GEZGİN SATICI PROBLEMİNİN ÇÖZÜMÜNE YÖNELİK ALGORİTMİK YAKLAŞIMLAR

Serçin ÖZKAN

YÜKSEK LİSANS TEZİ
ENDÜSTRİ MÜHENDİSLİĞİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HAZİRAN 2010
ANKARA

GEZGİN SATICI PROBLEMİNİN ÇÖZÜMÜNE YÖNELİK ALGORİTMİK YAKLAŞIMLAR

Serçin ÖZKAN

YÜKSEK LİSANS TEZİ
ENDÜSTRİ MÜHENDİSLİĞİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HAZİRAN 2010
ANKARA

Serçin ÖZKAN tarafından hazırlanan “GEZGİN SATICI PROBLEMİNİN ÇÖZÜMÜNE YÖNELİK ALGORİTMİK YAKLAŞIMLAR" adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Prof. Dr. Orhan TÜRKBEY …….
Endüstri Mühendisliği, Gazi Üniversitesi

Bu çalışma, jürimiz tarafından oy çokluğu ile Endüstri Mühendisliği Anabilim Dalında Yüksek Lisans olarak kabul edilmiştir.

Prof. Dr. Serpil EROL …….
Endüstri Mühendisliği, Gazi Üniversitesi
Prof. Dr. Orhan TÜRKBEY …….
Endüstri Mühendisliği, Gazi Üniversitesi
Prof. Dr. Hasan BAL …….
İstatistik , Gazi Üniversitesi

Tarih: 23/06/2010

Bu tez ile G.Ü. Fen Bilimleri Enstitüsü Yönetim Kurulu Yüksek Lisans derecesini onamıştır.

Prof. Dr. Bilal TOKLU …….
Fen Bilimleri Enstitüsü Müdürü

TEZ BİLDİRİMİ

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

Serçin ÖZKAN

GEZGİN SATICI PROBLEMİNİN ÇÖZÜMÜNE YÖNELİK ALGORİTMİK YAKLAŞIMLAR

(Yüksek Lisans Tezi)

Serçin ÖZKAN

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Haziran 2010

ÖZET

Günlük hayatımızda da karşılaşabileceğimiz bir problem olan gezgin satıcı probleminin çözülmesi amacı ile birçok yöntem denenebilmektedir. Bu yöntemlerin çözüm süresi ve bulunan değerin yeterliliği önemli noktalar olarak karşımıza çıkmaktadır. Farklı türdeki GSP'ler için çok sayıda ve farklı çözüm yöntemleri vardır. Gezgin satıcı problemi alanında kullanılan önemli yöntemlerden biri de genetik algoritmalardır (GA). Kombinatöryel eniyileme ve global sezgisel arama alanlarında yoğun bir şekilde araştırılan ve çalışılan bir problem olan Gezgin Satıcı Problemi (GSP) için de halen araştırılmakta olan genetik algoritmaların kullanılması oldukça yenidir. Bu çalışmada genetik algoritmanın nasıl çalıştığı ve yöneylem araştırması problemleri arasında yer alan gezgin satıcı probleminin genetik algoritma ile çözümü üzerinde durulmuştur. Gezgin Satıcı Problemi için bir genetik algoritma geliştirilmiş, geliştirilen yöntemin avantajları ve dezavantajları var olan yöntemler temel alınarak anlatılmıştır. Amaç ise işlem süresi kısa ancak en iyi değeri garanti etmeyen bir çözüm yöntemi olan genetik algoritma ile gezgin satıcı problemin çözülmesidir.

Bilim Kodu : **906.1.148**
Anahtar Kelimeler : **Gezgin Satıcı Problemi, Genetik Algoritmalar**
Sayfa Adedi : **126**
Tez Yöneticisi : **Prof. Dr. Orhan TÜRKBEY**

ALGORITHMIC APPROACHES ABOUT THE SOLUTION OF TRAVELLING SALESMAN PROLEM

(M. Sc. Thesis)

Serçin ÖZKAN

GAZİ UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY

June 2010

ABSTRACT

Many method can be experienced to solve the traveling salesman problem, which can be met in our daily life. The solution time and the efficency of the existent value are considered as important points. There are many and different solution techniques for different kind TSP's. An important technique for TSP is the genetic algorithms (GA). It is very new to use genetic algorithms, which are still being researched, in solving traveling salesman problem, which is widely studied and researched problem in combinatorial optimization and global search heuristics. In this study how genetic algorithm works and the solution of traveling salesman problem, which is among the operational research problems, using genetic algorithm are explained. A genetic algorithm has been developed for traveling salesman problem, and the advantages and the disadvantages of the developed method are explained taking into consideration also the existing methods. The purpose is to solve the traveling salesman using the genetic algorithm mothod, which has the shorter solution time but does not ensure the optimum value.

Science Code : **906.1.148**
Key Words : **Travelling Salesman Problem, Genetic Algorithms**
Page Number : **126**
Adviser : **Prof. Dr. Orhan TÜRKBEY**

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren, desteğini esirgemeyen Hocam Prof. Dr. Orhan TÜRKBEY'e, yüksek lisans çalışmalarım boyunca beni destekleyen TÜBİTAK'a, yine kıymetli tecrübelerinden faydalandığım çalışma arkadaşım Sedat ERDOĞAN'a, çalışmalarımda bana olumlu katkılar sağlayan Yüksek Endüstri Mühendisi Selim SEVİNÇ'e, ayrıca bugüne kadar maddi ve manevi desteklerini esirgemeyen anne ve babama; en sıkıntılı zamanda çalısma azmimi yeniden kazandıran eşim Barış AÇAN'a teşekkürü bir borç bilirim.

Kaynak araştırmalarında yardımcı olan ismini sayamayacağım birçok arkadaşıma da ayrıca teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

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

Simge	Açıklama
C_{ij}	i. merkezden j. Merkeze 1 birim ürün göndermenin maliyeti
D_j	j. Tüketim merkezinin talebi
i	üretim (arz) merkezi
j	tüketim (talep) merkezi
Qij	i. Düğümden j. Düğüme olan talep
S_i	i. Üretim merkezinin kapasitesi
X_{ij}	i.Merkezden J. Merkeze Ulaştırılan miktar

Kısaltmalar	Açıklama
ARP	Araç rotalama problemi
ÇGSP	Çoklu gezgin satıcı problemi
ÇPP	Çinli postacı problemi
GA	Genetik algoritma
GSP	Gezgin satıcı problemi
TGP	Taşıt Güzergah Problemi

1. GİRİŞ

İşletmeler, yoğun rekabetin olduğu müşteri odaklı pazarlarda var olabilmek için ve rekabetle mücadele edebilmek için mamullerin kalitesini yükseltirken, maliyetlerini minimize etmek durumundadırlar. Bu tez çalışması kapsamında da işletmelerin toplam maliyetleri içinde önemli bir büyüklüğe sahip olan maliyet kalemlerinden olan dağıtım maliyetlerinin ele alınmasında kullanılmakta olan araç rotalama problemi kapsamında gezgin satıcı problemi uygulamasının sezgisel yöntem ile çözümü ele alınmıştır

Gezgin Satıcı problemi matematiksel programlama problemlerinin bir türüdür. Gezgin satıcı problemi (GSP) verilen N düğüm (şehir) için, her düğüme bir kez uğramak şartıyla tekrar başlangıç düğümüne geri dönen en kısa (en az maliyetli) rotayı bulma problemidir[1]. Tanımlaması son derece kolay fakat çözümü NP-Zor bir problemdir[1]. Bir eniyileme problemi olan bu problemde bir çizge üzerine yerleştirilmiş noktalar ve aralarındaki maliyetler göz önüne alınarak her düğüme yalnız bir kere uğramak şartıyla en uygun maliyetle çizgedeki tüm düğümlerin dolaşılması olarak tanımlanabilir. Bu problemin çözümü bir Hamilton Döngüsü olarak da görülebilir [2]. Merkezi bir yerden seyahata başlayan bir gezgin satıcının gitmesi gereken şehirlere sadece bir defa uğradıktan sonra tekrar başladığı şehre en küçük toplam uzunluğa sahip bir yolculuk sonunda ulaşmasını sağlayan turu belirleme problemidir. Satıcı aslında yolculuk süresince uğrayabileceği şehirleri önceden belirleyerek süre kaybını ve maliyeti en küçük seviyede tutmayı hedefler.
Gezgin satıcı problemi için optimum ya da en yakın optimum sonucu elde etmek için pek çok sayıda farklı yöntem önerilmiştir[2].Gezgin satıcı problemini çözmek için kullanılan yöntemler doğrusal programlama ve dal-sınır [3-4] , sinir ağları [5], tabu araştırması [6], genetik algoritmalar gibi [7], yapay zeka yöntemleri şeklinde sıralanır.

Gezgin satıcı problemi çözümü en zor problemlerden biridir. Nedeni ise, en iyi turu bulabilmek için tüm turların eksiksiz gözden geçirilmesi yoğun işlem yükü getirir. N-şehirli bir tur için n! kadar olası yol vardır, ancak dejenerasyonlar yüzünden

birbirinden farklı çözümlerin sayısı, n!' den azdır. Burada, birbirinden farklının anlamı toplam tur mesafelerinin birbirinden farklı olmasıdır. Verilen bir tur için n şehrin hangisinden tura başlanacağı, toplam kat edilecek mesafe bazında, sonuca etki etmez. Bu dejenerasyon birbirinden farklı toplam tur sayısını n kadar bir faktörle azaltır. Benzer şekilde, verilen bir tur için, satıcının iki yönden hangisine doğru hareket ettiğinin de bir önemi yoktur. Bu da birbirinden farklı toplam tur miktarını 2 faktörü ile biraz daha azaltır. Böylece, n-şehirli bir tur için göz önüne alınacak n!/2n kadar farklı toplam tur vardır.

5 şehirli bir problem için 120/10 = 12 farklı tur vardır. Bu da bilgisayar ile çözümü kısa sürecek bir problemdir. Ancak 10 şehirli bir problem, 3628800/20 = 181440 farklı tura; 30 şehirli bir problem ise $4x10^{30}$ dan daha fazla olası farklı tura sahiptir. Şehir sayısı arttıkça çözüm uzayı katlanarak artmaktadır.

$$\frac{(n+1)! \ / \ 2(n+1)}{n! \ / \ 2n} = n \qquad (1.1)$$

GSP, "NP-Hard" olarak bilinen bir problem sınıfına dahildir. Ağır hesapsal yük yüzünden, optimizasyon problemlerinde çabuk bulunabilen bir sonuç, sık olarak, çok geç bulunabilen en iyi çözümden daha fazla kabul görmektedir .
GSP'yi çözmek için önerilen algoritmalar iki başlık altında toplanabilir[1]:

- Kesin Çözüm Veren Algoritmalar ("Exact algorithms")
- Sezgisel Algoritmalar ("Heuristics algorithms").

Kesin Çözüm Veren Algoritmalar

Kesin çözüm veren algoritmalar, genellikle, GSP'nin tamsayılı lineer programlama formülünden türetilen yaklaşımlardır. Aslında bu algoritmalar hesaplanabilirlik açısından pahalı olmaktadırlar[1]. Bu yaklaşıma örnek olarak "Branch & Bound" algoritması örnek olarak verilebilir. Diğer yöntemler olarak lineer programlam yöntemleri, dinamik programlama yöntemleri sayılabilir[8].

Sezgisel Algoritmalar

Kesin algoritmaların çalışması verimli olmayabilir. Bu durumda, ideal çözüme yakın bir çözümü, kesin algoritmaları kullanmadan da bulabiliriz. Pratikte sezgisel algoritmalar, kesin algoritmalara tercih edilmektedir[1]. GSP'yi çözen sezgisel algoritmaları üçe ayırabiliriz: Tur oluşturan sezgiseller, Turu geliştiren sezgiseller ve bu iki yöntemin melez şekilde kullanıldığı sezgiseller[1].

• *Tur oluşturan sezgiseller***:** Tur oluşturan algoritmaların ortak özelliği, bir sonuç buldukları zaman, bu sonucu geliştirmek için uğraşmamalarıdır[4]. Bu noktada algoritmaların çalışması sona erer. Bilinen tur oluşturan sezgisel algoritmalar şunlardır: En yakın komşu, Greedy, Ekleme sezgiseli ve Christofides algoritmalarıdır [9]. Bu algoritmaların en optimum değeri %10-15 arasıdır [9].

•*Turu geliştiren sezgiseller*: Bu algoritmalar turu geliştirmeyi amaçlamaktadırlar. Bu algoritmalar için örnek olarak 2-opt, 3-opt ve Lin-Kernighan gibi yerel eniyileme (optimization) algoritmaları örnek verilebilir,aynı zamanda Tabu araması, Genetik algoritmalar, Benzetim, Tavlama ve Karınca Kolonisi Algoritması gibi Yapay zeka yöntemleri de örnek olarak verilebilir.

•*Melez yöntemler*: Hem tur oluşturma hem de tur geliştirme sezgisellerini bir arada kullanan algoritmalardır. Bunlara örnek olarak "Yinelemeli ("iterated") Lin-Kernighan" örnek olarak verilebilir[10]. En başarılı sonuçlar melez yöntemlerden elde edilmektedir[1]. GSP'yi çözmek üzere en sık kullanılan evrimsel hesaplama yöntemi genetik algoritmalardır. Bir çok NP-tam problem için GA'nın oldukça başarılı meta-sezgisel bir yöntem olduğu ispatlanmıştır [11]. Bu problemler, olasılıklı algoritmalar sınıfına ait olmakla birlikte rasgele algoritmalardan çok farklıdır. GA evrimsel programlamanın en yaygın ve en çok kullanılan dalıdır. Türkiye dahil dünyada pek çok araştırmacı bu konuda çalışmaktadır. Son yıllarda genetik algoritmalara ilgi büyüyerek artmaktadır. Bu konu üzerine bir çok kitap basılmış ve buna ek olarak yılda iki kez genetik algoritmalar üzerine konferanslar yapılmaktadır. GA hem problem çözmek hem de modelleme için kullanılmaktadır.

Günümüzde genetik algoritmaların uygulama alanları genişlemektedir. Bunlardan bazıları: Atölye Çizelgeleme, Yapay Sinir Ağlar Tasarım, Görüntü Kontrolü, Elektronik devre tasarımı, Optimizasyon, Uzman sistemler, Paketleme Problemleri, Makine ve Robot Öğrenmesi, Gezgin Satıcı Problemi, Ekonomik Model Çıkarma v.b sayılabilir[12].

Canlıların yapılarında var olan birtakım özellikler sanal ortamlarda taklit edilerek modeller getirilmeye ve bu modellerle karşılaşılan problemlere çözümler bulunmaya çalışılmaktadır. Bu modellerin birisi olan genetik algoritmalar canlıların çevreye uyum ve genetik özelliklerinin araştırılmasıyla geliştirilmiştir[13].

Çalışma kapsamında 10 şehirden 100 şehire kadar değişen düğümlerden oluşan gezgin satıcı probleminin düğüm sayısının artmasına yönelik , büyük problem çözümünde oldukça başarılı bir meta-sezgisel olduğu kanıtlanmış olan Genetik Algoritma ele alınmıştır. Genetik algoritmaların özellikle çok noktalı problemlerde oldukça başarılı sonuçlar elde ettiği literatürde incelendikten sonra, genetik algoritmada kodlama ile program hazırlanmıştır.İlgili programda optimum çözüm sağlayan paket programlarda 30 şehirden sonrası için elde edilemeyen çözümler, 100 şehire kadar makul bilgisayar süresi ile çözdürülmüştür.

İlgili program çıktıları da benzer şekilde paket program sonuçları ile kıyaslanarak genetik algoritmanın pratikte kullanılmak üzere oldukça başarılı sonuçlar elde ettiği ortaya konulmuştur. Örneğin, 20 şehir için optimum sonuç 12 sn'de 3446 olarak elde edilirken , genetik algoritma bu sonucu 1 sn ile 9 sn arasında zaten yakalamaktadır.

Değerlendirilen sonuçlar ile de genetik algoritmaların günlük yaşamda da işletmelerde sıkça karşılaşılan bir problem olan gezgin satıcı probleminin çözümünde ne kadar başarılı olduğu gösterilmektedir.

2. DOĞRUSAL PROGRAMLAMA MODELLERİ

2.1. Doğrusal Programlama

Doğrusal Programlama; kaynakların optimal dağılımının, kaynakların seçenekli dağılımının, optimal üretim bileşiminin, minimum maliyeti veren girdi bileşiminin, en uygun karın ve en az maliyetin belirlenmesinde kullanılmaktadır.

Doğrusal programlama değişkenlere ve kısıtlayıcı şartlara bağlı kalarak amaca en iyi ulaşma tekniğidir.

2.1.1. Doğrusal programlama ile ilgili varsayımlar ve tanımlar

Doğrusal programlama modelinden tutarlı sonuçların elde edilmesi aşağıda ele alınacak varsayımlara bağlıdır.

Doğrusallık Varsayımı

Bu varsayım işletmenin girdileriyle çıktıları arasında doğrusal bir ilişkinin bulunduğunu gösterir. Üretim düzeyi artarken aynı oranda üretim girdileri de artar. Ayrıca amaç fonksiyonu açık bir şekilde matematik olarak ifade edilmelidir. Amaç fonksiyonunun doğrusal olabilmesi için karar değişkenleri X_j'lerin birinci dereceden ve C_j katsayılarının da sabit olması gerekmektedir.

Toplanabilirlik Varsayımı

Bu varsayım değişik üretim faaliyetlerine kaynak olan üretim girdilerinin toplamının her bir işlem için ayrı ayrı kullanılan girdilerin toplamına eşit olduğunu gösterir. Örneğin bir iş iki saatte, diğeri üç saatte yapılıyorsa, iki işi birden yapmak için beş saate gerek vardır.

Sınırlılık Varsayımı

Üretimde kullanılan kaynaklar sonludur. Bu nedenle üretime giren girdiler ile üretim miktarı kısıtlanır.

Negatif Olmama Varsayımı

Doğrusal programlamada yer alan temel, aylak ve artık değişkenlerin değeri sıfır ya da sıfırdan büyük olmalıdır.

Doğrusal modeller 1947 yılında George B. Dantzig, Marshall Wood ve arkadaşları tarafından ortaya atılmıştır[14]. Bundan sonra bu teknik kullanımı basitleştirilerek değişik alanlarda uygulanabilir hale getirilmiş ve böylece modern ekonomilerin önemli bir aracı haline gelmiştir.

Doğrusal programlama önce iktisatçılar tarafından 1758'de kullanılmaya başlamıştır. 1930'ları takip eden yıllarda iktisatçılar doğrusal programlama modelleri geliştirmişlerdir. Walras'ın genel denge sistemi üzerinde çalışan diğer bir isim Leontief input-output analizi ile Amerikan ekonomisi için sayısal bir model kurmuştur[15]. Bu model ABD Hava Kuvvetleri Komutanlığının planlaması amacıyla kurulmuş bir grubun üyesi olan Dantzig genel doğrusal programlama modelini formüle ederek SIMPLEX çözüm metodunu bulmuştur.

Doğrusal programlama yönteminin uygulanabilme şartları aşağıdaki gibidir:

•Amaç kesin ve açık olarak bilinmelidir. İstenilen maksimizasyon ve minimizasyon hedefi bulunmalıdır.
•Amaca ulaşmada kullanılacak değişik yollar bulunmalı ve bunlardan biri amacı gerçekleştirmelidir.
•Amaç ve kısıtlamalar denklem veya eşitsizlikler şeklinde ifade edilmelidir.
•Kaynaklar sınırlı olmalıdır.
•Problemdeki değişkenler birbiriyle ilişkili olmalıdır.

2.2. Ulaştırma Modelleri

2.2.1.Ulaştırma modelinin tanımı ve gelişimi

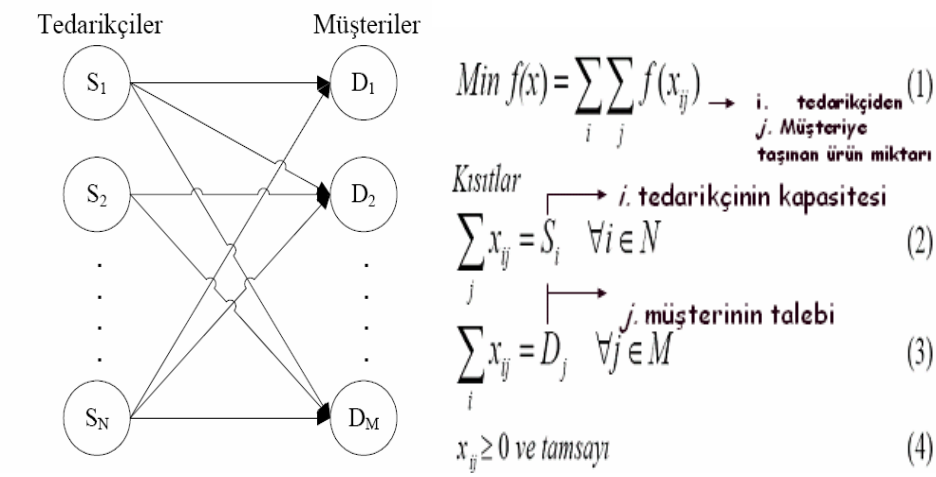
Ulaştırma modelleri, üretimdeki malların belli hedefe veya hedeflere minimum maliyetle gönderilmesini amaçlayan, doğrusal programlama modellerinin özel bir türüdür. Amaç, üretim merkezlerinden dağıtım merkezlerine mallar dağıtılırken, bir işi minimum maliyetle gerçekleştirmektir.

Ulaştırma modelleri, doğrusal programlama modelleriyle benzer biçimde çözülebilir.

- Üretim ve tüketim merlezleri arasındaki optimal mal dağıtımının belirlenmesinde
- İşlerin makinalara dağıtımında
- Üretim planlamada
- Şebeke ağı problemlerinde
- Tesis yeri seçiminde

Ulaştırma problemleri ile çözüme ulaşılabilmektedir.

2.2.2. Ulaştırma probleminin matematiksel modeli



Şekil 2.1. Ulaştırma şebekesi ve ulaştırma modeli

Amaç fonksiyonu (1) toplam ulaştırma maliyetini en küçüklemektedir. (2) nolu kısıt tedarikçilerden yapılacak olan toplam taşımanın tedarikçi kapasitesine eşit olmasını sağlamaktadır. (3) nolu kısıt benzer şekilde, her müşteriye yapılacak olan toplam taşımanın müşteri talebine eşit olmasını sağlamaktadır. İşaret kısıtı (4) ise karar değişkenlerinin pozitif ve tamsayı olmasını garantilemektedir.

Eğer ilgili ulaştırma tablosunda Toplam Arz=Toplam Talep eşitliği sağlanıyorsa, ulaştırma problemi dengededir. Eğer eşitlik sağlanmıyorsa, ulaştırma probleminin çözümüne başlamadan önce, problemin dengeye getirilmesi gerekir. Bunun için;

- Talep < Arz ise, probleme yapay talep merkezi eklenir ve sanki eklenen merkeze de mal gönderiliyormuş gibi düşünülür.
- Talep > Arz ise, probleme yapay üretim merkezi eklenir ve yapay merkezden de eksik olan talep karşılanabilirmiş gibi düşünülür[16].

Ulaştırma problemleri başlangıç temel uygun çözümüne ulaşmak için başvurulacak yöntemler şöyle sıralanabilir[17].

- Kuzey-Batı köşesi yöntemi
- En küçük maliyet yöntemi
- Sıra veya sütun en küçüğü kullanımı yöntemi
- Vogel Yaklaşım yöntemi (VAM)

2.3. Taşıma Ve Dağıtım Modelleri

Taşıma hem maliyetli bir iş hem de çevresel, finansal, zamansal kaynakları fazla kullanan bir lojistik işlevidir. Taşıma karayolu, havayolu, suyolu ve boru hattı ya da bunların birlikte yapıldığı intermodal biçimde olabilir.

Taşıma ve dağıtım modellerinin bir diğer türü, bir şebekedeki ara noktaların varlığını dikkate alıp incelemektedir. Bir şirket ürünlerini fabrikalarından bölgesel depolarına,

oradan da daha küçük bölgesel depolara ve satış noktalarına taşıyabilir. Böyle bir durumda bölgesel depolar taşıma ve dağıtım noktalan olarak işlev görür. Daha genel ifadelerle, bir şebekede malzeme alıp gönderen herhangi bir nokta taşıma ve dağıtım noktası olarak adlandırılır. Literatürde sadece malzeme gönderen noktalara kaynak, sadece malzeme alan noktalara ise hedef adı verilmektedir[18].

2.4. Şebeke Modelleri

Şebeke sözcüğü ok şeklinde çizilen ve şebeke planlama tekniklerinin esas kısmını oluşturan, birbirine bağlı faaliyetlerin genel olarak şekiller ya da bir diyagram olarak gösterilmesinden gelmektedir. Buna ilaveten,şebeke ise ayrıntılı olarak şu şekilde tanımlanabilir. Faaliyet ve olayların aslında birbirleri ile olan bağlantı ve ilişkilerini planlamaya bağlantılı gösteren, program amacına ulaşabilmek için gerekli faaliyet ve olaylardan meydana gelen şemaya şebeke denir.

Şebeke analizi ise bir planlama tekniği olup genellikle büyük ölçekli projelerin planlanmasında kullanılır. Bir noktadan diğer noktaya olan en kısa yolun bulunmasında, inşaat planlaması, yeni ürünlerin pazarlamasının programlanmasında, belli sistemlerdeki maksimum akışın (Örneğin, trafik akışı, sıvı akışları v.b.) bulunmasında, büyük çaplı ihalelerin hazırlanmasında, petrol dağıtım sistemleri telekomünikasyon uygulamaları, bilgisayar şebekeleri ve havayolu programlarının yapılması gibi çok değişik alanlarda şebeke analizinin kullanılması mümkündür.

3. TEMEL GÜZERGAH PROBLEMLERİ

Taşımacılık giderleri genellikle toplam lojistik giderlerinin üçte biri veya üçte ikisi arasında değişmesi sebebi ile, taşımacılık teçhizatının ve personelinin maksimum kullanımıyla etkinliğin geliştirilmesi ve arttırılması önemli bir sorun olarak karşımıza çıkmaktadır. Mamullerin taşıma sürelerinin uzunluğu, belli bir zaman içerisinde bir araçla yapılabilen yükleme miktarına ve bütün yüklemeler için harcanan toplam taşıma masraflarına yansır. Taşımacılık maliyetini azaltmak ve müşteri memnuniyetini artırmak için "bir aracın zamanı veya mesafeyi minimize edecek şekilde takip etmesi gereken en iyi karayolu, demiryolu, denizyolu veya havayolu güzergahını bulmak" sıkça rastlanan bir problemdir[19].

Araç Rotalama Problemleri (ARP) depolardan müşterilere, talep edilen ürünlerin araç filoları vasıtasıyla taşınması olarak düşünülebilir. ARP'ler çözülürken tüm müşterilere hizmet veren en iyi rota kümeleri, operasyonel kısıtlar (araç kapasitesi, zaman kısıtları, sürücülerin maksimum çalışma zamanları gibi) dikkate alınarak bulunur. Günümüzde ARP'ler tedarik zinciri ve dağıtım sistemlerinin anahtar konusudur. Bu nedenle dinamik, stokastik ve daha fazla kısıta sahip araç rotalama modellerine ilgi hızla artmaktadır. Böylece ARP'ler kompleks yapıya sahip amaç fonksiyonlarıyla ifade edilmektedir.

Literatürde yer alan çalışmalar ise; farklı özellikler içeren ARP problemlerinin modellenmesinde, bu problemlerinin optimum çözümlerinin araştırılmasında farklı çözüm algoritmalarının kullanılması ve gerçek hayattaki çeşitli sorunların çözümü için uygulamalar yapılması şeklindedir [20,21].

Araçların ne zaman ve hangi sıra ile servis yapacakları hususunda ise herhangi bir kısıtlama yoktur. Amaç her araç için mümkün olan en düşük maliyetli güzergahı belirlemektedir. Burada güzergah, bir aracın servis esnasında sırası ile dolaşması gereken noktaların birleşiminden oluşan yolu belirtmektedir [22].

Temel Taşıt Güzergah Problemleri (TGP) yedi başlık altında gruplandırılmakla birlikte, en çok kullanılan model olarak Gezgin Satıcı Problemleri karşımıza çıkmaktadır. Diğer yöntemlerin ise, Gezgin Satıcı Probleminin türevleri olarak değerlendirilmesi daha uygun olacaktır [23].

3.1. Gezgin Satıcı Problemi (GSP)

Bu tip problemler ise aslında ayrı başlangıç ve bitiş noktaları olan problemlerin geliştirilmiş bir şeklidir. Aracın başlangıç noktasına dönmesi, turun tamamlanması için şarttır. Bu da duruma karışık bir boyut kazandırmaktadır. Amaç ise toplam seyahat süresini ve mesafeyi minimize edilecek şekilde noktaların hangi sırayla dolaşılacağının bulunmasıdır. Bu çeşit problemlere örnekler; dağıtım kamyonlarının bir depodan satış noktalarına gidiş ve dönüş güzergahlarının belirlenmesi, satış noktalarından müşterilere dağıtım yapan kamyonların mahalli tur dizaynı, okul otobüslerinin, gazete dağıtım ve çöp kamyonlarının güzergahlarının belirlenmesi gibi durumlardır[24]. Bu örnekleri çeşitlendirmek mümkündür. Bu tip problemler genel olarak "Gezgin Satıcı Problemi" olarak adlandırılmaktadırlar [25].

Gezgin satıcı problemleri için bugüne kadar birçok çözüm metodu denenmiştir. Ancak bu şekilde, çok nokta içeren özel problemler için manuel olarak optimal güzergahı bulmak pek pratik görünmemektedir. Gerçek boyutlardaki bir problem için bile en hızlı bilgisayarlarda dahil olmak üzere optimizasyon işlem süreleri bir hali uzun olmaktadır[19].

Gezgin satıcı problemi, başladığı noktaya tekrar dönmek koşulu ile n-1 sayıda başka yerleşim yerlerine uğrayan satıcı ile ilgilidir.Gezgin satıcı probleminde amaç, sayısı n olan yerleşim yerlerinden bir satıcının n-1 sayıdaki kente en kısa sürede uğrayarak başladığı kente dönmesini sağlayacak bir gezi planını hazırlamaktır.

Böyle bir gezi planını sağlayabilecek yöntem aşağıdaki adımları içermektedir.

Adım 1 : Verilen problemin maliyet matrisinde boş gözler var ise bu gözlere çok büyük değerli sayı yerleştirir. Sonra matriste yer alan en küçük değerli eleman daire içine alınarak gezi planının halkasına eklenir.

Adım 2 : Adım 1 de daire içine alınan elemanın satırında ve sütununda bulunan tüm diğer elemanların yerine çok büyük değerli sayılar yerleştirilerek yeni bir maliyet matrisi teşkil edilir.

Adım 3 : Adım 2 işlemi ile ulaşılan maliyet matrisinde daire içine alınmayan en küçük değerli eleman işaretlenir ve sonra henüz tamamlanmamış gezi planında karşılık olduğu halkaya deneme yönünde eklenir. Sonuçlanan gezi uygun değil ise işaretlenen maliyet daire içine alınarak Adım 5 e geçirilir.

Adım 4 : Eğer sonuçlanan gezi planı uygun değil ise, en son halka plandan çıkarılır ve onun maliyeti yerine daha önce verilen büyük değerli maliyet verilerek Adım 3' e geçilir.

Adım 5 : Gezi planı tamamlanmış ve önce bulunan değerden daha iyi ise kabul edilir. Aksi halde Adım 2 işlemlerine geçirilir. Boş bırakılan herhangi bir yerleşim yeri Adım 2 deki işlemler ile yeniden geriye bırakılmaz ve herhangi bir yerleşim yeri gezi planına girdi ise tekrar plana girmez.

Bu tez çalışması kapsamındaki çalışma Gezgin Satıcı Problemi üzerine olacağından bu konu 4. Bölümde daha detaylı ele alınmıştır.

3.2. Çinli Postacı Problemi (ÇPP)

Amaç minimum maliyetle her durağa en az bir defa uğramaya imkan verecek çözümü bulmaktadır. Glover ve Murty bu problemi çözecek algoritmayı 1967 yılında geliştirmişlerdir. Çinli postacı problemi, ilk olarak 1962 yılında Çinli bir matematikçi olan Mei-Ko Kwan tarafından incelenmiştir. Problem, bir postacının postaneden aldığı mektupları mümkün olan en kısa yoldan şehirdeki tüm sokaklara uğrayarak

dağıtmak istemesiyle ortaya çıkmıştır. Mektupların dağıtımından başlayarak sonra postacı başladığı nokta olan postaneye geri dönmek zorundadır[26]. ÇPP, birleşi en iyilemenin yine temel problemlerinden biri olan gezgin satıcı problemine (traveling salesman problem/GSP) benzerlik göstermesine rağmen önemli farklılıklara sahiptir. GSP, bir düğüm rotalama problemi olup çizgedeki her bir düğüme yalnızca bir kez uğramak koşuluyla en kısa turun (Hamilton turun) bulunmasıdır. Tanımlarından da anlaşılacağı gibi ÇPP ve GSP arasındaki temel farklılık; ÇPP'nde bir çizgenin düğümleri yerine bu düğümleri birbirine bağlayan ayrıtlardan geçilmesi ve bunun da en az bir kez gerçekleşmesi şartıdır[27]. ÇP probleminin çizgesinde eğer bir Euler tur elde edilemiyorsa turun tamamlanabilmesi için ayrıtlardan birden fazla geçilmesi gerekmektedir.

Bu tip problemler gerçek hayatta yer alan; mektup dağıtımı, yol bakımı, atık veya çöp toplama işlemleri, kar temizleme çalışmaları, elektrik sayaçlarının okunması, polis devriye araçlarının rotalarının belirlenmesi ve otobüs çizelgelemesi gibi geniş uygulama alanlarına sahiptir [28].

3.3. Çoklu Gezgin Satıcı Problemleri (ÇGSP)

Gezgin satıcı problemlerinin birden fazla araç için uygulandığı durumlardır. Bir filoda bulunan M sayıda araç bir depodan çıkacak ve aynı depoya geri dönecektir. Her aracın en az bir düğüm noktasına uğraması dışında, araçların uğramak zorunda olduğu düğüm sayısında bir kısıt yoktur.

ÇGSP'ler gerçek hayatta kolaylıkla kullanılabilirler. Örneğin böyle bir model kullanılarak belediye otobüslerine ya da herhangi bir mal dağıtımı yapan araçlara güzergah belirlenebilir.

3.4. Tek Depo, Çok Araç ve Çok Duraklı Dağıtım Problemleri

Klasik güzergah problemleridir. Bu yöntem, merkezi bir depodan hareketle, bütün noktalara servis yapan araçların, en kısa mesafeyi dolaşarak tekrar aynı depoya

dönmelerine imkan verecek şekilde dağıtım yollarını belirler. Her noktadaki talebin deterministik olarak belirlendiği ve her aracın belli bir kapasitesi olduğu kabul edilmektedir. Golden tarafından bu problem ile ilgili kapsamlı formülasyonlar yapılmıştır [24].

3.5. Çok Depo, Araç ve Çok Duraklı Dağıtım Problemleri

Bir araç filosu, birden çok depodan yola çıkmaktadır. Bu modelde klasik TGP'ne ilave bir kısıt da her aracın aynı depodan çıkıp aynı depoya dönmesi olarak eklenmiştir. Gillett ve Johnson, atama-süpürme (assignment-sweep) yaklaşımıyla bu tür problemleri iki aşamalı şekilde çözmüşlerdir. Birinci aşamada dağıtım noktaları depolara atanmaktadır. Daha sonra ise her depo ve ona atanmış dağıtım noktaları için klasik güzergah problemlerinin çözümü uygulanmaktadır. Her iki aşama birbirinden ayrı olarak ele alınmaktadır.

3.6. Tek Depo, Çok Araça ve Tahmini Talepli Dağıtım Problemleri

Klasik TGP gibidir. Ancak burada tek fark dağıtım noktalarındaki talebin kesin olarak bilinmemesidir. Burada bir olasılık dağılımından yararlanılmaktadır. Bu metod en az TGP kadar zordur. Kısıtların doğrusal olmayışı ve tahmini taleplerin ortaya çıkardığı amaç fonksiyonu durumu iyice zorlaştırmaktadır. Bu sebeple, bu tür problemlerin çözümü için sezgisel (heuristîc) çözüm metodları kullanılmaktadır.

3.7. Kapasite Tahditli TGP

Bu problemde bütün duraklar birer depo gibi düşünülür. Problem, herbir müşterinin bir araç tarafından sadece bir kez ziyaret edilmesi esasına göre oluşturulacak araç rotalarının belirlenmesini modellemektedir. Aynı depoya gidip gelen araçların kısıtlı olan kapasiteleri asla aşılmamalıdır. Amaç araçların ve güzergahların toplam maliyetini asgarileştirmektir.

Şebekedeki her yolun Q kadar talebi vardır ($Q_{ij}>0$). Bu Q talebinin karşılanması içinse W kapasiteli bir araç filosu görevlendirimektedir. Bu araçlar bir depodan geçerek deponun devamındaki yolun talebini minimum maliyetle karşılarlar .

3.8. Maliyet Tahditli TGP

Kapasite tahditli TGP problemi ile aynıdır. Tek farkı araç kapasite tahditinin yerine araç maliyet tahditinin almasıdır.

Bu tür problemlerde alternatif düğüm noktaları arasından asgari düzeyde ihtiyaç duyulan sayıda durak noktası belirlenir. Her durağa sadece bir kez uğranacak şekilde güzergahların her araç için ayrı ayrı saptandığı toplam güzergahın aynı depoda başlayıp bittiği durumlar dikkate alınarak modelleme yapılır.

3.9. Maliyet ve Kapasite Tahditli TGP

Kapasite tahditli ve Maliyet Tahditli problemlerin çözüm tekniklerinde belirtilen hususlar bu tür problemlerin çözümünde de kullanılmaktadır. Ancak her iki kısıt da aynı anda karşılanacak şekilde model kurulmaktadır.

3.10. Zaman Tahditli TGP

Araç rota problemlerinin geliştirilmiş bir halidir. Bu problemde, her müşteriye belli bir zaman aralığında hizmet verecek şekilde yapılan planlama esası üzerinde durulur.

3.11. Araçlar İçin Güzergah Tespiti Ve Zaman Planlaması

Araçlar için güzergah tespiti ve zaman planlaması, aslında güzergah problemlerinin kapsamca genişletilmiş bir halidir. Bu konuda artık daha gerçekçi kısıtlamalar da mevcuttur;

- Her durakta, o durakta inen personel miktarı kadar insan araca binebilir.
- Birçok araç için farklı yük (ağırlık) taşıma kapasiteleri olabilir.
- Güzergah üzerinde izin verilen maksimum toplam araç kullanma süresinden sonra en az 8 saat dinlendirilmelidir (Trafik Yasası).
- Duraklardan personel toplanması ve duraklara personel dağıtılması günün belli zamanlarında yapılacaktır.
- Güzergah üzerindeki toplama işleri, dağıtımlar güzergaha yapıldıktan sonra gerçekleştirilecektir.
- Araç sürücülerine günün belli vakitlerinde dinlenme ve yemek ihtiyaçları için mola verilebilir.

Bu kısıtlamalar, problemi daha karmaşık ve zor hale getirmektedir.

3.12. Güzergah Planlama Prensipleri

Karar vericiler, aşağıdaki ilgili prensipleri uygulayarak en iyi güzergahları tespit edebilmekte ve zaman planlamalarını geliştirebilmektedirler:

- Duraklara yükleme için gönderilen kamyonların kapasitesiyle, duraklardaki miktar mümkün olduğu kadar birbirine yakın olmalıdır.
- Farklı günlerde uğranılacak duraklar, birbirine daha yakın duraklardan oluşan kümeler-gruplar oluşturmak suretiyle düzenlenmelidir.
- Güzergahlar, depoya en uzak duraktan başlayarak depoya gelecek şekilde oluşturulmalıdır.
- Bir güzergahdaki durakların sıralanışı geniş bir damla gibi olmalıdır.
- En etkili güzergahlar ise, mevcut olan en büyük araçların kullanılması ile oluşturulabilir.

Duraklardan toplama işlemi, dağıtım güzergahlarının içinde düşünülmelidir.

4.GEZGİN SATICI PROBLEMİ

4.1. Gezgin Satıcı Probleminin Uygulamaları

GSP'nin n müşteri veya şehir arasında en kısa Hamilton çeviriminin bulunmasına dayalı çeşitli uygulamaları bulunmaktadır. Buna ek olarak GSP, ilişkisiz gibi görünen bazı problemlerinde GSP ' ye göre formüle edilerek çözülmesine yardımcı olmaktadır. Ong and Huang, bu tür problemlerden bazılarına Lenstra and Rinnooy Kan' in GSP formülasyon ve algoritmalarını kullanarak bazı sezgisel yöntemlerden daha iyi sonuçlar bulduğunu belirtmektedir [29]. GSP' ye göre formüle edilerek çözülebilen ve çok kullanılan uygulama alanları şunlardır :

İşlerin sıralanması

n adet iş tek bir makinada ardışık olarak gerçekleştirilmek istenilsin. Her bir iş birbirinden farklı olduğundan, herhangi bir i işi tamamlandıktan sonra j işinin gerçekleştirilebilmesi için bazı hazırlıklara (makinanın basınç, ısı ayarı vb) gerek duyulur. Herhangi bir iş kendisinden önceki bir iş için yapılan hazırlıkların bir bölümünü kullanabilirse, bu geçiş süresi daha kısa olacaktır. Cij≠Cji olduğu göz önüne alınır ise, problemin aslında asimetrik GSP olduğu düşünülebilir. Bu problemlerde amaç, minimum geçiş süresi kullanarak, n adet işin tamamlanmasıdır.

Devre levhaların delinmesi

Bazı üretim sektörlerinde, tahta veya metal levhalar üzerine delik açmaya gerek duyulmaktadır. Baskılı devre levhası imalat edenler, levha üzerinde daha sonrası için elektronik parçaların lehimleneceği iğnelere karşılık gelen yerleri delerler. Bir baskılı devre levhasına yaklaşık 500 delik açmaya gerek duyulabilir. Burada amaç, minimum zaman alan delik açma sırasının belirlenmesidir.

Bilgisayar bağlantısı

Bu tür bir problem, bilgisayarların veya diğer sayısal (digital) elektronik sistemlerin tasarımında ortaya çıkabilmektedir. Bir sistem, aslında üzerine iğnelerin (pin) monte edilmiş olduğu birimlerden (modüle) meydana gelmektedir. Burada amaç, birimlerin üzerinde yer alan iğneleri minimum uzunlukta tel kullanımıyla karşılıklı olarak bağlanmasıdır. Ancak bu bağlantıyı sınırlayan bazı kısıtlar bulunmaktadır. Örneğin, iğnelerin küçük boyutlu olması nedeniyle, bir iğneye en fazla iki tel tutturabilmektedir. Her iğne grafiğin bir düğümünü, Cij ise iki iğne arasındaki uzaklığı gösterirse, problem bir GSP' ye indirgenir. Eğer bir iğneye ikiden fazla tel bağlanabilseydi, problem minimum yayılan ağacın bulunmasıyla çözülebilirdi.

Duvar kağıdı kesimi

Bir birim uzunluğunda bir modelin her defasında tekrar ettiği uzun bir top duvar kağıdı düşünülsün. Bu duvar kağıdı topundan n adet sayfa kesilmek istenilsin, i. sayfasının modeldeki başlangıç noktası a_i bitiş noktası b_j olsun. Burada $0 < a_i < 1$ ve $0<bj <1$ sınırları geçerli olmaktadır, i. kağıdın hemen ardından j. kağıdın kesimi yapıldığında, oluşan kağıt ziyanı şu şekildedir:

$$C_{ij}= \begin{cases} \{a_j-b_i, & b_i<=a_j \\ \{1+ a_j - b_i , & b_i > a_j \end{cases} \qquad (4.1)$$

Burada amaçlanan, toplam kağıt ziyanını minimum yapan kağıt kesme sırasının bulunmasıdır. Bu problemin GSP olarak tanımlanabilmesi için ise, (n+1). kağıt kesileceği düşünülür. Burada C_i, $n+1 = 0$ ve C_{n+1} ,$j = 0$, tüm i, $j = 1,...,n$ için geçerlidir. Kesim işleminin başında b_{n+1} , topun son konumu olarak tanımlanır. Son kağıdın kesiminden sonra, topun ilk pozisyona getirilmesi için, son bir kesimin yapılacağı düşünülür. Böylece, $a_{n+1} =b_{n+1}$ olur ve Cij; i, $j = 1,..., n+1$, "Eş. 4.1"' deki gibidir.

Toplama ve dağıtım problemleri

Farklı yerleşim yerlerinde periyodik dağıtım veya toplama işlemini gerektiren bir çok problem, GSP olarak düşünebilir. Örneğin, paket veya mektup dağıtımı yapan posta servisleri, her müşterisine sadece bir kez uğrayarak ve toplam seyahat edilen süreyi minimum tutarak zaman kazanabilir. Posta servislerinin telefon kulübelerin de yaptığı periyodik toplama işlemi de GSP olarak düşünülebilir. Yol ağı, çok sayıda tek yöne sahip caddeler içermedikçe, Cij = Cji olacaktır.

4.2. Gezgin Satıcı Probleminin İlişkili Olduğu Problemler

GSP'nin grafik teorisinde yer alan bazı problemler ile yakın ilişkisi bulunmaktadır.

4.2.1 Atama problemleri

C = (Cij) maliyet matrisine sahip bir atama problemi ele alalım,

$$Z = \sum_{j=1}^{n} \sum_{i=1}^{n} C_{ij} X_{ij} \qquad (4.2)$$

$$\sum_{i=1}^{n} X_{ij} = 1, \qquad j=1,\ldots, n \qquad (4.3)$$

$$\sum_{j=1}^{n} X_{ij} = 1, \qquad i=1,\ldots,n \qquad (4.4)$$

$$X_{ij} = 0, 1 \qquad i=j=1,\ldots,n \qquad (4.5)$$

olarak formüle edilebilir. Burada Xij, (i, j) değişkeni optimal çözümde kullanıldı ise 1, aksi halde 0 değerini almaktadır. “Eş. 4.3” ve “Eş.4.4”, her bir düğüme sadece bir değişkenin gelmesini ve o düğümden sadece tek bir değişkenin çıkmasını sağlayan derece kısıtlarıdır. Böylece her bir düğümün derecesi ikiyi geçmeyecektir. Atama probleminin çözümünde gezgin satıcı probleminden farklı olarak tek bir tur yerine

ise birkaç tur ortaya çıkabilmektedir. Örneğin n=4 olduğundan atama probleminden $X_{12}=X_{21}=X_{34}=X_{43}=1$ çözümü elde edilebilir. Bu sonuç ise "Eş. 4.3" ve "Eş.4.4" kısıtlarını sağlamasına karşın, alt tur oluşmasını önleyici kısıtların olmamasından dolayı GSP için uygun olarak kabul edilmeyen iki alt tur ile sonuçlanmaktadır. "Eş. 4.2" ve "Eş.4.5" eşitliklerine çözümünün ayrık turlar yerine, bir Hamilton çevrimli olmasını sağlayan ek kısıtların ilave edilmesi ise, GSP' nin formülasyonunu temsil etmek için kullanılabilir. Atama problemi, GSP' nin bir gevşetmesi veya diğer bir deyişle GSP için, alt turların oluşmasını engelleyen kısıtların eklenmesi ile atama probleminin kısıtlanmasıdır.

4.2.2. Minimum yayılan ağaç

Yayılan ağaç, ilgili grafiğin tüm düğümlerini birleştiren ağaç anlamına geliyor. Her Hamilton yolu aslında bir yayılan ağaçtır. Eğer yayılan ağaçda yer alan hiçbir düğümün derecesi ikiyi geçmiyorsa, bu yayılan ağaç bir Hamilton yoludur.
Minimum yayılan ağaç problemi GSP' nin gevşetmesi, GSP ise derece kısıtlarından dolayı minimum yayılan ağaç probleminin kısıtlanmasıdır.

4.3. Gezgin Satıcı Probleminin Matematiksel Modeli

Gezgin satıcı problemi için doğrusal programlama modeli aşağıdaki gibi formüle edilebilir.

$$X_{ij}= \begin{cases} 1, \text{Çözümde eğer şehir i'den şehir j'ye gidildi ise;} \\ 0, \text{diğer durumda.} \end{cases}$$

Gezgin satıcı problemi bir minimizasyon problemi olup amaç fonksiyonu aşağıda verilmiştir:

$$\text{Min } Z = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij} * X_{ij} \qquad (4.6)$$

Gezgin satıcı probleminde turun tamamlanması için gereken, her bir şehre uğramaktır. Problem için gerekli kısıtlar aşağıda verilmiştir.

$$\sum_{i=1}^{N} X_{ij} = 1 \qquad (j = 1,2,3.....N) \qquad (4.7)$$

$$\sum_{j=1}^{N} X_{ij} = 1 \qquad (i = 1,2,3.....N) \qquad (4.8)$$

C_{ij} = Şehir i den Şehir j ' ye olan Uzaklık;

Bütün $X_{ij} = 0$ veya 1 $i=j=1,...,N$ $i \neq j$ (4.10)

Kabul edilebilir ayrışmış alt tur çiftlerine karşı koymak amacıyla, n–1 tane yeni değişken $u_2, u_3,...,u_n$ ile karşılaşmaktayız.

$u_i \geq 0$ ve $(n - 1)^2 - (n-1)$ türevlenmesi ile yeni kısıt aşağıdaki gibidir.

$$U_i - U_j + n * X_{ij} \leq n - 1 \text{ için } i, j = 2,3...n \text{ ve } i \neq j \qquad (4.11)$$

Yukarıdaki kısıtlar dahilinde herhangi bir turun alt turlara bölünmesi ise mümkün değildir. Varsayalım ki, tur C_1 ile başlasın, alt tur olarak varsayılsın, böylece C_1 ile başlayan tur C_1' e bütün şehirler dolaşılmadan geri dönülerek son bulmaktadır. Ve bu da başka bir alt tura neden olmaktadır. C_1'e varılmadan önce tur $r \leq (n - 1)$ şehri içermelidir.

Kısıtlara; u_i'lerin bütününü çift elde edene kadar, her sıfırdan farklı x_{ij} 'lere r eklenir.

$$n * r \leq (n-1) * r$$

Aşağıda verilen bütün kısıtlar Gezgin satıcı probleminin herhangi bir sonucunda geçerlilik sağlamaktadır.

Geçerli kısıtların gösterimi için tanımlanan u_i değişkenleri yeterlidir. u_i verilen rotada uğranılan şehrin kaçıncı pozisyonda olduğunun göstermektedir.

$C_1 \rightarrow C_3 \rightarrow C_5 \rightarrow C_4 \rightarrow C_2 \rightarrow C_1$

$u_2 = 5$, $u_3 = 2$ ve saire.

Şimdi kısıtları hesaba katalım:

$$U_i - U_j + n * X_{ij} \leq n - 1 \quad \text{belirli } i \text{ için, } i \text{ ve } j;\ j \neq i. \tag{4.12}$$

Dikkat edilmesi gereken

$U_j \leq n$ olduğu zaman $U_i \geq 2$, öyleyse $X_{ij} = 0$,

$$U_i - U_j + n * X_{ij} \leq n - 2 \tag{4.13}$$

ve eşitsizliğimizi oluşturuyoruz.

Son durumda , $X_{ij} = 1$, öyleyse C_i den sonra C_j 'ye uğranılmıştır ve,

$$U_j = U_i + 1$$

$$Ui - Uj + nXij = Ui - (Uj + 1) + n = n - 1 \tag{4.14}$$

4.4. Algoritma Karmaşıklığı

Algoritma istenen cevabı üretirken gereksinim duyduğu süre, problem genişliğinin bir fonksiyonudur ve algoritmanın süre karmaşıklığı olarak adlandırılır. Problemin genişliği, bir grafik probleminde ayırt sayısı, matris çarpım probleminde çarpılacak en büyük matris boyutu ise, GSP'de ise satıcının uğrayacağı şehir sayısıdır.

Aslında probleme ait iki yada daha fazla algoritma var ise her bir algoritmanın gerektirdiği elemanter işlem sayısı, algoritmik performansın değerlendirilmesinde kullanılan ölçütlerden birisidir. Bu işlemler; toplama, çıkartma, çarpma, bölme ve karşılaştırma vb basit işlemler içerir.

Bir algoritmanın işlem süresi olan g(n) için değer,

$$g(n) <= c\, f(n) \tag{4.15}$$

eşitliğini sağlayan bir c>0 sabiti varsa, g(n) fonksiyonu O(n)) ile gösterilir. Örneğin bir algoritmanın işlem süresi m. dereceden,

$g(n)=a_m n^m+. \ . \ . \ +aın+a_0$ (4.16)

polinom ise,

$g(n)=On^m$ (4.17)

ile gösterilir[30].

Sabit sayıda işlem gerektiren program kesimlerinde algoritma karmaşıklığı O(1), matris toplamada $O(n^2)$, matris çarpanında $O(n^3)$, bazı sıralama yöntemlerinde ise $O(n\log_2 n)$' dir.

Algoritma karmaşıklığı analizinde, asimptotik davranış yani algoritmanın bir hayli büyük giriş verilerine uygulanması durumu ile ilgilenilir. Örneğin 34 367 291 n^2 karmaşıklığına sahip algoritma $O(n^2)$ ile gösterilmesine rağmen, n' nin büyük değerleri hariç $0{,}002n^5$ karmaşıklığı bulunan bir algoritma kadar etkili olamaz. Bu nedenle karmaşıklık analizi sadece asimptotik durumunda geçerlidir.

Algoritma karmaşıklığı, problem genişliğinin bir polinomu ile sınırlı ise, algoritma polinomiyal sınırlıdır. O(n), O(nlogn), $O(n^{1.5})$ ve $O(n^{100})$ bu tür karmaşıklıklara örnektir. Diğer algoritmaların pek çoğu üstel (polinomiyal olmayan) karmaşıklığa sahiptir. $O(2^n)$, O(n!), ve $O(n^n)$ üstel karmaşıklıklara örnek olarak verilebilir.

Problem genişliği arttıkça polinomiyal algoritmalar ise üstel algoritmalara oranla kullanılmaz duruma gelmektedir.

Çizelge 4.1. 'de değişik süresel karmaşıklıklara sahip algoritmaların, hesaplama sürelerini aslında nasıl etkilediği görülülebilmektedir.

Çizelgeye göre, üstel süre karmaşıklığına sahip algoritmalar için hesaplama süreleri, aslında polinomiyal süre karmaşıklığına sahip algoritmalara göre daha uzun olup, problem büyüklüğü ile daha da artmaktadır.

Çizelge 4.1' de çeşitli karmaşıklık fonksiyonlarına sahip algoritmaların hesaplama süreleri gösterilmektedir[30].

Polinomiyal sınırlı bir algoritmaya sahip olmayan problemler içerisinde en bilineni GSP'dir.

Çizelge 4.1 Hesaplama süreleri

Karmaşıklık fonksiyonları	Büyüklük			
	10	100	500	1000
1000n	0,01sn	0,1sn	0,5sn	1sn
100nlogn	0,003sn	0,06sn	0,45sn	1sn
n*	0,001sn	1sn	2dk	10dk
2n	0,001sn	40000yy		

Çok büyük problemlerde ise, optimal sonuç veren bir algoritmanın yardımıyla çözülemediğinden çözüm bulmak amacıyla sezgisel yöntemler kullanılır.

4.5. Çözüm Yöntemleri

GSP'nin çözüm yöntemleri ikiye ayrılmaktadır. Bunlar;

- Kesinlikle optimal çözümle sonuçlanan şehir sayısı fazla olan problemlerde aşırı işlem gerektirmelerinden dolayı, sonuca ulaşması güçleşen yöntemler,
- Her zaman optimal çözümle sonuçlanacağı kesin olmayan ancak daha az sayıda işlem gerektiren yöntemler.

Dal ve sınır yöntemleri 1., sezgisel yöntemler ise 2. sınıfta yer alır.

4.5.1. Optimal çözüm yöntemleri

Dal ve sınır yöntemlerinde öncelikle GSP'nin bir gevşetmesi çözülür. Buradan elde edilen çözüm GSP için uygun bir çözüm ise, yani bir Hamilton çevrimi veriyorsa optimal çözüm elde edilmiş oiur. Çoğu zaman ise yapılan gevşetme sonucunda elde

edilecek olan çözüm bir Hamilton çevrimi olmaz ve bir alt sınır elde edilir. GSP'ye uygun olabilecek bir çözümde, gevşetme sonunda bulunacak çözümde yer alan ayrıtlardan en az bir tanesi bulunmamaktadır. Bu yüzden tespit edilen değişkenler optimal çözümü ararken tur dışında bırakılmalıdır. Belirlenen değişkenlerin tur dışına zorlanması sonucunda elde edilecek olan çözüm, alt sınır değerlerinden daha düşük bir değere sahip olamayacaktır. Eğer alt sınır en azından bilinen herhangi bir uygun tur değeri kadar ise, kısıtlama sonucunda çözüm değeri artacağı için artık değişkenleri tur dışında bırakmaya zorlayarak inceleme yapmak gereksiz olacaktır. Değişkenleri kısıtlama işlemi ise daha iyi bir uygun tur buluncaya veya başka kısıtlamaların daha iyi olacağını gösterene kadar tekrar edilir.

GSP'nin optimal turu elde edildğinde bir tur yerine ayrık turlar elde edildiğinde, çözümü uygun hale getirmek için ayrık turlardan herhangi bir tanesinde yer alan değişkenler, sırasıyla tur dışında bırakılarak yeni alt problemler elde edilir. Her alt problem tur dışına zorlanacak değişkene büyük bir sayısal değer verilmesiyle elde edilen maliyet matrisine, Macar atama yöntemi uygulanarak çözülür. Eğer bir alt problemin çözümü GSP için uygun yani bir Hamilton turu veriyor ise optimal çözüm için bir üst sınır elde edilmiş olur. Herhangi bir alt problemden üst sınır değeri elde edildikten sonra, diğer alt problemlerin çözümüne geçilmektedir. Bu alt problemler ise tekrar Macar atama yöntemi ile çözülmektedir ve yeni çözüm değerleri bulunmaktadır. Bu çözüm değerleri GSP için uygun değilse ve bulunmuş olan üst sınır değerini aşıyorsa, artık bu alt problemlerdeki değişkenlerin dışarıda bırakılmasına gerek olmayacaktır. Çünkü yeni değer kısıtlamalar sonucunda elde edilecek olan çözüm değerleri, üst sınır değerlerinden aslında daha büyük olacaktır. Daha fazla dallanma olmadan o alt problemdeki işlemlerin kesilmesi ise kolaylık sağlayacaktır. Alt problem çözümlerinden üst sınır değerlerini aşmayan bir alt sınır değeri bulunuyor ise dallanma yapılarak yeni bir üst sınır değeri alınır. Bu işlemler sonucunda bulunan en küçük değere sahip Hamilton tur uzunluğu, optimal GSP'nin tur uzunluğudur.

Dal ve sınır yöntemleri GSP ile birlikte ilk olarak Little ve arkadaşları tarafından 1963 yılında kullanılmıştır [26]. Büyük problemleri dal ve sınır yardımıyla çözmek

oldukça zaman alıcı ve çoğu zaman olanaksız olduğundan, daha kısa sürede sonuca ulaşabilmeyi sağlayan sezgisel yöntemler kullanılır.

GSP'nin bazı özel durumlarında çözüm, polinomiyal sürede elde edilebilir. Bu tür problemlere şu örnekler verilebilir:

- C=(Cij)'nin üst üçgen matrisi yani C_{ij}=0 (tüm i >= j için) olduğu GSP'ler,
- GSP'nin duvar kağıdı kesme örneği,
- İş sıralama problemleri.

İş sıralama problemlerinde bir tuğla ocağında gerçekleştirilecek olan (n-1) adet iş mevcuttur, i.iş a_i gibi bir başlangıç sıcaklığına sahiptir, ve bi sıcaklığında bitmelidir. Diğer işlerde ise ocak başlangıç ısısının a_i ve son sıcaklığın b_i olması gerektiğini varsayar. Bu durumda problem GSP olarak,

$$Cij = \begin{cases} \int_{bi}^{ai} f(x)\,d(x) & bi <= ai \text{ ise} \quad (4.18) \\ \int_{bi}^{ai} g(x)\,d(x) & ai <= bi \text{ ise} \quad (4.19) \end{cases}$$

gibi formüle edilebilir.

Burada f ve g maliyet yoğunluk fonksiyonları olup, tüm x'ler için f(x) + g(x)>= 0 dır.

4.5.2. GSP'nin çözümünde kullanılan sezgisel yöntemlerin sınıflandırılması

Sezgisel yöntemlere, NP-tam sınıfında yer alan bir probleme aşırı süre kullanmadan uygun çözüm arandığında gerek duyulmaktadır. Sezgisel yöntemler çok çok hızlıdır ve çoğu kez problemin optimal çözümünü çok az geçen bir çözüm sağlar.

GSP'de şehir sayısı fazla olduğu için, optimale yakın çözüm sağlamış olan sezgisel yöntemlerden faydalanılmaktadır.

İyi bir sezgisel yöntemse aşağıda verilen özelliklere sahiptir[29].

- Çözüm optimale yakın olmalıdır.
- Optimalden sapan bir çözüm bulma olasılığı düşük olmalıdır.
- Uygun hesaplama süresinde çözüm vermelidir.
- Bilgisayardaki bellek gereksinimi küçük olmalıdır.

Literatürde en iyileme (optimizasyon) problemlerinin çözümünde kullanılabilecek birçok teknik mevcuttur. Bu teknikleri kullanarak doğrusal programlamalarda klasik yöntemlerle sonuca ulaşılabilirken, problemin boyutunun büyümesi ise bu çözümleri imkansız hale getirmektedir. Bu sebeple genetik algoritmalar, tabu arama, tavlama benzetimi gibi bir takım sezgisel arama yöntemleri geliştirilmiştir.

Son yıllarda kombinatoriyel problemlerin çözülmesinde sezgisel yöntem kullanımları önemli bir ölçüde artmaktadır. Jones ve diğerlerine göre aslında sezgisel yöntemlerin popülaritesinin 1991 yılı itibarı ile hızlı bir şekilde artış göstermesinin nedenlerini birinci olarak hesaplama gücünün iyi olmasından, ikinci olarak ise dönüştürülebilir olmasından oluşmaktadır. Sezgisel yöntemlerin en büyük avantajları arasında çözüm zamanının sayım (enumeration) tekniğine göre çok kısa olması ve her tür problem için kolay bir şekilde entegre edilebilmesidir. Dezavantajları ise bu yöntemlerin optimum çözümü garanti etmemesi ve iyi çözüm verebilmesi için bir çok parametrenin uygun bir şekilde ayarlanması gerekliliğidir.

Şehir sayısı arttığı durumlarda GSP'yi bir polinomiyal algoritma yardımıyla çözmek mümkün olmadığı için (NP-tam sınıfında yer alması sebebiyle) kısa sürede çözüm sağlayan çeşitli sezgisel yöntemler önerilmiştir. Bu sezgisel yöntemler, üç sınıfta toplanabilir:

- Tur kurma yöntemleri (tour construction procedures)
- Tur ilerletme yöntemleri (tour improvement procedures)
- Karma yöntemler (composite procedures)

Yaklaşık fakat optimal olmayan bir çözüm arandığında, tur kurma yöntemlerinden biri kullanılarak bir yaklaşık çözüm elde edilebilir. Tur kurma yöntemleri uzaklık matrisini kullanarak, çok kısa sürede yaklaşık çözüm bulabilmektedir. Tur ilerletme yöntemleri ise, herhangi bir tur kurma yöntemiyle kısa sürede elde edilmiş olan yaklaşık çözümle başlayıp, bu çözüme göre optimale daha yakın bir çözüm bulmayı amaçlar. Karma yöntemler, tur kurma yöntemlerinden biri yardımıyla başlangıç turu oluşturup, ilerletme yöntemlerinin bir ya da birkaçını kullanarak daha iyi bir tur oluşturmaya çalışır.

4.5.3. Sezgisel yöntemlerin değerlendirilmesi

Herhangi bir sezgisel yöntem ile bulunan yaklaşık çözüm değerlendirilmeye çalışıldığında ise, sezgisel yöntemlerin davranışını analiz edebilen yöntemlerden yararlanmak gereklidir. Bu şekilde o sezgisel yöntemin yardımıyla elde edilen çözümü değerlendirebilmek ancak mümkün olabilecektir. Sezgisel yöntemlerin davranışlarının iyi öğrenilmesi, aslında yeni bazı sezgisel yöntemler geliştimek için de önem taşımaktadır.Sezgisel yöntemlerin davranışını incelemek için üç yöntem vardır [31].

- Performans garantisi (en kötü durum analizi)
- Olasılıklı analiz
- Deneysel analiz

Performans garantisi, sezgisel yöntem kullanılarak elde edilen bir tur uzunluğunun, optimal tur uzunluğunu aslında ne kadar geçebileceğine ilişkin bir en kötü durum sınırı (worst-case bound) vermektedir. Olasılıklı bir analizde, bir sezgisel yöntemin değerlendirilebilmesi, şehir konumlarının bilinen bazı olasılıksal dağılımlara

uyduğu varsayımına bağlı olarak yapılır. Örnek olarak, şehirlerin birim uzunluk, genişlik ve yükseklikteki bir küp üzerinde üniform dağılan bağımsız rasgele değişkenler olduğu varsayılabilir. Daha sonra sezgisel her yöntem için aşağıdaki oran hesaplanmalıdır:

$$= \frac{\text{Sezgisel yöntem yardımıyla bulunan turun beklenen uzunluğu}}{\text{Optimal turun uzunluğu}}$$

Bu oran 1'e yaklaştığı sürece sezgisel yöntem o denli iyi olacaktır. Deneysel analizde de, sezgisel yöntemler aslında optimal tur uzunluğu bilinen problemler ile karşılaştırılmaktadır. Örnek olarak 100 şehirli 5 problemde en yakın komşu sezgisel yöntemini her şehre uygulandığında elde ettiği tüm çözümler içerisinden en iyisini alarak bulunduğu tur uzunluğunun, optimal tur uzunluğundan %15 daha uzun olduğu görülmüştür[24]. Aynı problemler için aslında en ucuz ekleme sezgisel yöntemi tüm şehirlere uygulandığında da elde ettiği çözüm ise, optimal turu, %15 artırmıştır.

5. GENETİK ALGORİTMAYA GİRİŞ

Michigan Üniversitesinde psikoloji ve bilgisayar bilimi uzmanı olan John Holland bu konuda ilk çalışmaları yapan kişidir. Mekanik öğrenme konusunda çalışan Holland, Darwin' in evrim kuramından etkilenerek canlılarda yaşanan genetik süreci bilgisayar ortamında gerçekleştirmeyi düşündü. Tek bir mekanik yapının öğrenme yeteneğini geliştirmek yerine bu yapılarda oluşan bir topluluğun çoğalma, çiftleşme, mutasyon, vb. genetik süreçlerden geçerek başarılı (öğrenebilen) yeni bireyler oluşturabildiğini gördü. Çalışmalarının sonucunu açıkladığı kitabının 1975'te yayınlanmasından sonra geliştirdiği yöntemin adı ise Genetik Algoritmalar (kısaca GA) olarak yerleşmiştir. Ancak 1985 senesinde Holland'ın öğrencisi olarak doktorasını veren David E. Goldberg isimli inşaat mühendisi 1989 da konusunda bir klasik sayılan kitabını yayınlayana dek genetik algoritmaların pek de pratik yararı olmayan bir araştırma konusu olduğu düşünülüyordu. Aslında gaz boru hatlarının denetimi üzerine yaptığı doktora tezi ona sadece 1985 National Science Foundation Genç Araştırmacı ödülünü kazandırmakla kalmamıştır, aynı zamanda genetik algoritmaların pratik kullanımının da olabilirliğini kanıtladı. Üstelik kitabında genetik algoritmalara dayalı tam 83 uygulamaya yer vererek GA'nın dünyanın her tarafında çeşitli konularda kullanılmakta olduğunu gösterdi.

Genetik Algoritmalar(GA) daha çok;

- Matematiksel modeli kurulamayan,
- Çözüm alanı oldukça geniş,
- Problemi etkileyen faktörlerin çok fazla olduğu

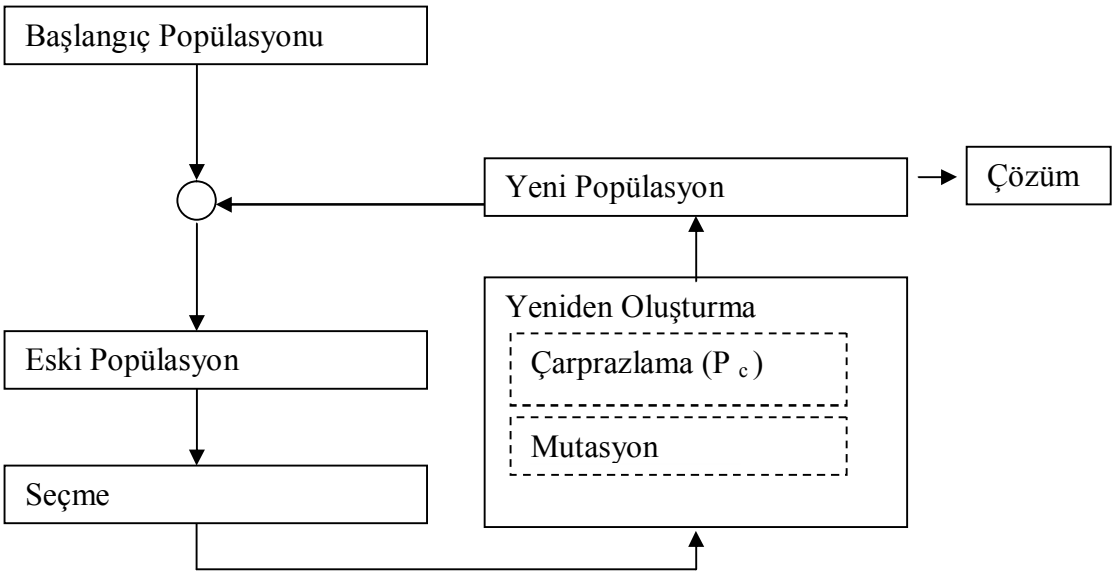
problemlerin çözümünde etkin olarak kullanılmaktadır.

5.1. Genetik Algoritmanın Tanımı

Genetik algoritma, doğadaki evrim mekanizmasını örnek alan bir arama metodudur ve bir veri grubundan özel bir veriyi bulmak için kullanılır.

Genetik algoritmalar doğada var olan en iyinin yaşaması kuralına dayanarak sürekli iyileşen çözümler üretir. Bu nedenle "iyi"nin ne olduğunu belirleyen bir uygunluk fonksiyonu ve yeni çözümler üretmek için de yeniden kopyalama, değiştirme gibi operatörleri kullanmaktadır Genetik algoritmaların bir diğer önemli özelliği ise bir grup çözümle uğraşıy-yıda çözümün içinden iyileri seçilip kötüleri elenebilir.

Genetik algoritmaların diğer algoritmalardan ayrılan en önemli özelliklerden biri ise seçmedir. Genetik algoritmada çözümün uygunluğu seçilme şansını artırır,fakat bunu garanti etmez. Seçim de ilk grubun oluşturulması gibi rasgeledir ancak bu rasgele seçimde seçilme olasılıkları çözümlerin uygunluğu belirler. Genetik algoritmanın yaptığı işleri temelini akış diyagramı olarak Şekil 5.1' de görebiliriz.



Şekil 5.1. Genetik algoritmanın çözüm süreci

5.2. Genetik Algoritmaların Diğer Yöntemlerden Farkları

- Genetik algoritma parametrelerin kodlarıyla uğraşmaktadır. Parametrelerde kodlanabildiği sürece fark etmez.
- Genetik algoritma bir tek yerden değil de, bir grup çözüm içinden arama yapar.

- Genetik algoritma ne yaptığı konusunda bilgi içermez, nasıl yaptığını bilir. Bu nedenle bir kör arama metodudur .
- Genetik algoritmalar olasılık kurallarına göre çalışmaktadır. Programın ne kadar iyi çalışacağı ise önceden kesin olarak belirlenemez. Ama olasılıkla hesaplanabilmektedir.

5.3. Genetik Algoritmanın Kullanılma Nedenleri

Problemlerin maksimizasyon ve minimizasyonunda türünde, eniyileme veya optimizasyonunda öncelikle niçin diğer yöntemlerin kullanılmadığı belirtilmelidir.

Genelde bu problemlerin çözümünde;

- Türev ve integral hesap,
- Numaralama yöntemleri,
- Rasgele arama yöntemleri

gibi üç tip temel çözüm yönteminden yararlanılır. Türev ve integral hesaba dayanan hesaplama yöntemleri çok yoğun olarak kullanılmıştır. Bu yöntemler fonksiyonun 1.mertebe türevini sıfır yapan köklerinin fonksiyona en küçük ve en büyük değer veren noktalar olmasına dayanır. Gerçek problemlerde bu noktaları bulmak çok daha ayrı bir problemdir. Bilinen diğer bir yöntem ise, alınan bir başlangıç noktasından yukarı yönde ilerleyerek en iyi sonucu bulmayı hedefler. Tepe tırmanma adı verilen bu yöntem fonksiyon grafiğinin tepelerine tırmanır. Ancak çok sayıda dönme noktası içeren bir fonksiyonda çok sayıda tepe oluşur. Hangi tepenin en iyi çözüm olduğunu belirlemek zordur. Numaralama yöntemleri ise oldukça alışılagelmiştir. Sürekli olan gerçel sayı aralıkları belli sayıda parçaya ayrılarak her bir parça denenir. Ancak problemlerin boyutu bu yöntem için büyük olabilir. Bu yöntemin daha geliştirilmiş şeklini dinamik programlama oluşturur. Dinamik programlamada parçalar arasından iyi görünenler seçilir. Bu parçalar tekrar parçalara ayrılarak işlemci tekrarlanır. Bu

yöntem de tepe tırmanma yöntemi gibi yanlış tepeleri araştırabilmektedir. Dinamik programlama tepelerin fazla olmadığı aralıklarda başarılı ve hızlıdır.

Kısaca en iyilemenin;

- Bir işin daha iyi yapılması,
- En doğru şekilde yapılması

olmak üzere iki amacı vardır.

Günümüzde rasgele aramaların kullanımı artmaktadır. Bu tip aramalar en iyilemenin daha iyi yapılmasını sağlamakta olup daha başarılıdırlar. İnsanların bilgisayarlardan beklentisi mükemmellik olduğu için bu tip aramalar şüpheye neden olabilir. Genetik Algoritmalar, klasik yöntemlerin çok uzun zamanda yapabilecekleri işlemleri kısa bir zamanda çok net olmasa da yeterli bir doğrulukla yapabilir [32].

5.4. Genetik Algoritmaların Temel Kavramları

Bu bölümde GA yı daha iyi anlamak için bazı temel kavramlar tanıtılacaktır.

5.4.1. Gen

Kendi başına anlamı olan ve genetik bilgi taşıyan en küçük genetik birimdir. Kısmi bilgi taşıyan bu küçük yapıların bir araya gelmesi ile tüm bilgileri içermekte olan kromozomlar meydana gelmektedir. Programlama açısından genlerin tanımlanması programcının olayı aslında iyi tanımasına bağlıdır. Bir gen A, B gibi bir karakter olabileceği gibi, 0 veya 1 ile ifade edilen bir bit veya bit dizisi olabilmektedir. Örneğin, bir cismin yalnızca x koordinatındaki yerini gösteren bir gen 101 gibi bir bit dizisi şeklinde gösterilebilir. Bu cisme ait hız, ağırlık gibi diğer özellikler de benzer şekilde ifade edilebilir.

5.4.2. Kromozom

Bir ya da daha fazla genin bir araya gelmesiyle oluşan ve probleme ait tüm bilgileri içeren genetik yapılardır. Bir grup kromozom ise bir araya gelerek bir topluluk (popülasyon) oluştururlar. Yani kromozomlar toplumdaki birey ya da üyelere karşılık gelirler. Ele alınan problem açısından bakılırsa kromozomlar geçerli alternatif aday çözümler anlamına gelir.

Örneğin bir kromozom ele alınan bir tasarım probleminde koordinat, açı, boyut gibi değişkenlerden meydana gelen bir bütün olabilmektedir. Aynı kromozom bir üretim planlama probleminde miktar, işlem rotası, zaman gibi değişkenleri içerebilir. Basit olarak 100 011 101 bit dizisi; 4x3x5 birim boyutlarında tasarlanan ve aynı zamanda dikdörtgen yüzeylerden oluşan bir kutunun boyutları olabilmektedir. Kromozomlar, genetik algoritma yaklaşımının üzerinde uygulandığı en temel birimler olduğundan, olayın bilgisayar ortamında çok iyi ifade edilmesi gereklidir. Kromozomun hangi kısmına ne anlam yükleneceği ve ne tür bir bilgi gösterimi kullanılacağı kullanıcının olaya bakışına ve probleme göre değişecektir.

5.4.3. Popülasyon(Topluluk)

Popülasyon kromozomlar veya bireyler topluluğu olarak tanımlanabilir. Popülasyon aynı zamanda üzerinde durulan problem için geçerli alternatif çözümler kümesidir. Aynı anda bir popülasyonda bulunan birey sayısı sabit ve probleme bağlı olup kullanıcı tarafından belirlenmektedir. Gerçek hayatta olduğu gibi GA'ın çalışması esnasında popülasyonun bir kısım bireyleri yok olmakta ve yerlerini ise yenileri almaktadır. İleride anlatılacak genetik operatörler(işlemciler)le sağlanan bu sürekli yenilenme sayesinde yeni çözümlere ulasılmakta ve böylece probleme daha uygun çözümler bulunabilmektedir. Literatürde popülasyon büyüklüğü için kesin bir ifade kullanılmamakla birlikte kullanıcının kendisinin bir büyüklük atamasının uygun olduğu belirtilmektedir. Ancak fazla sayıdaki kromozom çözüm süresinin uzamasına neden olabileceği gibi az sayıdaki kromozom da çözüme ulaşmayı güçleştirebilir. Bu

çalışmada bahsedilen popülasyon kavramı ise, aslında gruplar (operasyonlar) bazında kromozomlardır.

5.4.4. Uygunluk fonksiyonu

Kromozomların, problemin çözümünde gösterdiği performansı belirlemekte olan ve problemden probleme değişen bir değerlendirme kriteridir. Kromozomun problemin çözümüne uygunluğunu gösteren başarı ölçüsü olarak da düşünülebilmektedir. Hangi kromozomun bir sonraki nesilde de hayat sürdürebileceğinin belirlenmesinde ve yeni kromozomları oluşturacak olan eşlerin oluşturulmasında kromozomların uygunluk fonksiyon değerleri ağırlıklı olarak göz önünde bulundurulmaktadır. Benzer şekilde popülasyonda yeni bireylere yer açmak amacı ile popülasyondan eski bireyleri çıkarma işlemcisinde uygunluk değeri etkin rol oynamaktadır Uygunluk fonksiyonu aslında, problem için en uygun çözümü belirlemede bir kriter olduğundan üzerinde durulan konuyla ilgili kar veya verimliliği maksimum yapacak, maliyet veya kaybı minimum yapacak değişkenlerin ölçülmesini sağlayacak bir fonksiyon olmalıdır.Fonksiyonun belirlenmesi için problem iyi tahlil edilerek objektif bir değerlendirme kriteri seçilmelidir.

5.4.5. Genetik işlemciler

Genetik işlemciler (operatörler), genetik algoritma yaklaşımının problemler üzerine uygulanmasını sağlayan temel araçlardır. Problemde çözüme ulaşmayı sağlayan bu işlemcilerdir. Genetik algoritmaların en temel işlemcileri kromozomlar arasında aslında bilgi alışverişini sağlamakta olan“çarprazlama” ve küçük değişimi sağlayan “değişim(mutasyon)” dur. Genetik algoritmaya genetik özelliği veren de bu işlemcilerdir. Kromozomlardan bazılarının bir sonraki evrime geçmek için aynı şekilde kalmalarını sağlamakta olan “kopyalama” işlemcisi, çarprazlamaya tabi tutulacak kromozomların seçilmesinde ise “eşleme” işlemcisi, popülasyondan çıkacak olan ya da bu yeni topluluğa girecek kromozomları belirleyen “seçme” işlemcisi ve ilk olarak başlangıç popülasyonunu oluşturan “üretme” işlemcisi gibi yardımcı işlemcilerden faydalanılarak problemin çözümü gerçekleştirilebilir. Genetik

işlemcilerin en önemli özelliği ise problemden probleme farklı şekilde tanımlanabilmeleridir.

5.4.6. Çarprazlama işlemcileri

Bu işlemci, iki kromozom arasındaki karşılıklı bilgi değişimini gerçekleştirerek yeni kromozomların oluşmasını sağlamakta olan işlemcidir. Bilgi değişimini sağlayacak kromozomların seçiminde ise uygunluk değerleri göz önünde bulundurulmaktadır. Çarprazlanarak bilgi alışverişinde bulunacak olan kromozomların seçiminden önce, başlangıçta kromozomların çarprazlamaya tutulma olasılığını belirtebilecek sabit bit çarprazlama oranı tanımlanır. Bu çarprazlama oranı kromozomların çarprazlamaya tabi tutulma olasılığını belirtir. Çarprazlamadaki diğer önemli bir husus ise, çarprazlama stratejisidir. Çarprazlanacak ilk kromozom en yüksek uygunluk değerli kromozomdan başlanarak seçilirken ikincisi de diğerleri arasından rasgele seçilebilir. Kromozomlardaki bilgi değişimi ise, seçilen bazı gen gruplarının karşılıklı olarak değiştirilmesiyle gerçekleşir. Çarprazlama işlemi de çeşitli şekilllerde olabilir[33]. Pozisyona dayalı çarprazlamada;çarprazlanacak iki eş kromozom üzerinde bir grup çarprazlama noktası (bir veya daha fazla nokta) rasgele seçilir İkinci kromozomun bu pozisyonlardaki parçası birinci kromozomun ilgili pozisyonlarına konur ve daha sonra boş kalan pozisyonlar yeni kromozomda olmayan birinci kromozom elemanları sırasıyla alınarak doldurulur, böylelikle bir yeni kromozom meydana gelmektedir. Aynı işlemci diğer kromozom için de yapılarak yeni bir kromozom daha elde edilir. Sıraya dayalı çarprazlamada ise; bir grup nokta rasgele seçilir. Birinci kromozomun seçilmiş olan noktalara karşılık gelen karakterleri aynen yerlerini korumaktadır. İkinci kromozomun seçilen noktalara ait karakterleri ise birinci kromozomun aynı noktalarındaki karakterlerinin önüne getirilir. Geriye kalan boş pozisyonlara, ikinci kromozomdan aktarılmış olan yeni karakterler de göz önünde bulundurularak ilk kromozomunda kullanılmamış olan karakterleri tarafından sırasıyla (soldan sağa) yerleştirilerek yeni bir kromozom elde edilir. Bu tür çarprazlama, kromozomu oluşturan karakterlerin sayı ve sıralarının önem taşıdığı durumlarda kullanılır. Bu çarprazlama işlemcisine ait birer çarprazlama örneği ise Şekil 5.2.'de verilmektedir.

	Çarprazlamadan	
	Önce	**Sonra**
Pozisyona göre çarprazlama	**A B C D E F G**	**G A C D E**
B F	| |	
	G F E D C B A	**A G E D**
C F B		
Sıraya göre çarprazlama	**A B C D E F G**	**A G C D**
E F B	| |	
	G F E D C B A	**G A E D**
C B F		

Şekil 5.2. Pozisyona ve sıraya göre çarprazlama işlemleri

Çarprazlamada önemli olan bir diğer faktör ise çarprazlama noktasının sayısı olarak karşımıza çıkmaktadır. Çarprazlama bir veya daha fazla noktadan olabilmektedir. Basit ve pratik olması açısından ise tek noktalı çarprazlamada ise yaygın olarak kullanılmaktadır. Çarprazlamada bilgi değişimi de asıl amaç olduğundan problemin yapısına göre iki ve daha çok noktalı çarprazlamanın uygulamaları da tercih edilebilmektedir. Diğer geliştirilmiş çarprazlama işlemcileri hakkında da geniş bilgi de bulunabilir[34]. En klasik olan ise tek noktalı çarprazlamadır. Birinci ve ikinci kromozom üzerinde ortak belirlenmiş olan rasgele nokta temel alınarak birinci kromozomun bu noktadan önceki kısmı ile ikinci kromozomun bu noktadan sonraki kısmı birleştirilerek yeni bir kromozom elde edilmektedir. İkinci kromozom için de kromozomların diğer kısımları birleştirilmektedir. Çift noktalı çarprazlamada ise kromozomlar üzerinde rasgele belirlenen iki nokta esas alınarak kromozomların bu noktalar arasında kalan kısımlarının karşılıklı değiştirilmiş olduğu çarprazlama olarak belirtilir. Tek noktalı ve çift noktalı çarprazlamaya ait birer örnek aşağıdaki Şekil 5.3' de verilmektedir.

Tek noktalı çarprazlama

Kromozom 1 1 0 1 1 **0 1 0 0**	Yeni kromozom1 1 0 1 1 ***0 1 1 0***
Kromozom 2 1 1 0 0 ***0 1 1 0***	Yeni kromozom 2 1 1 0 0 **0 1 0 0**

Şekil 5.3. Tek ve çift noktalı çarprazlama işlemcileri

Çift noktalı çarprazlama

Kromozom 1 1 0 1 **1** ***1*** 1 0 0	Yeni kromozom 1 1 0 1 **0** ***0*** 1 0 0
Kromozom 2 1 1 0 **0** ***0*** 1 1 0	Yeni kromozom 2 1 1 0 **1** ***1*** 1 1 0

Şekil 5.3.(Devam) Tek ve çift noktalı çarprazlama işlemcileri

5.4.7. Değişim (Mutasyon) işlemcisi

Kromozomların genleri veya genleri oluşturan küçük birimleri üzerinde değişiklik yapılmasını sağlayan işlemci olarak ifade edilemektedir. Değişime uğratılacak olan kromozomun seçilmesinde, kromozomun değişime uğrama ihtimalini gösteren ve başlangıçta sabit olarak tanımlanan bir değişim oranı söz konusu olmaktadır. Genetik algoritmalarda değişime tabi tutulacak kromozomların belirlenmesinde bazılarının istisna tutuluyor olabilmesi veya özellikle değişime uğratılması gibi özel stratejiler tanımlanabilir. Değişim işlemci, kromozom üzerinde seçilen geni oluşturan alt birim üzerinde yapılacak olan küçük bir değişiklikle (0'ın 1 yapılması veya tersi gibi) gerçekleşir. Gösterim olarak buna uymayan bir yapıda da yine rasgele seçilecek iki genin yerleri veya sırası değiştirilerek değişim gerçekleştirilebilmektedir. GA'da değişimin sağladığı avantaj, aslında problemin çözüm alanını araştırmada yön değişikliklerini sağlayarak araştırmanın aslında kısır döngüye girmesini önlemektir.

Çeşitli değişim işlemcileri vardır. Pozisyona bağlı değişimde ise; rastgele seçilen karakterlerin(genlerin) yerleri değiştirilerek gerçekleştirilir. Sıraya göre değişim ise kromozomun rastgele seçilmiş olan iki karakterinden ikincisinin, birincinin önüne getirilmesiyle oluşmaktadır. Kromozomun gösterimine göre sıranın ve de karakter sayısının sınırlı olmadığı bir ikili sistem kromozom gösteriminde de rasgele seçilen bir karakterin karşıt(0'ın yerine 1 gibi) değeriyle değiştirilmesiyle olur.

Yukarıda bahsedilen çarprazlama türlerine ait birer örnek Şekil 5.4' de verilmektedir.

	“Değişimden	
	Önce	**Sonra**
Pozisyona göre değişim	A B C D E **F**	**F** B C D E A
Sıraya göre değişim	A B C D E **F**	**F** A B C D E
Kromozom	1 1 0 1 0 1 **1** 0	1 1 0 1 0 1 **0** 0

Şekil 5.4. Çeşitli değişim (mutasyon) işlemci örnekleri

Genetik algoritma kendi içinde sanal olarak şemalar oluştumaktadır Toplumun bireyleri incelenerek bu şemalar ortaya çıkarılabilir. GA şemaları oluşturmak için toplum üyelerinin kodları dışında bir bilgi tutulmamaktadır. GA’ nın bu özelliğine ise içsel paralellik denir. Her nesilde, iyiyi belirleyen şemalardaki belirsiz yada önemsiz elemanlar azalır. Böylece GA sonuca doğru belirli kurallar içerisinde ilerler.

5.5. GA Çözüm Aşamaları

GA nın basit görülen yaklaşımı oldukça karmaşık yapıdaki bir çok problemin çözümünü etkin bir şekilde gerçekleştirebilme yeteneğine sahiptir.

GA genel olarak başlıca şu temel adımlarla uygulanır

Adım 1: Problemin değişken tanım aralığını sabit uzunluğa sahip bir kromozom olarak belirle, örnek topluluğun N- Kromozom sayısını, çarprazlama olasılığı Pc yi ve değişim olasılığı Pm yi seç.

Adım 2: Problemin tanım bölgesindeki her bir kromozom için performans ya da uygunluğunu ölçmek için bir uygunluk fonksiyonu belirle(Bu uygunluk fonksiyonu yeni üretilen kromozomların seçiminde temel oluşturur .

Adım 3: Kromozomlar için rassal olarak N boyutlu bir başlangıç örnek topluluğu türet.

x1 , x2, x3, ... , xn

Her bir kromozomun uygunluğunu hesapla.
f(x1), f(x2), ... , f(xn)

Adım 5: Çiftleştirmek için mevcut örnek topluluktan bir çift kromozom seç. (Bu çiftlerin uygunluk olasılığına göre ebeveyn kromozomlar seçilir. Çiftleştirmek için seçilen yüksek uyumlu kromozomlar düşük uyumlu kromozomlardan daha yüksek olasılığa sahiptir.

Adım 6: Genetik işlemcilerden çarprazlama ve değiştirmeyi uygulayarak bir çift çocuk kromozomu oluştur.

Adım 7: Yeni oluşturulan çocuk kromozomları yeni örnek topluluğa yerleştir.

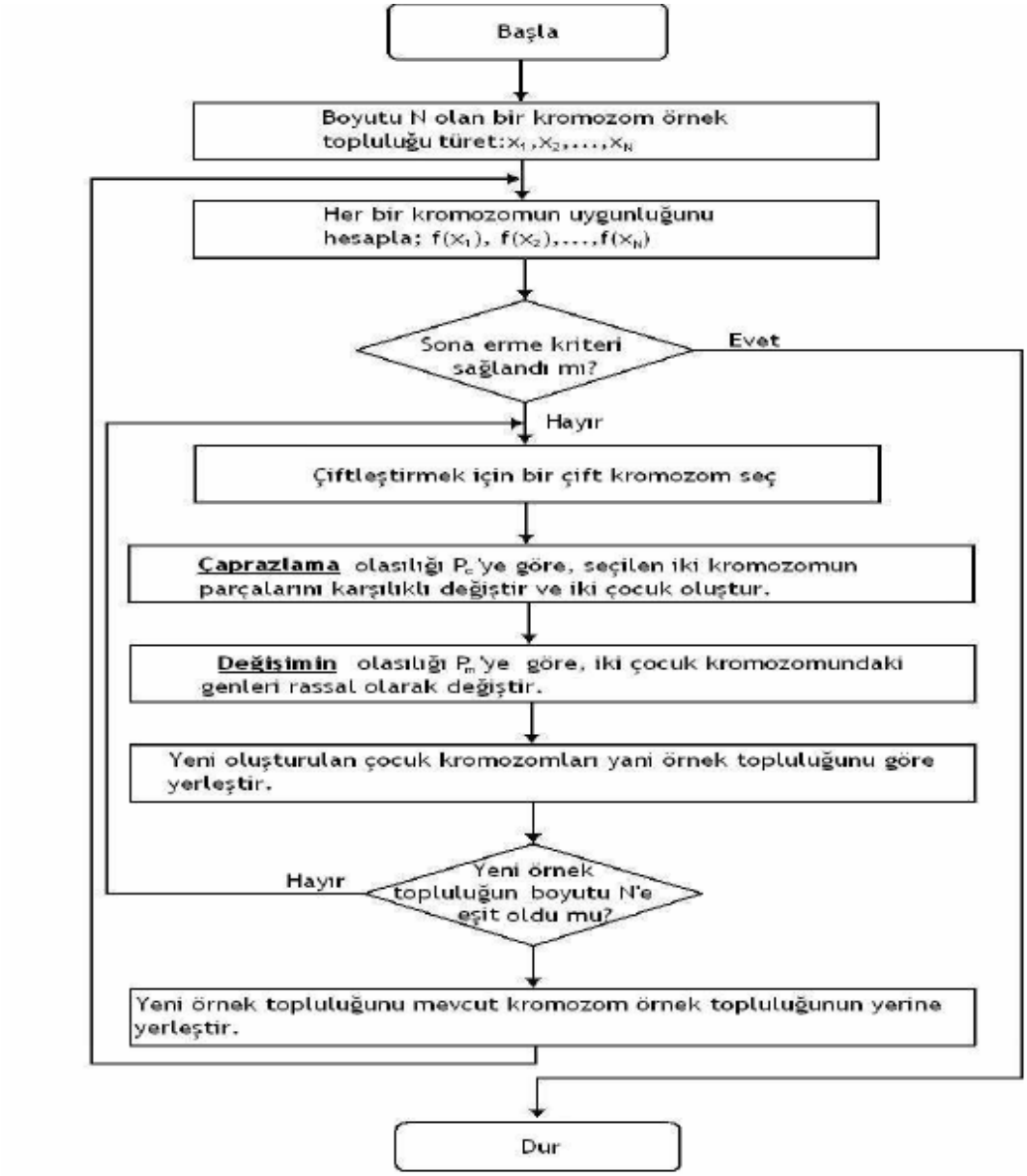
Adım 8: Yeni Kromozom örnek topluluğunun boyutu N oluncaya kadar Adım 5' i tekrarla.

Adım 9: Yeni çocuk örnek topluluğunu başlangıç ebeveyn örnek topluluğunun yerine yerleştir.

Adım 10: Adım 4'e git ve sona erme kriteri sağlanıncaya kadar süreci tekrarla.

Şekil 5.5. deki akış şemasından da görüleceği gibi GA sıralı adımlarla uygulanan bir süreçtir. Her bir sıralı adım sonucunda da bir nesil elde edilmektedir. GA için türetilen nesillerin sayısı 50 'den 500'e kadar değişebilmektedir.

Nesillerin bütün setini elde etmeye toplam süreç adı verilir. Bir toplam sürecin sonunda yüksek uygunlukta bir veya daha fazla kromozom bulmaya çalışırız.



Şekil 5.5. Temel GA algoritması akış diyagramı

Standart bir GA yöntemi aşağıdaki gibi verilebilir:

- Başlangıç popülasyonunu rastlantsal olarak üret.
- Popülasyon içindeki tüm kromozomlara ait olan amaç fonksiyonu değerlerini hesapla.
- Tekrar üreme, çarprazlama ve mutasyon operatörlerini uygula.
- Oluşturulan her yeni kromozomun amaç fonksiyonu değerlerini bul.

- Amaç fonksiyonu değerleri kötü olan kromozomları popülasyondan çıkar.
- 3-5 arasındaki adımları tekrar et.

GA'nın basit algoritması Şekil 5.6'da görülebilir. GA gerçekleştirmeleri bu basit taslağın benzerlerini, değiştirilmiş sürümlerini kullanırlar.

```
BEGIN GA
        t:0
        popülasyon P(t)'yi ilkle
        P(t)'deki her bireyin uygunluk değerini hesapla
        WHILE (NOT durma-koşulu) DO
                BEGIN WHILE
                t:t+1
                P(t-1)'den P(t)'yi oluştur
                P(t)'ye GA uygula
                P(t)'deki her bireyin uygunluk değerini hesapla
        END WHILE
END GA
```

Şekil 5.6. Genetik algoritma sözdekodu

5.5.1. Başlangıç popülasyonunun oluşturulması

Bu adıma, öncelikle toplumda bulunacak birey sayısını belirleyerek başlanmaktadır. Kullanılacak sayı için bir standart yoktur. Genel olarak önerilen 100-300 aralığında bir büyüklüktür. Büyüklük seçiminde yapılan işlemlerin karmaşıklığı ve aramanın derinliği önemlidir.

Başlangıç popülasyonunu oluşturmak için ilk adım popülasyonların genlerinin nasıl kodlanacağıdır. Kromozomların kodlanmasında kullanılan birkaç yöntem aşağıda verilmiştir.

Kodlama tipine karar verdikten sonra başlangıç popülasyonların oluşturulmasına gelinir. Literatürde en çok kullanılan yöntem popülasyonun tüm genlerinin rassal üretilmesidir. Bu yöntem ile popülasyondaki her kromozomun her genine rassal olarak mümkün değerlerden biri atanır.

Kromozom Tipleri ;

- İkili Düzen (0101 . . . 1100)
- Reel sayılar (43. 2 -33. 1 . . . 0. 0 89. 2)
- Eleman permütasyonları (E11 E3 E7 . . . E1 E15)
- Liste kuralı (R1 R2 R3 . . . R22 R23)
- Program elemanları (Genetik Programlama)

Başlangıç popülasyonun oluşturulmasında kullanılabilecek diğer bir yöntemde ise, popülasyonu diğer bazı teknikler kullanarak elde edilen çözümlerle doldurmaktır.

5.5.2. Popülasyon uygunluğunun hesaplanması

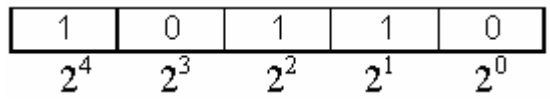
Kromozomların ne kadar iyi olduğunu bulan fonksiyona uygunluk fonksiyonu denir. Bu fonksiyon işletilerek kromozomların uygunluklarının bulunmasına ise hesaplama adı verilir. Bu fonksiyon genetik algoritmanın beynini oluşturmaktadır. Genetik algoritmada probleme özel çalışan tek bölüm bu fonksiyondur. Uygunluk fonksiyonu kromozomları problemin parametreleri haline getirerek onların bir bakıma şifresini çözmektedir, sonra bu parametrelere göre hesaplamayı yaparak kromozomların uygunluğunu bulur. Çoğu zaman genetik algoritmanın başarısı bu fonksiyonun verimli ve hassas olmasına bağlı olmaktadır. Kromozom bir bireyi temsil eder ve genlerden oluşur. Bir kromozom örneği aşağıdaki gibi gösterilebilir:

g_1	g_2	g_3	.	.	.	g_{n-1}	g_n

Burada g_i, i=1,…n kromozomu oluşturan genlerdir ve n kromozomun uzunluğunu yani kromozomdaki gen sayısını belirtmektedir. Kromozomun uzunluğu fonksiyonun alabileceği değerlere bağlı olmaktadır. Fonksiyon kodlanmasında kullanılan genel yöntem ikili kodlamadır. Örneğin kromozomun uygunluk değeri için kısıtlar söz konusu olsun;

$$0 \le F(x) \le 31 \text{ ve } F(x) \text{ tamsayı}$$

Örnek kodlama için gerekli gen ihtiyacı verilen sınır değerlerin ikilik tabana çevrilmesi ile bulunur. Yapılan işlem çevrim sonucu problemin çözümü içen 5 gene ihtiyaç vardır. Aşağıda örnek bir kromozom ve deşifre işlemi gösterilmiştir.



$F(x)=(2^4 \times 1) + (2^3 \times 0) + (2^2 \times 1) + (2^1 \times 1) + (2^0 \times 0) = 22$

Böylece kromozomun sahip olduğu uygunluk değeri hesaplanmış olur.

İşlemde fonksiyon tamsayı değerler alabildiğinden dolayı işler kolaylaşmıştır. Eğer fonksiyon değeri gerçel değerler alabilselerdi işlemler biraz daha karmaşık olacaktı. Gerçel değerleri alabilen fonksiyonları değerlerini kodlayabilmek için bir hassasiyet değerine ihtiyaç duyulur. Bu değer kullanıcı tarafından tanımlanır ve parametrenin değer aralığının kaç eşit parçaya bölüneceğini belirlemede kullanılır. Bir F(x) değeri için şu kısıt söz konusu olsun:

$$f_{min}(x) \leq F(x) \leq f_{max}(x)$$

Burada f_{min} ve f_{max} sırayla F(x) 'in alabileceği en küçük ve en büyük değerlerdir. Böylece F(x) değeri (f_{max}-f_{min}) uzunluğundaki bir aralıktaki bir bölgede değerler alabilir. Eğer hassasiyet değeri s ile gösterilirse F(x)' in değer aralığının en azından (f_{max}-f_{min}) $x10^s$ eşit parçaya bölünmesi gerekir. F(x) değerini kodlamak için gerekli gen sayısı ise şöyle belirlenir:

$$2^{m-1} < (f_{max}-f_{min}) \times 10^s \leq 2^m \qquad (5.1)$$

koşulunu sağlayan en küçük m değeri F(x) değerini kodlamak için gerekli olan gen sayısıdır.

Kromozomun deşifre edilmesi iki adımda gerçekleşir:

- Öncelikle (g_0, g_1, …. . g_{m-1}) dizisi iki tabanından onluk tabana çevrilir.

$$x = \sum_{i=0}^{m-1} g_i * 2^i \tag{5.2}$$

- Eş. 5.3'e karşılık gelen fonksiyon değeri olan gerçel sayı bulunur.

$$F(x) = f_{min} + x. \frac{(f_{max} - f_{min})}{2^m - 1} \tag{5.3}$$

Bir modele ilişkin çözümün nasıl kodlanacağı ise genetik algoritmalar için anahtar noktadır. Holland yaptığı teorik çalışmalar sonucunda ikili kodlamanın en uygun kodlama yöntemi olduğunu göstermiştir [35].

Eldeki modele ilişkin amaç fonksiyonundaki değerlendirme fonksiyonu olarak kullanılabilmesi için, amaç fonksiyonunun yapısının en büyükleme yapılı olması gerekmektedir. Eğer amaç fonksiyonu en küçükleme yapılıysa onun en büyükleme yapılı fonksiyona çevrilmesi gerekir. Bunu gerçekleştirmenin en kolay yolu,

$$\min F(x) = \max (-F(x)) \tag{5.4}$$

dönüşümü yapmaktır. Bu dönüşümü yaparken dikkat edilmesi gereken nokta ise, dönüşümün negatif uygunluk değerine sebebiyet vermemesidir.

Amaç fonksiyonunu en büyükleme yapılı bir fonksiyona dönüştürmenin diğer bir yolu da fonksiyonu belirli bir M sabitinden çıkarmaktır.

$$\min f(x) = \max (M - f(x)) \tag{5.5}$$

Bu yol ile uygun bir M büyüklüğü kullanıldığında negatif fonksiyon değerinden de kaçınılabilir. Bölüm 6'da M değeri popülasyonda ki en büyük F(x) değeri olarak atanmıştır.

5.5.3. Genetik operatörler

Genetik algoritmalarda kullanılırken üç klasik operatör vardır:

- Kopyalama
- Çarprazlama
- Mutasyon

Bu operatörler algoritmanın evrimsel işlem safhasını oluştururlar

Kopyalama

Kopyalama operatörünün mantığı tamamıyla Darwinin evrim teorisine dayalıdır. Darwin' in evrim teorisinde ise, değişmekte olan çevreye hızlı ve uygun adaptasyon sağlayabilen canlılar ancak yaşamlarını sürdürdüğünden ve uygun adaptasyonu sağlayamayan canlıların soylarının tükendiği bahsedilir. Kopyalama operatöründe, Darwinin evrim teorisinde sözü geçen çevre rolünü değerlendirme fonksiyonu oynar. Kopyalama operatörü tıpkı Darwinin evrim teorisinde olduğu gibi daha iyi olan bireylerin yaşamını sürdürmesini ve çoğalmasını, kötü olan bireylerin ise ölmesini sağlamaktadır. Buradaki iyilik ve kötülük kıstası değerlendirme fonksiyonu sonucu elde edilen uygunluk değerleri yoluyla belirlenir. Uygunluk değerlerine dayalı olarak her bir kromozom için sonraki jenerasyona seçilme olasılığı tanımlanır. Tabi ki uygunluk değerleri aslında yüksek olan kromozomların sonraki jenerasyona seçilme olasılıkları daha fazla ve uygunluk değerleri düşük olanların seçilme olasılıkları da daha düşük olacaktır.

Genetik algoritmaların evrim süreci ile ilişkili iki önemli faktör vardır. Bunlar popülasyon çeşitliliği ve seçim baskısıdır. Popülasyon çeşitliliği kromozomların ne ölçüde çeşitlilik gösterdiği ile ilgili olmaktadır. Seçim baskısı ise aslında kopyalama operatörü sırasında seçme işlemi yapılırken iyi çözümlerin seçilmesine ne kadar ağırlık verileceği ile ilişkili olmaktadır. Seçim baskısındaki artış ağırlıklı olarak iyi

kromozomların seçilmesine neden olur ve dolayısıyla popülasyon çeşitliliği azalır. Seçim baskısındaki azalış, kötü olan kromozomların da seçilmesi olasılığını arttırır ve böylece popülasyon çeşitliliği artar.

Güçlü bir seçim baskısı ise, genetik algoritmanın erken safhalarda bulduğu iyi bir çözüme yakınsamasına neden olabilir. Ancak erken safhalarda çözüm uzayı yeteri kadar incelenememiştir ve bu nedenle güçlü seçim baskısı genetik algoritmanın bir yerel en iyiye takılmasına neden olabilir. Bu duruma erken yakınsama denir.

Düşük bir seçim baskısı ise aslında popülasyondaki bütün kromozomları seçilme olasılıklarını birbirine yaklaştırarak popülasyon çeşitliliğini arttırmaktadır. Ancak bu durumda da iyi ve kötü kromozomların seçilme olasılıkları arasında çok az fark olacağından algoritma bir rassal arama algoritmasına dönüşmektedir. Bu nedenlerle popülasyon çeşitliliği ve seçim baskısı arasında uygun bir denge kurmak oldukça önemlidir.

Kopyalama operatöründe kullanılan seçim çeşitli kriterlere göre yapılabilir. Rulet seçimi, Boltzman seçimi, turnuva seçimi, sıralı seçim bunlardan bazılarıdır.

Rulet seçimindeki kromozomlar uyumluluk fonksiyonuna göre bir rulet etrafına gruplanır. Uyumluluk fonksiyonu herhangi bir kritere uyan bireylerin seçilmesi için kullanılır. Bu rulet üzerinden rasgele bir birey seçilir. Daha büyük alana sahip bireyin seçilme şansı daha fazla olacaktır .

Rulet seçimi eğer uyumluluk çok fazla değişiyorsa sorun çıkartabilir. Örneğin en iyi kromozomun uyumluluğu %90 ise diğer kromozomların seçilme şansı azalacaktır. Bunu önlemek için sıralı seçim kullanılabilir. Sıralı seçimde en kötü uyumlulukta olan kromozoma 1 değeri sonrakine 2 değeri verilir ve böylelikle seçilmede bunlara öncelik tanınmış olur. Bu şekilde onların da seçilme şansı artar fakat bu çözümün daha geç yakınsamasına neden olabilir.

Rulet tekerleği seçimi ise çözümlerin uygunluk değerlerinin negatif olmamasını gerektirir. Çünkü eğer olasılıklar negatif olursa bu çözümlerin seçilme şansı yoktur. Çoğunluğunun uygunluk değeri negatif olan bir toplumda ise yeni nesiller belli noktalara takılıp kalabilir. Rulet tekerleği seçim tekniği kullanılarak bir sonraki jenerasyon (t+1) için her bir kromozom şöyle seçilir:

- Her bir kromozom için uygunluk değeri hesaplanır
- Popülasyonun toplam uygunluğu hesaplanır

$$F = \sum_{i=1}^{n} f_i^t \tag{5.6}$$

- Her bir kromozom için seçilme olasılığı *pi* hesaplanır.

$$pi = \frac{f_i^t}{F} \qquad i = 1,2,.....,n \tag{5.7}$$

- Her bir kromozomun qi birikimli olasılığı hesaplanır

$$q_i = \sum_{j=1}^{i} p_i \qquad j = 1,2,.....,n \tag{5.8}$$

- Seçim süreci rulet tekerinin popülasyon büyüklüğü (n) kere çevrilmesinden ibarettir. Her çevirişte bir kromozom seçilir. seçme işlemi aşağıdaki şekilde özetlenebilir:
- [0, 1] aralığında bir r rassal sayısı üretilir
- $q_{j-1} < r \leq q_j$ koşulunu sağlayan j. kromozom seçilir($q_0=0$).

Çarprazlama

Çarprazlama özellikleri;

- İki ebeveyn arasında genetik materyalin değiş dokuş yapılır.
- Çarprazlama noktası veya noktaları rasgele belirlenir.
- GA 'da çarprazlama operasyonu en önemli özelliktir.

Tek noktalı çarprazlama

Eski birey 1	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**
Eski birey 2	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**1**

Yeni Birey 1	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**1**
Yeni Birey 2	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**

Şekil 5.7. Tek noktalı çarprazlama işlemi

Tek nokta çarprazlama operatörünün en önemli eksikliği ise çarprazlama işleminin kromozom uzunluğundan bağımsız olması olarak açıklanır[35]. Kromozom uzunluğu veya kromozom tarafından kodlanan değişken sayısı ne olursa olsun yapılan işlemler aslında tek nokta üzerinden çarprazlama yapmaktadır. Bu da kromozomlar tarafından saklanan bilginin etkin bir şekilde paylaşılamamasına neden olabilmektedir.

Tek noktalı çarprazlama ise operatörünün eksikliklerinden dolayı bazı araştırmacılar diğer çeşitlerini araştırmışlardır. İki nokta çarprazlama operatörü bunlardan biridir.

İki noktalı çarprazlama

Eski Birey 1	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**
Eski Birey 2	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**1**
Yeni Birey 1	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0**
Yeni Birey 2	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**1**

Şekil 5.8. İki noktalı çarprazlama işlemi

Gen takası toplumda çeşitliliği sağlamaktadır. Takas, iyi özelliklerin bir araya gelmesini kolaylaştırarak böylece iyiye yaklaşmayı sağlar. Değiştirme kromozomun bir parçasının dışarıdan değiştirilmesi olarak tanımlanmaktadır. Değiştirme görünüşte genetik algoritmanın dayanak noktası olarak görünür, ancak etkisi bir çözüm üzerindedir. Bu da yalnız başına başarılı olmasını zorlaştırır. İkilik dizilerde değiştirme rasgele bir bit'in değiştirilmesiyle sağlanabilir.

Çok düşük bir değiştirme olasılığı toplumda bazı özelliklerin kaybolmasına neden olabilmektedir. Bu da en iyi sonuçların bulunmasına engeldir. Ancak yüksek bir değiştirme olasılığı da eldeki çözümleri bozarak sonuca ulaşmayı zorlaştırmaktadır. Gen takası ve değiştirmenin olasılıkları için kesin bir sayı yoktur. Değiştirme (mutasyon) olasılığı 0.01-0.001, gen takası olasılığı 0.5-1.0 genel olarak kabul gören oranlardır [36].

Mutasyon

Çarprazlama işleminin ardından mutasyon işlemi gerçekleştirilir. Mutasyon, oluşan yeni çözümlerin önceki çözümü kopyalamasını ve sonucun yerel en iyiye takılmasını önlemek amacıyla kullanılmaktadır. Mutasyon oluşan yeni bireyin bir bitini (eğer ikili düzende ifade edilmiş ise) rasgele değiştirir. Mutasyon işlemi popülasyondaki genler üzerinde aşağıdaki şekilde işlem yapar.

- [0, 1] aralığında bir r rassal sayı üretilir.
- Eğer $r<p_m$ ise söz konusu gen mutasyona uğratılır. Bu mutasyon işlemi ise, genin değerini diğer alternatif değerlerden birisinin değeri ile rassal olarak değiştirmek yoluyla gerçekleştirir. İkili kodlama söz konusu ise tabi ki tek alternatif söz konusudur ve eğer genin değeri 0 ise1, 1 ise 0 yapılır.

Özel mutasyon operatörleri ise GSP uygulamalarında kullanılmaktadır, mutasyona uğrayan gen değeri yerine geçebilecek olan gen ile yer değiştirmelidir.

Yeni Birey	1	1	0	1	0	0	0	0	1	1
Mutasyondan Sonra	1	1	0	1	1	0	0	0	1	1

Şekil 5.9. Mutasyon işlemi

Mutasyon işleminin sona ermesi ile genetik algoritmaların evrimsel işlem safhası da sona ermiş olur.

5.6.Elitizm

Üreme, çarprazlama ve mutasyon işlemleri sonrasında kuşakta bulunmakta olan en iyi uyumluluğa sahip birey sonraki kuşağa aktarılamayabilir. Bunu önlemek için bu işlemlerden sonra oluşan yeni kuşağa bir önceki kuşağın en iyi (elit) bireyi, yeni kuşaktaki herhangi bir birey ile değiştirilir. Buna elitizm adı verilir.

5.7. Genetik Algoritma İşleyişi

Genetik algoritmalarda öncelikle rasgele veya bilinen çözümleri içeren ve uygunluk değerleri hesaplanan bir başlangıç kümesi (örnek topluluk) oluşturulmaktadır. Örnek topluluktaki kromozom sayısı sabit olarak alınır. Bu sayının çok küçük olması işlem kolaylığı sağlamakla beraber, aslında alternatif çözüm çeşitliliğini azaltacağından tercih edilmemektedir. Kromozomların fazla olması ise işlem hacmini artırarak işlem adımlarının(iterasyonların) uzamasına neden olacağından tercih edilmez.

Bu başlangıç çözümleri daha sonra probleme bağlı olarak belirlenen temel genetik işlemciler yardımıyla evrimden geçirilerek yeni çözümler oluşturulur. Yeni bireyleri oluşturma işleminde kullanılacak kromozomların seçiminde genellikle uygunluk değeri yüksek olanlara öncelik verilir. Bir veya daha fazla rasgele çarprazlama noktası seçilip, bu noktalara göre çarprazlama işlemi uygulanarak bilgi değişimi sağlanır ve yeni kromozomlar oluşturulmaktadır. Çarprazlama işleminden sonra veya çarprazlama sırasında değişim işlemi uygulanır. Değişim işlemcisi tek bir kromozom üzerinde işlem yapar. Kromozomun herhangi bir elemanı rasgele seçilir ve bu eleman onun alabileceği başka bir değer ile değiştirilir. Bu işlemciler uygulandıktan sonra mevcut örnek topluluk bireylerinden bazıları yeni bireylere yer açmak amacı ile örnek topluluktan çıkartılır. Optimal (en iyi) çözüme yaklaşmak amacı ile bu evrimden geçirme işlemi daha uygun bir çözüm bulundukça devam eder, bulunamayınca da durmaktadır. Oluşturulan topluluk uygunluk fonksiyonuna göre değerlendirilip uygunluk değerleri hesaplanır. Aradığımız sonucu veren kromozom olup olmadığı kontrol edilir. Kabul edilebilir sonuca ulaşılmamışsa uygunluk değeri göz önünde bulundurularak yeni kromozomlar oluşturmak üzere farklı kromozomlar

eşlenir. Eşleştirilen bireyler bilgi alışverişinde bulunmak üzere, daha önceden belirlenmiş olan çarprazlama stratejisi çerçevesinde değişime uğratılır. Bir önceki topluluktan doğru olan kromozomlar ve çarprazlama sonucu üretilmiş olan yeni kromozomlara yer açmak amacı ile eski kromozomlardan en düşük uygunluk değerli olanlar veya rasgele seçilenler silinir. Daha sonra oluşan topluluk uygunluk fonksiyonuna göre değerlendirilir. Belirlenen hedefe ulaşıncaya kadar bu evrime devam edilir.

5.7.1. Şema teorisi

Genetik algoritmalarda oluşan başarılı bireyler incelenirse, bu bireyler arasındaki benzerlikler bulunabilmektedir. Bu benzerlikler göz önünde bulundurularak şemalar oluşturulabilir. İkilik dizi kodlaması için aşağıdaki yöntem önerilebilir.
0, 1 ve # (‘#’ o konumda 0 veya 1 olmasının önemsiz olduğunu gösterir.

Örnek olarak ikinci ve dördüncü bitleri 1, altıncı biti 0 olan çözümlerin başarılı olduğu bir toplumda şu şema oluşturulabilir:

#1#1#0

Bu şemaya uygun aşağıdaki ikilik diziler yazılabilir:
010100, 010110, 011100, 011110, 110100, 110110, 111100, 111110.
Görüldüğü gibi aslında şemaların katılması ikilik dizilerle gösterilen arama aralığını büyütmektedir. Arama aralığının büyümesinin ise sonucun bulunmasını zorlaştırması beklenir ancak durum böyle değildir. Seçilim ve yeniden kopyalama ile iyi özellikler daha çok bir araya gelerek daha iyi değerlere sahip şemalara uygun çözümler elde edilir.

Genetik algoritma kendi içinde sanal olarak şemaları oluşturur. Toplumun bireyleri incelenerek bu şemalar ortaya çıkarılabilir. Genetik algoritmalar şemaları oluşturmak için toplum üyelerinin kodları dışında bir bilgi tutmamaktadır. Genetik algoritmaların bu özelliğine içsel paralellik denir. Her nesilde, iyiyi belirleyen şemalardaki belirsiz

ya da önemsiz elemanlar azalmaktadır. Böylece genetik algoritmalar aslında sonuca doğru belli kalıplar içinde ilerler.

5.7.2. Genetik Algoritmanın Performansını Etkileyen Nedenler

- *Kromozom sayısı*: Kromozom sayısını bir şekilde arttırmak aslında çalışma zamanını arttırırken azaltmak da kromozom çeşitliliğini yok eder.
- *Mutasyon Oranı*: Kromozomlar birbirine benzemeye başladığında hala çözüm noktalarının uzağında bulunuyorsa mutasyon işlemi aslında GA' nın sıkıştığı yerden kurtulmak için tek yoludur. Ancak yüksek bir değer vermek,aslında GA' yı kararlı bir noktaya ulaşmaktan alıkoyacaktır.
- *Kaç Noktalı Çarprazlama Yapılacağı*: Normal olarak çarprazlama tek noktada gerçekleştirilmekle beraber yapılan araştırmalar bazı problemlerde çok noktalı çarprazlamanın çok yararlı olduğunu göstermiştir.
- *Çarprazlamanın sonucu elde edilen bireylerin nasıl değerlendirileceği*: Elde edilen iki bireyin birden kullanılıp kullanılamayacağı bazen önemli olmaktadır.
- *Nesillerin birbirinden ayrık olup olmadığı*: Normal olarak her nesil tümüyle bir önceki nesle bağlı olarak yaratılır. Bazı durumlarda yeni nesli eski nesille birlikte yeni neslin o ana kadar elde edilen bireyleri ile yaratmak yararlı olabilir.
- *Parametre kodlanmasının nasıl yapıldığı*: Kodlananın nasıl yapıldığı en önemli noktalardan biridir. Örnek verilecek olursa kimi zaman bir parametrenin doğrusal ya da logaritmik kodlanması,aslında GA' nın performansında önemli bir farka yol açabilir.
- *Kodlama gösteriminin nasıl yapıldığı*: Bu da nasıl olduğu yeterince açık olmamakla beraber GA' nın performansını etkileyen bir noktadır. İkilik düzen, kayan nokta aritmetiği ya da gray kodu ile gösterim en yaygın yöntemlerdir.
- *Başarı değerlendirmesinin nasıl yapıldığı*: Akıllıca yazılmamış bir değerlendirme işlevi çalışma zamanını uzatabileceği gibi çözüme hiçbir zaman ulaşmamasına neden olabilir.

5.8. Uygulama Alanları

Genetik algoritmaların en uygun olduğu problemler geleneksel yöntemler ile çözümü mümkün olamayan ya da çözüm süresi problemin büyüklüğü ile üstel orantılı olarak artanlardır. Bugüne kadar GA ile çözümüne çalışılan konulardan bazıları şunlardır:

- *Optimizasyon*: GA, aslında sayısal optimizasyon ve kombinetoral optimizasyon problemleri olan devre tasarımı sistemlerinin çözümünde, doğrusal olmayan denklem sistemlerinin çözümünde ve fabrika-üretim planlamasında kullanılır.
- *Otomatik Programlama*: Genetik algoritma, bilgisayar programları yardımı ile network sıralamasında, ders programı hazırlanmasında kullanılır.
- *Makine öğrenmesi*: GA, robot sensorlerinde, neural networklerde, VLSI yonga tasarımı ve protein yapısal analizinde kullanır.
- *Ekonomi*: GA, ekonomik modellerin geliştirilmesinde ve işlemesinde kullanılır.
- *İmmün sistemler*: GA, aslında çok-gen'li ailelerin evrimi esnasında ve doğal immün sistem modellerinde kullanılır.
- *Popülasyon genetiği*: GA, evrim ile ilgili sorulara cevap bulmada kullanılır.
- *Evrim ve öğrenme*: GA, fertlerin öğrenmesini ve türlerin evrilmesinde kullanılır.
- *Sosyal sistemler*: GA, sosyal sistemlerin analizinde kullanılır.

5.9. Çeşitli Değerlendirme Stratejileri ve GA ile Aralarındaki Farklar

Doğal evrimin benzetimi için diğer bir yaklaşım ise 1960'lı yılların basında Almanya'da teklif edilmiştir. GA'nın aksine bu yaklaşım ise, "evrim stratejisi" olarak adlandırılır ve teknik optimizasyon problemlerinin çözümü için tasarlanmıştır. 1963'te Berlin teknik üniversitesinden iki öğrenci, Ingo Rechenberg ve Hans-Paul Schwefel, akıntıya kapılmış bir kütlenin optimal durumunun araştırılması üzerine bir çalışma yapmışlardır.

Böylece GA ile diğer bir değerlendirme stratejisi arasındaki temel fark "GA çarprazlama ve değişim işlemcilerinin her ikisini de kullanırken, diğerleri sadece

değişim işlemcisini aynı zamanda kullanmaktadır. Yani, diğer bir değerlendirme stratejisi kullanıyorsak problemin kodlanmış formuna ihtiyacımız yoktur.

Değerlendirme stratejileri tamamen sayısal optimizasyon yöntemi kullanmaktadır. (Monte Carlo Arama Yöntemi gibi). GA daha genel uygulamaları gerçekleştirme yeteneğine sahiptir. Fakat bir GA uygulamasının en güçlü yanı ise (bölümü) problemin kodlanmasıdır. Genellikle, herhangi yöntemin en iyi çalıştığı sorusuna cevap verebilmek için çeşitli uygulama denemeleri yapmak zorundayız.

Bilgisayar bilimindeki merkezi problemlerden biriside aslında tam bir programlama yapılmadan bir problemin bilgisayar çözümü nasıl yapılabileceğidir. GA doğal seçim yöntemleri ile bilgisayar programlarının değerlendirmesini kullanan bir çözümü tercih etmektedir. Aslında genetik programlama GA'nın basmakalıp bir açılımıdır, fakat genetik programlamanın amacı tamamen bazı problemlerin bit dizilerini değil daha çok problemi çözecek bilgisayar kodlarını değerlendirmektedir. Diğer bir ifade ile GA çözümü temsil eden ikili (binary) sayılar dizisini oluştururken, GP çözüm olarak bilgisayar programlarını oluşturur.

GP daha çok el ile problem çözmeye uyarlanmış bir programın olurlu (mümkün) bir bilgisayar programları uzayını arar. Herhangi bir bilgisayar programı değerlere uygulanan işlemcilerin bir dizisidir, fakat farklı tipteki deyim ve işlemcileri kapsayabilir ve farklı söz dizimi kurallarına ait sınırlamalara sahiptir. GP genetik işlemcileri uygularken, burada kullanılan programlama dili yeni oluşturulan verilere uygulanmasına ve bilgisayar programlarını yönlendirmeye izin verecek yeteneğe sahip olmalıdır.

6. GEZGİN SATICI PROBLEMİNE GENETİK ALGORİTMA İLE ÇÖZÜM YAKLAŞIMI

GA kullanılarak çözülen problemlerden en günceli gezgin satıcı problemi (travelling salesman problem) dir. Her ne kadar bu problem ve benzerleri için çok sayıda çözüm yöntemi önerilse de bu tür problemleri büyüklükleri ne olursa olsun çözebilecek kesin bir yöntem bulunamadı. Bu nedenle araştırmacılar bilgisayarlar daha etkin kullanmak amacıyla bazı sezgisel yöntemler geliştirmişlerdir. Evrimsel programlama yöntemlerinden GA larda bu yöntemlerden birisidir. Problemin çözümünde genel olarak genetik algoritma esas alınsa da uygulamada farklılıklar olduğundan dikkat çekilmesi gerekir. Çözümün esas genetik operatörlerin kullanılmasıdır Fakat, bu operatörler özüne bağlı kalınarak değişik şekilde kullanılmıştır.

Bu bölümde ise genetik algoritma kullanılarak gezgin satıcı probleminin çözüm işleminde kullanılmakta olan genetik kodlamaların bir örnek ile açıklanmasına,yer verilecektir. Gezgin satıcı probleminde kullanılmakta olan genetik algoritma genel akış şeması Ek-1' de verilmiştir.

6.1. Literatür Taraması

Araç rotalama problemlerinin çözüm yöntemleri gelişiminde bu yöntemlerin gezgin satıcı problemlerinden yola çıkarak oluştuğu görülmektedir [37]. 1960'lı yıllardan başlayarak 30 yıllık dönemin sonunda, yaklaşık 50 müşteriye kadar bazı problemler optimal olarak çözülebilmiştir. 1990'lı yıllarda,araştırmalar modern bulgusal yöntemlere kaymıştır.Literatürde metahuristikler olarak geçen yöntemler arasında yasaklı arama,tavlama benzetimi,karınca kolonileri ve genetik algoritmalar sayılabilir [38]. Çalışmaya konu olan genetik algoritma ise Darwin'in çevre şartlarına uyum sağlayabilen en iyilerin hayatta kalması ilkesine dayanmaktadır [39]. Sinir ağları ve genetik algoritmaların araç rotalama problemlerinin çözümünde karma olarak kullanıldığı hibrid bir yöntem de Potvin, Dube ve Robillard tarafından önerilmiştir [40]. Baker ve Ayechew'in hibrid genetik algoritması yayınlanmış en iyi çözümden %0.50 daha uzak bir sonuç vermiştir [40]. Christian Prins ise daha farklı bir kodlama

yöntemi kullanarak hibrid bir genetik algoritma önermiştir.Farklılığı ise, gezgin satıcı probleminde de kullanılan permütasyon tipi bir kodlama kullanması ve mutasyon operatörü olarak klasik operatörlerin kullanıl mayıp onun yerine bir yeel arama prosedürünün kullanılmasıdır.En iyi çözüm değerlerine göre %0,8 uzak sonuç vermiştir. Golden'a ait 20 test problemi üzerinde yapılan çalışmalarda ise önerdikleri hibrid genetik algoritmalar 11 test problemi içi yayınlanmış en iyi sonuçlardan daha iyi sonuç vermiş, toplam olarak da yayınlanmış en iyi sonuçlardan %0,78 daha iyi sonuç üretmiştir [41].

Sezgisel bir yöntem olan genetik algoritmalar açısından GSP bir karşılaştırma ve değerlendirme ölçütü işlevi de görmektedir. Gezgin satıcı probleminin çözümünde, genetik algoritmalar başarılı bir şekilde uyarlanabilmiştir. GSP' de genetik algoritma yaklaşımı Potvin tarafından sunulmuştur[42]. Günümüz de rastgele aramaların kullanımı giderek artmaktadır. Bu tip aramalar eniyilemenin yerine daha iyi yapma amacını sağlayarak da başarılı olmaktadırlar. İnsanların bilgisayarlardan genel beklentisi mükemmellik olduğu için bu tip aramalar başarısız görünebilir. Genetik algoritmalar klasik yöntemlerin çok uzun zamanda yapacakları işlemleri kısa bir zamanda çok net olmasa da yeterli bir doğrulukla yapabilir.

Ayrıca sadece GA'ların değil, aynı zamanda diğer tüm yeni algoritmaların (karınca sistemi, evrimsel yöntemler, sinir ağları, tabu arama, benzetimli tavlama, açgözlü aramalar, vb.) karşılaştırılıp , değerlendirilmesinde ve de aynı zamanda ölçülmesinde de kullanılmaktadır [11,43]. Bu sezgisel yöntemler makul bir sürede iyi sonuçlar sağlamaktadırlar. Johnson vd. yaptıkları çalışmada yaklaşık sonuçlar üreten bazı yerel arama metotlarını da incelemişlerdir [44].

GSP çözümünde genetik algoritma kullanımı, problemin optimal ya da en iyi çözümünü elde etmek için ilgili evrim mekanizmasının elde edilmesi üerine olan araştırmalarda oldukça yer almaktadır. GA kullanılarak GSP çözümünde literatürde pek çok sayıda farklı çarprazlama metodu önerilmiştir. Son zamanlarda kullanılan operatörleri içerenlerin bazıları şu şekildedir:Sıralı çarprazlama [34,45-47] , Parçalı eşleme çarprazlaması [7,34,46,47] ,Döngü çarprazlaması [34,46,47] .

Tez çalışması,literatürde yer alan genetik algoritma uygulamalarının etkinliğini sergilemek amacı ile kodlama ile hazırlanan programda çözümünü ve dğrusal programlama ile kıyaslamasını kapsamaktadır.

6.2. Başlangıç Popülasyonunun Oluşturulması

Gezgin satıcı problemi için başlangıç popülasyonunun oluşturulmasında öncelikli olarak gezilecek şehir sayısının belirlenmesi de gerekmektedir. Belirlenen bu şehir sayısına bağlı olarak başlangıç kromozomları oluşturulur. Kromozomlarının oluşturulması işleminde kromozomların her bir genine rassal olarak şehirler atanır, bu atama işlemi sırasında alt kromozomların engellenmesi amacı ile her atama işleminde daha önceden atanmış bütün genler kontrol edilir ve atama işlemi ise, kontrol doğru değeri sağlayana kadar rassal üretime devam edilir. Kromozomlarının oluşturulmasında kullanılan işlem akış şeması Ek-2' de verilmiştir. Yapılan işlemlerin daha açık anlatılabilmesi için örnek bir problem üzerinde işlemler gösterilecektir. Bu amaç doğrultusunda rassal olarak oluşturulan 8 şehirli, 6 adet başlangıç kromozomu Şekil 6.1' de verilmiştir.

Kromozom 1	**5**	**1**	**4**	**2**	**3**	**8**	**7**	**6**
Kromozom 2	**6**	**8**	**2**	**7**	**3**	**5**	**1**	**4**
Kromozom 3	**4**	**7**	**2**	**1**	**6**	**5**	**3**	**8**
Kromozom 4	**8**	**6**	**1**	**3**	**4**	**2**	**5**	**7**
Kromozom 5	**7**	**2**	**3**	**4**	**1**	**6**	**5**	**8**
Kromozom 6	**8**	**5**	**6**	**2**	**7**	**3**	**1**	**4**

Şekil 6.1. Başlangıç kromozomları

6.3. Kromozom Uygunluğunun Hesaplanması

Bu aşamada aslında önceden oluşturulan başlangıç kromozomları problemimizde turları ifade etmektedir. Her bir tur uzaklık matrisi kullanılarak sahip oldukları tur mesafeleri hesaplanmaktadır; ve de hesaplanan bu mesafeler her bir kromozomun uygunluk değeridir. Hesaplamada kullanılan uzaklık matrisi Ek-6' da verilmiştir.

$$\text{Min } Z = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} * X_{ij} \qquad (6.1)$$

Çizelge 6.1. Kromozom uygunluk değerleri

	Uygunluk Değerleri F(x)		Uygunluk Değerleri F(x)
Kromozom 1	67	Kromozom 4	72
Kromozom 2	65	Kromozom 5	70
Kromozom 3	47	Kromozom 6	63

Programın içinde kullanılan kromozom uygunluk değerlerinin hesaplanmasında kullanılan akış şeması Ek-3' de verilmiştir.

6.4. Yeni Kromozomlarının Seçilmesi

Gezgin satıcı probleminin çözüm sürecinin bu aşamasında ise rulet seçim tekniği kullanılmaktadır. Seçim işleminin akış şeması ise Ek-4'de verilmektedir. Problemin çözümlenmesinde seçim aşamaları şu şekilde sıralanmıştır;

- Eldeki popülasyonda en büyük uygunluk değerine sahip olan kromozom seçilir. Aşağıda, işlemlerin genel akış süreci verilmiştir:

```
f:=0;  i:=0;
Repeat
  i:=i+1;
   if f >= f(xi) then
   begin
   f:=f (xi) ;
  end;
Until i= pop_say;
```

- Gezgin satıcı problemi amaç fonksiyonunun minimizasyon olmasından dolayı en yüksek uygunluk değeri olan bütün kromozomlardan çıkartılarak yeni uygunluk değerleri bulunur. Aşağıda işlemin prosedürü verilmiştir:

```
i:=0;
 Repeat
 i:=i+1;
Yf (xi):=1+f- f (xi) ;
Until i=Popülasyon sayısı
```

Burada dikkat edilmesi gereken her çıkarma işlemine +1 eklenmesidir bunun nedeni ise belli bir jenerasyondan sonra, popülasyondaki bütün kromozomların birbirine benzemekte ve $_{min}\, F(x) = {}_{max}\, F(x)$ olmaktadır. Böyle bir durumda çıkarma işleminden kalan her bir kromozom için 0 uygunluk değeri olacaktır. Bunun anlamı bir sonraki adımda gerçekleştirilecek olan kromozom değerlerinin toplam değerini 0 yapacaktır ve böylece kromozomlar arasında seçim yapılmasına engel olacaktır.

- Bu aşamada yeni bulunan kromozomlarının uygunluk değerleri toplanır. Toplama işlemi fonksiyonu prosedürü aşağıda verilmiştir:

```
for i:=1 to pop_say do
  begin
  TF(xi):=TF(x[i-1] +Yf(xi)
  end;
```

Seçim aşamasının en önemli kısmı kromozomlar arasında seçimin yapılmasıdır. Seçimin yapılmasında birkaç yöntemden biri olan rulet tekerleği tekniği bizim problemimizin çözümlenmesinde de kullanılmıştır. Seçim işleminde toplam F(x) değerine göre kromozom sayısı -1 kadar rassal sayı üretilir. Üretilen her sayı ise toplam fonksiyonda hangi kromozomun sahip olduğu aralığa da isabet ederse o kromozom seçilir. Burada kromozom sayısı -1 tane seçim yapılmasının nedeni en iyi uygunluk değerine sahip olan kromozomun seçilememe ihtimali göz önünde tutularak seçime tabi tutulmadan direkt yeni popülasyona seçilir. Bu işlemin prosedürü gösterildiği gibidir:

```
for i:=1 to Pop_Say do
 begin
 r:=random(TF(xn));
 Yf(xi):=c;
 pos:=-1;
  repeat
    pos:=pos+1;
     if (r>= TFj(x) and (r<= TFj(x) then
      begin
      secim(xi):=pos+1;
      end;
  until pos=pop_say;
```

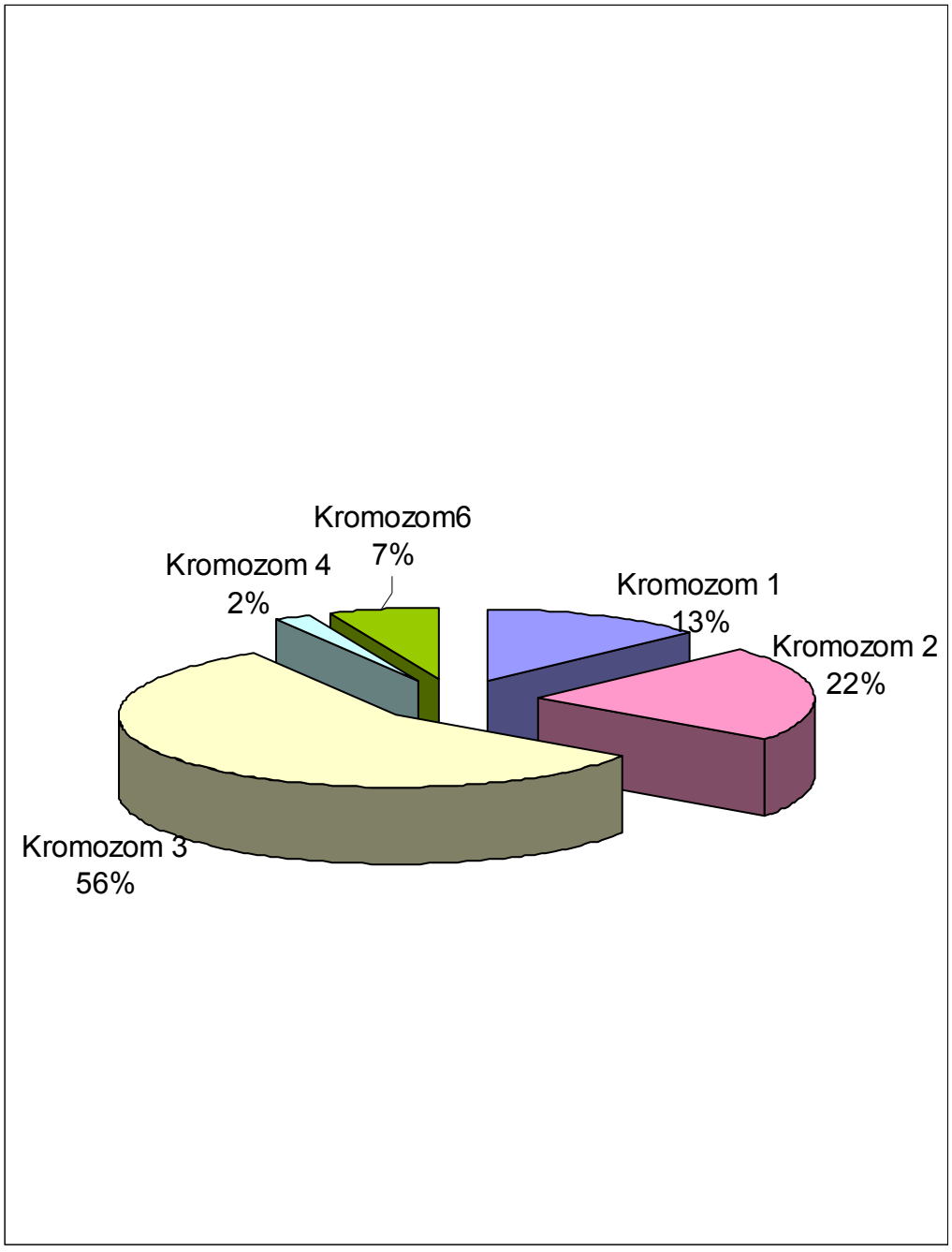
Çizelge 6.2. Kromozom sonraki uygunluk değerleri

	Uygunluk Değerleri F(x)	Yeni Uygunluk Değerleri	Uygunluk Değerleri Toplamı
Kromozom 1	67	1+72-67 =6	6
Kromozom 2	65	1+72-65=8	6+8=14
Kromozom 3	47	1+72-47=26	14+26=40
Kromozom 4	72	1+72-72=1	40+1=41
Kromozom 5	70	1+72-70=3	41+3=44
Kromozom 6	63	1+72-63=10	44+10=54
	Max F(x)=72		F(x)=∑ 54

Bulunan uygunluk değerine göre kromozomların rulet tekerleği tekniğine göre sahip oldukları seçim yüzdeleri Şekil 6.2' de verilmiştir.

Çizelge 6.3. Kromozom Seçimi

	Üretilen Rassal Sayı	Seçilen Rotalar
Deneme 1	3	Kromozom 1
Deneme 2	16	Kromozom 3
Deneme 3	49	Kromozom 6
Deneme 4	19	Kromozom 3
Deneme 5	41	Kromozom 4



Şekil 6.2. Kromozom seçim oranları

Yeni kromozomun seçiminin son aşaması olan uygunluk değerlerine göre seçilmiş kromozomun yeni kromozom kümesine atanması ile bitmektedir. Yeni oluşturulan popülasyon Şekil 6.3' de verilmiştir.

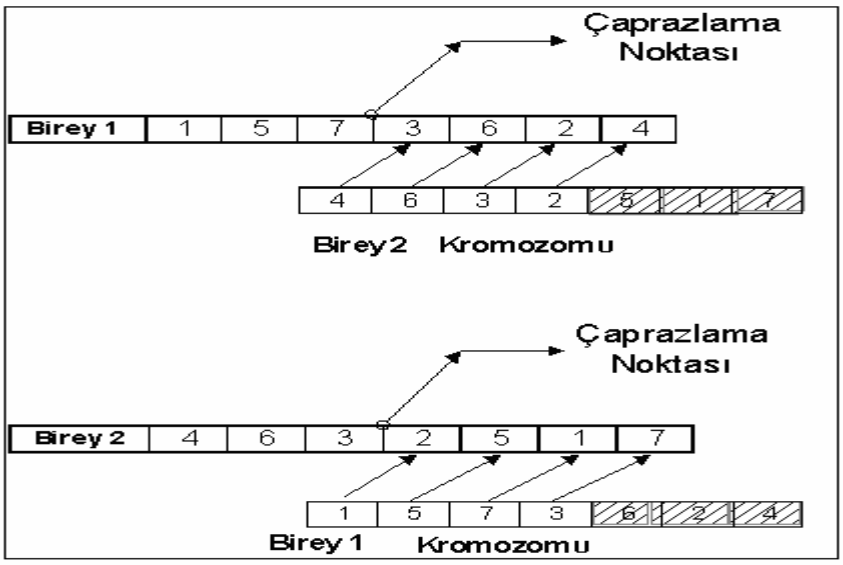
Seçim 1	Kromozom 3	4	7	2	1	6	5	3	8
Seçim 2	Kromozom 1	5	1	4	2	3	8	7	6
Seçim 3	Kromozom 3	4	7	2	1	6	5	3	8
Seçim 4	Kromozom 6	8	5	6	2	7	3	1	4
Seçim 5	Kromozom 3	4	7	2	1	6	5	3	8
Seçim 6	Kromozom 4	8	6	1	3	4	2	5	7

Şekil 6.3. Yeni popülasyon

6.5. Çarprazlama

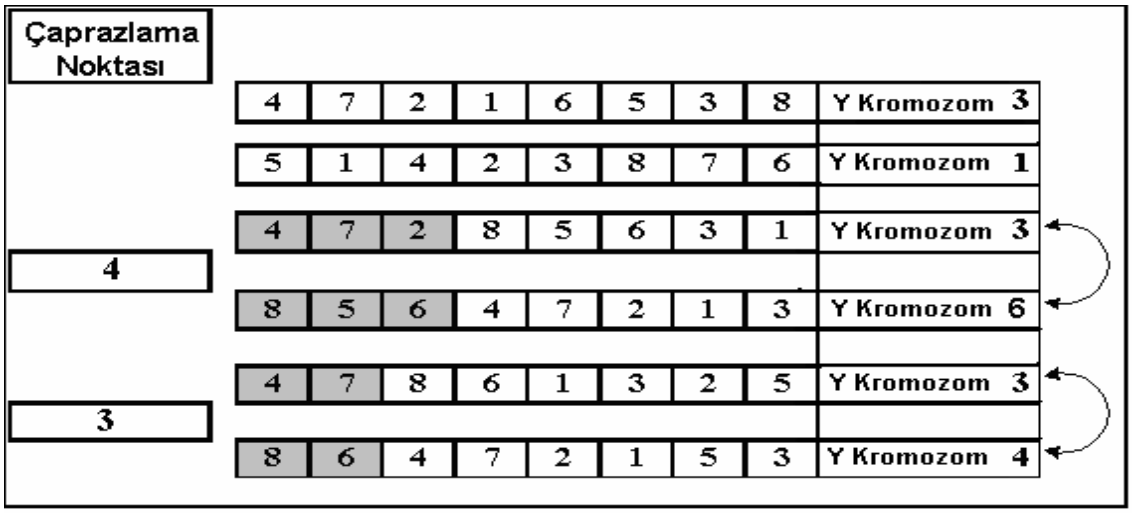
Gezgin satıcı probleminin çözümlenmesinde kullanılacak olan çarprazlama işleminin aşamaları aşağıdaki gibi gerçekleşmiştir.

Çarprazlama işleminin başlangıcı ise aslında ilk olarak çarprazlama noktasının belirlenmesidir.Çarprazlama noktası kromozom genişliği dikkate alınarak belirlenmelidir. Belirlenen çarprazlama noktası 2'şerli kromozom çiftlerinde kullanılmalıdır. Problemimiz için çarprazlama işleminde dikkat edilmesi gereken nokta ise alt turların engellenmesidir. Çarprazlama noktasından itibaren kromozomlar arasındaki gen takası işleminde kromozomlar da bulunan genlerin kromozoma aktarımı söz konusudur.



Şekil 6.4.Çarprazlama işlemi

Şekil 6.4'de gösterildiği gibi 3. noktadan sonra gen takasları gerçekleştirilmiştir. Bu işlemde de daha önceki bölümde anlatıldığı gibi çarprazlama noktalarından sonraki genler direkt olarak yer değiştirmezler. İşlemde çarprazlamaya tabi tutulacak birey 1 çarprazlama noktasından sonra çarprazlanacağı bireyin 2'nin 1. kromozomundan itibaren genleri kontrol edilir ve birey 1 çarprazlama noktası öncesindeki genler birey 2 'den gelecek genlerden silinir ve daha sonrasında birey 2' de kalan genler birey1 'in çarprazlanma noktasından sonra direkt atanır. Aynı işlem birey2 için gerçekleşir, böylece bireyler arası çarprazlanma tamamlanmış olur.



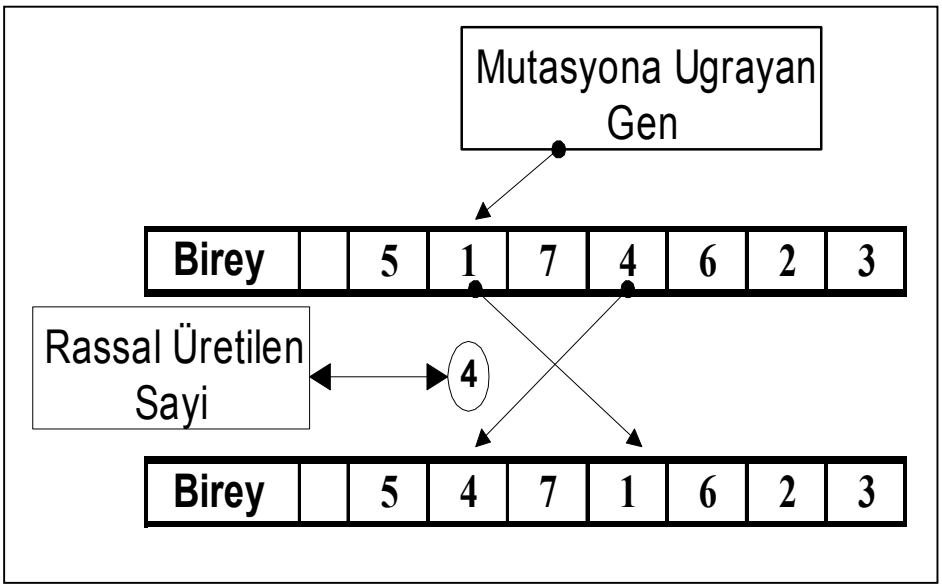
Şekil 6.5. Çarprazlama işlemi sonrası yeni popülasyon

Şekil 6.5'de örnek olarak anlatılan problemde ise kromozomların çarprazlama işleminden sonra ki yeni kromozomlar verilmiştir. Burada dikkat edilmesi gereken önemli nokta ise, 1. ve 2. bireyler çarprazlanmaya tabi tutulmamasıdır. Bunun nedeni en iyi uygunluk değerine sahip olan bireyin aslında çarprazlanmaya tabi tutularak kaybedilmesini önlemektir. Programımızda kullanılan çarprazlama işlemi akış şeması Ek-4 ' de verilmiştir.

6.6. Mutasyon

Mutasyon işlemi,çalışmada uygulanması gereken genlerin aslında yer değiştirme esasına dayanmaktadır. Problemimiz de belirlenen %3 olan mutasyon olasılığı ile her kromozomun her bir genine işlem uygulanmaktadır. Mutasyon işlemi akış şeması Ek-5' de verilmiştir.

Mutasyona uğrayacak olan genin değeri X_1 olsun rassal olarak kromozom genişliği aralığında üretilen değer X_2 olsun. Bulunan X_2 değeri X_1 yerine konur kromozomun herhangi bir geninde bulunan X_2 değeri yerinede X_1 yazılır ve böylece çarprazlama işlemi tamamlanmış olur. Şekil 6.6'da mutasyon işlemi bir örnekte gösterilmiştir. .



Şekil 6.6. Mutasyon işlemi

Şekil 6.6’ da görüldüğü gibi kromozomun 2. geni mutasyona uğramıştır. İşlemde 2. genin değeri olan 1 rassal üretilen 4 değeri ile yer değiştirmektedir. Kromozomda 4 değerine sahip olan 4. gene de 1 değeri atanarak mutasyon işlemi sonlandırılmıştır.

										Mutasyona Uğrayan Genler	Rassal Üretilen Sayı
Y Kromozom 3		4	7	2	1	6	5	3	8	-	-
Y Kromozom 1		3	1	4	2	5	8	7	6	1. Gen	3
Y Kromozom 3		2	7	4	8	3	6	5	1	3. ve 7. Gen	4 ve 5
Y Kromozom 6		8	5	7	4	6	2	1	3	5. Gen	6
Y Kromozom 3		4	7	8	6	3	1	2	5	6. Gen	1
Y Kromozom 4		8	3	4	7	2	1	5	6	2. Gen	3

Şekil 6.7. Mutasyon işlemi sonrası yeni popülasyon

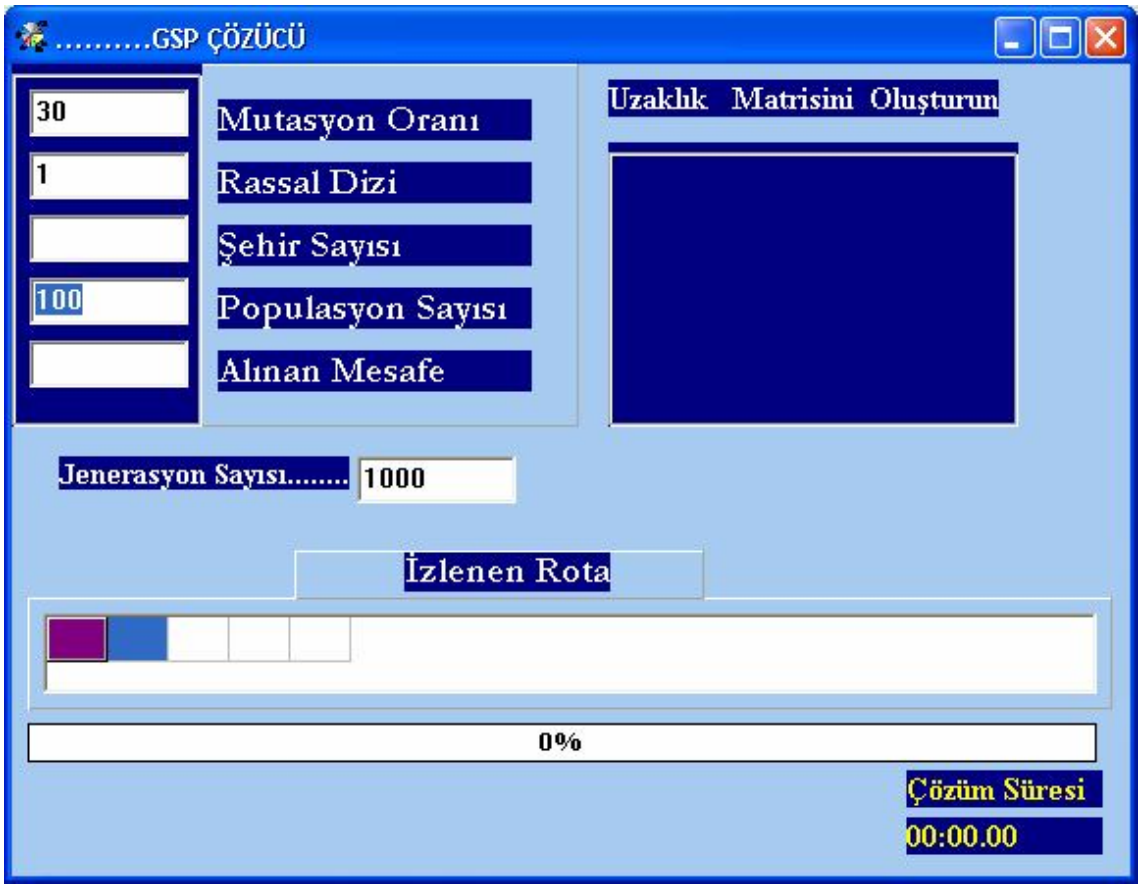
Mutasyon işlemi ile böylece ilk nesil oluşturulmuştur. Mutasyon işlemi ile ilk etap tamamlanmıştır ve bundan sonrasında ise bu aşamaya kadar yapılan her işlem yeni popülasyon oluşturma işlemi haricinde, kabul edilen jenerasyon sayısı kadar tekrarlanır.

7. UYGULAMA

Çalışmanın amacı doğrultusunda yapılmış olan ve Delphi 6 programında hazırlanan programın problemler karşısında ne kadar etkili olduğunu görmek amacı ile bir dizi problem ele alınarak değerlendirilmiştir.

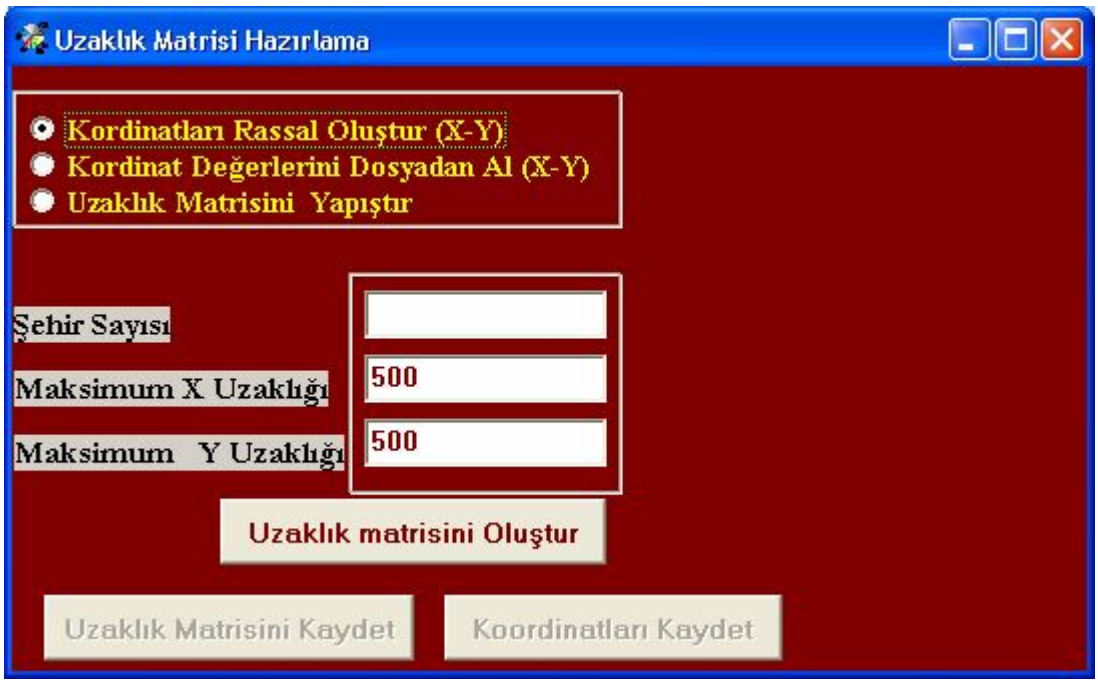
Problem 10, 20, 30, 40, 50, 60, 70, 80, 90 ve 100 şehirli problemler olup her problem için, programın kullanmış olduğu rassal dizi ile 1'den 10' a kadar deneme yapılmış ve sonuçları tablolaştırılmıştır.

7.1. Hazırlanan Programın Tanıtılması



Şekil 7.1. Form1 görüntüsü

Şekil 7.1' de hazırlanan programın ana form görüntüsü verilmektedir. Bu programın başlaması için ilk adım ise mutasyon oranını belirlemek olacaktır. Girilen bu değer 30/1000 olacaktır. Kullanıcının değiştirebileceği ikinci değer ise rassal dizi değeridir. Rassal dizi değeri: programın çalışması esnasında oluşturduğu rassal atama dizisidir. Kullanıcı bu değerleri değiştirmeden de ikinci adım olan 'Uzaklık Matrisi Oluştur' seçeneğini seçtikten sonra,aşağıda Şekil 7.2.' deki form2 ekranını açabilecektir.



Şekil 7.2. Form2 görüntüsü

Koordinatları rassal oluştur

Bu seçeneğin seçilmesinden sonra karşımıza girilmek istenen şehir sayısı ve üretilen rassal sayıların X koordinatı ile Y koordinatları sınırları belirlenir. Bu aşamadan sonra 'uzaklık matrisi oluştur' seçeneği seçilir.

Seçimden sonra 'Uzaklık matrisi kaydet' ve 'Koordinatları kaydet' seçenekleri aktif olacaktır. Üretilen bu değerler bir text dosyası olarak kaydedilir.

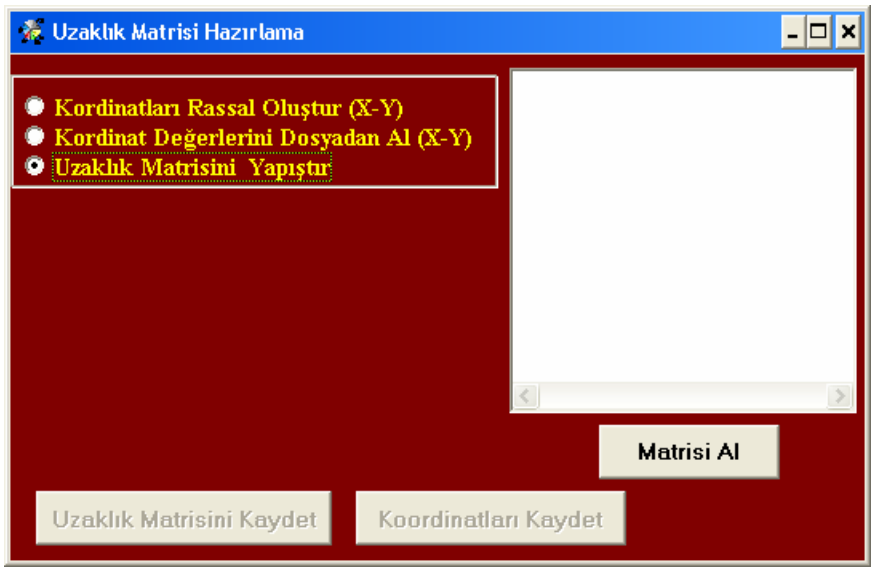
Koordinatları dosyadan al

Belirli bir dosyadan X ve Y koordinatları alınmasını sağlar. Değerlerin alınmasından sonra ise tekrardan ‘Uzaklık Matrisi Oluştur’ seçeneğinin seçilmesi ve aynı zamanda basılması gerekmektedir.

Uzaklık matrisi yapıştır

Bu adımda formun sağında bir kopyalanan uzaklık matrisinin yapıştırılması için bir bölüm ve altında ‘matrisi al’ seçeneği görünecektir. Burada dikkat edilmesi gereken kopyalanan uzaklık matrisinin Excel’ den alınması gerekmektedir.

Son olarak ise ‘Matrisi al’ veya ‘Uzaklık matrisi’ oluştur seçildiğinde Form1 aktif olacaktır.



Şekil 7.3. Form2 uzaklık matrisi ekran görüntüsü

Aktif olan Form1 artık kullanıcıya farklı seçenekler sunmaktadır. ‘Kaydedilecek Dosya’ seçeneği ise program sonuçlarının bir dosyaya kaydedilmesin sağlamaktadır. Bu seçenek seçilmeden ise ‘Hazırla’ ve ‘Hesapla’ seçenekleri aktif olamayacaktır. ‘Jenerasyon sayısı’ ise programın kaç jenerasyon yapmasını istediğimizi belirtir.

Şekil 7.4. Form 1 diğer seçenek görüntüsü

'En İyi Mesafeyi Giriniz' seçilmesi durumunda ise kaşımıza Şekil 7.5' de gösterilen form 3 çıkacaktır. Form 3' de varsa en iyi değer girilerek programın bu en iyi değere istenen bir yüzde yaklaşma değerinden sonra durması sağlanabilecektir. Ayrıca form 3' deki en iyi değere ait çözüm de ilgili dosyadan alınarak çözüm çizdirilebilecektir. Çizdirme istenmiyorsa tur girmek istemiyorum seçilir ve form 1' e geri dönülür.

Şekil 7.5.Form 3 görüntüsü

7.2. Problem Sonuçları

Uygulama bölümünde rassal olarak türetilmiş 10 adet farklı şehirlere sahip, örneğin; 10, 20, …,100 şehirli problemler, hazırlandı. Problemlerde şehirlerin koordinatları rassal oluşturuldu ve bu değerlerden şehirlerin aralarındaki uzaklıklar ise hazırlanan program tarafından bulundu. Hazırlanan programın sonuçlarının ne kadar geçerli olduğunu görmek amacı ile uygulamada en iyi sonucu garanti eden Lingo 5.0 paket programı ile aynı problemler çözdürülmeye çalışıldı. Lingo 5.0 paket programı ile sadece 30 şehre kadar olan problemlerin en iyi değerleri bulunabilmiş, 40 şehirli ve daha fazla şehirli problemlerin çözümü program 25 saat çalışmasına rağmen bulunamamıştır. Problemlerin çözümünde kullanılan şehir koordinatları Ek-7, Ek-8, …., Ek-16' da sıra ile verilmiştir. Verilen bu koordinatlara ait uzaklık matrisleri program tarafından hesaplandığı için verilmemiştir. Uygulama aşamasında her bir problem için sonuç tablosu: bu sonuç tablosunda 1' den 10'a kadar olan rassal dizi sonuçları, çözüm süreleri verilmiştir, ayrıca tabloda Lingo 5.0' ın çözüm süresi, bu sürenin sonunda verilmiş olan sonuç ve iterasyon sayısı verilmiştir. Sonrasında her bir problem için hazırlanan programda bulunabilen iyi sonucun, turu gösteren bir şekil ve her bir jenerasyonda bulunan mesafe değerini göstermesi amacı ile grafik verilmiştir. Turların çizilmiş olduğu şekillerde kalın çizgi ile gösterilen hazırlanan programın sonucunu, ince kırmızı ile gösterilen çizgi ise Lingo 5.0' ın 25 saate kadar bulmuş olduğu sonucu göstermektedir. Bir de en iyi değeri bulunan 10, 20, 30 şehirli problemler için tek bir çizim verilmiştir.

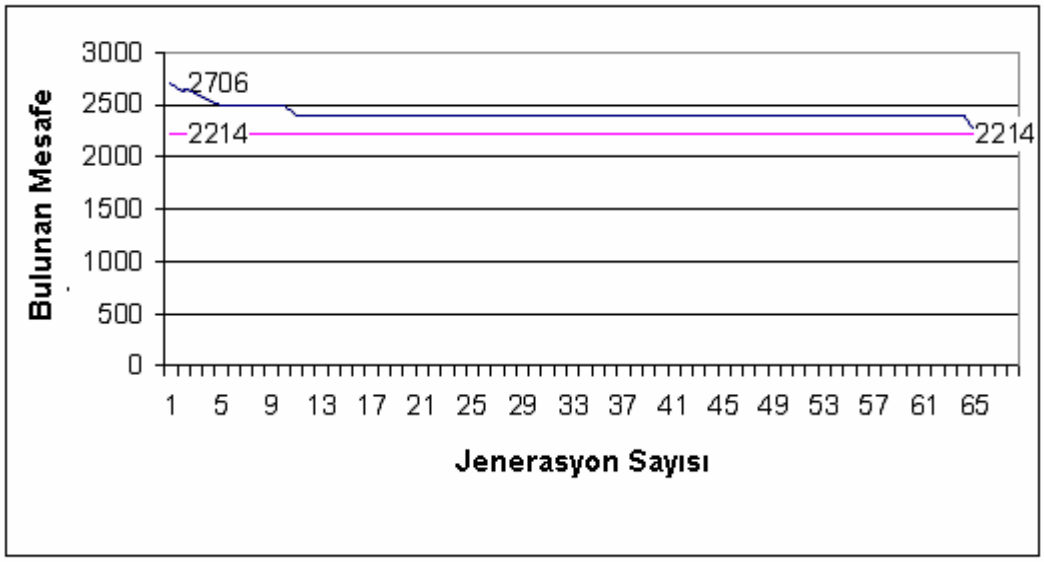
7.2.1. 10 şehirli problem

Programın 10 şehirli problem için vermiş olduğu sonuçlara bakılırsa çözüm süresinin 0,1 sn ile 0,5 sn arasında değiştiği görülebilmektedir. Lingo 5.0 ile sonuçlar karşılaştırılırsa eğer çözüm sürelerin de büyük bir fark olduğu görülmektedir. Bu da yapılan programın çok kısa sürede sonuca ulaştığını göstermektedir.

Çizelge 7.1. Hazırlanan programın 10 şehirli problem için sonuçları

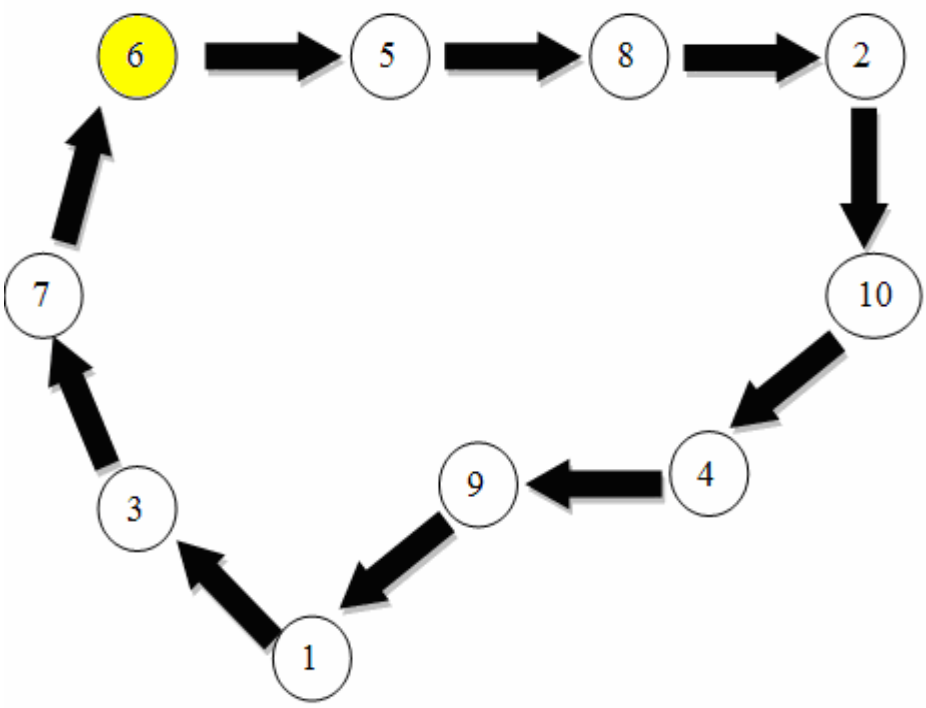
ŞEHİR SAYISI=10			LİNGO 5.0 ÇÖZÜM	2214
MUTASYON ORANI=%3			ÇALIŞMA SÜRESİ	7 sn
			İTERASYON SAYISI	22724
RASSAL DİZİ	JENERASYON SAYISI	BULUNAN UZAKLIK	GEÇEN SÜRE	EN İYİ DEĞERDEN SAPMA MİKTARI (%)
1	68	2214	00:00.180	0
2	13	2214	00:00.060	0
3	28	2214	00:00.080	0
4	30	2214	00:00.080	0
5	90	2214	00:00.200	0
6	45	2214	00:00.110	0
7	301	2214	00:00.501	0
8	9	2214	00:00.030	0
9	55	2214	00:00.120	0
10	98	2214	00:00.190	0

10 şehirli problem için elde edilen en iyi değere sahip olan rassal dizi 1'in her jenerasyonda bulmuş olduğu sonuçlar Şekil 7.6' da verilmiştir.

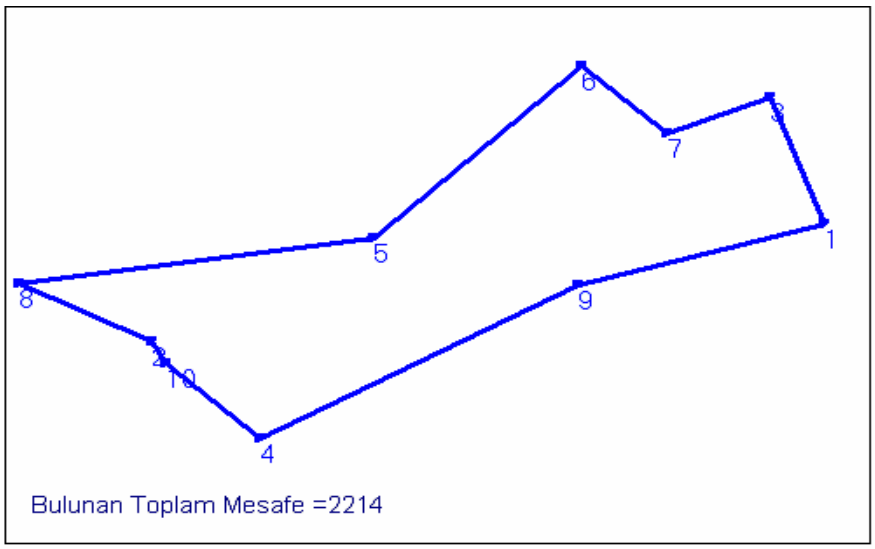


Şekil 7.6. Rassal dizi 1 ile 10 şehirli problem için bulunan mesafeler

Problemin en iyi turu aşağıda verilmiştir, Şekil 7.8 bu tura göre çizilmiştir. Lingo 5. 0 ve hazırlanan programın sonuçlarının aynı olmasından dolayı tek çizim verilmiştir. Programın çalıştırılması ile elde edilen rota ise Şekil 7.7' de verilmektedir:



Şekil 7.7. Program çözümü ile 10 şehir için elde edilen rota



Şekil 7.8. 10 şehirli problem için bulunan en iyi tur

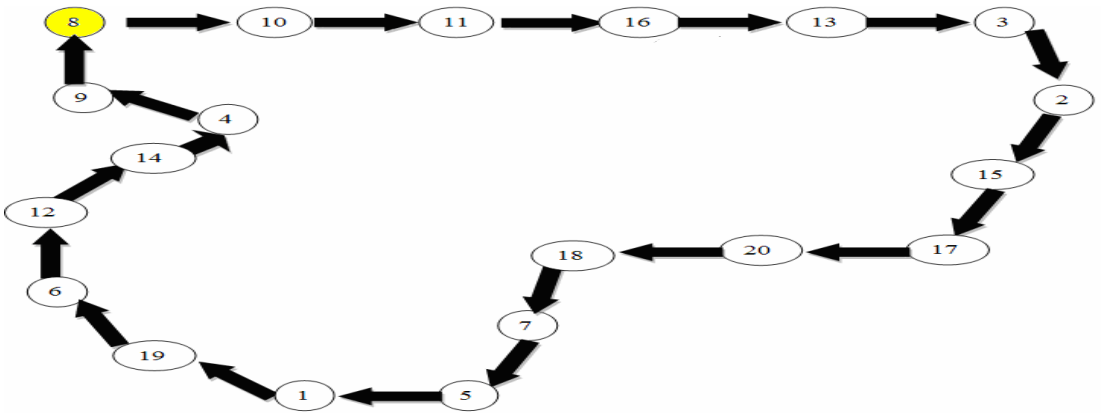
7.2.2. 20 şehirli problem

Programın 20 şehirli problem için vermiş olduğu sonuçlar Çizelge 7.2' de verilmiştir. Sonuçlar Lingo 5.0 ile karşılaştırıldığında ise en iyi değeri kısa bir sürede bulduğu görülmektedir Çözümün bulunması için geçen süre 1 sn ile 9 sn arasında değişmesi ise çözümün ne kadar iyi bir sürede bulunabildiğini göstermektedir.

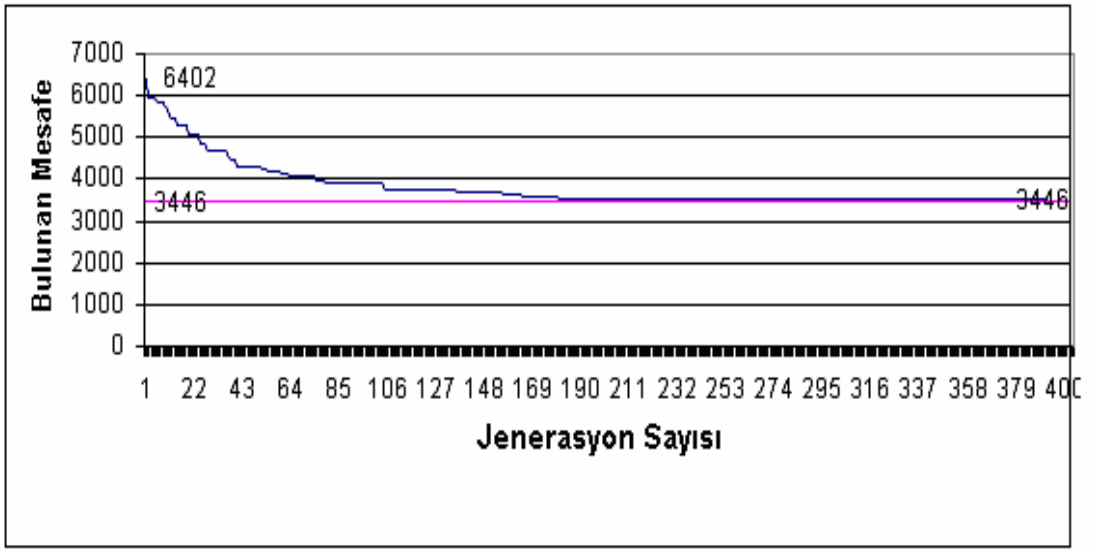
Çizelge 7.2. Hazırlanan programın 20 şehirli problem için sonuçları

ŞEHİR SAYISI=20			**LİNGO 5.0 ÇÖZÜM**	**3446**
			ÇALIŞMASÜRESİ	**12 sn**
MUTASYON ORANI=%3			**İTERASYON SAYISI**	**11628**
RASSAL DİZİ	**JENERASYON SAYISI**	**BULUNAN UZAKLIK**	**GEÇEN SÜRE**	**EN İYİ DEĞERDEN SAPMA MİKTARI (%)**
1	1036	3446	00:02.974	0
2	404	3446	00:01.162	0
3	3404	3446	00:09.504	0
4	408	3446	00:01.161	0
5	1310	3446	00:03.565	0
6	825	3446	00:02.153	0
7	1303	3446	00:03.305	0
8	2553	3446	00:07.250	0
9	625	3446	00:01.532	0
10	1451	3446	00:04.376	0

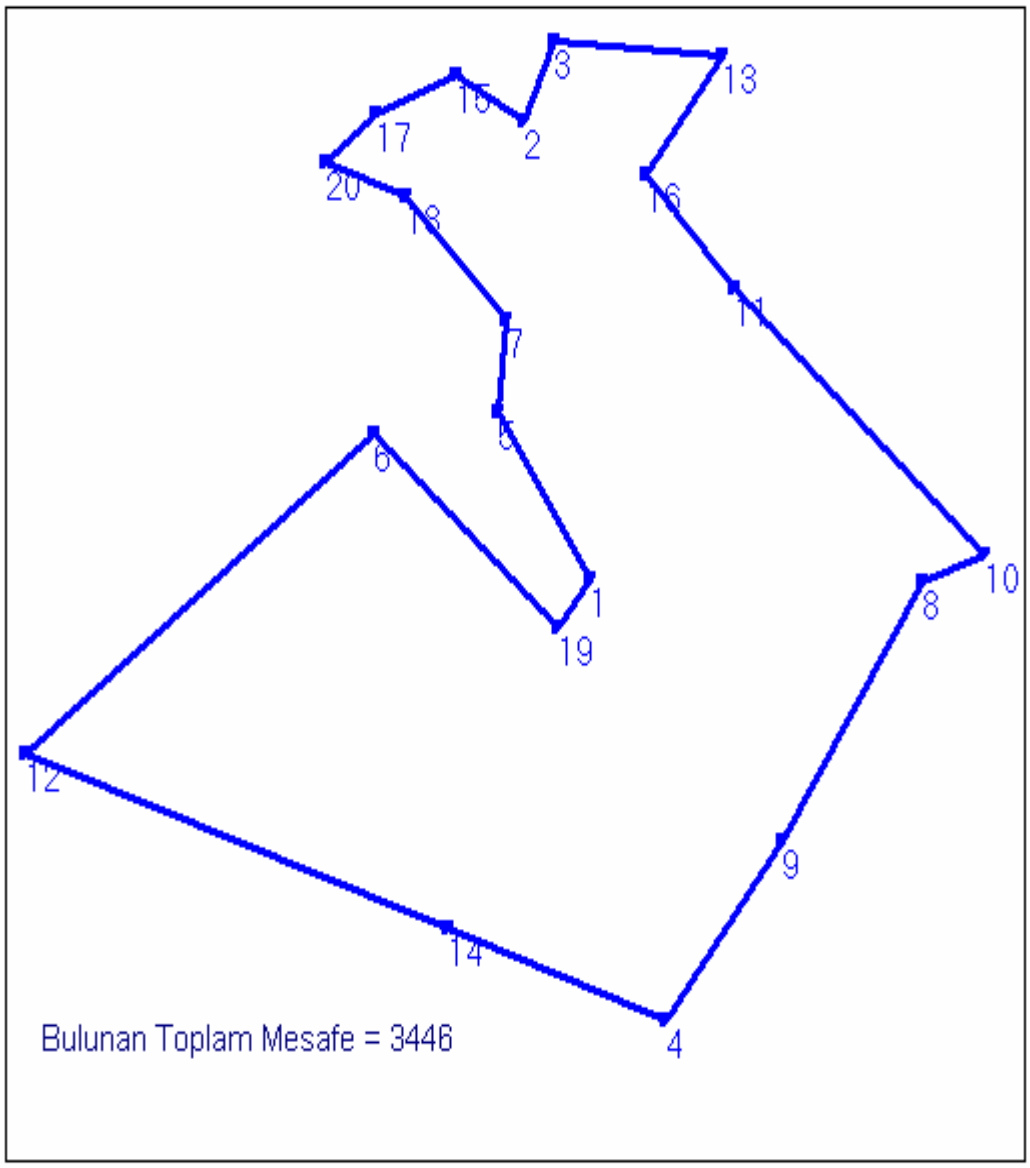
20 şehirli problem için bulunmakta olan en iyi değere sahip olan rassal dizi 2'in her jenerasyonda bulmuş olduğu sonuçlar ise Şekil 7.10' da verilmektedir. Problemin çözümlenmesinden elde edilen tur Şekil 7.9'da yer almaktadır ve bu tura göre Şekil 7.11 çizilmiştir.



Şekil 7.9. Program çözümü ile 20 şehir için elde edilen rota



Şekil 7.10. Rassal dizi 2 İle 20 şehirli problem için bulunan mesafeler



Şekil 7.11. 20 şehirli problem için bulunan en iyi tur

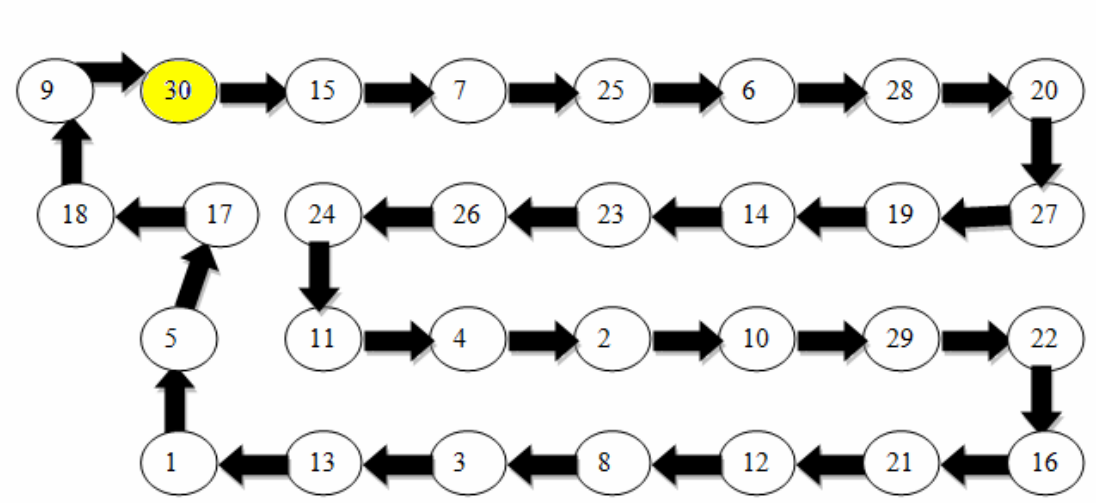
7.2.3. 30 şehirli problem

Hazırlanan programın 30 şehirli problem için vermiş olduğu sonuçlar Çizelge 7.3' de verilmiştir. Sonuçlar incelendiğinde çözüm süresinin 3 sn ile 1 dak 37 sn arasında değiştiği görülmektedir.

Çizelge 7.3. Hazırlanan programın 30 şehirli problem için sonuçları

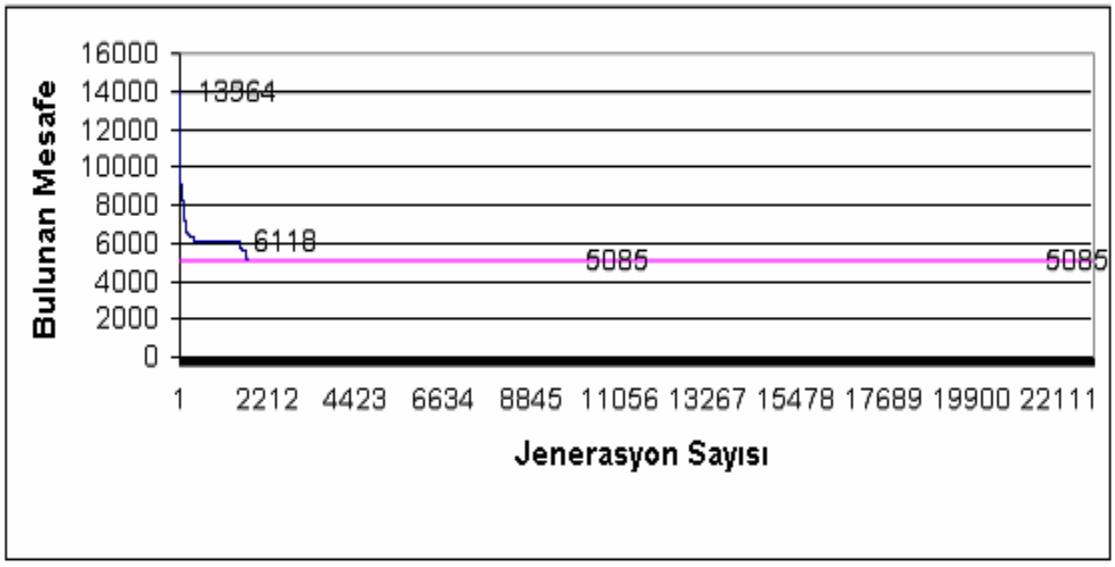
ŞEHİR SAYISI=30			LİNGO 5.0 ÇÖZÜM	5085
MUTASYON ORANI=%3			ÇALIŞMASÜRESİ	2 saat 55 dak
			İTERASYON SAYISI	4939348
RASSAL DİZİ	JENERASYON SAYISI	BULUNAN UZAKLIK	GEÇEN SÜRE	EN İYİ DEĞERDEN SAPMA MİKTARI (%)
1	21223	5085	00:57.773	0
2	9961	5085	00:36.312	0
3	3562	5085	00:13.299	0
4	23799	5085	01:04.042	0
5	23186	5085	01:02.530	0
6	34312	5085	01:37.440	0
7	14067	5085	00:39.627	0
8	1093	5085	00:03.155	0
9	15538	5085	00:44.705	0
10	38507	5085	03:00.700	0

30 şehirli problemin en iyi turu aşağıda verilmiştir ve bu tura göre Şekil 7.14 çizilmiştir.

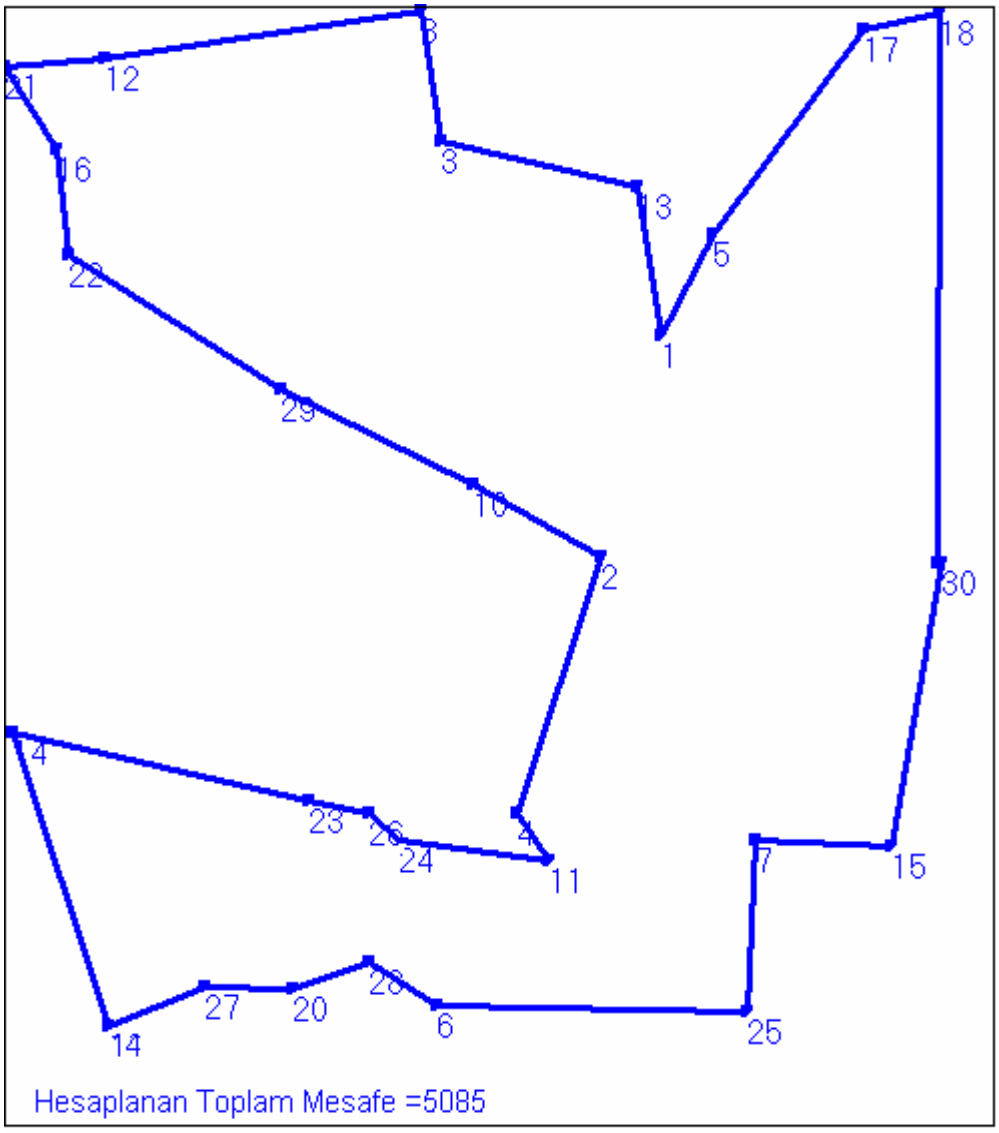


Şekil 7.12. Program çözümü ile 30 şehir için elde edilen rota

30 şehirli problem için bulunan en iyi mesafeye sahip olan rassal dizi 5'in her jenerasyonda bulduğu mesafe Şekil 7.13' de verilmiştir.



Şekil 7.13. Rassal dizi 5 ile 30 şehirli problem için bulunan mesafeler



Şekil 7.14. 30 şehirli problem için bulunan en iyi tur

7.2.4. 40 şehirli problem

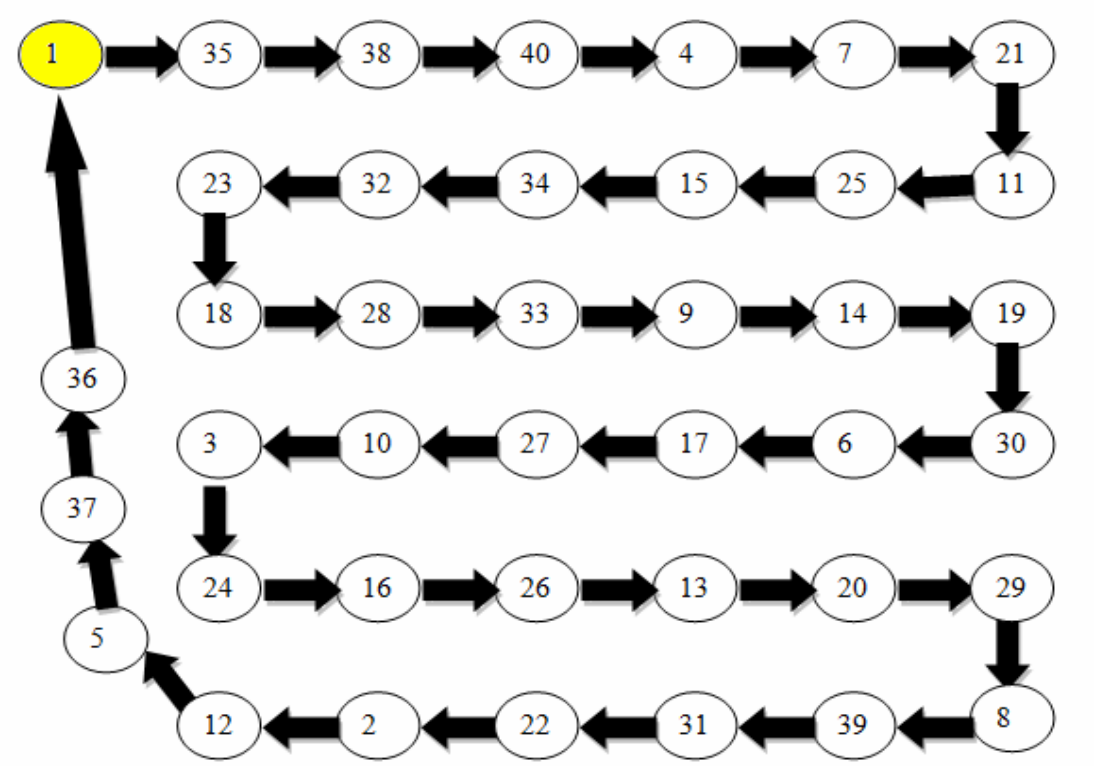
40 şehirli problemin sonuçlarına bakıldığında ise bulunan değerin en iyi değer olduğunun bilinmesi gereken programın ne kadar etkili çalıştığı hakkında pek yorum yapılamamaktadır. Ancak programın 1 dak 26 sn de bulmuş olduğu 4848 değerinin çizimine bakıldığında en iyi değere yakın olduğu söylenebilir. Bunun yanında kısa bir sürede Lingo 5.0 ile bulunan değerden daha iyi bir değer bulunması programın en iyiyi garanti etmemesine rağmen iyi sonuçlar verebildiğinin söylenmesine neden oluyor.

Çizelge 7.4. Hazırlanan programın 40 şehirli problem için sonuçları

ŞEHİR SAYISI=40			**LİNGO 5.0 ÇÖZÜM**	**7738**
			ÇALIŞMASÜRESİ	**25 saat**
MUTASYON ORANI=%3			**İTERASYON SAYISI**	**43000000**
RASSAL DİZİ	**JENERASYON SAYISI**	**BULUNAN UZAKLIK**	**GEÇEN SÜRE**	**EN İYİ DEĞERDEN SAPMA MİKTARI (%)**
1	14950	4869	01:22.919	-
2	34319	4960	03:13.689	-
3	25893	4893	02:08.725	-
4	23799	5085	01:04.042	-
5	19034	4848	01:26.684	-
6	12517	4893	00:47.308	-
7	55649	4893	03:28.099	-
8	30628	4869	01:56.067	-
9	10198	4869	00:41.189	-
10	158412	4960	09:41.266	-

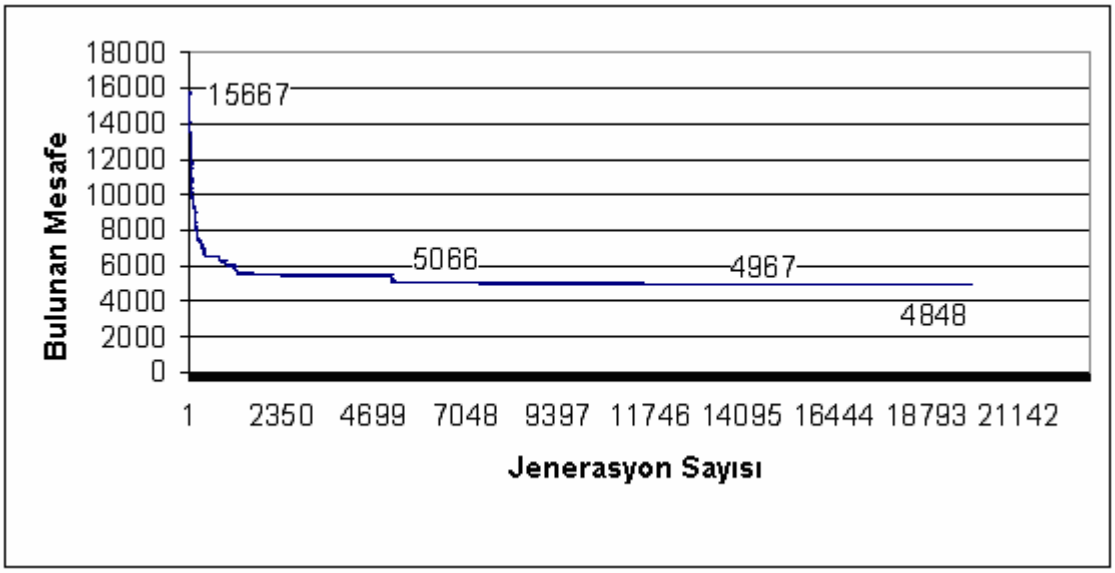
İlgili çözüm çıktıları değerlendirildiğinde açıkça görülecektir ki , Lingo 5.0 programının 25 saat çalıştırılmasına rağmen en iyi sonuç elde edilememiştir.Bu nedenle çizelgede Genetik algoritma ile kodlanan program sonuçlarının en iyi değerden sapma miktarı yer almamaktadır.

Programın ve Lingo 5.0’ ın bulmuş olduğu turlar aşağıda verilmiştir. Bu turlara göre Şekil 7.17 çizilmiştir. Şekil 7.17‘ de iki programında farklı sonuç vermesinden dolayı iki çizim yapılmıştır. Çizimlerden mavi renkli olan çizgiler programın, kırmızı renkli olan çizgiler Lingo 5.0’ ın vermiş olduğu çizimlerdir.

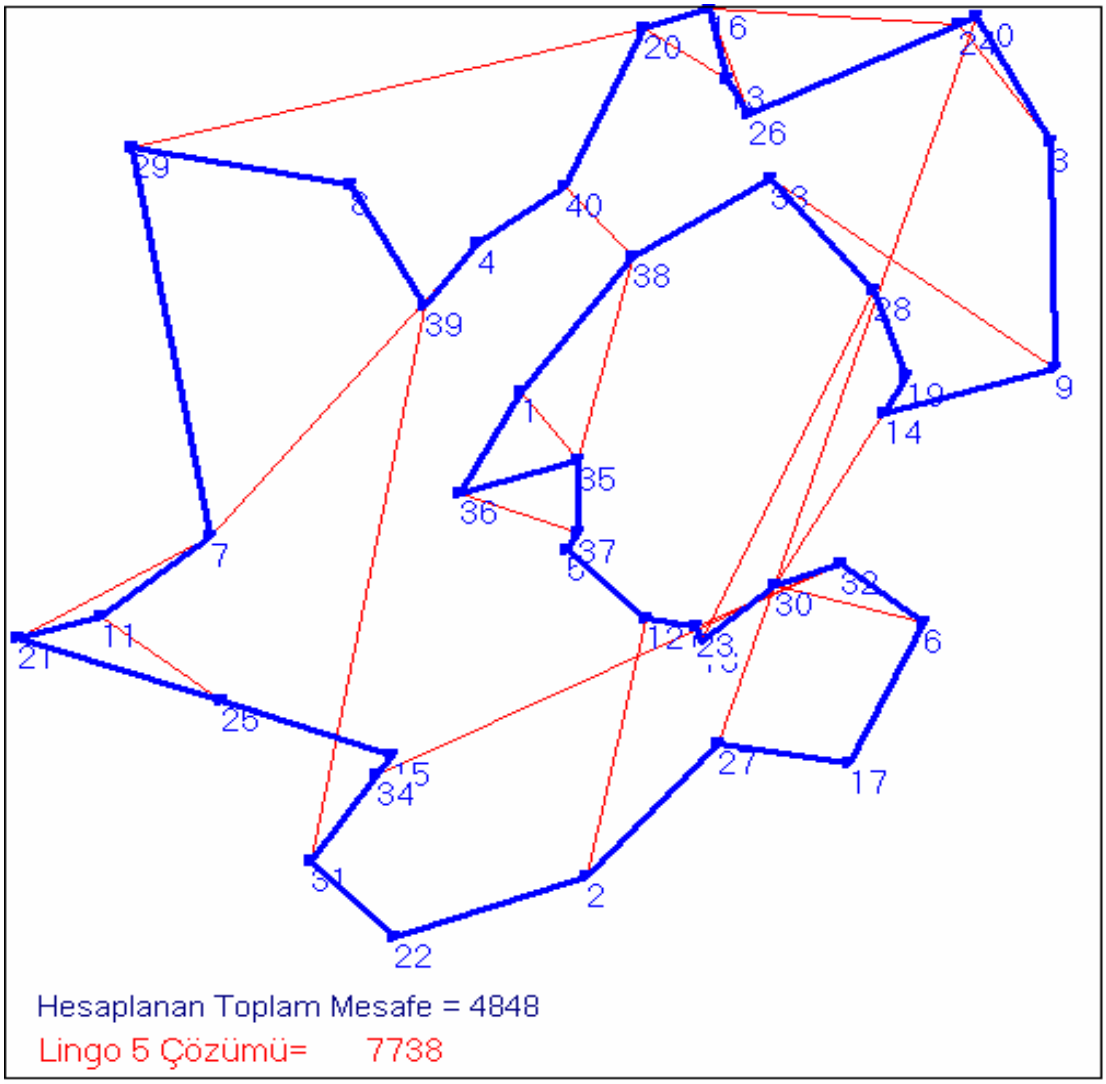


Şekil 7.15. Lingo 5.0 Çözümü ile 40 Şehir için Elde Edilen Rota

40 şehirli problem için bulunan en iyi mesafe sahip olan rassal dizi 5'in her jenerasyonda bulmuş olduğu mesafe Şekil 7.16' da verilmiştir.



Şekil 7.16. Rassal dizi 5 ile 40 şehirli problem için bulunan mesafeler



Şekil 7.17. 40 şehirli problem için bulunan turlar

7.2.5. 50 şehirli problem

Programın 50 şehirli problem için bulduğu sonuçlar Çizelge 7.5' de verilmiştir. Hazırlanan program az zaman da oldukça düşük bir sonuç ile Lingo 5.0' ın uzun bir süre çalışmasına rağmen bulmuş olduğu değerden, ayrılmaktadır. Ancak bulunan bu sonucun en iyi değere ne kadar yakın olduğu hakkında bilgi sahibi olunmaması, hazırlanan program sonuçlarının tam anlamıyla yorumlanmasına engel olmaktadır. Diğer problemlerde olduğu gibi hazırlanan program sonuca çabuk yakınsamaktadır.

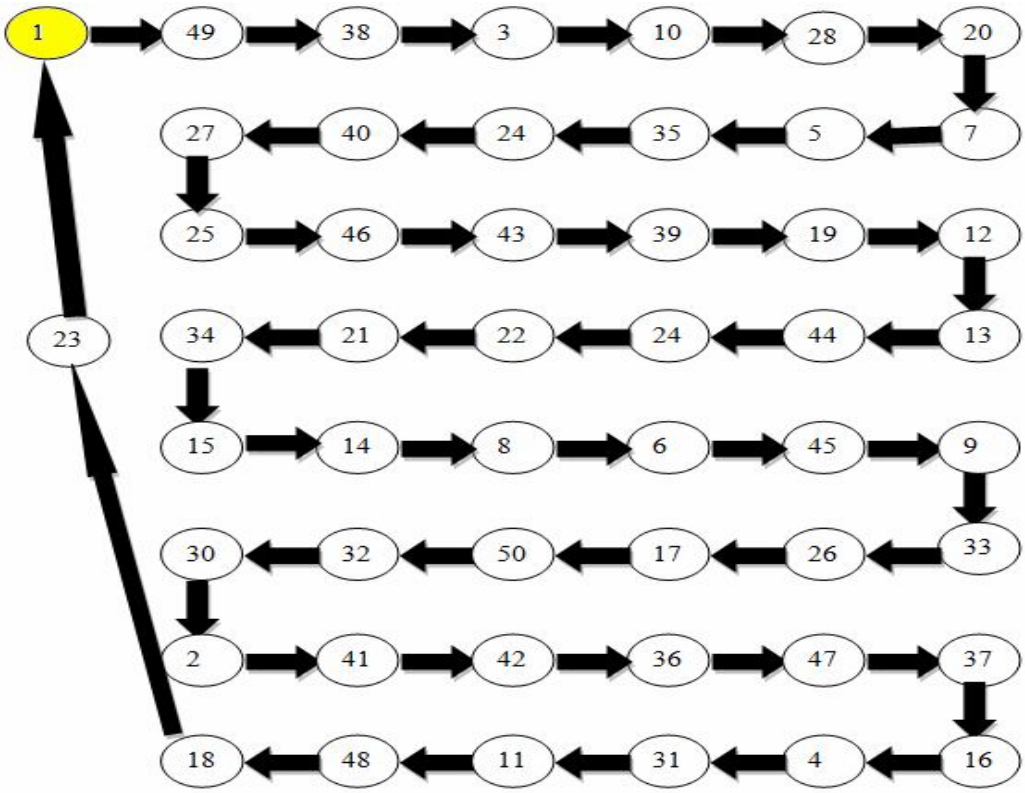
Ama rassal dizilerin hepsinde 5886 bulunamaması, hazırlanan programın vereceği sonucun başlangıç popülasyonuna bağımlı olduğu görülmektedir.

Çizelge 7.5. Hazırlanan programın 50 şehirli problem için sonuçları

ŞEHİR SAYISI=50			**LİNGO 5.0 ÇÖZÜM**	**9030**
			İTERASYON SAYISI	**110000000**
MUTASYON ORANI=%3			**ÇALIŞMA SÜRESİ**	**25 saat**
RASSAL DİZİ	**JENERASYON SAYISI**	**BULUNAN UZAKLIK**	**GEÇEN SÜRE**	**EN İYİ DEĞERDEN SAPMA MİKTARI (%)**
1	231474	5994	25:56.488	-
2	139235	5886	14:43.300	-
3	181622	5992	20:31.351	-
4	113996	6014	11:30.573	-
5	121499	6014	14:06.598	-
6	191830	5978	19:02.092	-
7	152020	5893	14:45.273	-
8	92157	6047	09:21.177	-
9	68889	6395	05:40.049	-
10	111491	5886	09:12.955	-

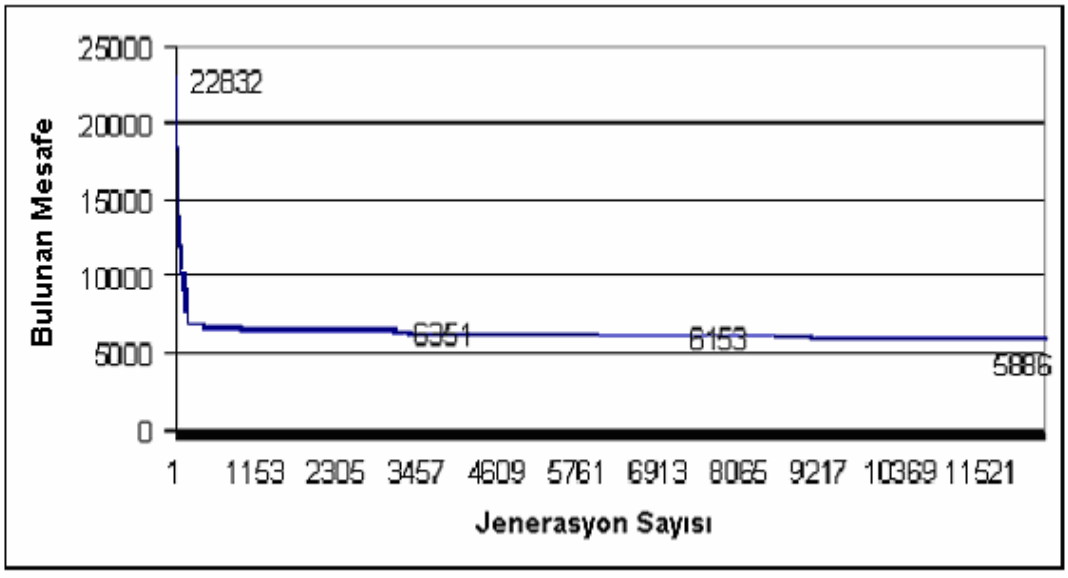
Bu çizelgede de benzer şekilde 40 şehirli problemde en iyi çözümün elde edilememesi gibi 50 şehir sayısı için de 25 saatten sonra program durdurulmuştur.Bu nedenle en iyi değerden sapma miktarı ele alınamamıştır.

Problemin, Lingo 5.0 ve hazırlanan programa göre bulmuş olduğu turlar aşağıda verilmiştir ve bu turlara göre Şekil 7.20'de çizilmiştir.

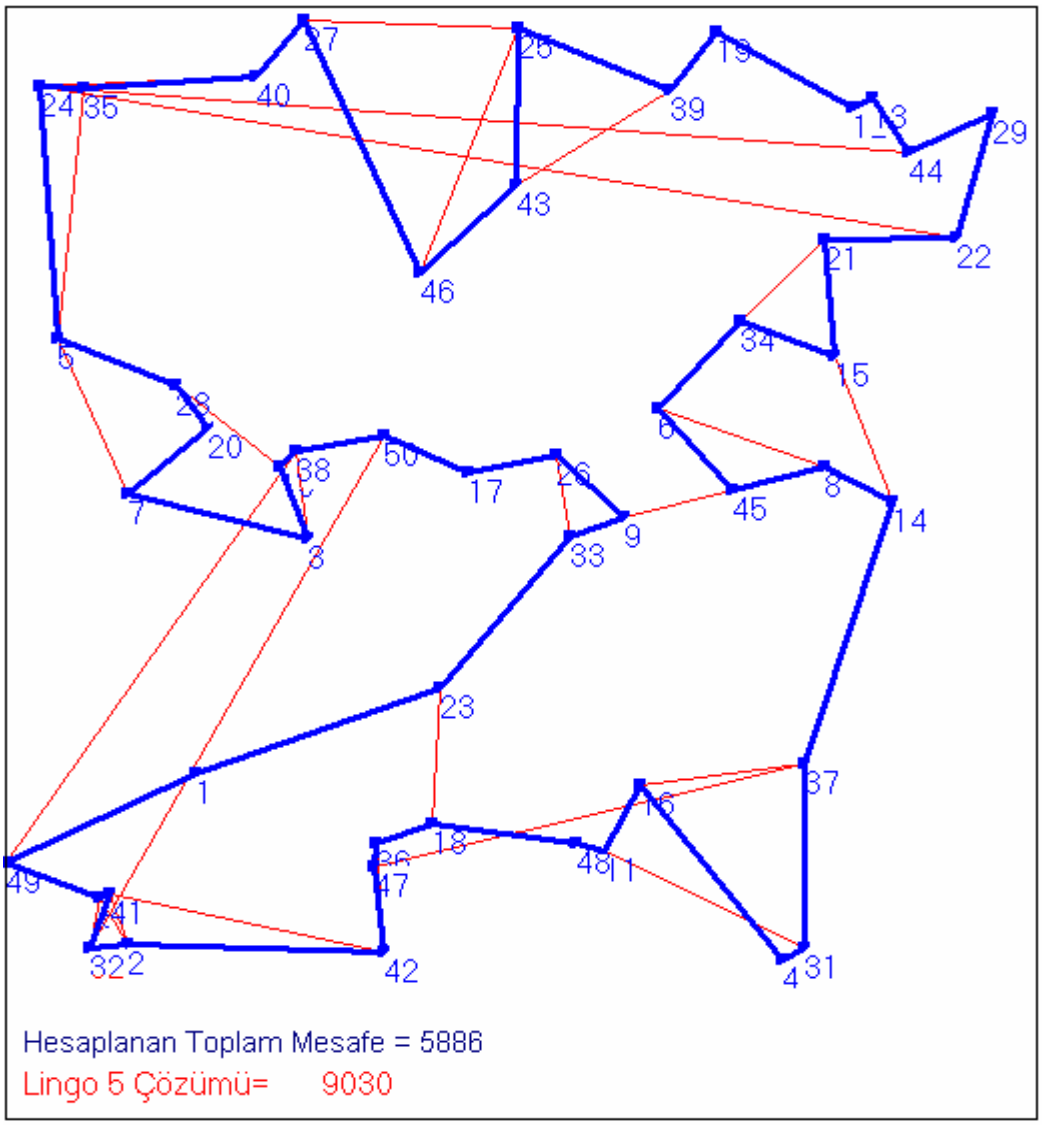


Şekil 7.18. Lingo 5.0 çözümü ile 50 şehir için elde edilen rota

50 şehirli problem için bulunan en iyi mesafeye sahip olan rassal dizi 10'un her jenerasyonda bulmuş olduğu mesafe Şekil 7.19' da verilmiştir.



Şekil 7.19. Rassal dizi 10 ile 50 şehirli problem için bulunan mesafeler



Şekil 7.20. 50 şehirli problem için bulunan turlar

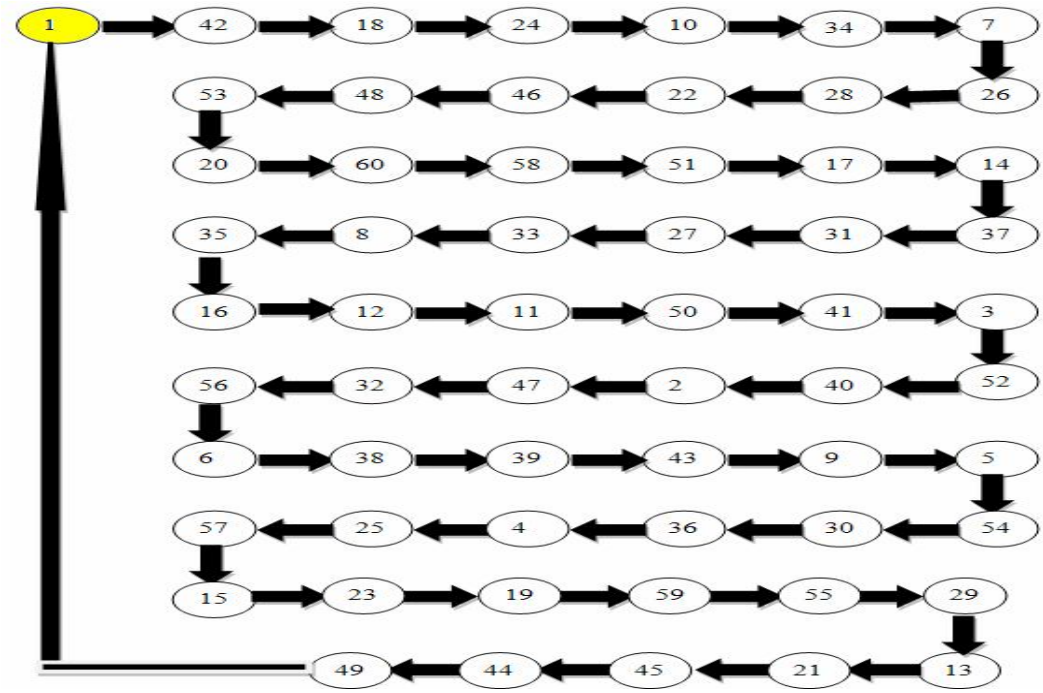
7.2.6. 60 Şehirli Problem

Önceki problemlerde olduğu gibi 60 şehirli problem için, Lingo 5.0 programın 25 saat çalıştırılmış olmasına rağmen en iyi değeri bulunamamıştır. Lingo 5.0 programın çalışma süresinin sonundaki çözüm değeri ile yapmış olduğumuz programın sonuçları karşılaştırıldığında ise yine kısa bir sürede farklı değerler elde edildiği görülmektedir. Programın kısa sürede bulmuş olduğu 6041 sonucuna sahip tur çizdirildiğinde en iyi değere yakın olduğu söylenebilir.

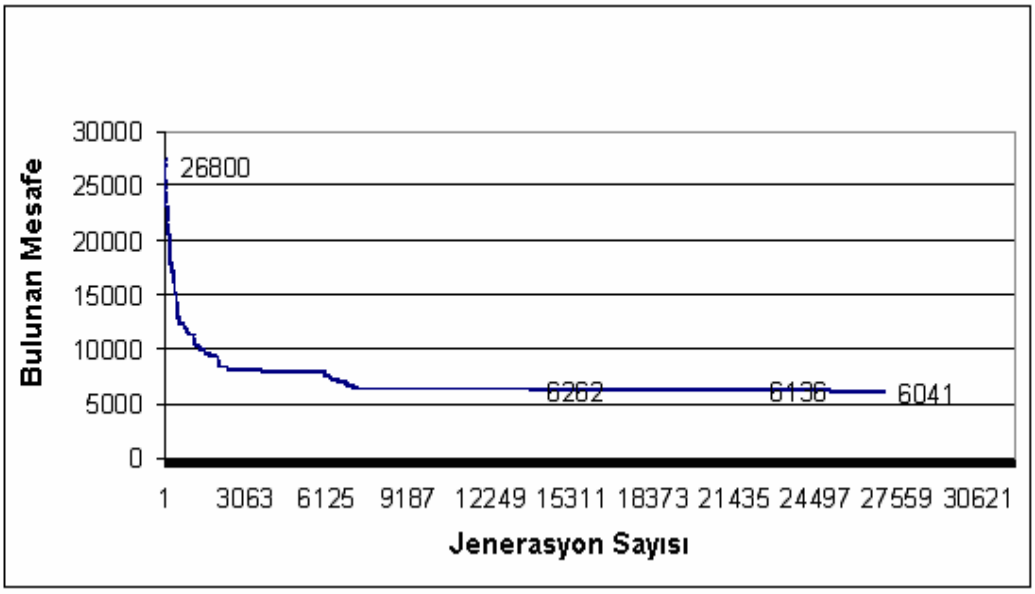
Çizelge 7.6. Hazırlanan programın 60 şehirli problem için sonuçları

ŞEHİR SAYISI=60			**ÇÖZÜM DEĞERİ**	**7004**
			ÇÖZÜM SÜRESİ	**25 saat**
MUTASYON ORANI=%3			**İTERASYON SAYISI**	**11552478**
RASSAL DİZİ	**JENERASYON SAYISI**	**BULUNAN UZAKLIK**	**GEÇEN SÜRE**	**EN İYİ DEĞERDEN SAPMA MİKTARI (%)**
1	46718	6041	46718	-
2	193841	6210	20:36.438	-
3	305269	6440	32:34.220	-
4	480536	6458	50:51.789	-
5	167975	6041	19:08.371	-
6	115951	6082	14:12.395	-
7	105882	6361	12:40.123	-
8	149900	6116	18:51.086	-
9	490059	6455	51:28.391	-
10	247636	6202	26:16.938	-

Elde edilen sonuçların en iyi değerin elde edilmemesi nedeni ile en iyiye yakınsadığı sadece oluşan rotanın geniş bir damla şeklinde olması ile yorumlanabilmektedir. Hazırlanan programın ve Lingo 5.0 programın çıktısı olan turlar aşağıda verilmiştir. Bulunan bu turlara göre Şekil 7.23 çizilmiştir.

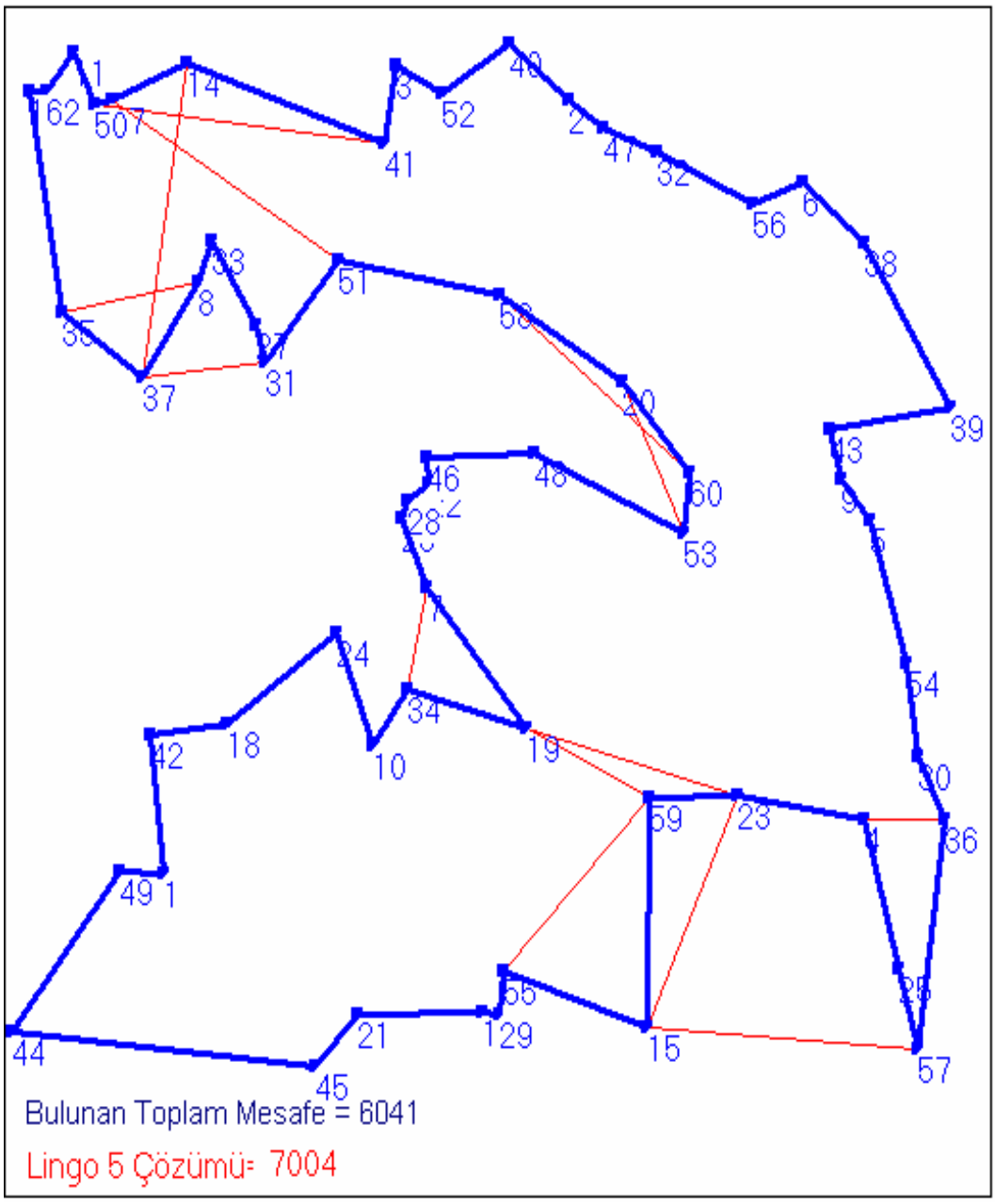


Şekil 7.21. Lingo 5.0 çözümü ile 60 şehir için elde edilen rota



Şekil 7.22. Rassal dizi 1 ile 60 şehirli problem için bulunan mesafeler

60 şehirli problem için bulunan en iyi değere sahip olan rassal dizi 1'in her jenerasyonda bulmuş olduğu sonuçlar Şekil 7.22' de verilmiştir.



Şekil 7.23. 60 şehirli problem için bulunan turlar

7.2.7. 70 Şehirli Problem

70 Şehirli problemin program sonuçlarına bakıldığında ise yeteri kadar iyi olduğu varsayımı yapılabilmektedir. Lingo 5.0 programın uzun süre çalışarak vermiş olduğu sonuçtan kısa bir sürede daha iyi bir sonuç bulunması hazırlanan programın kısa

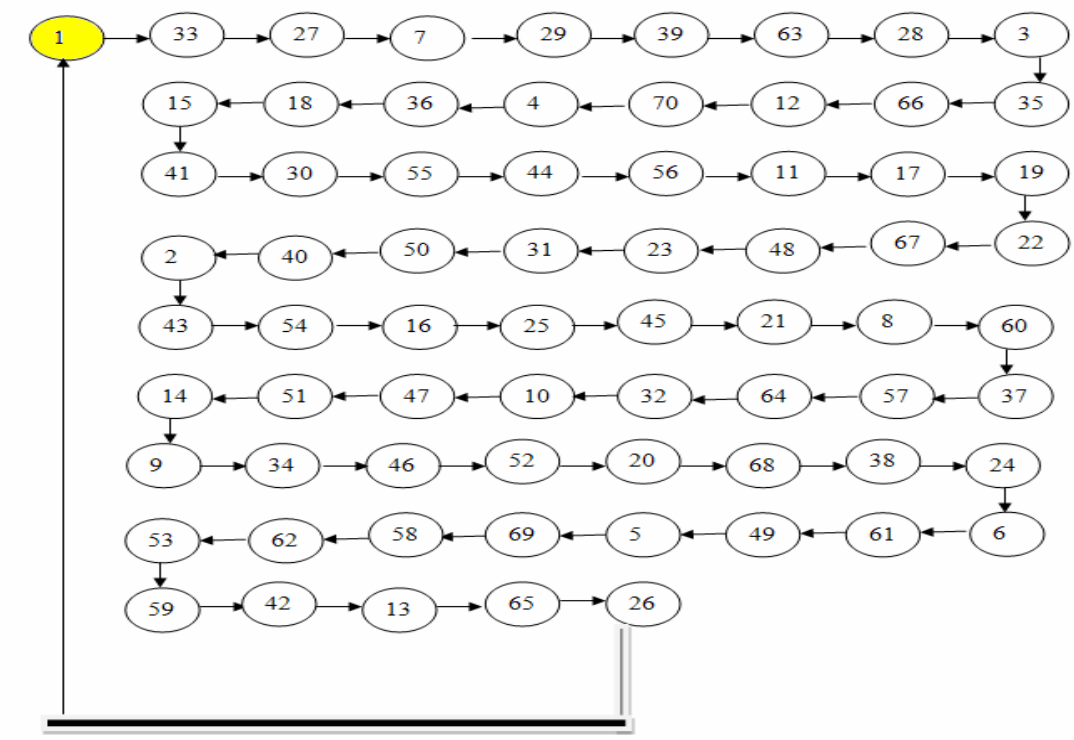
zamanda iyi sonuçlar bulabileceği önceki problemlerin çözüm sürelerine bakılarak da söylenebilmektedir.

Çizelge 7.7. Hazırlanan programın 70 şehirli problem için sonuçları

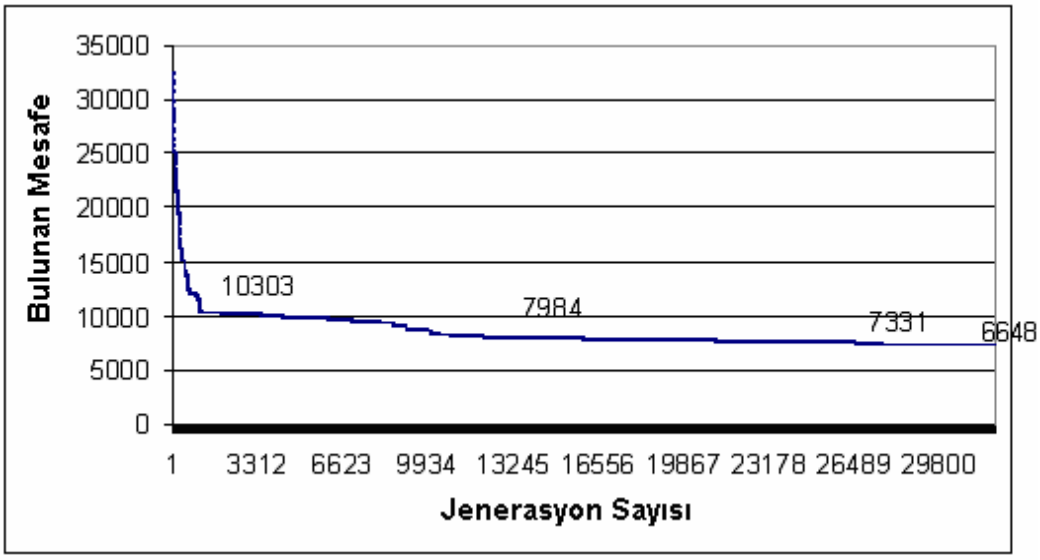
ŞEHİR SAYISI=70			**ÇÖZÜM DEĞERİ**	**11804**
			ÇÖZÜM SÜRESİ	**25 saat**
MUTASYON ORANI=%3			**İTERASYON SAYISI**	**22735350**
RASSAL DİZİ	**JENERASYON SAYISI**	**BULUNAN UZAKLIK**	**GEÇEN SÜRE**	**EN İYİ DEĞERDEN SAPMA MİKTARI (%)**
1	291513	6577	42:50.786	-
2	1240724	6648	03:00:40.979	-
3	199714	6911	2:29.634	-
4	79204	7109	13:37.756	-
5	160951	6539	28:19.744	-
6	246417	6987	43:31.915	-
7	199900	6778	34:15.836	-
8	86746	6670	14:18.365	-
9	115709	6751	19:22.812	-
10	120480	6654	19:51.093	-

70 şehir için çözüm sonuçları değerlendirildiğinde genetik algoritma hesaplama sürelerinin oldukça kısa olması ve 25 saatlik Lingo 5.0 paket programının çalıştırılması ile elde edilen sonuçların oldukça fazla olması göze çarpmaktadır.

70 şehirli problem için hazırlanan program ile bulunan en iyi mesafesine sahip olan rassal dizi 2'nin her jenerasyonda bulmuş olduğu mesafe Şekil 7.25' de verilmiştir.

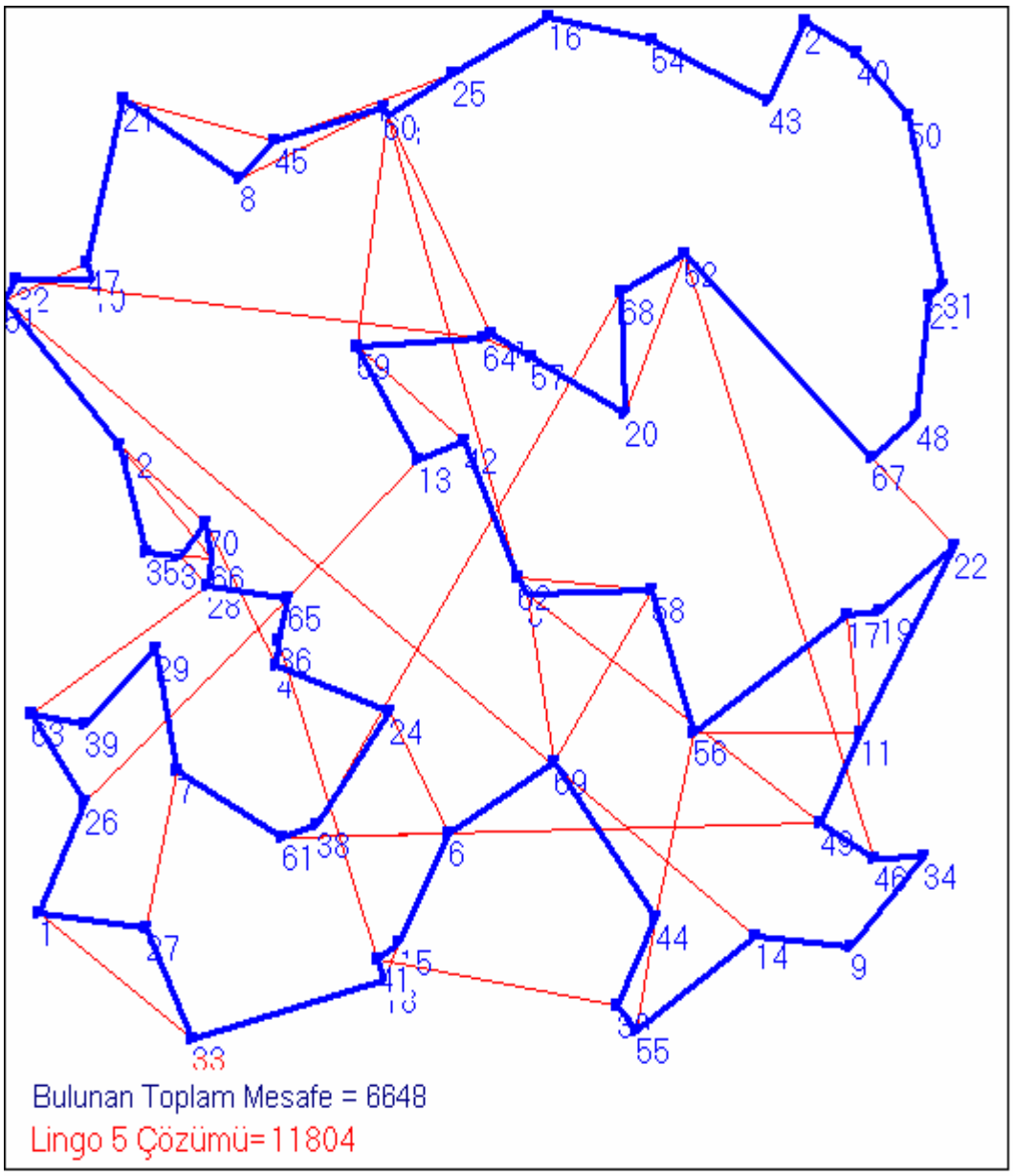


Şekil 7.24. Lingo 5.0 çözümü ile 70 şehir için elde edilen rota



Şekil 7.25. Rassal dizi 2 İle 70 şehirli problem için bulunan mesafeler

Hazırlanan programın ve Lingo 5.0‘ ın 70 şehirli problem için vermiş oldukları turlar aşağıda verilmiştir. Ve bu turlara göre Şekil 7.26 çizilmiştir.



Şekil 7.26. 70 şehirli problem için bulunan turlar

7.2.8. 80 Şehirli Problem

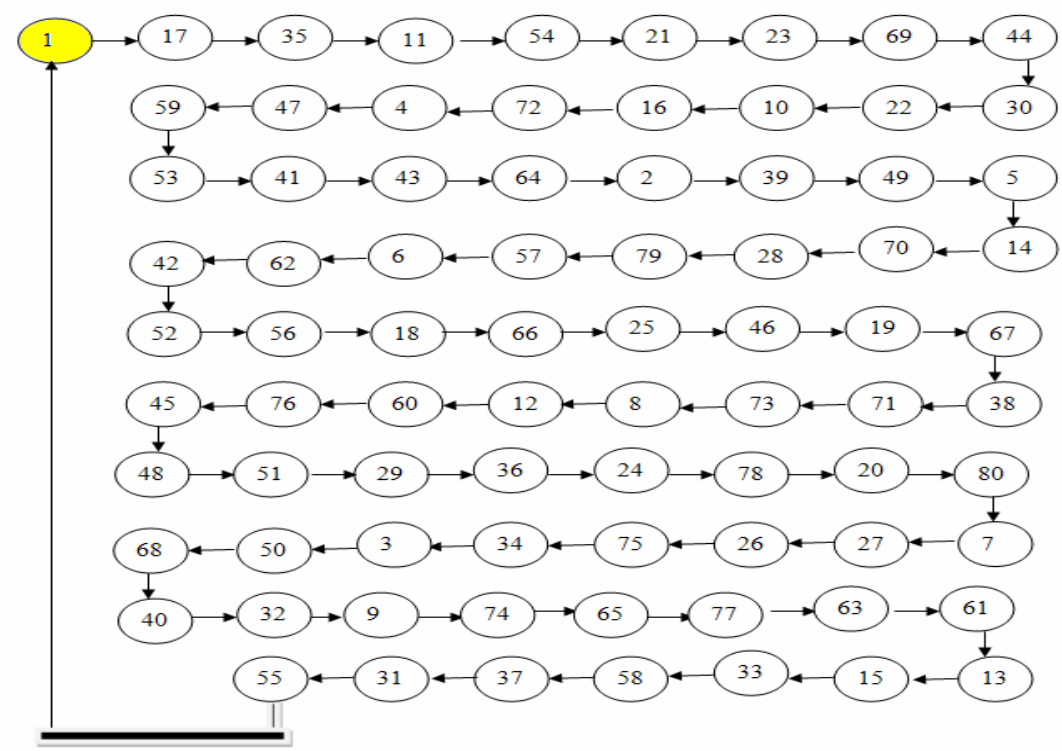
80 şehirli problem için hazırlanan programın sonuçlarına bakıldığında, Lingo 5. ile arasında büyük fark olduğu görülmektedir. Bulunan tur mesafesi 7781'in sadece 30 dk. da bulunması, ve de Lingo 5.0' ın 25 saatte 13786 bulması hazırlanan programın amacına uygun olduğunu açık bir şekilde göstermektedir. Fakat çizilmiş olan tura

bakıldığında bulunan değerin en iyi değer olmadığı açıktır. En iyi değerden sapma miktarı 80 şehir için de değerlendirilememektedir.

Çizelge 7.8. Hazırlanan programın 80 şehirli problem için sonuçları

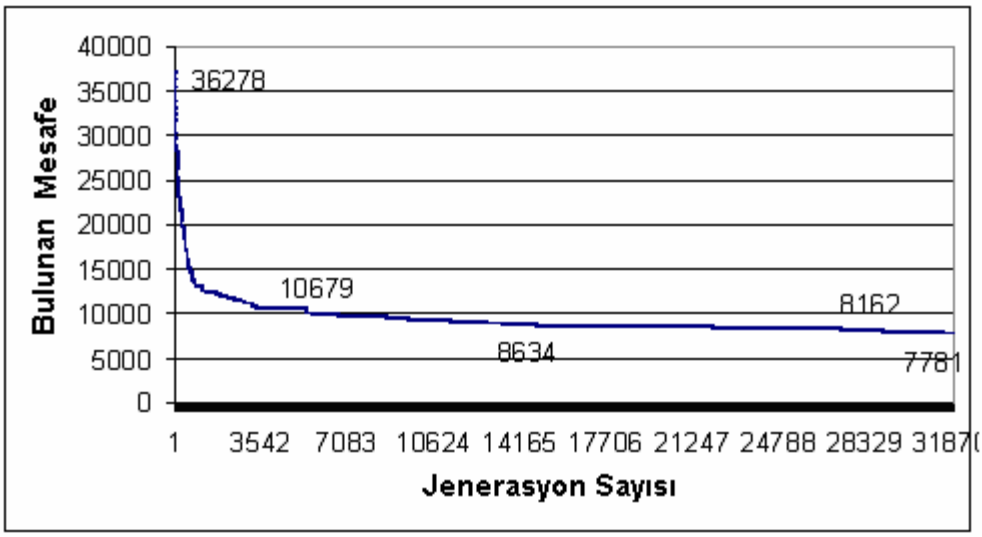
ŞEHİR SAYISI=80			**ÇÖZÜM DEĞERİ**	**13786**
			ÇÖZÜM SÜRESİ	**25 saat**
MUTASYON ORANI=%3			**İTERASYON SAYISI**	**18611494**
RASSAL DİZİ	**JENERASYON SAYISI**	**BULUNAN UZAKLIK**	**GEÇEN SÜRE**	**EN İYİ DEĞERDEN SAPMA MİKTARI (%)**
1	179937	7781	30:10.273	-
2	260054	7907	43:03.395	-
3	320727	8109	53:21.323	-
4	143136	7738	25:17.352	-
5	228128	8099	37:50.866	-
6	180346	7830	29:56.623	-
7	121100	7933	21:28.783	-
8	224365	7986	40:10.366	-
9	923100	8006	01:32:14.575	-
10	173099	8394	31:50.958	-

Hazırlanan programa göre programın ve Lingo 5.0' ın bulmuş olduğu turlar aşağıda verilmiştir. Bu turlar kullanılarak Şekil 7.29 çizilmiştir.

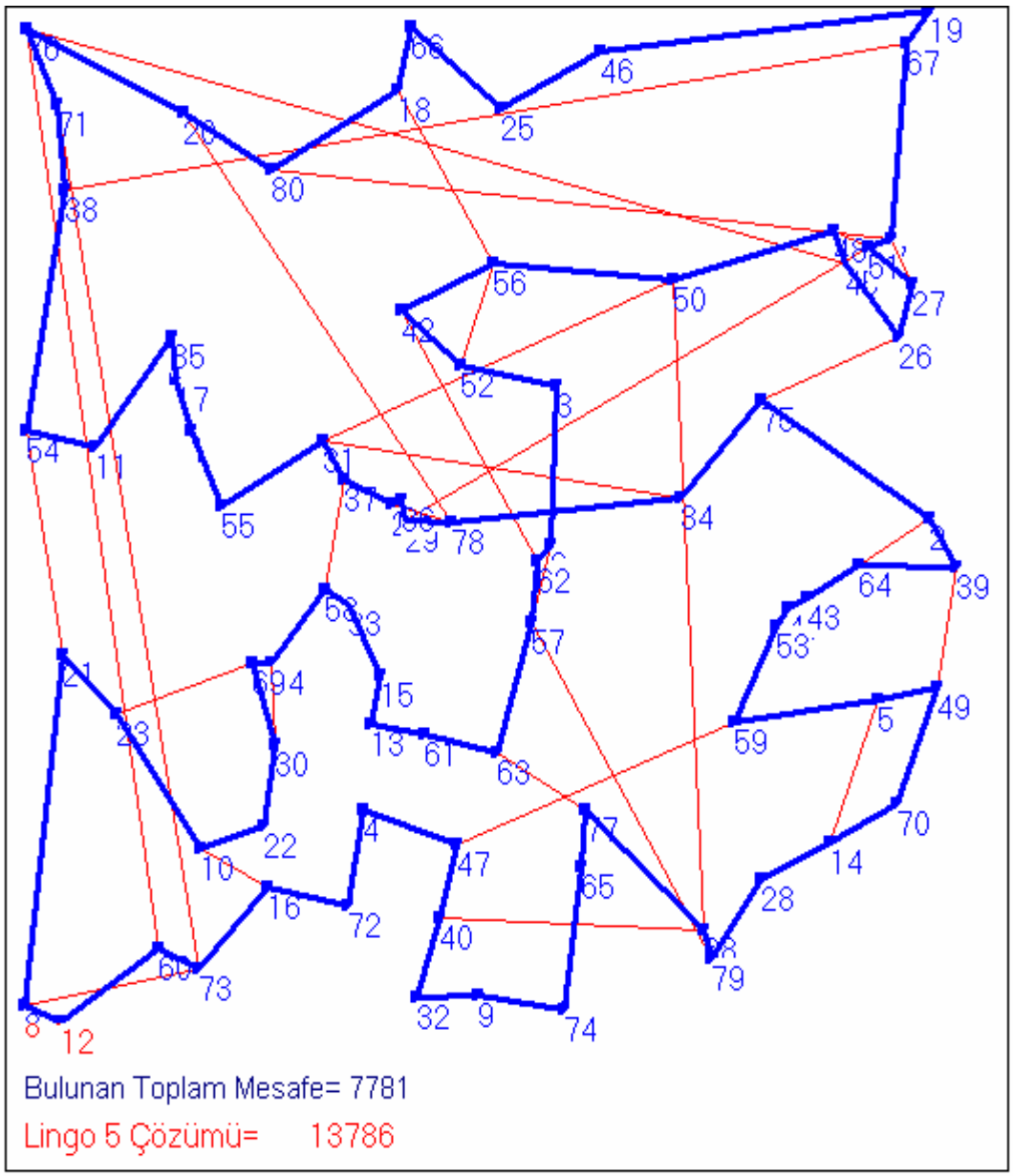


Şekil 7.27. Lingo 5.0 çözümü ile 80 şehir için elde edilen rota

80 şehirli problem için bulunmuş olan en iyi mesafeye sahip olan rassal dizi 1'in her jenerasyonda bulmuş olduğu mesafe Şekil 7.28' de verilmiştir.



Şekil 7.28. Rassal dizi 1 ile 80 şehirli problem için bulunan mesafeler



Şekil 7.29. 80 şehirli problem için bulunan turlar

7.2.9. 90 Şehirli Problem

90 şehirli problemin hazırlanan program ve Lingo 5.0 sonuçlarına bakıldığında yine büyük bir fark olduğu görülmektedir. Ancak hazırlanan programın vermiş olduğu sonuç Şekil 7.32' ye bakıldığında en iyi değer olmadığı açıktır. Ancak amacımız olan

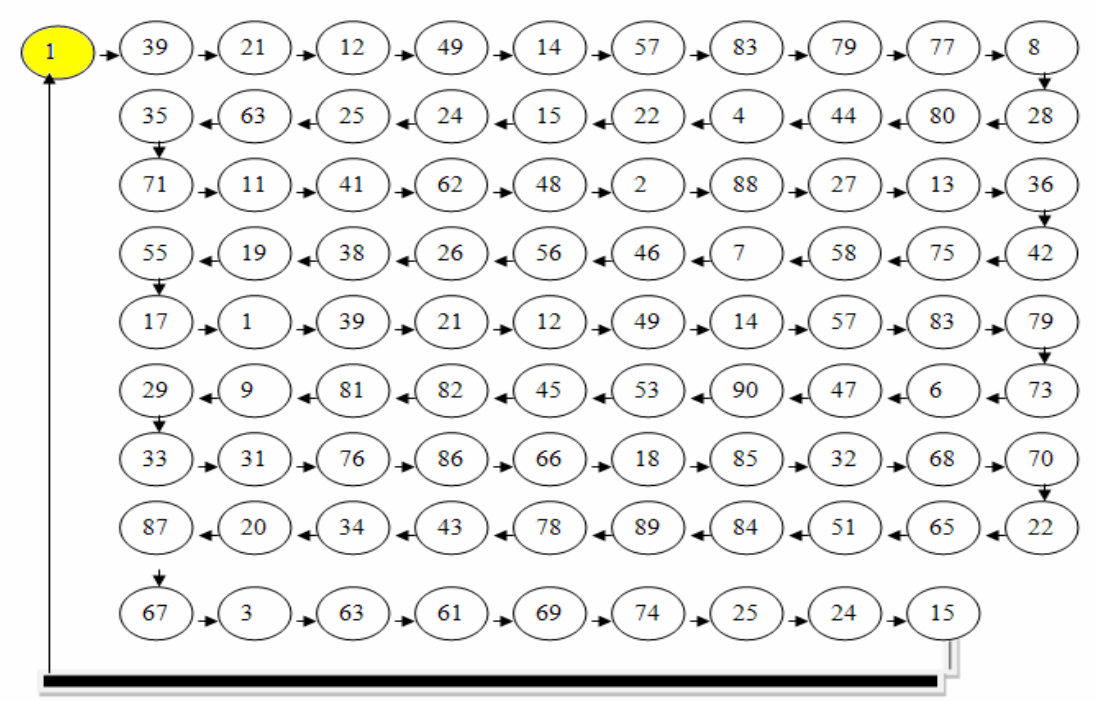
çok zaman da en iyi yerine az zaman da iyi bir sonuç bulmak olduğu için sonucun istenen netice elde edilmiş denebilir.

Çizelge 7.9. Hazırlanan programın 90 şehirli problem için sonuçları

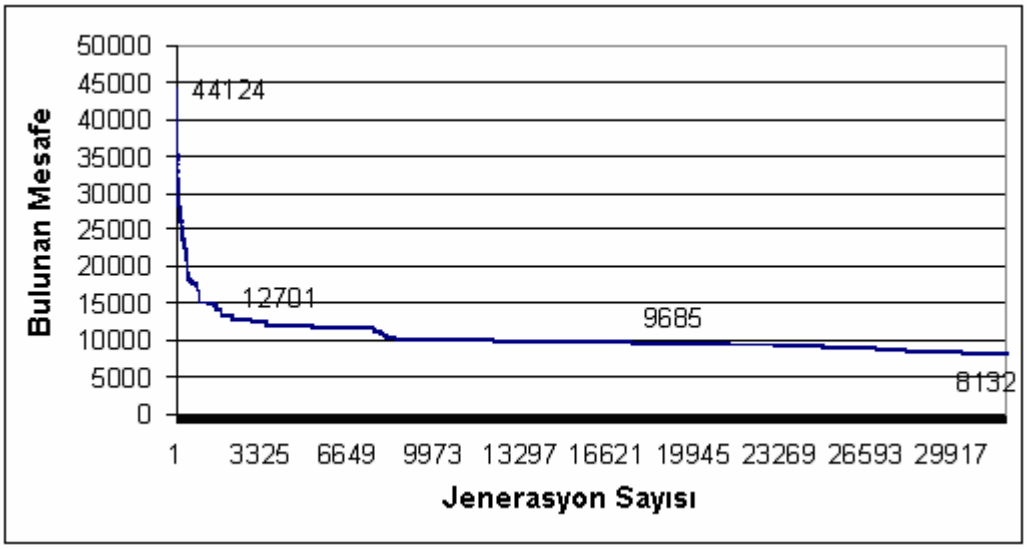
ŞEHİR SAYISI=90			**ÇÖZÜM DEĞERİ**	**12863**
MUTASYON ORANI=%3			**ÇÖZÜM SÜRESİ**	**25 saat**
			İTERASYON SAYISI	**11552478**
RASSAL DİZİ	**JENERASYON SAYISI**	**BULUNAN UZAKLIK**	**GEÇEN SÜRE**	**EN İYİ DEĞERDEN SAPMA MİKTARI (%)**
1	176404	8248	37:08.424	-
2	177347	8221	37:17.838	-
3	104764	8343	22:30.962	-
4	329104	8594	01:05:34.438	-
5	175916	8400	37:50.014	-
6	1035233	8329	02:24:34.920	-
7	270754	8132	57:00.088	-
8	222919	8930	49:06.056	-
9	923100	8006	01:32:14.575	-
10	499992	8292	01:44:50.134	-

En iyi dğerden sapma miktarı 90 şehir için elimzde en iyi değerin olması nedeni ile belirlenememiştir.

90 şehirli problem için bulunan en iyi mesafeye sahip olan rassal dizi 7’ nin her jenerasyonda bulmuş olduğu mesafe Şekil 7.31’ de verilmiştir.

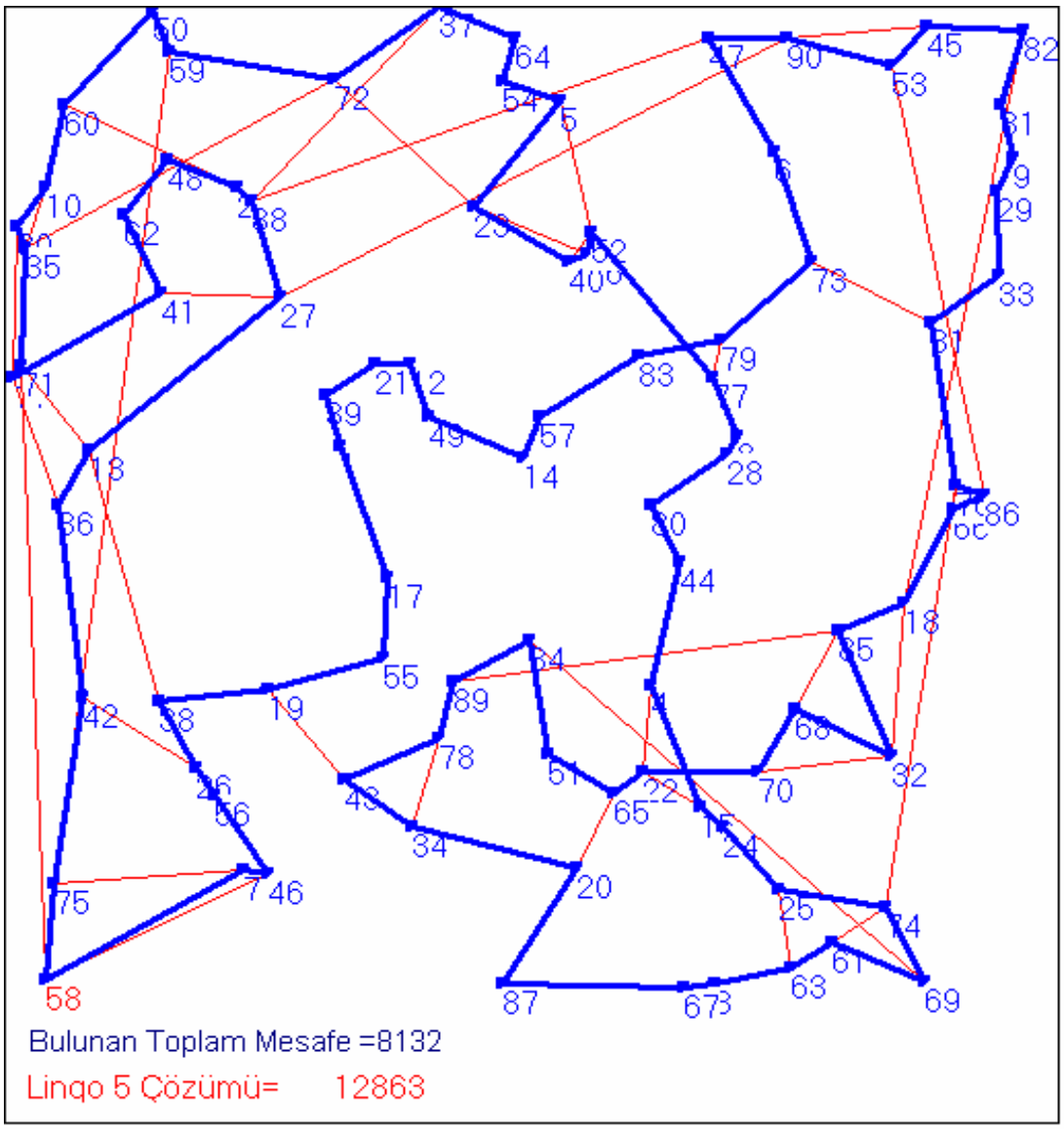


Şekil 7.30. Lingo 5.0 çözümü ile 90 şehir için elde edilen rota



Şekil 7.31. Rassal dizi 7 İle 90 şehirli problem için bulunan mesafeler

Hazırlanmış olan program ile 90 şehirli problem için hazırlanan program ve Lingo 5.0' ın bulmuş olduğu turlar aşağıda verilmiştir. Bu turlar kullanılarak Şekil 7.32 çizilmiştir.



Şekil 7.32. 90 şehirli problem için bulunan turlar

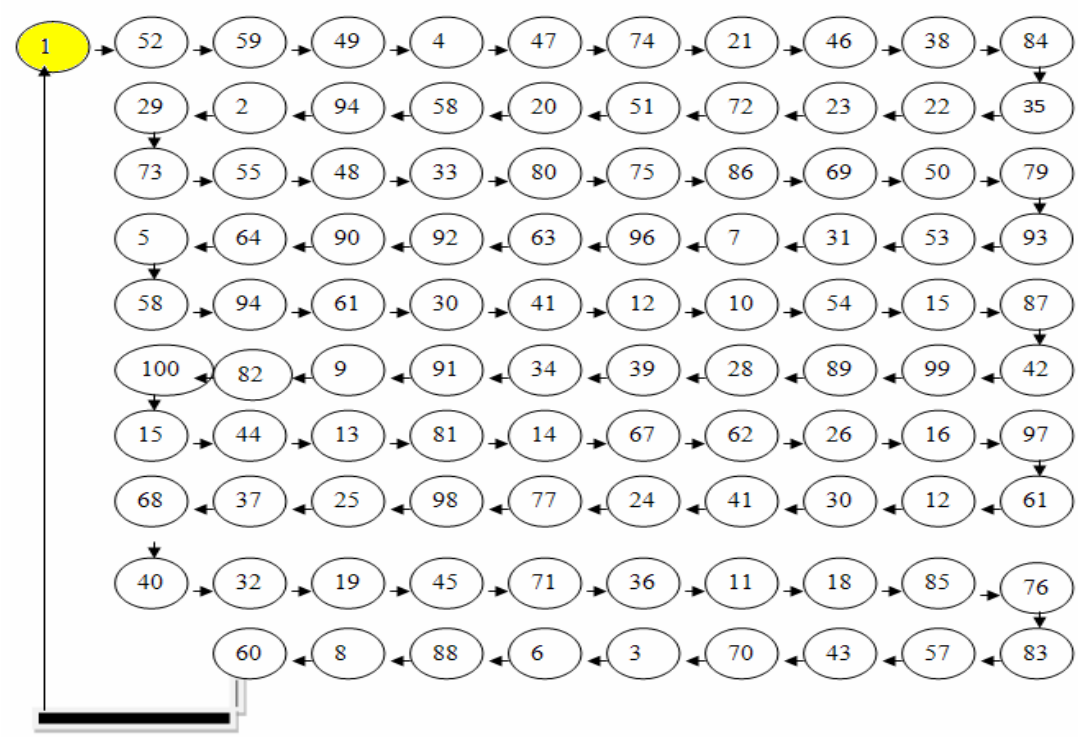
7.2.10. 100 Şehirli Problem

100 şehirli problem sonuçları değerlendirildiğinde ise sonucun 80, 90 şehirlik problemler için bulunan sonuçlara göre daha iyi bir olduğu görülmüştür. Şekil 7.35' e bakıldığında bulunan turun yeterince iyi olduğu görülmektedir. Ancak yine elimizde en iyi değer olmadığından bulunan mesafenin varsayımlar ile değerlendirilmesi söz konusudur.

Çizelge 7.10. Hazırlanan programın 100 şehirli problem için sonuçları

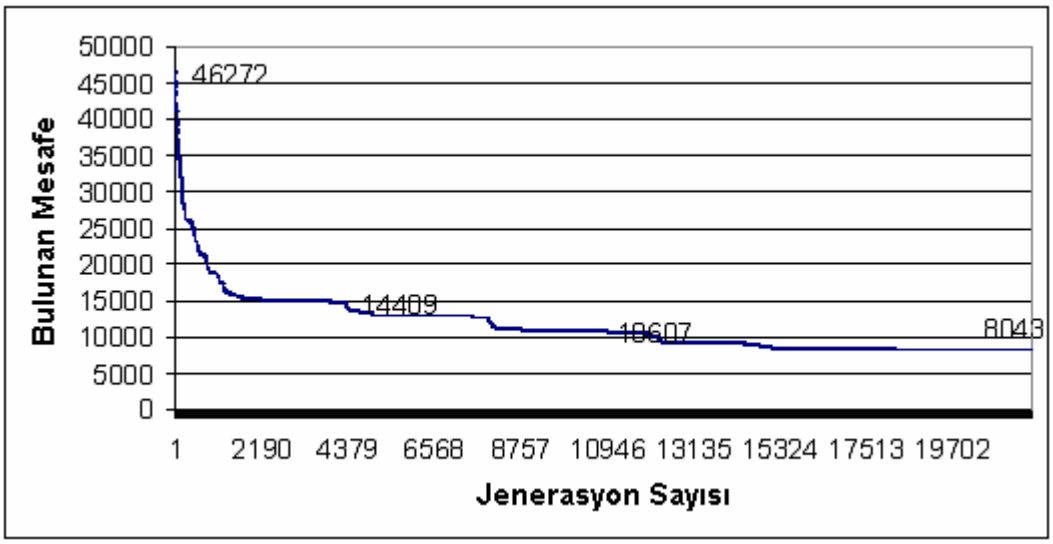
ŞEHİR SAYISI=100			**ÇÖZÜM DEĞERİ**	**13001**
			ÇÖZÜM SÜRESİ	**25 saat**
MUTASYON ORANI=%3			**İTERASYON SAYISI**	**1162297**
RASSAL DİZİ	**JENERASYON SAYISI**	**BULUNAN MESAFE**	**GEÇEN SÜRE**	**EN İYİ DEĞERDEN SAPMA MİKTARI (%)**
1	811322	8317	03:25:49.327	-
2	1181385	8561	03:22:30.962	-
3	117115	8752	29:05.210	-
4	293480	8205	01:13:49.449	-
5	134493	8770	33:51.351	-
6	269141	8961	01:07:21.331	-
7	569420	8043	02:21:26.543	-
8	167211	8625	39:05.210	-
9	197056	8504	43:51.351	-
10	252325	8341	01:03:10.900	-

100 şehirli problem için hazırlanmış olan program ve Lingo 5.0' ın buldukları turlar aşağıda verilmiştir. Bu turlar kullanılarak Şekil 7.35 çizilmiştir.

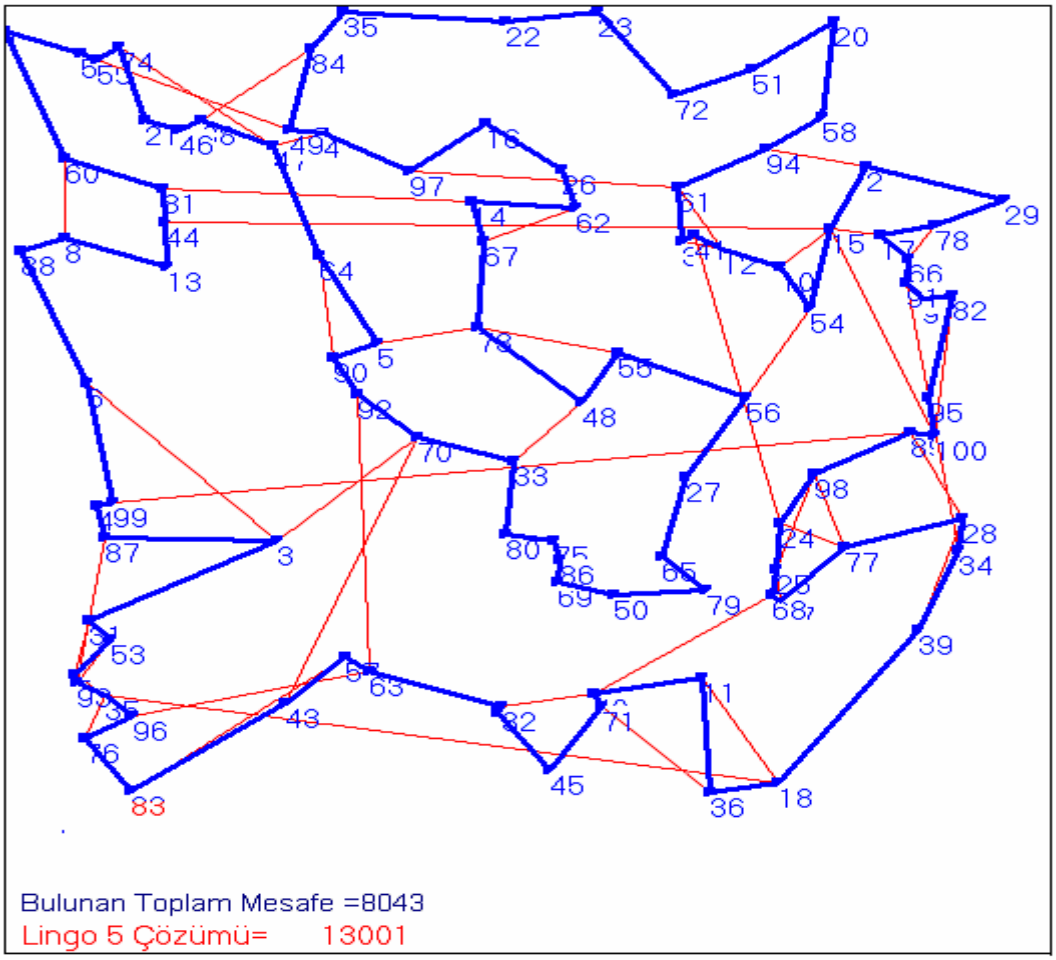


Şekil 7.33. Lingo 5.0 çözümü ile 100 şehir için elde edilen rota

100 şehirli problem için bulunan en iyi mesafeye sahip olan rassal dizi 7' nin her jenerasyonda bulmuş olduğu mesafe Şekil 7.34' de verilmiştir.



Şekil 7.34. Rassal dizi 7 İle 100 şehirli problem için bulunan mesafeler



Şekil 7.35. 100 şehirli problem için bulunan turlar

8. SONUÇ VE ÖNERİLER

Genetik algoritmaların sezgisel bir teknik olarak, yöneylem araştırması problemlerinde kullanımı giderek artmaktadır. GA yaklaşımlarının kullanılması ile yöneylem araştırması alanında ortaya çıkan en iyileme problemlerinin çözümüne ilişkin alternatif yöntemler ortaya atılmıştır ve bu yeni yöntemlerin baz problemlerin çözümünde kullanılması , uygulamaya açık olan diğer problemlere de farklı bir boyut kazandırmaktadır.

Genetik algoritma çalışmaya başladıktan bir süre sonra en anlamlı basamaklarda bir değer baskın gelir. Bir basamakta bir değerin baskın gelmesi ilgili noktanın genetik algoritmanın araştırma işleminde yer almaması anlamına gelir. Örneğin, toplumun %95'i çözümün bir kromozomunda 5 taşıyorsa bu pozisyonda bulunan değer çözümler arasında farklılık yaratmaz;bu noktanın değeri hemen hemen hepsinde aynıdır. Sonuç olarak, o noktada ki değerin araştırması bitmiş sayılabilir. Genetik algoritma bu aşamadan sonra o nokta hiç yokmuş gibi diğer noktalarla işleme devam eder. Tüm pozisyonlar için bir değer baskın hale geldiğinde arama bitmiştir. Bu aşamadan sonra genetik algoritmanın daha başarılı bir değer bulması daha zordur. Benzerlikler fazla olduğu için arama uzayının değişik bölgelerine ulaşılamaz. Toplum bu duruma ulaştığında en genel olan çözüm en iyi çözüm olmak zorunda değildir.

Çalışmanın amacı olan, gezgin satıcı probleminin en iyi sonucunun bulunmasından ziyade az zamanda en iyi ye yakın bir sonuç bulunmuştur. Çözümlerin her rassal dizi ile farklı bulunması, problem çözümünde programın vereceği toplam alınmış olan mesafenin başlangıç popülasyonuna ne kadar bağımlı olduğunu göstermektedir. Ayrıca çözüm sürecinde genetik algoritmanın yerel maksimumlara takılarak uzun süre çalışmasına rağmen sonucun değişmediği görülmüştür.

Lingo 5.0 paket programı ile çözümün en iyi olduğu garanti edilmekte fakat uzun süre çalışması gerekmektedir. Yapılan çalışmada ele alınan sonuçlara bakıldığında 10 şehir için en iyi değer Lingo 5.0 ile 7 sn'de 2214 olarak elde edilirken genetik

algoritma ile kodlanan programda 0.06 sn'de aynı sonuca ulaşılmaktadır. Şehir sayısı arttıkça çözüm sonuçları da benzer şekilde farklılaşmaktadır. 30 şehire kadar Lingo 5.0 paketi ile en iyi çözüm sonuçları elde edilmesine rağmen , 30 şehirden sonrasında Lingo 25 saat çalıştırılmasına rağmen en iyi sonuçlar elde edilememektedir. 80 şehir için çözüm sonuçları değerlendirilecek olursa Lingo 25 saat çalıştırılarak 13786 mesafe bulurken, genetik algoritma 7781 mesafaye sadece 30 dakika 10 saniyede ulaşmaktadır. Burada genetik algoritmanın nokta sayısı arttıkça oldukça kısa sürede daha iyi sonuçlar elde ettiği açıkça görülmektedir.

Ayrıca çözüm sürecinde genetik algoritmanın yerel maksimumlara takılarak uzun süre çalışmasına rağmen sonucun değişmediği de görülmüştür.

Elde edilen sonuçlara da bakılarak meta-sezgisel bir yöntem olan genetik algoritma ile kodlanan program çözüm sonuçlarının etkinliği açıkça görülmektedir. Aynı zamanda çalışma kapsamında ilgili programın işletmelerin ihtiyaçlarına ve değişkenlerine göre genişletilebileceği üzerinde de durulmaktadır.

KAYNAKLAR

1.Potvin, J., Y., "Genetic algorithms for the travelling salesman problem", forthcoming in Annals of Operations Research on "Metaheuristics in Combinatorial Optimization", eds. ***G. ,Laporte and I. H. Osman,*** 63: 339-370 (1996).

2.Carter, E.A., Ragsdale, C., T., "A new approach to solving the multiple travelling salesperson problem using genetic algorithms", ***European Journal of Operational Research,*** 175: 246–257 (2006).

3.Little, J., Murty, D. And Sweeney, D., Karel, C.,. "An algorithm for the travelling salesman problem", ***Operations Research,*** 11 (6): 972–989 (1963).

4.Bellmore, M., Malone, J., "Pathology of travelling-salesman subtour elimination algorithms", ***Operations Research ,***19 (2): 278–307 (1971).

5.Shirrish, B., Nigel, J., Kabuka, M.R.,. "A Boolean neural network approach for the travelling salesman problem",.***IEEE Transactions on Computers,*** 42 (10): 1271–1278 (1993).

6.Glover, F., "Artificial intelligence, heuristic frameworks and tabu search", ***Managerial and Decision Economics ,***11 (5): 365–378 (1990).

7.Goldberg, D.E., Lingle, R.J., "Alleles, loci, and the traveling salesman problem", In J. J. Grefenstette (Ed.), Proceedings of the First International Conference on Genetic Algorithms***, London*** , 154-159 (1985)

8.Applegate, D. L., Bixby, R. E., Chvátal, V., Cook, W. J., "The travelling salesman problem", A Computational Study***, Princeton University Press***, ISBN 978-0-691-12993-8 (2006).

9.İnternet: Christian Nilsson, "Heuristic Algorithms For Travelling Salesman Problem", LinköpingUniversity:http:/www.ida.liu.se/~TDDB19/report2003/htsp.pdf (2003).

10.Johnson, D., S., "Local optimization and the travelling salesman problem", In Proceedings of the 17th International Colloquium on Automata, Languages and Programming (July 16-20, 1990). M. Paterson, Ed. Lecture Notes In Computer Science,. ***Springer-Verlag, London***, 443: 446-461 (1990).

11.Uğur, A., "Path planning on a cuboid using genetic algorithms", ***Inf. Sci,***. 178: 3275-3287 (2008).

12.Mitchell, M., Forest, S., "Genetic algorithms and artificial life", Reprinted in C. G. Langton (Ed.) Artificial Life: an Overview***, MIT Press, Cambridge, MA,*** 1(3): 267-289 (1995).

13.Kömür, M., Altan, M. , "Betonarme bir kiri ve kolonun genetik algoritma ile optimum tasarım" , ***Mühendislikte Modern Yöntemler Sempozyum Kitabı,*** 127-133 (2001) **.**

14.Gass, S., "Linear programing, methods and applications", ***Mc Graw Hill***, ***New York*** , 356-357 (1964).

15.Leontief, W., "The structure of American economy", ***Oxford University,*** 119 *(1963)*.

16.Taha, Hamdi A. , Operations research":An Introduction, ***Prentice-Hall, ABD***, 5: 16 (1992).

17.Altaylı, B., "Yöneylem araştırması", ***Ankara,*** 66-68 (1996)**.**

18.Acar, A. S., "Linear programing for managerial decisions", ***China***, 7-15 (1987).

19.Ballaou, R., H., "Business logistics management", ***New York,*** 104-107 ***(***1992).

20.Gilbert L. G., Gill I. S., "The traveling salesman problem***", International Journal of Retail&Distribution Management***, 30 (6): 59,231-237 (2002).

21.Ropke, S., Pisinger, D., "A unified heuristic for a large class of vehicle routing problems with backhauls". ***European Journal of Operational Research,*** 171: 750-775 (2004).

22.Baker, E., Bodin, L., Finnegan, W., "Efficient heuristic solutions to an airline crew sheduling problem", ***AIIE Transactions*** ,11: 79-85 (1979).

23.Bellman, R., "On a routing problem", ***in Quarterly of Applied Mathematics***, 16(1): 87-90 (1958).

24.Golden, B. , "Operations research", May – June, 28(3): 694-711 (1980) .

25.Cruyssen, V. D. P., Rijkaert. M., "Heuristic for the asymetric travelling salesman problem", ***The Journal of the Operational Research Society***, 29(7):697–701, (1978).

26.Ahuja, R. K., Magnanti, T. L., Orlin, J. B., "Network Flows: Theory, algorithms and applications", ***Prentice Hall: New Jersey,*** 37:211-276 (1993)**.**

27.Eiselt, H. A., , Gendreau, M., Laporte, G., "Arc routing problems, Part 1:The Chinese Postman Problem", ***Operations Research,*** 43(2): 231–242 (1995).

28.Eglese, R. W., Li, Y. O, L., "Efficient routing for winter gritting", ***The Journal Of The Operational Research Society,*** 43(11): 1031-1034 (1992).

29.Ong, H. L., Huang, H. C., "Asymptotic expected performance of some TSP heuristics", ***European Journal of Operational Research***, 43: 231–238 (1989).

30.Horowitz, E., , Sahnı, S., "Fundementals of computers algorithms", ***California: Computer Science Press*** , 626 (1978).

31.Winston, W. L., "Operations research applications", ***Boston, MA ,*** 9(2):195-207 (1991).

32.Emel, G.,Taşkın, Ç., "Genetik algoritmalar ve uygulama alanları", ***Uludağ Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*** , XXI (1): 129-152 (2002).

33.Engin, O., Fığlalı, A., "Genetik algoritmalarla akış tipi çizelgelemede üreme yöntemi optimizasyonu"***, İTÜ Dergisi***, 3-5, (2002).

34.Goldberg, D. E., , "Genetic algorithm in search, optimization & machine learning , Mass", ***Addison Wesley***, Reading, 95-102, (1989).

35.Koç, İ. O., "Regresyon modellerinin parametrelerinin genetik algoritma kullanarak tahmin edilmesi"***, Ankara,*** 66-69 (2001).

36.Oğuz, M., ve Akbaş, S., "Sanayi problemi ve genetik algoritma ile çözümlenmesi", ***Mersin Üniversitesi Tıp Fakültesi Dergisi,*** 244-247 (2001)

37.Toth, P., Vigo, D., "Models, relaxations and exact approaches for the capacitated vehicle routing problem", ***Discrete Applied Mathematics***, 123(1-3): 487-512 (2002).

38.Crainic, G. T., Laporte, G,. "Fleet management and logistics", ***Kluwer, USA,*** 159-164 (1998)***.***

39.Jang, J. S. R., "Neuro-Fuzzy and soft computing", A Computational Approach to Learning and Machine Intelligence, ***Prentice-Hall, USA,*** 369-400 (1997).

40.Baker, B. M., Ayechew M.,A., "A genetic algorithm for the vehicle routing problem", ***Computers&Operations Research,*** 30 (2): 787-800 (2003).

41.Prins, C., " A simple and effective evolutionary algorithm for the vehicle routing problem", ***Computers&Operations Research***, Article in Press, 34(6): 1561-1584 (2003).

42. Chiung, M., Jongsoo, Gyunghyun, K., Choi, Yoonho, S., " An efficient genetic algorithm for the travelling salesman problem with precedence constraints" , ***European Journal of Operational Research ,***32: 741-747 (2001)***.***

43.Tsai, C. F., Tseng C. C., "A new hybrid heuristic approach for solving large traveling salesman problem"***, Information Sciences,*** 166 (1–4): 67–81 (2004).

44.Johnson, D. S., McGeoch, L. A., “The traveling salesman problem: a case study in local optimization”, in: E. H. L. Aarts, J. K. Lenstra (Eds.), Local Search in Combinatorial Optimization, ***John Wiley & Sons***, New York, 215–310 (1997).

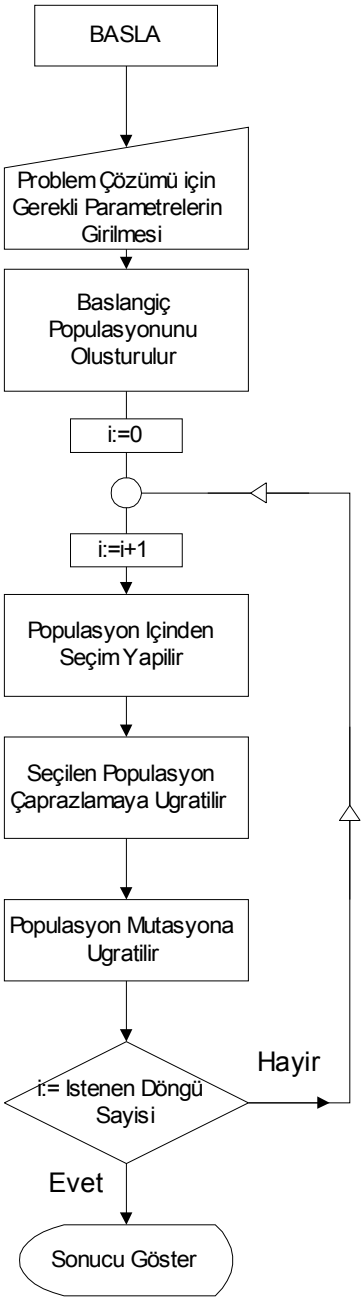
45.Davis, L., “Job shop scheduling with genetic algorithms”, In: Proceedings of the International Conference on Genetic Algorithms, ***London***, 136–140 (1985).

46.Starkweather, T., Whitley, D., Whitley, C., Mathial, K.,.”A comparison of genetic sequencing operators”, In: Proceedings of the Fourth International Conference on Genetic Algorithms***, Los Altos, CA,*** 69–76 (1991).

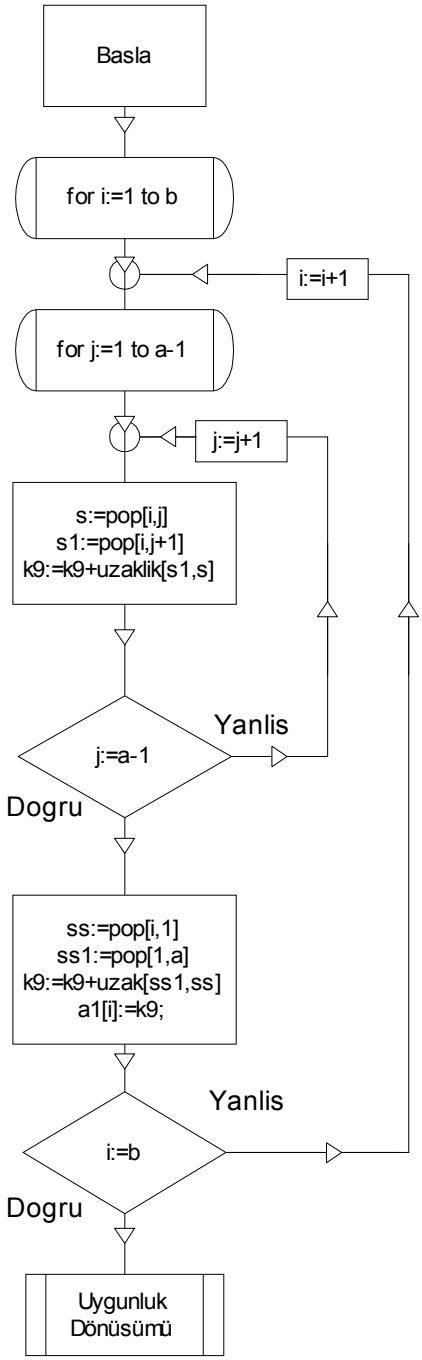
47.Oliver, I., Smith, D., Holland, J.,. “A study of permutation crossover operators on the traveling salesman problem”, In: Proceedings of the Second International Conference on Genetic Algorithms***, London***, 224–230 (1987).

EKLER

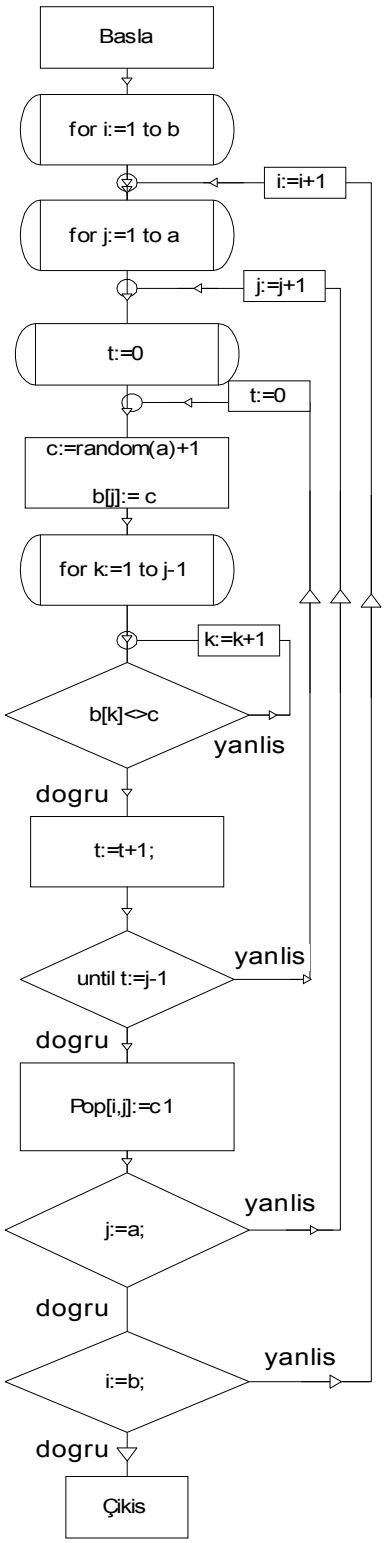
EK-1. Genetik algoritma genel akış şeması



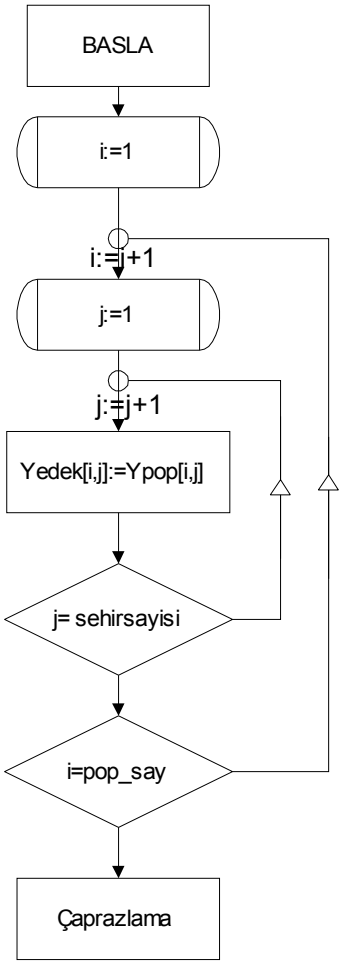
EK-2. Başlangıç tur oluşturma akış şeması



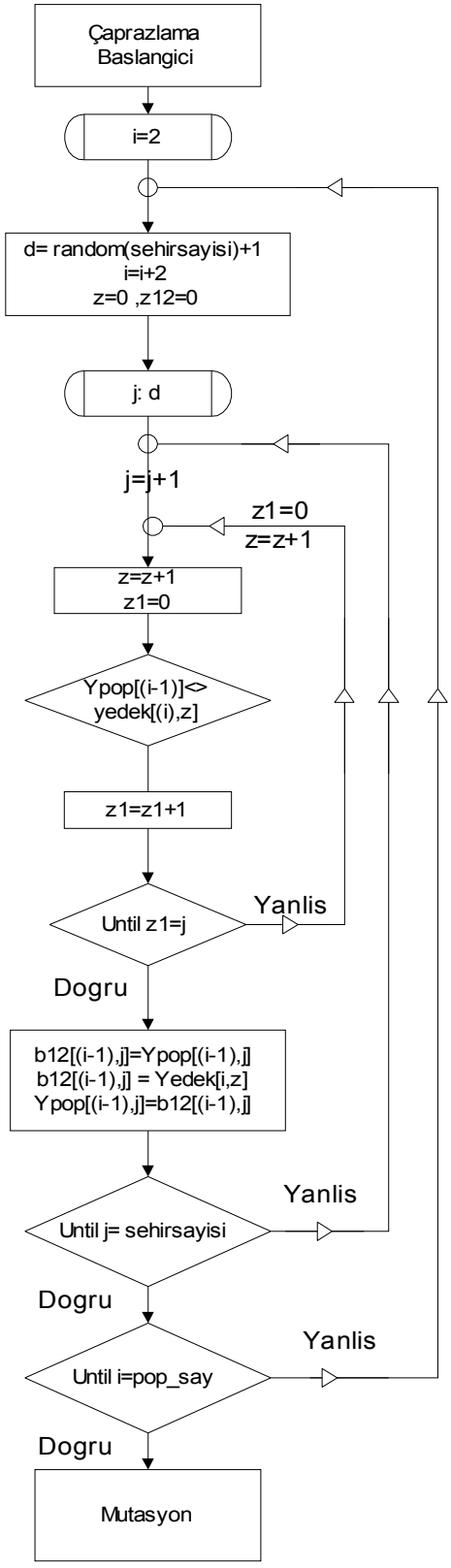
EK-3. Tur uygunluk değeri hesaplama akış şeması



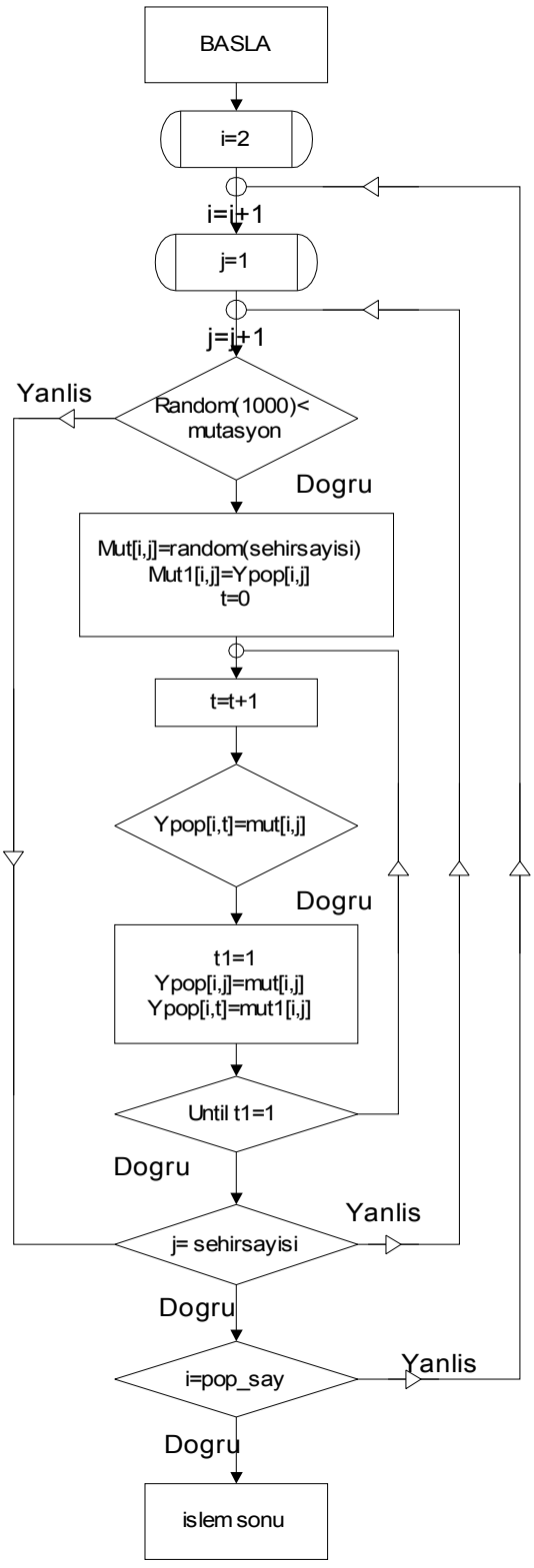
EK-4. Çarprazlama akış şeması



EK-4. Çarprazlama akış şeması(Devam)



EK-5. Mutasyon işlemi akış şeması



EK-6. Uzaklık matrisi

*	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	**0**	4	5	10	8	7	12	15
2	4	**0**	8	9	14	5	7	3
3	5	8	**0**	6	4	10	13	7
4	10	9	6	**0**	10	9	5	7
5	8	14	4	10	**0**	6	4	10
6	7	5	10	9	6	**0**	3	11
7	12	7	13	5	4	3	**0**	16
8	15	3	7	7	10	11	16	**0**

EK-7. 10 şehirli problem koordinatları

10 Şehirli Problem		
i. Şehir	**X**	**Y**
1	950	566
2	180	715
3	888	405
4	304	841
5	434	583
6	673	365
7	770	450
8	29	642
9	668	644
10	195	743

EK-8. 20 şehirli problem koordinatları

20 Şehirli Problem					
i. Şehir	**X**	**Y**	**i. Şehir**	**X**	**Y**
1	605	483	**11**	749	243
2	538	106	**12**	40	628
3	567	40	**13**	737	53
4	681	849	**14**	462	772
5	511	346	**15**	470	69
6	389	364	**16**	659	150
7	520	270	**17**	390	100
8	936	486	**18**	419	168
9	795	701	**19**	573	524
10	998	465	**20**	341	140

EK-9. 30 şehirli problem koordinatları

30 Şehirli Problem								
i. Şehir	**X**	**Y**	**i. Şehir**	**X**	**Y**	**i. Şehir**	**X**	**Y**
1	656	321	**11**	543	827	**21**	1	61
2	596	534	**12**	99	51	**22**	65	242
3	435	132	**13**	633	176	**23**	304	770
4	513	782	**14**	9	703	**24**	394	808
5	708	222	**15**	886	814	**25**	742	974
6	432	969	**16**	52	140	**26**	363	782
7	750	808	**17**	858	25	**27**	200	950
8	415	6	**18**	934	7	**28**	364	926
9	931	541	**19**	104	988	**29**	277	371
10	467	464	**20**	289	951	**30**	935	541

EK-10. 40 şehirli problem koordinatları

40 Şehirli Problem					
i. Şehir	**X**	**Y**	**i. Şehir**	**X**	**Y**
1	472	397	**21**	13	649
2	533	893	**22**	356	954
3	956	139	**23**	631	635
4	432	245	**24**	872	19
5	514	558	**25**	198	713
6	839	631	**26**	681	111
7	189	544	**27**	651	755
8	316	183	**28**	794	293
9	960	373	**29**	115	146
10	888	12	**30**	704	594
11	89	626	**31**	281	876
12	586	629	**32**	764	573
13	659	76	**33**	699	178
14	805	418	**34**	339	787
15	354	768	**35**	524	466
16	643	4	**36**	415	499
17	773	775	**37**	524	541
18	637	651	**38**	574	258
19	825	381	**39**	383	307
20	585	25	**40**	513	186

EK-11. 50 şehirli problem koordinatları

50 Şehirli Problem					
i. Şehir	**X**	**Y**	**i. Şehir**	**X**	**Y**
1	191	793	**26**	553	463
2	123	967	**27**	299	17
3	303	550	**28**	172	392
4	778	985	**29**	987	113
5	54	344	**30**	95	920
6	654	416	**31**	799	972
7	123	505	**32**	86	972
8	821	475	**33**	566	548
9	619	529	**34**	737	326
10	275	475	**35**	81	86
11	600	871	**36**	372	865
12	847	106	**37**	800	782
13	869	95	**38**	291	460
14	889	512	**39**	666	89
15	830	362	**40**	253	74
16	637	804	**41**	106	916
17	463	482	**42**	380	976
18	429	845	**43**	511	186
19	711	29	**44**	903	152
20	203	435	**45**	727	500
21	820	242	**46**	415	276
22	951	239	**47**	370	888
23	437	704	**48**	571	863
24	35	83	**49**	2	883
25	514	23	**50**	380	444

EK-12. 60 şehirli problem koordinatları

60 Şehirli Problem								
i. Şehir	**X**	**Y**	**i. Şehir**	**X**	**Y**	**i. Şehir**	**X**	**Y**
1	166	802	**21**	368	931	**41**	395	129
2	588	89	**22**	442	442	**42**	151	674
3	407	56	**23**	764	730	**43**	860	391
4	897	751	**24**	346	580	**44**	9	949
5	902	476	**25**	931	890	**45**	325	979
6	833	165	**26**	414	474	**46**	439	418
7	439	538	**27**	262	297	**47**	624	114
8	202	255	**28**	419	458	**48**	551	414
9	873	438	**29**	517	931	**49**	121	801
10	384	683	**30**	953	694	**50**	94	92
11	73	45	**31**	272	330	**51**	347	237
12	43	81	**32**	680	135	**52**	455	82
13	498	930	**33**	216	218	**53**	707	488
14	190	54	**34**	420	631	**54**	940	607
15	670	943	**35**	61	283	**55**	519	893
16	26	80	**36**	980	752	**56**	779	185
17	113	89	**37**	143	343	**57**	953	964
18	233	665	**38**	895	220	**58**	517	269
19	545	668	**39**	986	371	**59**	671	731
20	643	349	**40**	526	36	**60**	714	433

EK-13. 70 şehirli problem koordinatları

70 Şehirli Problem					
i. Şehir	**X**	**Y**	**i. Şehir**	**X**	**Y**
1	36	863	**36**	286	603
2	837	14	**37**	508	311
3	184	523	**38**	326	779
4	283	629	**39**	83	684
5	546	561	**40**	890	45
6	463	787	**41**	390	909
7	179	728	**42**	480	414
8	246	164	**43**	798	90
9	883	896	**44**	679	869
10	92	259	**45**	282	129
11	894	691	**46**	909	811
12	121	418	**47**	86	244
13	431	432	**48**	953	391
14	784	886	**49**	853	778
15	413	891	**50**	944	105
16	568	10	**51**	0	281
17	880	581	**52**	710	237
18	395	929	**53**	400	103
19	914	577	**54**	675	33
20	649	388	**55**	660	977
21	124	87	**56**	719	692
22	993	514	**57**	550	334
23	966	277	**58**	675	557
24	402	671	**59**	367	325
25	469	64	**60**	396	96
26	83	758	**61**	290	793
27	149	878	**62**	536	545
28	211	551	**63**	29	674
29	158	613	**64**	501	316
30	639	952	**65**	295	563
31	979	266	**66**	217	526
32	11	260	**67**	906	430
33	196	984	**68**	643	272
34	959	810	**69**	574	721
35	149	520	**70**	210	493

EK-14. 80 şehirli problem koordinatları

80 Şehirli Problem											
i. Şehir	**X**	**Y**	**i. Şehir**	**X**	**Y**	**i. Şehir**	**X**	**Y**	**i. Şehir**	**X**	**Y**
1	195	415	**21**	62	636	**41**	821	590	**61**	440	714
2	969	502	**22**	272	803	**42**	417	298	**62**	558	545
3	578	371	**23**	118	694	**43**	841	579	**63**	515	732
4	375	788	**24**	406	488	**44**	281	643	**64**	894	548
5	914	681	**25**	521	100	**45**	880	253	**65**	603	845
6	573	527	**26**	937	324	**46**	623	45	**66**	426	21
7	929	228	**27**	950	273	**47**	474	822	**67**	944	37
8	22	979	**28**	793	855	**48**	867	221	**68**	732	906
9	495	970	**29**	420	503	**49**	977	668	**69**	259	643
10	206	826	**30**	285	725	**50**	700	267	**70**	934	782
11	95	432	**31**	334	426	**51**	905	236	**71**	56	96
12	61	997	**32**	432	973	**52**	478	352	**72**	359	882
13	384	704	**33**	361	586	**53**	807	607	**73**	203	944
14	865	820	**34**	709	482	**54**	24	415	**74**	586	985
15	394	655	**35**	175	324	**55**	229	490	**75**	791	386
16	275	865	**36**	417	483	**56**	512	251	**76**	25	22
17	181	366	**37**	356	465	**57**	553	604	**77**	607	787
18	412	82	**38**	64	180	**58**	336	572	**78**	469	506
19	968	5	**39**	997	550	**59**	764	702	**79**	739	934
20	187	104	**40**	457	894	**60**	162	924	**80**	279	161

EK-15. 90 şehirli problem koordinatları

90 Şehirli Problem								
i. Şehir	**X**	**Y**	**i. Şehir**	**X**	**Y**	**i. Şehir**	**X**	**Y**
1	326	448	**31**	899	322	**61**	803	954
2	226	184	**32**	862	763	**62**	115	213
3	690	996	**33**	966	274	**63**	763	980
4	629	691	**34**	396	836	**64**	497	32
5	540	97	**35**	20	246	**65**	593	802
6	747	148	**36**	52	507	**66**	922	511
7	232	880	**37**	422	0	**67**	660	999
8	713	438	**38**	150	709	**68**	769	715
9	980	154	**39**	312	395	**69**	894	994
10	39	184	**40**	547	260	**70**	732	781
11	9	378	**41**	151	292	**71**	16	366
12	394	363	**42**	76	705	**72**	319	74
13	82	453	**43**	330	789	**73**	783	260
14	504	460	**44**	655	566	**74**	855	918
15	675	817	**45**	896	21	**75**	48	894
16	566	253	**46**	255	884	**76**	924	488
17	372	582	**47**	684	32	**77**	689	378
18	874	607	**48**	158	155	**78**	422	746
19	257	696	**49**	412	418	**79**	696	341
20	557	878	**50**	143	3	**80**	629	509
21	360	363	**51**	529	762	**81**	968	101
22	621	780	**52**	570	230	**82**	990	24
23	457	205	**53**	862	61	**83**	616	356
24	698	837	**54**	485	75	**84**	510	646
25	751	901	**55**	367	664	**85**	810	635
26	186	775	**56**	205	803	**86**	951	498
27	267	295	**57**	521	418	**87**	485	996
28	702	456	**58**	39	992	**88**	239	198
29	964	188	**59**	160	46	**89**	437	689
30	12	224	**60**	58	101	**90**	759	32

Ek-16. 100 şehirli problem koordinatları

100 Şehirli Problem											
i. Şehir	**X**	**Y**	**i. Şehir**	**X**	**Y**	**i. Şehir**	**X**	**Y**	**i. Şehir**	**X**	**Y**
1	0	31	**26**	557	206	**51**	746	80	**76**	80	921
2	861	202	**27**	681	592	**52**	76	59	**77**	838	681
3	272	671	**28**	955	644	**53**	106	795	**78**	927	277
4	318	161	**29**	998	243	**54**	805	381	**79**	700	734
5	372	425	**30**	676	295	**55**	612	436	**80**	499	663
6	82	474	**31**	85	772	**56**	740	493	**81**	158	230
7	70	840	**32**	495	881	**57**	341	818	**82**	946	364
8	59	293	**33**	509	573	**58**	817	141	**83**	126	986
9	917	367	**34**	953	684	**59**	93	67	**84**	306	54
10	774	327	**35**	338	7	**60**	60	193	**85**	101	866
11	697	844	**36**	705	988	**61**	672	227	**86**	554	697
12	717	306	**37**	774	749	**62**	570	254	**87**	100	668
13	162	329	**38**	197	143	**63**	365	837	**88**	16	309
14	466	246	**39**	913	784	**64**	314	314	**89**	904	535
15	825	279	**40**	589	864	**65**	656	692	**90**	327	442
16	481	149	**41**	689	287	**66**	902	318	**91**	900	347
17	874	287	**42**	93	628	**67**	478	296	**92**	353	487
18	772	976	**43**	280	877	**68**	766	740	**93**	71	850
19	492	887	**44**	161	273	**69**	551	723	**94**	760	180
20	827	20	**45**	543	960	**70**	411	542	**95**	922	493
21	141	143	**46**	174	157	**71**	595	880	**96**	127	892
22	500	21	**47**	268	177	**72**	668	112	**97**	403	207
23	592	9	**48**	576	498	**73**	471	403	**98**	807	587
24	774	650	**49**	285	155	**74**	114	53	**99**	109	623
25	770	708	**50**	609	739	**75**	548	673	**100**	927	540

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Soyadı, adı : ÖZKAN, Serçin

Uyruğu : T.C.

Doğum tarihi ve yeri : 15. 07. 1984 Ankara

Medeni hali : Evli

Telefon : 0 (312) 241 23 41

Cep telefonu : 0 (505) 718 69 44

e-mail : sozkan@tai. com. tr

Eğitim Derece	**Eğitim Birimi**	**Mezuniyet tarihi**
Yüksek lisans	Gazi Üniversitesi /Endsütri Mühendisliği	2007-
Lisans	Gazi Üniversitesi/ Endsütri Mühendisliği	2007
Lise	Ankara Anadolu Lisesi	2002

İş Deneyimi

Yıl	**Yer**	**Görev**
2008-	TAI-Tusaş A. Ş	Program Yöneticisi

Yabancı Dil

İngilizce, Almanca

Hobiler

İnternet, Bulmaca çözmek, Arjantin Tango