunlar {0,40,80,120,160,200,220,255} alınabilir. Bu değerler ikili sayı sisteminde aşağıdaki gibi gösterilebilir:

0 00000000

40 00101000

80 01010000

120 01111000

160 10100000

200 11001000

220 11011100

255 11111111

Bu değerler başlangıç topluluğu olarak alınırsa, bireylerin uygunluk değerleri, sayının ondalık karşılığının uygunluk fonksiyonuna verdiği cevapla bulunabilir. Uygunluk değeri sayının alabileceği değerler ile orantılı olursa, bunun için maksimum uygunluk değeri olarak 255 ve minimum olarak ta 0 uygunluk değeri alınmalıdır. Eğer elde edilen uygunluk değeri 255 ten çıkarılırsa maksimum uygunluğa 255 sayısının karşılık düştüğü görülebilir.

Örnek:

x=00000000 için F(x)=255-ABS(255-0)=0

x=00101000 için F(x)=255-ABS(255-80)=80 şeklinde belirlenebilir.

Tablo 2.5. GA nın El ile Uygulaması I, Goldberg, 1989

Sayı	String	F(x)	fi/top(f)	fi/fort	Pseç(ruletten)
0	00000000	0	0	0	0
40	00101000	80	0,094118	0,752941	1
80	01010010	160	0,188235	1,505882	2
120	01111000	240	0,282353	2,258824	2
160	10100000	190	0,223529	1,788235	2
200	11000100	110	0,129412	1,035294	1
220	11010100	70	0,082353	0,658824	1
255	11111111	0	0	0	0
	TOPLAM	850			
	ORTALAMA	106,25			
	MAX	240			
	MİN	0			

Bireylerin uygunluk değerlerinin rulete yansımasından kaç adet kopya oluşturulacağı veya hiç kopya alınmayacağı belirlenir. Yukarıdaki tabloya göre 80,120 ve 160 sayılarından 2 şer kopya alınacaktır. 0 ve 255 sayılarından hiç kopya alınmayacak ve diğer sayılardan 1 er kopya alınacaktır.

Buna göre çaprazlama operatörü yardımıyla yeni bireyler üretmeden önce başlangıç topluluğu Tablo 2.6.' deki gibi yeniden oluşturulmalıdır.

Tablo 2.6. GA nın El ile Uygulaması II Goldberg,1989

ikili karşılığı	gele seçilen	Yeni topluluk	Değeri	255-2fi	fi/topfi	2-fi/fort	pseç
01111000	2	00111000	56	112	0,088889	0,711111	1
00101000	1	**01**101000	104	208	0,165079	1,320635	1
01010000	6	11**010000**	108	216	0,171429	1,371429	1
01111000	5	10**111000**	191	128	0,101587	0,812698	1
10100000	4	**01**100000	96	192	0,152381	1,219048	1
11001000	3	01001000	72	144	0,114286	0,914286	1
11011100	8	10**011100**	156	198	0,157143	1,257143	1
10100000	7	11100000	224	62	0,049206	0,393651	0
		TOPLAM		1260	1,000000	8,000000	
		ORT		158	0,125000	1,000000	
		MAX		216	0,171429	1,371429	
		MİN		62	0,049206	0,393651	

Tablo 2.6.' de verilen işlemlerde, bir nesilden diğer nesile kromozomlar aktarılırken, uygun kromozomların iyi özellikleri yeniden üretme adımında kullanılarak en iyi kromozomlar elde edilebilir. Elle yapılabilecek bu örnekte üç nesil sonra çözüm olabilecek bireylerin çözüm uzayını yeterince daraltarak birbirlerine daha çok benzeyecekleri görülecektir.

3. GENETİK ALGORİTMANIN FARKLI UYGULAMALARI

2. bölümde verilen uygulamalar oldukça basit seçilmiştir. Ancak yapılacak değişik uygulamalarda gerek uygunluk fonksiyonunu belirleme, gerekse sonucun iyi bir çözüm olup olmadığına karar vermenin oldukça güç olduğu görülür. Örneğin bir rota problemi için rotaları kodlamak gerekir. Ancak rota kodlama, sayıları kodlamadan çok daha zor ve karışıktır. Aynı şekilde çaprazlama operatörleri ve uygunluk fonksiyonları farklı seçilmiştir. çaprazlama operatörleri tanımlanırken doğrudan yerleşim merkezleri arasındaki rotaları birbirleriyle çaprazlamak bu tür problemlere kolaylık getirirken uygulamada ve mümkün olmayan çözüm sayısını arttırdığından probleme farklı zorluklar getirir. Çünkü rotalar çaprazlanırken rotanın dışında kalan yerleşim merkezleri veya mümkün olmayacak yeni rotalar oluşur. Dolayısıyla bu çözümleri de yeniden düzenlemek gerekir. Bu probleme çözüm olarak genetik programlama kullanılırken bir taraftan da iyileştirme algoritmaları olan Lin-Kernighan, IDT , HTT (Hill Climbing), Blok oluşturma (Building Block) algoritmaları kullanılmaktadır. Problem gerçekten parametre sayısı oldukça fazla olabileceğinden ve çözüm uzayının çok büyük olmasından dolayı bilinen matematiksel çözüm yöntemlerinin hepsi ve daha sonra yaklaşım yöntemleri de çokça uygulanmıştır. Ancak bütün bu yöntemleri gerçekte birbirleriyle karşılaştırmak, sadece belirli örnekler üzerinde zaman açısından ele alınarak yapılabilmektedir. Ancak optimum bulma yöntemleriyle sadece belirli sınırlamalar getirildiği taktirde çözüme gidilebileceği aksi durumda milyonlarca ihtimalli hesaplamalar gerektiği açıktır. Bundan dolayı son yıllarda araştırmacılar yeni geliştirilmekte olan yaklaşım yöntemleri uygulayarak daha fazla başarı elde etmişlerdir. Ancak bu yöntemlerde de uygulama çeşitliği olduğu için sonuca ulaşmak için her bir araştırmacının kendi kullandığı yardımcı algoritmalar ve basit sınırlamalar problem çözümüne ayrıca kolaylıklar sağlamış ve sonuca daha hızlı gitmede etkili olmuştur. Bunun yanında çözüm kalitesini arttırmada ve bunu diğer yöntemlerle karşılaştırmada da yine araştırmacıların farklı uygulamaları farklı sonuçlara götürmüştür. Bu araştırmacıların çalışmalarından bazıları ilerleyen alt bölümlerde verilecektir.

3.1. Rota Problemi

Jaen-Yves Potvin ve Sam Bangio, 1996, rota problemine ilişkin çalışmalarında iki farklı çaprazlama operatörleri tanımlamışlardır. Bunlar Ardışıl Temelli Çaprazlama (Sequence-Based Crossover SBX) ve Rota Temelli Çaprazlama (Route-Based Crossover RBX) dir. Bu operatörler aşağıda belirtilmiştir. Burada çözüm kalitesi için uygunluk kavramı A çözümü B çözümünden daha iyidir şeklinde tanımlanmıştır. Eğer:

a) A çözümü B çözümünden daha az rotaya sahipse veya
b) A ve B çözümleri aynı sayıda rotaya sahip fakat A çözümünün toplam rota zamanı daha küçükse.

En iyi çözümlere karşı tahminle seçme için bir lineer sıralama şekli kullanılır. Önce hazır olan topluluktaki bütün çözümler onların kalitelerine göre en iyi çözüm rank 1'e sahiptir ve daha kötü çözüm topluluk büyüklüğü olan rank N' e sahiptir. sıraya dizilir. Rank i çözümü için uygunluk değeri aşağıdaki gibi bulunabilir.

$$\textbf{Uygunluk}_i = \textbf{MAK} - [\,(\textbf{MAK} - \textbf{MİN}) * (i-1)/(N-1)\,] \qquad (3.1.)$$

MAK=1.6

MİN=0.4

Örnek:

N=5 için

$Uygunluk_1=1.6$

$Uygunluk_2=1.3$

$Uygunluk_3=1.0$

$Uygunluk_4=0.7$

$Uygunluk_5=0.4$

olacaktır.

1.6 ve üstündeki uygunluk değerine sahip olan bireylerden 2 adet kopya alınacak, 0.4 ve altındakiler topluluktan atılacak ve arada kalan uygunluk değerine sahip bireylerden birer kopya alınacaktır.

Orantılı uygunluk seçme şekli bunlara uygulanırsa Rank i çözümü için O_i i. elemanı seçme olasılığı formül 3.2. ile hesaplanabilir.

$$O_i = \frac{\text{Uygunluk}_i}{\sum_{J=1}^{N} \text{Uygunluk}_i} = \frac{\text{Uygunluk}_i}{N} \qquad (3.2.)$$

Topluluktaki bütün uygunluk değerlerinin toplamı N' ye eşit olacaktır. Çünkü MİN+MAK=2 dir ve uygunluk değerlerinin ortalaması 1 dir. Uygunluk değerleri doğrudan bireyin seçilme sayısını verecektir. Örneğin MAK=1.6 uygunluk ile en iyi çözüme götürecek birey, farklı denemeler üzerinden 1.6 defa seçilecektir.

3.1.1. Ardışıl Temelli Çaprazlama

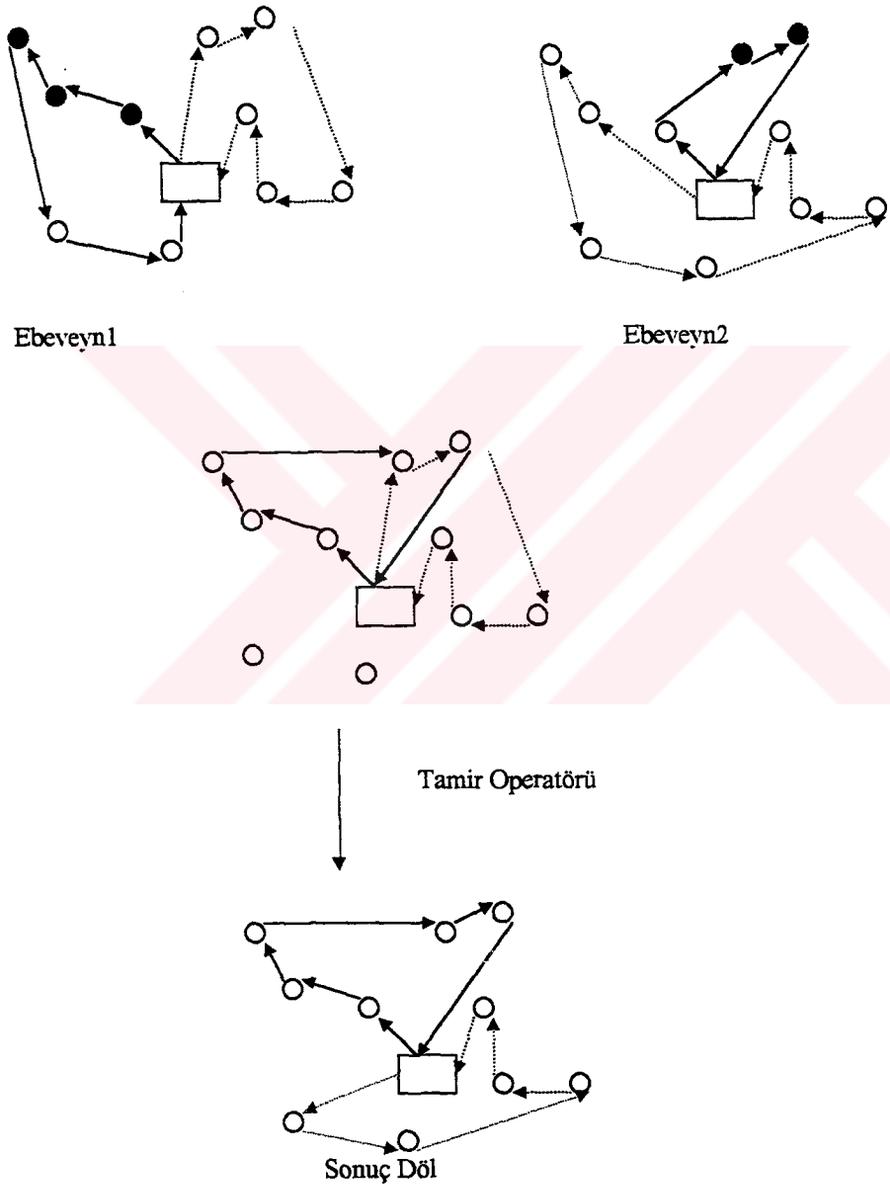
Bu çaprazlama şekil 3.3.' de gösterilmiştir. Rasgele seçilmiş her bir ebeveyn çözümden hareket eden bir bağlantıdır. Ebeveyn1 'in rotasında bağlantı noktasından önce servise konulmuş müşteriler, ebeveyn2' nin rotasında bağlantı noktasından sonraki müşterilere bağlanır. Son olarak yeni rota ebeveyn1 de eski olanıyla yer değiştirir. İkinci bir döl ebeveynlerin rolleri yer değiştirilerek elde edilebilir.

Mümkün olabilen çözümde, ebeveyn1' in rotasındaki ilk müşteriler ebnveyn2' nin rotasındaki son müşterilere bağlanır. Maalesef, yeni çözüm nadiren geçerlidir. Çünkü bazı müşteriler iki kez tekrarlanmış veya işlemin rotası dışındadır. Örnek olarak Şekil 3.1. deki iki müşteri şimdi farklı iki rotaya konumlanmıştır. Diğer iki müşteride rota dışındadır. Buna göre tamir operatörü, yeni mümkün bir çözüm elde edebilmek için döle uygulanır. Bu operatör aşağıdaki şekilde davranır.

a) Eğer bir müşteri yeni rota da iki defa görünüyorsa, iki kopyadan biri kopyadan silinir. Eğer bir müşteri, yeni rota da bir defa görünüyorsa ve eski rotada da bir tane ise, müşteri eski rotadan silinir. Bu durumlar çözüm için kolaydır. Çünkü; mümkün bir rota müşterilerden biri silinince yine mümkün olabilir.
b) Eğer bir müşteri rotada değil ise, bu müşteri eklenen dolambaçlı yolu minimuma indirecek şekilde araya konabilecek bir yere konulur. Rota dışındaki müşteriler daha zor elde edilecektir. Çünkü; onların her biri için mümkün olacak ara yer

garantili değildir. Eğer bu durum oluşursa döl atılır ve iki yeni ebeveyn seçmek için yeni bir seçim aşaması başlatılır. Rotada olmayan müşteriyi hatalı araya koyma, mümkün olmayan dölün ana kaynağıdır.

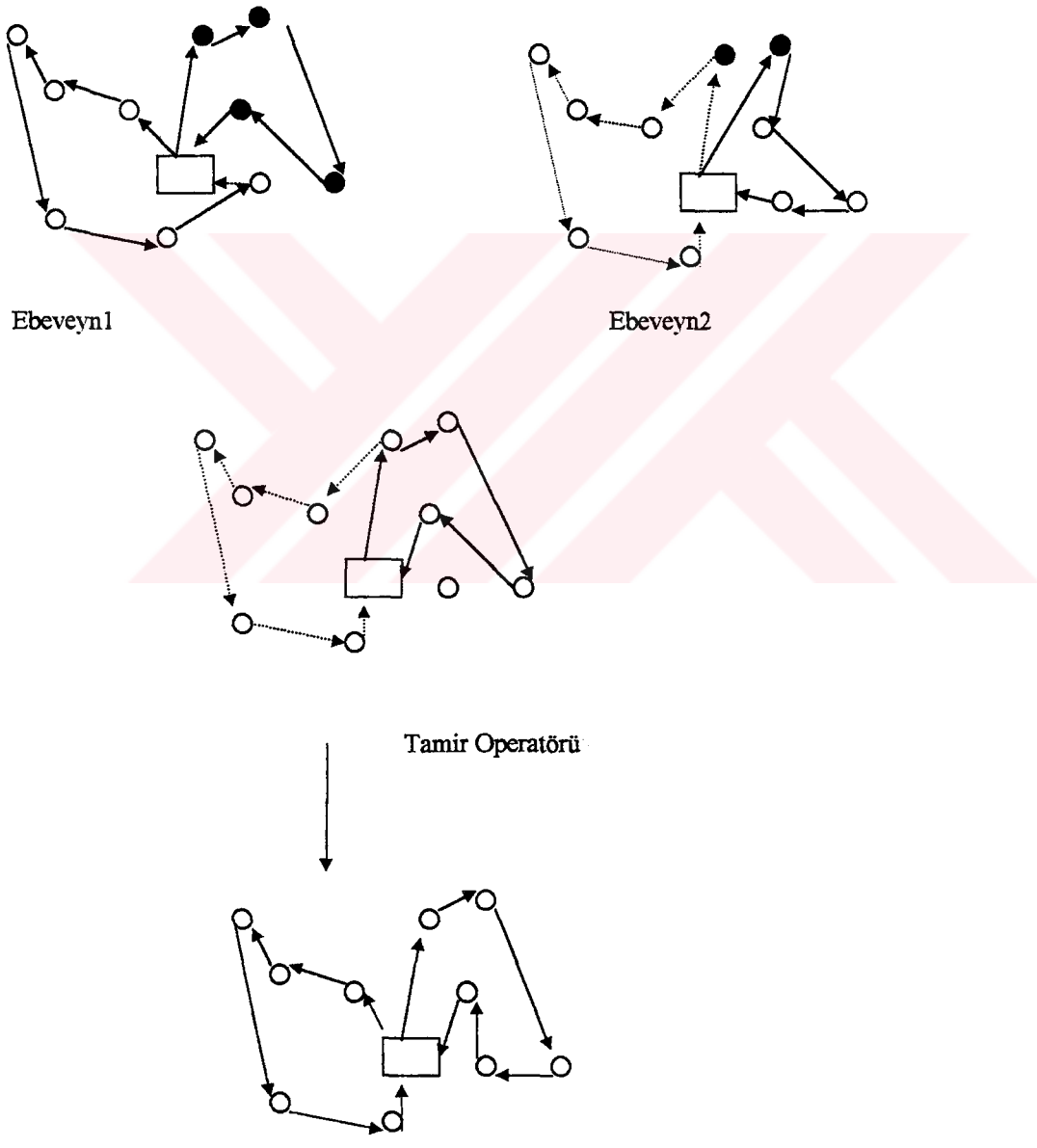
c) Yapılan test problemlerinde %50 döl mümkün olmamaktadır. N adet yeni döl topluluğu oluşturabilmek için 2N defa sırayla çaprazlama operatörü kullanılmalıdır.



Şekil 3.1. Ardışıl Temelli Çaprazlama

3.1.2. Rota Temelli Çaprazlama

Bu operatör, ebeveyn2 nin rotasıyla ebeveyn1 in rotasını yer değiştirerek yeni bir döl oluşturur. Rotada olmayan müşterileri ve iki defa alınan müşterileri Ardışıl Temelli Çaprazlama da olduğu gibi düzeltme yoluna gidilir. Şekil 3.2. de ebevey1 deki müşterilerin olduğu rota ile ebeveyn2 deki karşılayan rota ile yeri değiştirilir. Görüldüğü gibi bazı müşteriler rota dışında iken, bir müşteri şimdi iki ayrı rotada yerleşti. Bunun için daha önce anlatılan tamir operatörü ile yeni mümkün olabilecek döl oluşturulur.



Şekil 3.2. Rota Temelli Çaprazlama

3.2. Gezgin Satıcı Problemi ve Hopfield Ağı İle Çözümünün Değerlendirilmesi

Kate Smith (SMITH, K., 1996) çalışmasında, Gezgin Satıcı Problemi (GSP) çözümü için Hopfield ağının uygunluğunu sunan araştırmacılar ile bir optimum bulma tekniği olarak Hopfield ağının etkisini tanımlamaya çalışan araştırmacılar arasında bir fark çizmiştir. GSP' nin Alternatif lineer formüllerinin olması, sebepsiz karşılaştırmaların artması, genellikle bir Benchmark olarak yanlış kullanıldığından olduğunu vurgulamıştır.

Zor olan NP-hard optimum bulma problemlerinin çözümlerini ilerletmek için neural network kullanma fikri on yıldan fazladır araştırılmaktadır. 1985 'te Hopfield ve Tank 'in seminer makaleleri Gezgin Satıcı Probleminin (GSP) Hopfield ağı ile çözülebileceğini göstermiştir. Henüz birkaç terim ve parametre içeren bir enerji fonksiyonunun minimizasyonunu gerektiren bu teknik, sık-sık GSP için mümkün olmayan çözümlerin olduğunu göstermiştir. Bu on yıldaki hızlı gelişme için, araştırmacılar ya enerji fonksiyonunu değiştirmeyi veya ilişkili çok parametreyi optimal olarak düzenlemeyi denediler. Dolayısıyla, ağın çakıştığı bu çözüm GSP için mümkün olabilir çözüm hazırlayabilecektir. Sonrakiler, nihai çözümü mümkün kılabilecek sınırlara Hopfield ağını sınırlamak için düzenlemeye giderler. Bu başarıya karşın, GSP' nin çözümü için Hopfield ağı fikri yeniden ortaya çıkmamıştır.

Bu noktada, GSP' nin çözümü için Hopfield ağının uygunluğunu sunan araştırmacılar ile optimum bulma tekniği olarak Hopfield ağının gücünü tanımlamaya çalışan araştırmacılar arasındaki farkı vurgulamak faydalı olacaktır. Araştırmacıların büyük bir çoğunluğu on yıldan fazladır birinci konuya yoğunlaşmışlardır. En son sonuçlar geometrik GSP hariç, Hopfield ağının geleneksel teknikler ile karşılaştırılamayacağını göstermiştir. GSP' de Hopfield ağının performansının düşük olmasının diğer bir açıklaması, bu modelin yeteri derecede özgürlük içermemesidir. Elastik-ağ yöntemi gibi GSP için değişebilen alan yaklaşımı bu konuyu çözmek için düşünülmüştür.

GSP' de Hopfield ağının klasik arama işlemi, diğer tekniklerle karşılaştırılınca performansının düşük olması, çok basit düşünce olmasıyla açıklanabilirliği tartışılır. Hopfield ve Tank ' ın GSP' nin bir tamsayı quadratik formülünü kullanma olayının anlamı, araştırma topluluğu işlemleri genellikle bir tamsayı lineer programlama

formülünü kullanırken gözden çok kaçırılmıştır. Hopfield ağı, tanımsız quadratik formdaki GSP sonuçlarından oluşan, çok sayıda lokal minimumlar alan bir nesne fonksiyonunu minimize etmeye çalışır. Rasgele GSP için nesne fonksiyonu Euclidean GSP' den çok daha fazla lokal minimumlar içerir. Hopfield ağı orijinal şeklinde sıkı bir iniş tekniğidir ve buna rağmen bu alt optimum çözümlerden birinde yakalanmış olacaktır. O zaman Hopfield ağı sonuçları, araştırmacı işlemleri ve sık kullanılan bir lineer formülden gelen çözüm kalitelerini sunmak için düzenlenmiş en iyi olan yaklaşım ve kesin çözüm teknikleri ile karşılaştırma, başlangıçta eşit olmayan bir yarışma gibi görülüyor. Sadece çözüm kalitesinin terimlerinde lineer formülünü kullanan diğer optimum bulma tekniklerinden daha kötü işleyeceği beklenen GSP' nin bu formülü ile bir Hopfield ağı vurgulanması geçerli bir sonuçtur. Formülsüz aralarındaki benzerlik sabittir.

Kesinlikle standart Hopfield ağı için enerji fonksiyonunda lokal minimumlardan çıkışa izin veren bir çok düzeltme önerilir. Bu yaklaşım bazı seviyelerde sonuçları ilerletebilir. Fakat nesne fonksiyonunun lokal minimumlarından çıkışın gerekli olmadığı olaylar ile sınırlıdır. Bu (mean-field annealing) gibi alternatif HTT yaklaşımları enerji fonksiyonuna yükselir. Fakat nesne fonksiyonu tek ve genellikle her zaman etkili değillerdir. Hopfield ağı için, her zaman çözümlerin gerçekleşebilmesi sağlanırken nesne fonksiyonunun lokal minimumlarından çıkışa izin veren etkili bir HTT düzeltmesi önerilebilir. Pratikte optimum bulma problemlerinin çoğunda Hopfield ağı ve HTT ile düzenlemesi en iyi olan tekniklerin performansına yaklaştığı gösterilmiştir.

GSP' de Hopfield ağının zayıf sonuçları, ilgilenenleri ileride endüstride Hopfield ağının optimizaston problemlerine uygulamaktan alıkoyabilir. Eğer HTT yetenekleri Hopfield ağına eklenirse, sonuçlar IDT gibi diğer HTT yaklaşımlarıyla karşılaştırılabilir. Tanımlı formülleri kullanan alternatif teknikler ile bu karşılaştırmanın yapılmış olması önemlidir. Dolayısıyla performansı etkileyebilecek diğer bütün faktörler elenebilir. Bu soru eğer Hopfield ağlarının optimum bulma teknikleri gibi herhangi bir avantajı varsa tanımlamaya yardım edebilecektir.

3.3. GSP' nin Bağıntıları

Tezde ele alınan konu, her bir şehir e bir defa uğramak üzere N şehirden oluşan bir küme boyunca en kısa kapalı çevrim yolu tanımlanması gereken GSP' nin bilinen şeklidir. **N={1,2,....N}** gezgin satıcının uğrayacağı şehirler kümesidir ve i ve j şehirler, arasındaki mesafe $D_{i,j}$ ile verilirse.

$$X_{i,j=}\begin{cases} 1 \text{ Eğer } i. \text{ Şehiri } j. \text{ Şehir izliyorsa} \\ 0 \text{ Aksi durum} \end{cases} \qquad \textbf{(3.3.)}$$

fonksiyonu ile $X_{i,j}$ değişkenine bir mantıksal karar tanımlarsak, araştırmacılar tarafından aşağıdaki tamsayı lineer programlama problemi GSP' yi ifade eder.

(Lineer Formül ile-GSP) minimize

$$\sum_{i=1}^{N}\sum_{j=1}^{N} X_{i,j} D_{i,j} \qquad \textbf{(3.4.)}$$

$$\sum_{i=1}^{N} X_{i,j}=1 \qquad \forall\ j \in N \qquad \textbf{(3.5.)}$$

$$\sum_{j=1}^{N} X_{i,j}=1 \qquad \forall\ i \in N \qquad \textbf{(3.6.)}$$

$$\sum_{i\in S}\sum_{i\in \bar{S}} X_{i,j} >= 1 \qquad \forall\ S \subset N,\ \ \bar{S}= N \setminus S \qquad \textbf{(3.7.)}$$

$$X_{i,j} \in \{0,1\}, \qquad \forall\ i,j \in N$$

Formül (3.7.) teki sınırlandırmalar sonucu oluşan küme için değil ise bu lineer tamsayı programlama formülü standart ödev problemi olacaktır. Bunlar alt yol eleme sınırlandırmalarıdır ve bu sınırlandırmaların olmaması, problem boyutunun büyümesi ve kontrol edilemeyen formüllerin tanımlanmasını gerektirir. Hemen-hemen GSP için literatürde kayıtlı başarılı yaklaşımların ve kesin yaklaşımların hepsi bu lineer formülü temel almışlardır. Bu Hopfield ve Tank ın kullandığı ve quadratik formülden farklıdır. Eğer değişkenler yeniden

$$X_{i,j} = \begin{cases} 1 \text{ Eğer } i \text{ Şehiri } j. \text{ Pozisyonda ise} \\ 0 \text{ Aksi durum} \end{cases} \qquad (3.8.)$$

şeklinde tanımlanırsa alt yol eleme sınırlandırmaları gerekmez. Maalesef bu yaklaşım aşağıdaki gibi quadratik nesne fonksiyonunu gerekli kılar.

(Quadratik Polinomlar ile-GSP) minimize

$$\sum_{i=1}^{N} \sum_{k=1}^{N} \sum_{j=1}^{N} X_{i,j} D_{i,k} \left(X_{k,j+1} + X_{k,j-1}\right) \qquad (3.9.)$$

$$\sum_{i=1}^{N} X_{i,j}=1 \qquad \forall\ j \in N \qquad (3.10.)$$

$$\sum_{j=1}^{N} X_{i,j}=1 \qquad \forall\ i \in N \qquad (3.11.)$$

$$X_{i,j} \in \{0,1\}, \qquad \forall\ i,j \in N \qquad (3.12.)$$

Açık olarak, bu formül daha az sayıda sınırlandırmalara sahiptir. Fakat nesne fonksiyonunun quadratik yapısının manası sadece bir genel minimum içeren lineer fonksiyon olacağı yerde, nesne fonksiyonunun bir çok lokal minimumları olacaktır. Yine açık olarak sınırlandırmalar kümesindeki karmaşıklık anlaşılması çok güç nesne fonksiyonlarına dönüşür.

Şüphesiz Hopfield ağına en uygunu, lineer formülün karışık sınırlandırmalar kümesi yerine, herhangi bir uygulanamaz standart rota yapıncaya kadar quadratik formülü seçmektir. Bu duruma rağmen standart Hopfield ağı dik bir iniş tekniğidir ve enerji fonksiyonunun bir çok lokal minimumlarından birinde yakalanacaktır. Her ne kadar sonuç mümkün ise kullanılan belirli bir lineer formülle karşılaştırılamaz. GSP için kullanılan standart Hopfield ağı sonuçları ile kesin lineer programlama çözümü veya Lin' in ***k***' lı optimum yaklaşımı gibi var olan teknikler karşılaştırılınca bu sık-sık görülür. Seçilen formülün ikincimi yoksa ilk mi olup olmadığı kesin olarak belirlenemez. Bu soru sadece Hopfield ağı sonuçları ile diğer iniş tekniklerinden quadratik formülde işletilenler karşılaştırılınca cevaplanabilir. Benzer olarak, Eğer lokal minimumlardan Hopfield ağını kurtarmak için mekanizmalar birleşmez ise bu sonuçlar diğer benzer HTT , IDT gibi yaklaşımlar ile karşılaştırılabilir, ayrıca quadratik formüllerinde işletilmelidir. Formülden çıkarılabilecek mesafeler gibi diğer faktörler , tekniğin doğru etkisinin görülebilmesi için tek yoldur.

Son yapılan çalışmalar, optimum bulma tekniği ve HTT olarak Hopfield ağının gerçekten değerinin olmasına rağmen GSP için iyileştirme ve sınırlama algoritmaları olmadan yalnız başına pek iyi bir sonuç vermeyeceğini gösteriyor. HTT düzenlemesi, çözümün her zaman mümkün olabilirliği sağlandığı sürece, enerji fonksiyonundan oldukça çok, nesne fonksiyonunun lokal minimumlarından kurtuluşuna izin verir. İniş ve HTT Hopfield ağlarının ikisinin de birbirinden ayrı olarak, dik iniş ve IDT yaklaşımı ile karşılaştırmalarda, Hopfield ağı yaklaşımının aynı sınıf yaklaşımlardan daha iyi kalitede çözümler belirlediği gösterilmiştir. HTT düzenlemesi, yerleştirilmiş bir çok pratik optimum bulma problemlerinin genel minimumunu bulabilir. Bu problemler bir kuruluşun sahip olduğu değişik araba modellerini sıralama, posta dağıtım ağı merkezi faliyet bölgesi ve gezici radyo istasyonu haberleşme ağında kanal görevi gibi uygulamaları olabilir. Bütün bu problemler genelleştirilmiş quadratik polinom

ödevi problemleri gibi formülleştirilmiştir ve onların nesne fonksiyonları bir çok lokal minimum içeren tamamen belirsiz quadratik formlardır. Diğer araştırmalar göstermiştir ki **knapsack** problemi gibi bir lineer nense fonksiyonu ile problemlerde Hopfield ağı, lineer programlama ve yaklaşım tekniklerin performansını iyi yakalamıştır.

Sonuçlar genel olarak lineer formül temelli yaklaşım ve kesin çözümler ile karşılaştırılıncaya kadar Hopfield ağının bir iniş tekniği olarak etkisiz olup olmadığını kesin olarak belirleyemeyiz. Nöral ağ topluluğu GSP' nin lineer programlama formülünü (standart Hopfield ağı taslağına alt yol eleme sınırlandırmaları hemen birleştirilemez.) kullanışlı yapmadığı sürece, GSP' nin quadratik formülüne dayandırılan yaklaşım, iniş yaklaşımları ile tekniğin etkilerinin sunum terimlerinde sadece faydalı bir karşılaştırma olur. Bu yolla bütün diğer faktörler karşılaştırmadan çıkarılır.

Hopfield ağının, mümkün olabilirliğini sağlayan uygun bir enerji fonksiyonu ile, pratik optimum bulma problemlerinin geniş bir kısmında iniş yaklaşımlarının performansını yakalayabilen uygulaması henüz yapılmamıştır. Ayrıca HTT ve lokal minimumlardan kurtuluşa izin veren, içindeki dinamik yapı ile Hopfield ağı, etkili bir şekilde IDT gibi HTT yaklaşımlarına yakın çözümler verebilir. Ancak daha iyi bir çözüm vermesi pek mümkün görünmüyor.

3.4. GSP' nin Çözümünde Kullanılan Diğer Yöntemler

Diğer yöntemlere bir göz attığımız zaman, optimum güzergahı bulma problemi NP-complete dir. En kesin ana algoritma Dallan ve Sınırla ve Dallan ve Kes dir. Bu yöntemler son on yılda sürekli uygulanmış ve dolayısıyla bu mesafeler binlerce {N } şehir için optimum olarak çözülmüştür.

GSP için bir çok yaklaşım yöntemi uygulanmıştır. Bunların arasında yerel arama, IDT , Genetik Algoritma, ve Nöral ağ, yaklaşımları direkt tur yapısındadır. Lin ve Kernighan' (L-K) a kadar en iyi çözüm yerel arama olarak tanımlanıyordu. L-K ve IDT nın ikisi de kullanarak yapılıncaya kadar, IDT çoğu zaman L-K dan daha yavaş olmasına rağmen iyi turlar vermiştir. Alt bölümlerde konuyla ilgili detaylar verilmiştir.

3.4.1. Lin-Kernighan Yerel Arama Yöntemi

L-K değişken derinlikte bir yerel aramadır. Bütün mümkün olan çözümler kümesinde bir komşuluk yapısı notasyonu ile herhangi biri başlar. **T** turunun en çok **k** köşesini değiştirerek belirlenebilen bütün turlar **T** turunun komşusu olarak tanımlanır. Rasgele bir T_1 turu ile başlayarak ve birbirini izleyen T_1, T_2.. turlarını yapılandırarak **k'** lı optimum lokal tur aranabilir. Her bir tur **k** değişiklik ile ondan önceki bir tura bağlanabilir. **k** bağları silerek ve kaybedilen sonları tekrar bağlayarak yeni bir tur oluşturulabilir. Turun uzunluğunu azaltmak için **k** değişiklik yapmamız gerekir. Eğer bu **k** değişik durumdan hiç biri yolu daha fazla kısaltamıyorsa işlem durdurulur ve **k'** lı optimum bulunmuş olur. Lin **k=2** ve **k=3** durumlarından birinin oldukça iyi turları olması gerektiği üzerine çalışmış ve tanıtmıştır. Bu şekilde genel optimum turu bulabilmek için bir çok rasgele başlangıç önerilir. Daha sonra Lin ve Kernighan **k** nın değişken olmasının daha iyi olduğunun farkına varmışlardır. Özellikle onların algoritması **L-K** bir Depth-first Search (derinlemesine ilki arama) ile izlenen bir Bredth-first Search (genişlemesine ilki arama) tür. Daha önce test edilmiş bütün yaklaşımlara karşı performansı ölçmek için biri, çok sayıda mesafelerde yaklaşım algoritmalarını çalıştırarak belirlenmiş uzunlukların değişimini ölçebilir. Standart mesafeler:

1. Kare içinde rasgele dağılmış şehirler ile mesafeler,
2. Rasgele mesafe matrisleri $D_{i,j}$ ile mesafeler ve
3. Dallan ve Kes yöntemiyle optimum çözülmüş lokal ulaşılabilen mesafelerdir.

L-K nın bu problemlerde performansı çalışılmıştır. N büyüdükçe **L-K** ile bulunan turların uzunluklarının dağılımı (minimuma normalize edilmiş) $N^{-1/2}$ gibi bir genişlik azalmasıyla küresel (Gaussion) bir fonksiyon çizer. **L-K** nın bu performansı özellikle bu dağılımların ortalama değeriyle tanımlanır. 1. Kategorideki mesafeler için **L-K** nın ortalaması minimumun yaklaşık % 1.5 kadar üstündedir. 2. Kategorideki mesafeler için büyük N için teorik olarak minimum uzunluk olarak bilinir. **L-K** dışındaki tüm yerel arama yöntemleri bu kategori için oldukça zayıftırlar. Ortalama değerin büyümesi N in artmasıyla olur.

3.4.2. Isı Düşüşü Taklidi

Isı Düşüşü Taklidi başarılı bir şekilde GSP' ye uygulanmıştır. Biri turları T1, T2,.. ardışıl yapılandırarak başlatılıyor. Bu zincirin her bir adımı bir k' lı değişiklik (bir komşu tura taşıma) yaparak belirleniyor. Genellikle k 2 veya 3 tür. Turların ardışıl yapılanması, gürültü içeren yerel aramaların değiştirilmesi gibi görülebilir. GSP için IDT, Lin-Kernighan dan oldukça daha yavaştır. Fakat uzun zaman çalıştırılırsa çözümün kalitesini yavaş-yavaş arttırma avantajı vardır. Sonunda **L-K** ile karşılaştırılabilir veya daha iyi sonuçlara ulaşabilir. IDT' nin en basit versiyonunda T_n den T_{n+1} e gidince rasgele 2 değişik yer değiştirmedir (**2'** li değişiklik). Bir çok çalışmada çok daha karmaşık hareketler düşünülür ve ayrıca **2'** li değişmeyi seçmenin yolu da rasgele değildir. Son tur (derece 0 a gelince) sadece **2'** li optimum olmayı garanti eder. Mükemmel sonuçlara ulaşmak için **2** ve **3'** li değişiklik ikisini de içererek son tur **3'** lü optimum olur.

3.4.3. Problemi Çizgelere Ayırma

V_i i=1,..N vektörleri ve V_i ve V_j vektörlerini birleştiren $E_{i,j}$ köşelerinin toplamından oluşan bir (un-oriented) yönlendirilmemiş çizge tavsiye ediliyor. Çizge bölme problemi köşelerin ("Cut") sayısı minimum edilecek şekilde belirli büyüklükteki A ve B alt iki kümesinin içinde **V** nin bir kısmını bulmaktan ibarettir. Eğer son nokta farklı bir alt kümeye ait ise $E_{i,j}$ köşesi kesilir. Daha önce bir çalışmada A ve B nin eşit büyüklükte olduğu ve N in çift olduğu çizgelere ayırma probleminin "standart" formülüne yoğunlaşılmıştır. Buna bi-partioning ikili bölünmüş çizge (İBÇ) de deniyor. Bu eşit büyüklükte olmayanlar için bir sınırlama getirmez.

İBÇ' nin pratikte büyük önemi vardır. VLSI chip yüzeyi ve programı parçalara ayırma için kullanılan, hücre yerleştirme probleminin büyük çoğunluğuna uygulanabilir. Çizge bölmeleme işleminin bazı kısımları veya genelleştirilmiş hali optimum bulma problemlerinde kullanılır.

İBÇ bir NP-complete dir. GSP için kesin çözüm yöntemleri daha az geliştirilmiştir. Dolayısıyla optimum olarak kabul edilen için birkaç mesafe vardır. Yaklaşımlar için durum GSP' ye oldukça benzetilebilir. İki en iyi yaklaşım , değişken

derinlikli arama yaklaşımları Kernighan ve Lin e uygun **(K-L)** ve IDT dir. Bu iki algoritma çözümün kalitesi terimlerinde karşılaştırılabilir fakat; IDT daha yavaştır. Buna rağmen, bu algoritmalar bir çok pratik mesafeler için dengeli yüklenme ve yüzey problemlerinde olduğu gibi çok iyi değildirler. Problemde özel gelişmeler bulabilmek için çok fazla efor sarf edilmesi gerekir.

3.4.4. Kernighan-Lin Yerel Arama ve Genişletilmeleri

Lin-Kernighan da varılan sonuçtaki gibi bir **k'** lı değişiklik şekli tanıtmak mümkündür. B nin bir elemanına karşı A nın bir elemanını yeniden değiştirmek için biri 1'lı değişiklik çağırır. 1' li optimum vasat bir algoritmadır ve **k'** lı optimum dan daha kötü sonuca gider; hem çok zor ve yavaş işler hem de çok gelişmeler sağlayamaz. Kernighan ve Lin değişken bir **k'** lı değişiklik algoritması önerirler. Bu oldukça hızlı olurken diğerlerinden yani 1'li optimum ve 2'li optimumdan çok daha etkilidir. Onların algoritmaları aslında A ve B kümelerini sürükleyen aç bir **Tabu** (1-li değişiklik) dir: Her bir adımda, değişimi en uygun olabilecek eleman çiftini değiştirir, sürükleme boyunca, eğer bir eleman zaten değişmiş ise, sonraki değişim için, bu sürükleme boyunca, daha uzun olamaz. Eğer parçanın büyüklüğü sürükleme boyunca azalmazsa bölme **L-K** optimum olarak tanımlanır. Eğer azalırsa, sürükleme boyunca en küçük parça ile bir bölümünü alır ve bunu diğer bir sürükleme için başlangıç olarak kullanır. Parça büyüklüğü sürükleme sayısını azaltan etkendir ve bunlardan bir tanesi hızlı bir şekilde optimum parçaya ulaşır.

L-K performansı çözüm çizge parçalarının artmasına bağlıdır. G(N,P) Çizgelerin doğal parçalar kümesi, N vektörün rasgele bir çizgesinin parçalarının kümesidir. Her bir çift (V_i , V_j) p olasılık ile birleştirilir. Eğer p N den bağımsız ise büyük N değerleri için minimum parçanın boyunun ortalaması formül 3.13. ile bulunabilir.

$$<\text{minParça}> = pN^2/4 - UN^{3/2}\,[p(1-p)]^{1/2}/2 \qquad U=0.38.. \qquad \textbf{(3.13.)}$$

İlk terim basit olarak rasgele parçadan gelmiştir; ikinci terim optimum bulma boyunca gelişmeyi temsil eder. Seyrek çizge için (p=O(1/N)) d=p(N-1) ortalama derecesi ile minimum parça büyüklüğü N ile orantılı olarak formül 3.14.' de verilmiştir.

<minParça>=C(d)N **(3.14.)**

Algoritmaların çoğu, sık çizgelerde iyi işler ve seyrek çizgelerde onlardan daha kötü işler. Bu son parçalar topluluğunda, performansı diğer algoritmalar ile karşılaştırılabilir. Her bir köşede, parça büyüklüğünü **L-K'** nin $C_{k-l}(5)=0.49 \pm 0.01$ değerini verdiğini görebiliriz.

Dengeli yükleme uygulamalarında olduğu gibi, dengeli bir yapıya sahip çizgeler için **L-K** ve **IDT** çok daha zayıf olur. Bu tip çizgeler için **L-K** yi geliştirmeye çalışmak fazla zor olmaz. Sıkıştırma olarak adlandırılan bir yöntem, yakın vektör çiftiyle birleştirerek, yarı büyüklükteki problemlerde **L-K** yi çalıştırır. Daha sonra çözüm, kendi kendisine **L-K** optimum olacak şekilde, yeni boş bir çözüm olamaması için sıkıştırılmamıştır. Bu işlem tekrarlanarak yapılabilirse hiyerarşik veya çoklu skala algoritmasını verir. İkinci yaklaşım, rasgele başlama bölümlerinde daha iyisinde kalır. Genelde, kısmen başlangıç bölümleri için, bölen çizgiyi kullanmak önerilir ve **L-K** başlangıç için uygulanır. Sonuç algoritma **L-K-L** olarak adlandırılan geometrik çizgeler için daha basit olduğu sürece hiyerarşik sıkıştırma yaklaşımından daha iyi sonuçlar verir.

3.4.5. Çizge Üzerinden Isı Düşüşü Taklidi İyileştirmesi

İBÇ için IDT' de doğal seçim A ve B arasındaki vektör çiftini değiştirmedir. Diğer ihtimal aynı anda bir elemanı karşılıklı taşımaktır. Fakat asla dengesizliğin 1 veya **k** elemandan daha büyük olmasına müsaade edilmez. Bir tanesi herhangi bir dengesizliği kabul edebilir fakat; dengesizliğin büyümesi gibi bir sonuca asla gitmez.

Bundan dolayı dengesizliğin oranı küçük kalır. IDT ile etkili olmayan eleman çifti değiştirilmeye çalışılır. Bu herhangi bir dengesizliğe daha kolay ve daha hızlı izin verir ve genellikle bu dengesizlikte quadratik olan bir sonuç ile karşılaşılır. Bu zor

sınırlamanın daha basit olması, sisteme lokal minimumlardan kurtulmak için yeni rotalar tanımlama zorunluğunu getirir. Ayrıca N(N-1)/2 den N' e komşulukların sayısını azaltma avantajı vardır. IDT ve **L-K** yi karşılaştırma yapılmıştır. Rasgele çizgeler için IDT, **L-K'** dan daha iyidir. Rasgele çizgeler için ortalama derece d=5 C_{IDT}/C_{K-L} =0.918 bulunabilir. Fakat geometrik çizgeler için kesinlikle sonuç daha kötüdür.

3.4.6. Genetik Algoritmalar, Hızlı Tırmanma Tekniği ve Lokal Yapı

Optimum bulma problemlerinin çözümünde iki farklı yaklaşım, genetik algoritmalar (GA) ve HTT olarak gösterilmiştir. GA nın HTT i işletebileceği, birlikte kullanılabileceği bölgelerin olup olmayacağı araştırılmıştır. Bu işlemde HTT ile iddia edilen bazı eksikliklerin olabileceği durumda GA nın daha iyi çözüm sağlayıp sağlamayacağı denenmiştir.

Optimum bulma problemlerinde amaç verilen fonksiyonun maksimum veya minimum değerini bulmaktır. Bu problemleri çözmek için iki farklı yaklaşım genetik algoritmalar GA ve HTT dir. HTT algoritmalarının çok değişik şekilleri vardır; fakat hepsindeki işlem aynıdır. Birincisi arama uzayında bir başlangıç varsayımı rasgele seçilir. Bu varsayımından daha yüksek değer veren yeni bir başlangıç seçilir. Eğer bütün başlangıçlar mevcut varsayımdan daha düşük değer veriyorlarsa bir zirveye ulaşılmış demektir ve işlem durur.

Genetik algoritmalarda biyolojik evrim taklit edilir. GA birçok elemandan oluşan toplulukta aralıksız çalışırlar. Seçme, çaprazlama ve mutasyon varsayım topluluğunu geliştirir.

GA nın HTT şemasını işletebileceği bölgelerin olup olmadığını araştırılmıştır. Sıralamada karşılaştırmanın objektif olması için, iki algoritmanın da iyi başlangıçları denenmiştir. GA için (Mitchell, 1997) rulet tekerleği seçimi kullanılmış ve verilen genetik algoritmanın basit şekli kullanılmıştır. GA nın bir çok değişik şekilleri var ve aralarındaki farklar sınırlıdırlar. HTT' dede bir çok değişik şekiller vardır. En büyük gelişmeyi veren adım için arama yapmaksızın mevcut varsayımı geliştiren herhangi bir adımın alındığı (Hampson & Kibler, 1995) Greedy HTT algoritması seçilmiştir.

Bundan dolayı, her bir adımda yeni başlangıçların başlaması için, aramada yön değiştirerek herhangi bir özel tırmanma yönü için referans verilmemiştir.

3.4.6.1. Hızlı Tırmanma Tekniği ' nin Değerlendirilmesi

Optimum bulma problemlerini çözmek için HTT' nin kullanımında iki ana probleme işaret edilmiştir (Goldberg, 1989). Birincisi yerel maksimum ile bir problem vardır. HTT kullanılarak bir maksimum bulunduğunda, onun genel maksimum olduğuna ait hiçbir garanti yoktur. İkinci olarak HTT mevcut türevlerde veya türevlerin sayısal yaklaşımlarında, muhtemelen tekrar yanılır. GA, bu iki problem ile muhtemelen başa çıkabilecek, optimum bulma problemlerinin çözümü için alternatif bir yaklaşım olarak kullanılmaktadır.

HTT algoritmasının her defasında rasgele başlamasına müsaade ederek zirveye ulaşması, genel maksimum bulma ihtimalini yükseltir. GA, topluluğun evrimi için uygunluk fonksiyonunu kullandırır. HTT algoritmaları da aynı uygunluk fonksiyonunu tırmanma yönüne karar vermek için çok iyi kullanabilir. Diğer bir deyişle GA' nın kullanılabileceği her bir bölgede HTT de ayrıca kullanılabilir.

Bundan dolayı, bu rasgele başlatılacak HTT' nin GA dan daha iyi çalışabileceğine sebeptir. Ana konu genetik algoritmanın bir çevirimi (seçme, çaprazlama, mutasyon) süresince, HTT algoritmasının bir çok tırmanma işletmek için zamanı olacaktır. Çoklu maksimumun bulunduğu fonksiyonda, oldukça az adımda maksimumlardan birine ulaşacaktır ve rasgele başlangıcı işletebilecektir. Dolayısıyla GA topluluğu yavaş geliştirirken, HTT algoritması ile bir çok rasgele başlangıç yapılabilir.

Bu konuda sonuçları test etmek için sırasıyla, GA ve HTT algoritmaları 3.15. formülündeki fonksiyonda çalıştırılmış, GA ve HTT' nin her birinde aramayı ne zaman bitirmek gerektiğini belirlemek için dört farklı başlangıç kullanılmıştır. Fakat maksimumlardan biri ve her bir başlangıç için iki algoritmada 100 defa çalıştırılmıştır. Sadece 60 saniye içinde biten çalışmalar başarılı kabul edilip, rapor edilmiştir. GA için topluluk büyüklüğü 100, mutasyon oranı 0.1 ve tolerans 0.65 kullanılmıştır.

$$F(x,y)=Sin(x)/x - Cos(y)/y \quad (3.15.)$$

Tablo 3.4.6.2.1. de görüldüğü gibi, HTT algoritmasının performansı dikkate değer bir şekilde değişmemiştir. Başlangıçların sayısı tahmin edildiği gibi, en yüksek zirve tarafından tutulan başlangıç uzayının parçasını yansıtır. Bir kere uzay parçasında, bir başlangıç seçilir, maksimuma oldukça az sayıda adımla ulaşılır. Üstelik, eğer bir eşiğe ulaşılırsa, daha yüksek bir eşiğe ulaşmak için sadece çok az sayıda ek adıma ihtiyaç olacaktır. Buna rağmen GA için, eşik yükseldikçe performans düşer. GA nın parametrelerinin değerlerinin farklı seçilmesi daha iyi sonuçlar verebileceği tartışılabilir fakat; bu muhtemelen sabit performans desteğinden fazla olmayacaktır.

HTT için yerel maksimumların problem olduğu doğrudur fakat; bu bilinen herhangi bir optimum bulma tekniği içindir de. Tablo 3.1.' deki sonuçlarda gösterildiği gibi, GA probleme, rasgele başlatılan HTT den daha iyi çözümler bulamıyor. Sezgisel olarak, yüksek uygunluğa sahip birey ebeveyn olarak seçilmesine rağmen GA, yerel maksimumlara sıkışmak için daha yüksek bir eğilime sahiptir. Bireylerin bazı iyi özellikleri döllerinde toplanabilir. Fakat yerel maksimumlara ilerleyen bir yapının, fonksiyonun genel maksimumunu veren yapının bir parçası olması gerekmez. Eğer oldukça çok yapı yerel maksimuma giderse, iyi topluluktan uzaklaşılır, mutasyon çıkış için tek yoldur. Daha sonrası, rasgele başlatılan HTT' den uzak görüş değildir. Fakat HTT ' de bu rasgele başlangıçlar sadece çok ihtiyaç olduğunda olur. GA da mutasyon herhangi bir zamanda olur ve genel maksimuma giden topluluğu elde edip etmeyeceği garanti değildir.

3.4.6.2. GA ve Lokal Yapı

Bölüm 3.4.6.1.' de bahsedildiği gibi, yerel maksimuma giden bir yapının fonksiyonun, genel maksimumuna giden bir yapının bir parçası olması gerekmemektedir. Fakat bazı bölgelerde bu durum mümkün olabilir ve eğer çaprazlama operatörü bunun avantajını kullanarak programlanabilirse, GA HTT kadar, hatta ondan daha iyi olabilir. Eğer

fonksiyon büyük platolara, HTT şeması için muhtemelen en büyük problem, sahip ise, GA daha büyük avantaja sahip olur.

Tablo 3.1. : platolar ve yerel yapı ile fonksiyonda GA ve HTT (Youness H. I., 1998).

	Genetik Algoritma			*Hızlı Tırmanma Tekniği (Hill-Climbing)*			
Boyut	**Başarı**	**Nesil**	**CPU sec.**	**Başarı**	**Restarts**	**Climbs/restarts**	**CPU sec.**
2	100	1.5	0.022	100	117	6.4	0.568
3	100	3.3	0.043	95	1144	8.9	10.290
5	100	7.9	0.103	10	6399	9.0	22.951
10	100	29.3	0.468		-	-	-

Bazı yerel yapıların olduğunu gösteren topluluk için bir yüksek uygunluk değerine dikkat edin, ayrıca o genel maksimumu veren bir topluluğun parçası olduğuna dikkat edilmelidir. İki boyutlu durumda, döl oluşturmak için her bir ebeveynden bir koordinat alan çaprazlama operatörünü kullanırız, maksimuma ulaşmak için dar koridorun her birinden ebeveynleri eşleştirmeye ihtiyacımız vardır. Bu varsayım uzayından düzensiz olarak alınmış 20 bireyden oluşan başlangıç topluluğuna ihtiyaç duyar. Boyutlarda bir büyüme ile, performansta lineer bir düşüş beklenebilir. HTT yaklaşımı için, maksimuma götürmek için fonksiyon sadece büyük platolardan oluştuğunda, o fonksiyonun maksimuma sahip olduğu, dar çözüm uzayına düşmek üzere olan başlangıç noktasına ihtiyaç duyabilir.

3.4.7. Dallan ve Sınırla

Dallan ve Sınırla (Branch and Baound) genel bir arama yöntemidir. Bir F(x) fonksiyonunu minimize etmek istendiğinde; çözümün mümkün olmadığı bölgelerde, x in sıkıştırıldığı, dallan ve sınırla yöntemini uygulanabilir. Ayrıca optimum bulma

problemlerinde daha düşük bir sıralama gerektiğinde ve problemi mümkün olabilen bölgeye bölerek daha küçük alt programlara ayırmak için kullanılabilir. Bunun yanında, mümkün olabilecek çözümlerde, en azından pratik amaçlar için bazı mesafelerde, mümkün olabilecek bazı çözüm kümeleri için daha fazla sınırlama mümkün olmalıdır.

Bütün mümkün olabilen bölgeler ile orijinal problemde tahminle işleme başlanır. Bu kök problem olarak adlandırılır. Alttan sınırlama ve üstten sınırlama işlemleri, kök problemine uygulanır. Eğer sınırlama uyarsa, optimal çözüm bulunmuş olur ve işlem kesilir. Aksi durumda mümkün olabilecek bölge iki veya daha fazla bölgeye bölünür. Her bir sıkıştırılmış alt bölge ile birlikte, bütün mümkün olabilecek bölgeyi kapsarlar. İdealde bu alt problem bölümleri mümkün olabilecek bölgedir. Bu alt problemler kök arama noktasının çocukları olur. Algoritma yinelenerek alt problemlere uygulanır, alt problemlerin bir ağacı oluşturulur. Eğer bir alt problemde bir optimum çözüm bulunursa, bu bütün problemin mümkün olabilecek bir çözümüdür. Fakat gerekli olan global optimum değildir. O mümkün olduğundan ağacın geri kalan kısmını budamak için kullanılabilir. Eğer Alttan sınırlama, en iyi bilinen çözümü zorlayan bir nokta için ise, nokta tarafından hazırlanan mümkün olabilecek bölgenin alt boşluğunda global optimum çözüm olamaz. Bundan dolayı, nokta tahmin edilen çözümden çıkarılır. Arama bütün noktalar çözülünceye veya budanıncaya kadar veya bazı özel başlangıçlar en iyi çözümü bulunca ve alt sınırlama, çözülmemiş bütün alt problemler arasında bulununcaya kadar sürer.

3.4.8. Çok İhtimalli Optimum Bulmada, Genetik Aramanın Etkileri Üstüne Kihong Park ve Bob Carter' ın çalışmaları

Çok ihtimalli optimum bulmaların içeriğinde genetik algoritmaların etkileri araştırılmıştır. Kısmen, genetik aramanın bir operatörü olarak çaprazlamanın etkisi ele alınmıştır. Zor ve mümkün olmayan büyüklükteki problemler için, çaprazlama ile arama, mutasyon ve seçme ile birleştirilerek çalıştırıldığında, yalnız seçme ile birleştirilerek çalıştırıldığında veya mutasyon ve seçmeden oluşan arama işlemi karşılaştırıldığında; murasyon ve seçme operatörlerinden oluşan aramanın daha etkili olduğu gösterilmiştir. Çaprazlama ile aramayı kolaylaştıran bir topluluğu bulmanın çok uzun zaman alabilmesi ve zor olması, onun marjinal faydasını da göz önüne alırsak,

genetik aramanın uygun topluluğuna karşı metropolis işlemi ve **IDT** genişlemesinden, daha kötü duruma düşürür. Pratikte bir çok problemde, kalıtsal olarak bulunan optimum yakın çözümler için, çok sayıda geçiş durumuna ihtiyaç duyar, genetik aramayı imkansız kılar, genişletilmiş durum uzayında iterasyonla sonuç topluluğunu hesaplamanın zaman almasına ve zorlaşmasına sebep olur.

Genel amaçlı optimum bulma işlemi gibi görülen genetik algoritma, uçak endüstrisinde, protein parçalamada, topluluk listeleme sınırlamasında, bir bölüm problem çözüm bölgelerine, artarak uygulanmaya başladı. Araştırmaların çokluğuna rağmen, kanaatler hala genetik aramanın kusursuz bir optimum bulma tekniği gibi yardımcı olmadığı yönünde. Sonuçların farklı alanlarda karşılaştırma zorluğuna rağmen, bu bölümde mesafe ve zaman açısından bazı karşılaştırmalar verilmiştir. Bunun yanında problem mesafeleri, kendi özel güç durumları olmaksızın seçilmiştir. Problemin büyüklüğü, tipik sonuçların genel bir raporunu sağlamak için oldukça küçük olabilir. Fakat; zaten genetik algoritmayı diğer tekniklerle ilişkili olarak kullanmanın faydası, çokta açık gösterilememektedir. Gerçek hayat problemlerine genetik aramanın uygulamasının artmasını sağlamak, onun gücünün sınırlarını göstermek, potansiyel faydalarının oluşmasını ve bunun için sarf edilecek çabayı, beklenen performansı algılamayı, ve bunların üstüne diğer optimum bulma teknikleriyle karşılaştırmayı zorunlu kılar.

Bu bölümde, genetik aramanın ve bir çizgede maksimum büyüklükteki grubu bulma problemi olan maksimum parça (**MAX-CLIQUE**) 'nin içeriğindeki elemanlarının performansları verilecektir. Maksimum parça, geçenlerde NT-hard olduğu ispatlanan yaklaşım problemi olarak bilinen bir NT-complete tir. Bu genelde, kalıtsal zorluğu olabilen yaklaşım çözümleri bulmaya denktir. Bu mümkün olmayan çözüm grubu ile kolay ve zor problem mesafelerinin ikisinde de, çaprazlama operatörünün etkisi, tekrar üretme ile arama işlemindeki çaprazlama operatörüyle ayrılmıştır. Bu *ÇMS* olarak not edilmiş (çaprazlama, mutasyon, seçme) tam arama işlemi ve kısmen bir versiyonu *ÇS* (çaprazlama, seçme)' nin performansını sadece mutasyon ve seçme versiyonu olan ***MS*** ' ye karşı karşılaştırarak yapılmıştır. Sonra nitel algılamada Metropolis işlemine (Uygun sıcaklıkta IDT) uygunluğuna bakılmıştır; çünkü ikisi de tam olarak zamanı homojen Markov zinciri (time-homogenous Markov chains) dir, yüksek uygun durum tercih edilir; fakat eski nesil ile aynı topluluğa geri

dönmeyi engelleyemez ve bir sonraki durum lokal bozma ile elde edilir. *ÇMS*, tam genetik arama da, ayrıca zamanı homojen Markov zinciri gibi tanımlanabilir fakat; dikkat edilebilecek fark bir sonraki duruma lokal olmayan anlamda **çaprazlama** ' dan geçerek ulaşılır.

Belirli bir büyüklüğün üstünde, mümkün olmayan problem mesafeleri için gösterilen, genetik aramanın performansını düşürerek, problem boyutunun artması gibi, hoş olmayan bir duruma düşürür. İki ana sebep örnek gösterilmiştir, birincisi yüksek hesaplama maliyetinin olması ve diğeri farklı problem mesafelerinin sınırına blok oluşturma (**building-block hipotesis, BBH**) hipotezinin uygulanabilirliğinin sınırlı olmasıdır. Bunu aşmak ve çalıştırılabilmek, GA nın nesillerinin sayılarına bir nesil eklemek, gelecekte genetik aramayı olağanüstü başarılı kılacaktır. Blok oluşturma hipotezi, eğer ikisi de iyi ise aykırı alt çözümleri uygun olarak karıştıran bir düzene sahip problemlerin bir sınıfını sınırlandırmaya, ayırmaya çalışır, daha sonra önemli ihtimaller ile daha iyi çözümler belirlenir. Bu, genetik aramanın, Blok oluşturma hipotezinin kolayca uygulanabildiği problemler için etkili bir arama olduğunu gösteren önemli bir olaydır, fakat bunlar sadece yüzeysel lineer olmayan tanımlama ile süper-pozisyon ilkesi olan problemler olarak vurgulanmalıdır ve bundan dolayı girişimin daha çok etkili olması her zaman sağlanmayabilir.

Problemlerin, gerçek yaşamdaki uygulamalarının zorluğuyla karşılaşılınca, uzun süren ve onların gerçekte hazırlanışının ne kadar zor olduğu sorusuna gelinir. Cevap, probleme çok bağlı olması ve memnun edici bir belirlemenin zor olması olabilir. Genetik aramanın yardımına uyum sağlayan olayı, kötü sonuçlanacak korkutucu bir durum ile karşılaşılacak bir durumun, eldeki problemden bağımsız olduğu gösterilebilir. Bu **BOH'** nin doğru olduğu, kolay olarak tanınan, optimuma yakın bir çözüm bulmanın kalıtsallıktan sağlanan kolay bir taslak olduğu problemler içindir ve diğer daha etkili algoritmalar, hesaplamanın yoğun olduğu GA ya tercih edilirler. **BOH'** nin kolayca uygulanamayacağı daha zor problemler için, kaynakların uygun sınırları içindeki arama uzayındaki durumların sayısını sınırlama ile sıkıntılı bir duruma dönüşürken, çaprazlama sadece başarının küçük, önemsiz bir kısmını sağlar. Burada, genetik aramada çaprazlama elemanlarının yardımı ve tanımlamada ki yoğunluk, ilk yaklaşımın sonuçlarını destekler. Bundan başka, genetik arama ile IDT' yi karşılaştırmamızda, aşağıdaki sıralandırılmış ilişkileri kolaylaştırır.

3.4.8.1. Çözüm Kalitesi

(Park, K., 1994) verilen karşılaştırma sonuçlarına göre:
Genetik arama(*ÇMS*) ≅ *MS* < Metropolis işlemi ≤ IDT dır. Zaman faktörü alınırsa Genetik arama(*ÇMS*) < *MS* < Metropolis işlemi ≅ IDT olur.

Çözümün kalitesi, "≅" ortalamada eşit ve "<" kesinlikle daha iyi şeklinde kullanılan, birkaç saat CPU zamanını sınırlayan bir kaynak ile esas alınmıştır. Kuşkusuz, sıralı ilişkiler kesin değildir, fakat sonuçların kalitesini belirlemeye elverecektir. Zaman faktörünü sunmak ile gereksiz, şekilciliği arttırmaksızın, dikkatli bir tanımlama verilebilir. Zaman faktörü bir adım veya nesil hesaplanması için problem büyüklüğü ***n*** ve topluluk büyüklüğü ***m*** nin bir fonksiyonu gibi ölçülür. ***M*** yi gerçekleştirmek , $A<B \Leftrightarrow Time_B(n) = ()(Time_A(n))$, $Time_A$ ve $Time_B$, A ve B nin zaman boyutunda uygun fonksiyonlarıdır. Bazı fonksiyonal ilişkilerden geçerek ***n*** nin bir alt toplamıdır. Burada ***B, A*** dan daha hızlıdır. Önceki sıralamayla birleştirse:
Genetik Arama < IDT elde edilir.

Çözümün kalitesi ve zaman faktörü, ikisi de açıktır. Çaprazlama ve seçme için strateji, ***CS***, zaman faktörü uyumu ile, ***ÇMS < CS < MS*** ye götürür ve çözümün kalitesine uyumu ile, ***CS < ÇMS*** ve ***CS < MS*** e götürür. Zaman faktörü ilişkileri kurmada zorluğa yol açacaktır.

Bu çalışmadaki veriler düzenlenerek bir sonraki bölümde algoritmik ve deneyimsel veriler verilmiştir. Bunu ***genetik arama < IDT*** olayını destekleyen ana bölüm izliyor. Çoğaltma, arama işleminde topluluk büyüklüğünün etkilerini tanımlama ve uzun zaman farklı algoritmalar düşüncesinin tanımlandığı bölümdür. Burada mevcut yöntemin açıklığı ve güçlülüğünü, sonuçların uygulanabilirliğinin zemininde, onun etkileri verilerek sonuçlandırılmıştır.

3.4.8.2. Hazırlama

Bir genetik algoritma, çaprazlama (*C*), mutasyon (*M*), ve seçme (*S*) üç operatörden oluşur. Topluluk (population) **H** ve bir sonraki adımda üretilen yeni topluluk **H'** dir. Eğer biz **T**=*MCS* ile gösterirsek, genetik aramanın bir nesili

$$H'=T(H) \qquad (3.16)$$

Olarak belirlenir.

$(T'(H))^{\infty}_{t=0}$ hareketi parametrelerin seçimi, problemin kodlanma planı ve sonuç fonksiyonunun kullanımını içeren, değişik yardımcı mekanizmalar gibi birkaç faktöre bağlıdır. Bölüm 3.4.8.3.' te deneyimlerle çalışılmış algoritmik hazırlık tanımlamalarının bir özeti verilmiştir.

3.4.8.3. Problemi kodlama

Topluluğun bir elemanı, bir çizge hazırlama, $x_i = 1$ eğer **i** inci nokta çizgede varsa şeklinde, **n** uzunluğunda ikili bir **x** katari gibi açık olarak kodlanmıştır. Bir başlangıç adımı, birlikte uygulanan bir grup nokta gibi köşe etiketlerine izin verme fikri az bir etkiyle tanımlanmıştır. Olumlu sonuçların raporlarının, özel mühendislik çizgelerine de genelde çaprazlamayı biraz arttırarak uygulanabileceği büyük bir ihtimaldir.

3.4.8.4. Çaprazlama

Denenmiş birkaç çaprazlama şeması, tek tip iki noktadan çok yapılı şemaya sıralar. Maksimum parça için, eğer çaprazlanmış büyük parçaları **x** ve **y** gibi iki elemanla kodlarsak, sonuç elemanını çok ince bir parça ile bırakma ihtimali olan iki nokta şemasından geçer. Bu konuya iki yol ile yaklaşılıyor. Biri, büyük parça olmayanları, topluluğun üyesi gibi bırakmaya ve onu sonuç fonksiyonuna, uygunluk değeri olarak ilişkilendirmek ve ikincisi, parçalamayı düzgün olarak koruyan bir şekilde, çaprazlama ile büyük parça olmayanların hepsine birlikte izin vermemek. Son

yaklaşımın, çözümleri, aralıklarla kolayca geliştirdiği görülmüştür ve mevcut sonuçlar bu yöntem ile ilişkili verilmiştir.

3.4.8.5. Sonuç fonksiyonu

Dikkat edilmiştir ki, büyük parça olmayanlar, topluluğun elemanı gibi kabul edilince, büyük parça olmayanların derecelerini ölçerek ağırlık verilmiş, bir uygunluk değeri gösteren sonuç fonksiyonunun kullanımı, arama işlemine faydalıdır. Burada, sonuç fonksiyonu, geri dönen büyük parça büyüklüğünü, uygunluk değeri gibi yeniden oluşturur.

3.4.8.6. Mutasyon oranı

Bir çok mutasyon oranı denenmiştir ve en mükemmel değeri hariç, belirlenmiş bir farka dikkat edilmemiştir.

Tanımları baz alındığında, probleme özgün, iyileştirme algoritmalarının etkisi, çok büyük olacaktır. 4. Bölümde, bu yardımcı algoritmaların nasıl geliştirileceği ve problemin çözüm uzayını nasıl daraltacağı (blok oluşturma), ve çaprazlama operatöründe ne gibi değişikler yapılabileceği üzerine çalışmalar verilecektir.

4. YAPILAN ÇALIŞMA

Gsp' nin GA, GP ve yardımcı algoritmalarla çözümü üzerinde yapılabilecek katkılar araştırılmıştır. Problemde mesafeler baz alınarak, en kısa mesafeye sahip rota en iyi rota olarak belirlenecektir.

Doğrudan rotalar kodlandığında, oluşturulan bu kodlar üzerinde tanımlanacak olan çaprazlama operatörü için yeni bir tanımlama ile problem boyutunun küçültülerek, çözüm zamanının azaltılması sağlanacaktır. Mutasyon oranı ihtimal hesaplarıyla belirlenerek, yerel çözümlerden kurtulma fikrinin yanında iyi çözümden kolayca ayrılmak da önlenecektir.

Problem, çizge üzerinde tanımlanırsa, çizgeye ne gibi yardımcı algoritmalar tanımlanabilir araştırılmıştır. Ancak uygulama zorluğundan dolayı ortaya konulan yardımcı algoritmanın problem boyutunu nasıl küçültüleceği verilerek, daha sonra yapılabilecek çalışmalar olarak ele alınmıştır.

4.1. Çizge Üzerinde Yardımcı Algoritma Geliştirme

TSP için GA' yı özel bazı tanımlamalar ile kullanarak, GSP için daha iyi bir çözüm yakalanabileceği araştırılacaktır. 3. bölümde bahsedilen çalışmalardan da anlaşılacağı gibi probleme çizge, yönlendirilmemiş, matrisiyle bakmak kontrolde avantaj sağlayacaktır. Aynı zaman da GA için bireyi kodlama da, doğrudan noktaların numarasının kullanılması, rota üzerinde tanımlanacak çaprazlama ve mutasyon operatörlerini kolaylaştıracaktır. Her bir birey bir rota olarak alınacaktır. Bireyin uygunluk değeri toplam mesafesinden hesaplanacaktır.

N={1,2,....N} gezgin satıcının uğrayacağı şehirler kümesidir ve i ve j şehirler, arasındaki mesafe $\mathbf{X}_{i,j}$ ile verilirse.

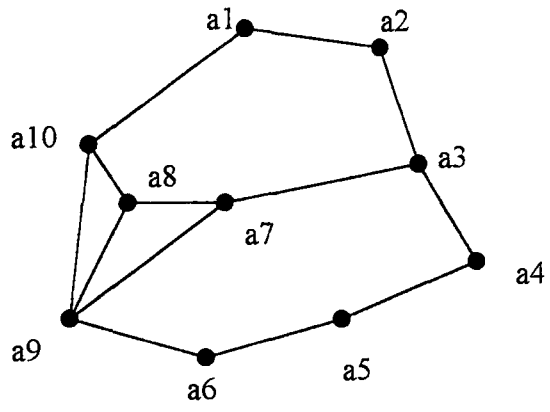
Çizge matrisi formül 4.1.' deki gibi tanımlanacaktır.

$$C_{i,j=} \quad \begin{matrix} 1 \text{ Eğer } i. \text{ Şehiri } j. \text{ Şehir izliyorsa} \\ 0 \text{ Aksi durum} \end{matrix} \qquad \textbf{(4.1.)}$$

Aynı şekilde D_i derece dizisi formül 4.2. ile verilmiştir.

$$D_i = \sum_{i=1}^{N} C_{i,j} \qquad \forall\ i,j \in N \qquad (4.2.)$$

Mümkün olabilecek çözümler elde edebilmek için; çizge matrisinden de görüleceği gibi dereceleri 2 olan noktaların hepsi rotada, mutlaka sabit yerlerde bulunacaklardır. Rotada her noktaya bir defa uğramak gerektiği için, komşu oldukları noktalara bağlanmak zorunda kalacaklardır. Bununla beraber dereceleri 3 ve daha fazla olan noktalar gerçekte derecelerinin büyüklüğü oranında rota içinde yer değiştirme serbestliğine sahiptirler. Dolayısıyla ilk etapta çizge matrisinden dereceleri 2 olan noktaları çıkarmak problem boyutunda hiç beklenmeyen bir oranda küçülme sağlayabilir. Fakat bu durumda rota içinde yer alacak, fakat GA operatörlerinde kullanmayacağımız bu noktaların yerlerinin nasıl sabitleştirileceği ve rotanın uygunluk değeri hesaplanırken, bunların yeniden işleme nasıl alınacağı problemi gelmektedir. Bütün bu uygulama zorluğuna rağmen, yapılabilirse probleme büyük kolaylık sağlayacağı muhakkaktır. rota 'a1a2a3a4a5a6a7a8a9a10' katarı için, şekil 4.1.' de oluşturulan çizge verilmiştir.



Şekil 4.1. Örnek Bir Rota İçin Oluşturulan Çizge

Şekil 4.1. de verilen çizgeden de görüleceği üzere noktalardan a1, a4, a5, a6 dereceleri 2 olanlardır. Bunların yerleri rota üzerinde sabittir. Yani a1, a10 ile a2 arasında olmak zorundadır. Aksi durumda mümkün olmayan çözüm ortaya çıkacaktır. Dolayısıyla 10 farklı noktadan oluşan bir rotada 4 noktayı çıkararak problem boyutu çok büyük ölçüde küçültülebilir. Çizge matrisi ve derece dizisi:

$$C[i,j]=\begin{bmatrix} 0&1&0&0&0&0&0&0&0&0 \\ 1&0&1&0&0&0&0&1&0&0 \\ 0&1&0&1&0&0&1&0&0&0 \\ 0&0&1&0&1&0&0&0&0&0 \\ 0&0&0&1&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0&1&0 \\ 0&0&1&0&0&0&0&1&1&0 \\ 0&1&0&0&0&0&1&0&1&1 \\ 0&0&0&0&0&1&1&1&0&1 \\ 1&0&0&0&0&0&0&1&1&0 \end{bmatrix} \qquad D[i]=\begin{vmatrix} 2 \\ 3 \\ 3 \\ 2 \\ 2 \\ 2 \\ 3 \\ 4 \\ 4 \\ 3 \end{vmatrix}$$

Rotada dereceleri 2 olan noktaları çıkardıktan sonra, geriye kalan noktaları kodlayıp bu kod üzerinden GA nın operatörleri kullanılabilir. Rotaya dikkat edilirse sabit noktalar hariç oluşturulan kod 'a2 a3 a7 a10 a8 a9' dur. Dolayısıyla 4 noktayı, GA operatörlerine uygulamamıza gerek kalmayacak, değişik ihtimaller p=10! Gibi bir rakamdan p=6! ' e düşecektir.

Bu yeni rota üzerinde Çaprazlama operatörünü uygulamak çok daha erken sonuca götürebileceği gibi mümkün olmayan çözümleri çözüm kümesinden çıkararak çözüm uzayını büyük ölçüde daraltacaktır.

Ancak örnekten de görüleceği üzere kodlama zorluğu çıkacaktır. Çünkü GA nın klasik kodlaması ikili bir bit dizisiyle elde ediliyordu. Eğer kodlama için ikili bit dizisi kullanılırsa; 100 noktadan oluşan bir rota için 700 bit uzunluğunda bir dizi kullanmak gerekecektir. Buda yine problem boyutunu büyütecektir. Bu yüzden bit dizisi yerine

ondalık sayı dizisiyle kodlama daha uygun olur. Böylece kod yukarıdaki indislerle doğrudan kullanılabilir.

Mümkün olmayan çözümlerden kurtulmanın bir yolu da böyle çözümlerin oluşmasını engellemektir. Mümkün olmayan çözümlere, rotada aynı merkeze birden fazla uğrama veya bazı merkezlere hiç uğramama sebep olmaktadır. Bunun için bölüm 4.2.' de belirtilen birey kırarak çaprazlama, Genetik programlamaya **yeni bir çaprazlama operatörü** tanımlayarak bu engeli aşabilir.

4.2. Birey Kırarak Çaprazlama

Gezgin Satıcı Probleminde karşılaşılan en büyük problemlerden biri, çok geniş olan çözüm uzayında, gerçekte mümkün olamayacak çözümlerin de bulunması ve bunların çok fazla olmasıdır. Çünkü daha önceki anlatılanlardan da görüleceği gibi problemi çözmek için elde edilen boş çözümleri, elemek veya mümkün hale getirmek, çok büyük ekstra bir yük getirmektedir. Hatta bazen bunlara harcanan zaman, problemi çözmek için harcanan zamanın büyük bir kısmını oluşturduğu görülebilir. Örneğin gerek "rota temelli çaprazlama" da gerekse "ardısıl temelli çaprazlama" da problemdeki bireyler çaprazlandıktan sonra rotada bazı noktaların birden fazla tekrarlandığı bunun yanında bazı noktalarında hiç yer almadığı görülmüş ve bundan sonra elde edilen dölleri iyileştirme algoritmaları tanımlanmış ve kullanılmıştır.

Benzer şekilde Hopfield ağı, IDT , Lin- Kernighan iyileştirmeleri, ve diğer optimum bulma yöntemlerinde de çözüm uzayında elde edilen boş çözümlerin çokluğundan ve elenmesinden şikayet edilir.

Yapılan bu çalışmada Genetik Algoritmanın Çaprazlama operatörü , boş çözümler elde etmemek için, nasıl tanımlanır konusu araştırılmış ve böylece bunları ayıklamak veya iyileştirmek için zaman harcamadan kurtulmaya çalışılmıştır. Bunun için en basit olarak iki birey rota arasında yapılan klasik çaprazlamada, aynı noktaların tekrarlanacağı ve bazı noktaların olmayacağı muhakkaktır. Ama eğer iki ebeveyn birey yerine bir birey alınırsa ve bu birey kendi içinde ikiye bölünürse (birey kırma) elde edilen bu alt rotaların çaprazlanmasından boş çözüm elde etmek mümkün olmayacaktır.

Yani elde edilen döller daha sonra birleştirildiğinde, ne aynı noktaların tekrarı olabilecektir ne de bazı noktaları kaybetme ihtimali olmayacaktır.

Örnek:

Rota kodlamada doğrudan rota üzerindeki noktaların sırasal numarası kullanılmıştır. Yani eğer merkezler

Elazığ

Malatya

Adana

Bingöl

Diyarbakır

Siirt

Van

Ağrı

Muş

Gaziantep

Erzurum

Erzincan

Tunceli

Mardin

Hakkari

Şırnak

Şeklinde seçilmiş ise verilen rota 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 şeklinde olacaktır. Dolayısıyla rotayı değiştirmek için sırasal numaraları değiştirmek yeterli olacaktır.

Ebeveyn

1 5 2 8 12 15 6 4 9 10 7 13 3 14 11 16 olursa;

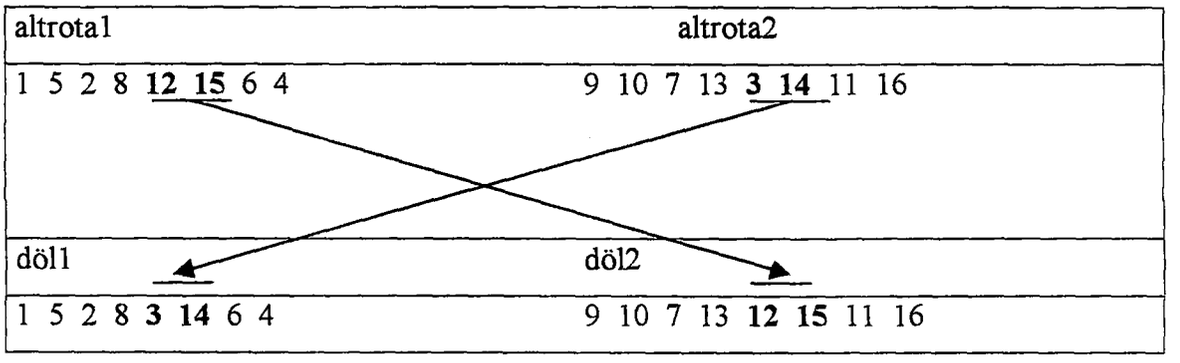
kırıldığı zaman alt rotalar tablo 4.1.1. deki şekilde olacaktır. Döller birleştirilirse;

Döl

1 5 2 8 **3 14** 6 4 9 10 7 13 **12 15** 11 16

şeklinde olacaktır.

Tablo 4.1. Kırarak Çaprazlama



Bütün bunlar göz önüne alındığında Genetik Algoritmanın diğer operatörlerine ve yeni nesil üretmeye daha fazla zaman kalacak ve muhakkak ki sonuca daha erken ulaşabilecektir.

Başlangıç topluluğu, verilen noktalar la rasgele bir şekilde oluşturulmaktadır. Topluluğun her bir bireyi için uygunluk değerinin hesaplanması için önce toplam mesafeler sıralanmakta (rotalarda buna göre sıraya konmaktadır) daha sonra her bir bireye 0 ile 2 arasında ağırlık verilmektedir.

n:topluluktaki birey sayısı olmak üzere, uygunluk değeri formül 4.3.' den hesaplanır

Uyg[i]=2-i*2/n **(4.3.)**

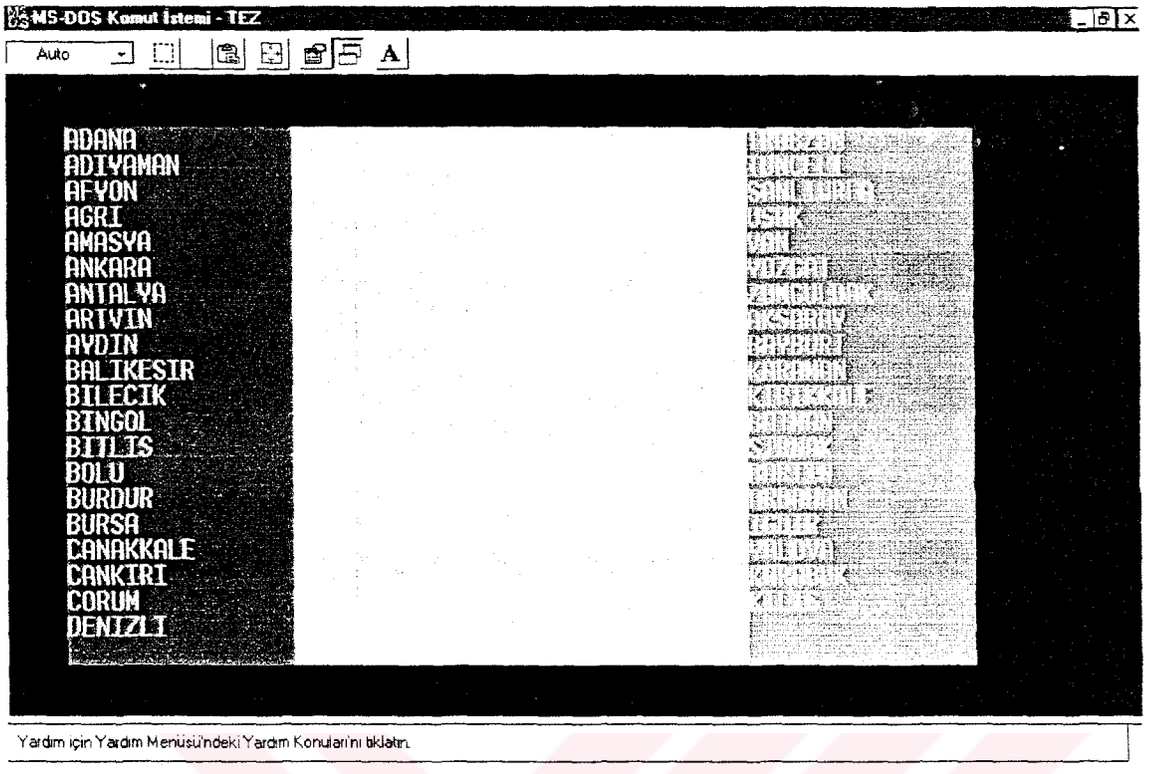
Uygunluk değerleri hesaplanan toplulukta, kalması gereken bireyler ve yok olması gereken bireyler, klasik rulet tekerleği yöntemiyle uygunluk değerlerinden faydalanılarak kararlaştırılır. Böylece oluşturulan yeni toplulukta birey kırarak çaprazlama yöntemiyle devam edilir. Ancak olası lokal minimumlardan kurtulmak için, benzeşen toplulukta, mutasyon operatörüne ihtiyaç vardır. Fakat eğer bu operatör gelişigüzel kullanılırsa, elde edilen çözümden daha kötü çözümlere yönelme de söz konusu olabilir. Bundan dolayı bu operatör de uygulanacak bireylerin uygunluk değerleriyle ters orantılı bir şekilde, ilişki kurulmalı ve yine rasgele yapılmalıdır.

Aslında sürekli mutasyon operatörü topluluk benzeştiğinde devreye girdiğinden yeniden üretmenin ne zaman sona erdirilmesi gerektiği açık değildir. Bu yüzden yeterince çok nesil için program çalıştırılmalı ve sonucun optimum olduğuna böyle karar verilmelidir.

Yakaladığı iyi çözümleri bırakmadığı için topluğun kötüye gitmesi imkansızdır. Bu hızlı bir tırmanma tekniğinin de özelliğini kullandığından, çok daha kesin bir şekilde en optimum çözüme topluluğu itecektir. Aslında değişken sayıdaki noktalardan oluşan alt rotaların sürekli iyi çözüm için değiştirilmesi IDT ve Lin Kernighan iyileştirmelerine benzer kolaylıklarda getirecektir.

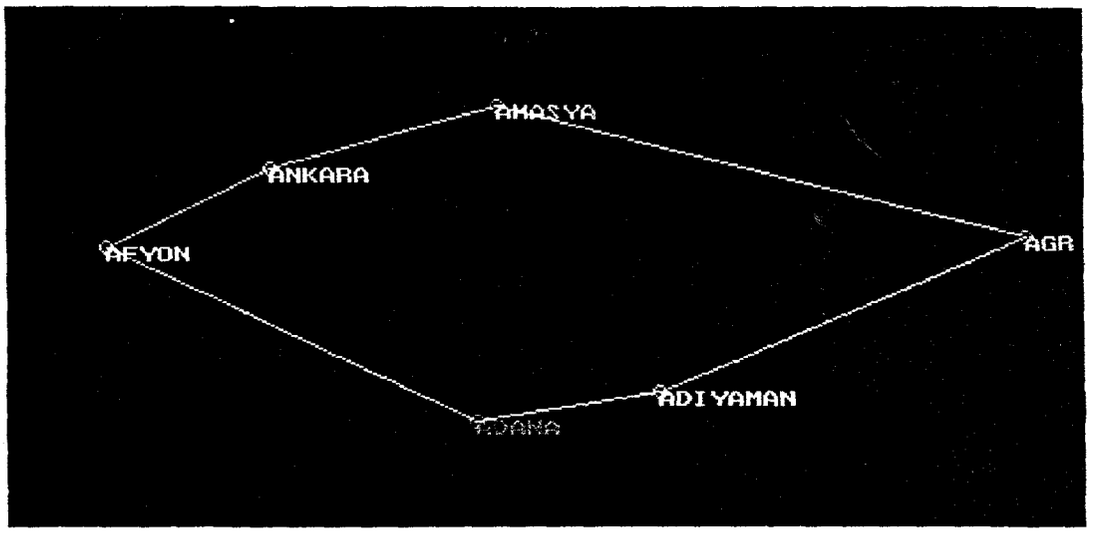
Topluluktaki birey sayısı dışarıdan isteğe bağlı seçilebileceği gibi örneklerde rotada bulunan nokta sayısına bağlı olarak seçilmiştir. Bu uygulamada matris boyutunda, tek parametre kullanma gibi bir kolaylık getirmiştir.

Yazılan uygulama programda, Türkiye'deki şehir isimlerinin verildiği bir menüden rotada bulunması gereken şehir isimleri Enter tuşu ile seçilmektedir. Plaka numaralarına göre sıralanmış 80 şehirden istenen noktalar seçildikten sonra ESC tuşu ile program başlatılır. Yeni nesiller boyunca, her hangi bir anda programın elindeki iyi rotayı vermesi, herhangi bir tuşa basarak sağlanabilir. Çizge olarak verilmiş rota tatmin edici değil ise veya daha iyi çözümler aranmak istenirse, yine <e,E > tuşlarından biriyle çözüme devam edilebilir. Sonuç, toplam mesafeleri ile rotalar çıkış dosyasına yazılarak verilirken, rotayı daha iyi görmek için şehir merkezlerinin harita üzerindeki yerleşiminden de çizge ile grafik ekrandan verilmiştir.



Şekil 4.2. Programın bilgi giriş menüsü

Programın bilgi giriş menüsü şekil 4.2.' de verilmiştir. Programın seçilen 1,2,3,4,5,6 numaralı şehirlere grafik cevabı şekil 4.3.' te verilmiştir.



Şekil 4.3. Önerilen rotanın çizge olarak grafik ekranda çıktısı

5. SONUÇLAR

Yukarıda tanımlanan operatörler aracılığıyla hazırlanmış Genetik Programlama nın probleme cevabı ekler kısmında program koduyla birlikte verilmiştir. Çözümde Pentium 133 (PC) kullanılmıştır. 1000 nesil üretmek için program 4 saniye gibi kısa bir zaman biriminde sonucu vermiştir. Nokta sayısını arttırdığımda sonuç, üzerinde belki de yorum yapılabilecek fakat optimum olabilecek çözümler verecektir.

Optimum veya optimuma yakın çözümü çok hızlı yakalaması hızlı tırmanma tekniği olan Hill Climbing yönteminin benzer özelliklerini taşıdığını göstermektedir. Yine 2 saniye gibi bir zaman sürecinde çözüme ulaştığı için 16 noktalı rotayı 500 nesil üretme için çok hızlı çalışabildiğini de göstermektedir. Ancak görülebildiği kadarıyla en yakın olmasına rağmen en optimum çözüm olmayabilir. Daha öncede belirttiğim gibi programın sonunu belirleme kriteri olmadığından 500. Nesilde yakalamayabilir. Fakat bunun yerine program 5000 nesil için çalıştırılırsa; değişik başlangıçlar ve rasgele çaprazlamalar sonucunda sonucu mutasyon operatörü yardımıyla yakalayabileceği görülebilir. Programa optimum çözümü yakalamada daha geniş imkan artı zamanla verilebilir. Eğer problemin genel minimum değeri yerel yakalanan herhangi bir minimum değerine çok yakın ise mutlak optimum noktayı yakalaması oldukça zor olacaktır. Çünkü mutasyon operatörü iyi çözümler arasındaki farkla orantılı olarak rasgele işletilmektedir.

Program da gerek çaprazlama oranı gerekse mutasyon oranı bireyin uygunluk değerine bağlı olarak rasgele üretildiği için; bu oranlar birer katsayı gibi görülüp üzerinde oynanmamıştır. El ile dışarıdan bu değerler üzerinde oynamak genetik algoritmanın rasgele arama özelliğini sınırlamak olacaktır. Çünkü el ile sınırlı sayıda ve program boyunca sabit olan oranlarla işlem yapma sınırlılığını getirecektir. Bunun yerine zaten rasgele arama yöntemi olan genetik algoritmanın yapısı burada da korunarak bir taraftan da iyi çözümlerin unutulmaması için sıralama yapılıp minimum bir tane en iyi çözüm saklanmıştır. Dolayısıyla yerel minimumlardan kurtulup genel minimumu bulmak için rasgele oranlara göre kullanılan mutasyon oranı elinizdeki çözümden daha iyi bir çözüm yakalamadıkça onu saklamaktadır.

Gezgin Satıcı Probleminde nokta sayısı arttıkça yerel minimumların sayısı gittikçe artacaktır. Dolayısıyla doğrudan Hill Climbing yönteminin bu yerel

minimumlardan kurtulması çok zordur. Hopfield Ağı ile bu problemi çözmekte aynı sebepten dolayı zorlanacaktır. Ancak tabi ki ağ için asıl problem mümkün olmayan çözümler olacaktır.

IDT ve Lin - Kernighan iyileştirmeleri yerel minimumlardan rahatça kurtulmasını sağlayacaktır. Buna rağmen oldukça yavaş işleyeceklerdir. Birlikte kullanıldıklarında daha iyi sonuçlar alınabilir.

Genetik programlamanın da bu problemi çözmede en az bu yöntemler kadar iyi, bununla birlikte bu yöntemlerden daha hızlı sonuç verebileceği görülebilir.

İlerde yapılabilecekler: Çizge üzerinde komşuluk ilişkileri göz önüne alınarak dereceleri 2 olan noktalar sabit alınmamıştır. Daha öncede vurgulandığı gibi uygulama zorluğu vardır. Ancak eğer yapılabilirse programın çalışma zamanını ve çözüm rotayı yakalama zamanını oldukça küçültecektir.

KAYNAKLAR

Aytuğ, H., 1996, Stopping Criteria for Finite Length Genetic Algorithms, INFORMS Journal on Computing Vol. 8, No:2, Spring 1996

Coit, D.W., 1996, Adaptive Penalty Methods for Genetic Optization of Constrained Combinatorial Problems, INFORMS Journal on Computing Vol. 8, No:2, Spring

Corno, F., ve diğ., 1996, GATTO: A Genetic Algorithm for Automatic Test Pattern Generation for Large Synchronous Sequential Circuits, IEEE Transactions on computer-aided design of integrated circuits and systems, vol. 15. No:8, August

Crecsenzi, P., 1997, A compendium of NP Optimization Problem, Technical report,

Duncan, T., 1998, The Vehicle Routing Problem, Technical report,

Forrest, S., 1993, Genetic Algorithms: Principles of Natural Selection Applied to Computation, SCIENCE Vol. 261 13 August.

Goldberg, D.E. ,1989, Genetic Algorithms in Search, Optimization, and Machine Learning By Addison-Wesley Publishing Company, Inc.

Karcı, A., 1998, Çizge Algoritmaları ve Çizge Bölmeleme, Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi

Kolcu, İ., 1996, Genetic Algorithms, Cranfield University, MPhil Thesis

Kuo, T., 1996, A Genetic Algorithm with Disruptive Selection, , IEEE Transactions on systems, man, and cybernetics- part B: cybernetics, vol. 26. No.2, April 1996

Liepins, G.E., 1992, Genetic Algorithms, Characterizing Crossover in Genetic Algorithms, Annals of Mathematics and Artificial Intelligence 5, 27-34.

Moscato, P., 1998, TSPBIB Home Page , Densis- FEEC-UNICAMP Universidade Estadual de Campinas Caixa Postal 6101 BRAZIL

Park, K., 1995, On the effectiveness of genetic search in combinatorial optimization, 10th ACM Symposium on Applied Computing, Genetic Algorithms and Optimization, February, 1995

Potvin, J., ve diğ., 1996, The Vehicle Routing Problem with Time Windows Part II: Genetic Search, INFORMS Journal on Computing Vol 8. No:2, Spring

Smith, K., 1996, An Argument for Abondoning the Travelling Salesman Problem as a Neural-Network Benchmark , University of East Anglia.

Youness, H. L., 1998, GAs, Hill-Climbing and Local Structure, University of California, Irvine Computer Science Building

EKLER

İLAVE (PROGRAM KODU Turbo Pascal 7.0)
GSP Genetik Programlama Program Kodu

```
unit altpler;
interface
uses crt,dos,veri;
type
diz_st15=array [1..50] of string[15];
diz_int=array [1..50] of integer;
st15=string[15];
const
secme=13;
var
   n,i:integer;
   cev:char;
   rota:diz_st15;
   konum:diz_int;


procedure pen1;
procedure pen2;
procedure pen3;
procedure pen4;
Procedure menu;
procedure tuslar(var i:integer; var  diz_rota:diz_st15; var
konum:diz_int);

implementation
procedure pen1;
begin
          window(5,3,20,23);
          textbackground(1);lowvideo;
end;
procedure pen2;
begin
          window(21,3,36,23);
          textbackground(2);lowvideo;
end;
procedure pen3;
begin
          window(37,3,52,23);
          textbackground(3);lowvideo;
end;
procedure pen4;
begin
          window(53,3,68,23);
          textbackground(4);lowvideo;
end;
Procedure menu;
var
   i:integer;
begin
window(1,1,80,25);
textbackground(black); clrscr;
          pen1; clrscr;
     for i:=1 to 20 do begin
          writeln(seh[i]);end;
```

```
          pen2; clrscr;
     for i:=21 to 40 do begin
          writeln(seh[i]);end;
          pen3; clrscr;
     for i:=41 to 60 do begin
          writeln(seh[i]);end;
          pen4; clrscr;
     for i:=61 to 80 do begin
          writeln(seh[i]);
     end;
end;

procedure tuslar(var i:integer; var diz_rota:diz_st15; var
konum:diz_int);
var
 sat:integer;
 C: Char;
 secildi:boolean;
begin
pen1;gotoxy(1,1);
sat:=1;i:=0;
secildi:=false;
 repeat
 C := Readkey;
 case ord(c) of
 sag: begin case lo(windmax) of
           1..19: begin
                  if not secildi then begin
textbackground(1);gotoxy(1,sat);write(seh[sat]);end;
                  pen2;
gotoxy(1,sat);highvideo;write(seh[sat+20]);lowvideo; end;
           20..35: begin
                  if not secildi then begin
textbackground(2);gotoxy(1,sat);writeln(seh[sat+20]);end;
                  pen3; gotoxy(1,sat);highvideo;write(seh[sat+40]);
lowvideo;end;
           36..51: begin
                  if not secildi then begin
textbackground(3);gotoxy(1,sat);writeln(seh[sat+40]);end;
                   pen4;
gotoxy(1,sat);highvideo;write(seh[sat+60]);lowvideo;end;
           52..67: begin
                  if not secildi then begin
textbackground(4);gotoxy(1,sat);writeln(seh[sat+60]);end;
                   pen1;
gotoxy(1,sat);highvideo;write(seh[sat]);lowvideo;end;
           end;
           secildi:=false;
           end;

 sol: begin case lo(windmax) of
           1..19: begin
                  if not secildi then begin
textbackground(1);gotoxy(1,sat);writeln(seh[sat]);end;

pen4;gotoxy(1,sat);highvideo;write(seh[sat+60]);lowvideo;end;
           20..35: begin
                  if not secildi then begin
textbackground(2);gotoxy(1,sat);writeln(seh[sat+20]);end;
```

```
pen1;gotoxy(1,sat);highvideo;write(seh[sat]);lowvideo;end;
           36..51: begin
                   if not secildi then begin
textbackground(3);gotoxy(1,sat);writeln(seh[sat+40]);end;

pen2;gotoxy(1,sat);highvideo;write(seh[sat+20]);lowvideo;end;
           52..67: begin
                   if not secildi then begin
textbackground(4);gotoxy(1,sat);writeln(seh[sat+60]);end;

pen3;gotoxy(1,sat);highvideo;write(seh[sat+40]);lowvideo;end;
          end;
          secildi:=false;
          end;

 asa: begin if sat>=20 then sat:=1;
           case lo(windmax) of
           1..19: begin
                   if not secildi then begin
textbackground(1);gotoxy(1,sat);writeln(seh[sat]);end;

inc(sat);gotoxy(1,sat);highvideo;write(seh[sat]);lowvideo;end;
           20..35: begin
                   if not secildi then begin
textbackground(2);gotoxy(1,sat);writeln(seh[sat+20]);end;
                   inc(sat);
gotoxy(1,sat);highvideo;write(seh[sat+20]);lowvideo;end;
           36..51: begin
                   if not secildi then begin
textbackground(3);gotoxy(1,sat);writeln(seh[sat+40]);end;
                   inc(sat);
gotoxy(1,sat);highvideo;write(seh[sat+40]);lowvideo;end;
           52..67: begin
                   if not secildi then begin
textbackground(4);gotoxy(1,sat);writeln(seh[sat+60]);end;
                   inc(sat);
gotoxy(1,sat);highvideo;write(seh[sat+60]);lowvideo;end;
     end;
     secildi:=false;
     end;
 yuk: begin if sat<=1 then sat:=20;
           case lo(windmax) of
           1..19: begin
                   if not secildi then begin
textbackground(1);gotoxy(1,sat);writeln(seh[sat]);end;
                   sat:=sat-1;
gotoxy(1,sat);highvideo;write(seh[sat]);lowvideo;end;
           20..35: begin
                   if not secildi then begin
textbackground(2);gotoxy(1,sat);writeln(seh[sat+20]);end;
                    sat:=sat-1;
gotoxy(1,sat);highvideo;write(seh[sat+20]);lowvideo;end;
           36..51: begin
                   if not secildi then begin
textbackground(3);gotoxy(1,sat);writeln(seh[sat+40]);end;
                    sat:=sat-1;
gotoxy(1,sat);highvideo;write(seh[sat+40]);lowvideo;end;
           52..67: begin
```

```
                  if not secildi then begin
textbackground(4);gotoxy(1,sat);writeln(seh[sat+60]);end;
                   sat:=sat-1;
gotoxy(1,sat);highvideo;write(seh[sat+60]);lowvideo;end;
     end;
     secildi:=false;
     end;
 secme:    begin secildi:=true;
           case lo(windmax) of
           1..19: begin
                  i:=i+1; konum[i]:=sat; diz_rota[i]:=seh[sat];

gotoxy(1,sat);highvideo;write(seh[sat]);lowvideo;end;
           20..35: begin
                   i:=i+1; konum[i]:=sat+20; diz_rota[i]:=seh[sat+20];

gotoxy(1,sat);highvideo;write(seh[sat+20]);lowvideo;end;
           36..51: begin
                   i:=i+1; konum[i]:=sat+40;diz_rota[i]:=seh[sat+40];

gotoxy(1,sat);highvideo;write(seh[sat+40]);lowvideo;end;
           52..67: begin
                   i:=i+1; konum[i]:=sat+60;diz_rota[i]:=seh[sat+60];

gotoxy(1,sat);highvideo;write(seh[sat+60]);lowvideo;end;
           end;
           end;
 end;
until ord(c)=27;

end;

Begin
writeln;
End.


program rotasyon;
uses graph,crt,veri,altpler;
label don;
type
     str8=string[8];
    diz_real=array [1..50] of real;
    mat_real=array [1..50,1..50] of real;
    mat_int=array [1..50,1..50] of integer;

var
es,n,nes:integer;
konum:diz_int;
sehler:diz_st15;
sehir:diz_st15;
mes_mat:mat_int;
tmes2,uyg,t_mes_diz:diz_real;
ciz_mat:mat_int;
der_diz:diz_int;
yeni_top,topluluk:mat_int;
rotada,rota:diz_int;
tus:char;
```

```
procedure mes_mat_ol(n:integer;konum:diz_int;var mes_mat:mat_int);
var
   i,j:integer;
begin
     for i:=1 to n do
         for j:=1 to n do
         if konum[i]<konum[j] then
             mes_mat[i,j]:=mes[konum[i],konum[j]]
            else
                mes_mat[i,j]:=mes[konum[j],konum[i]];
end;
procedure ciz_mat_ol(n:integer; mes_mat:mat_int;var ciz_mat:mat_int);
var
   i,j:integer;
begin
     for i:=1 to n do
         for j:=1 to n do begin
         if mes_mat[i,j]<>0 then
            ciz_mat[i,j]:=1
            else
            ciz_mat[i,j]:=0
         end;
end;
procedure derece_ol(n:integer;ciz_mat:mat_int; var der_diz:diz_int);
var
   t,i,j:integer;
begin
     for i:=1 to n do begin
     t:=0;
          for j:=1 to n do
          t:=t+ciz_mat[i,j];
     der_diz[i]:=t;
     end;
end;
procedure rota_ol(n:integer; var rota:diz_int);
var
   rotsay,i,k:integer;
begin
     for i:=2 to n do
         rotada[i]:=1;
         rotada[1]:=0;
         rota[1]:=1;
     rotsay:=2;
     repeat

     k:=random(n+1);
     if rotada[k]<>0 then begin
        rota[rotsay]:=k;
        rotsay:=rotsay+1;
        rotada[k]:=0;
                    end;
     until rotsay>n;
end;
procedure topl_ol(n:integer; var topluluk:mat_int);
var
   i,j:integer;
begin
     for i:=1 to n do begin
     rota_ol(n,rota);
```

```
     for j:=1 to n do
     topluluk[i,j]:=rota[j];
     end;
end;
function rota_mes(n:integer;rota:diz_int):real;
var
   i:integer;
   t_mes:real;
begin
     rota_mes:=0;
     t_mes:=0;
         for i:=1 to n-1 do
             t_mes:=t_mes+mes_mat[rota[i],rota[i+1]];
             rota_mes:=t_mes+mes_mat[rota[n],rota[1]];
end;
procedure top_mes_ol(n:integer;topluluk:mat_int;var
t_mes_diz:diz_real);
var
   i,j:integer;
   t:real;   rota:diz_int;
begin
     for i:=1 to n do begin
         t_mes_diz[i]:=0;
         for j:=1 to n do
        rota[j]:=topluluk[i,j];
        t_mes_diz[i]:=rota_mes(n,rota);
     end;
end;
procedure birey_kir(n:integer; rota:diz_int; var b1:diz_int; var
b2:diz_int);
var
   i,j:integer;
begin
     for i:=1 to trunc(n/2) do begin
              b1[i]:=rota[i];
              b2[i]:=rota[trunc(n/2)+i];
     end;
     if odd(n) then b2[i+1]:=rota[n];
     end;
procedure bir_capraz(n,kr_sy:integer;b1,b2:diz_int;var
dol1,dol2:diz_int);
var
   cap_yeri,i,j:integer;
begin
{     kr_sy:=random(n); }
     cap_yeri:=random(n-kr_sy-1)+1;
for i:=1 to cap_yeri do begin
    dol1[i]:=b1[i];
    dol2[i]:=b2[i];     end;
for i:=1 to kr_sy do begin
    dol1[cap_yeri+i]:=b2[cap_yeri+i];
    dol2[cap_yeri+i]:=b1[cap_yeri+i];
    end;
for i:=cap_yeri+kr_sy to n do begin
    dol1[i]:=b1[i];
    dol2[i]:=b2[i];
    end;
    dol2[i+1]:=b2[i+1];
end;
```

```
procedure dol_birlestir(n:integer;dol1,dol2:diz_int;var rota:diz_int);
var
   i,j:integer;
begin
     for i:=1 to n do begin
     rota[i]:=dol1[i];
     rota[n+i]:=dol2[i];
     end;
     rota[n+i+1]:=dol2[i+1];
end;
procedure caprazla(n:integer;topl:mat_int;var yeni_top:mat_int);
var
   kr_sy,n1,i,j:integer;
   b1,b2,dol1,dol2,rota:diz_int;
begin
n1:=trunc(n/2);
for i:=1 to n do begin
                 for j:=1 to n do
                 rota[j]:=topl[i,j];
    birey_kir(n,rota,b1,b2);
kr_sy:=random(round(n1-uyg[i]*n1/2));
    bir_capraz(n1,kr_sy,b1,b2,dol1,dol2);
    dol_birlestir(n1,dol1,dol2,rota);
            for j:=1 to n do
                yeni_top[i,j]:=rota[j];
end;
end;
procedure sirala(n:integer; var yeni_top:mat_int; var
mes_diz:diz_real);
var
   temp,i,j,k:integer;
   tmp:real;
begin
     for i:=1 to n-1 do
         for j:=i+1 to n do
             if mes_diz[i]>mes_diz[j] then begin
             tmp:=mes_diz[i]; mes_diz[i]:=mes_diz[j];mes_diz[j]:=tmp;
             for k:=1 to n do begin
                 temp:=yeni_top[i,k];
                 yeni_top[i,k]:=yeni_top[j,k];
                 yeni_top[j,k]:=temp;
                end;
             end;
end;
procedure uygunluk(n:integer; mes_diz:diz_real;var uyg:diz_real);
var i:integer;
begin
     uyg[1]:=2-2/n;
     for i:=2 to n do        begin
         uyg[i]:=2-2*i/n;
         if mes_diz[i-1]=mes_diz[i] then
            uyg[i]:=uyg[i-1];
            end;
end;
procedure yeni_nesil(n:integer;uyg:diz_real;eski_nesil:mat_int; var
y_nesil:mat_int);
var
   k,i,j:integer;
begin
```

```
     k:=1;
     for i:=1 to n do begin
         if round(uyg[i])=2 then begin
            for j:=1 to n do begin
                y_nesil[k,j]:=eski_nesil[i,j];
                y_nesil[k+1,j]:=eski_nesil[i,j]; end;k:=k+2;
            end
         else
             if round(uyg[i])=1 then
            for j:=1 to n do begin
                y_nesil[k,j]:=eski_nesil[i,j];end;k:=k+1;
                end;

     end;
function
esit(sat:integer;mes_diz1:diz_real;mes_diz2:diz_real):boolean;
begin
     esit:=true;
       if mes_diz1[sat]<>mes_diz2[sat] then begin
          esit:=false; exit; end;
end;
procedure mutasyon(n:integer; var topluluk:mat_int);
var
   tmp,k,kr_sy,mut_yeri,i,j:integer;
begin
for k:=1 to n div 4 do begin
     kr_sy:=random((n div 2)-1);
     mut_yeri:=random(n-2*kr_sy-2)+2;
    i:=random(n-2)+2;
for j:=mut_yeri to mut_yeri+kr_sy do
  begin
    tmp:=topluluk[i,j];
    topluluk[i,j]:=topluluk[i,j+kr_sy];
    topluluk[i,j+kr_sy]:=tmp
  end;
    end;
end;

procedure dizi_yaz(n:integer; diz:diz_real);
var
i:integer;
begin
for i:=1 to n do begin
 writeln(diz[i]:0:0);
end;                    end;
procedure mat_yaz(n:integer; mat:mat_int);
var
   i,j:integer;
begin
     for i:=1 to n do begin
         for j:=1 to n do
             write(mat[i,j]:3);
         writeln; end;
end;
procedure
dos_yaz(n:integer;y_topl:mat_int;t_mes_diz:diz_real;dosadi:str8);
var
   i,j:integer;
   rota:diz_int;
```

```
   dosya:text;
begin
     assign(dosya,dosadi);
{$i-}
     append(dosya);
     if ioresult<>0 then
                    rewrite(dosya);
{$i+}
     {rewrite(dosya);}
     writeln(dosya,'       ',nes,'. nesil rota ve toplam mesafe
uzunlu§u');
     writeln(dosya,'   ------------------------------------------------
------  ');
     for i:=1 to n do begin
         for j:=1 to n do begin
         write(dosya,y_topl[i,j]:3);end;
         writeln(dosya,'     ',t_mes_diz[i]:0:0);
     end;
writeln(dosya,'   ™nerilen rota       ');
for i:=1 to n do
write(dosya,seh[konum[topluluk[1,i]]],',');writeln(dosya,'');
     close(dosya);
end;
procedure ciz(n:integer;r:diz_int);
var
i,gd,gm:integer;
rota:array [1..80] of pointtype;
Begin
     gd:=detect;
     initgraph(gd,gm,'c:\tp\bgi');
     if graphresult<>0 then begin writeln('grafik
yüklenemedi');halt;end;

     for i:=1 to n do begin
     rota[i]:=nokta[konum[r[i]]];
     setcolor(i); outtextxy(rota[i].x,rota[i].y,seh[konum[r[i]]]);
     circle(rota[i].x,rota[i].y,3); end;
     rota[i+1]:=rota[1];
     setcolor(3);
         drawpoly(n+1,rota);readln;
closegraph;
end;
Begin
clrscr;
     randomize;
menu;
tuslar(n,sehler,konum);
window(1,1,80,25);
textbackground(black); clrscr;
mes_mat_ol(n,konum,mes_mat);
ciz_mat_ol(n,mes_mat,ciz_mat);
derece_ol(n,ciz_mat,der_diz);
topl_ol(n,topluluk);
top_mes_ol(n,topluluk,t_mes_diz);
sirala(n,topluluk,t_mes_diz);
dos_yaz(n,topluluk,t_mes_diz,'topl.dat');
uygunluk(n,t_mes_diz,uyg);
nes:=1;
repeat
```

```
repeat
gotoxy(20,24);write('Durdurmak i‡in herhangibir tuŸa bas□n');
caprazla(n,topluluk,yeni_top);
top_mes_ol(n,yeni_top,t_mes_diz);
sirala(n,yeni_top,t_mes_diz);
if nes-((nes div 100)*100)=0 then
                  dos_yaz(n,yeni_top,t_mes_diz,'topl.dat');

uygunluk(n,t_mes_diz,uyg);
yeni_nesil(n,uyg,yeni_top,topluluk);
sirala(n,yeni_top,t_mes_diz);
top_mes_ol(n,topluluk,tmes2);
es:=0;
for i:=1 to n div 2 do begin
    if esit(i,t_mes_diz,tmes2) then es:=es+1;
    if es>= n div 4 then mutasyon(n,topluluk);
    end;
nes:=nes+1;
          if nes-((nes div 100)*100)=0 then
             dos_yaz(n,topluluk,tmes2,'topl.dat');
until keypressed;
writeln('   ™nerilen rota       ');
for i:=1 to n do
writeln(seh[konum[topluluk[1,i]]]);readln;
for i:=1 to n do begin
writeln(sehler[konum[topluluk[1,i]]]);
rota[i]:=topluluk[1,i];end;
ciz(n,rota);
closegraph;
write('devam etmek i‡in e/E bas□n□z');readln(tus);
until upcase(tus)<>'E';
End.
```

```
unit veri;
interface
const
     yuk=72;
     asa=80;
     sag=77;
     sol=75;
     tab=9;
     stab=15;
seh:array[1..80] of
string[15]=('ADANA','ADIYAMAN','AFYON','AGRI','AMASYA','ANKARA',
'ANTALYA','ARTVIN','AYDIN','BALIKESIR','BILECIK','BINGOL','BITLIS','BO
LU','BURDUR',
'BURSA','CANAKKALE','CANKIRI','CORUM','DENIZLI','DIYARBAKIR','EDIRNE',
'ELAZIG',
'ERZINCAN','ERZURUM','ESKISEHIR','GAZIANTEP','GIRESUN','GUMUSHANE','HA
KKARI','HATAY',
'ISPARTA','ICEL','ISTANBUL','IZMIR','KARS','KASTAMONU','KAYSERI','KIRK
LARELI','KIRSEHIR',
'KOCAELI','KONYA','KUTAHYA','MALATYA','MANISA','KAHRAMANMARAS','MARDIN
','MUGLA','MUŠ',
'NEVSEHIR','NIGDE','ORDU','RIZE','SAKARYA','SAMSUN','SIIRT','SINOP','S
IVAS','TEKIRDAG',
```

```
'TOKAT','TRABZON','TUNCELI','SANLIURFA','USAK','VAN','YOZGAT','ZONGULD
AK','AKSARAY','BAYBURT',
'KARAMAN','KIRIKKALE','BATMAN','SIRNAK','BARTIN','ARDAHAN','IGDIR','YA
LOVA','KARABUK','KILIS','');

mes:array [1..80,1..80] of
integer=((0,332,573,969,643,490,556,1013,893,897,765,636,732,677,672,8
34,1102,583,572,767,522,1167,
492,745,810,685,208,884,869,1073,191,621,69,939,896,1012,697,333,1148,
374,
828,356,670,391,878,187,533,874,746,285,205,840,1042,791,747,715,884,4
99,1071,607,967,625,346,685,900,486,754,265,879,289,474,
656,712,769,1055,1071,891,701,245,0),
(0,0,905,692,636,767,888,771,1225,1229,1070,390,445,958,1004,1139,1407
,803,700,1099,245,1448,286,549,568,990,151,799,693,796,
321,953,401,1220,1228,770,889,448,1429,582,1109,688,1002,185,1210,164,
296,1206,504,552,537,755,866,1072,754,438,904,414,1352,
522,791,419,109,1017,623,638,1035,597,680,621,694,379,475,1050,813,794
,1174,982,219,0),
(0,0,0,1317,593,257,288,1256,347,324,207,1141,1305,423,166,276,534,388
,501,221,1095,685,997,946,1134,146,781,885,1043,1646,764,
165,560,457,323,1336,502,544,666,432,346,223,97,896,305,760,1106,367,1
251,440,472,841,1097,309,676,1288,696,700,589,656,1022,
1076,919,112,1473,476,492,365,1053,330,334,1229,1285,517,1379,1427,333
,449,818,0),
(0,0,0,0,737,1060,1486,380,1664,1569,1362,361,237,1146,1453,1418,1686,
985,829,1538,447,1636,501,371,183,1293,761,613,385,438,
958,1402,1038,1408,1640,215,990,810,1617,944,1297,1137,1371,602,1622,8
25,519,1684,248,914,938,657,531,1260,816,334,974,617,
1540,676,483,425,623,1429,234,841,1223,989,308,1126,983,375,431,1171,3
07,142,1362,1101,829,0),
(0,0,0,0,0,336,881,709,940,832,625,641,834,409,759,681,949,248,92,814,
702,899,551,366,554,569,698,338,450,1147,808,758,636,671,
916,756,253,348,880,308,560,559,647,469,898,632,798,960,751,362,438,29
4,550,523,131,895,268,222,803,114,475,496,719,705,971,
196,486,417,460,626,259,836,992,434,799,847,625,364,758,0),
(0,0,0,0,0,0,545,999,604,533,313,916,1109,191,423,382,650,131,244,478,
923,681,772,689,877,233,683,628,786,1422,681,422,483,453,
580,1079,245,319,662,185,342,258,311,671,562,603,1008,624,1026,275,346
,584,840,305,419,1116,439,443,585,399,765,819,821,369,
1249,219,268,225,796,363,77,1057,1187,283,1122,1170,407,215,735,0),
(0,0,0,0,0,0,0,1506,348,513,475,1192,1288,691,122,544,723,676,789,226,
1078,921,1048,1115,1303,430,764,1173,1239,1629,747,128,
487,725,450,1505,790,676,934,606,614,349,365,947,432,743,1089,318,1302
,572,572,1129,1385,577,964,1271,984,869,857,910,1310,
1181,902,294,1456,718,760,497,1249,384,622,1212,1268,805,1548,1596,601
,737,801,0),
(0,0,0,0,0,0,0,0,1603,1478,1271,381,526,1055,1422,1327,1595,911,755,14
77,526,1545,521,391,203,1232,840,371,332,819,1002,1421,
1082,1317,1579,269,892,830,1526,964,1206,1157,1310,622,1561,845,622,16
23,443,934,958,415,159,1169,580,623,748,637,1449,648,234,
445,702,1368,614,855,1132,1009,328,1146,922,608,720,1073,235,401,1271,
1003,908,0),
(0,0,0,0,0,0,0,0,0,299,527,1461,1625,722,272,450,446,735,848,126,1415,
661,1317,1293,1481,489,1101,1232,1390,1966,1084,293,835,
693,127,1683,849,864,678,779,582,542,417,1210,163,1080,1426,101,1571,7
60,792,1188,1444,608,1023,1608,1043,1047,632,1003,1369,
1423,1239,278,1793,823,791,685,1400,649,681,1549,1605,864,1726,1774,51
9,796,1138,0),
```

(0,0,0,0,0,0,0,0,0,0,245,1439,1629,423,396,151,210,658,775,287,1419,40
8,1295,1198,1386,300,1105,1107,1282,1945,1088,395,884,
394,172,1588,668,842,425,708,283,547,227,1194,136,1084,1430,400,1549,7
64,796,1063,1319,309,898,1612,862,976,379,932,1244,1328,
1243,224,1772,752,492,689,1292,654,610,1553,1609,581,1631,1679,220,557
,1142,0),
(0,0,0,0,0,0,0,0,0,0,0,1219,1412,216,353,94,362,444,557,401,1226,478,1
075,991,1179,80,973,900,1075,1725,956,352,752,250,417,
1381,461,622,459,488,139,415,110,974,381,906,1298,547,1329,578,644,856
,1112,102,691,1419,655,756,382,712,1037,1121,1111,253,
1552,532,285,523,1085,522,390,1360,1477,374,1424,1472,126,350,1010,0),
(0,0,0,0,0,0,0,0,0,0,0,0,197,1050,1240,1288,1556,889,733,1335,145,1540
,144,275,178,1139,459,572,380,510,625,1189,705,1312,1464,
380,894,597,1521,731,1201,924,1217,245,1446,468,241,1481,114,701,725,5
98,532,1164,720,286,878,473,1444,580,478,145,321,1253,
337,697,1127,776,303,913,839,227,383,1075,423,471,1266,1005,527,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,1243,1404,1481,1749,1082,926,1499,210,1733,
337,468,329,1332,524,759,531,341,721,1353,801,1505,1628,
411,1087,790,1714,924,1394,1088,1402,438,1610,593,282,1606,83,894,918,
791,677,1357,913,97,1071,666,1637,773,629,338,386,1417,
168,890,1320,969,454,1021,1032,138,194,1268,503,339,1459,1198,592,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,569,272,540,235,352,617,1111,490,960,775,
963,296,874,684,859,1556,888,568,670,262,595,1165,245,
510,471,376,151,445,326,862,559,794,1199,763,1160,466,533,640,896,114,
475,1304,439,631,394,523,821,905,1012,469,1380,410,159,
412,869,550,268,1245,1378,174,1208,1256,216,134,922,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,422,606,554,667,150,1194,804,1096,1082,
1270,308,880,1051,1206,1745,863,51,609,603,374,1472,668,
643,812,573,492,316,243,995,356,859,1205,242,1350,539,571,1007,1263,45
5,842,1387,862,836,735,822,1188,1212,1018,172,1572,642,
638,464,1216,423,500,1328,1384,683,1515,1563,479,615,917,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,268,507,624,438,1295,417,1144,1047,12
35,149,1042,956,1131,1794,1025,421,821,243,323,1437,517,
691,434,557,132,484,179,1043,287,975,1367,551,1398,647,713,912,1168,15
8,747,1488,711,825,375,781,1093,1177,1180,310,1621,601,
341,592,1141,591,459,1429,1546,430,1480,1528,69,406,1079,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,775,892,497,1563,217,1412,1315,1503
,417,1310,1224,1399,2062,1293,605,1089,320,319,1705,785,
959,234,825,400,752,437,1311,330,1243,1635,547,1666,915,981,1180,1436,
426,1015,1756,979,1093,188,1049,1361,1445,1448,434,1889,
869,609,860,1409,859,727,1697,1814,698,1748,1796,337,674,1347,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,156,609,922,725,771,614,802,364,7
19,540,698,1395,774,553,576,497,711,1004,114,355,706,221,
386,389,442,689,693,639,1018,755,999,311,391,496,752,349,331,1115,308,
442,629,334,677,744,857,500,1219,255,312,318,708,494,113,
1056,1212,279,1047,1095,451,195,779,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,722,766,842,615,458,646,477,641
,384,542,1239,737,666,565,614,824,848,196,277,823,216,
503,467,555,533,806,561,862,868,843,291,367,340,596,466,175,959,312,28
6,746,178,521,588,779,613,1063,104,429,325,552,535,167,
900,1056,377,891,939,568,307,701,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1289,695,1191,1167,1355,363,9
75,1106,1264,1840,958,167,713,651,224,1557,723,738,712,
653,540,416,291,1090,206,954,1300,146,1445,634,666,1062,1318,503,897,1
482,917,921,666,877,1243,1297,1113,152,1667,697,686,559,
1274,523,555,1423,1479,738,1600,1648,507,670,1012,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1601,151,406,323,1146,314,7
03,525,551,511,1143,591,1373,1418,525,955,604,1582,738,

```
1262,878,1192,252,1400,383,96,1396,259,708,727,729,677,1225,820,193,97
0,480,1505,588,623,276,176,1207,378,704,1188,783,448,811,
846,134,290,1136,568,549,1327,1066,382,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1450,1285,1453,558,1364,1
174,1349,2046,1358,803,1160,228,534,1655,735,1000,62,866,
1339,893,588,1352,544,1284,1689,762,1650,956,1023,1130,1386,376,965,17
94,929,1121,137,1013,1311,1395,1502,632,1870,900,559,902,
1359,1000,758,1735,1868,648,1698,1746,404,624,1412,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,263,318,995,348,560,407
,650,481,1045,561,1222,1320,520,804,453,1431,587,1111,
780,1073,101,1302,324,247,1337,254,557,581,586,580,1074,669,344,819,32
9,1354,437,505,133,327,1109,477,553,1037,632,394,769,695,
285,441,985,563,611,1176,915,408,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,188,922,611,297,144,7
81,744,1031,765,1037,1269,390,619,439,1246,573,926,766,
1000,364,1251,587,502,1313,385,543,567,323,317,889,445,561,603,246,116
9,305,242,130,582,1058,605,470,852,618,154,755,612,502,
658,800,433,481,991,730,671,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1110,637,430,202,62
2,799,1219,879,1225,1457,202,807,627,1434,761,1114,954,
1188,419,1439,642,419,1501,246,731,755,474,354,1077,633,426,791,434,13
57,493,300,242,499,1246,417,658,1040,806,125,943,800,405,
523,988,245,293,1179,918,705,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,893,861,1019,1645
,876,307,672,330,414,1312,478,542,539,408,219,335,78,894,
396,826,1218,509,1249,498,564,817,1073,182,652,1339,672,676,462,632,99
8,1052,1031,221,1472,452,365,443,1029,442,310,1280,1397,
454,1355,1403,206,425,930,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,861,755,865,197
,829,277,1136,1104,839,833,364,1345,498,1025,564,878,247,
1086,80,325,1082,573,468,413,817,928,988,816,507,953,476,1268,584,853,
481,138,893,692,561,951,473,742,497,610,448,504,966,882,
863,1090,898,68,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,235,1052,977,
1050,889,946,1208,632,521,578,1155,596,835,838,939,632,
1190,801,799,1252,676,615,691,44,212,798,209,856,377,385,1078,277,137,
427,879,997,847,484,761,690,305,879,551,799,953,702,598,
723,900,632,921,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,824,888,115
5,889,1121,1366,404,703,563,1330,697,1010,890,1097,508,
1348,731,621,1410,448,667,691,279,173,973,444,628,612,370,11253,389,98
,274,701,1155,619,594,936,742,77,879,709,607,725,884,447,
495,1075,814,815,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1062,1694
,1142,1818,1969,569,1400,1103,2027,1237,1707,1429,1723,
751,1951,934,623,1947,396,1207,1231,1096,970,1670,1226,438,1384,979,19
50,1086,922,651,727,1758,205,1203,1633,1282,747,1362,
1345,479,535,1581,655,431,1772,1511,933,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,812,260
,1130,1087,1001,888,460,1339,565,1019,547,861,380,1069,
176,522,1065,735,476,396,933,1061,982,912,704,1049,592,1262,700,986,61
4,335,876,889,657,945,456,875,480,665,645,701,960,1044,
1060,1082,892,148,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,587,6
02,382,1421,667,592,811,522,491,265,242,944,364,808,1154,
293,1299,488,520,1006,1262,454,841,1336,861,785,734,821,1187,1161,967,
171,1521,634,637,413,1165,372,499,1277,1333,682,1464,
1512,478,614,866,0),
```

(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,932
,883,1081,690,326,1141,367,821,343,657,460,865,256,602,
805,815,278,198,845,1062,784,740,784,877,519,1064,612,987,694,415,672,
969,479,747,258,899,236,467,725,781,762,1124,1140,878,
694,314,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5
66,1427,507,772,209,638,111,665,380,1124,530,1056,1461,
794,1422,728,795,902,1158,148,737,1566,701,893,132,785,1083,1167,1274,
503,1642,672,331,674,1131,772,530,1507,1640,420,1470,
1518,176,396,1184,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,1659,825,867,551,755,455,546,336,1219,36,1083,1429,228,
1574,763,795,1164,1420,481,999,1611,1019,1023,505,979,1345,1399,1242,2
11,1796,799,664,688,1376,653,657,1552,1608,753,1702,1750,
392,729,1141,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,1009,829,1636,963,1316,1156,1390,621,1641,844,621,
1703,328,933,957,676,420,1279,835,508,993,636,1559,695,495,444,701,144
8,364,960,1242,1008,327,1145,1002,549,605,1190,92,138,
1381,1120,907,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,469,716,335,396,503,556,722,804,753,1051,869,1004,
425,505,477,733,359,312,1148,194,475,639,367,658,749,971,614,1224,296,
270,432,713,608,227,1089,1245,181,1052,1100,461,111,893,
0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,981,134,661,327,620,352,849,284,689,884,707,104,
128,534,736,624,452,797,589,193,904,301,661,569,502,658,930,197,587,17
9,573,316,248,738,868,602,872,920,726,534,424,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,847,320,874,569,1333,561,1265,1670,779,1631,937,
1004,1111,1367,357,946,1775,910,1102,118,994,1292,1376,1483,649,1851,8
81,540,883,1340,981,739,1716,1849,629,1679,1727,385,605,
1393,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,527,257,486,486,737,418,823,799,841,90,170,
552,808,490,391,931,528,327,770,319,733,703,636,544,1064,112,453,109,7
07,319,112,872,1002,468,1006,1054,592,400,558,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,554,249,1013,419,945,1350,683,1311,617,684,
791,1047,37,626,1455,590,782,243,674,972,1056,1163,392,1531,561,220,65
3,1020,661,419,1398,1529,300,1350,1407,65,285,1073,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,320,679,528,543,889,558,1034,223,255,794,
1050,517,642,1071,697,520,797,561,975,896,702,335,1256,369,522,148,900
,113,300,1012,1068,537,1199,1247,541,469,601,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,972,318,857,1203,437,1327,537,569,895,
1151,212,730,1385,750,754,492,710,1076,1130,1016,143,1550,530,395,462,
1107,427,388,1326,1382,484,1433,1481,236,460,915,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,1201,223,348,1236,355,456,480,588,681,
976,587,445,737,247,1256,355,606,234,268,1008,578,471,939,531,495,668,
598,386,542,903,664,712,1078,833,307,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,1065,1411,264,1556,745,777,1146,
1402,445,981,1593,998,1005,515,961,1327,1381,1224,193,1778,781,628,670
,1358,635,639,1534,1590,717,1684,1732,356,693,1123,0),

(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,405,1061,578,388,392,757,904,908,
736,576,873,416,1188,524,829,457,218,872,761,481,871,452,718,476,530,5
17,584,886,887,932,1010,818,140,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,1407,355,793,738,825,773,1313,
916,224,1066,576,1593,684,719,372,187,1218,450,800,1276,798,544,822,93
5,144,202,1232,664,621,1415,1162,393,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,1591,780,812,1208,1464,649,
1043,1589,1063,1067,733,1023,1389,1443,1220,298,1774,843,832,705,1420,
665,701,1530,1586,884,1746,1794,620,816,1119,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,811,835,708,594,1274,830,
180,988,583,1554,890,546,255,435,1363,223,807,1237,886,371,1023,949,22
1,277,1185,420,390,1376,1115,641,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,80,571,827,580,466,901,
603,297,860,338,752,673,606,552,1034,202,543,75,677,268,202,842,972,55
8,976,1024,682,490,528,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,647,864,647,542,920,679,
321,927,414,789,697,551,584,1058,282,610,121,701,188,282,861,917,625,1
000,1048,749,557,450,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,256,754,165,884,333,
341,1034,233,181,453,838,953,891,440,717,646,349,835,507,825,981,658,6
42,767,856,588,877,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1010,421,774,589,
543,1290,489,75,447,853,1209,765,696,973,902,250,1052,763,759,871,914,
386,552,1112,844,988,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,589,1418,553,745,
280,637,935,1019,1126,355,1494,524,183,526,983,624,382,1359,1492,272,1
322,1370,102,248,1036,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1006,168,340,
869,232,346,575,837,788,1050,279,552,500,514,710,342,947,1103,493,807,
926,691,423,876,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1163,673,1698,
781,726,431,369,1400,265,897,1381,976,551,1004,1039,88,97,1329,600,436
,1520,1259,575,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,490,833,382,
514,733,987,808,1208,416,464,626,682,802,421,1104,1260,375,975,1064,65
5,305,1013,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1025,108,
468,376,497,812,806,224,708,372,380,509,356,614,770,665,679,727,847,58
6,536,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,917,
1215,1299,1408,603,1774,804,463,806,1263,904,662,1639,1772,552,1602,16
50,308,528,1316,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,414,

```
435,605,768,910,207,600,413,399,602,322,722,878,548,738,786,739,478,64
4,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,372,
799,1134,717,621,898,827,175,977,688,705,823,839,461,593,1037,769,913,
0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
452,1188,478,600,982,748,261,885,742,372,528,930,487,535,1121,860,541,
0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
1031,554,699,1089,611,624,635,748,310,366,1104,744,725,1228,1036,206,0
),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,1585,588,536,477,1165,442,446,1341,1397,627,1491,1539,377,561,930,0)
,
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1030,1457,1109,542,1189,1172,306,362,1405,450,226,1596,1335,780,0)
,
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,487,221,604,431,142,838,994,477,903,951,626,407,621,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,489,946,627,345,1322,1455,89,1285,1333,285,173,999,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,752,210,209,917,977,504,1051,1099,628,436,510,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,889,719,530,648,894,370,418,1085,824,802,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,405,945,1001,642,1188,1236,648,574,534,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,980,1114,360,1045,1093,484,292,670,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,185,1270,641,477,1461,1200,516,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,1426,697,533,1594,1356,572,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,1233,1281,374,84,1014,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,224,1424,1163,950,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,1472,1211,931,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,350,1138,0),
```

```
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,946,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0));
implementation
end.
```

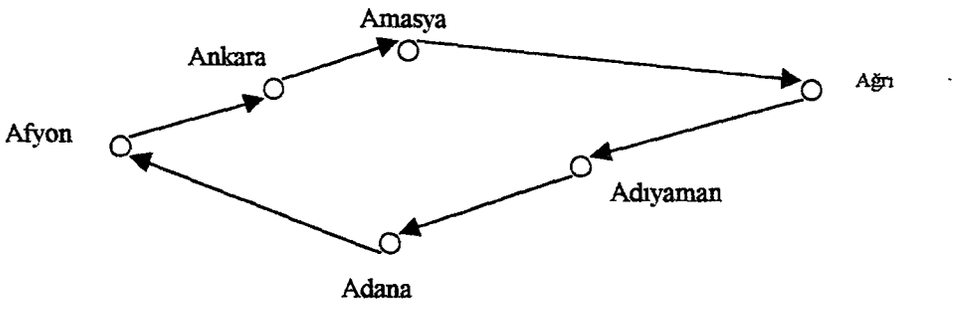
Program Çıktısı

Gelişigüzel seçilen 6 nokta için 100 nesil üretilmiş ve her 10 nesilde bir elde edilen çözüm rota aşağıda verilmiştir.

```
0. nesil rota ve toplam mesafe uzunluğu
 ------------------------------------------------------
1  2  5  4  6  3     3595   Adana,Adıyaman,Amasya,Ağrı,Ankara,Afyon
1  2  6  3  5  4     3655
1  4  3  6  5  2     3847
1  6  4  3  5  2     4428
1  4  5  3  2  6     4461
1  5  2  3  4  6     5051
     10. nesil rota ve toplam mesafe uzunluğu
 ------------------------------------------------------
1  3  6  5  4  2     2927   Adana,Afyon,Ankara,Amasya,Ağrı,Adıyaman
1  3  6  5  4  2     2927
1  3  6  5  4  2     2927
1  5  3  6  4  2     3577
1  6  3  4  5  2     3769
1  6  5  4  3  2     4117
     20. nesil rota ve toplam mesafe uzunluğu
 ------------------------------------------------------
1  3  6  5  4  2     2927   Adana,Afyon,Ankara,Amasya,Ağrı,Adıyaman
1  3  6  5  4  2     2927
1  3  6  5  4  2     2927
1  3  5  4  6  2     4062
1  3  5  4  6  2     4062
1  6  4  3  5  2     4428
     30. nesil rota ve toplam mesafe uzunluğu
 ------------------------------------------------------
1  3  6  5  4  2     2927   Adana,Afyon,Ankara,Amasya,Ağrı,Adıyaman
1  3  6  5  4  2     2927
1  3  6  5  4  2     2927
1  4  5  3  6  2     3655
1  6  3  4  5  2     3769
1  6  5  4  3  2     4117
     40. nesil rota ve toplam mesafe uzunluğu
 ------------------------------------------------------
1  3  6  5  4  2     2927   Adana,Afyon,Ankara,Amasya,Ağrı,Adıyaman
1  3  6  5  4  2     2927
1  3  6  5  4  2     2927
1  6  5  3  4  2     3760
1  6  5  4  3  2     4117
1  4  6  5  3  2     4195
```

```
     50. nesil rota ve toplam mesafe uzunluğu
 ----------------------------------------------------
1  3  6  5  4  2     2927   Adana,Afyon,Ankara,Amasya,Ağrı,Adıyaman
1  3  6  5  4  2     2927
1  3  6  5  4  2     2927
1  6  5  3  4  2     3760
1  3  5  4  6  2     4062
1  4  6  5  3  2     4195
     60. nesil rota ve toplam mesafe uzunluğu
 ----------------------------------------------------
1  3  6  5  4  2     2927   Adana,Afyon,Ankara,Amasya,Ağrı,Adıyaman
1  3  6  5  4  2     2927
1  3  6  5  4  2     2927
1  3  6  4  5  2     3595
1  3  5  4  6  2     4062
1  6  5  4  3  2     4117
     70. nesil rota ve toplam mesafe uzunluğu
 ----------------------------------------------------
1  3  6  5  4  2     2927 Adana,Afyon,Ankara,Amasya,Ağrı,Adıyaman
1  3  6  5  4  2     2927
1  3  6  5  4  2     2927
1  3  6  4  5  2     3595
1  3  5  4  6  2     4062
1  3  5  4  6  2     4062
     80. nesil rota ve toplam mesafe uzunluğu
 ----------------------------------------------------
1  3  6  5  4  2     2927 Adana,Afyon,Ankara,Amasya,Ağrı,Adıyaman
1  3  5  6  4  2     3586
1  6  5  3  4  2     3760
1  3  5  4  6  2     4062
1  3  5  4  6  2     4062
1  4  6  5  3  2     4195
     90. nesil rota ve toplam mesafe uzunluğu
 ----------------------------------------------------
1  3  6  5  4  2     2927 Adana,Afyon,Ankara,Amasya,Ağrı,Adıyaman
1  3  6  5  4  2     2927
1  3  6  5  4  2     2927
1  3  6  5  4  2     2927
1  3  5  6  4  2     3586
1  3  5  4  6  2     4062
     100. nesil rota ve toplam mesafe uzunluğu
 ----------------------------------------------------
1  3  6  5  4  2     2927 Adana,Afyon,Ankara,Amasya,Ağrı,Adıyaman
1  6  5  3  4  2     2927
1  3  5  6  4  2     3586
1  3  6  5  4  2     2927
1  3  6  5  4  2     2927
1  3  5  4  6  2     4062
```

Yukarıda görüldüğü üzere 10. Nesilden itibaren optimum çözüme ulaşmıştır ve çözüm tekrar asla kaybedilmemiştir. Çizge üzerinde çözüm rotayı gösterirsek:



Şek.5.1.

Aşağıdaki örnekte nokta sayısı 16 olarak alınacaktır. Daha öncede verildiği üzere topluluktaki rota sayısı da buna eş yani 16 alınacaktır.

```
      0. nesil rota ve toplam mesafe uzunluğu
 ------------------------------------------------------
 1  5 12 15 13 14  3  7  2  9  6  8 11 10  4 16     9273
 1 13 15  6 14  5 12  2  7  9  8  4 11  3 10 16     10162
 1 13  7 11  9 16 12 14  8  6 10 15  4  5  2  3     11397
 1 11  4  2  9  8 12  5  3  7 15  6 16 10 13 14     11625
 1  8 12 13 11  2  3  7 15  6 16 10 14  9  4  5     12196
 1  3 16 13 12  8  4  5  2  6 10 11 14  7  9 15     12361
 1 12  4 11 13  5 14  3 16 10 15  9  8  7  2  6     12595
 1 11 13 15  6 16  9  8  2 14 10  3  5 12  7  4     12836
 1 16  2  3  9  6 15  4 11 14 12  7  5  8 10 13     13034
 1 13  7  6  2  3  9 11 14  4 16  5  8 15 10 12     13073
 1 10 13  2 14 11  6  5  8  7  4  3 12 16  9 15     13279
 1 14  8  2 16 12 11 13  9  6 10 15  4  7  5  3     13681
 1  8 10 12  6 11 15  9  4 14 13 16  2  3  7  5     13835
 1 11  2 16  6  7  3 10 12  9 13 14 15  8  4  5     14031
 1 10 14 12  5  2 16  4  6  9  3 11 13  8  7 15     14101
 1  7  8 11  6  4  5 13 10 12 16  3  9 15  2 14     16071

   Önerilen rota
ELAZIG, GAZIANTEP, BINGOL, BATMAN, BITLIS, SIRNAK, ERZURUM, MUGLA,
ERZINCAN, NEVSEHIR, MARDIN, MUS, BILECIK, BALIKESIR, ESKISEHIR,
KIRIKKALE
```

50. nesil rota ve toplam mesafe uzunluğu

1	2	8	13	3	14	6	15	12	5	11	10	4	7	9	16	6319
1	2	8	13	3	14	6	15	12	5	11	10	4	7	9	16	6319
1	2	8	13	3	14	6	15	12	5	11	10	4	7	9	16	6319
1	2	8	13	3	14	6	15	12	5	11	10	4	7	9	16	6319
1	2	8	13	3	14	6	15	12	5	11	10	4	7	9	16	6319
1	2	8	13	3	14	6	15	12	5	11	10	4	7	9	16	6319
1	5	11	10	4	7	9	15	12	2	8	13	14	6	3	16	7277
1	5	11	10	4	7	9	15	12	2	8	13	14	6	3	16	7277
1	5	13	2	14	6	3	15	12	8	11	10	4	7	9	16	7425
1	6	14	2	4	11	10	8	13	3	15	12	5	7	9	16	8165
1	5	11	10	3	14	6	15	12	2	8	13	4	7	9	16	8309
1	3	14	6	8	11	10	4	13	2	15	12	5	7	9	16	9019
1	9	14	7	10	13	5	16	2	11	4	6	8	15	12	3	10478
1	9	5	16	6	7	10	15	3	4	12	8	14	2	11	13	12373
1	9	10	3	12	8	5	2	4	14	13	16	11	6	7	15	12638
1	7	16	5	12	14	2	13	9	10	8	11	6	4	15	3	13671

Önerilen rota

ELAZIG, ERZINCAN, MUS, BITLIS, ERZURUM, SIRNAK, MARDIN, BATMAN, BINGOL, GAZIANTEP, BILECIK, BALIKESIR, ESKISEHIR, MUGLA, NEVSEHIR, KIRIKKALE

100. nesil rota ve toplam mesafe uzunluğu

1	2	3	8	6	15	13	14	12	5	11	10	4	7	9	16	6074
1	2	3	8	6	15	13	14	12	5	11	10	4	7	9	16	6074
1	2	3	8	6	15	13	14	12	5	11	10	4	7	9	16	6074
1	2	3	8	6	15	13	14	12	5	11	10	4	7	9	16	6074
1	5	3	8	6	15	13	14	12	2	11	10	4	7	9	16	6442
1	2	3	14	6	5	9	10	4	15	13	8	12	7	11	16	7761
1	5	11	8	6	15	13	14	12	2	3	10	4	7	9	16	8199
1	13	8	3	14	11	10	4	7	2	6	12	5	15	9	16	8422
1	2	11	8	6	15	13	14	12	5	3	10	4	7	9	16	8765
1	2	11	8	6	15	13	14	12	5	3	10	4	7	9	16	8765
1	3	6	13	16	8	2	7	9	5	10	4	11	12	14	15	9503
1	8	12	5	13	10	4	7	3	14	6	15	11	2	9	16	9950
1	5	3	8	4	15	13	14	12	2	11	10	6	7	9	16	10500
1	11	14	4	10	16	5	9	6	7	8	13	2	12	15	3	11504
1	7	14	11	4	9	2	16	6	15	13	5	8	10	12	3	11931
1	12	4	3	5	16	10	8	9	6	7	14	11	2	15	13	13841

Önerilen rota

ELAZIG, ERZINCAN, ERZURUM, MUS, MARDIN, BATMAN, BITLIS, SIRNAK, BINGOL, GAZIANTEP, BILECIK, BALIKESIR, ESKISEHIR, MUGLA, NEVSEHIR, KIRIKKALE

150. nesil rota ve toplam mesafe uzunluğu

1	2	3	8	13	14	12	15	6	5	11	10	4	7	9	16	5757
1	2	3	8	13	14	12	15	6	5	11	10	4	7	9	16	5757
1	2	3	8	13	14	12	15	6	5	11	10	4	7	9	16	5757
1	2	3	8	13	14	12	15	6	5	11	10	4	7	9	16	5757
1	2	3	8	13	14	12	15	6	5	10	4	11	7	9	16	5762
1	2	3	8	13	14	5	11	15	6	12	4	10	7	9	16	7712
1	2	15	6	5	11	10	4	8	13	14	12	3	7	9	16	8017
1	2	3	8	4	7	12	15	6	5	11	10	13	14	9	16	9542
1	5	3	10	4	7	12	15	6	2	11	8	13	14	9	16	10000

1	12	5	3	15	7	8	13	14	2	11	10	4	6	9	16	10145
1	15	4	5	11	10	8	7	9	16	6	2	13	14	12	3	10776
1	2	4	14	12	15	13	5	11	10	6	7	9	8	3	16	11241
1	4	7	9	5	14	12	15	10	2	3	8	13	11	6	16	11474
1	10	9	14	5	4	2	12	6	13	8	16	3	11	7	15	11521
1	3	9	7	4	12	16	6	11	2	14	15	8	5	10	13	12248
1	12	16	14	2	13	5	4	7	8	15	9	11	6	10	3	12813

Önerilen rota
ELAZIG, ERZINCAN, ERZURUM, MUS, BITLIS, SIRNAK, BINGOL, BATMAN, MARDIN, GAZIANTEP, BILECIK, BALIKESIR, ESKISEHIR, MUGLA, NEVSEHIR, KIRIKKALE

200. nesil rota ve toplam mesafe uzunluğu

1	2	3	8	13	14	15	6	12	5	11	10	4	7	9	16	5707
1	2	3	8	13	14	15	6	12	5	11	10	4	7	9	16	5707
1	2	3	8	13	14	15	6	12	5	11	10	4	7	9	16	5707
1	2	3	8	13	14	15	6	12	5	11	10	4	7	9	16	5707
1	2	3	8	13	14	15	6	12	5	11	10	4	7	9	16	5707
1	14	15	6	12	5	3	8	13	2	11	10	4	7	9	16	6626
1	2	3	8	5	11	10	4	7	9	14	15	6	12	13	16	7543
1	5	11	8	13	14	15	6	12	2	3	10	4	7	9	16	7832
1	2	14	15	6	8	10	4	7	13	3	12	5	11	9	16	8983
1	14	11	10	12	8	13	2	3	5	15	6	4	7	9	16	9088
1	8	13	14	15	10	4	7	3	6	12	5	11	2	9	16	9102
1	14	15	6	4	7	13	2	3	5	11	10	12	8	9	16	9875
1	16	9	12	8	11	4	13	6	2	7	5	10	14	15	3	11254
1	5	13	6	11	10	8	9	4	14	16	7	2	12	15	3	11305
1	10	11	14	5	6	13	4	12	2	16	8	9	3	7	15	13293
1	7	8	11	6	4	14	2	9	15	3	5	16	10	12	13	14448

Önerilen rota
ELAZIG, ERZINCAN, ERZURUM, MUS, BITLIS, SIRNAK, BATMAN, MARDIN, BINGOL, GAZIANTEP, BILECIK, BALIKESIR, ESKISEHIR, MUGLA, NEVSEHIR, KIRIKKALE

250. nesil rota ve toplam mesafe uzunluğu

1	2	3	8	13	14	15	6	12	5	11	4	7	10	9	16	5626
1	2	3	8	13	14	15	6	12	5	11	4	7	10	9	16	5626
1	2	3	8	13	14	15	6	12	5	11	4	7	10	9	16	5626
1	2	3	8	13	14	15	6	12	5	11	4	7	10	9	16	5626
1	6	12	5	11	4	7	10	3	8	13	14	15	2	9	16	6945
1	6	12	5	11	4	7	10	3	8	13	14	15	2	9	16	6945
1	14	3	8	12	10	11	4	7	2	15	6	13	5	9	16	7727
1	2	3	8	13	14	15	6	11	4	7	5	12	10	9	16	7831
1	14	15	6	12	5	11	4	7	2	3	8	13	10	9	16	8152
1	2	14	15	3	10	4	7	8	13	6	12	5	11	9	16	8810
1	2	9	8	13	14	11	4	7	10	3	6	12	5	15	16	8988
1	7	6	8	14	3	12	9	2	5	15	11	10	4	16	13	9964
1	2	3	8	4	7	15	6	10	11	5	9	13	14	12	16	10250
1	12	7	11	13	4	9	10	5	15	6	16	14	8	2	3	11092
1	11	10	2	12	9	16	5	8	13	7	14	4	3	6	15	11509
1	4	13	12	16	7	15	6	5	11	2	14	10	8	9	3	13703

Önerilen rota

ELAZIG, ERZINCAN, ERZURUM, MUS, BITLIS, SIRNAK, BATMAN, MARDIN, BINGOL, GAZIANTEP, BILECIK, ESKISEHIR, MUGLA, BALIKESIR, NEVSEHIR, KIRIKKALE

300. nesil rota ve toplam mesafe uzunluğu

1	2	3	8	13	14	15	6	12	5	11	7	10	4	9	16	5618
1	2	3	8	13	14	15	6	12	5	11	7	10	4	9	16	5618
1	2	3	8	13	14	15	6	12	5	11	7	10	4	9	16	5618
1	2	3	8	13	14	15	6	12	5	11	7	10	4	9	16	5618
1	13	14	15	6	8	7	10	4	2	3	12	5	11	9	16	7701
1	13	11	7	10	4	15	6	12	5	14	8	3	2	9	16	7775
1	2	11	7	10	14	15	6	12	5	3	8	13	4	9	16	8532
1	15	5	4	11	14	12	16	9	10	7	6	8	13	2	3	8590
1	15	5	4	11	14	12	16	9	10	7	6	8	13	2	3	8590
1	2	11	7	13	14	15	6	12	5	3	8	10	4	9	16	8757
1	13	14	8	3	4	15	6	12	5	11	7	10	2	9	16	8846
1	2	3	7	10	4	15	6	12	5	11	8	13	14	9	16	9224
1	2	3	7	13	14	15	6	12	5	11	8	10	4	9	16	10327
1	13	16	9	10	7	5	11	6	4	8	2	14	15	12	3	10506
1	3	2	6	12	10	16	11	14	8	4	7	13	5	9	15	10925
1	5	14	6	3	9	11	12	7	15	4	16	2	10	8	13	12381

Önerilen rota
ELAZIG, ERZINCAN, ERZURUM, MUS, BITLIS, SIRNAK, BATMAN, MARDIN, BINGOL, GAZIANTEP, BILECIK, MUGLA, BALIKESIR, ESKISEHIR, NEVSEHIR, KIRIKKALE

350. nesil rota ve toplam mesafe uzunluğu

1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	6	13	5	7	10	11	4	3	8	12	14	15	2	9	16	6840
1	6	13	5	7	10	11	4	3	8	12	14	15	2	9	16	6840
1	8	13	14	7	10	11	4	3	6	12	5	15	2	9	16	7461
1	5	7	10	11	14	15	6	12	2	3	8	13	4	9	16	7641
1	9	14	15	5	7	10	11	4	2	3	8	6	12	13	16	7845
1	2	7	8	13	14	15	6	12	5	3	10	11	4	9	16	8216
1	2	3	8	13	4	15	6	12	5	7	10	11	14	9	16	9309
1	9	15	12	13	8	5	16	4	10	14	6	11	2	7	3	10931
1	5	8	12	6	2	11	16	9	14	4	3	13	7	10	15	11013
1	5	13	16	12	9	8	7	14	2	11	6	15	10	4	3	13804
1	4	8	5	11	14	15	9	12	10	6	2	3	16	7	13	13998

Önerilen rota
ELAZIG, ERZINCAN, ERZURUM, MUS, BITLIS, SIRNAK, BATMAN, MARDIN, BINGOL, GAZIANTEP, MUGLA, BALIKESIR, BILECIK, ESKISEHIR, NEVSEHIR, KIRIKKALE

400. nesil rota ve toplam mesafe uzunluğu

1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205

1	8	7	10	11	4	5	13	14	2	3	15	6	12	9	16	7415
1	5	7	10	4	14	15	6	12	2	3	8	13	9	11	16	7446
1	2	7	10	11	4	15	6	12	5	3	8	13	14	9	16	7454
1	14	3	10	11	4	7	8	13	2	15	6	12	5	9	16	8037
1	2	3	10	11	14	15	6	12	5	7	8	13	4	9	16	10071
1	2	3	10	13	14	15	6	12	5	7	8	11	4	9	16	10166
1	5	3	10	4	9	15	6	12	2	7	8	13	14	11	16	10414
1	16	6	4	9	7	14	13	8	5	2	12	15	11	10	3	10984
1	15	2	8	11	10	12	16	7	13	6	14	4	9	5	3	11133
1	6	3	10	14	9	16	5	7	11	15	8	12	2	4	13	11635
1	16	9	13	8	6	2	7	12	10	4	3	11	14	5	15	12267

Önerilen rota
ELAZIG, ERZINCAN, ERZURUM, MUS, BITLIS, SIRNAK, BATMAN, MARDIN, BINGOL, GAZIANTEP, MUGLA, BALIKESIR, BILECIK, ESKISEHIR, NEVSEHIR, KIRIKKALE

450. nesil rota ve toplam mesafe uzunluğu

1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	2	3	14	15	6	13	8	12	5	7	10	11	4	9	16	5443
1	5	7	10	11	4	9	6	12	2	3	8	13	14	15	16	6533
1	2	7	10	11	4	15	6	12	5	3	8	13	14	9	16	7454
1	2	3	8	12	5	7	10	11	14	15	6	13	4	9	16	7812
1	12	13	10	11	4	14	15	5	2	3	8	6	7	9	16	8809
1	2	3	7	10	11	6	12	15	4	5	8	13	14	9	16	9255
1	2	3	8	13	14	5	7	10	12	15	4	6	11	9	16	9897
1	9	16	10	7	15	8	11	5	4	6	3	14	2	12	13	10342
1	11	3	8	13	14	2	4	10	5	7	15	6	12	9	16	10357
1	4	6	5	11	8	16	10	14	7	9	13	2	15	12	3	12961
1	14	11	15	2	12	16	6	13	7	9	5	10	8	4	3	14296
1	3	6	7	5	9	2	10	8	11	14	16	12	4	13	15	14637

Önerilen rota
ELAZIG, ERZINCAN, ERZURUM, MUS, BITLIS, SIRNAK, BATMAN, MARDIN, BINGOL, GAZIANTEP, MUGLA, BALIKESIR, BILECIK, ESKISEHIR, NEVSEHIR, KIRIKKALE

500. nesil rota ve toplam mesafe uzunluğu

1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	2	3	8	13	14	15	6	10	11	4	9	5	7	12	16	5205
1	16	9	2	8	11	4	10	7	12	13	15	14	6	5	3	7417
1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	5	7	10	13	14	3	6	12	2	15	8	11	4	9	16	5205
1	11	9	6	14	2	16	5	3	10	7	12	4	8	13	15	12830
1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	13	12	5	7	10	11	4	9	15	6	3	8	2	14	16	5205
1	5	6	15	12	8	14	7	11	10	4	2	3	9	16	13	7525
1	2	6	12	5	7	10	11	13	14	8	15	3	4	9	16	5205
1	2	3	6	12	5	13	14	15	10	7	8	11	4	9	16	5205
1	13	8	9	6	11	2	16	7	10	5	15	12	14	4	3	9196
1	2	3	8	13	14	15	6	12	5	7	10	11	4	9	16	5205
1	2	3	8	13	14	15	12	5	10	11	4	7	6	9	16	5205
1	5	7	8	13	14	15	6	12	2	3	10	11	4	9	16	7437
1	5	3	8	13	14	15	6	12	2	7	10	11	4	9	16	5786

```
   Önerilen rota
ELAZIG, ERZINCAN, ERZURUM, MUS, BITLIS, SIRNAK, BATMAN, MARDIN,
BINGOL, GAZIANTEP, MUGLA, BALIKESIR, BILECIK, ESKISEHIR, NEVSEHIR,
KIRIKKALE
```

Burada sağdaki toplam rota mesafelerinden görüldüğü gibi çözüm rotayı 350. nesilde bulmuştur. Fakat 500. nesil e kadar daha iyi bir çözüm yakalayamamıştır.

Aynı örnek 1000 nesil sonunda toplam mesafeyi 200 km kadar daha küçültebilmiştir.

Başlangıç Topluluğu

```
  1  5 16 13 15  6  3 14  9  4  7 11  8 12  2 10     9951
  1  9 10 11  7 13 16  6  5  8 14  4 15  2  3 12     10550
  1 11 16 15  8  3 10  7 12 14  2 13  5  9  4  6     10643
  1  6 13 15  8  2  4  5  9 16  3 11 14 10  7 12     10848
  1 14 12 16 15  3  7 11 10 13  2  5  4  9  8  6     10853
  1  5 12  8  4 13 14  2 16  9  7 15  3 11 10  6     10984
  1  5  3  4  2 10 11 15  6  7 14 12 16  9 13  8     11612
  1  5 14  2  8 15  9 16  7 11 13 10  6 12  4  3     11687
  1 13 16 10 11  5  8  6  4  2  9 14  3 12  7 15     11777
  1 14 13  9  7  2  3  6 16  8 11  4 15  5 10 12     11938
  1 14  6  2  7 10  8  5  4 15 12 16 11 13  9  3     11964
  1  3  6  2 13 12  4 15  7  5 10 14  8 16 11  9     12400
  1  8 11 15  3  6  2 12  9 10 16  5  7 13 14  4     12503
  1 12  9 11  8  5 13  2 10  4  6 16 15  7  3 14     12943
  1 11  6  2  8 10  3  4  9 12 13 14 16  7 15  5     13036
  1  4 10  6  7 13  3 16 12  8  2 14  9 15  5 11     13173
   Önerilen rota
ELAZIG, GAZIANTEP, KIRIKKALE, BITLIS, BATMAN, MARDIN, ERZURUM, SIRNAK,
NEVSEHIR, ESKISEHIR, MUGLA, BILECIK, MUS, BINGOL, ERZINCAN, BALIKESIR
      1000. nesil rota ve toplam mesafe uzunlu§u
  1  7 10 11  4 16  9  2  3 12  8 13 14 15  5  6     5079
  1  7 10 11  4 16  9  2  3 12  8 13 14 15  5  6     5079
  1  7 10 11  4 16  9  2  3 12  8 13 14 15  5  6     5079
  1  7 14 15  5 13 10 11  4 16  9  2  3 12  8  6     8171
  1  7 10 13 14 15  9  2  3 12  8 11  4 16  5  6     8511
  1 12  8 13 14 16  9  7  3 15 10 11  4  2  5  6     8520
  1  7 10  2  3 12  8 13  4 16  9 11 14 15  5  6     8602
  1  7  2  3 12  8 13 14 11  4 16  9 10 15  5  6     8813
  1 12  8  2 13 14  9 15 10 11  4 16  3  7  5  6     9262
  1  7 10 11 14 16  9  2  3 12  8 13  4 15  5  6     9513
  1  2 13  3  7  4 14 15  9 12  8 11 10 16  5  6     9675
  1  2  3 12  8  7 14 16 10 11  4 15  9 13  5  6     10081
  1 14 15 11 12  2  9  8 13  7 10  3  4 16  5  6     10911
  1  9  8 11 14 16 10  4  7 13  3  5 12 15  2  6     11216
  1  6  3  7  5 16  8 10  9 12 13  2 14  4 11 15     12267
  1  2  6  7 15 16  4  5  3 12 11 14 13  8 10  9     12543
   Önerilen rota
ELAZIG, MUGLA, BALIKESIR, BILECIK, ESKISEHIR, KIRIKKALE, NEVSEHIR,
ERZINCAN, ERZURUM, BINGOL, MUS, BITLIS, SIRNAK, BATMAN, GAZIANTEP,
MARDIN
```