EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

AÇILABİLİR YÜZEYLERDE GEZGİN SATICI PROBLEMİNİN YAPAY ZEKA TEKNİKLERİYLE ÇÖZÜLMESİ

Kazım ERDOĞDU

Tez Danışmanı : Prof. Dr. Ali ÇALIŞKAN

Matematik Anabilim Dalı

Bilim Dalı Kodu : 403.02.01
Sunuş Tarihi : 29.12.2011

Bornova-İZMİR

2011

Kazım ERDOĞDU tarafından Yüksek Lisans tezi olarak sunulan "**Açılabilir Yüzeylerde Gezgin Satıcı Probleminin Yapay Zeka Teknikleriyle Çözülmesi**" başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 29.12.2011 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza**
Jüri Başkanı	**: .....**	**.....**
Raportör Üye	**: .....**	**.....**
Üye	**: .....**	**.....**

ÖZET

AÇILABİLİR YÜZEYLERDE GEZGİN SATICI PROBLEMİNİN YAPAY ZEKA TEKNİKLERİYLE ÇÖZÜLMESİ

ERDOĞDU, Kazım

Yüksek Lisans Tezi, Matematik Bölümü
Tez Danışmanı: Prof. Dr. Ali ÇALIŞKAN
Aralık 2011, 67 sayfa

Bu tezde açılabilir yüzeyler üzerinde gezgin satıcı probleminin yapay zeka tekniklerinden biri olan genetik algoritmalar ile çözümü incelenmiştir.

Açılabilir yüzeylerin tanımı ve özellikleri verildikten sonra, gezgin satıcı problemi ve genetik algoritmalar tanıtılmıştır. Genetik algoritmaların her bir aşaması (çaprazlama, mutasyon, sıralı seçim, elitizm) açıklanmıştır. Genetik algoritmalarda birden fazla çaprazlama ve mutasyon işlemleri mevcut olduğundan, sadece pozisyona dayalı, kısmi planlı ve tek noktalı çaprazlama çeşitleri ile 2-opt mutasyon incelenmiştir.

Tüm bu verilerin ışığında, gezgin satıcı problemini Öklid uzaklıklarıyla hesaplayan bir bilgisayar programı geliştirilmiştir. TSPLIB kütüphanesindeki Berlin52 veri seti için iyi sonuçlar elde edildikten sonra, gezgin satıcı problemi silindir yüzeyine uygulanmıştır. Farklı çaprazlama (Pozisyona dayalı, kısmi planlı ve tek noktalı) ve mutasyon (2-opt) işlemleriyle elde edilen deneysel sonuçlar sunulmuş ve kıyaslanmıştır.

Anahtar sözcükler: Açılabilir Yüzeyler, Silindir, Gezgin Satıcı Problemi, Genetik Algoritmalar, Yapay Zeka.

ABSTRACT

SOLVING THE TRAVELING SALESMAN PROBLEM ON DEVELOPABLE SURFACES USING ARTIFICIAL INTELLIGENCE TECHNIQUES

ERDOĞDU, Kazım

MSc. in Mathematics
Supervisor: Prof. Dr. Ali ÇALIŞKAN
December 2011, 67 pages

In this thesis, a solution with genetic algorithms, which is one of the artificial intelligence techniques, for traveling salesman problem on developable surfaces is studied.

Having given the definition and the characteristics of developable surfaces, traveling salesman problem and genetic algorithms are defined. Each process of genetic algorithms (i.e. crossover, mutation, rank selection, elitism) are explained. Since there is more than one crossover and mutation types in genetic algorithms, only position based, partially mapped, and single point crosover types and 2-opt mutation operator are studied.

In light of this information, a computer program is developed which solves the Euclidean traveling salesman problem. After obtaining good results for Berlin52 instances in TSPLIB, traveling salesman problem is adopted to cylinder surface. Experimental results for different crossover (Position Based, Partially Mapped and Single Point) and mutation (2-opt) operators are presented and compared.

Keywords: Developable Surfaces, Cylinder, Traveling Salesman Problem, Genetic Algorithms, Artificial Intelligence.

TEŞEKKÜR

Geometri alanında kendisinden çok şey öğrendiğim ve bu tez çalışmam boyunca beni yönlendiren hocam Prof. Dr. Ali ÇALIŞKAN'a, hem Yapay Zeka alanında ufkumu açan hem de bu tez konusunda yapay zeka uygulaması fikrini bana veren hocam Doç. Dr. Aybars UĞUR'a ve bilgisayar programcılığında kendisine çok şey borçlu olduğum hocam Yard. Doç. Dr. Murat Erşen BERBERLER'e teşekkürlerimi bir borç bilirim.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

ŞEKİLLER DİZİNİ (devam)

ÇİZELGELER DİZİNİ

ÇİZELGELER DİZİNİ (devam)

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler	Açıklama
$\vec{r}(u,v)$	Regle yüzey denklemi
$\vec{r}(v)$	Regle yüzeyin dayanak eğrisi
$\vec{\imath}(v)$	Regle yüzeyin doğrultmanı
u,v	Regle yüzeyin parametreleri
$[\mathrm{I}]$	Birinci esas form
ds	Yay uzunluğu
E, F, G	Birinci esas form bileşenleri
$\alpha(t)$	Dairesel helis eğrisi
α'	Dairesel helis eğrisinin t ye göre birinci türevi
α''	Dairesel helis eğrisinin t ye göre ikinci türevi
$\vec{\imath}_\alpha$	Dairesel helis eğrisinin normali
k_g	Jeodezik eğrilik
t	Eğri parametresi
b, r	Katsayılar
A, B	Eğri üzerindeki noktalar
i, j	Şehir indisleri
x_{ij}	i ile j şehirleri arasında yol olup olmadığını gösteren değişken

SİMGELER VE KISALTMALAR DİZİNİ (devam)

d_{ij}	i ile j şehirleri arasındaki uzaklık
n	Şehir sayısı
N	Şehirler kümesi
z	Amaç fonksiyonu
$O(n)$	Algoritma karmaşıklığı

Kısaltmalar

CX	Cycle Crossover – Dairesel Çaprazlama.
GSP	Gezgin Satıcı Problemi
LOX	Linear Order Crossover – Doğrusal Sıralı Çaprazlama.
OBX	Order Based Crossover – Sıraya Dayalı Çaprazlama.
OX	Ordered Crossover – Sıralı Çaprazlama.
PBX	Position Based Crossover – Pozisyona Dayalı Çaprazlama.
PMX	Partially Mapped Crossover – Kısmi Planlı Çaprazlama.
SPX	Single Point Crossover – Tek Noktalı Çaprazlama.

1. GİRİŞ

Geometri, Grekçe γεωμετρία (*geometria*) kelimesinden gelmektedir. Burada "geo" yer, "metria" ise ölçüm, yani geometri "yer ölçümü" anlamına gelmektedir. Geometri, matematiğin belki de en görsel ve çekici dalı olmasına rağmen, genellikle zor bir alan olarak bilinir. Örneğin, dönemin kralı I. Ptolemy (M.Ö. 323–283) okumakta zorluk çektiği Elementler'in yazarı olan Öklid'e (M.Ö. 300) şu soruyu sorar "Geometri'yi kestirmeden öğrenmenin yolu yok mu?" Öklid'in cevabı ise, "Özür dilerim ama geometriye giden bir kral yolu yoktur" olur. Başka bir gün dersini bitirdikten sonra öğrencilerden birisi Öklid'e "Hocam, verdiğimiz ispatlar çok güzel, ama pratikte bunlar neye yarar?" diye sorar. "Öklid'in yanıtı ise biraz sıra dışı olur. Öklid kapıda bekleyen kölesini çağırır ve "Bu delikanlıya 5-10 kuruş ver, vaktinin boşa gitmediğini görsün!" der (Yıldırım, 2001). Geometri zor bir alan olmasına rağmen, hem doğayı anlamamızda bize yardımcı olmakta hem de teknolojiye katkıda bulunmaktadır. Özellikle de tasarım alanında geometrinin katkıları büyüktür. Teknolojik tasarımlarda açılabilir regle yüzeyler sıklıkla kullanılmaktadır.

Gezgin satıcı problemi yöneylem araştırması ve bilgisayar bilimleri alanlarında incelenen bir kombinatorik optimizasyon problemidir. Gezgin satıcı probleminin ve varyasyonlarının uygulama alanı oldukça geniştir. Örnek olarak araç rotalama, bilgisayar ve network ağları, elektronik devre tasarımı, ulaşım ve lojistik uygulamaları, akış çizelgesi, vb. alanlar gösterilebilir (Lawler et al., 1985).

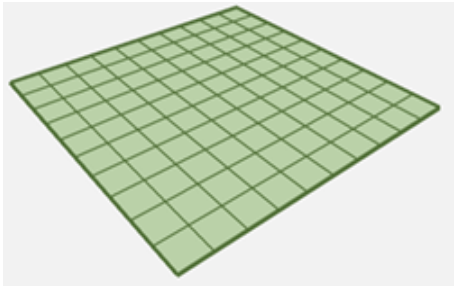
Yapay zeka ise bilgisayarların, dolayısıyla da makinelerin, verileri temsil etme, arama yapma, çıkarsama yapma, öğrenme ve mantıklı sonuçlar üretme (ya da mantıklı hareket etme) işlevlerini gerçekleştirebilmeleri üzerinde yapılan çalışmaları kapsamaktadır. Yapay zeka teknikleri arasında robotbilim, doğal dil işleme, uzman sitemler, yapay sinir ağları, karınca kolonisi sitemleri, bulanık mantık ve genetik algoritmalar yer almaktadır. Yapay zekanın kullanıldığı alanlara örnek olarak tıp, mühendislik, endüstri, internet, vb. gösterilebilir.

Bu tez çalışmasının amacı, geometri, bilgisayar, kombinatorik ve yapay zeka alanları birleştirilerek, gezgin satıcı problemi bir açılabilir yüzey olan silindir üzerinde, bir yapay zeka tekniği olan Genetik Algoritma yöntemiyle çözmektir. Bu tez çalışmasının ikinci bölümünde, regle yüzeyler ve açılabilir yüzeylerin tanımı verilmiş, bu yüzeylere ait örnekler sunulmuş ve bu yüzeyler üzerindeki jeodezik eğriler incelenmiştir. Üçüncü bölümde, gezgin satıcı problemi

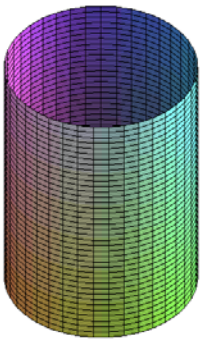
tanımlanmış, türlerinden bahsedilmiş ve de bu tez çalışmasında kullanılan gezgin satıcı türünün matematiksel modeli verilmiştir. Dördüncü bölümde, genetik algoritmaların tarihinden, yapısından ve kullanım alanlarından bahsedilmiştir. Genetik algoritmanın genel şablonu oluşturularak, algoritmanın her bir parçası (çaprazlama ve çeşitleri, mutasyon ve çeşitleri, elitizm, sıralı seçim) açıklanmıştır. Beşinci bölümde, gezgin satıcı problemini silindir üzerinde genetik algoritma tekniğiyle çözen ve Borland Delphi 7.0 bilgisayar programlama dili ile yazılmış bir bilgisayar programı tanıtılmış, birçok veri setiyle bu program denenmiş ve elde edilen sonuçlar tablolarla gösterilmiştir. Son bölüm olan sekizinci bölümde ise çalışmanın bir değerlendirmesi yapılarak konuyla ilgili ilerde yapılabilecek çalışmalara değinilmiştir.

2. REGLE YÜZEYLER

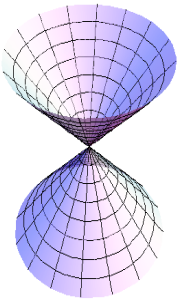
Bir doğrunun hareketiyle oluşmuş yüzeylere regle yüzeyler denir (Regle sözcüğü, Fransızcada "cetvel" ve "kural" anlamlarına gelmektedir). Bu yüzeylere en basit örnekler olarak düzlem, silindir, koni, helikoid, hiperbolik paraboloid, vb. gösterilebilir (Şekil 2.1) (Manfredo, 1976).



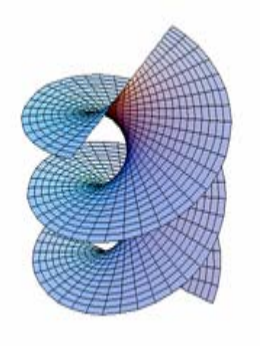
(a)



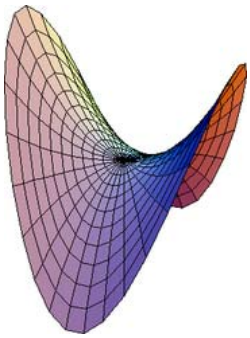
(b)



(c)



(d)



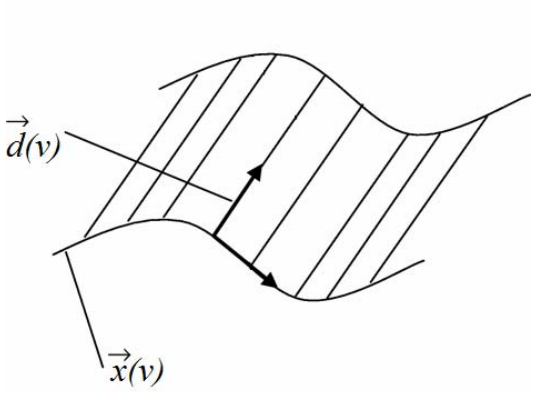
(e)

Şekil 2.1 – Regle Yüzey Örnekleri a) Düzlem, b) Silindir,
c) Koni, d) Helikoid, e) Hiperbolik Paraboloid

Regle yüzeylerin genel formu ise aşağıdaki gibidir:

$$\vec{y}(u,v) = \vec{x}(v) + u.\vec{d}(v) \tag{1}$$

Burada $\vec{x}(v)$ vektörüne regle yüzeyin dayanak eğrisi, $\vec{d}(v)$ vektörüne ise regle yüzeyin doğrultmanı denir (Şekil 2.2) (Biran, 1970).



Şekil 2.2 – Regle yüzeyin dayanak eğrisi ve doğrultmanı

Örneğin, bir silindirin parametrik denklemi, $y(u,v) = (r.\cos v,\ r.\sin v,\ u)$ dir. Ancak aynı denklem, silindir bir regle yüzey olduğu için regle yüzey denklemi olarak

$$\begin{aligned} &\vec{x}(v) = (r.\cos v,\ r.\sin v,\ 0) \\ &\vec{d}(v) = (0,0,1) \\ &\Rightarrow \vec{y}(u,v) = (r.\cos v,\ r.\sin v,\ 0) + u.(0,0,1) \end{aligned} \tag{2}$$

biçiminde de yazılabilir.

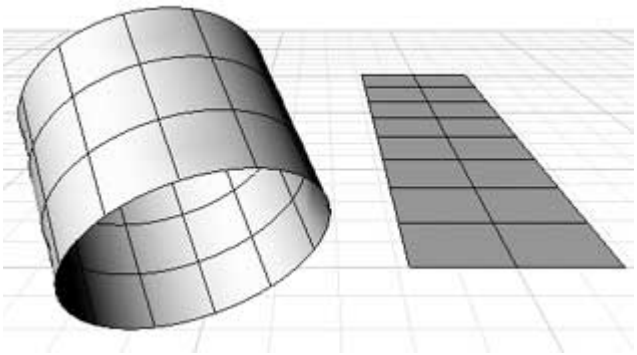
Benzer şekilde bir koninin regle yüzey denklemi $\vec{y}(u,v) = (r.u.\cos v,\ r.u.\sin v,\ 0) + u.(0,0,1)$, bir helikoidin regle yüzey denklemi $\vec{y}(u,v) = (r.u.\cos v,\ r.u.\sin v,\ 0) + u.(0,0,v/u)$ ve bir hiperbolik paraboloidin regle yüzey denklemi ise $\vec{y}(u,v) = (a.v,\ 0,\ v^2) + u.(a,b,2.v)$ şeklinde gösterilebilir.

2.1 Açılabilir Yüzeyler

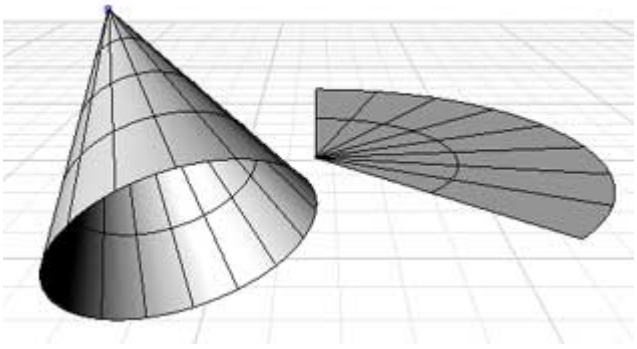
Bir regle yüzeyde

$$\left(\vec{d}, \vec{x}_v, \vec{d}_v\right) = 0 \qquad (3)$$

koşulu sağlandığı takdirde bu regle yüzeye "açılabilir yüzey" adı verilir. Diğer bir deyişle, bir regle yüzey bu koşulu sağladığında, bu regle yüzey özelliklerini koruyarak bir düzleme açılabilmektedir (Şekil 2.3) (Şekil 2.4). Açılabilir regle yüzeylere en basit örnekler olarak bir önceki başlık altında bahsettiğimiz yüzeyleri verebiliriz (Biran, 1970).



Şekil 2.3 – Açılabilir bir yüzey olan silindir (Rhino3DE'den)



Şekil 2.4 – Açılabilir bir yüzey olan koni (Rhino3DE'den)

Örneğin, silindir açılabilir bir yüzeydir, çünkü $\vec{x}(v) = (r.\cos v,\ r.\sin v,\ 0)$ ve $\vec{d}(v) = (0, 0, 1)$ olmak üzere, silindirin denklemi $\vec{y}(u, v) = x(v) + u.d(v)$ ise,

$$\left(\overset{\mathrm{r}}{d}, \overset{\mathrm{r}}{x}_v, \overset{\mathrm{r}}{d}_v\right) = \begin{vmatrix} 0 & 0 & 1 \\ -r.\sin v & r.\cos v & 0 \\ 0 & 0 & 0 \end{vmatrix} = 0 \qquad (4)$$

şartı sağlanır. Bu da silindirin bir açılabilir yüzey olduğunu gösterir.

2.2 Jeodezik Eğriler

Jeodezik eğriler, bulundukları yüzey üzerinde iki nokta arasındaki en kısa mesafeyi veren eğrilerdir. Bir yüzey üzerinde iki nokta arasında birden fazla eğri geçeceği açıktır. Ancak bu eğrilerden, bu iki nokta arasındaki en kısa mesafeyi veren eğriler, jeodezik eğrilerdir. Örneğin, bir düzlemin jeodezik eğrileri doğrular, kürenin jeodezik eğrileri ise küredeki en büyük yarıçaplı çemberler, ve silindirin jeodezik eğrileri ise silindir üzerinde o iki noktadan geçen ve en büyük adıma sahip dairesel helislerdir (Şekil 2.5) (Alshina, 2000; Kühnel, 2005).

(a) (b)

Şekil 2.5 – Jeodezik eğriler a) Küre, b) Slindir

Bir yüzeyin jeodezik eğrileri, o yüzeyin birinci esas formundan hesaplanmaktadır. Yani, $\overset{\mathrm{r}}{Y}(u,v)$ şeklindeki bir yüzeyimiz için birinci esas form [I] şu şekilde hesaplanmaktadır:

$E = \left\langle \overset{\mathrm{r}}{Y}_u, \overset{\mathrm{r}}{Y}_u \right\rangle$, $F = \left\langle \overset{\mathrm{r}}{Y}_u, \overset{\mathrm{r}}{Y}_v \right\rangle$, $G = \left\langle \overset{\mathrm{r}}{Y}_v, \overset{\mathrm{r}}{Y}_v \right\rangle$ olmak üzere, yüzeyin birinci esas formu,

$$\sqrt{[\mathrm{I}]} = ds = \int \sqrt{E.du^2 + 2.F.du.dv + G.dv^2} \qquad (5)$$

dir. Bu integralin sonucu bize, belirli bir yüzey üzerindeki iki nokta arasındaki en kısa uzaklığı vermektedir.

Örnek olarak silindir yüzeyini ele alalım. Silindirin denklemi, $\overset{\imath}{Y}(u,v)=(r.\cos u,\ r.\sin u,\ v)$ dir. O halde silindir üzerindeki iki nokta arasındaki uzaklık,

$$E=\left\langle \overset{\mathrm{r}}{Y}_u,\overset{\mathrm{r}}{Y}_u\right\rangle=\left\langle(-r.\sin u,\ r.\cos u,\ 0),(-r.\sin u,\ r.\cos u,\ 0)\right\rangle=r^2\left(\sin^2 u+\cos^2 u\right)=r^2$$
$$F=\left\langle \overset{\mathrm{r}}{Y}_u,\overset{\mathrm{r}}{Y}_v\right\rangle=\left\langle(-r.\sin u,\ r.\cos u,\ 0),(0,\ 0,\ 1)\right\rangle=0$$
$$G=\left\langle \overset{\mathrm{r}}{Y}_v,\overset{\mathrm{r}}{Y}_v\right\rangle=\left\langle(0,\ 0,\ 1),(0,\ 0,\ 1)\right\rangle=1$$
$$\Rightarrow\sqrt{[\mathrm{I}]}=ds=\int\sqrt{E.du^2+2.F.du.dv+G.dv^2}=\int\sqrt{r^2.du^2+dv^2}$$

ile hesaplanır. Bu diferansiyel denklem, tek parametreye indirgenerek – ya bir parametre diğerine bağlanarak, ya ikisi de tek bir parametreye bağlanarak, ya da birisi sabit tutularak – hesaplanır.

Bu uzaklığın en kısa çözüm yolu, silindir üzerindeki iki nokta arasından geçen ve en büyük adıma sahip helisi bulup, bu iki nokta arasındaki helisin yay uzunluğunu hesaplamaktır. Ama önce helisin gerçekten de silindirin jeodezik eğrisi olduğunu göstermemiz gerekmektedir.

Denklemi $\alpha(t)=(r.\cos t,\ r.\sin t,\ b.t)$ olan bu dairesel helisin $\overset{\imath}{Y}(u,v)=(r.\cos u,\ r.\sin u,\ v)$ silindirimizin üzerinde bir jeodezik eğri olması için gerekli koşul jeodezik eğriliğinin sıfıra eşit olmasıdır ($k_g=0$).

$\overset{\mathrm{r}}{n}_\alpha=(\cos t,\ \sin t,\ 0)$, $\alpha'=(-r.\sin t,\ r.\cos t,\ b)$, $\alpha''=(-r.\cos t,\ -r.\sin t,\ 0)$ den dolayı

$$k_g=\left(\overset{\mathrm{r}}{n}_\alpha,\ \alpha',\ \alpha''\right)=\begin{vmatrix}\cos t & \sin t & 0\\ -r.\sin t & r.\cos t & b\\ -r.\cos t & -r.\sin t & 0\end{vmatrix}=b.r.\cos t.\sin t-b.r.\sin t.\cos t=0 \qquad (7)$$

dır. Jeodezik eğriliğin sıfıra eşit olması, dairesel helisin silindir üzerinde bir jeodezik eğri olduğunu göstermektedir. Ancak silindir üzerindeki iki nokta arasındaki en kısa mesafeyi veren $\alpha(t)=(r.\cos t,\ r.\sin t,\ b.t)$ helisinin adımının (b katsayısının) bilinmesi gerekir. Bunu da, vida adımından yararlanarak

hesaplayabiliriz. Vidalarda adım ötelemenin döndürmeye oranı olarak hesaplanmaktadır. O halde silindir üzerindeki jeodezik helis eğrisinin adımı da, birleştirdiği noktaların dikey açılarının farkının (öteleme) yatay açılarının farkına (döndürme) eşit olacaktır. Bunun için, silindir üzerindeki birinci noktamız $A=(u_1,v_1)$ ve ikinci noktamız da $B=(u_2,v_2)$ ise, helisin denklemindeki adım olan b katsayısı, $b=\dfrac{v_1-v_2}{u_1-u_2}$ olur. O halde jeodezik helis eğrimiz de $\alpha(u)=\left(r.\cos u,\ r.\sin u,\ \dfrac{v_1-v_2}{u_1-u_2}.u\right)$ halini alır. A ile B arasındaki uzaklık da,

$$ds=\int_{u_1}^{u_2}\sqrt{(\alpha_1')^2+(\alpha_2')^2+(\alpha_3')^2}\,du=\int_{u_1}^{u_2}\sqrt{(-r.\sin u)^2+(r.\cos u)^2+\left(\frac{v_1-v_2}{u_1-u_2}\right)^2}\,du$$

$$=\sqrt{r^2+\left(\frac{v_1-v_2}{u_1-u_2}\right)^2}.|u_2-u_1| \qquad (8)$$

dır.

3. GEZGİN SATICI PROBLEMİ

Her ne kadar gezgin satıcı probleminin kökeni belirsiz olsa da, 1832 yılında Almanya'da basılmış olan bir kitapta gezgin satıcı probleminin temellerinden bahsedilmiştir (Lawler et al., 1985). Matematik alanında ise bu problem ilk defa 1932 yılında Karl Menger'in çalışmalarında ele alınmıştır. Menger, aslında gezgin satıcı probleminin bir varyasyonu olan *Botenproblem (Haberci problemi)* üzerinde çalışmıştır. Haberci problemi, aralarındaki uzaklıkları bilinen belirli sayıdaki nokta arasından, bir habercinin bir noktadan başlayıp varacağı noktaya en kısa mesafeli yol oluşturması problemidir. Menger'in bu çalışması, gezgin satıcı problemi üzerine yapılmış ilk çalışmadır. Ancak gezgin satıcı probleminin kombinatorik optimizasyon olarak ilk defa sistematik bir şekilde ele alınması ise 1954 yılında G. B. Dantzig, D. R. Fulkerson ve S. M. Johnson'un çalışmalarıyla başlamıştır (Gutin, 2004).

Gezgin satıcı problemi, yöneylem araştırması ve bilgisayar bilimleri alanlarında incelenen bir kombinatorik optimizasyon problemidir. Günümüzde birçok alanda bu problemi ya da varyasyonlarını görebiliriz. Bu tez çalışmasında kullanılan gezgin satıcı problemi, simetrik gezgin satıcı problemidir. Simetrik gezgin satıcı probleminde her bir şehir çifti için, *i.* şehrin *j.* şehre uzaklığı, *j.* şehrin *i.* şehre uzaklığına eşittir. Gezgin satıcı probleminin en çok görüldüğü alanlar arasında araç rotalama, bilgisayar ve network ağları, elektronik devre tasarımı, ulaşım ve lojistik uygulamaları, akış çizelgesi, vb. yer almaktadır (Lawler et al., 1985).

Gezgin satıcı problemini kısaca şu şekilde tanımlayabiliriz: bir satıcı elindeki malları gideceği n adet şehirde satmak istiyor. Satıcının her şehre sadece ve sadece bir kez uğraması ve başladığı şehre tekrar geri dönmesi gerekiyor. Problem, bu gezgin satıcının bu işi en az maliyetle yapması için nasıl bir rota çizmesi gerektiğini araştırmaktadır.

Simetrik gezgin satıcı probleminin matematiksel modeli şu şekilde tanımlanabilir (Gutin, 2004; Hoffman, 2011; Lawler et al., 1985):

$$x_{ij} = \begin{cases} 1, & i.\text{şehirden } j.\text{şehre gidilmişse} \\ 0, & \text{aksi takdirde} \end{cases} \tag{9}$$

d_{ij} : $i.$ şehir ile $j.$ şehir arasındaki uzaklık

n : *şehir sayısı*

N : *şehirler kümesi*

Amaç fonksiyonu:

$$z = \min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} . x_{ij} \tag{10}$$

Kısıtlamalar:

$$1. \quad \sum_{i=1}^{n} x_{ij} = 2, \qquad j = 1, 2, ..., n \tag{11}$$

$$2. \quad \sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \qquad \forall S \subset N \tag{12}$$

Bu kısıtlamalardan ilki her şehrin en fazla bir kez kullanıldığını ve hiçbir şehrin açıkta kalmamasını, ikincisi ise şehirler arasında alt turların oluşmamasını kontrol etmektedir. Dolayısıyla gezgin satıcı probleminin amacı z amaç fonksiyonunu minimum yapmaktır, yani tüm şehirler dolaşıldıktan sonra kat edilen mesafenin minimum olmasıdır.

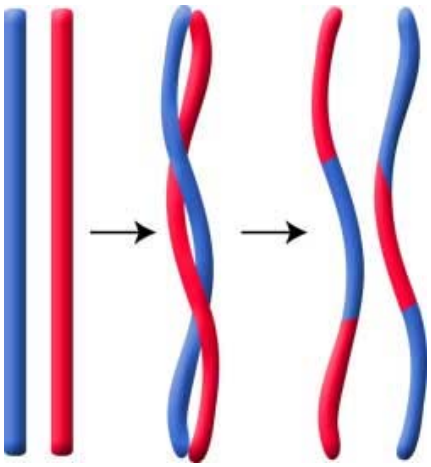
Gezgin satıcı problemi NP-Tam (Non-deterministic polynomial time, complete) problemdir, yani bu problem belirsiz Turing makinesiyle (non-deterministic Turinc machine) polinom zamanda çözülebilen (NP, Non-deterministic polynomial time) ve sınıfının en zor çözülen (NP-hard, Non-deterministic polynomial time - hard) problemlerinden birisidir (Lawler et al., 1985). Gezgin satıcı problemini tüm olası durumlar için polinom zamanda çözebilecek bir algoritma yoktur. Brute-force algoritmalar ile gezgin satıcı probleminin zaman karmaşıklığı $O(n!)$ ve dinamik algoritmalarla ise $O(n^2.2^n)$ dir (Gutin, 2004; Woeginger, 2011). Yani problemin kesin çözümüne ancak $n!$ veya $n^2.2^n$ adımdan sonra ulaşılabilmektedir. Bunun anlamı, şehir sayısı arttıkça, kesin çözüm için gereken zaman da hızlı bir şekilde artmaktadır. Hatta bazı durumlarda kesin çözüm için gereken zaman gerçel zamanda çözülemeyecek kadar büyüktür. Bu yüzden de gezgin satıcı problemini uygun zamanda ve etkin bir şekilde çözdürmek için, günümüzde sezgisel yöntemler ve yapay zeka yöntemleri kullanılmaktadır. Bu yöntemlerin çoğu, ya en iyi sonucu ya da en iyiye

yakın sonuçları, gerçel zaman içerisinde ve klasik yöntemlere göre daha kısa sürede çözmektedir. Bu yöntemlerden birisi de, bir evrimsel programlama yöntemi olan “Genetik Algoritmalar”dır (Nabiyev, 2005; Russell and Norvig, 2002; Uğur, 2008).

4. GENETİK ALGORİTMALAR

Genetik algoritmaların tarihi 1950'li yıllara dayanmaktadır. Norveç-İtalyan asıllı matematikçi Nils Aall Barricelli'nin symbiogenesis alanındaki deneyleri, yapay zeka çalışmalarına öncülük etmiştir. Ancak, tam anlamıyla ve belirgin olarak genetik algoritmalar üzerindeki çalışmalar 1975 yılında John Holland tarafından yapılmıştır. 1975 yılında Holland, "*Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*" ("*Doğal ve Yapay Sistemlerde Adaptasyon: Biyoloji, Kontrol ve Yapay Zeka Alanlarında Uygulamalarıyla Birlikte Bir Giriş Analizi*") adlı çalışmasında genetik algoritmalara değinmiştir. Ardından Holland'ın tez danışmanlığı yaptığı Kenneth De Jong'un çalışmalarıyla da genetik algoritmalar daha da gelişmiştir (Mitchell, 1996). Günümüzde ise genetik algoritmalar, birçok alanda başarıyla uygulanmakta ve değişik yöntemlerle daha da geliştirilmektedir. Bu alanların başında da optimizasyon gelmektedir. Özellikle gezgin satıcı problemi, genetik algoritmaların uygulandığı yaygın bir alandır.

Genetik algoritmalar, adından da anlaşılacağı üzere genetik bilimden faydalanılarak oluşturulmuştur. Gen, temel kalıtım birimidir. Bir canlı organizmanın genleri, hücre içerisinde DNA (Deoxyribo Nucleic Acid) adlı bir çift kromozomu oluştururlar. Her canlı türünün hücre içi kromozom sayısı farklıdır. Örneğin; ayçiçeğinde 34, bezelyede 14, kedide 38, köpekte 78 ve insanda 46 adet kromozom vardır (Genome News Network, 2003a). Bir kromozomda, canlının belirgin bir karakteristiğini belirleyen gen kümeleri vardır (örneğin saç rengi ve biçimi gibi). Bu gen kümelerine "allel" adı verilmektedir. Bu allellerin kombinasyonlarına göre bireyin kişisel özellikleri belirlenir. Genetikte yeni bireylerin oluşumu anne ve baba bireylerden gelen kromozomların bölünmesi ve bu bölünen kısımların birbirleriyle birleşmesiyle (çaprazlamayla) oluşmaktadır (Şekil 4.1). Çaprazlama sonucunda baskın olan alleller bu yeni bireyin kişisel niteliklerini belirlemektedir (Mitchell, 1996).



Şekil 4.1 – Kromozom çaprazlaması (Genome News Network'dan 2003b)

Bilgisayar programcılığındaki genetik algoritmalar da benzeri bir yöntemi kullanmaktadırlar. Bilgisayarda, kromozomlar dizilere, genler dizinin elemanlarına ve de popülasyonlar da bu dizilerden oluşan matrislere karşılık gelmektedir. Problemin niteliğine göre, genlerin ve kromozomların ne anlama geldiği de değişmektedir. Eğer problem bir zamanlı akış tipi çizelgeleme problemi ise o zaman her bir gen yapılacak işin indisini temsil etmekte, her bir kromozom da hangi işin hangi sırada yapılacağını göstermektedir. Eğer problemimiz bir gezgin satıcı problemi ise, genler şehir indislerini temsil etmekte, kromozomlar ise bu şehirler arasında nasıl bir tam tur yapılacağını belirtmektedir. Genetik algoritmalarda, yapılan işlemin amacı, mevcut popülasyonlardaki en iyi bireyi ya da kromozomu bulmaktır. Bunu yaparken de, problemin amacına yönelik bir uygunluk fonksiyonu bulunur ve bu fonksiyon, popülasyonda var olan tüm bireylere uygulanır. İyi bireyler, yani en iyi sonuçlar üreten birey ya da bireyler, sonraki nesillere aktarılır. Bu işleme "elitizm" denmektedir. Aynı zamanda, popülasyondaki bireylere istenilen miktarda çaprazlama ve mutasyon işlemleri uygulanır ve yeni oluşan bireyler de uygunluk değerleri hesaplanarak yeni nesle aktarılır. Oluşan yeni neslin boyutu da problemin en başında belirlenen boyuta indirgenir ve artık algoritmaya bu yeni nesilden devam edilir. Bu yeni nesle de aynı işlemler uygulanarak algoritma istenilen miktarda çalıştırılır. Algoritma ise, ya önceden belirlenen iterasyon sayısı kadar uygulandığında ya da elde edilen sonuçların önceden belirlenen bir hata payıyla elde edilmesinden sonra sonlandırılır. Tüm bu adımlardan sonra o ana kadar elde edilen en iyi birey, Genetik algoritmanın üretmiş olduğu en iyi çözümdür (Larranaga et al., 1999; Russel and Norvig, 2002).

Genel olarak bir Genetik Algoritma'da şu adımlar takip edilir (Uğur vd., 2009):

Rastgele n adet kromozomlu bir nesil oluştur.

Repeat

Uzaklık Matrisini kullanarak nesildeki tüm kromozomların uygunluk değerlerini hesapla.

Uygunluk değerlerine göre nesilden iki ebeveyn kromozom seç.

Belirli çaprazlama ve mutasyon türleri ve yüzdeleriyle yeni çocuk kromozomlar oluştur.

Until *(Gerekli koşul sağlandığında algoritmayı durdur)*

4.1 Çaprazlama ve Çeşitleri

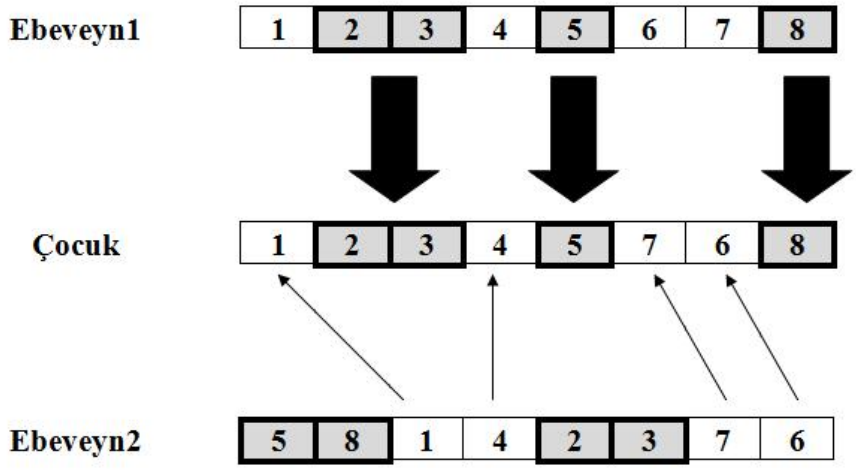
Genetik algoritmalarda yapılan çaprazlama, biyolojik çaprazlamaya benzemektedir. Genetik algoritmalardaki çaprazlamada da amaç, ebeveyn bireylerden (dizilerden) yeni çocuk bireyler (diziler) oluşturmaktır. Öncelikle, tüm ebeveyn bireyler arasından çaprazlamaya sokulacak iki birey rastgele ya da sezgisel olarak belirlenir. Bu iki bireyin genleri (dizinin elemanları) belirli koşullar altında karşılıklı olarak değiştirilir. Bu sayede yeni bireyler (diziler) oluşturulur ve gerek mutasyona uğratılarak gerekse hiçbir değişiklik yapmadan doğrudan uygunluk fonksiyonundan geçirilerek uygunluk değerleri ölçülür. Bu değerlerin sonucuna göre de belirlenen kriterlerden geçen yeni bireyler, algoritmayı devam ettirecek olan yeni nesle aktarılır. Çaprazlama işlemi her bir ebeveyn bireyine uygulanmak zorunda değildir. Programcı ya da kullanıcı bu yüzdeyi kendisi belirlemektedir.

Genetik algoritmalarda birden fazla çaprazlama yöntemi vardır. Bunlara örnek olarak Tek Noktalı Çaprazlama (Single Point Crossover – SPX), Pozisyona Dayalı Çaprazlama (Position Based Crossover – PBX), Kısmi Planlı Çaprazlama (Partially Mapped Crossover – PMX), Sıralı Çaprazlama (Ordered Crossover – OX), Sıraya Dayalı Çaprazlama (Order Based Crossover – OBX), Dairesel Çaprazlama (Cycle Crossover – CX), Doğrusal Sıralı Çaprazlama (Linear Order

Crossover – LOX), vb. gösterilebilir (Engin ve Fığlalı, 2002). Aşağıda, bu tez çalışmasında kullanılan üç çaprazlama türü açıklanmaktadır.

4.1.1Pozisyona dayalı çaprazlama (Position based crossover – PBX)

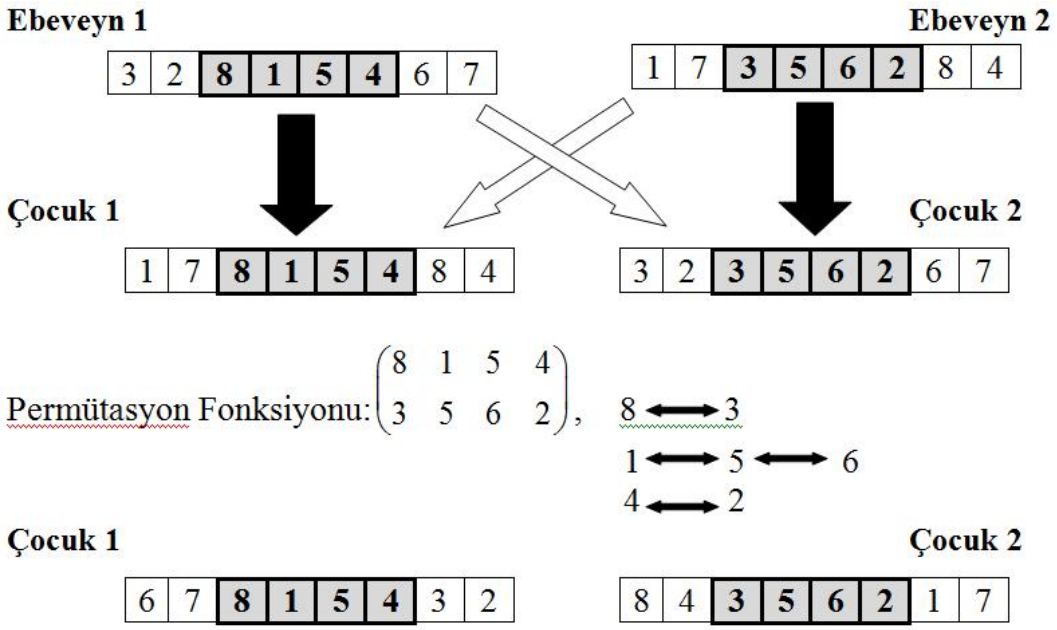
Bu çaprazlama yönteminde, seçilen ebeveynlerin birincisinin genlerinden çaprazlanacak olanlar rastgele seçilir ve bu genler çocuklarda da aynı konumlara gelecek şekilde aktarılırlar. Çocuklarda eksik kalan diğer genler ise, çocuğa gen aktarmayan diğer ebeveynin genlerinden sırasıyla alınarak tamamlanır (Şekil 4.2).



Şekil 4.2 – PBX

4.1.2 Kısmi planlı çaprazlama (Partially mapped crossover – PMX)

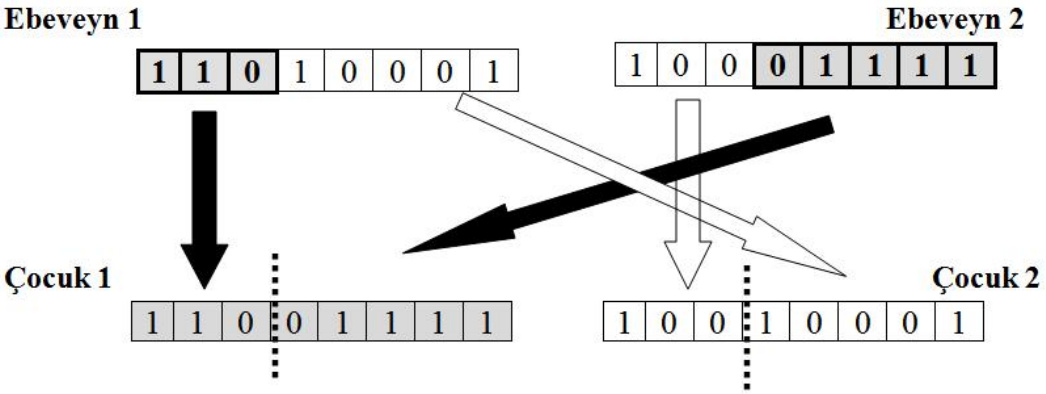
Kısmi planlı çaprazlamada, öncelikle iki ebeveynden de çocuklara aktarılacak blok halindeki gen grupları belirlenir. Bunun için de iki ebeveyn için ortak olan başlangıç ve bitiş genleri rastgele (ya da sabit bir şekilde) belirlenir. Ardından, bu noktalar arasındaki genler birinci ebeveynden birinci çocuğa, ikinci ebeveynden de ikinci çocuğa aktarılır. Çocuklarda geriye kalan genler ise karşıt ebeveynlerden, yani o çocuğa gen aktarmamış ebeveynden alınarak tamamlanır. Permütasyon tipindeki problemlerde (örneğin gezgin satıcı problemi) PMX uygulanırken, çocuklarda boş kalan yerlere genler aktarıldığı sırada aynı elemanın aktarılmamasına dikkat edilmelidir. Bu durumda, aktarılacak her bir gen, permütasyon fonksiyonu ile tekrarlar engellenecek şekilde ayarlanır (Şekil 4.3).



Şekil 4.3 – PMX ve Permütasyon fonksiyonu

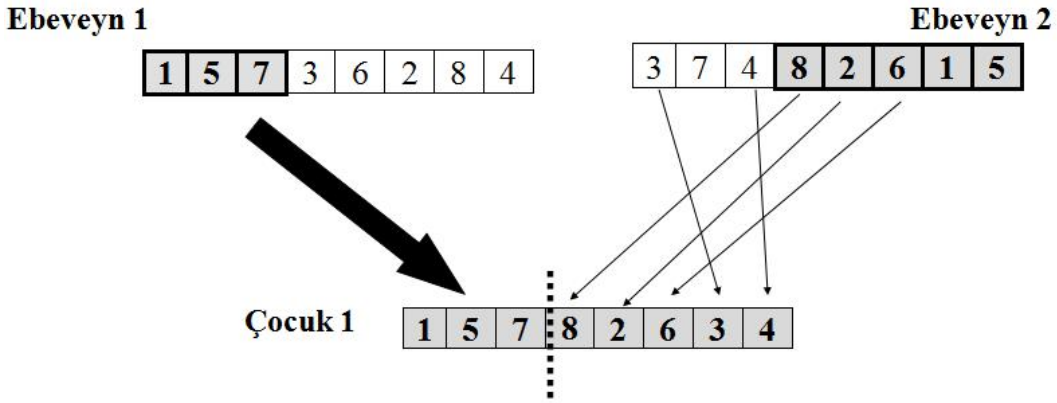
4.1.3 Tek noktalı çaprazlama (Single point crossover – SPX)

Bu çaprazlama türünde, diğerlerinden farklı olarak ebeveyn genleri tek noktadan çaprazlanmaktadır. Diğer bir değişle, öncelikle iki ebeveyn için de ortak bir çaprazlama noktası seçilir. Bu sayede her bir ebeveynin genleri iki gruba ayrılmış olur: 1) birinci genden çaprazlama noktasına kadar olan kısım ve 2) çaprazlama noktasından son gene kadar olan kısım. Birinci ebeveynin ilk kısmı ile ikinci ebeveynin ikinci kısmı birinci çocuğa, ikinci ebeveynin birinci kısmı ile birinci ebeveynin ikinci kısmı ise ikinci çocuğa aktarılarak yeni bireyler oluşturulur (Şekil 4.4).



Şekil 4.4 – SPX

Eğer problem gezgin satıcı probleminde olduğu gibi permütasyon tipi bir problem ise, o zaman ikinci sırada aktarılan ebeveyn genleri tekrar etmeyecek şekilde yeni nesle aktarılmalıdır. Bu da genellikle, ya Şekil4.4'teki şekildeki gibi, birinci ve ikinci ebeveyndeki kısımlar doğrudan yeni nesle aktarıldıktan sonra tekrar eden genler düzenlenerek yapılır, ya da, birinci ebeveynden aktarılacak gen bloğu yeni nesle aktarıldıktan sonra, ikinci ebeveynden aktarılacak genlerinin yeni nesilde var olanlarla kontrol edilip aynı olmaması durumunda aktarılmasıyla elde edilir (Şekil 4.5).



Şekil 4.5 – Permütasyon tipi problemlerde SPX

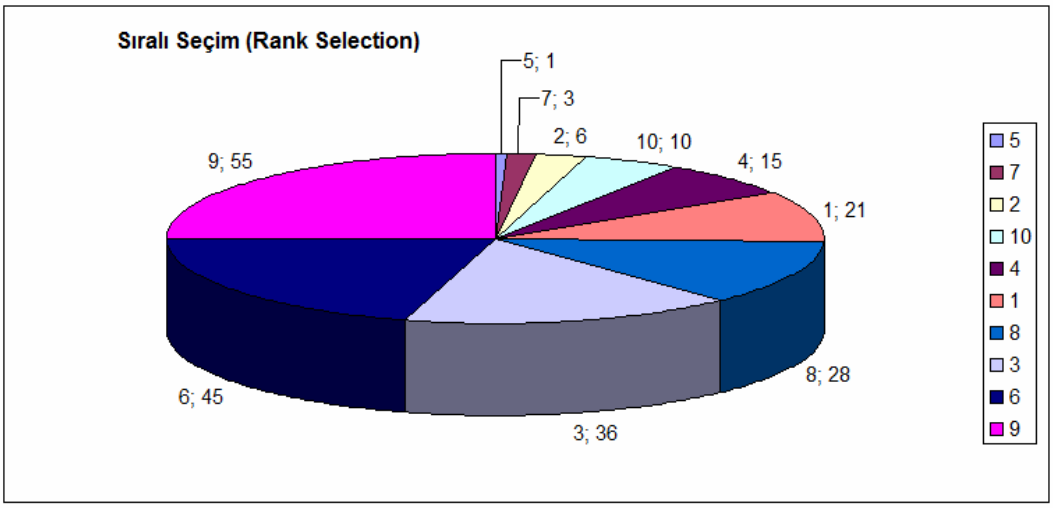
4.2 Sıralı Seçim (Rank Selection)

Sıralı seçim (rank selection), çaprazlanacak ebeveynlerin seçiminde kullanılan bir yöntemdir. Seçim metodunun uygulanması için, ilk önce ebeveyn neslindeki tüm bireylerin (kromozomların) toplam uygunluk değeri hesaplanır. Ardından tüm bu bireyler, uygunluk değeri en kötüden en iyiye, yani gezgin satıcı probleminde uygunluk değeri en büyükten en küçüğe doğru sıralanırlar. Sıralanmış bu bireylere ardışık olarak sıra (rank) sayısı atanır. En kötü olan birinci bireye 1, sonraki ikinci bireye 2, üçüncü bireye 3, vb. olacak şekilde işlem devam eder. Bunun anlamı, en kötü bireyin seçilme payı 1, sonraki kötü bireyin 2, ondan sonraki bireyin 3, vb. olacağıdır. Tüm bu paylar kümülatif olarak toplanır ve sıralı seçimin üst sınırı belirlenir (Çizelge 4.1) (Mitchell, 1996).

Sıralı seçime göre ebeveynlerin seçilmesi ise, 1 ile sıralı seçimin üst sınırı arasında rastgele bir sayı ürettirilip bunun sıralı seçimdeki karşılığı olan bireyin (kromozomun) seçilmesiyle olmaktadır. Bu şekilde seçilen iki ebeveyn çaprazlanarak ve gerekirse mutasyona uğrayarak sonraki nesle aktarılır.

Çizelge 4.1 – Örnek bir sıralı seçim

Ebeveyn	5	7	2	10	4	1	8	3	6	9
Sıra (Rank)	1	2	3	4	5	6	7	8	9	10
Kümülatif Toplam	1	3	6	10	15	21	28	36	45	55



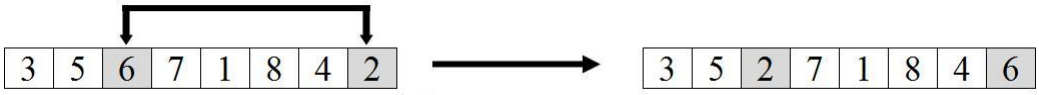
4.3 Mutasyon

Genetik algoritmalarda mutasyon işlemi, çaprazlamadan farklı olarak tek bir birey üzerinde yapılmaktadır. Yani, çaprazlama ile oluşan yeni bireylerin kendi içerisinde yapılan değişikliklere genetik algoritmalarda mutasyon işlemi denmektedir. Mutasyon işleminin amacı da çaprazlama gibi en iyi sonucu veren bireylerin üretilmesine yardımcı olmaktır. Bazen çaprazlama ile üretilen sonuçlar, yerel optimumlara takılmaktadır. Ancak, mutasyon işlemi bu tür durumlarda programın yerel optimumdan kurtulmasını sağlayarak en iyi çözümün bulunmasına katkı sağlamaktadır. Diğer taraftan, genetik algoritmalarda mutasyon işleminin fazla uygulanması da, sonuçları en iyi çözümden uzaklaştırabilmektedir. Bunun için mutasyon işleminin, her bireye uygulanmaması, popülasyonun çok düşük bir yüzdesine uygulanması (örneğin %3 gibi), genetik algoritmanın daha verimli ve hızlı çalışmasına olanak sağlamaktadır.

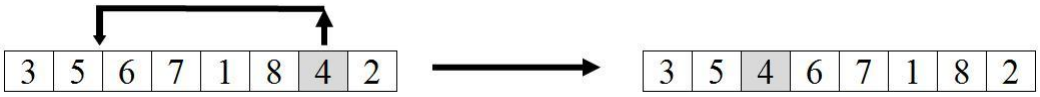
Genetik Algoritmalarda birden fazla mutasyon çeşidi vardır. Bunlara örnek olarak, sadece bir genin değiştirilmesi (Şekil 4.6), iki genin yer değiştirmesi (Şekil 4.7), iki genin arasına başka bir genin eklenmesi (Şekil 4.8), ya da belirli aralıktaki genlerin tersten yazılması (Şekil 4.9), 2-opt (Şekil 4.10), vb. verilebilir. Bu mutasyon çeşitlerinden özellikle 2-opt yönteminde, gezgin satıcı turundaki 2 yol silinip bunlar tersten birleştirilir (Şekil 4.10). Bu yeni dizilişin uygunluk değeri ölçülür. Eğer elde edilen sonuç bir önceki durumun sonucundan iyi ise yola

bu diziliş (birey) ile devam edilir. Tüm bu işlem, kromozomdaki tüm ikililere uygulanır ve sonunda mevcut en iyi birey elde edilir (Özkan, 2008; Uğur, 2008).

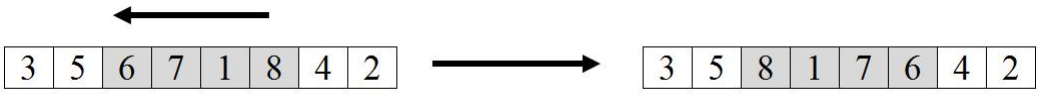
Şekil 4.6 – Tek genin değiştiği mutasyon



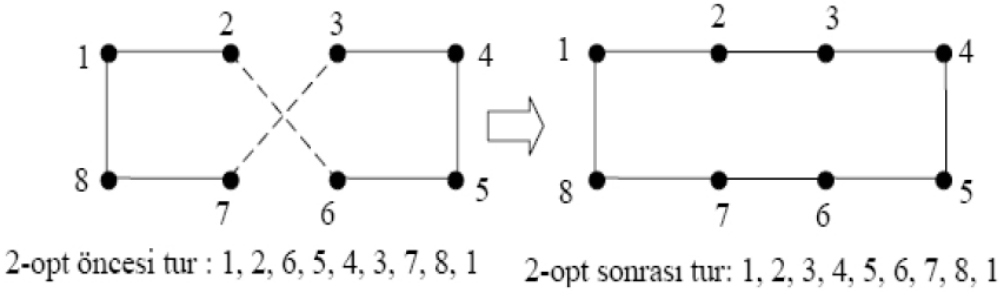
Şekil 4.7 – İki genin yer değiştirdiği mutasyon



Şekil 4.8 – İki genin arasına başka bir genin eklendiği mutasyon



Şekil 4.9 – Belirli aralıktaki genlerin tersten yazıldığı mutasyon



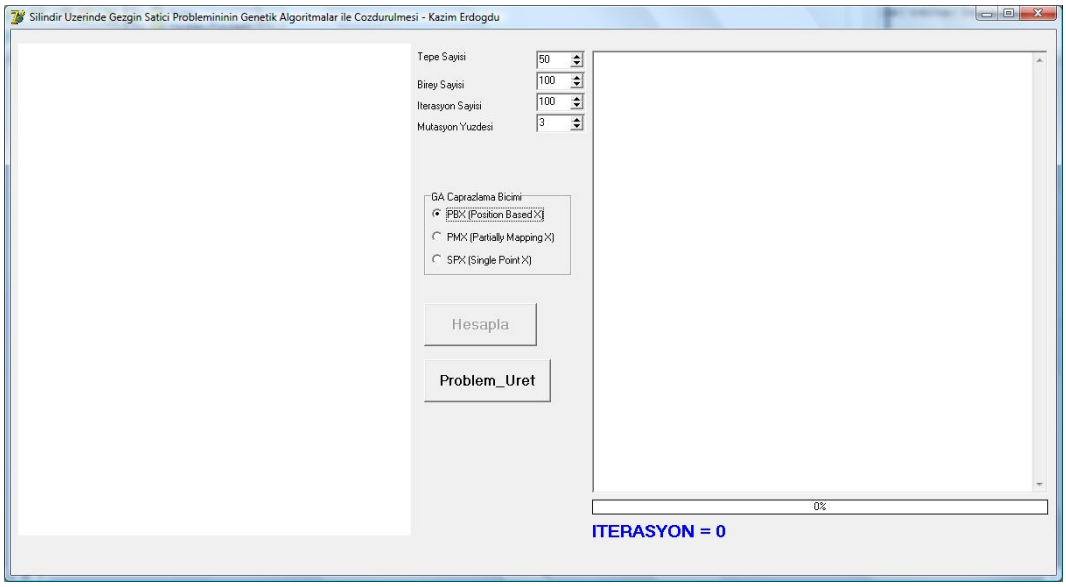
Şekil 4.10 – 2-Opt mutasyon (Özkan'dan 2008)

5. GEZGİN SATICI PROBLEMİNİN SİLİNDİR ÜZERİNDE GENETİK ALGORİTMA İLE ÇÖZÜLMESİ

5.1 Programın Yapısı ve Çalışma Şekli

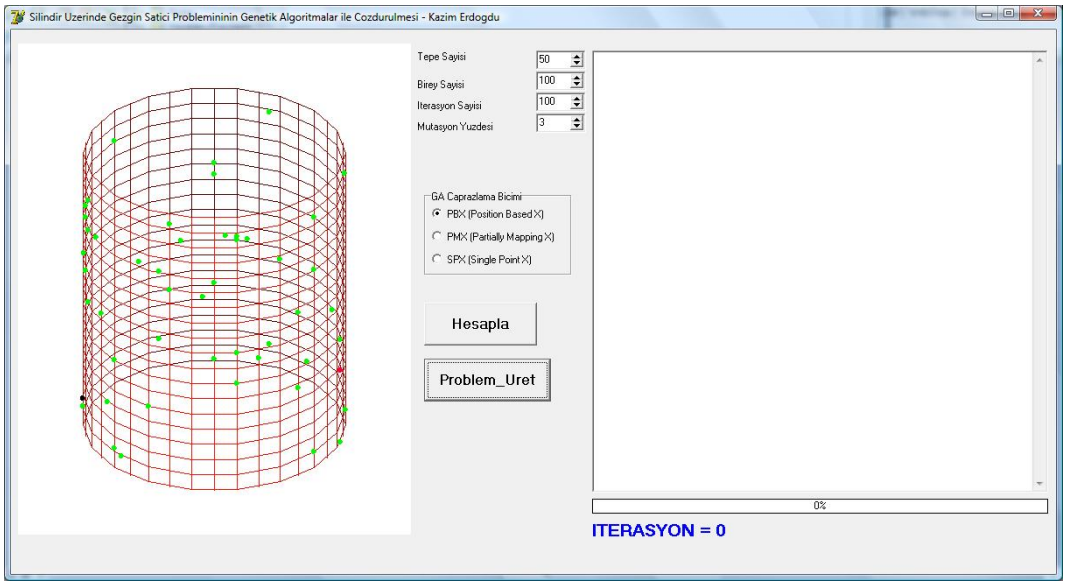
Bu tez çalışmasında, uygulama olarak, açılabilir bir yüzey olan silindir üzerinde gezgin satıcı problemi genetik algoritma yöntemiyle çözdürülmüştür. Programlama dili olarak ise Borland Delphi 7.0 kullanılmıştır.

Program çalıştırıldığında ekrana çizim alanının, sonuç panelinin, problemin tanımına ve çözümüne yönelik ayarların yapıldığı kutucukların ve butonların bulunduğu bir form gelmektedir (Şekil 5.1). Ekrana gelen bu formda, başlangıçta kullanıcıya kolaylık olması için bazı değerler otomatik olarak verilmiştir. Kullanıcı gerek bu değerleri aynen kullanabilmekte, gerekse de onları değiştirerek kendi problemini şekillendirebilmektedir.



Şekil 5.1 – Ana form

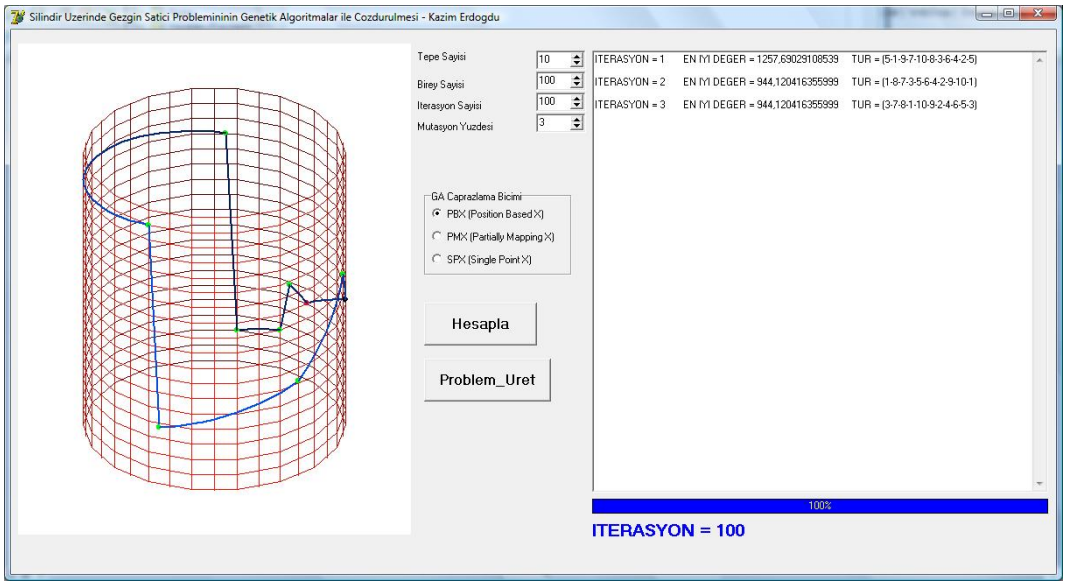
Kullanıcı, öncelikle ortada bulunan kutucuklardan "Tepe Sayısı" adlı olanına tıklayarak gezgin satıcı probleminin tepe sayısını belirlemeli ve "Problem_Uret" butonuna basarak problemi üretmelidir. Programdaki minimum tepe sayısı 4 ve maksimum tepe sayısı 200'dür. "Problem_Uret" butonuna tıklar tıklamaz sol panelde bir silindir ve üzerinde belirlenen sayıda rastgele oluşturulmuş tepeler görünmektedir (Şekil 5.2).



Şekil 5.2 – Problemin üretilmesi

Bu sayede silindir üzerindeki gezgin satıcı problemi tanımlanmış olur. Problemin çözümü içinse, ilgili kutucuklara gidilerek çalıştırılacak genetik algoritmanın parametreleri, yani birey sayısı, iterasyon sayısı, mutasyon sayısı ve çaprazlama şekli, belirlenmelidir. Programda seçilebilecek birey sayısı 2 ile 1000 arasında, iterasyon (nesil) sayısı 1 ile 1000 arasında ve de mutasyon yüzdesi ise 0 ile 100 arasında değişmektedir. Kullanılabilecek çaprazlama türleri ise Pozisyona Dayalı Çaprazlama (PBX), Kısmi Planlı Çaprazlama (PMX) ve Tek Noktalı Çaprazlama (SPX)'tir. Kullanıcı tüm bu parametreleri istediği gibi ayarlayabilmekte ve seçebilmektedir.

Tüm bu parametreler ayarlandıktan sonra, "Hesapla" tuşuna basıldığında program, belirlenmiş sayıda tepe ile silindir üzerinde tanımlanmış bu gezgin satıcı problemini, belirlenmiş birey sayısı, iterasyon sayısı, mutasyon yüzdesi ve çaprazlama türü aracılığıyla genetik algoritmayla çözer. Sonuçlarını sağdaki panele yazdırır. Sonuç olarak yazılan değerler ise, tüm iterasyonlar boyunca bulunan en iyi değer (yani en kısa mesafe), bu değeri veren şehir (gen) sıralaması ve en iyi sonucun yer aldığı iterasyondur. Silindir ve noktaların bulunduğu sol paneldeyse, en iyi sonucun grafiksel gösterimi yer almaktadır (Şekil 5.3).



Şekil 5.3 – Örnek bir çözüm

Programda, "Problem_Uret" butonuna tıklandığında, aşağıdaki prosedür çalışmaktadır.

```
Procedure TForm1.Problem_UretClick(Sender: TObject);
 begin
   n:=spinedit1.Value;
   setlength(nokta,n+1,3);
   setlength(sehir_uzaklik_matrisi,n+1,n+1);
   setlength(eniyi_birey,n+1);
   setlength(aradizi,n+1);

   hesapla.Enabled:=False;
   silindir;
   noktalar;
   uzaklik_matrisi_olustur;
   hesapla.Enabled:=True;
 end;
```

Diğer bir deyişle, "Problem_Uret" butonuna tıklandığında, öncelikle gerekli veri dizi ve matrisleri oluşturulmakta, sonra silindir çizdirilmekte ve ardından da rastgele noktalar belirlenerek silindir üzerinde gösterilmektedir. Belirlenen bu noktalar, n boyutlu "nokta" adlı bir dinamik matris veri yapısında tutulmaktadır. Her aşamada bulunan en iyi birey de n boyutlu "en_iyi_birey" adlı dizide

saklanmaktadır. Noktalar arasındaki uzaklık ise, “uzaklik_matrisi_olustur” adlı prosedürde hesaplanıp n x n boyutlu bir matriste saklanmaktadır.

Programda 3-Boyutlu gösterim için izometrik projeksiyon kullanılmış (Egerton and Hall, 1998; Laszlo, 1995) ve de silindirin arka ve ön kısımlarının ayırt edilebilmesi için hem silindirin hem de üzerindeki eğrilerin renkleri açıktan koyuya doğru çizdirilmiştir. Koyu renkte çizdirilen eğriler silindirin arkasında, açık renktekiler ise önünde olduğunu belirtmektedir. Silindir ve üzerindeki noktalar çizdirildikten sonra, “Hesapla” butonuna basıldığında ise aşağıdaki şu prosedür çalışmaktadır.

```
procedure TForm1.HesaplaClick(Sender: TObject);
var
 u1,u2,v1,v2,ustsinir : integer;
 deger : real;
 txt : string;

label
 bas;

begin
  n:=spinedit1.Value;
  m:=spinedit2.Value;
  setlength(ebeveyn,2*m+1,n+1);
  setlength(cocuk,2*m+1,n+1);
  setlength(rankd,2*m+1,4);
  max_iterasyon:=spinedit3.Value;
  mutas:=round(2*m*spinedit4.Value/100);

  gauge1.MinValue:=0;
  gauge1.MaxValue:=max_iterasyon;

  form1.Hesapla.Enabled:=false;
  form1.SpinEdit2.Enabled:=false;
  form1.SpinEdit3.Enabled:=false;
  form1.SpinEdit4.Enabled:=false;
  form1.RadioGroup1.Enabled:=false;
```

```
  form1.Memo1.Lines.Clear;

  iterasyon:=1;
  rastgele_ilk_nesil;
  bas:

    Gauge1.Progress:=iterasyon;
    form1.Label5.Caption:='ITERASYON     =     '     +
inttostr(iterasyon);
    form1.Label5.Refresh;

    toplam_uzakliklar;
    rank_secimi;
    elitizm;
    case radiogroup1.ItemIndex of
       0 : PBX;
       1 : PMX;
       2 : SPX;
     end;
    iki_opt_mutasyon;
    ebeveyn_cocuk_degisim;
    inc(iterasyon);
  if (iterasyon<=max_iterasyon) then goto bas;

  form1.Hesapla.Enabled:=true;
  form1.SpinEdit2.Enabled:=true;
  form1.SpinEdit3.Enabled:=true;
  form1.SpinEdit4.Enabled:=true;
  form1.RadioGroup1.Enabled:=true;
end;
```

Bu prosedürde, öncelikle genetik algoritmada kullanılacak 2m x n boyutlu dinamik ebeveyn ve çocuk matrisleri ile 2m x 3 boyutlu bir rank matrisi (rankd) ve mutasyon yüzdesi belirlenmektedir. Rankd matrisi sıralı seçim (rank selection) için kullanılacak matrisidir. Rankd matrisinin her bir satırındaki birinci sütun bireyin indisini, ikinci sütun bireyin sırasını (rank) ve üçüncü sütun da bireyin hesaplanmış uygunluk değerini saklamaktadır. Bu veri yapıları oluşturulduktan sonra, "rastgele_ilk_nesil" prosedürü çalışmaktadır. Bu prosedürde, şehir

indislerinin (genlerin) rastgele permütasyonlarından (kromozomlardan) oluşan dinamik ebeveyn matrisi (nesli) oluşturulmaktadır. İlk defa oluşan bu matris aynı zamanda genetik algoritmanın ilk neslidir. Artık bundan sonraki tüm nesiller bu ilk nesilden türeyecektir. Ardından, bu ebeveyn neslinin tüm bireyleri "toplam_uzakliklar" prosedürüne gönderilerek her bir bireyin uygunluk değeri, yani silindir üzerinde belirlenen şehirlerin (genlerin) rastgele sırada dolaşılıp bir tam tur atılmasıyla elde edilen toplam uzaklığı hesaplanmakta ve sehir_uzaklik_matrisi'nin içerisi bu hesaplamalarla doldurulmaktadır. Hesaplamalarda silindir üzerindeki jeodezik uzaklığı veren (8) formülü kullanılmıştır. Elde edilen sonuçlara göre bir sıralı seçim (rank selection) matrisi oluşturulmaktadır. Ardından "elitizm" prosedürü çağrılmakta ve ebeveyn neslinin tüm bireylerinin en iyisi bir sonraki nesle aktarılmaktadır. Bu bir sonraki nesil de "cocuk" adlı bir dinamik matristir. Ebeveyn neslindeki tüm bireyler, sıralı seçim matrisi (rankd) yardımıyla, kullanıcının seçtiği çaprazlama türüne göre (PBX, PMX ya da SPX) ikişer ikişer çaprazlanarak "cocuk" matrisine aktarılır. Dolayısıyla, programda kullanılan çaprazlama yüzdesi %100'dür. Oluşan bu çocuk nesli daha sonra mutasyona uğrar. Programda 2-opt mutasyon türü kullanılmaktadır. Mutasyon işlemi "mutasyon" adlı prosedürde, kullanıcının 0 ile 100 arasında belirlemiş olduğu yüzde ile yapılır. Yani, mutasyon işlemi tüm neslin eleman sayısının belirlenen yüzdesi kadarına uygulanmaktadır. Çaprazlanacak ve mutasyona uğrayacak bireyler rastgele seçilmektedir. Son olarak da, çaprazlama ve mutasyon işlemleriyle son haline kavuşmuş bu çocuk nesli, artık ebeveyn nesli olur ve tüm bu işlemler belirlenen iterasyon sayısı kadar devam eder. Program da belirlenen iterasyon sayısına ulaşıldığında sonlanır ve sonuçlar ekranda görünür.

5.2 Deney Sonuçları

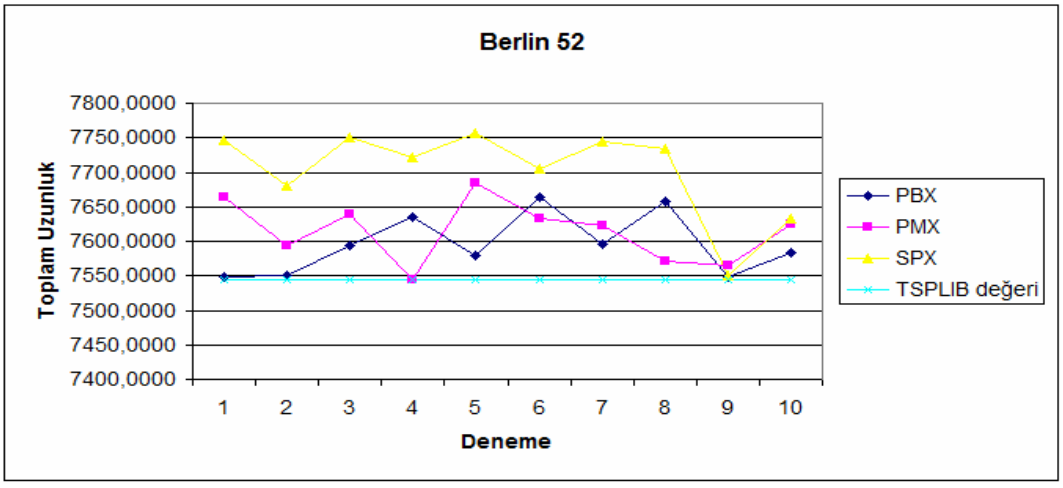
Silindir üzerinde çalışmaya başlamadan önce, programda kullanılan algoritma öncelikle, TSPLIB kütüphanesinden alınan Berlin52 veri seti (Reinelt, 2008) ile test edilmiştir. Programda kullanılan her çaprazlama türü ve 2-opt mutasyon için ayrı ayrı denemeler yapılmıştır. Testlerde kullanılan sabit parametreler ise şöyledir: birey sayısı 250, iterasyon 100 ve mutasyon %3. Her bir çaprazlama türünün en iyi, en kötü, aritmetik ortalama, standart sapma ve hata oranları hesaplanmıştır. Çizelge 5.1 bu sonuçları göstermektedir.

Çizelge 5.1 – Berlin52 test sonuçları

Berlin 52				
	PBX	PMX	SPX	TSPLIB değeri
1	7548,9927	7664,1072	7745,5996	
2	7549,8912	7593,0167	7681,0175	
3	7593,3686	7638,5497	7750,0291	
4	7634,4874	7544,3659	7721,7922	
5	7579,3801	7684,6370	7756,1599	
6	7664,0609	7632,5905	7705,6914	
7	7596,7187	7621,7712	7743,4029	
8	7658,5818	7571,4596	7734,3877	
9	7549,2890	7565,8702	7550,1876	7544,3659
10	7583,7106	7623,7313	7632,7635	
En iyi Değer	7548,9927	7544,3659	7550,1876	
En Kötü Değer	7664,0609	7684,6370	7756,1599	
Aritmetik Ortalama	7595,8481	7614,0099	7702,1031	
Standart Sapma (σ)	43,4385	44,7450	65,4815	
Hata Oranı (Aritmetik Ortalama - Optimum Değer) / Optimum Değer	0,0068	0,0092	0,0209	

TSBLIB kütüphanesindeki verilere göre Berlin'deki 52 kasaba üzerindeki gezgin satıcı turu, Öklid uzaklıklarıyla hesaplandığında optimum olarak 7544,3659 km olmaktadır. Programda kullanılan üç çaprazlama türü (PBX, PMX ve SPX) 10'ar defa belirtilen sabit parametreler ile denenmiştir. PBX ile bulunan en iyi değer gerçek en iyi değerden 4,6268 km fazladır. PMX ile bulunan en iyi değer, gerçek en iyi değer ile aynıdır. SPX ile bulunan en iyi değer ise gerçek en iyi değerden 5,8217 km fazladır. Tüm denemelerin sonunda aritmetik ortalama, standart sapma ve hata oranlarına bakıldığında, algoritmada kullanılan üç çaprazlama türünün de gerçek en iyi değerden çok uzak olmadığı ve hata oranlarının oldukça düşük olduğu görülmektedir (Çizelge 5.2).

Çizelge 5.2 – Berlin 52 test sonuçları grafiği



Test aşamasından sonra program bu kez silindir üzerinde çalıştırılmıştır. Öncelikle silindirin yarıçapı ve yüksekliği sabit tutularak (r = 150 ve h = 360), silindir üzerinde 10, 20, 30, 40, 50, 100 ve 150 tepe (nokta) ile ayrı ayrı denemeler yapılmıştır. Her bir tepe seti için üç adet çaprazlama türü (PBX, PMX ve SPX) hem mutasyonsuz hem de %3 2-opt mutasyon ile 10'ar defa program çalıştırılmıştır. Tüm bu denemelerde birey sayısı 250 ve iterasyon sayısı da 100 olarak sabitlenmiştir. Elde edilen sonuçların hepsi Çizelge 5.3, Çizelge 5.4, Çizelge 5.5, Çizelge 5.6, Çizelge 5.7, Çizelge 5.8, Çizelge 5.9, Çizelge 5.10, Çizelge 5.11, Çizelge 5.12 ve Çizelge 5.13'te gösterilmiştir.

Çizelge 5.3 – Silindir üzerindeki 10 tepeli GSP'nin 3 ayrı çaprazlamaya göre programdaki çözüm sonuçları

10 Tepe							
Mutasyonsuz				Mutasyon = %3			
	PBX	PMX	SPX		PBX	PMX	SPX
1	1153,1441	943,8206	1152,4190	1	942,8596	942,8596	942,8596
2	1152,7974	1047,5995	1204,7588	2	942,8596	942,8596	942,8596
3	943,8310	1100,6721	1126,1690	3	942,8596	942,8596	942,8596
4	1257,0025	943,4713	1074,1936	4	942,8596	942,8596	942,8596
5	1021,7868	1153,1166	1126,6340	5	942,8596	942,8596	942,8596
6	1074,1850	1178,3886	996,3754	6	942,8596	942,8596	942,8596
7	1074,0506	1047,7322	1179,2221	7	942,8596	942,8596	942,8596
8	942,9591	1100,6363	969,9482	8	942,8596	942,8596	942,8596
9	969,9376	1073,7558	1100,6910	9	942,8596	942,8596	942,8596
10	969,3916	943,4405	1100,4238	10	942,8596	942,8596	942,8596
En iyi Değer	942,9591	1178,3886	1204,7588	En iyi Değer	942,8596	942,8596	942,8596
En Kötü Değer	1257,0025	943,4405	969,9482	En Kötü Değer	942,8596	942,8596	942,8596
Aritmetik Ortalama	1055,9086	1053,2633	1103,0835	Aritmetik Ortalama	942,8596	942,8596	942,8596
Standart Sapma (σ)	106,0941	86,0842	74,2485	Standart Sapma (σ)	0,0000	0,0000	0,0000

Çizelge 5.4 – Silindir üzerindeki 20 tepeli GSP'nin 3 ayrı çaprazlamaya göre programdaki çözüm sonuçları

20 Tepe							
Mutasyonsuz				Mutasyon = %3			
	PBX	PMX	SPX		PBX	PMX	SPX
1	3063,8109	2673,5997	2959,8317	1	947,5398	947,5398	947,5398
2	3064,2828	3014,3337	2804,3338	2	947,5398	947,5398	947,5398
3	2384,9206	3273,6257	2698,2457	3	947,5398	947,5398	947,5398
4	2723,9512	3197,6409	3248,0912	4	947,5398	947,5398	947,5398
5	2776,9568	3142,1819	3534,7763	5	947,5398	947,5398	947,5398
6	3142,4788	3273,4186	2566,7797	6	947,5398	947,5398	947,5398
7	2828,2482	3220,8303	2986,2779	7	947,5398	947,5398	947,5398
8	3063,5984	2804,4837	3353,8433	8	947,5398	947,5398	947,5398
9	3063,9942	2985,5345	2801,8155	9	947,5398	947,5398	947,5398
10	3090,1770	2856,1046	3011,6456	10	947,5398	947,5398	947,5398
En iyi Değer	2384,9206	2673,5997	2566,7797	En iyi Değer	947,5398	947,5398	947,5398
En Kötü Değer	3142,4788	3273,6257	3534,7763	En Kötü Değer	947,5398	947,5398	947,5398
Aritmetik Ortalama	2920,2419	3044,1754	2996,5641	Aritmetik Ortalama	947,5398	947,5398	947,5398
Standart Sapma (σ)	239,1966	211,8810	303,8379	Standart Sapma (σ)	0,0000	0,0000	0,0000

Çizelge 5.5 – Silindir üzerindeki 30 tepeli GSP'nin 3 ayrı çaprazlamaya göre programdaki çözüm sonuçları

30 Tepe							
Mutasyonsuz				Mutasyon = %3			
	PBX	PMX	SPX		PBX	PMX	SPX
1	4534,7612	4739,5130	5263,6378	1	952,2459	952,2459	952,2459
2	4901,2214	5054,5529	4923,1388	2	952,2459	952,2459	952,2459
3	5028,4839	4613,5434	4270,2667	3	952,2459	952,2459	952,2459
4	4822,7063	4781,4376	4876,6944	4	952,2459	952,2459	952,2459
5	4582,8419	5265,0976	4767,4457	5	952,2459	952,2459	952,2459
6	4793,7874	5134,2172	4975,7890	6	952,2459	952,2459	952,2459
7	4588,3062	4583,4710	4479,4090	7	952,2459	952,2459	952,2459
8	4662,6470	4766,0692	4975,5525	8	952,2459	952,2459	952,2459
9	5054,1857	4793,1420	4766,6420	9	952,2459	952,2459	952,2459
10	4924,3236	5032,3727	5081,2169	10	952,2459	952,2459	952,2459
En iyi Değer	**4534,7612**	**4583,4710**	**4270,2667**	**En iyi Değer**	**952,2459**	**952,2459**	**952,2459**
En Kötü Değer	**5054,1857**	**5265,0976**	**5263,6378**	**En Kötü Değer**	**952,2459**	**952,2459**	**952,2459**
Aritmetik Ortalama	**4789,3265**	**4876,3417**	**4837,9793**	**Aritmetik Ortalama**	**952,2459**	**952,2459**	**952,2459**
Standart Sapma (σ)	**189,4793**	**229,8268**	**288,2809**	**Standart Sapma (σ)**	**0,0000**	**0,0000**	**0,0000**

Çizelge 5.6 – Silindir üzerindeki 40 tepeli GSP'nin 3 ayrı çaprazlamaya göre programdaki çözüm sonuçları

40 Tepe							
Mutasyonsuz				Mutasyon = %3			
	PBX	PMX	SPX		PBX	PMX	SPX
1	7043,6872	6968,2301	7125,9180	1	965,5677	965,5677	965,5677
2	6991,2704	6993,1019	6602,2653	2	965,5677	965,5677	965,5677
3	7153,6850	6782,1626	6756,1444	3	965,5677	965,5677	965,5677
4	7020,3350	6972,6764	7069,8339	4	965,5677	965,5677	965,5677
5	7043,8224	7514,9350	6967,5199	5	965,5677	965,5677	965,5677
6	6939,6871	7391,5087	6526,2399	6	965,5677	965,5677	965,5677
7	7096,2758	7413,0379	7045,8359	7	965,5677	965,5677	965,5677
8	6814,6727	7546,7036	7462,8942	8	965,5677	965,5677	965,5677
9	6808,5482	6913,0483	6787,8192	9	965,5677	965,5677	965,5677
10	6501,2204	7436,2854	6971,2244	10	965,5677	965,5677	965,5677
En iyi Değer	**6501,2204**	**6782,1626**	**6526,2399**	**En iyi Değer**	**965,5677**	**965,5677**	**965,5677**
En Kötü Değer	**7153,6850**	**7546,7036**	**7462,8942**	**En Kötü Değer**	**965,5677**	**965,5677**	**965,5677**
Aritmetik Ortalama	**6941,3204**	**7193,1690**	**6931,5695**	**Aritmetik Ortalama**	**965,5677**	**965,5677**	**965,5677**
Standart Sapma (σ)	**190,3776**	**290,9643**	**274,7297**	**Standart Sapma (σ)**	**0,0000**	**0,0000**	**0,0000**

Çizelge 5.7 – Silindir üzerindeki 50 tepeli GSP'nin 3 ayrı çaprazlamaya göre programdaki çözüm sonuçları

50 Tepe							
Mutasyonsuz				Mutasyon = %3			
	PBX	PMX	SPX		PBX	PMX	SPX
1	8934,7897	9587,0151	9453,3175	1	969,8746	969,8746	969,8746
2	9191,3621	9036,4137	9089,2804	2	969,8746	969,8746	969,8746
3	8746,2803	9091,7326	9170,5961	3	969,8746	969,8746	969,8746
4	8855,1852	8042,8253	9167,4223	4	969,8746	969,8746	969,8746
5	9479,5813	9534,2768	8878,8651	5	969,8746	969,8746	969,8746
6	9196,0669	9036,1270	8958,5898	6	969,8746	969,8746	969,8746
7	8751,6795	9453,0742	9638,9455	7	969,8746	969,8746	969,8746
8	9115,5359	9642,5223	8805,6197	8	969,8746	969,8746	969,8746
9	9217,2181	9559,3800	8906,1322	9	969,8746	969,8746	969,8746
10	9008,9452	9253,1728	9247,6334	10	969,8746	969,8746	969,8746
En iyi Değer	**8746,2803**	**8042,8253**	**8805,6197**	**En iyi Değer**	**969,8746**	**969,8746**	**969,8746**
En Kötü Değer	**9479,5813**	**9642,5223**	**9638,9455**	**En Kötü Değer**	**969,8746**	**969,8746**	**969,8746**
Aritmetik Ortalama	**9049,6644**	**9223,6540**	**9131,6402**	**Aritmetik Ortalama**	**969,8746**	**969,8746**	**969,8746**
Standart Sapma (σ)	**233,8725**	**477,5133**	**264,5471**	**Standart Sapma (σ)**	**0,0000**	**0,0000**	**0,0000**

Çizelge 5.8 – Silindir üzerindeki 100 tepeli GSP'nin 3 ayrı çaprazlamaya göre programdaki çözüm sonuçları

100 Tepe							
Mutasyonsuz				Mutasyon = %3			
	PBX	PMX	SPX		PBX	PMX	SPX
1	18470,3543	19589,0867	18940,0962	1	1027,5574	1030,0202	1029,4079
2	19298,2817	19930,7970	20087,0525	2	1030,5962	1029,2696	1032,3532
3	19798,2503	20275,0341	19460,8111	3	1027,5025	1029,2115	1031,3464
4	19121,7505	19755,3338	20715,8506	4	1031,9822	1033,3441	1028,2994
5	18883,0831	19331,4576	19491,9902	5	1029,0392	1030,4818	1031,7961
6	18964,5126	20188,0005	20432,1704	6	1030,6470	1032,6327	1029,4450
7	18885,2657	19227,7759	19255,0857	7	1031,9882	1029,0182	1028,5992
8	19274,3939	20061,5342	19933,4196	8	1029,2434	1029,7578	1031,7276
9	18939,1074	19509,7020	19683,4305	9	1029,8591	1030,5333	1028,7962
10	18701,3050	19116,0718	19535,3429	10	1027,7974	1030,6036	1028,6717
En iyi Değer	**18470,3543**	**19116,0718**	**18940,0962**	**En iyi Değer**	**1027,5025**	**1029,0182**	**1028,2994**
En Kötü Değer	**19798,2503**	**20275,0341**	**20715,8506**	**En Kötü Değer**	**1031,9882**	**1033,3441**	**1032,3532**
Aritmetik Ortalama	**19033,6305**	**19698,4794**	**19753,5250**	**Aritmetik Ortalama**	**1029,6213**	**1030,4873**	**1030,0443**
Standart Sapma (σ)	**366,1397**	**408,8097**	**542,3139**	**Standart Sapma (σ)**	**1,6929**	**1,4453**	**1,5730**

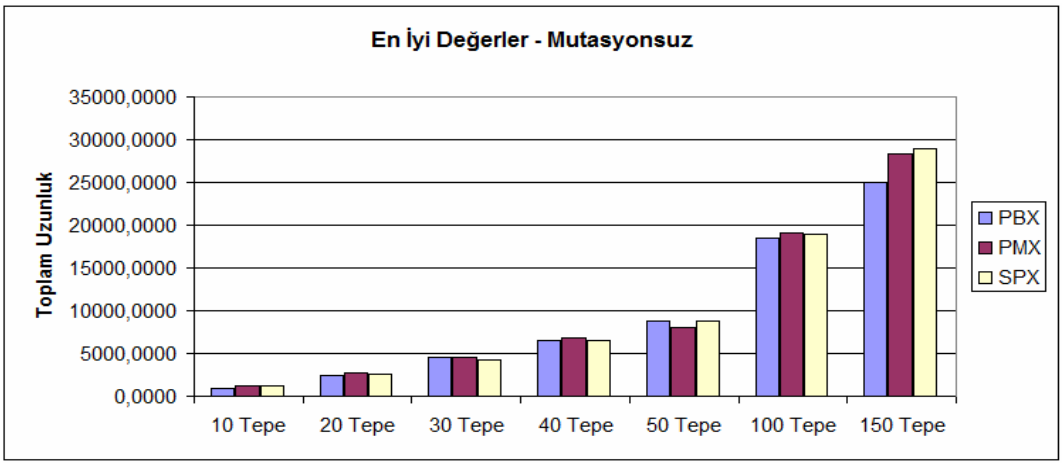
Çizelge 5.9 – Silindir üzerindeki 150 tepeli GSP'nin 3 ayrı çaprazlamaya göre programdaki çözüm sonuçları

150 Tepe							
Mutasyonsuz				Mutasyon = %3			
	PBX	PMX	SPX		PBX	PMX	SPX
1	24979,1550	28832,6311	29677,3817	1	1033,7322	1030,6770	1034,9566
2	28000,7846	28408,8734	29758,8265	2	1035,5143	1032,9299	1030,0671
3	26424,3956	29978,9269	29243,8044	3	1035,4192	1029,4056	1031,8961
4	27254,2937	29048,8782	29034,3334	4	1031,2418	1030,6770	1030,0697
5	28502,6480	29872,2086	30406,7570	5	1031,9595	1030,7480	1029,2009
6	27264,1870	29868,4445	29233,2376	6	1036,1828	1033,1347	1028,7885
7	26955,7222	28733,5171	28959,7689	7	1033,5831	1030,8738	1027,0118
8	25582,3278	29575,2125	29048,1894	8	1034,1859	1030,1238	1031,9528
9	27471,5213	30710,6192	29240,2481	9	1032,9686	1029,4526	1028,8409
10	27469,7207	29156,4400	29242,3785	10	1037,0466	1028,8480	1029,5120
En iyi Değer	**24979,1550**	**28408,8734**	**28959,7689**	**En iyi Değer**	**1031,2418**	**1028,8480**	**1027,0118**
En Kötü Değer	**28502,6480**	**30710,6192**	**30406,7570**	**En Kötü Değer**	**1037,0466**	**1033,1347**	**1034,9566**
Aritmetik Ortalama	**26990,4756**	**29418,5752**	**29384,4926**	**Aritmetik Ortalama**	**1034,1834**	**1030,6870**	**1030,2296**
Standart Sapma (σ)	**1067,8605**	**703,4515**	**443,5601**	**Standart Sapma (σ)**	**1,8599**	**1,4113**	**2,2130**

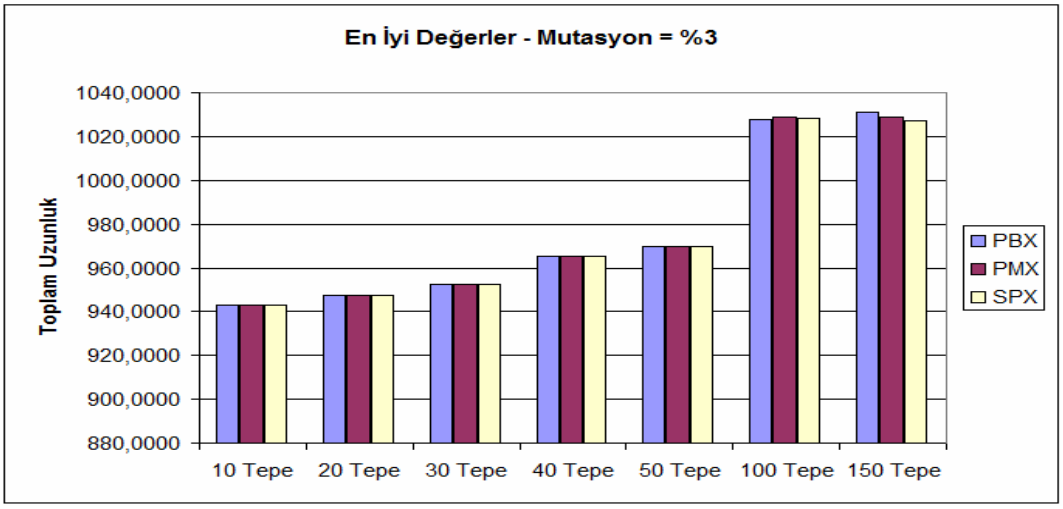
Çizelge 5.10 – Tüm denemeler sonucunda PBX, PMX ve SPX'in en iyi değerleri

EN İYİ DEĞERLER							
Mutasyonsuz				Mutasyon = %3			
	PBX	PMX	SPX		PBX	PMX	SPX
10 Tepe	942,9591	1178,3886	1204,7588	**10 Tepe**	942,8596	942,8596	942,8596
20 Tepe	2384,9206	2673,5997	2566,7797	**20 Tepe**	947,5398	947,5398	947,5398
30 Tepe	4534,7612	4583,4710	4270,2667	**30 Tepe**	952,2459	952,2459	952,2459
40 Tepe	6501,2204	6782,1626	6526,2399	**40 Tepe**	965,5677	965,5677	965,5677
50 Tepe	8746,2803	8042,8253	8805,6197	**50 Tepe**	969,8746	969,8746	969,8746
100 Tepe	18470,3543	19116,0718	18940,0962	**100 Tepe**	1027,5025	1029,0182	1028,2994
150 Tepe	24979,155	28408,8734	28959,7689	**150 Tepe**	1031,2418	1028,848	1027,0118

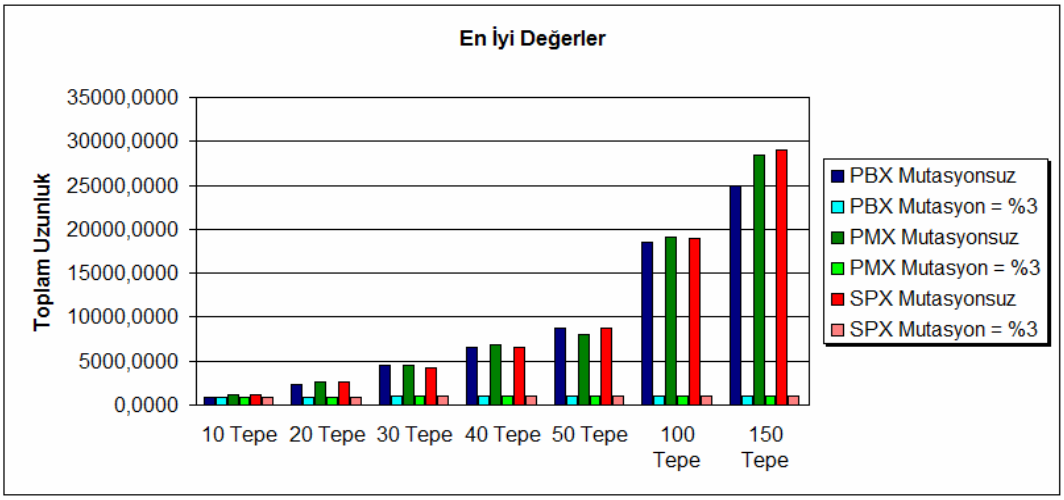
Çizelge 5.11 – Mutasyonsuz PBX, PMX ve SPX'in genel başarı grafiği



Çizelge 5.12 – %3 Mutasyonlu PBX, PMX ve SPX'in genel başarı grafiği



Çizelge 5.13 – Mutasyonsuz ve %3 Mutasyonlu PBX, PMX ve SPX'in kıyaslanması



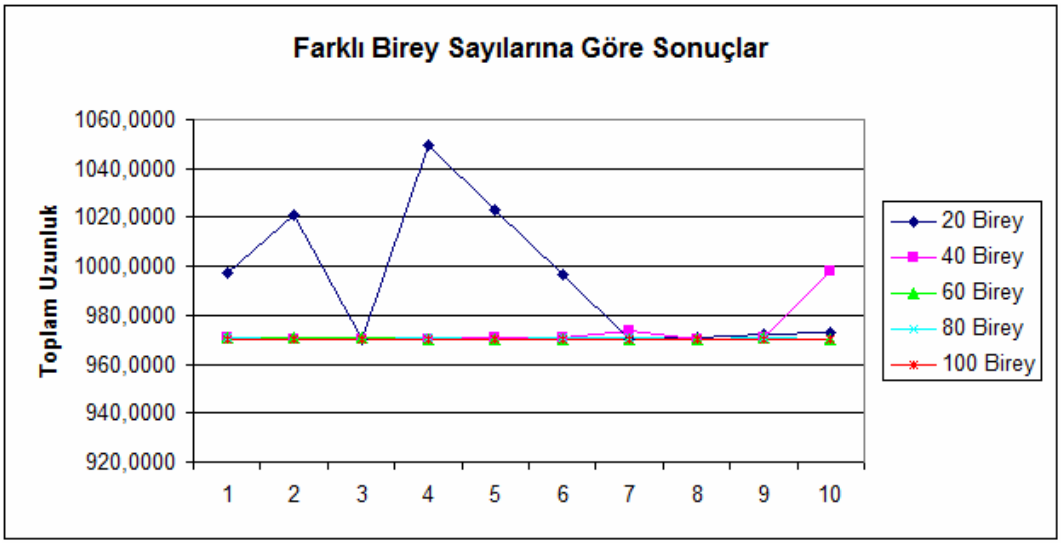
Yapılan tüm bu denemeler sonucunda, çok küçük hata payları ile programın başarılı olduğu görülmektedir. Birey sayısı 250, iterasyon sayısı 100, silindirin yarıçapı 150 ve yüksekliği 360 birimde sabitlenerek, 10, 20, 30, 40, 50, 100 ve 150 tepe (nokta) için yapılan testlerin çoğunda en başarılı çaprazlama türünün PBX, ardından PMX ve son olarak da SPX olduğu gözlenmektedir. Ayrıca, 2-opt mutasyon yöntemi de, programı yerel optimuma takılmaktan kurtarmakta ve ya en iyi ya da en iyiye yakın bir sonucun elde edilmesine olanak sağlamaktadır. Tepe (nokta) sayısı arttıkça her bir çaprazlama türünün ve 2-opt mutasyonun çözüme katkıları net bir şekilde görülmektedir.

Genetik algoritmanın iyi sonuçlar vermesinde etkin olan bir diğer faktör de birey sayısıdır. Birey sayısı az olduğu durumlarda, çaprazlamalardan çok fazla çeşitlilik elde edilemediği için sonuçlar yerel optimuma takılabilmektedir. Ancak, birey sayısı arttıkça, nesil içerisinde bireylerin çaprazlanma kombinasyonları artacak bu da oluşacak çocuk neslinde çeşitliliğin artmasına neden olacaktır. Çizelge 5.14 ve Çizelge 5.15, tepe sayısı 50, iterasyon sayısı 100, mutasyon yüzdesi %3 ve de çaprazlama türü PBX olan problemin bir nesildeki birey sayısı 20, 40, 60, 80 ve 100 alınarak elde edilen program sonuçları göstermektedir.

Çizelge 5.14 – Farklı bireylerin programın çözümüne etkisi

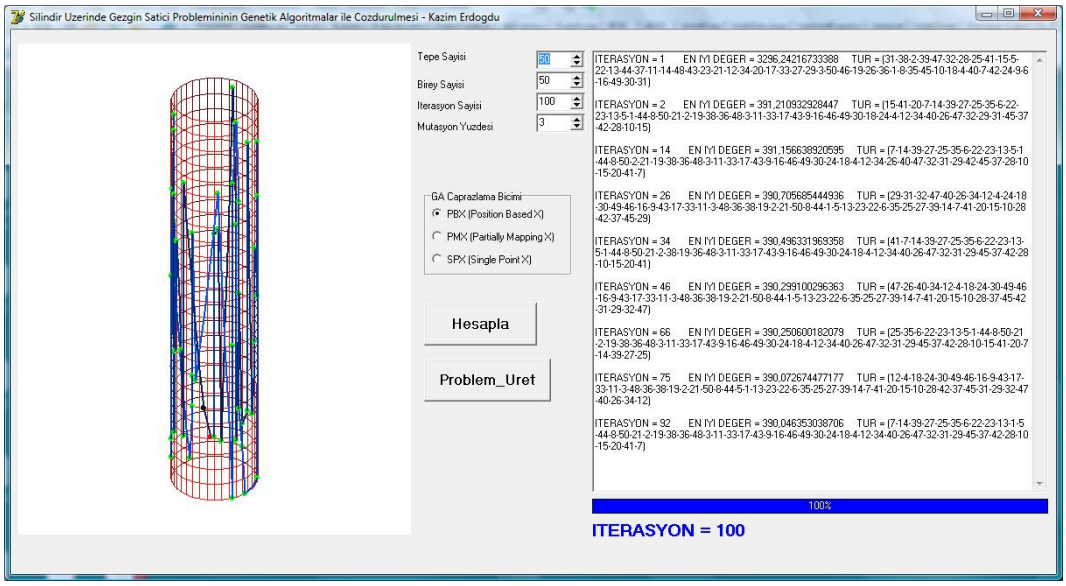
Farklı Birey Sayıları					
	20 Birey	**40 Birey**	**60 Birey**	**80 Birey**	**100 Birey**
1	997,4126	970,5688	970,5028	970,5469	970,2977
2	1020,7963	970,3535	970,6232	970,3581	970,2977
3	970,4424	970,2977	970,5286	970,2977	970,2977
4	1049,8156	970,4424	970,4769	970,5469	970,2977
5	1022,8939	970,5188	970,3581	970,3581	970,2977
6	996,5093	971,1362	970,4769	970,5019	970,2977
7	970,2977	973,5395	970,2977	970,5791	970,2977
8	970,7985	970,3581	970,2977	970,2977	970,2977
9	971,9245	970,7985	970,6451	970,5245	970,2977
10	972,6366	997,6686	970,3581	970,2977	970,2977
En iyi Değer	**970,2977**	**970,2977**	**970,2977**	**970,2977**	**970,2977**
En Kötü Değer	**1049,8156**	**997,6686**	**970,6451**	**970,5791**	**970,2977**
Aritmetik Ortalama	**994,3527**	**973,5682**	**970,4565**	**970,4309**	**970,2977**
Standart Sapma (σ)	**28,4483**	**8,5233**	**0,1253**	**0,1186**	**0,0000**

Çizelge 5.15 – Farklı bireylerle elde edilen sonuçların grafiği

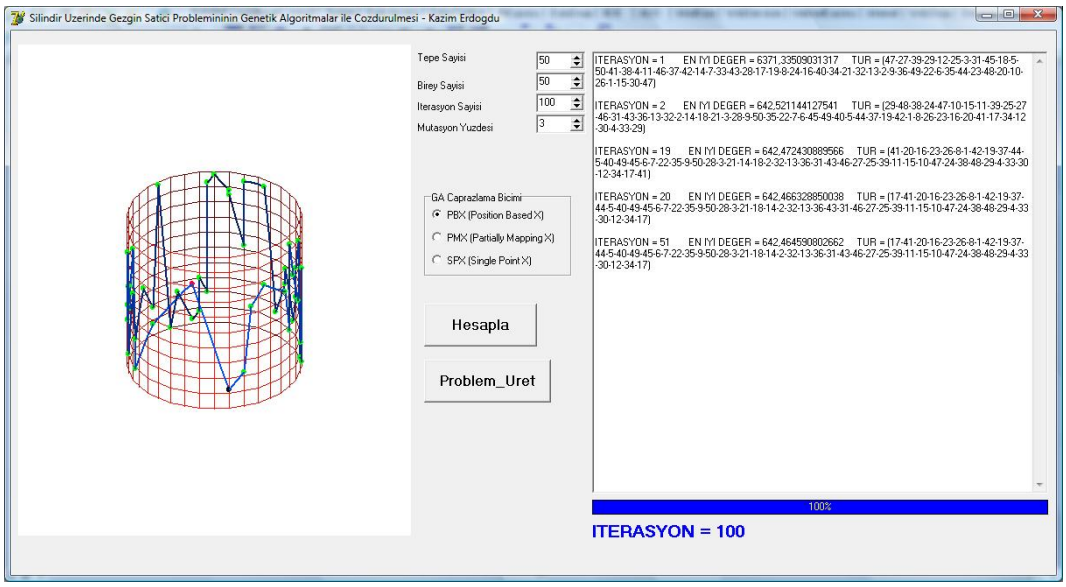


Çizelge 5.14'te de görüldüğü gibi, her ne kadar farklı bireylerle yapılan denemelerin hepsinin en iyi değeri aynı olsa da, standart sapmalar az sayıdaki bireylerden çok sayıdaki bireylere doğru azalmaktadır. Bunun anlamı, birey sayısı azken, elde edilen sonuçlar aritmetik ortalamadan çok farklı olmakta, fakat birey arttıkça bu fark azalmaktadır.

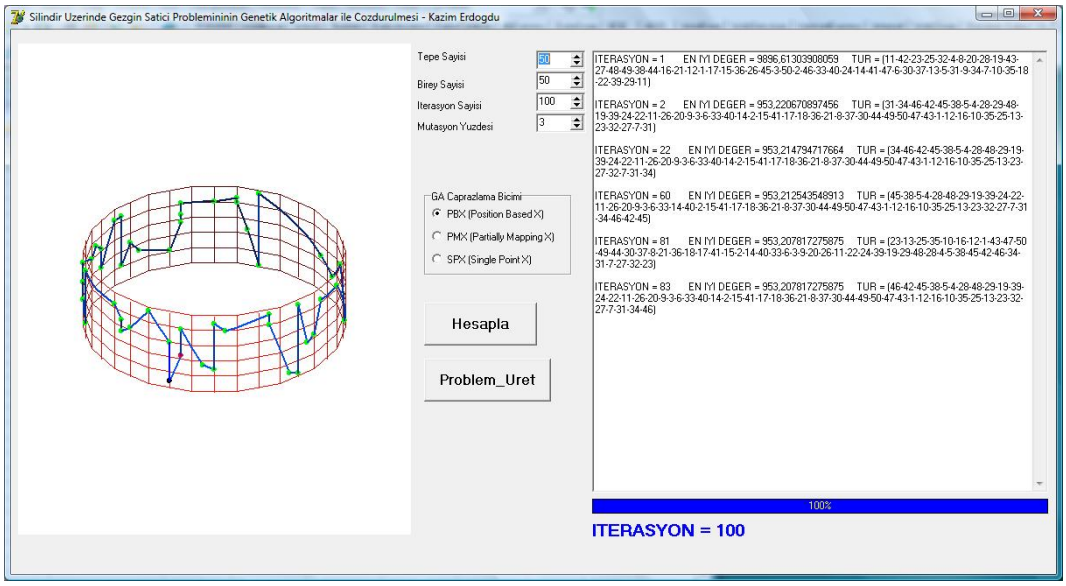
Bu çalışmada son olarak, silindir üzerindeki gezgin satıcı problemi genetik algoritmalar aracılığıyla değişik silindir boyutlarıyla çözdürülmüştür. Tepe (nokta) sayısı 50, birey sayısı 100, iterasyon sayısı 100, mutasyon sayısı %3 sabit alınarak ve yalnızca PBX çaprazlama türü ile, boyutları r = 50 ve h = 500 (Şekil 5.4), r = 100 ve h = 200 (Şekil 5.5), r = 150 ve h = 100 (Şekil 5.6), r = 200 ve h = 60 (Şekil 5.7), r = 200 ve h = 0 (Şekil 5.8), r = 100 ve h = 0 (Şekil 5.9) olan silindirler üzerinde gezgin satıcı problemi 10'ar kez çözdürülmüştür. Çizelge 5.16 elde edilen sonuçları göstermektedir.



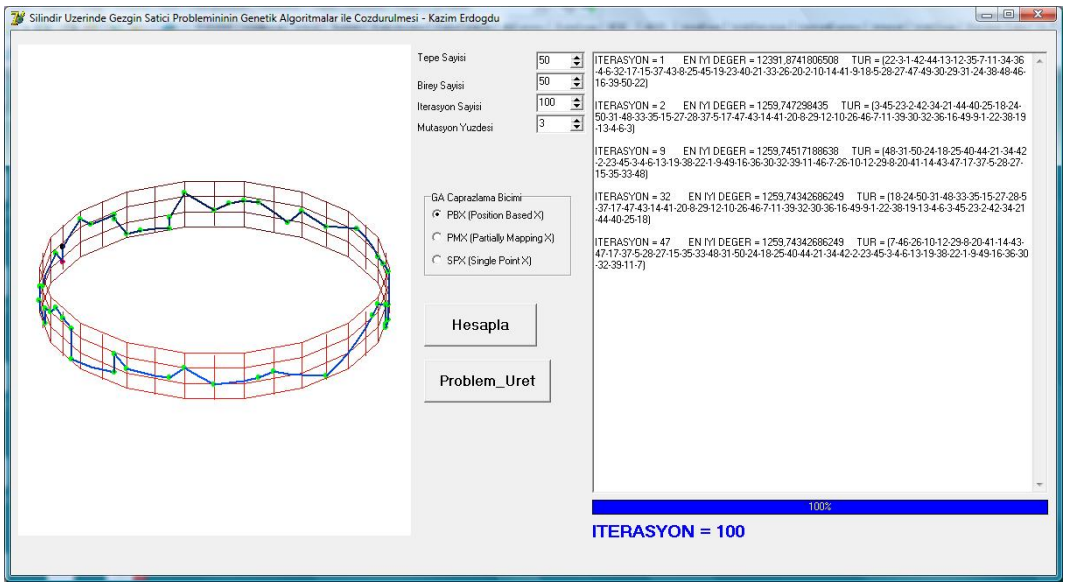
Şekil 5.4 – r=50 ve h=500 olan silindir



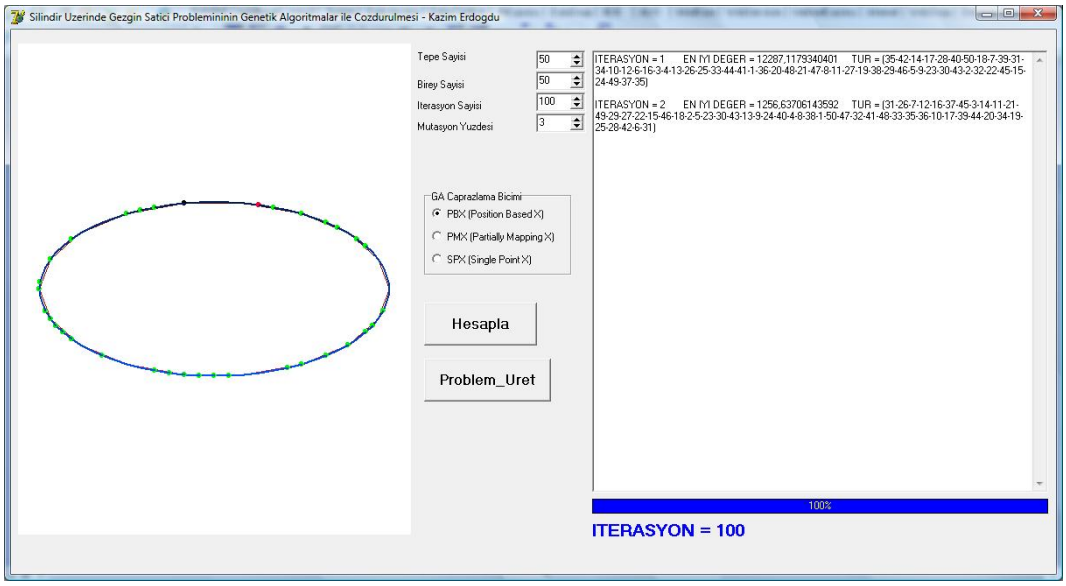
Şekil 5.5 – r=100 ve h=200 olan silindir



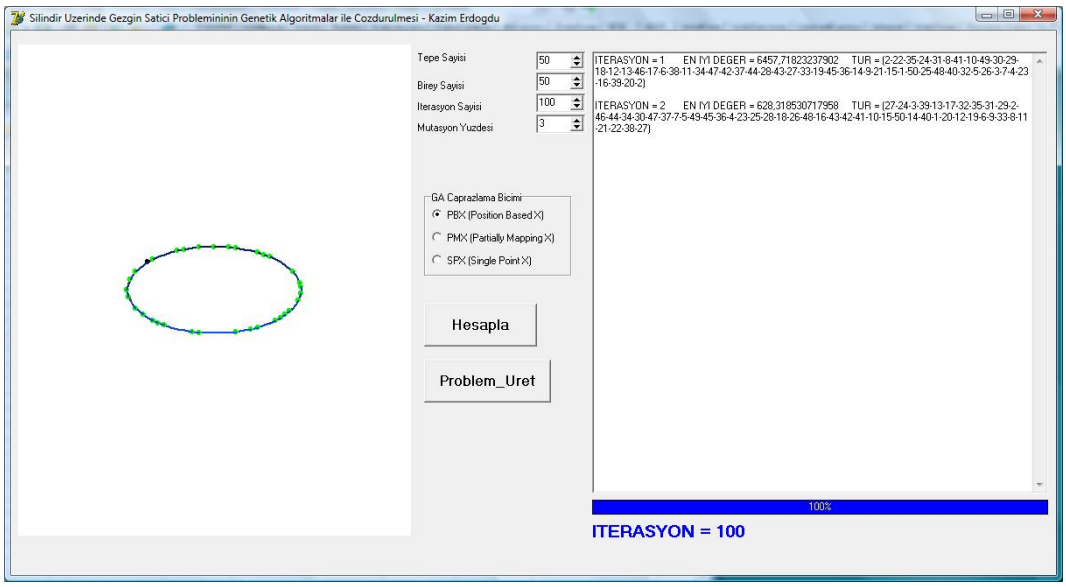
Şekil 5.6 – r=150 ve h=100 olan silindir



Şekil 5.7 – r=200 ve h=60 olan silindir



Şekil 5.8 – r=200 ve h=0 olan silindir



Şekil 5.9 – r=100 ve h=0 olan silindir

Çizelge 5.16 – Farklı silindir boyutları ile yapılan çözümler

	50 Tepe					
	r=50, h=500	r=100, h=200	r=150, h=100	r=200, h=60	r=200, h=0	r=100, h=0
1	390,0046	642,3252	953,2078	1259,7447	1256,6370	628,3185
2	389,7716	642,3321	953,2078	1259,7434	1256,6370	628,3185
3	389,7716	642,3252	953,2078	1259,7434	1256,6370	628,3185
4	389,7716	642,3252	953,2125	1259,7434	1256,6370	628,3185
5	389,7716	642,3252	953,2078	1259,7434	1256,6370	628,3185
6	389,7716	642,3252	953,2078	1259,7434	1256,6370	628,3185
7	389,7716	642,3252	953,2078	1259,7434	1256,6370	628,3185
8	389,7716	642,3252	953,2078	1259,7434	1256,6370	628,3185
9	389,7716	642,3252	953,2078	1259,7434	1256,6370	628,3185
10	389,7716	642,3252	953,2078	1259,7434	1256,6370	628,3185
En iyi Değer	**389,7716**	**642,3252**	**953,2078**	**1259,7434**	**1256,6370**	**628,3185**
En Kötü Değer	**390,0046**	**642,3321**	**953,2125**	**1259,7447**	**1256,6370**	**628,3185**
Aritmetik Ortalama	**389,7949**	**642,3259**	**953,2083**	**1259,7435**	**1256,6370**	**628,3185**
Standart Sapma (σ)	**0,0737**	**0,0022**	**0,0015**	**0,0004**	**0,0000**	**0,0000**

Sonuçlardan da görüldüğü gibi, silindirin boyutlarının değişmesi, sonuçları da değiştirmektedir. Silindirin yarı çapı sabit iken, yüksekliği arttıkça toplam uzunluk da orantılı oranda artmaktadır. Bunun sebebi silindirin yüzey alanının genişlemesi ve dolayısıyla da noktaların daha geniş bir yüzeye yerleşerek aralarındaki mesafenin artmasıdır. Aynı durum silindirin yüksekliği sabit tutulup, yarıçapı değiştiğinde de görülmektedir. Ancak, silindirin yüksekliği ve yarıçapı ne olursa olsun, algoritmanın bulduğu toplam uzaklıklardaki standart sapmalar çok düşüktür. Bu durum da algoritmanın verimli çalıştığını göstermektedir.

6. SONUÇ

Bu çalışmada, regle yüzeyler ve açılabilir yüzeyler tanıtılarak, bu tür yüzeylere ait özel örneklerden bahsedilmiştir. Ayrıca bu yüzeyler üzerindeki en kısa mesafeyi veren jeodezik eğriler incelenerek, açılabilir bir yüzey olan silindir üzerindeki iki noktadan geçen jeodezik eğrinin yay uzunluğunun hesabı yapılmıştır. Ardından, gezgin satıcı problemi açıklanarak matematiksel modeli oluşturulmuştur. Gezgin satıcı probleminin açılabilir yüzeyler üzerinde çözdürmek için de bir yapay zeka tekniği olan genetik algoritmalar anlatılmıştır. Genetik algoritmanın her bir aşaması ve en çok kullanılan bazı çeşitleri açıklanmıştır. En son olarak, tüm bu verilerin ışığında Delphi 7.0 programlama dilinde yazılmış bir programla önce TSPLIB kütüphanesinden alına Berlin52 veri seti için yazılan algoritma denenmiş ve başarılı sonuçlar alındıktan sonra da silindir üzerinde gezgin satıcı problemi bu algoritma ile çözdürülmüştür. Test sonuçlarının ardından algoritmanın performans analizleri yapılmıştır.

Gezgin satıcı problemi birçok yüzey üzerinde çözdürülebilir. Birçok bilimsel çalışmada gezgin satıcı problemi düzlem, küp ve küre üzerinde çözdürülmüştür. Bu çalışmada ise açılabilir bir yüzey olan silindir üzerinde çözdürülmüştür. Bu çalışmada özellikle jeodezik eğrilerin uzaklık hesabında kullanılması, bu tez çalışmasının katkısını oluşturmaktadır. Bu çalışma diğer özel açılabilir yüzeyler üzerinde (örneğin koni, helikoid, vb.) ve ayrıca genel açılabilir yüzeyler üzerinde çalışılarak ve de diğer yapay zeka teknikleri kullanılarak (örneğin karınca kolonisi, yapay sinir ağları, vb.) daha da ileriye görütülebilir.

Yaşadığımız evren değişik geometrik şekiller barındırdığıdan, gezgin satıcı probleminin her zaman düz bir yüzey üzerinde değil, değişik yüzeyler üzerinde düşünülmesi ve bu yüzeylere göre çözüm yollarının bulunması gerekmektedir. Bu çözüm yollarının aynı zamanda mevcut teknoloji ile en verimli ve en kısa sürede yapılması, hem bilim hem de teknoloji alanlarında ilerleme sağlayacaktır. Bu anlamda yapay zeka tekniklerinin kullanılması çözüm yollarının iyileştirilmesini ve hızlandırılmasına neden olacaktır. Dolayısıyla, bu çalışmada yapılmak istenen şey, gezgin satıcı probleminin içerisinde bulunduğu kombinatorik alanını, geometriyi ve yapay zeka alanını birleştirerek özel bir problemin bilgisayar yardımı ile çözümünü araştırmaktır. Bu çalışmanın ve ileride bu alanlarda yapılacak diğer çalışmaların geometri, kombinatorik, yapay zeka, mühendislik ve robot bilimi alanlarına katkıda bulunacağı düşünülmektedir.

7. PROGRAM KODLARI

```
unit UnitYLTez;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, ComCtrls, UnitGA, Spin, Gauges;

type
  TForm1 = class(TForm)
    Hesapla: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Problem_Uret: TButton;
    Cizimalani: TLabel;
    Memo1: TMemo;
    SpinEdit1: TSpinEdit;
    SpinEdit2: TSpinEdit;
    RadioGroup1: TRadioGroup;
    Label4: TLabel;
    SpinEdit3: TSpinEdit;
    Label3: TLabel;
    SpinEdit4: TSpinEdit;
    Gauge1: TGauge;
    Label5: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure HesaplaClick(Sender: TObject);
    procedure Problem_UretClick(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure Problem_UretKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure HesaplaKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure FormKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
```

```
  private
    { Private declarations }
  public
    { Public declarations }
  end;

const
  r = 200;
  artim = 5;
  umin = 0;
  umax = 360;
  vmin = 0;
  vmax = 0;

Type
  TDizi = array [1..3] of real;
  TEkran = array [1..2] of integer;

var
  Form1: TForm1;
  ox, oy :  integer;
  alfa, beta : real;
  sakla : string;
  n,m : integer;
  ebeveyn, cocuk : array of array of integer;
  nokta : array of array of integer;
//  ruletd : array of array of real;
  rankd : array of array of real;
  aradizi : array of integer;
  eniyi_uzaklik : real;
  eniyi_birey : array of integer;
  max_iterasyon, mutas : integer;
  iterasyon : integer;
  sehir_uzaklik_matrisi : array of array of real;
  rank_ust_sinir : integer;

implementation
```

```
{$R *.dfm}

// Ekranda Gosterim Kisminin Baslangici

function isometric(x,y,z : real) : TEkran;
var
 ekr : TEkran;
begin
  ekr[1]:=trunc(x*cos(beta*pi/180)-z*sin(beta*pi/180));

ekr[2]:=trunc(x*sin(alfa*pi/180)*sin(beta*pi/180)+y*cos(alfa*pi/180)+z*sin(alfa
*pi/180)*cos(beta*pi/180));
  Result:=ekr;
end;

procedure silindir;
var
  u,v,rr: integer;
  nokta : TDizi;
  ekran : TEkran;

begin
  form1.Cizimalani.Refresh;
  form1.Cizimalani.Canvas.Pen.Width:=1;
  form1.Cizimalani.Canvas.Pen.Color:=RGB(250,0,0);

  // yatay
  u:=umin; v:=vmin;

  while (v<=vmax) do
   begin
    u:=umin;
    while (u<=umax) do
      begin
        nokta[1]:=r*cos(u*pi/180);
        nokta[2]:=r*sin(u*pi/180);
        nokta[3]:=v;
```

```
      ekran:=isometric(nokta[1],nokta[2],nokta[3]);

     case u of
          0..89   :   form1.Cizimalani.Canvas.Pen.Color:=RGB(165-u,(165-u)   div
10,10);
          90..179  :  form1.Cizimalani.Canvas.Pen.Color:=RGB(75+u-90,(75+u-90)
div 10,10);
          180..269        :        form1.Cizimalani.Canvas.Pen.Color:=RGB(165+u-
180,(165+u-180) div 10,10);
          270..360   :   form1.Cizimalani.Canvas.Pen.Color:=RGB(255-u+270,(255-
u+270) div 10,10);
         end;

      if u=umin then form1.Cizimalani.Canvas.MoveTo(ox + ekran[1], oy -
ekran[2])
             else form1.Cizimalani.Canvas.LineTo(ox + ekran[1], oy - ekran[2]);

      u:=u+20;
    end;

  v:=v+20;
 end;

 // dikey

 u:=umin; v:=vmin;

 while (u<=umax) do
  begin
   v:=vmin;
   while (v<=vmax) do
    begin
      nokta[1]:=r*cos(u*pi/180);
      nokta[2]:=r*sin(u*pi/180);
      nokta[3]:=v;

      ekran:=isometric(nokta[1],nokta[2],nokta[3]);
```

```
      rr:=trunc((umax-umin)/4);

     case u of
         0..89   :   form1.Cizimalani.Canvas.Pen.Color:=RGB(165-u,(165-u)   div
10,10);
         90..179  :  form1.Cizimalani.Canvas.Pen.Color:=RGB(75+u-90,(75+u-90)
div 10,10);
         180..269        :        form1.Cizimalani.Canvas.Pen.Color:=RGB(165+u-
180,(165+u-180) div 10,10);
         270..360   :   form1.Cizimalani.Canvas.Pen.Color:=RGB(255-u+270,(255-
u+270) div 10,10);
        end;

     if v=vmin then form1.Cizimalani.Canvas.MoveTo(ox + ekran[1], oy -
ekran[2])
            else form1.Cizimalani.Canvas.LineTo(ox + ekran[1], oy - ekran[2]);

      v:=v+10;
    end;

   u:=u+10;
 end;

end;

procedure nokta_koy(u1,v1, rnk: integer);
var
 nokta : TDizi;
 ekran : TEkran;

begin

 form1.Cizimalani.Canvas.Pen.Width:=5;

 case rnk of
  1 : form1.Cizimalani.Canvas.Pen.Color:=RGB(0,0,0);
  2 : form1.Cizimalani.Canvas.Pen.Color:=RGB(255,0,50);
```

```
   else form1.Cizimalani.Canvas.Pen.Color:=RGB(0,250,0);
  end;

  nokta[1]:=r*cos(u1*pi/180);
  nokta[2]:=r*sin(u1*pi/180);
  nokta[3]:=v1;

  ekran:=isometric(nokta[1],nokta[2],nokta[3]);

  form1.Cizimalani.Canvas.Ellipse(ox + ekran[1]-1, oy - ekran[2]-1,ox +
ekran[1]+1, oy - ekran[2]+1);

end;

procedure noktalari_goster;
var
 i: integer;
begin
  for i:=1 to n do
   begin
    nokta_koy(nokta[i,1],nokta[i,2],i);
   end;
end;

procedure egri(u1,v1,u2,v2,nokta_renk: integer);
var
  u, v, k, basla_u, bitir_u, basla_v, bitir_v, fark1, fark2, delta_u, delta_v : integer;
  b : real;
  nokta : TDizi;
  ekran : TEkran;

begin

  // nokta
  nokta_koy(u1,v1,nokta_renk);
  nokta_koy(u2,v2,(nokta_renk + 1) mod n);

  // egri
```

```
 form1.Cizimalani.Canvas.Pen.Width:=2;
 form1.Cizimalani.Canvas.Pen.Color:=RGB(0,0,255);

 fark1:=(u1-u2); fark2:=(u2-u1);

 if fark1<0 then fark1:=fark1 + 360;
 if fark2<0 then  fark2:=fark2 + 360;

 if (fark1 < fark2) then
  begin
    basla_u:=u2; bitir_u:=u1; basla_v:=v2; bitir_v:=v1;
    delta_u:=fark1 mod 360;
  end
 else
  begin
    basla_u:=u1; bitir_u:=u2; basla_v:=v1; bitir_v:=v2;
    delta_u:=fark2 mod 360;
  end;

 delta_v:=bitir_v - basla_v;

 if (delta_u mod 360 = 0) then // Eger u1=u2 ise
  begin
      nokta[1]:=r*cos(u1*pi/180);
      nokta[2]:=r*sin(u1*pi/180);
      nokta[3]:=v1;
      ekran:=isometric(nokta[1],nokta[2],nokta[3]);
      form1.Cizimalani.Canvas.MoveTo(ox + ekran[1], oy - ekran[2]);

      nokta[1]:=r*cos(u2*pi/180);
      nokta[2]:=r*sin(u2*pi/180);
      nokta[3]:=v2;

         case u of
          0..89 : form1.Cizimalani.Canvas.Pen.Color:=RGB(5,(165-u) div 3,165-u);
          90..179   :   form1.Cizimalani.Canvas.Pen.Color:=RGB(5,(75+u-90)   div
3,75+u-90);
```

```
          180..269 : form1.Cizimalani.Canvas.Pen.Color:=RGB(5,(165+u-180) div
3,165+u-180);
          270..360 : form1.Cizimalani.Canvas.Pen.Color:=RGB(5,(255-u+270) div
3,255-u+270);
          end;

       ekran:=isometric(nokta[1],nokta[2],nokta[3]);
       form1.Cizimalani.Canvas.LineTo(ox + ekran[1], oy - ekran[2]);
    end
  else // u1 ile u2 farkli ise
  begin

    u:=basla_u;
    b:=(delta_v)/(delta_u);
    k:=0;

    if (bitir_u < basla_u) then bitir_u:=bitir_u + 360;

       while (u<=bitir_u) do
        begin

          nokta[1]:=r*cos(u*pi/180);
          nokta[2]:=r*sin(u*pi/180);
          nokta[3]:=basla_v + b*k;

          k:=k + artim;

          ekran:=isometric(nokta[1],nokta[2],nokta[3]);

          case u of
           0..89 : form1.Cizimalani.Canvas.Pen.Color:=RGB(5,(165-u) div 3,165-u);
           90..179 : form1.Cizimalani.Canvas.Pen.Color:=RGB(5,(75+u-90) div
3,75+u-90);
          180..269 : form1.Cizimalani.Canvas.Pen.Color:=RGB(5,(165+u-180) div
3,165+u-180);
          270..360 : form1.Cizimalani.Canvas.Pen.Color:=RGB(5,(255-u+270) div
3,255-u+270);
          end;
```

```
          if u=basla_u then form1.Cizimalani.Canvas.MoveTo(ox + ekran[1], oy -
ekran[2])
                    else form1.Cizimalani.Canvas.LineTo(ox + ekran[1], oy -
ekran[2]);

           u:=u + artim;
         end;
  end;

end;
// Ekranda Gosterim Kisminin Sonu

// Genetik Algoritma Kismi Baslangici

procedure noktalar;
var
 i,uu,vv : integer;
 txt : string;

begin
   randomize;
  uu:=trunc((umax-umin)/artim);
  vv:=trunc((vmax-vmin)/artim);
  for i:=1 to n do
    begin
      nokta[i,1]:=random(uu)*artim + umin;
      nokta[i,2]:=random(vv)*artim + vmin;
      nokta_koy(nokta[i,1],nokta[i,2],i);
    end;
end;

function jeodezik_uzaklik(u1,v1,u2,v2 : integer) : real;
var
 d,k : real;
 delta_u, delta_v, fark1, fark2 : integer;
begin
  delta_v:=v1-v2;
```

```
 fark1:=abs(u1-u2);
 if (fark1<360-fark1) then delta_u:=fark1
                  else delta_u:=360-fark1;
 if (delta_u = 0) then d:=abs(v1-v2)*pi/180
                else
                 begin
                  k:=delta_v / delta_u;
                  d:=sqrt(r*r + k*k)*abs(delta_u)*pi/180;
                 end;
 Result:=d;
end;

procedure uzaklik_matrisi_olustur;
var
 i,j : integer;

begin
 for i:=1 to n do
   for j:=1 to n do
    begin
     sehir_uzaklik_matrisi[i,j]:=0;
    end;

 for i:=1 to n do
   for j:=i+1 to n do
    begin

sehir_uzaklik_matrisi[i,j]:=jeodezik_uzaklik(nokta[i,1],nokta[i,2],nokta[j,1],nokta
[j,2]);
     sehir_uzaklik_matrisi[j,i]:=sehir_uzaklik_matrisi[i,j];
    end;

//   form1.Caption:='(1,1)=' + floattostr(sehir_uzaklik_matrisi[1,1]) + ', (2,5)=' +
floattostr(sehir_uzaklik_matrisi[2,5]) +', (5,2)=' +
floattostr(sehir_uzaklik_matrisi[5,2]);

end;
```

```
function uygunluk_degeri : real; // bir ebeveyndeki TSP turunun maliyetini
hesapliyor
var
 i : integer;
 sonuc : real;
begin
  sonuc:=0;
  for i:=1 to n-1 do
    sonuc:=sonuc + sehir_uzaklik_matrisi[aradizi[i],aradizi[i+1]];
  sonuc:=sonuc + sehir_uzaklik_matrisi[aradizi[n],aradizi[1]];
 Result:=sonuc;
end;

procedure toplam_uzakliklar; //tum nesillerdeki uygunluk degerlerinin ve rulet
matrisinin hesaplanmasi
var
 i,j : integer;
 toplam : real;

begin

  for i:=1 to 2*m do
   begin
     toplam:=0;

     for j:=1 to n do
       aradizi[j]:= ebeveyn[i,j];

     toplam:=uygunluk_degeri;

     rankd[i,1]:=i;
     rankd[i,2]:=0;
     rankd[i,3]:=toplam;
   end;

end;
```

```
procedure rastgele_ilk_nesil;
var
 i, j, k, indis, sayac : integer;
 kontrol : array of integer;
 txt : string;
begin

  setlength(kontrol,n+1);

  for i:=1 to 2*m do
    for j:=1 to n do
      ebeveyn[i,j]:=0;

  randomize;

  for i:=1 to 2*m do
   begin

    for j:=1 to n do
      kontrol[j]:=j;

    sayac:=n;

    for j:=1 to n do
      begin
       indis:=random(sayac)+1;
       ebeveyn[i,j]:=kontrol[indis];
       for k:=indis to sayac-1 do
         kontrol[k]:=kontrol[k+1];
       sayac:=sayac-1;
      end;
   end;

end;
```

```
procedure rank_secimi;
var
 i,j,dongu : integer;
 sakla, enk : real;
 txt : string;

begin
 // rankd'yi en kotuden en iyiye dogru (uygunluk degeri en buyukten en kucuge
dogru) Siraliyor
  for i:=1 to 2*m-1 do
   for j:=i+1 to 2*m do
     begin
       if rankd[i,3]<rankd[j,3] then
        begin
         sakla:=rankd[i,1];
         rankd[i,1]:=rankd[j,1];
         rankd[j,1]:=sakla;

         sakla:=rankd[i,3];
         rankd[i,3]:=rankd[j,3];
         rankd[j,3]:=sakla;
        end;
     end;

  if (iterasyon=1) or (rankd[2*m,3] < eniyi_uzaklik) then
    begin
     eniyi_uzaklik:=rankd[2*m,3];
     txt:=' (';
     for i:=1 to n do
      begin
       eniyi_birey[i]:=ebeveyn[trunc(rankd[2*m,1]),i];
       txt:=txt + inttostr(eniyi_birey[i]) + '-';
      end;
     txt:=txt + inttostr(eniyi_birey[1]);
     txt:=txt + ')';
       txt:='ITERASYON = ' + inttostr(iterasyon) + '          EN IYI DEGER = ' +
floattostr(eniyi_uzaklik) + '     TUR =' + txt;
       form1.Memo1.Lines.Add(txt);
```

```
        form1.Memo1.Lines.Add('');

        // en iyi permutasyonun cizdirilmesi
        form1.Cizimalani.Refresh;
        silindir;
        //noktalari_goster;
        for dongu:=1 to n-1 do

egri(nokta[eniyi_birey[dongu],1],nokta[eniyi_birey[dongu],2],nokta[eniyi_birey[d
ongu+1],1],nokta[eniyi_birey[dongu+1],2],dongu);

egri(nokta[eniyi_birey[n],1],nokta[eniyi_birey[n],2],nokta[eniyi_birey[1],1],nokta
[eniyi_birey[1],2],n);

    end;

// rank tekerleginde kucukten buyuge dogru dilim (+1) verilmesi
  rankd[1,2]:=1;
  for i:=2 to 2*m do
   begin
     rankd[i,2]:=rankd[i-1,2]+i;
   end;

   rank_ust_sinir:=trunc(rankd[2*m,2]);

  // form1.Caption:=inttostr(rank_ust_sinir);

end;


procedure elitizm;
var
 i : integer;
begin
   for i:=1 to n do // en iyi bireyin cocuk nesle aktarilmasi
    cocuk[1,i]:=ebeveyn[trunc(rankd[2*m,1]),i];
end;
```

```
procedure PBX;
var
 i,j,k,t,e1,e2,cn1,cn2, rsayi, basla, bitir : integer;
 cocuk1,cocuk2, kontrol : array of integer;
 sonuc1, sonuc2 : real;
 txt : string;

label
  etiket, etiket2;

begin

  setlength(cocuk1,n+1);
  setlength(cocuk2,n+1);
  setlength(kontrol,n+1);

 randomize;

// caprazlanacak ebeveynlerin secilmesi

  for i:=2 to 2*m do
    begin

      rsayi:=random(rank_ust_sinir)+1;    // rank tekerinden bir sayi secme

      j:=0;
      while (rankd[j,2]<=rsayi) and (j<2*m) do
        j:=j+1;
      e1:=trunc(rankd[j,1]);    // ebeveyn 1

   etiket:             // ebeveyn 2
      rsayi:=random(rank_ust_sinir)+1;

      j:=0;
      while (rankd[j,2]<=rsayi) and (j<2*m) do
        j:=j+1;
```

```
      e2:=trunc(rankd[j,1]);
      if (e1=e2) then goto etiket;

// caprazlanacak genlerin secimi
 for j:=1 to n do
   kontrol[j]:=0;

 j:=0;
 repeat
   repeat
     k:=random(n)+1;
   until kontrol[k]=0;
   kontrol[k]:=1;
   j:=j+1;
 until j=n div 2;

// caprazlama basliyor

 for j:=1 to n do
 begin
   cocuk1[j]:=0;
   cocuk2[j]:=0;
    if kontrol[j]=1 then
      begin
       cocuk1[j]:=ebeveyn[e1,j];
       cocuk2[j]:=ebeveyn[e2,j];
      end;
 end;

 for j:=1 to n do
  begin
   t:=0;
   for k:=1 to n do
    if ebeveyn[e2,j]=cocuk1[k] then t:=1;
   if t=0 then
    begin
     t:=1;
     while (kontrol[t]<>0) do
```

```
      inc(t);
     kontrol[t]:=2;
     cocuk1[t]:=ebeveyn[e2,j];
    end;
  end;

 for j:=1 to n do
    if kontrol[j]=2 then kontrol[j]:=0;

 for j:=1 to n do
  begin
   t:=0;
   for k:=1 to n do
    if cocuk2[k]=ebeveyn[e1,j] then t:=1;
   if t=0 then
    begin
     t:=1;
     while (kontrol[t]<>0) do
       inc(t);
     kontrol[t]:=2;
     cocuk2[t]:=ebeveyn[e1,j];
    end;
  end;


// iyi olan cocugun (cocuk1,cocuk2) cocuk matrisine aktarilmasi
    for j:=1 to n do
      aradizi[j]:=cocuk1[j];
    sonuc1:=uygunluk_degeri;

    for j:=1 to n do
      aradizi[j]:=cocuk2[j];
    sonuc2:=uygunluk_degeri;

    if sonuc1<sonuc2 then
         for j:=1 to n do
          cocuk[i,j]:=cocuk1[j]
        else
```

```
        for j:=1 to n do
         cocuk[i,j]:=cocuk2[j];
   end;
end;


procedure PMX;
var
 i,j,k,e1,e2,cn1,cn2, rsayi, basla, bitir : integer;
 cocuk1,cocuk2,permutasyon_sol,permutasyon_sag : array of integer;
 sonuc1, sonuc2 : real;
 txt : string;

label
  etiket, etiket2;

begin

  setlength(cocuk1,n+1);
  setlength(cocuk2,n+1);
  setlength(permutasyon_sol,n+1);
  setlength(permutasyon_sag,n+1);

  randomize;

// caprazlanacak ebeveynlerin secilmesi

  for i:=2 to 2*m do
    begin
      rsayi:=random(rank_ust_sinir)+1;    // rulet tekeri icin yuzde belirleme
      j:=1;
      while (rankd[j,2]<rsayi) and (j<n) do
        j:=j+1;
      e1:=trunc(rankd[j,1]);    // ebeveyn 1

   etiket:              // ebeveyn 2
      rsayi:=random(rank_ust_sinir)+1;
      j:=1;
```

```
    while (rankd[j,2]<rsayi) and (j<n) do
      j:=j+1;
    e2:=trunc(rankd[j,1]);
    if (e1=e2) then goto etiket;

 // rastgele 2 caprazlama noktasinin secilmesi

    cn1:=random(n-2)+2; // caprazlama noktasi 1

  etiket2:    // caprazlama noktasi 2
    cn2:=random(n-2)+2;
    if (cn1=cn2) then goto etiket2;

    if (cn1<cn2) then
      begin
        basla:=cn1;
        bitir:=cn2;
      end
      else begin
            basla:=cn2;
            bitir:=cn1;
          end;

// caprazlama basliyor

    for j:=1 to n do
      begin
        permutasyon_sol[j]:=0;
        permutasyon_sag[j]:=0;
      end;

    for j:=basla to bitir do  // orta kisimlarin caprazlanmasi
      begin
        cocuk1[j]:=ebeveyn[e1,j];
        cocuk2[j]:=ebeveyn[e2,j];
        permutasyon_sol[cocuk1[j]]:=cocuk2[j];
        permutasyon_sag[cocuk2[j]]:=cocuk1[j];
      end;
```

```
    for j:=1 to basla-1 do  // baslangic kisminin caprazlanmasi
      begin
        k:=ebeveyn[e2,j];
        while (permutasyon_sol[k] <> 0) do
          k:=permutasyon_sol[k];
        cocuk1[j]:=k;

        k:=ebeveyn[e1,j];
        while (permutasyon_sag[k] <> 0) do
          k:=permutasyon_sag[k];
        cocuk2[j]:=k;
      end;

    for j:=bitir+1 to n do  // son kisminin caprazlanmasi
      begin
        k:=ebeveyn[e2,j];
        while (permutasyon_sol[k] <> 0) do
          k:=permutasyon_sol[k];
        cocuk1[j]:=k;

        k:=ebeveyn[e1,j];
        while (permutasyon_sag[k] <> 0) do
          k:=permutasyon_sag[k];
        cocuk2[j]:=k;
      end;

// iyi olan cocugun (cocuk1,cocuk2) cocuk matrisine aktarilmasi
    for j:=1 to n do
      aradizi[j]:=cocuk1[j];
    sonuc1:=uygunluk_degeri;

    for j:=1 to n do
      aradizi[j]:=cocuk2[j];
    sonuc2:=uygunluk_degeri;

    if sonuc1<sonuc2 then
        for j:=1 to n do
```

```
        cocuk[i,j]:=cocuk1[j]
       else
        for j:=1 to n do
         cocuk[i,j]:=cocuk2[j];
  end;
end;

procedure SPX;
var
 i,j,k,t,kont, e1,e2,cn, rsayi, basla, bitir : integer;
 cocuk1,cocuk2 : array of integer;
 sonuc1, sonuc2 : real;
 txt : string;

label
  etiket, etiket2;

begin

  setlength(cocuk1,n+1);
  setlength(cocuk2,n+1);

  randomize;

// caprazlanacak ebeveynlerin secilmesi

  for i:=2 to 2*m do
    begin
      rsayi:=random(rank_ust_sinir)+1;    // rulet tekeri icin yuzde belirleme
      j:=1;
      while (rankd[j,2]<rsayi) and (j<n) do
        j:=j+1;
      e1:=trunc(rankd[j,1]);    // ebeveyn 1

   etiket:             // ebeveyn 2
      rsayi:=random(rank_ust_sinir)+1;
      j:=1;
      while (rankd[j,2]<rsayi) and (j<n) do
```

```
      j:=j+1;
     e2:=trunc(rankd[j,1]);
     if (e1=e2) then goto etiket;

// caprazlama basliyor

   cn:=random(n-2)+2;

   for j:=1 to n do
     begin
       cocuk1[j]:=0;
       cocuk2[j]:=0;
     end;

      // caprazlama noktasina kadar olan ilk kismin ebeveynlerden cocuklara aktarilmasi
   for j:=1 to cn do
     begin
       cocuk1[j]:=ebeveyn[e1,j];
       cocuk2[j]:=ebeveyn[e2,j];
     end;

      // caprazlama noktasindan sonraki kismin ebeveynlerden cocuklara aktarilmasi
   j:=cn+1;
    for k:=1 to n do
      begin
        kont:=0; t:=0;
        repeat
          t:=t+1;
          if ebeveyn[e2,k]=cocuk1[t] then kont:=1;
        until (kont=1) or (t=n);
        if kont=0 then
          begin
           cocuk1[j]:=ebeveyn[e2,k];
           j:=j+1;
          end;
      end;
```

```
   j:=cn+1;
    for k:=1 to n do
      begin
        kont:=0; t:=0;
        repeat
          t:=t+1;
          if ebeveyn[e1,k]=cocuk2[t] then kont:=1;
        until (kont=1) or (t=n);
        if kont=0 then
          begin
           cocuk2[j]:=ebeveyn[e1,k];
           j:=j+1;
          end;
      end;

 // iyi olan cocugun (cocuk1,cocuk2) cocuk matrisine aktarilmasi
    for j:=1 to n do
      aradizi[j]:=cocuk1[j];
    sonuc1:=uygunluk_degeri;

    for j:=1 to n do
      aradizi[j]:=cocuk2[j];
    sonuc2:=uygunluk_degeri;

    if sonuc1<sonuc2 then
        for j:=1 to n do
         cocuk[i,j]:=cocuk1[j]
       else
        for j:=1 to n do
         cocuk[i,j]:=cocuk2[j];
   end;
end;

procedure iki_opt_mutasyon;
var
 mut,sakla,ind,k1,k2,k3,k4,ilk,son,ustsinir : integer;
 deger1, deger2 : real;
```

```
 anahtar : boolean;

label
  etiket;

begin
  randomize;
  if mutas<>0 then
   begin

    etiket:

     for ind:=1 to mutas do
       begin
        mut:=random(2*m)+1;

        for k3:=1 to n do
          aradizi[k3]:=cocuk[mut,k3];

        deger1:=uygunluk_degeri; // mutasyondan onceki uygunluk degeri

        for k1:=1 to n do  // ikiser ikiser genlere 2-opt uygulaniyor
          for k2:=1 to n do
            begin

              for k3:=1 to n do
                aradizi[k3]:=cocuk[mut,k3];

              if k1<>k2 then
                begin
                  sakla:=aradizi[k1];
                  aradizi[k1]:=aradizi[k2];
                  aradizi[k2]:=sakla;

                  if k1>k2 then begin  // secilen 2 genden kucuk olana 'ilk' buyuk
olana 'son' adi veriliyor
                              ilk:=k2+1;
                              son:=k1-1;
```

```
                    end
                else begin
                    ilk:=k1+1;
                    son:=k2-1;
                   end;

            ustsinir:=trunc((son-ilk)/2);

            for k3:=0 to ustsinir do // ilk ve son arasindaki genler tersten
yaziliyor
              begin
               sakla:=aradizi[ilk+k3];
               aradizi[ilk+k3]:=aradizi[son-k3];
               aradizi[son-k3]:=sakla;
              end;

            deger2:=uygunluk_degeri; // mutasyondan sonraki uygunluk degeri

            if deger2<deger1 then begin
                              deger1:=deger2;
                              for k4:=1 to n do
                                cocuk[mut,k4]:=aradizi[k4];
                              goto etiket;
                             end;
          end;
        end;
    end;

  end;
end;

procedure ebeveyn_cocuk_degisim;
var
 i,j : integer;
begin
  for i:=1 to 2*m do
   for j:=1 to n do
    ebeveyn[i,j]:=cocuk[i,j];
```

```
end;

// Genetik Algoritma Kismi Sonu

// Formdaki procedurler

procedure TForm1.FormCreate(Sender: TObject);
begin

 Form1.Caption:='Silindir Uzerinde Gezgin Satici Problemininin Genetik Algoritmalar ile Cozdurulmesi - Kazim Erdogdu';

 hesapla.Enabled:=False;

 alfa:= 60;
 beta:= 0;

 n:=spinedit1.Value;
 m:=spinedit2.Value;
 max_iterasyon:=spinedit3.Value;
 mutas:=round(2*m*spinedit4.Value/100);

 setlength(ebeveyn,2*m+1,n+1);
 setlength(cocuk,2*m+1,n+1);
 setlength(nokta,n+1,3);
 setlength(sehir_uzaklik_matrisi,n+1,n+1);
 //setlength(ruletd,2*m+1,4);
 setlength(rankd,2*m+1,4);
 setlength(eniyi_birey,n+1);
 setlength(aradizi,n+1);

 form1.Cizimalani.Refresh;

 ox:=trunc(Cizimalani.Width/2);
 oy:=trunc(Cizimalani.Height/2);
 Cizimalani.Canvas.MoveTo(ox,oy);
 Cizimalani.Canvas.LineTo(Cizimalani.Width,oy);
 Cizimalani.Canvas.MoveTo(ox,0);
```

```
 Cizimalani.Canvas.LineTo(ox,oy);
 silindir;

end;

procedure TForm1.Button5Click(Sender: TObject);
begin
  form1.Cizimalani.Refresh;
  silindir;
end;

procedure TForm1.HesaplaClick(Sender: TObject);
var
 u1,u2,v1,v2,ustsinir : integer;
 deger : real;
 txt : string;

label
 bas;

begin

  n:=spinedit1.Value;
  m:=spinedit2.Value;
  setlength(ebeveyn,2*m+1,n+1);
  setlength(cocuk,2*m+1,n+1);
  //setlength(ruletd,2*m+1,4);
  setlength(rankd,2*m+1,4);
  max_iterasyon:=spinedit3.Value;
  mutas:=round(2*m*spinedit4.Value/100);

  gauge1.MinValue:=0;
  gauge1.MaxValue:=max_iterasyon;

  form1.Hesapla.Enabled:=false;
  form1.SpinEdit2.Enabled:=false;
  form1.SpinEdit3.Enabled:=false;
  form1.SpinEdit4.Enabled:=false;
```

```
  form1.RadioGroup1.Enabled:=false;

  form1.Memo1.Lines.Clear;

  iterasyon:=1;
  rastgele_ilk_nesil;
  bas:

    Gauge1.Progress:=iterasyon;
    form1.Label5.Caption:='ITERASYON = ' + inttostr(iterasyon);
    form1.Label5.Refresh;

    toplam_uzakliklar;
    //rulet;
    rank_secimi;
    elitizm;
    case radiogroup1.ItemIndex of
      0 : PBX;
      1 : PMX;
      2 : SPX;
     end;
    //mutasyon;
    iki_opt_mutasyon;
    ebeveyn_cocuk_degisim;
    inc(iterasyon);
  if (iterasyon<=max_iterasyon) then goto bas;

  form1.Hesapla.Enabled:=true;
  form1.SpinEdit2.Enabled:=true;
  form1.SpinEdit3.Enabled:=true;
  form1.SpinEdit4.Enabled:=true;
  form1.RadioGroup1.Enabled:=true;

end;

procedure TForm1.Problem_UretClick(Sender: TObject);
begin
```

```
  n:=spinedit1.Value;
  setlength(nokta,n+1,3);
  setlength(sehir_uzaklik_matrisi,n+1,n+1);
  setlength(eniyi_birey,n+1);
  setlength(aradizi,n+1);

  hesapla.Enabled:=False;
  silindir;
  noktalar;
  uzaklik_matrisi_olustur;
  hesapla.Enabled:=True;
end;

procedure TForm1.FormResize(Sender: TObject);
var
 dongu : integer;

begin
 // en iyi permutasyonun cizdirilmesi
       form1.Cizimalani.Refresh;
       silindir;
       //noktalari_goster;
       for dongu:=1 to n-1 do

egri(nokta[eniyi_birey[dongu],1],nokta[eniyi_birey[dongu],2],nokta[eniyi_birey[d
ongu+1],1],nokta[eniyi_birey[dongu+1],2],dongu);

egri(nokta[eniyi_birey[n],1],nokta[eniyi_birey[n],2],nokta[eniyi_birey[1],1],nokta
[eniyi_birey[1],2],n);

end;

procedure TForm1.Problem_UretKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if key=27 then halt;
end;
```

```
procedure TForm1.HesaplaKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if key=27 then halt;
end;

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if key=27 then halt;
end;

end.
```

KAYNAKLAR DİZİNİ

Alshina C. 2000, "A geometric journey into space: Geodesic in a cylinder", http://www.upc.edu/ea-smi/personal/claudi/web3d/english/geocil.htm (Erişim tarihi: 6 Haziran 2011)

Biran, L., 1970, Diferensiyel Geometri Dersleri, İstanbul Üniversitesi Fen Fakültesi Basım Evi, İstanbul, 132s.

Egerton, P. A. and Hall W. S., 1998, Computer Graphics: Mathematical First Steps, Prentice Hall., New Jersey, 344p.

Engin O. ve Fığlalı A., 2002, Akış tipi çizelgeleme problemlerinin genetik algoritma yardımı ile çözümünde uygun çaprazlama operatörünün belirlenmesi, *Doğuş Üniversitesi Dergisi*, 6:27-35.

Genome News Network 2003a, "What types of genome maps are there?" http://www.genomenewsnetwork.org/resources/whats_a_genome/Chp3_2.shtml (Erişim tarihi 15 Haziran 2011)

Genome News Network 2003b, "Chromosomes: How many chromosomes do organisms have?" http://www.genomenewsnetwork.org/resources/whats_a_genome/Chp1_2_1.shtml (Erişim tarihi 15 Haziran 2011)

Gutin G. and Punnen A. P., 2004, The Traveling Salesman Problem and Its Variations (Combinatorial Optimization), Kluwer Academic, New York, 848p.

Hoffman K. and Padberg M., "Traveling Salesman Problem" http://iris.gmu.edu/~khoffman/papers/trav_salesman.html (Erişim tarihi:10 Haziran 2011)

Kühnel W., 2005, Differential Geometry: Curves–Surfaces–Manifolds (2nd Edition), American Mathematical Society, Road Island, 380p.

Larranaga P., Kuijpers C. M. H., Murgha R. H., Inza I. and Dizdarevic S., 1999, Genetic algorithms for the traveling salesman problem: a review of representations and operators, *Artificial Intelligence Review*, 13:129-170.

Laszlo M. J., 1995, Computational Geometry and Computer Graphics in C++, Prentice Hall, New Jersey, 266p.

Lawler E. L., Lenstra J.K., Rinnooy Kan A. H. G. and Shmoys D. B., 1985, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, Wiley-Interscience, New York, 476p.

KAYNAKLAR DİZİNİ (devam)

Manfredo P. D. C., 1976, Differential Geometry Of Curves And Surfaces, Prentice-Hall, New Jersey, 503p.

Mitchell M., 1996, An Introduction to Genetic Algorithms, MIT Press, Cambridge, 224p.

Nabiyev V. V., 2005, Yapay Zeka: Problemler – Yöntemler – Algoritmalar, Seçkin Yayıncılık, Ankara, 764s.

Özkan B., 2008, Dinamik Gezgin Satıcı Probleminin Çözümü İçin Bir Eniyileme Kütüphanesinin Tasarımı ve Görsel Yazılım Geliştirme Ortamı ile Birlikte Gerçekleştirimi, Yüksek Lisans Tezi, Ege Üniversitesi, 106s. (Yayınlanmamış)

Reinelt G. 2008, "TSPLIB – Traveling Salesman Library", http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ (Erişim Tarihi: 15 Haziran 2011)

Rhino3DE, "Developable Surfaces", http://www.rhino3.de/design/modeling/developable/ (Erişim Tarihi: 10 Haziran 2011)

Russell S. and Norvig P., 2002, Artificial Intelligence: A Modern Approach (2nd Edition), Prentice Hall, New Jersey, 1132p.

Uğur A., 2008, Path planning on a cuboid using genetic algorithms, *Information Sciences*, 178(16):3275-3287.

Uğur A., Korukoğlu S., Çalıskan A., Cinsdikici M. and Alp A., 2009, Genetic algorithm based solution for Tsp on a sphere, *Mathematical and Computational Applications*, 14(3):219-228.

Woeginger G. J., "Exact Algorithms for NP-Hard Problems: A Survey", http://www.win.tue.nl/~gwoegi/papers/exact.pdf (Erişim tarihi 15 Haziran 2011)

Yıldırım, C., 2001, Bilimin Öncüleri, Tübitak Popüler Bilim Yayınları, Ankara, 226s.

ÖZGEÇMİŞ

Kazım ERDOĞDU, 23.12.1981 tarihinde Ankara'da doğdu. 1987-1988 yılında ilkokul birinci sınıfı Ankara'da okudu. 1988-1992 yılları arasında ise ilkokulunu İzmir'de okudu. Ortaokulunu 1992-1995 yıllarında İzmir'de Şehit Üsteğemen Sadullah Sever İlköğretim Okulu'nda ve liseyi 1995-1999 yıllarında İzmir Atatürk Lisesi'nde okudu. 1999 yılında Ege Üniversitesi Fen Fakültesi Matematik Bölümü'nü kazandı ve 2003 yılında mezun oldu. 2009 yılında Ege Üniversitesi Fen Bilimleri Enstitüsü'ne bağlı olarak Matematik Anabilimdalı Geometri Bilimdalı'nda yüksek lisans eğitimine başladı.