**T.C.**
**PAMUKKALE ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**
**ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI**

# SİMETRİK GEZGİN SATICI PROBLEMİ İÇİN YENİ BİR META-SEZGİSEL: KÖR FARE ALGORİTMASI

**YÜKSEK LİSANS TEZİ**

**TEVFİK YILDIRIM**

**DENİZLİ, ŞUBAT - 2014**

**T.C.**
**PAMUKKALE ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**
**ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI**

# SİMETRİK GEZGİN SATICI PROBLEMİ İÇİN YENİ BİR META-SEZGİSEL: KÖR FARE ALGORİTMASI

**YÜKSEK LİSANS TEZİ**

**TEVFİK YILDIRIM**

**DENİZLİ, ŞUBAT - 2014**

# KABUL VE ONAY SAYFASI

**TEVFİK YILDIRIM** tarafından hazırlanan "**SİMETRİK GEZGİN SATICI PROBLEMİ İÇİN YENİ BİR META-SEZGİSEL: KÖR FARE ALGORİTMASI**" adlı tez çalışmasının savunma sınavı 13.01.2014 tarihinde yapılmış olup aşağıda verilen jüri tarafından oy birliği ile Pamukkale Üniversitesi Fen Bilimleri Enstitüsü ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

| Jüri Üyeleri | İmza |
|---|---|
| Danışman<br>Yrd. Doç. Dr. Özcan MUTLU | ..... |
| Üye<br>Yrd. Doç. Dr. Kenan KARAGÜL | ..... |
| Üye<br>Yrd. Doç. Dr. Can Berk KALAYCI | ..... |

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 12/02/2014 tarih ve ..08/26..... sayılı kararıyla onaylanmıştır..

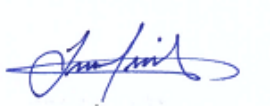
.....

Prof. Dr. Nuri KOLSUZ

Fen Bilimleri Enstitüsü Müdürü

**Bu tez çalışması Pamukkale Üniversitesi Bilimsel Araştırma Projeleri Koordinasyon Birimi tarafından 2012FBE057nolu proje ile desteklenmiştir.**

**Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğini beyan ederim.**



**TEVFİK YILDIRIM**

# ÖZET


**SİMETRİK GEZGİN SATICI PROBLEMİ İÇİN YENİ BİR META-SEZGİSEL: KÖR FARE ALGORİTMASI**
**YÜKSEK LİSANS TEZİ**
**TEVFİK YILDIRIM**
**PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ**
**ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI**
**(TEZ DANIŞMANI: YRD. DOÇ. DR. ÖZCAN MUTLU)**
**DENİZLİ, ŞUBAT - 2014**

Gezgin Satıcı Problemi (GSP), başlangıç ve bitiş şehirleri aynı olan ve her şehrin sadece bir kez ziyaret edildiği minimum mesafeli turu bulma problemidir. Problemin tanımı kolay olmasına rağmen şehir sayısı arttığında problemin çözümü zorlaşmaktadır. Bu nedenle Gezgin Satıcı Problemi üzerinde, en çok çalışılan kombinatoryel en iyileme problemlerinden biridir.

Gezgin satıcı problemi NP-Tam kategorisine dahildir ve kesin matematiksel yöntemler ile çözüme ulaşmak problem büyüklüğü arttıkça imkânsızlaşmaktadır. Bu sebeple makul çözüm sürelerine sahip çok sayıda sezgisel yöntem geliştirilmiştir. Doğadan esinlenen sezgisel algoritmalar oldukça fazla çalışılan yöntemler arasında yer almaktadır.

Bu çalışmada, kör farelerin doğadaki davranışlarından ve yaşadıkları tünellerde karşılaştıkları engelleri geçme stratejilerinden esinlenilerek geliştirilen bir meta-sezgisel arama yöntemi önerilmiştir. Ayrıca yöntemin kendisine özgü bir mutasyon operatörü bulunmaktadır. Yönteme ilişkin parametrelerin en uygun değerleri Taguchi L32 tasarımı ile belirlenmiştir. Önerilen yöntem farklı boyutlardaki simetrik GSP test problemleri için uygulanmış ve sonuçlar bilinen en iyi değerlerle kıyaslanmıştır. Bu yöntemle gerçekleştirilen deneylerden elde edilen sonuçlara dayanarak yöntemin umut verici olduğunu söyleyebiliriz.



**ANAHTAR KELİMELER:** Gezgin Satıcı Problemi, Kombinatoryel Eniyileme, Meta sezgiseller, Kör Fare Algoritması

# ABSTRACT

**A NEW META-HEURISTIC FOR SYMMETRIC TRAVELING SALESMAN PROBLEM: BLIND MOLE-RAT ALGORITHM**

**MSC THESIS**

**TEVFİK YILDIRIM**

**PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE**

**INDUSTRIAL ENGINEERING**

**(SUPERVISOR: ASSIST. PROF. DR. ÖZCAN MUTLU)**

**DENİZLİ, FEBRUARY 2014**

Traveling Salesman Problem (TSP) can be described as the problem of finding a minimum distance tour of *n* cities, beginning and ending at the same city and that each city are visited only once. Although the definition of the problem is quite simple; it is considerably difficult to solve the problem when the numbers of cities increase. Hence, Traveling Salesman Problem is one of the most widely studied combinatorial optimization problem.

TSP is the member of NP-Complete class and the solution of TSP is impossible with exact methods when the size of problem increase. Hence, a large number of heuristics which have reasonable computation time were developed. In developed heuristic algorithms inspired by nature are quite popular.

In this study, a new meta-heuristic called “blind mole-rat algorithm” was proposed. This heuristic inspired by behaviors and strategy of blind mole-rats in the nature. Additionally algorithm has specific mutation operator. The optimum values of parameters of algorithm were set with Taguchi L32 design. Then algorithm was applied to different size of symmetric TSP problems. The results of application benchmarked with known optimum solutions. Through the experiments carried out with this method, promising results were obtained.

**KEYWORDS:** Traveling Salesman Problem, Combinatorial Optimization, Meta-heuristics, Blind mole-rat algorithm.

# İÇİNDEKİLER

**Sayfa**

# ŞEKİL LİSTESİ

**Sayfa**

## TABLO LİSTESİ

**Sayfa**

# SEMBOL LİSTESİ

| | |
|---|---|
| **GSP** | Gezgin Satıcı Problemi |
| **SGSP** | Simetrik Gezgin Satıcı Problemi |
| **AGSP** | Asimetrik Gezgin Satıcı Problemi |
| **ÇGSP** | Çoklu Gezgin Satıcı Problemi |
| $\mathbf{c_{ij}}$ | *i'*nci düğüm ile *j*'nci düğüm arasındaki mesafe/maliyet |
| $\mathbf{d_{ij}}$ | Kör fare algoritmasındaki *i'*nci düğüm ile *j*'nci düğüm arasındaki uzaklık |
| $\mathbf{s_{ij}}$ | *i*'nci düğümden*j*'nci düğüme gelen sinyal değeri |
| $\mathbf{R_{ij}}$ | *i'*nci düğüm ile *j*'nci düğüm arasındaki sinyal iletim katsayısı |
| $\mathbf{S(m)_{ijk}}$ | *m*'inci fare için o an bulunduğu *i*'nci düğüme *j*'nci düğümden ve *k*'ıncı düğümden*j*'nci düğüme gelen toplam sinyal değeri |
| $\mathbf{\Delta s_{ij}}$ | *i-j* yolunun sinyal kayıp değeri |
| **w** | Sinyal kayıp katsayısı |
| **v** | Sabit sinyal ekleme değeri |
| **q** | Geçişte arama yapılacak en yakın komşu sayısı |
| **SAO** | Bir yol üzerinde iletilebilecek maksimum sinyal değerini |
| **AP** | Atama Problemi |
| **LK** | Lin&Kernighan |
| **TB** | Tavlama Benzetimi |
| **YA** | Yasaklı Arama |
| **GA** | Genetik Algoritma |
| **YSA** | Yapay Sinir Ağı |
| **KKO** | Karınca Kolonisi Optimizasyonu |
| **PSO** | Parçacık Sürü Optimizasyonu |
| **KFA** | Kör fare algoritması |
| **RS** | Rassal Sayı |
| **DFJ** | Dantzig Fulkerson Johnson |
| **P** | Polynomial |
| **NP** | Non-deterministic polynomial |
| **S/N** | Signal to noise |

# ÖNSÖZ

Yüksek lisans tez çalışmalarım süresince değerli zamanını benden esirgemeyen, bilgi ve tecrübesi ile bana yön gösteren değerli danışman hocam Yrd. Doç. Dr. Özcan MUTLU'ya desteklerinden dolayı teşekkürü bir borç bilirim.

Tez uygulama yazılımının java dilinde geliştirilmesi sürecinde, değerli desteklerini esirgemeyen Murat Kemal BAYGÜN'e gönülden teşekkür ederim. Tez çalışmamın her safhasında düşünceleri ile katkıda bulunan, çalışma arkadaşım Mehmet Ulaş KOYUNCUOĞLU'na samimi teşekkürlerimi sunarım.

Bugünlere gelmemde en büyük pay sahibi olan annem Zeynep YILDIRIM ve babam Selahattin YILDIRIM'a, ayrıca bana her konuda destek olan eşim Fatma Filiz YILDIRIM'a, neşe kaynağım biricik oğlum Yalın YILDIRIM'a, dua ve desteklerini eksik etmeyen tüm geniş aileme teşekkür ederim.

**TevfikYILDIRIM**

# 1. GİRİŞ

Optimizasyon, alternatifler içinde en iyisini seçmek olarak tanımlanabilir. Matematiksel olarak, bir fonksiyonu maksimum/minimum noktasını belirleyen değerlerin elde edilmesidir.

Kombinatoryel optimizasyon, ayrık çözüm uzayına sahip problemlere en iyi çözümü arayan metotlara verilen genel isimdir [120]. Bu yöntemle amaç fonksiyonunu eniyileyecek şekilde kesikli karar değişkenlerinin değerlerinin bulunması hedeflenmektedir. Karmaşıklık teorisine göre kombinatoryel problemler, P ile temsil edilen polinom türde problemler ve NP ile temsil edilen deterministik olmayan polinom problemler şeklinde sınıflandırılabilir [22]. Karmaşıklık, verilen girdilere karşılık kaynak kullanım miktarının nasıl değiştiği ile ilgilenir. Bir problemin girdi boyutu *n*'in büyümesine karşılık ilgili algoritmanın çalışma zamanı *N*'in büyüme oranı problemin zorluk derecesini belirler. Karmaşıklık çoğu durumda zaman ile bağıntılıdır ve O ile ifade edilir. Örnek olarak $O(n^2)$ ifadesi, çalışma zamanının girdi boyutunun karesi ile doğru orantılı arttığını gösterir. Alan Turing tarafından geliştirilen Turing makinesi, karmaşıklık teorisini açıklarken kullanılan en önemli kavramlardan birisidir. Turing'in sanal makinesi (belirlenimci), aynı girdiler için her zaman aynı sonuçları üretmektedir [127]. P sınıfı problemler Turing makinesinde polinom zamanda çözülebilirler. NP sınıfındaki problemler ise, polinom zamanda ancak belirlenimci olmayan Turing makinesi ile çözülebilirler.

NP-Tam sınıfındaki problemler NP problemlerin bir alt kümesini oluştururlar ve hızlı çözüm sağlayan algoritmaları yoktur. Bu sınıfta yer alan problemlere ait algoritmalar için, problem boyutunun büyümesi ile çözüm sürelerinin daha hızlı büyüdüğü ortaya çıkmıştır. NP-Tam kümesindeki problemler çözüm süreleri açısından en az üstel zaman gerektiren problemlerdir.

Gezgin satıcı problemi (GSP) NP-Tam sınıfındaki en yaygın bilinen problemlerden birisidir.

GSP'de amaç, *n* şehir bulunan bir ağda, her şehire sadece bir kez uğrayan ve başlangıç şehrine geri dönen en kısa mesafeli turu bulmaktır. Şehirler arasındaki mesafe $d_{ij}$ ile ifade edildiğinde eğer $d_{ij} = d_{ji}$ ise simetrik GSP, $d_{ij} \neq d_{ji}$ ise asimetrik (simetrik olmayan) GSP olarak adlandırılır. Şehir sayısı arttıkça problemin çözümü oldukça zor bir hal almaktadır. GSP genellikle NP-Tam problem olarak sınıflandırılır [2]. Literatürde GSP üzerinde en çok çalışılan kombinatoryel en iyileme problemlerinden birisidir.

GSP'nin çözümü için çok sayıda yöntem geliştirilmiştir. Özellikle son yıllarda bilgisayar teknolojisinde önemli gelişmelerin olması ile birlikte problemin çözümü için önerilen sezgisel yöntemlerin sayısında bir artış meydana gelmiştir. Sezgisel yöntemler, en iyi çözümü garanti etmemekle birlikte makul sürelerde en iyi/en iyiye yakın çözümler bulabilen yöntemlerdir. GSP'nin çözümünde en sık kullanılan yöntemlerden bazıları şunlardır: Genetik Algoritmalar, Karınca Kolonisi Optimizasyonu, Parçacık Sürü Optimizasyonu, Yapay Arı Kolonisi Optimizasyonu. Matematiksel yöntemlere dayanan sezgiseller; Diferansiyel Gelişim, Tabu Arama ve Scatter(dağılım) Araması algoritmalarıdır [2].

Bu tezde simetrik GSP için Kör Fare Algoritması (KFA) yeni bir sezgisel olarak önerilmiştir. Algoritma, kör farelerin yer altındaki tünellerde ilerleme ve engelleri geçme davranışlarına dayanmaktadır. Yöntemin kendine özgü bir mutasyon operatörü tasarlanmıştır. Bu operatör kromozom yapısındaki belirli iki genin sabitlenmesi ve sabitlenmiş genlerin aralığındaki ilk sağda ve solda kalan genlerin karşılıklı yer değiştirmesi esasına dayanmaktadır. Geliştirilen yöntem GSP için veri sunan internet sitelerinden alınan veri setlerine uygulanmış ve bilinen en iyi sonuçlarla karşılaştırılmıştır.

Çalışmanın ikinci bölümünde, GSP'nin tanımı, matematiksel modeli, sınıflandırılması, GSP uygulamaları, hesaplama karmaşıklığı ve tarihsel gelişimi hakkında bilgiler verilmiştir.

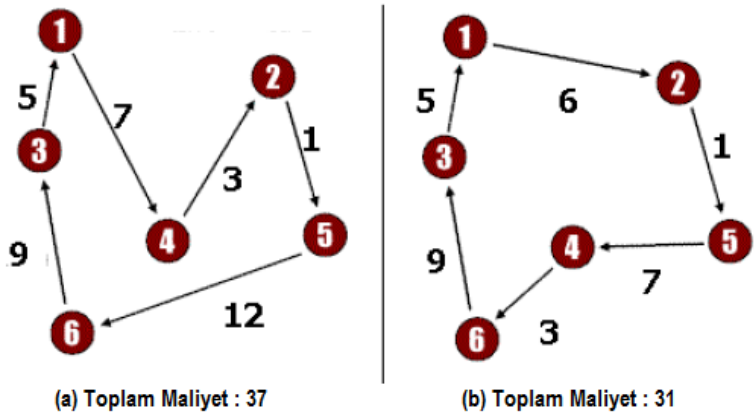
Üçüncü bölümde, GSP'nin çözümünde kullanılan yöntemlere yer verilmiştir. Çözüm yöntemleri kesin ve yaklaşık algoritmalar başlıklarında incelenmiştir. Yaklaşık algoritmalar tur kurucu sezgiseller, tur geliştirici sezgiseller ve karma sezgiseller olmak üzere üç alt bölümde anlatılmıştır.

Dördüncü bölümde, kör farelerin doğadaki yaşamları ve davranışları anlatılmıştır. Doğadaki kör farelerin yaşam ve davranışlarını temel alan algoritma tasarımı ortaya konulmuştur. Tasarlanan KFA, literatürde yer alan simetrik GSP test problemleri ile denenmiştir.

Sonuç ve öneriler kısmında, KFA'nın etkinliği incelenerek, gelecek çalışmalar ve uygulama imkanları anlatılmıştır.

# 2. GEZGİN SATICI PROBLEMİ

Gezgin satıcı problemi (GSP), başlangıç ve bitiş şehirleri aynı olan ve her bir şehrin sadece bir kez ziyaret edildiği minimum mesafeli turu bulma problemidir [1, 2, 5, 6, 9]. GSP graf teorisinde, minimum maliyetli Hamilton çevriminin bulunması olarak tanımlanır [21]. *V* düğümler kümesini, *A* kenarlar kümesini, $C=(c_{ij})$ *A* kümesi ile ilişkilendirilmiş mesafe ya da maliyet matrisini temsil etmek üzere *G= (V,A)* grafında, her bir düğüme sadece bir kez uğrayan en kısa mesafeli kapalı yol Hamilton çevrimi olarak tanımlanır [3]. Şekil 2.1'de farklı maliyetlere sahip Hamilton çevrimi örnekleri gösterilmiştir.




**Şekil 2.1: Hamilton Çevrimi Örnekleri**

## 2.1 Gezgin Satıcı Probleminin Matematiksel Modeli

GSP için farklı yaklaşımları temel alan birçok matematiksel model geliştirilmiştir. Simetrik Gezgin Satıcı Problemi (SGSP)'nin matematiksel modeli aşağıdaki şekilde ifade edilebilir [114]. Bu modelde $x_{ij}$ karar değişkenidir.

$$x_{ij} = \begin{cases} 1, & i'\text{den } j'\text{ye gidildiğinde} \\ 0, & \text{diğer durumlarda} \end{cases}$$

$c_{ij}$ = i′nci düğüm ile j′nci düğüm arasındaki maliyet/mesafe

$$\text{Minimum} \qquad \sum_{i \neq j} c_{ij} x_{ij} \tag{2.1}$$

Kısıtları altında;

$$\sum_{j=1} x_{ji} = 1, \quad (1 \leq i \leq n) \tag{2.2}$$

$$\sum_{i=1} x_{ij} = 1, \quad (1 \leq i \leq n) \tag{2.3}$$

$$u_i - u_j + (n-1)x_{ij} \leq n-2 \qquad (2 \leq i, j \leq n; i \neq j) \tag{2.4}$$

$$1 \leq u_i \leq n-1 \qquad (2 \leq i \leq n) \tag{2.5}$$

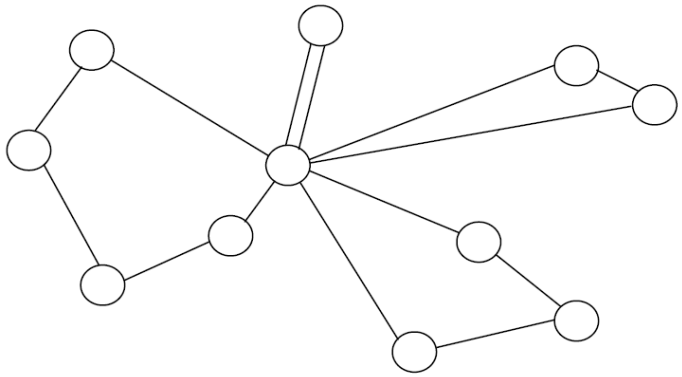
$$x_{ij} \in \{0,1\}, \quad (1 \leq i, j \leq n; i \neq j)$$

Denklem (2.1) tur mesafesini minimize eden amaç fonksiyonudur. Denklem (2.2) ve (2.3) her bir şehire bir kez girileceğini ve her şehirden bir kez çıkılacağını garanti eden kısıtlardır. Denklem (2.4) ve (2.5) oluşabilecek alt turları engellemek için kullanılan kısıtlardır. Bu model 1960 yılında Miller ve arkadaşları tarafından geliştirilmiş olup 1991 yılında Desrochers ve Laporte [115] tarafından yeni alt tur eleme kısıtları eklenmiştir.

### 2.2 Gezgin Satıcı Probleminin Sınıflandırılması

GSP, simetrik [2, 4, 7], asimetrik [2, 4, 7] ve çoklu [2] olmak üzere sınıflandırılır [2, 4, 7]. Şehirler arasındaki maliyeti/mesafeyi $c_{ij}$ olarak tanımlarsak eğer $c_{ij} = c_{ji}$ ise problem Simetrik Gezgin Satıcı Problemi (SGSP) olur. Eğer $c_{ij} \neq$

$c_{ji}$ ise problem Asimetrik Gezgin Satıcı Problemi (AGSP) olur. Şehirler arasındaki mesafe/maliyet üçgen eşitsizliğini ($c_{ij} + c_{jk} \geq c_{ik}$) sağlanmalıdır [2, 3,7, 8, 14, 19, 20]. Çoklu Gezgin Satıcı Problemi (ÇGSP)'de ise başlangıç şehrinde konumlanmış bir birden fazla satıcı bulunmaktadır. GSP aynı zamanda araç rotalama probleminin en temel şeklidir. ÇGSP' nin amacı her bir satıcının başladığı şehre geri döndüğü ve diğer şehirlerin sadece bir kez ziyaret edildiği minimum maliyetli turların her bir satıcı için bulunmasıdır. Şekil 2.2'de örnek ÇGSP gösterilmiştir.



**Şekil 2.2: ÇGSP ve 3 gezgin satıcı için Hamilton turları**

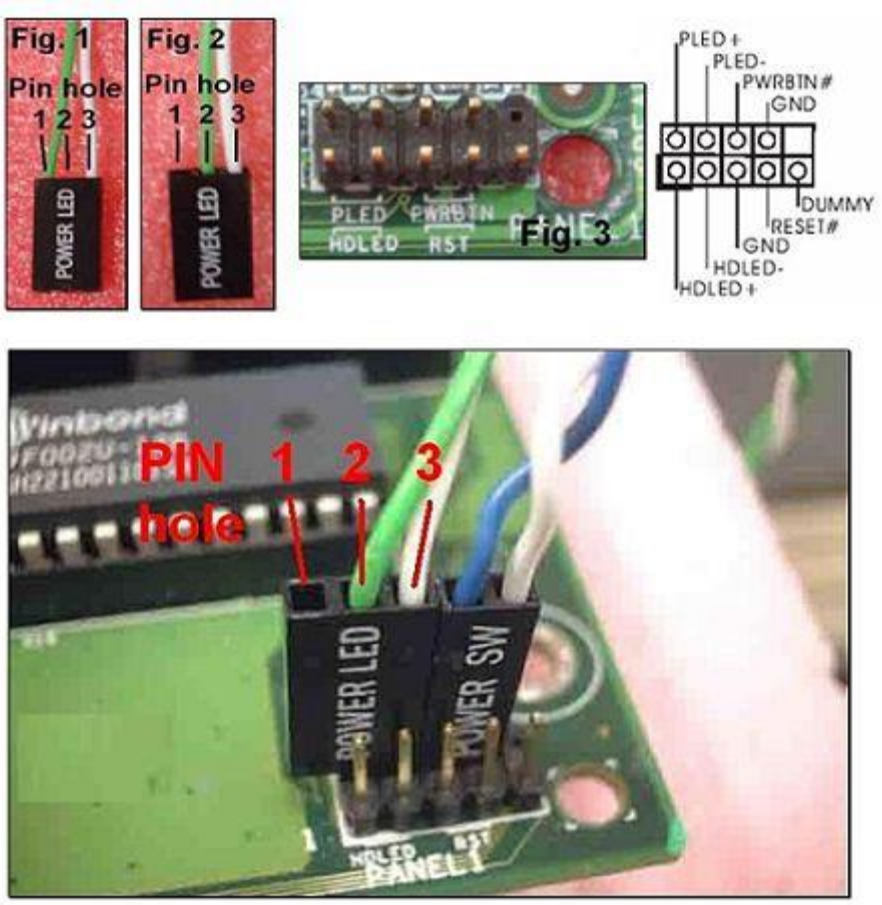
## 2.3 Gezgin Satıcı Problemi Uygulamaları

GSP'nin en yaygın uygulaması bir satıcının *n* şehir arasından en kısa turun bulunmasıdır. Bu basit problem birçok gerçek hayat problemin altında yatmaktadır. Aşağıda bazı uygulama örnekleri verilmiştir.

### 2.3.1 Bilgisayar Bağlantısı

Bu problem Amsterdam Nükleer Fizik Araştırmaları Enstitüsünde bazı bilgisayar arabirimlerinin tasarımı sırasında sıklıkla karşılaşılan bir problemdir. Bir arabirim üzerinde pinlerin yer aldığı bir dizi modülden oluşur. Her bir modülün konumu önceden belirlenmiş durumdadır. Pinlerin belirli bir alt

kümesinin teller ile birbirine bağlanmış olması gerekir. Gelecekteki olası değişiklikler veya düzeltmeler ve pinlerin küçük boyutları dikkate alındığında en fazla iki tel bir pine bağlanabilir. Çapraz sinyali önlemek ve kablolama izlenebilirliğini geliştirmek için toplam tel uzunluğunun en aza indirilmesi gerekir. Problemin yapısı ve toplam tel kullanımının en küçüklenmesi dikkate alındığında GSP olarak ifade edilebilir [27]. Örnek bir resim Şekil 2.3'de gösterilmiştir.




**Şekil 2.3: Pinler ve kablo girişleri**[1]

### 2.3.2 Devre Levhalarının Delinmesi

Devre levhalarının üzerine iletkenlerin yerleştirilebilmesi için levhaların üzerinde delikler açılır. Bu delikler farklı çaplarda olabilir. Farklı çaplarda

[1]http://www.asrock.com/support/faq/20040315-01.jpg

deliklerin açılması için delgi makinesinin kafa kısmı delik çapına göre değiştirilir. Bu değişimler zaman kayıplarına neden olur. Devre levhalarının delinmesi işlemi bir seri GSP olarak tanımlanabilir. Bu problemde amaç kafa kısmının hareket zamanının en küçüklenmesidir [2]. Devre levhalarının delinmesi GSP'nin doğrudan uygulamalarından biridir [28]. Farklı çaplarda delik açma işlemi yapan bir delgi makinesi Şekil 2.4'de gösterilmiştir.

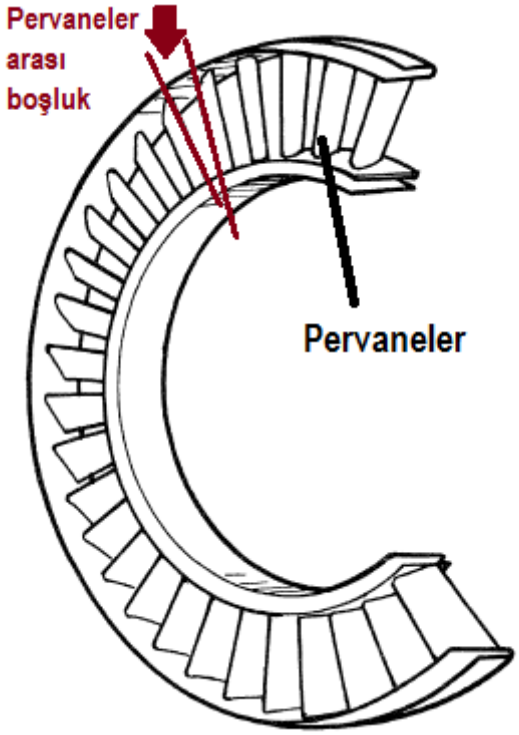

**Şekil 2.4: Devre delgi makinesi**[2]

### 2.3.3 Gaz Tribünü Motorlarının Bakımı

Gaz tribünü motorlarının bakımında kullanılan pervanelerin doğru konumlandırılması ekonomik faydalar sağlamaktadır. Pervanelerin en iyi konumda yerleştirilmesi GSP olarak modellenebilir. Bu uygulama Plante ve arkadaşları tarafından bir GSP olarak önerilmiştir [29]. Şekil 2.5'de gaz sıkıştırma montajı gösterilmiştir. Montaj esnasında pervanelerin hangi sırada

[2] Resim http://www.youtube.com/watch?v=r1Nqs5nttmI&feature=share adresindeki videodan alınmıştır.

yerleştirileceğinin belirlenmesi ile düzgün gaz akışının sağlanması amaçlanmaktadır. Pervaneler arasındaki boşluk akışı biçimlendiren bir etkendir.




**Şekil 2.5: Pervane montajı**

### 2.3.4 Kristalografi

Kristalografi mineralojinin bir dalı olup, minerallerin şekillerini ve iç yapılarını inceler. X-ışını kırınım ölçer cihazı kristal yapıdaki materyallerin iç yapısı hakkında bilgi elde etmek için kullanılır. Bu amaçla bir dedektör farklı pozisyonlardan kristalin X-ışını yansıma yoğunluğunu ölçer. Bazı deneyler kristaller üzerindeki X-ışınlarının yoğunluğunu tespit etmek için çok sayıda ölçüm gerektirir. Tipik bir ölçüm deneyi kırınım ölçer ile 5000 ile 30000 arasında okuma gerektirir. Ölçümler arasında okuma aparatlarının değişimi için gereken zaman toplamda büyük miktarlara ulaşmaktadır. X-ışını kaynağı kullanımı ve kırınım ölçerin faydasını geliştirmek için okuma sırası önemli bir parametredir. Burada her bir okuma, GSP'deki düğümlere karşılık gelir. Okuma aparatlarının yeniden konumlandırma işlemi sırasındaki gecikme zamanı GSP'deki iki düğüm arasındaki uzaklık olarak temsil edilir [30]. Toplam gecikmeyi en küçükleyecek

okuma sırası GSP'nin bir çözümü olarak değerlendirilir. Kristal yapısının analizi çalışmaları GSP'nin önemli uygulamalarından biridir [2].

### 2.3.5 Duvar Kâğıdı Kesimi

Belirli uzunlukta desene sahip bir şablonun her defasında tekrar ettiği uzun bir top duvar kâğıdından *n* adet kesim yapılması durumunda toplam kâğıt israfını en aza indirgeyecek kesim sırasının bulunması GSP olarak modellenebilir [31].

### 2.3.6 Sipariş Toplama

Bu problem bir depodaki malzeme taşıma ile ilişkilendirilir [32]. Bir depo ve bu depoda bulunan parçaların bir kümesi için bir sipariş geldiğini varsayalım. Bu siparişleri müşteriye göndermek için parçalar depodaki araçlar tarafından toplanmalıdır. GSP ile ilişkisi ise parçaların depodaki konumları bir grafın düğümlerine karşılık gelmektedir [2]. En kısa süreli toplama süresi sağlayan en kısa rotanın bulunması problemi GSP olarak çözülebilir.

### 2.3.7 İş Sıralama

*N* adet işin tek bir makinede ardışık olarak gerçekleştirildiğini ve $c_{ij}$'nin *j* işinin *i* işinden sonra hemen başladığı durumda değişim süresi olduğunu varsayalım. Yapay bir iş eklenerek bu problem GSP olarak formüle edilebilir [3].

## 2.4 Gezgin Satıcı Probleminin Karmaşıklığı

Hesaplama teorisi teorik bilgisayar biliminde ve matematikte, bir algoritma kullanılarak bir problemin hesaplama modeli üzerinde etkin bir şekilde nasıl çözüleceği ile ilgilenen bir daldır. Hesaplama teorisi; hesaplanabilirlik teorisi, karmaşıklık teorisi ve hesaplama karmaşıklığı teorisi adında üç ana dala ayrılmaktadır. Karmaşıklık teorisi ise bir problemin sadece tümüyle bilgisayarda

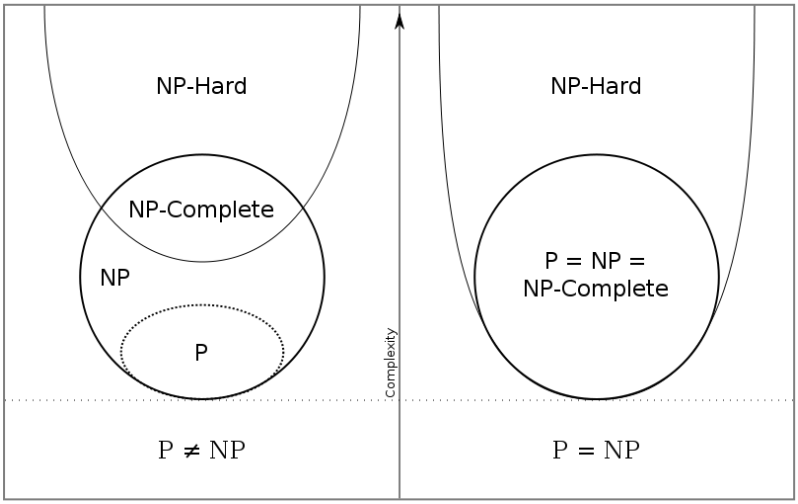
çözülüp çözülemeyeceğini değil aynı zamanda etkin bir şekilde nasıl çözülebileceği ile de ilgilenir. Karmaşıklık zaman ve alan karmaşıklığı olmak üzere iki önemli açıdan ele alınır. Hesaplamayı gerçekleştirmek için kaç adım ve ne kadar bellek gerektirdiği önemlidir. Bilgisayar bilimcileri bu gereksinimleri analiz etmek için zaman veya belleği problemin fonksiyonuna bir girdi olarak alırlar. Örneğin uzun bir sayı listesi içinden belirli bir sayının bulunması sayıların listesi büyüdükçe daha zor bir hal alır. Eğer sıralanmamış veya indeklenmemiş *n* adet sayıdan oluşan bir liste içinden aranan sayı bulunmak istenirse tüm sayılara bakmak gerekebilir. Bu nedenle bilgisayar bu problemi çözmek için problemin büyüklüğüne göre doğrusal şekilde artan işlem adımlarına ihtiyaç duyar. Girdi problemini basitleştirmek için bilgisayar bilimciler büyük O gösterimini benimsemişlerdir [33]. Örnek olarak, O(1) ifadesi, çalışma zamanının girdi büyüklüğü ile bağlantısız olarak sabit zamanda çalıştığını, O(*n*) ifadesi, çalışma zamanının *n* girdi büyüklüğü ile doğru orantılı olduğunu, O($n^2$) ifadesi ise çalışma zamanının girdi büyüklüğünün karesi ile doğru orantılı olduğunu gösterir.

Karmaşıklık teorisine göre bir algoritma ile polinom zamanda bir çözüm sağlanıyorsa problem P (Polynomial) sınıfında, eğer polinom zamanda bir çözüm sağlanamıyor ancak doğrulanabiliyorsa NP (Non-deterministic polynomial) sınıfındadır [34]. Hesaplama teorisine göre problem sınıflarının gösterimi Şekil 2.6'da verilmiştir. Karmaşıklık, verilen girdilere göre kaynak kullanım miktarının değişimi ile ilgilenir. Bir problemin girdi boyutu *n*'in büyümesine karşılık ilgili algoritmanın çalışma zamanı *N*'in büyüme oranı problemin zorluk derecesini belirlemektedir. Karmaşıklık çoğu durumda zaman ile bağıntılıdır ve O ile ifade edilir. Örnek olarak O($n^2$) ifadesi ise çalışma zamanının girdi büyüklüğünün karesi ile doğru orantılı olduğunu gösterir. Alan Turing tarafından geliştirilen Turing makinesi, karmaşıklık teorisini ifade ederken kullanılan en önemli kavramlardan biridir. Turing'in tasarladığı sanal makine, aynı girdiler için her zaman aynı sonuçları üretmektedir.

P sınıfındaki problemler Turing makinesinde önceden hesaplanabilen bir polinom zamanda çözülebilirler. NP sınıfındaki problemler ise, polinom zamanda ancak belirlenimci olmayan Turing makinesi ile çözülebilmektedirler. Belirlenimci olmayan Turing makinesi Belirlenimci Turing makinasından farklı

olarak, aynı durum için birkaç adım arasından seçim yapabilir [127]. Uygulamada belirlenimci olmayan Turing makinesi bulunmaması sebebiyle sonucu sağlayan ve çalışma zamanı açısından verimli olan bir algoritmaları olmadığı varsayılır.

NP-Tam türündeki problemler NP kümesinin bir alt kümesini oluştururlar ve hızlı çözüm sağlayan bilinen bir algoritmaları yoktur. Bu kümeye ait problemlerin çözümü için kullanılan algoritmalar, problemin büyüklüğünün artması ile birlikte çözüm sürelerinde çok büyük miktarlarda artış meydana gelir. NP-Tam sınıfındaki problemler çözümleri için en az üstel zamana ihtiyaç olan problemlerdir.

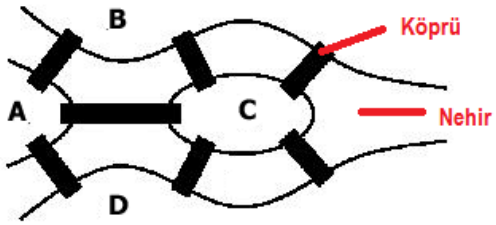



**Şekil 2.6:P, NP, NP-Tam ve NP-Zor Sınıfları**

GSP'nin NP-Tam kategorisinde olmasının gerçek sebebi ise, çözüm için araştırılacak yolların fazla olmasından değil çözüm elde etmek için harcanacak zamandaki artışın, problem girdisiyle olan ilişkisinden ileri gelmektedir [22]. GSP'nin ilk adımında (n-1) düğüm arasından, sonraki adımda ise (n-2) düğüm arasından seçim yapılır. Bu şekilde tüm düğümler ziyaret edildiğinde (n-1)! seçeneğin değerlendirilmesi gerekmektedir. 5 şehirli bir problem için 120 uygun çözüm varken, 10 şehirli bir problem için 3.600.000'den fazla uygun çözüm vardır. GSP'nin en iyi çözümünü üretmek için üstel zamana ihtiyaç duyulduğu için GSP NP-Tam sınıfına ait bir problemdir [2, 35].

## 2.5 Gezgin Satıcı Probleminin Tarihsel Gelişimi

GSP adının ilk kez ne zaman kullanıldığı bilinmemekle birlikte çalışmaların Euler tarafından başlatıldığı düşünülmektedir. Euler turu, serim kuramının da kurucusu olarak kabul edilen İsviçreli matematikçi Leonhard Euler(1707-1783) tarafından Königsberg köprüleri üzerinde tasarlanan bir problemin çözümü için tanımlanmıştır. Königsberg köprüleri (Şekil 2.7) problemi, eski doğu Prusya topraklarında kalan Königsberg kentinin (bugünkü adı Kaliningrad) halkı tarafından, kentin içinden geçen Pregel nehri üzerindeki 7 köprüden geçiş yapmaya dair pazar eğlencesi olarak tasarlanmış bir oyundur. Oyundaki temel soru şudur: Acaba her köprüden yalnızca bir kere geçmek suretiyle, bir yerden başlayıp tekrar aynı yere dönülmesini sağlayacak yol var mıdır? [36].



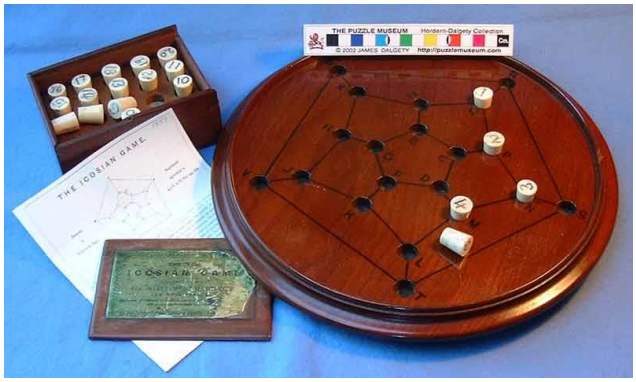

**Şekil 2.7: Königsberg köprüleri**[3]

Grafların üzerinde kenarlardan yalnız bir kez geçildiği ve en küçük maliyetle kapalı bir tur oluşturan grafa Euler çevirimi denir. Königsberg köprüleri problemi bir Euler grafı değildir. Problemin bu hali ile çözülmesinin mükün olmadığını kanıtlayan Euler, çözümü mümkün kılacak garfların sahip olması gereken özellikleri göstermiştir [122, 128].

Euler'den sonra 1800'lerde İrlandalı matematikçi Sir William Hamilton ve İngiliz matematikçi Thomas Penyngton Kirkman tarafından GSP ile ilgili matematiksel problemler çalışılmıştır. Hamilton'ın Icosian oyunu Hamilton

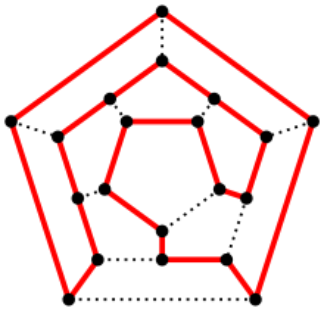
[3] http://mail.colonial.net/~abeckwith/atm/topo.html

çevrimini bulma üzerine kurulmuş eğlenceli bir bulmaca oyunudur [37]. Şekil 2.8'de Icosian oyun tahtası ve topları gösterilmiştir.




**Şekil 2.8: Hamilton Icosian oyunu**[4]

Şekil 2.9'da Icosian oyunun çözümü verilmiştir [39].



**Şekil 2.9: Icosian oyununun çözümü**

GSP'nin genel biçimi başta Karl Menger olmak üzere 1930'larda Viyana ve Harvard'daki matematikçiler tarafından çalışılmıştır. Menger problemi tanımlamış, kaba-kuvvet (brute-force) algoritmasını ve en yakın komşu sezgiselinin neden en iyi çözümü veremediğini incelemiştir [37]. Kaba-kuvvet algortiması problemin çözümü için çözüm uzayındaki tüm alternatiflerin denenmesidir [123].

1950 ve 1960'lı yıllarda problem bilimsel çevrelerde daha çok çalışıan bir hale gelmiştir. Dantzig ve arkadaşları doğrusal programlamayı kullanarak kesme düzlemi metodunu geliştirmişler ve bu yöntemle 49 şehirli bir GSP'yi

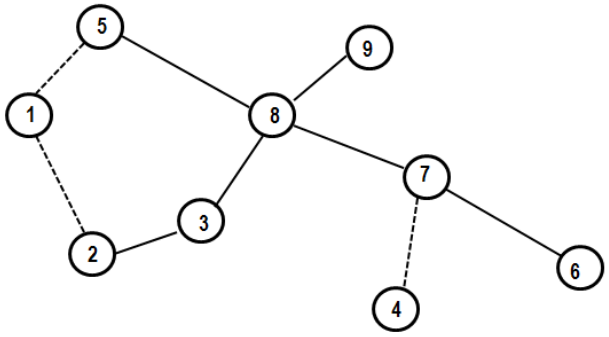
[4] Şeklin alındığı yer [38] numaralı kaynakta gösterilmiştir.

çözmüşlerdir [40]. 1956'da Flood en yakın komşu ve 2-opt algoritmalarının tartışıldığı bir makale yayınlamıştır [41]. 1960'da Miller ve arkadaşları GSP'nin tamsayılı programlama formülasyonunu ortaya koymuşlardır [42]. 1962'de Held ve Karp GSP'nin kesin çözümü için dinamik programlama algoritmasını ortaya koymuşlardır [43].

1963 yılında Little ve arkadaşları dal ve sınır algoritmasını geliştirmiştir [44]. Dal – sınır yöntemi, tüm uygun çözüm seçeneklerini belirlemeye yönelik bir tekniktir. Ancak optimal çözüme götürmeyen bazı çözüm seçenekleri önceden elenmelidir. Bu nedenle gerekli değerlendirmelerin sayısı, genellikle çözüm alanını küçük alt kümelere böler. Bu alt kümelere " dallandırma noktaları" adı verilir. Her alt küme, daha fazla arastırma gerekip gerekmediğini belirlenmek üzere değerlendirilir. Değerlendirme, amaç fonksiyon değerlerini sınırlarla karşılastırarak gerçekleştirilir. Maliyet minimizasyonu problemlerinde alt kümenin olurlu çözümleri için amaç fonksiyon değerlerine bir alt sınır hesaplanır. Eğer alt sınır ≥ bir üst sınır ( baslangıçta kullanılır) ise tüm alt kümeler elenir. Algoritma, her değerlendirmeden sonra üst sınırın yeniden düzenlenmesini olanaklı kılacak biçimde tasarlanmıştır. Dallandırma elenmemiş ve en küçük alt sınırlı alt kümelerde yapılır. Sınırlar arası fark, daha iyi çözümler bulundukça azalır ve en iyi çözüme bu şekilde ulaşılır [124].

1958 yılında Croes 2-opt algoritmasnı önermiştir [125]. 2-opt algoritmasında turdaki iki kenar (bağlantı) silinir ve böylece tur iki parçaya ayrılır. Sonra bu iki parça farklı geçerli bir yol oluşturacak şekilde tekrar birleştirilir. 1958 yılında Bock tarafından 3-opt algoritması geliştirilmiştir [126]. 3-opt bir ağ ya da tur içinde farklı üç bağlantının silinmesi ve turun farklı bir yol oluşturacak şekilde yeniden birleştirilmesidir. Karg ve Thompson 1964'te en ucuz yerleştirme (cheapest insertion) sezgiselini geliştirmişlerdir [45]. Yine 1964 yılında Clarke ve Wright tarafından bir tasarruf (savings) algoritması geliştirilmiştir [50]. Tasarruf algoritması her bir adımda turlar kümesinin daha iyi bir küme elde etmek üzere değiştirilmesine dayanmaktadır. Araç turu belirleme problemi için en yaygın kullanılan algoritmadır. 1965'teki Lin'in yayını GSP için sezgisel yöntemler üzerine yazılmış makaleler önemli bir yere sahiptir. k-opt sezgiseli ve 3-opt uygulaması ile elde edilen tura etkili bir çözüm verilmiştir [46]. k-opt sezgiseli, 2-

opt ve 3-opt sezgiselinin özel bir halidir. Burada k>3 olmak zorundadır. Yani bir turdan en az dört bağlantının silinmesi gerekir. 1970'de Held ve Karp 1-Ağaç gevşetmesini ve düğüm ağırlıklarını kullanmışlardır [47]. Şekil 2.10'da görülen 1-ağacı için öncelikle 1 şehri dışında kalan şehirler için en küçük yayılan ağaç belirlenir. Daha sonra 1 şehri ile yayılan ağacı birleştiren en kısa iki bağlantı belirlenerek ağaca eklenir.




**Şekil 2.10: 1-Ağacı**

Christofides 1971'de kendisinden önce GSP'nin en iyi çözümü için önerilen minimum örten ağaç ve atama problemine dayalı alt sınır hesaplamalarına ek olarak, yeni bir alt sınır algoritması önermiştir. 14 test problemi için bu yeni alt sınır algoritması uygulanmış ve simetrik durumda en iyi değerin %4.7, asimetrik durumda ise en iyi değerin sadece %3.8 altında alt sınır değerleri elde edilmiştir [48]. 1973'de Lin ve Kernighan etkin bir sezgisel yöntem (LK) geliştirmişlerdir [49]. LK kısaca yeni bir tur oluştumak için alt tur çiftlerinin değiştirilmesi esasına dayanır. 2-opt ve 3-opt sezgisellerinin genelleştirilmiş halidir. 2-opt ve 3-opt turu kısaltmak için iki veya üç yolu değiştirirler. LK sezgiseli ise her bir adımda kaç tane yolun değiştirileceğine karar veren uyarlanabilir bir sezgiseldir. 1977 yılında Rosenkrantz ve arkadaşları tarafından tur kurma sezgiselleri çalışılmıştır [51]. Yine 1977'de Karp tarafından bölümleme (partitioning) algoritması sunulmuştur. Bu algoritmada şehirler küçük gruplara bölünür, her grup için en iyi tur bulunur ve turlar birleştirilir [52]. Grötschel ve Padberg kesme düzlemleri hakkında bir literatür araştırması sunmuşlar ve kesme düzleminin etkinliğini göstermişlerdir [53]. 1980 yılında Golden ve arkadaşları sezgisel algoritmaları gruplamış ve performanslarını

değerlendirmişlerdir [56]. 1980 yılında Crowder ve Padberg önceki çalışmalarında tanımladıkları kesme düzlemi algoritmasını geliştirerek 318 şehirli bir problemi çözmüşlerdir [54]. 1987'de Padberg ve Rinaldi dal-kesme algoritması ile 532 şehirli bir problemi çözmüşlerdir [55]. 1989'da Carpaneto ve arkadaşları SGSP için yeni bir alt sınır (lower bound) tanımlamışlardır [57]. 1990 Smith ve arkadaşları SGSP için 2'li eşleme gevşetmesine karşılık gelen doğrusal problemin çözümünden yeni altı sınırlar elde etmişlerdir [58]. 1991'de Grötschel ve arkadaşları yeni bir kesme düzlemi prosedürü önererek 1000 şehire kadar SGSP'yi çözmüşlerdir [59]. 1992 yılında Laporte bazı kesin matematiksel ve yaklaşık algoritmaları incelemiştir [3]. Bu algoritmalar Dantzig ve diğerlerinin tamsayılı doğrusal programlama formülasyonu, alt sınır ataması ve ilişkili dal sınır algoritmaları, en kısa örten ağaç algoritmaları, 2'li eşleme alt snır algoritmalarıdır. David Applegate, Robert E. Bixby, Vašek Chvátal ve William J. Cook tarafından geliştirilen Concorde TSP solver ile 1992 yılından itibaren büyük problemlerin çözümünde oldukça mesafe kat edilmiştir.Tablo 2.1'de GSP'nin Concorde ile çözümleri gösterilmiştir.

**Tablo 2.1: Concorde ile çözülen GSP problemleri**[5]

| | | | |
|---|---|---|---|
| 1992 | Concorde | 3038 | pcb3038 |
| 1993 | Concorde | 4461 | fnl4461 |
| 1994 | Concorde | 7397 | pla7397 |
| 1998 | Concorde | 13509 | usa13509 |
| 2001 | Concorde | 15112 | u15112 |
| 2004 | Concorde | 24978 | sw24978 |
| 2004 | Concorde with Domino-Parity | 33810 | pla33810 |
| 2006 | Concorde with Domino-Parity | 85900 | pla85900 |

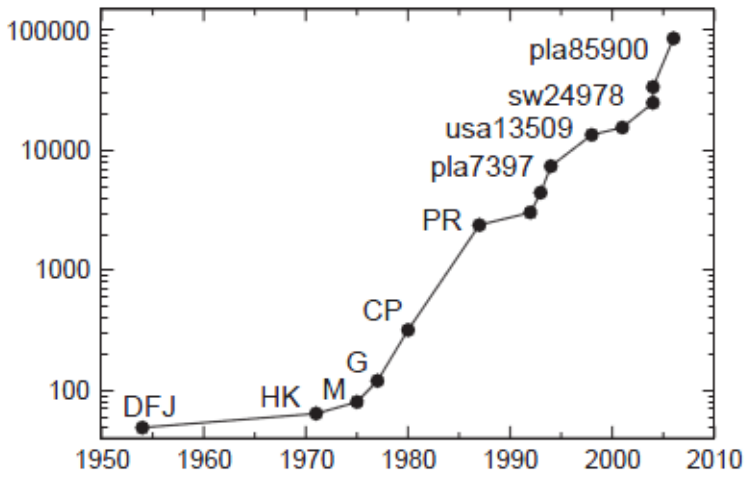
## 2.6 Gezgin Satıcı Probleminin Kilometre Taşları

1954 yılında Dantzig ve arkadaşlarının 49 şehirli problemi çözmelerinden bu yana GSP'nin çözümünde çok büyük mesafe kat edilmiştir. Tablo 2.2'de ve Şekil 2.11'de GSP'nin çözümünde görülen önemli dönüm noktaları [13] gösterilmiştir.

[5] Veriler [13] numaralı kaynaktan alınmıştır.

**Tablo 2.2: GSP'nin çözümünde dönüm noktaları**[6]

| Yıl | Araştırma Ekibi | Şehir Sayısı | Test Problemi |
|---|---|---|---|
| 1954 | G. Dantzig, R. Fulkerson, and S. Johnson | 49 cities | dantzig42 |
| 1971 | M. Heldand R.M. Karp | 64 cities | 64 randompoints |
| 1975 | P.M. Camerini, L. Fratta, and F. Maffioli | 67 cities | 67 randompoints |
| 1977 | M. Grötschel | 120 cities | gr120 |
| 1980 | H. Crowderand M.W. Padberg | 318 cities | lin318 |
| 1987 | M. Padbergand G. Rinaldi | 532 cities | att532 |
| 1987 | M. Grötscheland O. Holland | 666 cities | gr666 |
| 1987 | M. Padbergand G. Rinaldi | 2,392 cities | pr2392 |
| 1994 | D. Applegate, R. Bixby, V. Chvátal, and W. Cook | 7,397 cities | pla7397 |
| 1998 | D. Applegate, R. Bixby, V. Chvátal, and W. Cook | 13,509 cities | usa13509 |
| 2001 | D. Applegate, R. Bixby, V. Chvátal, and W. Cook | 15,112 cities | d15112 |
| 2004 | D. Applegate, R. Bixby, V. Chvátal, W. Cook, an K.Helsgaun | 24,978 cities | sw24798 |
| 2005 | D. Applegate, R. Bixby, V. Chvátal, and W. Cook | 33,810 cities | pla33810 |
| 2006 | D. Applegate, R. Bixby, V. Chvátal, and W. Cook | 85,900 cities | pla85900 |




**Şekil 2.11: GSP'nin yıllara göre gelişim grafiği**

[6] Tablodaki veriler http://math.uwaterloo.ca/tsp/history/milestone.html adresinden alınmıştır.

# 3. GEZGİN SATICI PROBLEMİNİN ÇÖZÜM YÖNTEMLERİ

GSP anlaşılması kolay olmasına rağmen çözümü zor bir problemdir [19, 20, 25]. Geçtiğimiz altmış yıl boyunca GSP bir çok araştırmacı tarafından çalışılmış ve çözümü için çeşitli algoritmaların önerildiği bir optimizasyon problemi olmuştur. Çözüm için önerilen yöntemler kesin ve yaklaşık algoritmalar olmak üzere genellikle iki kategori altında sınıflandırılır [6, 20, 26, 79].

## 3.1 Kesin Algoritmalar

Kesin algoritmalar en iyi çözümü garanti ederler [20]. Dal sınır ve dal kesme algoritmaları özellikle büyük ölçekli problemlerde iyi derecede zaman tasarrufu sağlayan algoritmalardır.

### 3.1.1 Sayma Yöntemi

Bu yöntemde olası tüm kombinasyonlar göz önünde bulundurularak en iyi çözüm elde edilir. *n* şehirli SGSP için *(n-1)!* farklı çözüm içinden en kısa tur uzunluğuna sahip çözümün bulunması gerekir. Yöntemin zaman karmaşıklığı O(n-1!) olduğu düşünülürse büyük problemler için kabul edilebilir bir zaman içinde eniyi çözümü bulmak imkansızdır.

### 3.1.2 Dal - sınır algoritması

Dal - sınır algoritması, bir probleme ait gevşetilmiş çözümden yola çıkar. Dallandırmalar yaparak çözüm uzayının bölümlendirilmesi ve böylece çözüm uzayının daraltılarak araştırılması yoluyla istenen çözümün bulunmasını sağlayan bir yöntemdir [60]. GSP için başlangıç alt sınır, kısıtların gevşetilmesi yoluyla Dantzig Fulkerson Johnson (DFJ) formülasyonundan elde edilir. Atama problemi

(AP) olarak sonuçlanır ve problem O($n^3$) zaman karmaşıklığında çözülür [3]. Birçok yazar tarafından AP gevşetmesine dayanan dal - sınır algoritmaları önerilmiştir [44, 61, 62, 63, 64, 65, 66, 67, 68]. En Küçük Örten Ağaç Probleminde de amaç, GSP'de olduğu gibi bütün noktalara en az taşıma maliyetiyle yayılmaktır. Şekil 2.10'da gösterilen 1-ağacı (1-tree) , 1 şehri dışında kalan şehirler için en küçük örten ağaç belirlenir daha sonra 1 şehri ile mevcut yayılan ağacı birleştiren en kısa iki bağ belirlenerek ağaca eklenir. Bu şekilde elde edilen çözümler GSP turlarını da içermekle birlikte örten ağaç mantığıyla bir noktadan ikiden çok bağın çıkması da mümkündür. Bu durumun engellenmesi amacıyla atama probleminde olduğu gibi dallanmaların gerçekleştiği noktalara bağlanan yollar için dal-sınır algoritması kullanılarak çözüm elde edilir [60]. 1-Ağaç gevşetmesiyle ilgili ilk çalışmalar Christofides [69] ve Held - Karp [47] tarafından yapılmıştır. Sonraki yıllarda yöntem ile ilgili geliştirmeler ve iyileştirmeler önerilmiştir [70, 71, 72, 73, 74].

### 3.1.3 Dal Kesme Yöntemi

Dal kesme yöntemi dal sınır metodu ile çok yüzlü(polyhedral) metodun birleşimiyle oluşmaktadır. *P* problemi, *S* olurlu sonuçların kümesi *P*: min {cx, $x \epsilon S$} olarak tanımlanabilir. *P* probleminin gevşetilmesi ise P' = min {cx, ,Ax≤ b, xϵRn} şeklinde tanımlanır. Gevşetilmiş problem *P'*çözülmüş ve *x* çözümü elde edilmiştir. Eğer *x, S*'in dışında ise *x, S*'den, *S* üzerindeki tüm noktaları sağlayan ve x çözümünü içermeyen bir kesme düzlemi ile ayrılır. Bu eklemeyle daha dar bir gevşetilme mümkün olmaktadır. Bu yöntem S içerisinde bir çözüm bulana kadar devam eder. Dal-kesme metodunun en önemli yanı, GSP için olurlu çözümlerin ortaya çıkmasına yarayan yüzey (facet) eşitsizliklerinin bulunmasıdır. Yıllar geçtikçe literatürde GSP için bazı yüzey eşitlikleri geliştirilmiştir [75]. Dal kesme yönteminin temelleri Dantzig ve arkadaşları [40] tarafından ortaya atıldıktan sonra dal-kesme temelli algoritmalar Crowder ve Padberg [54], Padberg ve Hong [76], Grötschel ve Holland [59] tarafından daha ileriye götürülmüştür.

### 3.1.4 Dinamik Programlama

Dinamik programlama sayma yöntemine benzer bir şekilde çözme tekniğidir. Ancak sayma sırasında tekrarlı hesaplamaların önüne geçerek daha kısa sürelerde çözüm elde edilmesini sağlar. GSP'nin dinamik programlama ile çözümü için Held ve Karp [43] tarafından önerilmiştir.

Kesin çözüm algoritmaları ile ilgili detaylı bilgi için Donald [2], Lawler ve diğerlerine [77] ait çalışmalara bakılabilir.

## 3.2 Sezgisel Yöntemler

Reeves'in tanımına göre "Sezgisel yöntem, çözümün uygunluğu veya en iyi çözümü garanti etmeksizin kabul edilebilir hesaplama maliyetleri ile bir çözüm arayan tekniktir." [78]. Sezgisel yöntemler genellikle basittirler ve kısa çözüm sürelerine sahiptirler. Bazı sezgisel yöntemler en iyi çözümden küçük sapmalarla çözüm sağladıkları için bu yöntemlere yaklaşık algoritmalar demek uygun olabilir [20]. Yaklaşık algoritmaların karmaşıklığı $O(n(\log_2 n)^{O(c)})$ ile ifade edilir [2]. GSP için geliştirilen sezgisel yöntemler üç sınıf altında toplanabilir [6,20].

- Tur kurucu sezgiseller
- Tur geliştirici sezgiseller
- Karma sezgiseller

### 3.2.1 Tur Kurucu Sezgiseller

Tur kurucu sezgisel yöntemlerde her defasında mevcut tura yeni bir şehir ilave edilerek bir tam tur elde edilmeye çalışılır. Bulunan çözüm üzerinde bir iyileştirme yapılmaz. Bu sınıftaki sezgiseller ile elde edilen çözümlerin en iyi çözümden sapmaları %10 -15 aralığındadır. Aşağıda literatürde yer alan bazı tur kurucu sezgisel yöntem açıklanmaktadır.

#### 3.2.1.1 En Yakın Komşu Sezgiseli

En yakın komşu sezgiselinde (nearest neighbor) temel yaklaşım her zaman en yakın şehrin ziyaret edilmesidir. Bu yöntem O ($n^2$) karmaşıklık seviyesine sahiptir. En yakın komşu sezgiselinin adımları aşağıdaki gibidir:

**Adım1:** Rastgele bir şehir seç

**Adım2:** Ziyaret edilmemiş en yakın şehri bul ve git

**Adım 3:** Ziyaret edilmemiş şehir kaldı mı? Eğer kaldı ise adım 2'yi tekrar et.

**Adım 4:** Başlangıç şehrine dön.

#### 3.2.1.2 Clarke ve Wright'ın Tasarruf Algoritması

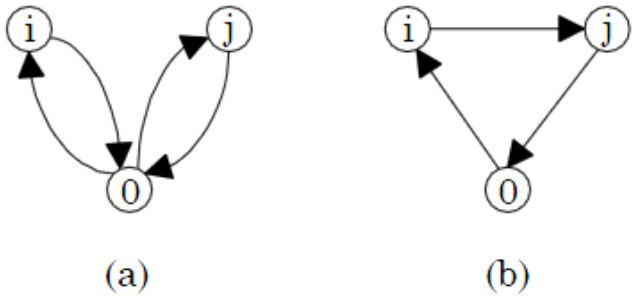
Bu yöntem Clarke ve Wright tarafından geliştirilmiştir [50]. Bu yöntem ARP için geliştirilmiştir. Ancak ARP zaman ve kapasite kısıtları kaldırıldığında GSP olarak çözülebilir. Tasarruf (savings) algoritması aşağıdaki şekilde gösterilen ayrı iki rotaya farklı bir rotanın eklenmesi sonucu elde edilen tasarrufu açıklar. Algoritmanın adımları aşağıdaki gibidir:

**Adım 1:** Rastgele bir P çıkış noktası belirle ve $s_{ij} = d_{pi} + d_{pj} - d_{ij}$

**Adım 2:** $s_{ij}$'leri azalan şekilde sırala

**Adım 3:** Her bir ($i$,$j$) 'yi sıralı bir şekilde çevrim elde edecek şekilde birleştir.

Algoritma Şekil 3.1'de örneklendirilmiştir. Bu algoritmanın karmaşıklık seviyesi O ($n^2$) 'dir.




**Şekil 3.1: Tasarruf algoritması gösterimi**

Şekil 3.1 (a)'da müşteri *i* ve *j* ayrı ayrı ziyaret edilir. Buna alternatif olarak Şekil 3.1 (b)'de iki müşterinin aynı rota içerisinde ziyaret edilmesidir. Şekil 3.1 (a)'daki gibi bir ziyaret söz konusu olduğunda $c_{ij}$ iki nokta arasındaki maliyet olmak üzere Toplam Maliyet 1 $= c_{0i} + c_{i0} + c_{0j} + c_{j0}$ olur. Şekil 3.1(b)'de gösterilen rotada ziyeret yapılırsa Toplam Maliyet 2 $= c_{0i} + c_{ij} + c_{j0}$ olur. Şekil 3.1(b)'deki bir rota ile elde edilen tasarruf ise;

$s_{ij}$= Toplam Maliyet 1 – Toplam Maliyet 2 = $c_{i0} + c_{0j} - c_{ij}$

Tasarruf ne kadar büyükse *ij* rotası o kadar önceliklidir. Burada başlangıç noktası bir depodur. GSP'de başlangıç noktası depo yerine gezgin satıcının çıkış şehri olmalıdır.

### 3.2.1.3 Açgözlü Sezgiseli

Açgözlü (greedy) algoritmasında her adımda alt turları engelleyecek şekilde en kısa bağlantılı noktaları birleştirerek tam bir tur oluşturulur. Algoritma adımları şu şekildedir [2]:

**Adım 1:** Tüm kenarları büyükten küçüğe doğru sırala.

**Adım 2:** Alt turları engelleyecek şekilde en kısa kenarı tura ekle.

**Adım 3:** Tüm düğümlere bağlantı yapılıncaya kadar Adım 2'yi tekrar et.

Açgözlü sezgiseli ile elde edilen çözümlerde Held ve Karp tarafından geliştirilen alt sınıra %15-20 oranında yaklaşılmaktadır. Algoritma karışıklığı $O(n^2(\log_2(n))$ 'dir.

### 3.2.1.4 Ekleme Sezgiseli

Temel ekleme (insertion) sezgiseli tüm şehirlerin bir alt kümesi ile başlar ve geri kalan şehirler bazı sezgisellerle eklenir. Başlangıç turu genellikle bir üçgendir. Algoritmanın karmaşıklığı $O(\ n^2)$'dir. Algoritmanın adımları şu şekildedir [22]:

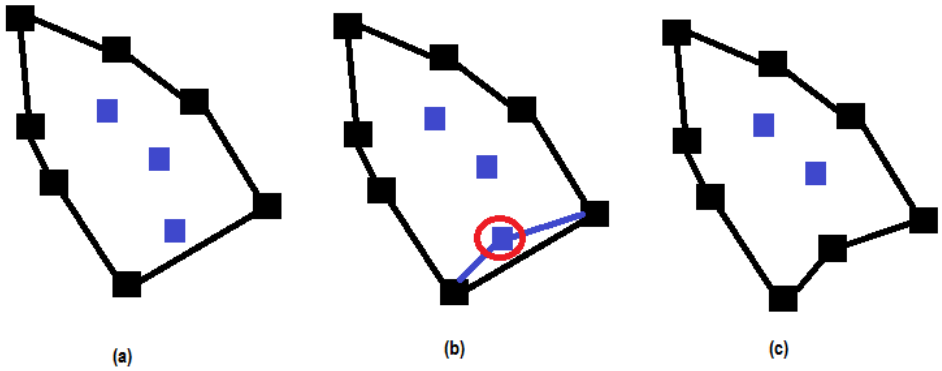
**Adım 1:** En kısa olan kenarı seç ve bu kenar üzerinden bir alt tur oluştur.

**Adım 2:** Alt turda olmayan ve alt turdaki düğümlerden birine en kısa uzaklıktaki bir düğüm seç.

**Adım 3:** Alt tur içinde öyle bir kenar seçilsin ki, Adım 2 seçilen düğümü bu kenara ait düğümlerin arasına koymak en az maliyet sağlasın.

**Adım 4:** Eklenecek düğüm kalmayıncaya kadar Adım ''ye dön.

Algoritmayı örnekleyen adımlar Şekil 3.2'de gösterilmiştir.




**Şekil 3.2: Ekleme sezgiselinin adımları**

Şekil 3.2(a) oluşturulan alt turu göstermektedir. Şeklin b kısmında alt tura eklenecek en az maliyetli düğüm seçimi ve (c)'de seçilen düğümün alt turla birleştirilmesi gösterilmiştir.

#### 3.2.1.5 Christofides Sezgiseli

Bu algoritma Christofides tarafından geliştirilmiştir [48]. Bu algoritma minimum örten ağaç algoritmasına dayanmaktadır. Algoritma karmaşıklığı $O(n^3)$ 'dür. Algoritmanın adımları şu şekildedir:

**Adım 1:** Tüm şehirler kümesinden bir minimum örten ağaç ($T$) oluştur.

**Adım 2:** Tek dereceli düğümleri seç ve bu düğümler üzerinde tüm düğümleri tek bir kez ziyaret eden bir ($M$) bağlı grafı oluştur.

**Adım 3:** $M$ ve $T$'yi birleştirerek bir $G$ grafı oluştur.

**Adım 4:** $G$'de bir Euler döngüsü bul.

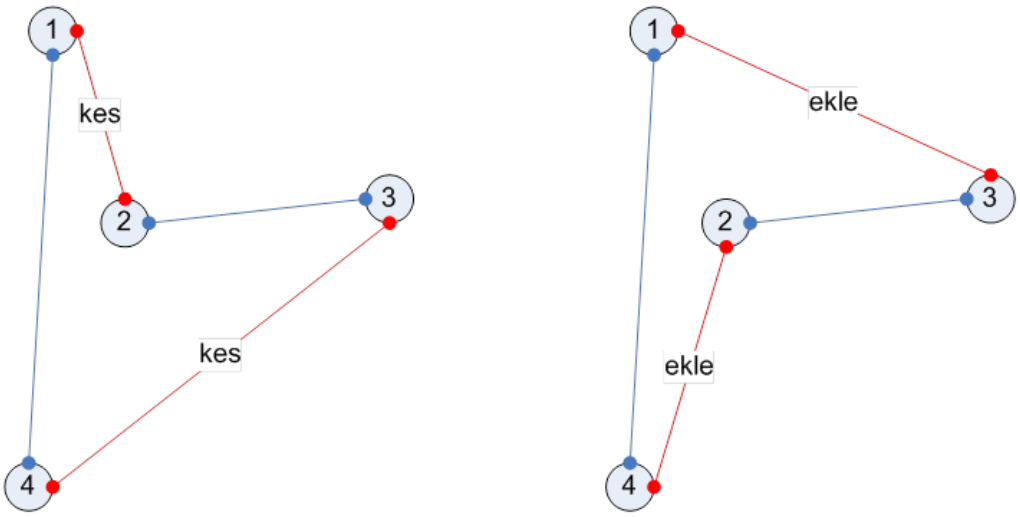
### 3.2.2 Tur Geliştirici Sezgiseller

Tur geliştirici sezgiseller bir tur kurucu sezgisel tarafından bulunan turun iyileştirilmesi için uygulanır. Bu sezgisellerin en yaygın olanları 2-opt ve 3-opt'tur [2]. Tur geliştirme sezgiselleri komşuluk arama süreci olarak değerlendirilebilir. Çünkü her tur bir önceki tura göre daha iyi bir komşuluğun olup olmadığın arar. Bu arama daha iyi bir komşuluk kalmayıncaya kadar devam eder [14].

#### 3.2.2.1 2-Opt

2-opt algortimasında daha önceden oluşturulmuş bir turdan iki bağlantı kopartılarak döngü oluşturmayacak şekilde yeni bir bağlantı oluşturulur. 2-opt

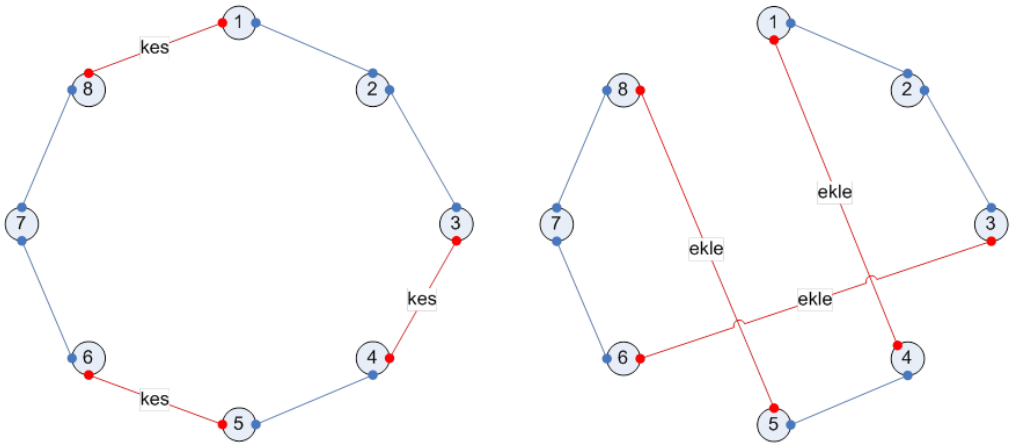
yöntemi ile en iyi çözüme ortalama %5 yaklaşık çözümler elde edilmektedir. Şekil 3.3'de (1-2) ve (3-4) bağlantıları kopartılarak (1-3) ve (2-4) bağlantıları yapılmıştır.




**Şekil 3.3: Orijinal tur ve 2-Opt sonucu elde edilen tur**

### 3.2.2.2 3-Opt

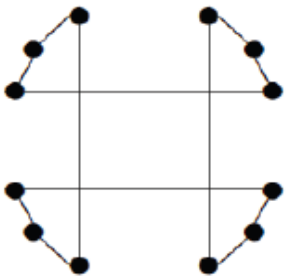
3-Opt algoritması Bock tarafından geliştirilmiştir [126]. 3-opt, 2-opt algoritmasına benzer şekilde çalışır. Ancak farklı olarak iki kenar yerine üç kenar çıkarılır. 3-opt yöntemi ile en iyi çözüme ortalama %3 yaklaşık oranında çözümler elde edilmektedir. Şekil 3.4'de 3-opt ile elde edilen bir tur gösterilmiştir. Şekilde (1-8), (3-4) ve (5-6) bağlantıları kesilerek (5-8), (1-4) ve (3-6) bağlantıları oluşturlmuştur.




**Şekil 3.4: Orijinal tur ve 3-opt sonucu elde edilen tur**

### 3.2.2.3 k-Opt

Üretilmiş bir turun iyileştirilmesi için k-opt algoritması uygulanabilir. Ancak burada çıkarılacak kenar sayısı k > 3 olmalıdır. k> 3 olduğunda hesaplama zamanı artmaktadır. Genellikle köprülerin çaprazlanması (crossing bridges) (Şekil 3.5) adı verilen 4-opt kullanılır.



**Şekil 3.5: Çift köprü geçişi**

### 3.2.2.4 Lin - Kernighan (LK) Sezgiseli

LK sezgiseli Lin ve Kernighan tarafından 1973 yılında önerilmiştir [49]. LK sezgiseli, Held ve Karp alt sınırına ortalama olarak %2 yaklaşık sonuç verir. LK değişebilen k-yol değişim sezgiselidir. Her iterasyonda uygun k değerine karar verirlir. Bu durum sezgiseli oldukça karmaşık bir hale getirir. LK sezgiseli k-opt yönteminden farklı olarak ilk iyileştirmeyi değil en büyük iyileştirmeyi hedefler [23]. Algoritmanın zaman karmaşıklığı $O(n^{2.2})$ 'dir. LK sezgiseli SGSP'nin çözümü için en etkili yöntemlerden biridir [20].

### 3.2.3 Meta-sezgiseller

Metasezgisel, etkili ve en iyi çözümleri bulmak için arama uzayında araştırma ve işletim için farklı yaklaşımları akıllıca birleştiren alt seviye sezgisellere rehberlik eden yinelemeli üretim süreci olarak tanımlanır [117].

**Meta-sezgisel Yöntemlerin Özellikleri**

Metasezgisel yöntemler aşağıda belirtilen özelliklere sahiptir [118]:

1. Meta-sezgiseller arama sürecine rehberlik eden stratejilerdir.
2. Amaç en iyi sonucu bulmak için arama uzayını etkin bir biçimde araştırmaktır.
3. Meta-sezgiseller, basit yerel arama prosedürlerinden karmaşık öğrenme süreçlerine kadar geniş bir aralığı kapsamaktadır.
4. Yaklaşık algoritmalardır ve genellikle deterministik değildirler.
5. Yerel en iyiye takılmayı önlemek için mekanizmalar içerebilirler.
6. Problemlere özgü değil genel yöntemlerdir.

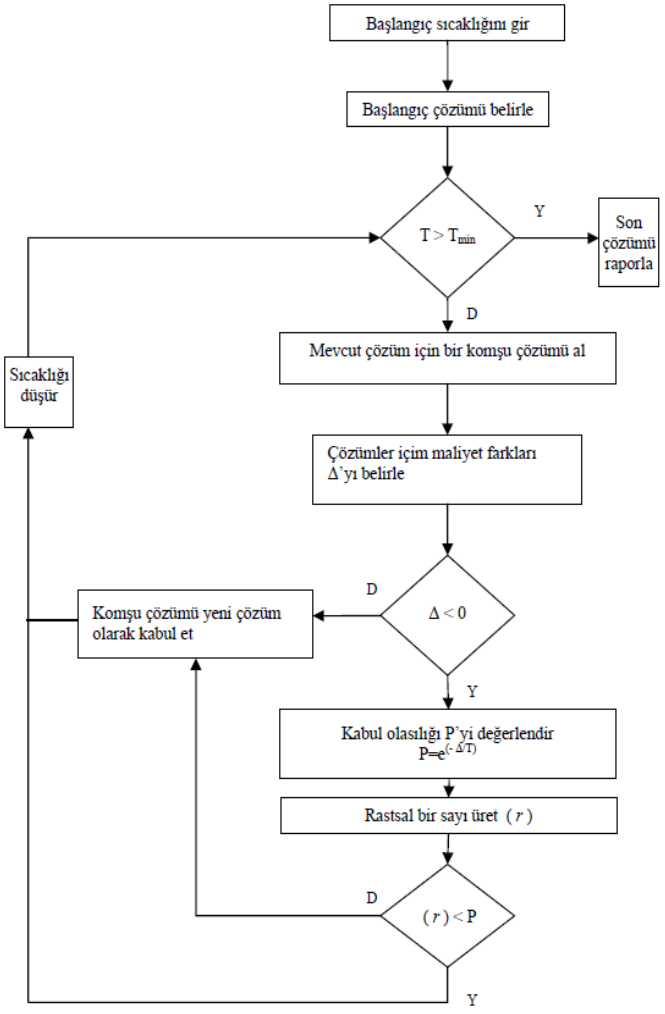
### 3.2.3.1 Tavlama Benzetimi

Tavlama benzetimi (TB) fizikteki bir olgu ile kombinatoryal bir problemin en iyi çözümünün bulunmasına uyarlanmasıdır. Bu iki durum arasındaki benzerliği daha iyi anlatabilmek için bir sıvının donma noktasındaki kristal yapısının elde edilmesi için soğuma sürecini düşünelim. Sıvının hızlı soğutulması istenileni başaramazken, sıvının yavaş soğutulması kristalleşme sıcaklığında uygun olmayan yapıların yeniden ayarlanmasına ve mükemmel dizilimin elde edilmesine izin verir. Her sıcaklık derecesinde sistem minimum enerji düzeyine doğru gevşer [80]. Amaç fonksiyonu değeri fiziki sistemde enerjiye karşılık gelir ve uygun çözümler sistemin kararlı haline karşılık gelir. Belirli sıcaklıklardaki gevşemeler, sıcaklığın derecesi ile kontrol edilen o andaki uygun çözümlerin rassal olarak değişimine izin verilmesi olarak modellenir. Sistem dinamikleri mevcut uygun çözümün yerel düzenlemeleri olarak taklit edilir. Bu benzeşim Tablo 3.1‘de [81] gösterilmiştir.

**Tablo 3.1: Termodinamik benzetimin optimizasyon karşılığı**

| **Termodinamik Benzetim** | **Optimizasyon** |
|---|---|
| Sistem kararlı bir hal alır | Uygun çözüm bulunur |
| Enerji | Maliyet |
| Durum değişkliği | Komşu çözüm |
| Sıcaklık | Kontrol parametreleri |
| Donmuş hal | Sezgisel çözüm |

TB ilk olarak Kirkpatrick ve arkadaşları tarafından 1983 yılında GSP'nin çözümü için kullanılmıştır [83]. Daha sonra Cerny [84] tarafından büyük GSP problemlerine uygulanmıştır. 1990'da Dueck ve Scheuer tarafından eşik değeri (Threshold) kabulü adıyla TB'nin basitleştirilmiş hali yayınlanmıştır [85]. TB algoritmasının akış şeması Şekil 3.6'de verilmiştir.




**Şekil 3.6: TB algoritması adımları iş akışı**

### 3.2.3.2 Yasaklı Arama

Yasaklı Arama (YA) algoritması ilk defa Glover [87] tarafından önerilmiştir. GSP için test sonuçları 1989 yılında Knox ve Glover [88], Fiechter [89] tarafından yayınlanmıştır.

YA, çözümlerin araştırılması esnasında yerel minimuma takılmayı önlemek için daha önce denenmiş çözümlerin tekrar araştırılmasını yasaklamak amacı ile sürekli olarak güncellenen bir yasaklı listesi mantığı ile çalışır [3]. Bu algoritmanın adımları şu şekilde özetlenebilir [86]:

**Adım 1:** Bir başlangıç çözümü (S) bul. Başlangıçta değer atanması gereken parametreler için değerlerini ata.

**Adım 2:** Komşu çözümler üret ve bu çözümler arasından en iyi olanı ($S_{eniyi}$) seç [$S_{eniyi}$ çözümü, yasaklı listesinde olmayan tüm $S^{'}$in komşuluk kümesindeki N(S)' lerin en iyisi].

**Adım 3:** Mevcut çözümü (S), $S_{eniyi}$ ile yer değiştir ve yasaklı listesini yenile.

**Adım 4:** Durdurma kriteri sağlanana kadar Adım 2 ve Adım 3' ü tekrar et.

### 3.2.3.3 Genetik Algoritmalar

Genetik algoritma (GA) fikri ilk kez 1975 yılında Holland tarafından ortaya atılmıştır [90]. GA doğal seleksiyon mekanizmaları ve biyolojik evrim mekanizmalarını temel alan arama algoritmalarıdır [4]. GA'lar kromozom olarak adlandırılan olası çözümlerin popülasyonları üzerinde işlem yaparlar [21]. GA'da ilk olarak problemin yapsı kromozom olarak kodlanır. Daha sonra bu kromozomlardan bir popülasyon oluşturulur. Popülasyon içinden bazı kromozomlar seçilerek çaprazlama ve mutasyon işlemleri uygulanarak yeni kromozomlar elde edilir. Yeni kromozomlarından daha iyi uygunluk değerine sahip olanlar sonraki nesillere aktarılır. Bu süreç durdurma kriteri sağlalana kadar

yinelemeli olarak gerçekleştirilir [5]. Basit bir genetik algoritmanın adımları şu şekilde özetlenebilir [86]:

**Adım 1:** Çözümlere karşılık gelen bir başlangıç popülasyonu oluştur.

**Adım 2:** Popülasyondaki her çözümün uygunluk değerini hesapla.

**Adım 3:** Durdurma kriteri sağlanıyorsa araştırmayı durdur. Yoksa aşağıdaki adımları gerçekleştir.

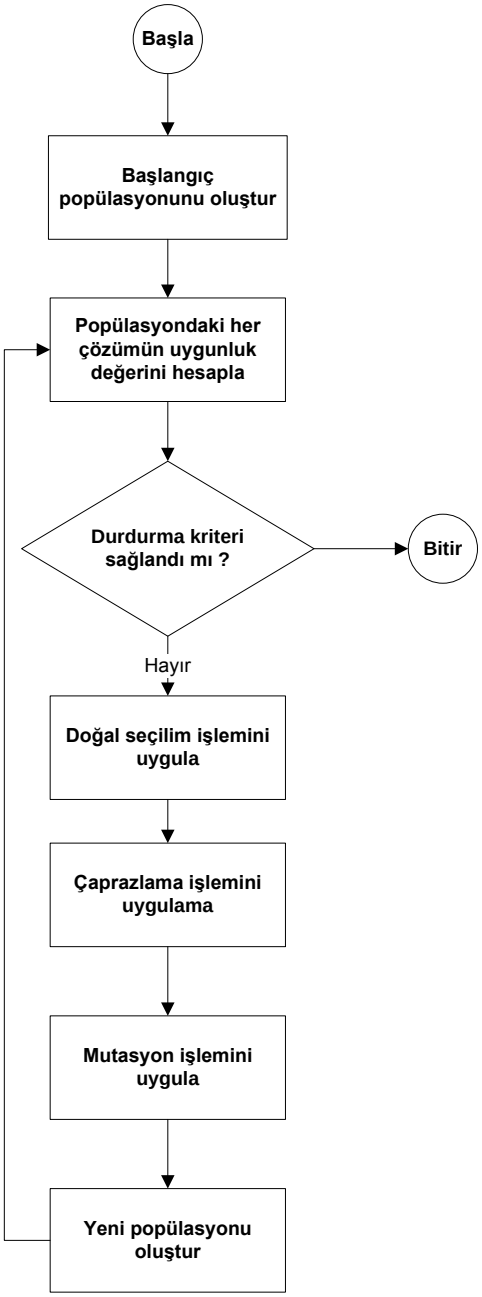
**Adım 4:** Doğal seçilim işlemini uygula (uygunluk değerleri daha yüksek olan çözümler yeni popülasyonda daha fazla temsil edilirler).

**Adım 5:** Çaprazlama işlemini uygula (Mevcut iki çözümden yeni iki çözüm üretilir).

**Adım 6:** Mutasyon işlemini uygula (Çözümlerde rastgele değişimler meydana getirilir). Yeni popülasyonu oluştur.

**Adım 7:** Adım 2' ye git.

Şekil 3.7'de GA'nın akış şeması gösterilmiştir. GA'nın GSP' ye uygulandığı ilk çalışma Brady tarafından yapılmıştır [91]. GA'nın GSP üzerindeki ilk çalışmaları uygun genetik gösterim ve uygun genetik operatörlerin belirlenmesine yöneliktir [93]. Standart GA'nın tek başına performansı düşük olduğu için GA içine yerel arama teknikleri eklenmiştir. Literatürde bu şekilde birçok çalışma bulunmaktadır. Detaylı bilgiye [4, 94]'ten ulaşılabilir.

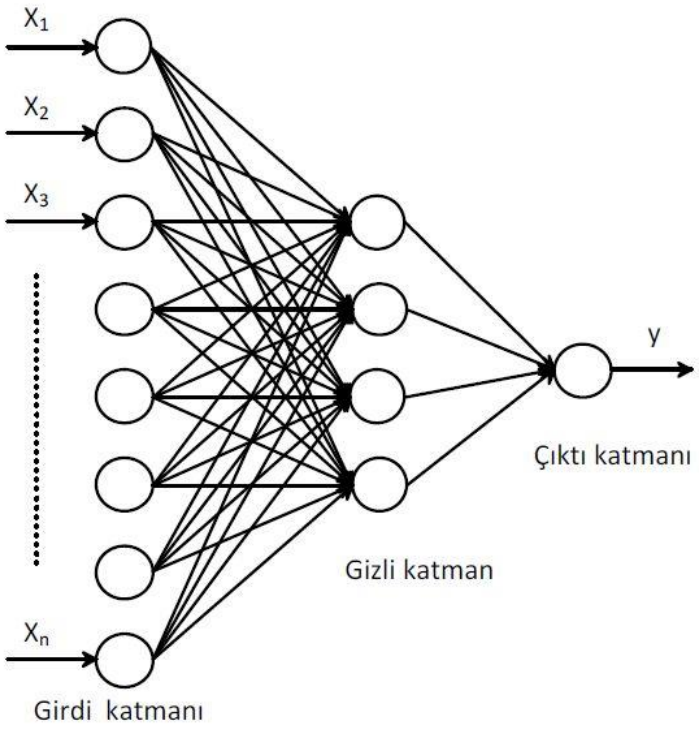



**Şekil 3.7: GA akış şeması**

### 3.2.3.4 Yapay Sinir Ağları

Yapay sinir ağlarında (YSA) insan beyninin çalışması taklit edilir. Temel olarak bir sinir ağı bir çok bağımsız nöron ve nöronlar arasındaki bağlantılardan oluşan bir ağdır. Bu ağ, öğrenme, hafızaya alma ve veriler arasındaki ilişkiyi

ortaya çıkarma kapasitesine sahiptir. Genel olarak bir YSA modeli Şekil 3.8'de görüldüğü gibi, üç adet katman, her katmanda biyolojik sinir hücrelerinin işlevini yerine getiren ve değişik sayılarda olabilen hesaplama elemanları, katmanlar boyunca bu hesaplama elemanları arasındaki yoğun bağlantılardan meydana gelmektedir.



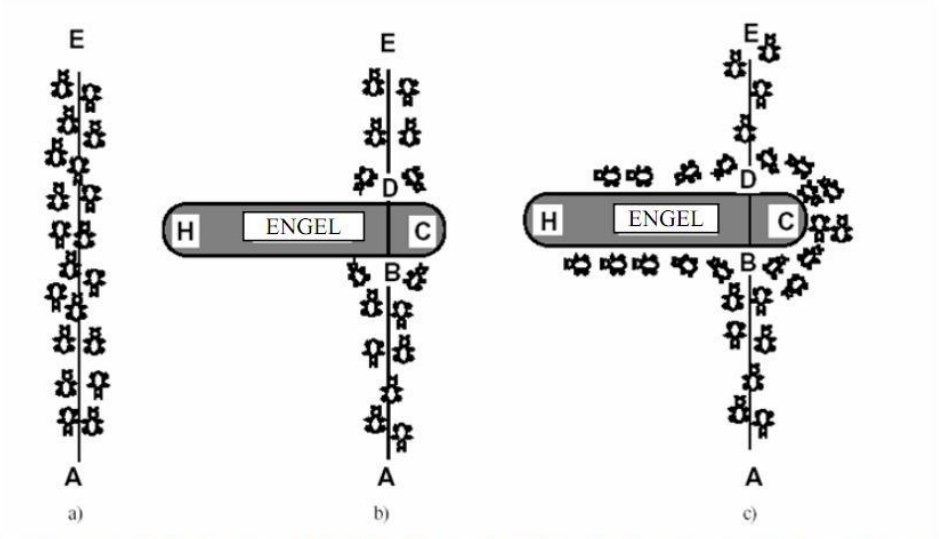

**Şekil 3.8: Genel bir YSA modeli**

GSP açısından nöron temelli olan ilk çalışma Hopfield ve Tank tarafından tarafından yapılmıştır. [95]. GSP için YSA ile yapılan çalışmalara [96, 97, 98, 99, 100] kaynaklarından ulaşılabilir.

### 3.2.3.5 Karınca Kolonisi Optimizasyonu

Karınca kolonisi optimizasyonu (KKO) gerçek karıncaların yiyecek ile yuva arasındaki en kısa yolu bulma davranışından esinlenmiş bir sezgiseldir [4]. Karıncalar, hareket halinde bulundukları yola belirli miktarda feromon maddesi bırakırlar. Karıncaların, feromon maddesinin daha yoğun bulunduğu yönleri tercih

etme olasılığı daha fazladır. Feromon maddesi yoğunluğu sebebiyle tercih edilen yoldaki karınca miktarı artar [86]. Bu durum Şekil 3.9'da gösterilmiştir.




a) Karıncalar A-E arasındaki yolu izlemektedirler. b)Yolun bir yerine bir engel konulmuştur ve karıncalar hangi yönü seçeceklerine rasgele karar verirler. c)Kısa olan yolda daha fazla feromen birikir.

**Şekil 3.9: Gerçek karınca davranışı**[7]

KKO ile ilgili ilk çalışma Dorigo ve arkadaşları [101] tarafından yapılmıştır ve çalışmalarını "Karınca Sistemi" olarak tanıtmışlardır. KKO'nın temel adımları aşağıdaki gibi özetlenebilir [102]:

**Adım 1:** Başlangıç feromon değerleri belirlenir.

**Adım 2:** Karıncalar her düğüme rassal olarak yerleştirilir.

**Adım 3:** Her karınca, sonraki şehri lokal arama olasılığına bağlı olarak seçer ve turunu tamamlar.

**Adım 4:** Her karınca tarafından katedilen yolların uzunluğu hesaplanır ve lokal feromon güncellemesi yapılır.

[7] http://img03.blogcu.com/images/i/n/n/inndustry/978bb85fae07f064d6da7d6f94028493_1364557050.jpg

**Adım 5:** En iyi çözüm hesaplanır ve global feromon güncellemesinde kullanılır.

**Adım 6:** Maksimum iterasyon sayısı yada durdurma kriteri sağlanana kadar Adım2' ye gidilir.

GSP için ilk KKO çalışması Dorigo ve Gambardella [103] tarafından yapılmıştır. KKO ile detaylı bilgiye Dorigo ve Stützle'in "Ant Colony Optimization" isimli kitabından ulaşılabilir [104].

#### 3.2.3.6 Parçacık Sürü Optimizasyonu

Parçacık sürü optimizasyonu (PSO) kuş sürülerinin davranışlarından esinlenmiş bir evrimsel hesaplama tekniğidir [4]. İlk defa doğrusal olmayan fonksiyonların optimizasyonu için Kennedy ve Eberhart tarafından 1995 yılında tanıtılmıştır [116].

PSO'da her bir parçacığın kendine ait bir hızı vardır ve bu hız parçacığı diğer parçalardan aldığı bilgilerle en iyi sonuca doğru hızlandırır. Her bir nesilde bu hız önceki en iyi sonuçlardan faydalanarak yeniden hesaplanır. Bu sayede popülasyonda bulunan bireyler giderek daha iyi pozisyona gelirler. Algoritmayı adımları şu şekildedir [105]:

**Adım 1:** **Popülasyonun oluşturulması;** parçacıklar, rastgele üretilen başlangıç pozisyonları ve hızları ile birlikte oluşturulur.

**Adım 2:** **Uygunluk değerlerinin hesaplanması;** popülasyon içindeki tüm bireylerin uygunluk değerleri hesaplanır.

**Adım 3:** **En iyi üyenin bulunması;** her jenerasyonda bütün bireyler bir önceki nesilde bulunan en iyi (pbest) ile karşılaştırılır. Eğer daha iyi birey varsa yer değiştirilir.

**Adım 4:** **Global en iyinin bulunması;** nesildeki en iyi değer global en iyi değerden daha iyi ise yer değiştirilir.

**Adım 5:** Pozisyon ve hızlarının güncellemesi;

**Adım 6:** Durdurma kriteri sağlanıncaya kadar adım 2-5 tekrar et.

GSP'nin çözümü için Wang ve diğerlerinin PSO için geliştirdikleri özel metodları içeren çalışması [106], Pang ve diğerlerinin parçacıkların hız ve pozisyonları gösteren Bulanık matris gösterimini kulandıkları çalışma [107], Goldbarg ve diğerlerinin hız operötürünün yerel arama yapısı üzerine kurdukları çalışma [108], Shi ve diğerlerinin çalışması [109] bu alandaki önemli çalışmalardır.

#### 3.2.3.7 Yapay Arı Kolonisi Optimizasyonu

Arı davranışlarının optimizasyon alanındaki ilk uygulaması Sato ve Hagiwara tarafından yapılmıştır. Sato ve Hagiwara gerçek arıların davranışlarından esinlenerek "Arı Sistemi" olarak isimlendirilen bir sezgisel geliştirmişlerdir [110]. Daha sonra Teodorovic Arı Sistemi'nin bir türevi olan Arı kolonisi algoritmasını geliştirmiştir [111]. Son olarak Karaboğa arıların yiyecek arama davranışını modelleyerek Yapay arı kolonisi (YAK) algoritmasını geliştirmiştir [112]. Doğada arılar yiyecek ararken aşağıdaki adımları takip ederler [86].

- Yiyecek arama sürecinin başlangıcında, kaşif arılar çevrede rastgele arama yaparak yiyecek aramaya başlarlar.
- Yiyecek kaynakları bulunduktan sonra, kaşif arılar işçi arı olurlar ve kovana nektar taşırlar. Arılar nektarı kovana boşlattıktan sonra ya yiyecek kaynağına dönerler ya da kaynakla ilgili bilgiyi dans ederek kovanda bekleyen gözcü arılara iletirler.
- Gözcü arılar zengin kaynakları işaret eden dansları izlerler ve yiyeceğin kalitesi ile orantılı dans frekansına bağlı olarak yiyecek kaynağını tercih ederler.

Bu açıklamalar doğrultsunda YAK algoritmasının temel adımları aşağıdaki gibidir [86]:

**Adım 1:** Başlangıç yiyecek kaynağı bölgelerinin üretilmesi

**Adım 2:** Görevli arıların yiyecek kaynağı bölgelerine gönderilmesi

**Adım 3:** Olasılıksal seleksiyonda kullanılacak olasılık değerlerinin görevli arılardan gelen bilgiye göre hesaplanması

**Adım 4:** Gözcü arıların olasılık değerlerine göre yiyecek kaynağı bölgesi seçmeleri

**Adım 5:** Bırakılacak kaynakların bırakılışı ve kaşif arı üretimi

**Adım 6:** Maksimum çevrim sayısına ulaşılıncaya kadar Adım 2 – 5 tekrar et.

Bu bölümde özellikle GSP'nin çözümünde en çok tercih edilen algoritmalar açıklanmıştır. Bu algoritmalara ek olarak yine özellikle GSP'nin çözümü için doğadan esinlenilerek geliştirilmiş birçok sezgisel yöntem bulunmaktadır. Sivrisinek konak arama [11], akıllı su damlaları [12], kurbağa sıçrama [22] algoritmaları bu yöntemlere örnek olarak verilebilir.

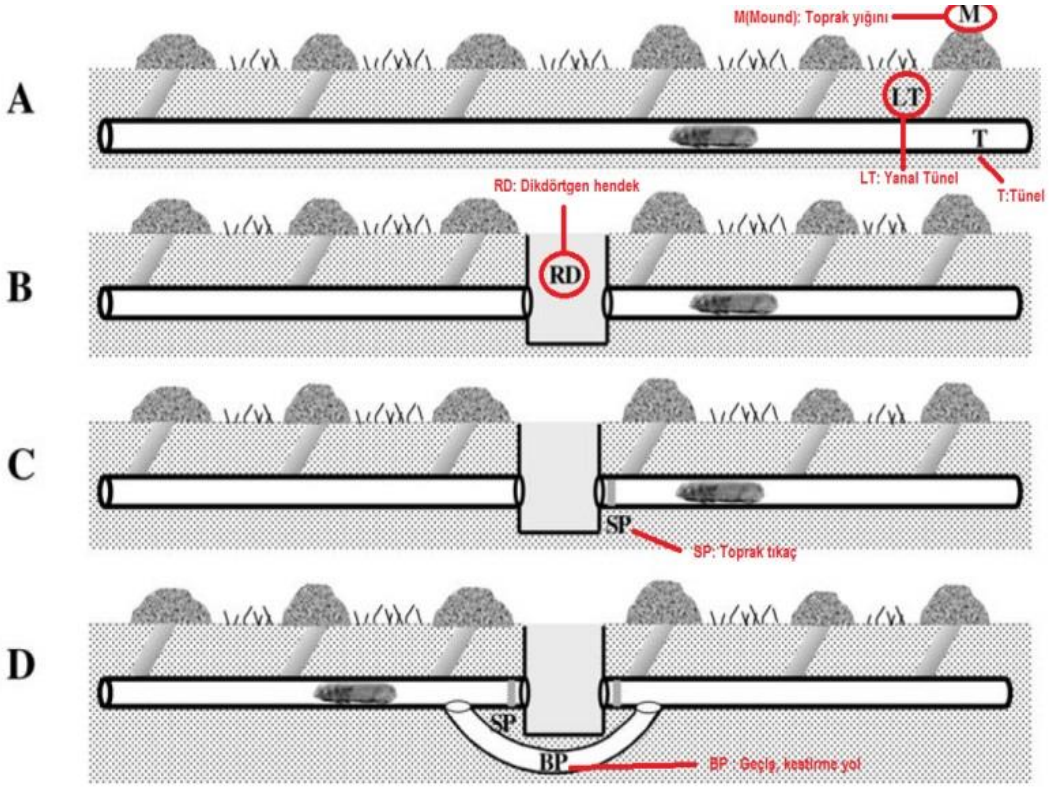
# 4. KÖR FARE ALGORİTMASI

Bu bölümde doğadaki kör fareler hakkında bilgi verilerek kör farelerin hareketlerinden esinlenilerek geliştirilen algoritma detaylı olarak anlatılmıştır.

## 4.1 Kör Fareler

Bio-sonar (yankı ile konumlama, echolocation) birçok hayvan türü tarafından kullanılan biyolojik radardır. Bio-sonar yeteneğine sahip hayvanlar ses dalgaları göndererek bu dalgaların nesnelerden yansımalarını dinlerler. Bu yansımaları çevrelerindeki nesneleri tanımlamak ve konumlandırmak için kullanırlar. Yankı ile konumlama çeşitli ortamlarda yön bulma ve avcılık için kullanılır [113]. Örneğin yarasalar ağız veya burunlarından ses dalgaları gönderiler. Ses dalgaları bir nesneye çarptığında yansımalar oluşturur. Bu yansımalar yarasanın kulaklarına geri döner. Yarasa nesnenin nerede olduğunu, büyüklüğünü ve şeklini belirlemek için bu yansımaları kullanır. Yankı ile konumlama ile yarasalar insan saç teli kalınlığındaki bir nesneyi bile saptayabilirler.

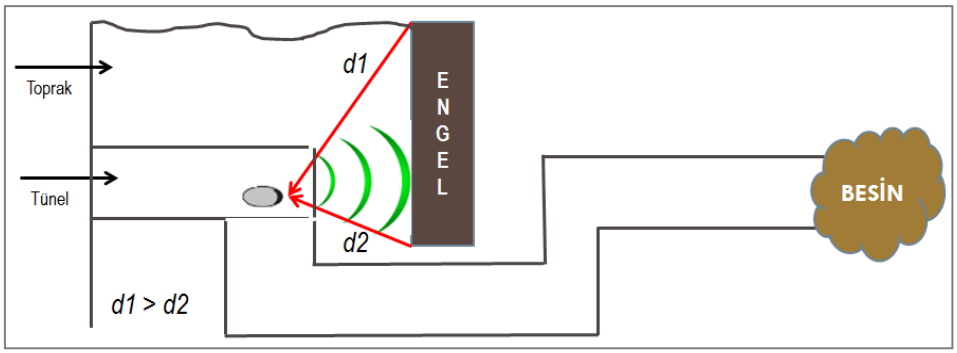
Bu canlılara çarpıcı örneklerden bir diğeri de kör farelerdir (blind mole-rat). Kör fareler koloniler halinde yaşayan en üst düzey sosyal örgütlü (eusocial) canlılardır [15]. Yer altında kazdıkları tünellerde yaşayan kör fareler, tünel kazarken bir engelle karşılaştıklarında enerjilerini ekonomik kullanmak için en kısa yolları tercih ederler [17]. Bu hayvanlar bir nesnenin konum, boyut, uzaklık ve türünü tahmin etmek için ultrasonik sesler (yüksek frekans) yayarlar ve nesnenin yüzeyinden yansıyan ses dalgalarını değerlendirirler [16]. Şekil 4.1'de bir kör farenin bir engelle karşılaştığında engeli geçmesi gösterilmiştir [17].




**Şekil 4.1: Engelin geçilmesi**

Şekil 4.1'de **A** aktif bir kör farenin düz bir hatta ilerleyen tünelini gösterir. **B 'de** dikdörtgen bir hendek (RD:rectangular ditch) tüneli kesecek şekilde çakılır. **C'**de kör fare tünelin hendeğe temas eden açık kısmını toprakla doldurur. **D'**de kör fare tünelin iki ayrı bölümünü engelin altından birleştirir.

Kör fareler engeli aşmak için tünel içinde ilerlerken kafalarını tünelin tavanına vurarak küçük depremler ile dalgalar yaratırlar. Bu dalgaların engele çarpıp yansımasına göre engelin hangi tarafından geçeceklerine karar verirler. Şekil 4.2'de kör fare bir dalga göndermekte ve geri dönen sinyallerden engelin büyüklüğü ve konumunu belirlemektedir. $d_1$ meafesi $d_2$*'den* büyük olduğundan engelin altından tüneli kazarak daha kısa yoldan besine ulaşır.




**Şekil 4.2: Engel geçiş stratejisi**

Koloni yaşamında bir birey yeni bir besin kaynağı bulduğunda, yuvaya geri dönerken geçtiği yolda koku izi bırakır ve özel bir ses çıkararak kolonideki diğer üyeleri uyarır. Kolonideki genç bireyler besin kaynaklarını bulmak için başarılı kaşiflerin koku izini takip ederek besine ulaşırlar [15]. Kör farelerin dikkat çekici bir diğer özelliği ise bu kemirgenlerin dokularındaki zengin moleküler hiyalüronik asit (hyaluronan) sayesinde kansere yakalanmamalarıdır [18].

Taherdangkoo ve arkadaşları tarafından 2012 yılında kör çıplak farelerin yiyecek arama ve koloni yaşamları temel alınarak bir numerik fonksiyon optimizasyon algoritması geliştirilmiştir. Algoritmaya Kör, Çıplak Fare Algortiması adını vermişlerdir. Çalışmalarında yiyecek kaynaklarının ve toprağın sıcaklık, nem, yoğunluk gibi özellikleri algoritmaya uyarlanmışlardır [92].

Bu tez çalışması kapsamında Kör Fare Algoritması adı ile alınan algoritma yukarıda bahsedilen algoritma ile isim olarak benzerlik göstermesine rağmen çıkış noktası ve algoritma adımları, kullanılan parametreler açısından tamamen farklılık göstermektedir.

## 4.2 Kör Fare Algoritması

Bu çalışmada kör farelerin bir engeli geçerken en kısa mesafeyi bulma davranışı GSP çözümüne uyarlanmıştır. Bu fikirle kör fare algortiması (blind mole-rat algorithm) (KFA) olarak isimlendirilen bir algoritma tasarlanmıştır.

KFA'da öncelikle düğümler arasındaki mesafeler $\boldsymbol{d_{ij}}$ kullanılarak mesafe matrisi ve her bir düğüm için küçükten büyüğe sıralı bir şekilde yakınlık matrisi oluşturulur. Bu ön hazırlık işleminden sonra problemde yer alan her düğüme bir fare yerleştirilir. Her bir fare için farenin bulunduğu düğüm ziyaret edilen düğümler listesine eklenir. Tüm düğümler arasındaki yollara $\boldsymbol{s_{ij}}$ başlangıç sinyal değeri ve $R_{ij}$ sinyal iletim katsayısı atanır. Her bir fare bulunduğu düğümden sonra gideceği düğüm çiftini yakınlık matrisinden elde ettiği en yakın $\boldsymbol{q}$ adet komşu arasından $S(m)_{ijk}$ formülünü kullanarak seçer. $S(m)_{ijk}$ değeri en büyük olan düğüm çifti sırası ile gidilecek düğüm çiftini belirler. Burada $i$ farenin o an bulunduğu düğümü gösterir. $j$ ve $k$ ise sırası ile gidilecek olan düğümleri ifade eder. Her bir fare gideceği düğüm çiftini seçtikten sonra seçilen düğümler ziyaret edilen düğümler listesine eklenir, yakınlık matrisinden çıkarılır ve yakınlık matrisi güncellenir. Her bir fare tüm düğümler ziyaret edilene kadar $S(m)_{ijk}$ formülünü kullanarak turunu tamamlar. Daha sonra her bir fare için farenin düğümleri ziyaret sırasına göre toplam tur uzunluğu bulunur ve küçükten büyüğe sıralanır. Tüm ($i$,$j$) yollarında mesafe ve $\boldsymbol{w}$ değerleri alınarak $\boldsymbol{\Delta s_{ij}}$ formülü ile sinyal kayıpları bulunur. Tüm ($i$,$j$) yolları için hesaplanan sinyal kayıpları, $\boldsymbol{s_{ij}}$ başlangıç sinyal değerlerinden çıkarılarak tüm sinyal değerleri güncellenir. Sinyal güncelle işleminden sonra en kısa turu yapan farenin kullandığı yollardaki $\boldsymbol{s_{ij}}$ değerlerine $\boldsymbol{v}$ sabit sinyal değeri eklenir. Başlangıç sinyal değerine, başlanıç sinyal değerinin ***Maksimum sinyal artış oranı*** ile çarpımından elde edilen değer eklenerek ulaşılabilecek maksimum toplam sinyal değeri bulunur. Herhangi bir ($i$,$j$) yolunda maksimum toplam sinyal değerine ulaşan veya geçen bir ($i$,$j$) yolu varsa bu yolun $\boldsymbol{s_{ij}}$ değeri maksimum toplam sinyal değerine sabitlenir. Bu işlemle birlikte bir iterasyon tamamlanmış olur. Bir sonraki iterasyona geçildiğinde $R_{ij}$ sinyal iletim katsayısı değerleri yeniden atanır. $\boldsymbol{s_{ij}}$ sinyal değerleri bir önceki iterasyon sonunda elde edilen son güncel sinyal değerleri olarak kullanılır. Bir önceki iterasyonda olduğu gibi fareler yine yukarıda belirtilen kurallara göre turları tamamlarlar ve sinyal güncelleme işlemleri gerçekleştirilir. Durdurma kriteri maksimum iterasyon sayısına ulaşılana kadar iterasyonlar gerçekleştirilir. İterasyonlar tamamlandığında o ana kadar bulunmuş en kısa tur mutasyon işlemine tabi tutulur. Mutasyon işlemi için en iyi tur kromozom yapısına dönüştürülür. Kromozom uzunluğu içinden bir rassal sayı (RS) üretilir. Bu rassal sayıya mutasyon aralığı değeri eklenir. Bu iki

sayının bulunduğu genler sabitlenir. Sabit olan her iki genin mutasyon aralığında kalan sağındaki ve solundaki ilk genlerin yerleri değiştirilir. Belirlenen mutasyon sayısı kadar iyileşme sağlanan tur üzerinden mutasyon gerçekleştirilir. Mutasyon işlemi tamamlandığında bulunan en kısa tur algoritmanın bulduğu sonuç olarak elde edilir.

KFA parametreleri ve algoritma adımları aşağıda verilmiştir. Tablo 4.1'de kör fare algoritmasının parametreleri gösterilmiştir.

**Tablo 4.1: Kör fare algoritmasının parametreleri**

| | |
|---|---|
| $d_{ij}$ | *i*'nci düğüm ile *j*'nci düğüm arasındaki uzaklık |
| $s_{ij}$ | *i*'nci düğümden *j*'nci düğüme gelen sinyal değeri |
| $R_{ij}$ | *i*'nci düğüm ile *j*'nci düğüm arasındaki sinyal iletim katsayısı |
| $S(m)_{ijk}$ | *m*'inci fare için o an bulunduğu *i*'nci düğüme *j*'nci düğümden ve *k*'ıncı düğümden *j*'nci düğüme gelen toplam sinyal değeri |
| $\Delta s_{ij}$ | *i-j* yolunun sinyal kayıp değeri |
| $w$ | Sinyal kayıp katsayısı |
| $v$ | Sabit sinyal ekleme değeri |
| $q$ | Geçişte arama yapılacak en yakın komşu sayısı |
| ***Maksimum sinyal artış oranı*** | Bir yol üzerinde iletilebilecek maksimum sinyal değerini sınırlar |
| ***İterasyon Sayısı*** | Maksimum tur sayısı (durdurma kriteri) |
| ***Mutasyon sayısı*** | Mutasyon işlemin kaç kez yapılacağını belirtir |
| ***Mutasyon aralığı*** | Mutasyonun gerçekleşeceği aralık |

**Algoritmanın adımları:**

***i)Her düğüme bir fare yerleştir:*** Her fare için bulunduğu düğümü ziyaret edilen düğümler listesine ekle.

***ii)Tüm (i,j) yollarına $s_{ij}$ başlangıç sinyal değerini ata:*** Tüm düğümler arasındaki yollara bir başlangıç sinyal değeri atanır. Problem simetrik olduğu için $s_{ij} = s_{ji}$'dir.

***iii)Tüm (i,j) yollarına $R_{ij}$ sinyal iletim katsayısını ata:*** Tüm düğümler arasındaki yollara bir sinyal iletim katsayısı atanır. $R_{ij}$ katsayısı $R_{min}$ ile $R_{max}$ arasından üretilen sayılardan rassal olarak atanır. Problem simetrik olduğu için $R_{ij} = R_{ji}$'dir. Doğada ses dalgaları her ortamda farklı iletilirler. Tüm düğümler arasındaki yollar farklı ortamlar olarak varsayılarak tüm yollara farklı $R_{ij}$ katsayısı atanır. Düğümler arasındaki yolların farklı ortamlar olarak düşünülmesinin sebebi farklı komşuluk kombinasyonlarının denenebilmesi içindir.

***iv)Her bir fare için gideceği sonraki 2 düğümü birlikte seç:*** *Seçilecek* sonraki 2 düğüm tüm düğüm seçenekleri yerine en yakın ***q*** komşu arasından yapılır. Seçim için;

$$S(m)_{ijk} = s_{ij} - \left(\frac{{d_{ij}}^2}{s_{ij}}\right) * R_{ij} + s_{jk} - \left(\frac{{d_{jk}}^2}{s_{jk}}\right) * R_{jk} \qquad (4.1)$$

formülü kullanılır. Her bir fare $\boldsymbol{S(m)_{ijk}}$ değeri en yüksek düğüm çiftini seçer ve fare sırasıyla *j*, *k*'ncı düğüme ilerletilir. *j* ve *k* düğümleri ziyaret edilen düğümler listesine eklenir. $\boldsymbol{S(m)_{ijk}}$ değerlerinin eşitliği durumunda *i'*nci düğüme toplam uzaklığı en küçük düğüm çifti seçilir.

***v)Tüm düğümleri ziyaret***: Tüm fareler iv adıma göre tüm düğümleri ziyaret eder. Ziyaret edilecek düğüm kalmadığında her bir fare başlangıç düğümüne döner ve turunu tamamlar.

***vi)Her bir farenin tur uzunluğunu bul:*** Her bir farenin düğümleri ziyaret sırasına göre toplam tur uzunluğu bulunur ve küçükten büyüğe sıralanır.

***vii)Tüm yollardaki sinyal kayıplarını hesapla ve sinyal değerlerini güncelle:*** Gerçek hayatta olduğu gibi algoritmada sinyallerin farklı ortamlarda ve uzaklıklarda ilerlerken soğurulduğu prensibinden hareketle tüm (i,j) yollarında mesafe ve ***w*** değerine göre sinyal kayıpları bulunur. Sinyal kayıpları her bir *(i,j)* yolu için;

$$\boldsymbol{\Delta s}_{ij} = (\frac{d_{ij}}{w}) \qquad (4.2)$$

formülü ile hesaplanır. Hesaplanan sinyal kayıpları başlangıç sinyal değerlerinden çıkarılarak *(i,j)* yollarının sinyal değerleri;

$$s_{ij} = s_{ij} - \boldsymbol{\Delta s}_{ij} \quad (4.3)$$

formülü kullanılarak güncellenir.

***viii)En kısa turu yapan farenin kullandığı rotaya sinyal eklemesi yap:*** En kısa turu yapan farenin kullandığı *(i,j)* yollarına sinyal eklemesi yapılır ve

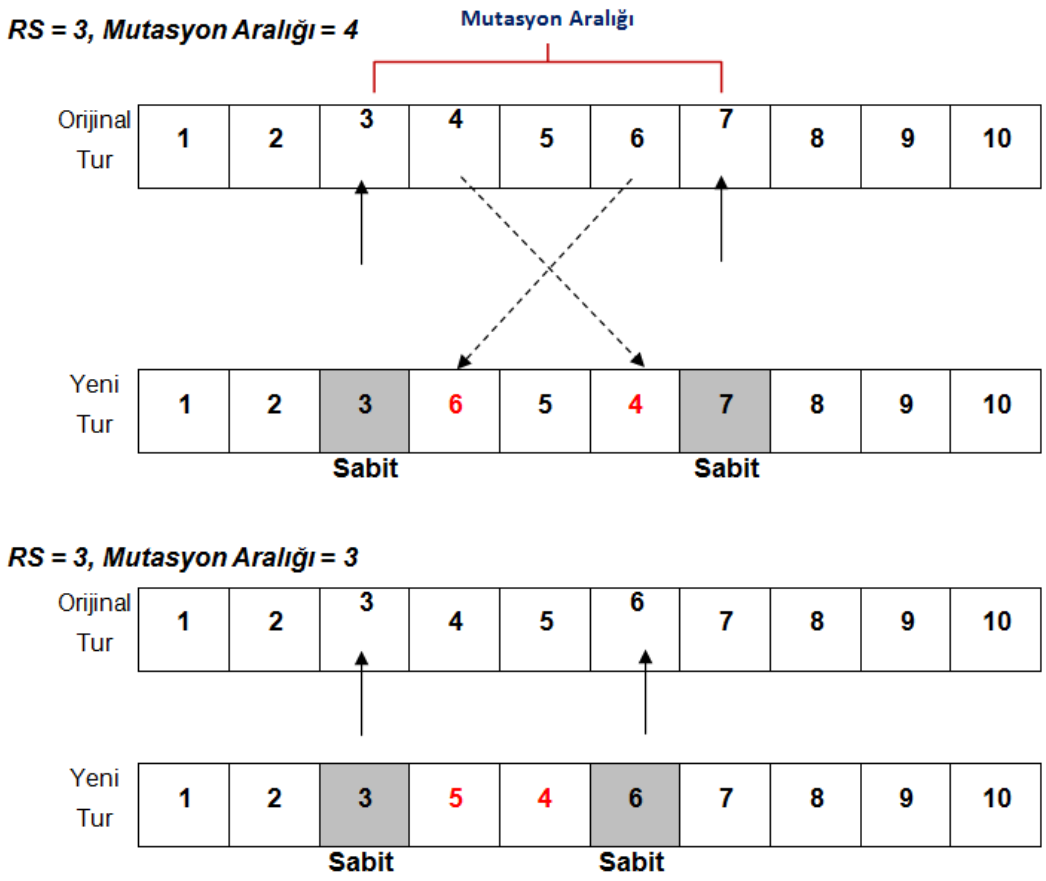
$$s_{ij\ güncel} = s_{ij} + v \quad (4.4)$$

formülüne göre sinyaller güncellenir.

***ix)Bir sonraki iterasyona geç:*** Adım ii-viii tekrarlanır. Her yeni iterasyonda bir önceki iterasyon sonunda güncellenmiş sinyal değerleri kullanılır. $R_{ij}$katsayıları her iterasyonda yeniden üretilir.

***x)Sonlandır:*** Maksimum tur sayısına ulaşıldığında algoritma durdurulur. Durdurma anına kadar bulunan en iyi tura ait düğüm sıralaması mutasyona aktarılır.

***xi)Mutasyon*:** Kör farelerin yaşamları boyunca kansere dirençli oldukları bilgisinden hareketle daha iyi bir çözüm bulabilmek için algoritmanın bulduğu en iyi tur mutasyona tabi tutulur. Mutasyon için en iyi tur kromozom yapısına dönüştürülür. Kromozom uzunluğu içinden bir rassal sayı (RS) üretilir. Bu rassal sayıya mutasyon aralığı değeri eklenir. Bu iki sayının bulunduğu genler sabitlenir. Sabit olan her iki genin mutasyon aralığında kalan sağındaki ve solundaki ilk genlerin yerleri değiştirilir. Mutasyon aralığına göre farklı mutasyon örnekleri Şekil 4.3'de gösterilmiştir.




**Şekil 4.3: Mutasyon örnekleri**

Kör fare algoritmasının akış şeması Şekil 4.4'te verilmiştir.

Algoritma çalıştırılmadan önce düğümler arasındaki mesafe matrisi ve yakınlık matrisi oluşturulur. Düğümler arasındaki mesafe düğümlerin konumları arasındaki öklidyen uzaklık formülüne göre hesaplanır. Her bir düğüme ait x ve y koordinatları kullanılarak $i\ (x_i, y_i)$ $ve\ j\ (x_j, y_j)$noktaları arasındaki uzaklık

$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$formülüne göre hesaplanır.

Algoritmanın ***iv.*** adımında her bir fare bir sonraki gideceği düğüm çiftini seçerken en yakın ***q*** komşu arasından seçim yapar. Eğer şebekedeki tüm olası düğüm çiftleri arasından seçim yapılırsa işlem miktarı çok fazla miktarda

artacaktır. Örneğin 50 şehirli bir problemde ilk düğüm çiftinin seçiminde bir fare için denenmesi gereken sıralamaların sayısı 49'un 2'li permütasyonu kadardır.

$P(49,2) = 2352$

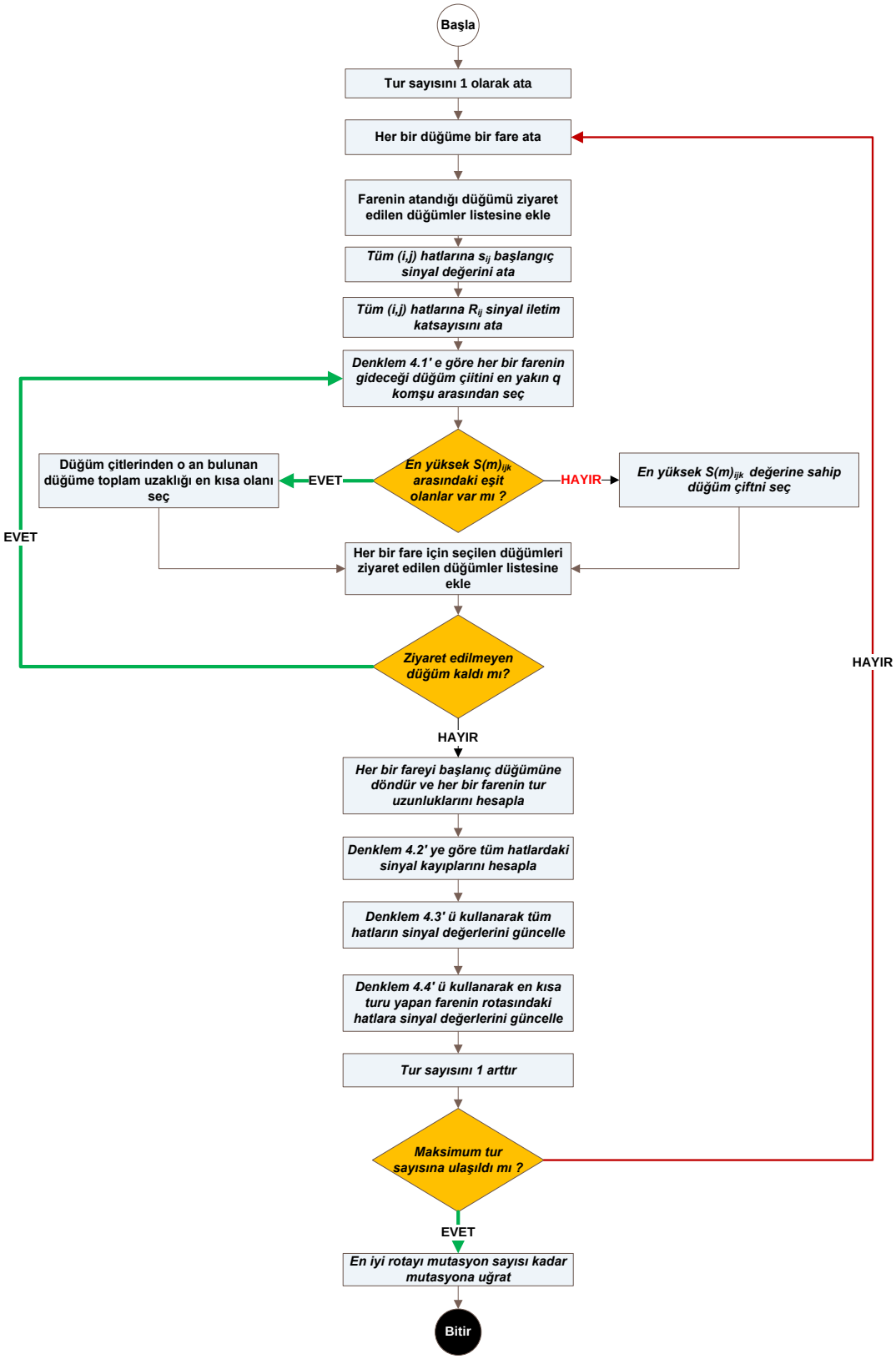
Her bir düğümde bir fare olduğu için 50 şehirli bir problemde sadece ilk düğüm çiftinin seçiminde tüm seçimler için 50*2352 = 117600 işlem yapılır.

Bunun yerine en yakın $\boldsymbol{q}$ komşu arasından seçim yapıldığı takdirde ($\boldsymbol{q}$ = 5 olsun);

$P(5,2) = 20$ olur.

Her bir düğümde bir fare olduğu için 50 şehirli bir problemde ilk düğüm çiftinin seçiminde tüm seçimler için 50*20 = 1000 işlem yapılır.

Görüldüğü gibi en yakın $\boldsymbol{q}$ komşu arasından seçim yapıldığında yapılan işlem sayısı yaklaşık 118'de 1'ine düşmektedir. Her düğüm çifti seçiminden sonra tüm farelere ait yakınlık matrisi ziyaret edilen düğümler matristen çıkartılarak güncellenir.




**Şekil 4.4: Kör fare algoritması akış şeması**

### 4.3 Kör Fare Algoritmasının Gelişim Süreci

Algoritmanın son halini alana kadar test problemleri üzerinde pek çok deneme yapılarak en iyi çözüme elde etmek ve çözüm süresinin azaltılması konusunda oldukça önemli adımlar elde edilmiştir. Krolonojik sıraya göre algoritma versiyonları aşağıda detaylı olarak anlatılmıştır.

**Versiyon 1:**

Algoritmanın ilk halidir.

**1**.Her bir düğüm çifti için başlangıç sinyal değeri ata,

$s_{ij} = 300$

$s_{ij}$ : *i*'nci düğümden *j*'nci düğüme gelen sinyal değeri *i,j*=1'den n'e kadar; *n:*toplam düğüm sayısı

**2.**Her bir düğüme bir fare yerleştir, *m*=1'den *n*'e kadar; *m=n*

**3**.Farenin bulunduğu düğümü farenin tabu listesine ekle, ve tur uzunluk değeri olarak $L_m = 0$ ata, $L_m$:*m*'inci farenin tur uzunluğu;

**4**.Her bir fare için bulunduğu düğümden itibaren hareket etmek için sonraki iki en yakın düğümden gelen sinyal değerlerini hesapla,

$$S(m)_{ijk} = s_{ij} - \left(\frac{d_{ij}}{300}^2\right) + s_{jk} - \left(\frac{d_{jk}}{300}^2\right)$$

$S(m)_{ijk}$ : *m*'inci fare için o an bulunduğu *j*'nci düğümden *i*'nci düğüme ve *k*'ıncı düğümden *j*'nci düğüme gelen toplam sinyal değeri

**5**.Her bir fare için ikili toplam sinyal değeri ($S(m)_{ijk}$) en yüksek olanı seç ve fareyi sırasıyla *i*'nci düğümden *j*'nci düğüme ve *j*'nci düğümden *k*'ncı düğüme ilerlet, bu düğümleri*(i,j ve k*) farenin tabu listesine ekle, 3 düğüm arasındaki mesafeyi ($dij+d_{jk}$), farenin tur uzunluğuna ekle.

$$L_m = L_m + d_{ij} + d_{jk}$$

***Eğer başlangıç düğümü dışında ikili düğüm kalmadı ise kalan düğümü seç*

**6**.Her bir fare için tüm düğümler ziyaret edilene kadar adım 4, 5 tekrarla.

**7**.En son düğüm ziyaret edildiğinde başlangıç düğümüne döndür ve bu mesafeyi tur uzunluğuna ekle.

**8**.Her bir fare için turu tamamla ve tur uzunluklarından en küçüğünü bul.

$\boldsymbol{L_{eniyi}}$ **:** En kısa turu yapan farenin tur uzunluğu

**9**.Farelerin tur esnasında kullanmış olduğu *i-j* yollarında sinyal zayıflama değerlerini hesapla

$$\boldsymbol{\Delta s}_{ij} = (d_{ij}/c) * w$$

$\Delta s_{ij}$ : *i-j* yolunun sinyal zayıflama değeri

***c:*** her birim uzunluk başına sinyalin zayıflatılacağını belirten katsayı

***w:*** Sinyal zayıflatma katsayısı

Her c=3 birim uzunluk başına w=0,03

Her c=4 birim uzunluk başına w=0,06

Her c=5 birim uzunluk başına w=0,09

Kullanılmayan yolların sinyal değerlerinde zayıflatma yapılmaz.

**10**. En iyi turu yapan farenin tur uzunluğuna bağlı olarak farelerin tur uzunluğu ile ters orantılı olarak tüm kullanılan *i-j* yolları için sinyal güncellemesi yap. Başka bir deyişle tüm fareler tur uzunluklarına bağlı olarak kullandıkları *i-j* yoluna sinyal eklemesi yapar. Tüm fareler sinyal eklemesi yaptıktan sonra *i-j* yolunun tur sonundaki sinyal değeri elde edilir.

$$\boldsymbol{s_{ij}} = \left(\boldsymbol{s_{ij}} - \boldsymbol{\Delta s_{ij}}\right) + \left(\frac{\boldsymbol{L_{eniyi}}}{\boldsymbol{L_m}}\right) * \boldsymbol{v} \qquad \text{m=1'den n'e kadar}$$

***v:*** Sabit değer

**11**.Durdurma kriteri sağlanana kadar adım 1 -7 yi tekrarla.

Durdurma kriteri: Maksimum tur sayısı

**Versiyon 2:**

Bu versiyonda versiyon 1'in 4'ncü adımındaki sinyal değerlerinin güncellenmesinde kullanılan formül değiştirilmiştir. Eski formülde sinyal değeri güncellenirken sabit 300'e bölünüyordu yeni formülde mevcut sinyal değerlerine bölünmüştür. Aşağıda eski ve yeni formüller verilmektedir

$$S(m)_{ijk} = s_{ij} - \left(\frac{{d_{ij}}^2}{300}\right) + s_{jk} - \left(\frac{{d_{jk}}^2}{300}\right)$$

formülü;

$$S(m)_{ijk} = s_{ij} - \left(\frac{{d_{ij}}^2}{s_{ij}}\right) + s_{jk} - \left(\frac{{d_{jk}}^2}{s_{jk}}\right)$$

olarak değiştirildi.

Bu değişiklik ikili sinyal değeri hesabında her tur sonundaki güncel sinyal değerlerinin kullanılmasını sağlamak amacıyla yapılmıştır.

**Versiyon 3:**

Versiyon 1'deki kullanılmayan yolların sinyal değerlerinde zayıflatma yapılmaz şeklinde olan 9'ncu adım tüm yollarda sinyal değerleri zayıflatılır olarak değiştirilmiştir. Bu değişiklik yerel en en iyilere takılmayı önlemek için yapılmıştır.

**Versiyon 4:**

Versiyon 1'de 4'ncü adımda karşılaştırılan $\boldsymbol{S(m)_{ijk}}$ değerlerinde eşitlik söz konusu olduğunda *i-j-k* yolu rassal olarak seçilmiştir.

Daha önceki versiyonlarda tüm farelerin sinyal eklemesine izin verilirken bu versiyonda sadece en iyi turu yapan farenin sinyal eklemesine izin verilmiştir. Bunun sebebi bazı yollarda sinyal değerleri diğer yollara göre yaklaşık 100 kat daha fazla artmıştır. Bu durum sürekli aynı rotanın kullanılmasına neden olduğu için algoritma yerel minimuma takılmıştır.

**Versiyon 5:**

Versiyon 1'de 4'ncü adımındaki

$$S(m)_{ijk} = s_{ij} - \left(\frac{{d_{ij}}^2}{s_{ij}}\right) + s_{jk} - \left(\frac{{d_{jk}}^2}{s_{jk}}\right)$$

olan gidilecek bir sonraki düğüm çiftini belirleme formülünde yeniden bir değişiklik yapılarak yansıma değeri eklenmiştir.

$S(m)_{ijk} = s_{ij} - \left(\frac{{d_{ij}}^2}{s_{ij}}\right) * R_{ij} + s_{jk} - \left(\frac{{d_{jk}}^2}{s_{jk}}\right) * R_{jk}$ olarak değiştirildi.

$R_{ij}$: *i-j* bağlantısı için rassal olarak belirlenen sinyal yansıma değeridir. Bu çarpanın eklenmesinin sebebi ses dalgalarının farklı ortamlarda farklı olarak yansımasıdır. Çözüm uzayındaki farklı alanlarındaki çözümlerin araştırılması amacı ile problemde her bir yolun birbirinden farklı ortamlar olduğu varsayılmıştır. $R_{ij}$ değerleri $R_{min}$ ve $R_{max}$ olarak belirlenen iki parametre aralığından üretilmiştir.

**Versiyon 6:**

Bazı yollarda sinyal değerleri çok büyük değerlere ulaştığı için maksimum sinyal değeri kısıtlaması eklenmiştir. Sinyal değerlerinin çok büyümesi erken yakınsamaya neden olmaktadır. Maksimum sinyal değeri belirlenirken aşağıdaki formül kullanılmıştır.

Maksimum sinyal = $s_{ij}$ (1 + S.A.O)

S.A.O = Sinyal Artış Oranı

S.A.O 0 ile 1 arasında bir sayıdır. Herhangi bir yolda maksimum değere ulaşan sinyal değeri maksimum değere sabitlenmektedir.

**Versiyon 7:**

Bir farenin bulunduğu düğümden sonra gideceği iki düğüm belirlenirken tümü düğümler arasından seçim yapılmaktaydı. Bunun yerine bir düğüme ***q*** komşu arasından seçim yapılması algoritmaya eklenmiştir. Bu durum algoritmanın hızını oldukça arttırmıştır.

Örneğim 100 düğümlü bir problemde bir farenin bir sonraki aşamada gidebileceği iki düğüm için $P(99,2) = 9702$ farklı seçenek söz konusu iken en yakın 5 komşu arasından seçim yapılırsa seçenek sayısı $P(5,2) = 20$ olmaktadır.

**Versiyon 8:**

Daha önce kullanılan ***c*** parametresi algoritmadan çıkartılarak ***w*** parametresi ***c*** ve ***w*** parametrelerinin yerine kullanılmıştır.

Eğer verilen bir İterasyon sayısı boyunca iyileşme sağlanmazsa sinyal değerlerinin güncellenmesi (poking) yapılmıştır.

Örneğin 20 iterasyon boyunca rota uzunluğunda bir iyileşme sağlanmazsa 20'nci İterasyon sonunda yollardaki sinyal değerlerinin ortalaması ve standart sapması bulunur. Yollardaki sinyal değerleri Ortalama + 1 standart sapma, Ortalama +2 standart sapma, Ortalama + 3 standart sapma ve ortalama altı grupları aralıklarına göre listelenir. Aşağıdaki parametrelerden değerleri 1 olan gruptaki sinyaller o andaki maksimum sinyal değerine eşitlenir. Ortalama altında kalan sinyal değerleri applyAvgMinusRatio oranına göre arttırılır. Bu yöntemle en iyi sonuçlar elde edilememiştir.

**Versiyon 9:**

Sinyal ekleme yöntemi yolların kullanım sıklığı dikkate alınarak değiştirildi. Böylece daha sık kullanılan yolların sinyal değerlerin arttırılmıştır. Bunun için aşağıdaki formül kullanılmıştır.

$$\text{Sinyal artışı} = \left(\frac{f_{ij}}{n}\right)(v)$$

Burada $f_{ij}$ (i,j) yolunun kullanım sıklığı, $n$ şehir sayısı, $v$ sabit değerdir. Şekil 4.5'te sinyal artışı gösterilmektedir.

| | Round 0 used nodes | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 0 | 10 | 0 | 0 | 1 | 5 | 0 | 0 | 0 | 4 |
| 2 | 10 | 0 | 1 | 0 | 1 | 5 | 0 | 0 | 0 | 3 |
| 3 | 0 | 1 | 0 | 4 | 1 | 0 | 2 | 3 | 3 | 6 |
| 4 | 0 | 0 | 4 | 0 | 9 | 0 | 4 | 3 | 0 | 0 |
| 5 | 1 | 1 | 1 | 9 | 0 | 7 | 0 | 1 | 0 | 0 |
| 6 | 5 | 5 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 3 |
| 7 | 0 | 0 | 2 | 4 | 0 | 0 | 0 | 4 | 10 | 0 |
| 8 | 0 | 0 | 3 | 3 | 1 | 0 | 4 | 0 | 6 | 3 |
| 9 | 0 | 0 | 3 | 0 | 0 | 0 | 10 | 6 | 0 | 1 |
| 10 | 4 | 3 | 6 | 0 | 0 | 3 | 0 | 3 | 1 | 0 |

n:Toplam düğüm sayısı

Kullanım sayısı 0(sıfır) olanlar dikkate alınmayacak

ks [i][j] : *i-j* hatının kullanım sayısı

Increase signal: signal[i][j] += (ks [i][j]/ n) * v;

ÖRNEK: ks [5][6] =7

n=10 signal[5][6] =500

signal[5][6] += (7/ 10) * 12

signal[5][6] = 508,4

signal[1][6] += (5/ 10) * 12

signal[1][6] = 506

**Şekil 4.5: Kullanım sayısına göre sinyal ekleme**

**Versiyon 10:**

Sinyal güncellemde kullanım sıklığına ek olarak en iyi turu yapan farenin rotasında yer alan yolların sinyal değerleri daha fazla arttırılarak bu yolların daha sık tercih edilmesi amaçlanmıştır. Bu amaçla yeni bir parametre tanımlanmıştır.

*r_ij*'lerin rassal başlaması başlangıç çözümünü yukarı çektiği için *r_ij* çarpanının ilk turdaki etkisini ortadan kaldırmak amacı ile *r_ij* 'lere 1'nci tur için 1(bir) değeri atanmıştır. Sonraki turlar için rassal üretmeye devam edilmiştir.

**Versiyon 11:**

Farenin o an bulunduğu düğümden bir sonraki gideceği düğüm çifti seçimi yerine 3'lü ve 4'lü seçim yapabilmesi sağlanmıştır. 3'lü ve 4'lü seçim rota uzunluklarını arttırmıştır.

**Versiyon 12:**

İterasyonlar sonucu elde edilen en iyi çözümü iyileştirmek amacı ile algoritmaya mutasyon operatörü eklenmiştir. Mutasyon için iterasyonlar sonucu elde edilen en iyi turun sıralaması genetik gösterime dönüştürülür. Kromozom uzunluğu içinden bir rassal sayı (RS) üretilir. Bu rassal sayıya mutasyon aralığı değeri eklenir. Bu iki sayının bulunduğu genler sabitlenir. Sabit olan her iki genin mutasyon aralığında kalan sağındaki ve solundaki ilk genlerin yerleri değiştirilir.

**Versiyon 14:**

Mutasyon işlemini tüm iterasyonlar bittikten sonra yapmak yerine her İterasyon sonunda en iyi tur sonucuna uygulama gerçekleştirilmiştir. Ancak çözüm süresi uzamış ve iyileşme sağlanamamıştır.

### 4.4 Deneysel Sonuçlar

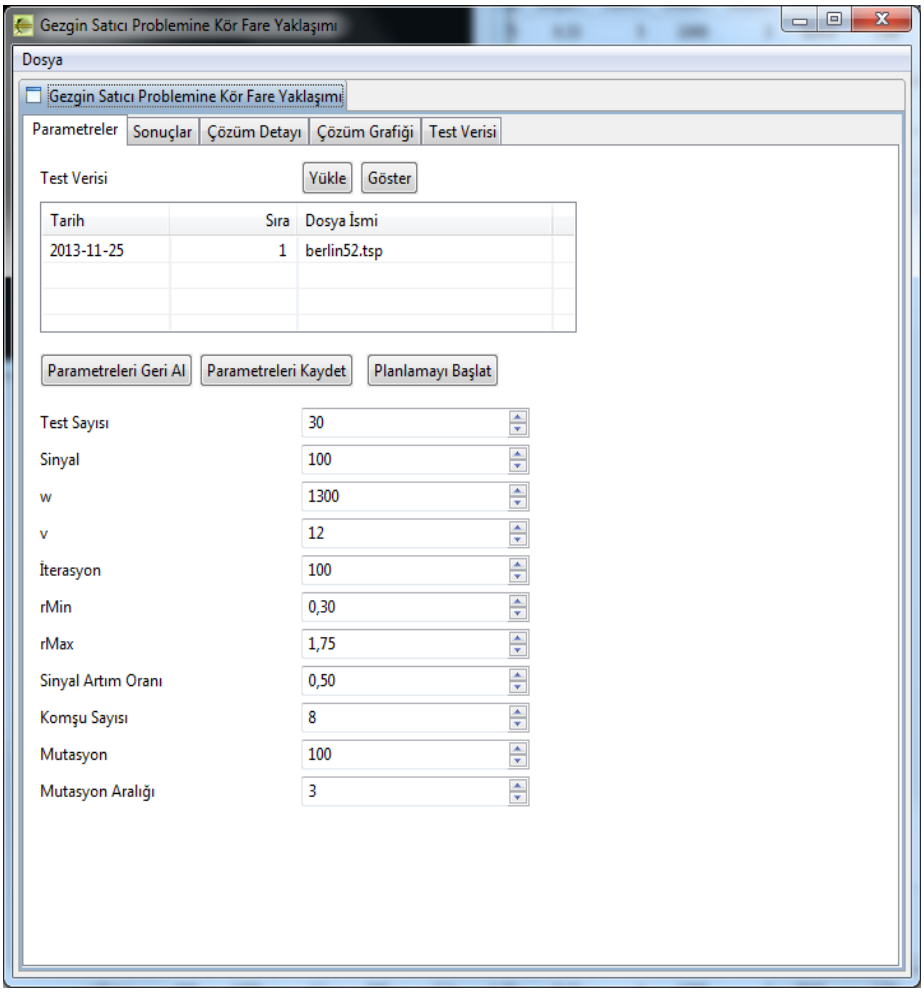
Bu bölümde SGSP için yapılan testlerin sonuçları, parametrelerin analizi ile ilgili yapılan testler ayrıntılı olarak anlatılmıştır. Testler için kullanılan veriler GSP için veri sunan internet sitelerinden[8] alınmıştır.

Bu çalışmada *wi29, dj38, eil51, berlin52, st70, eil76, eil101* test veri setleri kullanılmıştır. Kullanılan veri setleri EUC_2D normundadır. EUC_2D normundaki veriler öklidyen uzaklık formülüne uyan verilerden oluşur. Algoritmanın test edilmesi için java programlama dili ile bir yazılım

[8] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/

http://www.math.uwaterloo.ca/tsp/world/countries.html

geliştirilmiştir. Test işlemleri Intel Dual Core 3.33 GHZ, 4GB RAM özelliklerine sahip bir bilgisayarda gerçekleştirilmiştir. Yazılım arayüzü Şekil 4.6'da gösterilmiştir. Yazılıma ait diğer arayüzler ve yazılımda kullanılan değişken fonksiyonlar EK.A'da sunulmuştur.




**Şekil 4.6: Geliştirilen yazılım arayüzü**

Deneylerde kullanılacak en iyi parametre seti kombinasyonunun bulunması için deneysel tasarım çalışması gerçekleştirilerek sonuçları analiz edilmiştir. En iyi kombinasyonu bulmak için iki farklı yaklaşım söz konusudur. Birinci yaklaşımda, bir parametre değiştirilirken diğerleri sabit tutulur. İkinci yaklaşım ise, parametrelerin farklı değerlerinin birbiriyle farklı kombinasyonlarda birleştirilmesi ile ortaya çıkan sonuçların karşılaştırılmasına dayalıdır [119].

Geliştirilen algoritmada 10 farklı parametre ve her parametrenin iki seviyesi için birinci yaklaşımla uygun parametre seti belirlenmek istenseydi $2^{10}$ =

1024 adet test gerçekleştirilmesi gerekirdi. Bu deneyin en az 30 kere tekrarlanması gerekliliği düşünülürse 1024x30 = 30720 adet yapılması gerekliliği sebebiyle uygun parametre setinin belirlenmesi için ikinci yaklaşım benimsenmiştir. Deneysel tasarım için belirlenen faktörler ve seviyeleri Tablo 4.2'de gösterilmiştir.

**Tablo 4.2: Deney tasarımında kullanılan faktörler ve seviyeleri**

| | **Seviyeler** | |
|---|---|---|
| **Faktörler** | **1.Seviye** | **2.Seviye** |
| *Başlangıç Sinyali($s_{ij}$)* | 100 | 500 |
| *w değeri* | 20 | 1300 |
| *v* | 2 | 12 |
| $R_{min}$ | 0,30 | 0,95 |
| $R_{max}$ | 1,05 | 1,75 |
| *Sinyal Artım Oranı* | 0,33 | 1,00 |
| *Komşu Sayısı(q)* | 4 | 8 |
| *Mutasyon Aralığı* | 3 | 6 |
| *Mutasyon Sayısı* | 50 | 200 |
| *İterasyon Sayısı* | 100 | 300 |

Tablo 4.2'de verilen 10 adet parametre setinin tüm kombinasyonlarının ortalama bir test probleminde denenmesi (bir test için ortalama 50 saniye) 30.720x50=1536000 saniye =25600 dakika = 426 = 18 gün sürecektir. 7 farklı veri seti için 18x7=126 güne ihtiyaç duyulacaktır. Deney sayısını azaltmak için Taguchi deney tasarımı ile uygun parametre seti belirleme çalışması gerçekleştirilmiştir.

Minitab programı ile 10 faktör 2 seviye parametre testi için L32 hesap tablosu kullanılarak yapılan deneyler sonucunda elde edilen ortalama tur değerleri ve "en küçük en iyi" durumu için hesaplanan S/N(signal to noise) oranları EK.B'de yer almaktadır. En iyi tur uzunluğu için hesaplanan S/N oranı ve ortalamalar için varyans analizi sonuçları Tablo 4.3'de ve Tablo 4.4'te sunulmaktadır. S/N oranları göz önüne alındığında %95 güven düzeyinde sinyal, w, Rmax ve Mutasyon aralık faktörlerinin sonuçlar üzerinde etkili faktörler ($p<0.05$) olduğu görülmektedir. Tablo 4.5'da gösterilen S/N oranları yanıt tablosuna göre faktörlerin en iyi kombinasyonu Tablo 4.6'da gibi elde edilmiştir. Parametrelere ait S/N oranı grafikleri Şekil 4.7'de gösterilmiştir.

**Tablo 4.3: S/N oranları için tahmini model katsayıları**

| Term | Coef | SE Coef | T | P |
|---|---|---|---|---|
| Constant | -78,1526 | 0,03559 | -2195,775 | 0,000 |
| Sinyal 100 | -0,0859 | 0,03559 | -2,413 | 0,025 |
| w 20 | -0,0922 | 0,03559 | -2,592 | 0,017 |
| v 2 | -0,0192 | 0,03559 | -0,539 | 0,596 |
| İterasyo 100 | 0,0068 | 0,03559 | 0,192 | 0,850 |
| Rmin 0,30 | 0,0360 | 0,03559 | 1,012 | 0,323 |
| Rmax 1,05 | -0,0934 | 0,03559 | -2,624 | 0,016 |
| SAO 0,33 | 0,0360 | 0,03559 | 1,012 | 0,323 |
| q 4 | -0,0135 | 0,03559 | -0,380 | 0,708 |
| Mut.Sayı 50 | 0,0289 | 0,03559 | 0,811 | 0,426 |
| Mut.Aral 3 | 0,0914 | 0,03559 | 2,569 | 0,018 |

S = 0,2013  R-Sq = 58,2%  R-Sq(adj) = 38,3%

Taguchi deney tasarımı ile yapılan testlerin %5 anlam seviyesinde ANOVA analizi sonuçlarına göre parametrelerin performans ölçütü üzerindeki etkileri aşağıda incelenmiştir.

A. Sinyal değeri: Sinyal değeri için p değeri 0,025 < 0,05 olduğu için en kısa mesafeli turu bulma üzerinde *sinyal değerinin* etkili olduğunu söyleyebiliriz.
B. *w* (Sinyal kayıp katsayısı): 0,017 < 0,05 olduğu için *w* parametresinin en kısa mesafeli turu bulma üzerinde etkisi olduğunu söyleyebiliriz..
C. *v* (Sabit sinyal ekleme değeri): 0,596 > 0,05 olduğu için *v* parametresinin en kısa mesafeli turu bulma üzerinde etkisi olmadığını söyleyebiliriz.
D. İterasyon sayısı: 0,850 > 0,05 olduğu için *iterasyon sayısı* parametresinin en kısa mesafeli turu bulma üzerinde etkili olmadığını söyleyebiliriz.
E. $R_{min}$: 0,323 > 0,05 olduğu için $R_{min}$ parametresinin en kısa mesafeli turu bulma üzerinde etkisi olmadığını söyleyebiliriz.
F. $R_{max}$: 0,016< 0,05 olduğu için $R_{max}$ parametresinin en kısa mesafeli turu bulma üzerinde etkisi olduğunu söyleyebiliriz.

G. SAO (Sinyal Artış Oranı): 0,323 > 0,05 olduğu için *SAO* parametresinin en kısa mesafeli turu bulma üzerinde etkisi olmadığını söyleyebiliriz.

H. *q* (Komşu sayısı): 0,708 > 0,05 olduğu için *q* parametresinin en kısa mesafeli turu bulma üzerinde etkisi olmadığını söyleyebiliriz.

İ. Mutasyon Sayısı: 0,426 > 0,05 olduğu için *mutasyon sayısı* parametresinin en kısa mesafeli turu bulma üzerinde etkisi olmadığını söyleyebiliriz.

J. Mutasyon Aralığı: 0,018 < 0,05 olduğu için *mutasyon aralığı* parametresinin en kısa mesafeli turu bulma üzerinde etkisi olduğunu söyleyebiliriz.

**Tablo 4.4: Ortalama değerler için varyans analizi**

| Source | DF | Seq SS | Adj SS | Adj MS | F | P |
|---|---|---|---|---|---|---|
| Sinyal | 1 | 209790 | 209790 | 209790 | 5,93 | 0,024 |
| w | 1 | 230012 | 230012 | 230012 | 6,51 | 0,019 |
| v | 1 | 11666 | 11666 | 11666 | 0,33 | 0,572 |
| İterasyon | 1 | 1391 | 1391 | 1391 | 0,04 | 0,845 |
| Rmin | 1 | 36518 | 36518 | 36518 | 1,03 | 0,321 |
| Rmax | 1 | 242730 | 242730 | 242730 | 6,87 | 0,016 |
| SAO | 1 | 34782 | 34782 | 34782 | 0,98 | 0,333 |
| q | 1 | 4118 | 4118 | 4118 | 0,12 | 0,736 |
| Mut.Sayısı | 1 | 22419 | 22419 | 22419 | 0,63 | 0,435 |
| Mut.Aralık | 1 | 235470 | 235470 | 235470 | 6,66 | 0,017 |
| Residual Error | 21 | 742423 | 742423 | 35353 | | |
| Total | 31 | 1771318 | | | | |

**Tablo 4.5: En küçük en iyi durumu için S/N oranı yanıt tablosu**

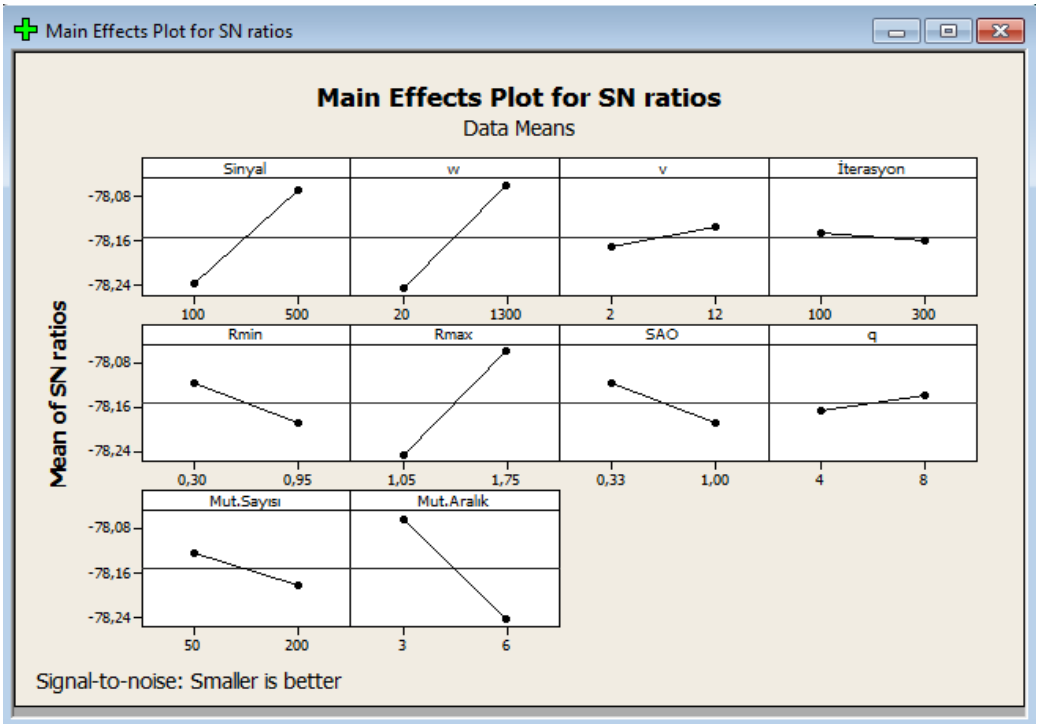
Response Table for Signal to Noise Ratios
Smaller is better

| Level | Sinyal | w | v | İterasyon | Rmin | Rmax | SAO | q |
|---|---|---|---|---|---|---|---|---|
| 1 | -78,24 | -78,24 | -78,17 | -78,15 | -78,12 | -78,25 | -78,12 | -78,17 |
| 2 | -78,07 | -78,06 | -78,13 | -78,16 | -78,19 | -78,06 | -78,19 | -78,14 |
| Delta | 0,17 | 0,18 | 0,04 | 0,01 | 0,07 | 0,19 | 0,07 | 0,03 |
| Rank | 4 | 2 | 8 | 10 | 6 | 1 | 5 | 9 |

| Level | Mut.Sayısı | Mut.Aralık |
|---|---|---|
| 1 | -78,12 | -78,06 |
| 2 | -78,18 | -78,24 |
| Delta | 0,06 | 0,18 |
| Rank | 7 | 3 |

**Tablo 4.6: Taguchi deney tasarımı sonucu en uygun parametre değerleri**

| **Parametre** | **$S_{ij}$** | **$R_{ij}$(min-max)** | **w** | **v** | **q** | **S.A.O** | **İterasyon Sayısı** | **Mutasyon Sayısı** | **Mutasyon Aralığı** |
|---|---|---|---|---|---|---|---|---|---|
| Değer | 500 | 0.30-1.75 | 1300 | 12 | 8 | 0.33 | 100 | 50 | 3 |




**Şekil 4.7: S/N oranı grafikleri**

Deney tasarımı sonucuna göre bulunan en uygun parametre değerleri ile test veri setleri için 30'ar defa deney yapılmış ve Tablo 4.7'deki sonuçlar elde edilmiştir. *wi29, dj38, eil52* veri setleri için en iyi sonuçlar elde edilirken *eil51,st70, eil76 ve eil101* için sırasıyla en iyiden %3.2, %6.7, %6.7 ve %24.8

sapma ile sonuçlar elde edilmiştir. Tablodaki süre değerleri algoritmanın bulduğu en iyi sonucun elde edildiği deneyin çalışma süresidir. Test problemlerine ait çözüm grafikleri ve en iyi yol çizimi EK C'de yer almaktadır.

**Tablo 4.7: Veri setleri için sonuçlar**

| *Veri Seti* | Düğüm Sayısı | Deney Sayısı | Bilinen En İyi Sonuç | KFA ile Bulunan En İyi | KFA ile Bulunan Ortalama | KFA ile Bulunan En Kötü | Bulunan en iyi sonuç sayısı /Deney Sayısı | Sapma (%) | Süre (sn) |
|---|---|---|---|---|---|---|---|---|---|
| ***wi29*** | 29 | 30 | 27603 | 27603 | 28121 | 29074 | 1/30 | 0 | 1,09 |
| ***dj38*** | 38 | 30 | 6656 | 6656 | 6706 | 6847 | 5/30 | 0 | 1,99 |
| ***eil51*** | 51 | 30 | 426 | 440 | 464 | 491 | 0/30 | 3,2 | 4,68 |
| ***berlin52*** | 52 | 30 | 7542 | 7542 | 7861 | 8264 | 1/30 | 0 | 5,57 |
| ***st70*** | 70 | 30 | 675 | 720 | 775 | 836 | 0/30 | 6,7 | 15,04 |
| ***eil76*** | 76 | 30 | 538 | 574 | 592 | 622 | 0/30 | 6,7 | 20,45 |
| ***eil101*** | 101 | 30 | 629 | 785 | 829 | 877 | 0/30 | 24,8 | 63,86 |

Yapılan deneyler geliştirilen mutasyon işleminin etkili olduğunu göstermiştir. İterasyon sonucu tur değerleri ile mutasyon sonucu tur değerleri karşılaştırıldığında 1322 deneyin 1087 tanesinde mutasyon işlemi iyileşme sağlamıştır. Mutasyon işlemi ile sağlanan iyileşme oranı %82'dir. Örneğin *wi29* test problemi için en iyi parametre değerleri ile yapılan 30 deneyin 25'inde (Tablo 4.8) mutasyon işlemi ile farklı miktarlarda iyileşme elde edilmiştir. Bu deney seti için mutasyon işleminin sağladığı iyileşme oranı %83,3'tür.

**Tablo 4.8: İterasyonu sonucu ve mutasyon sonucu değerler**

| Deney No | İterasyon Sonucu | Mutasyon Sonucu | İyileşme Miktarı | Deney No | İterasyon Sonucu | Mutasyon Sonucu | İyileşme Miktarı |
|---|---|---|---|---|---|---|---|
| 1 | 28615 | 28602 | 13 | 16 | 27946 | 27944 | 2 |
| 2 | 27750 | 27750 | 0 | 17 | 28557 | 28386 | 171 |
| 3 | 28617 | 27750 | 867 | 18 | 28386 | 28386 | 0 |
| 4 | 28887 | 28602 | 285 | 19 | 28557 | 28386 | 171 |
| 5 | 28790 | 28651 | 139 | 20 | 27878 | 27750 | 128 |
| 6 | 27750 | 27750 | 0 | 21 | 28386 | 28386 | 0 |
| 7 | 29128 | 29074 | 54 | 22 | 27946 | 27944 | 2 |
| 8 | 28074 | 27944 | 130 | 23 | 27616 | 27603 | 13 |
| 9 | 28074 | 27944 | 130 | 24 | 28388 | 28239 | 149 |
| 10 | 28387 | 28386 | 1 | 25 | 28362 | 28226 | 136 |
| 11 | 28074 | 27944 | 130 | 26 | 27946 | 27944 | 2 |
| 12 | 28752 | 28031 | 721 | 27 | 28478 | 28386 | 92 |
| 13 | 27946 | 27944 | 2 | 28 | 27770 | 27750 | 20 |
| 14 | 27878 | 27750 | 128 | 29 | 27750 | 27750 | 0 |
| 15 | 27946 | 27944 | 2 | 30 | 28634 | 28497 | 137 |

# 5. SONUÇ VE ÖNERİLER

Bu tez çalışması kapsamında literatürde üzerinde en çok çalışılmış olan kombinatoryel optimizasyon problemlerinden GSP'nin çözümü için bir meta-sezgisel yöntem önerilmiştir. GSP'de amaç başlangıç ve bitiş şehri aynı olan ve tüm şehirlerin sadece bir kez ziyaret edildiği en kısa mesafeli turu bulmaktır. Geliştirilen yöntem kör farelerin doğadaki davranışlarından ve engelleri geçme stratejilerinden esinlenilerek tasarlanmış ve kör fare algoritması adı verilmiştir. Bu yöntem yeni bir meta-sezgisel olma özelliği taşımaktadır. Bu sezgisel anlaşılması kolay, hesaplama adımları basit ve sadedir. Ayrıca algoritma içinde kullanılan mutasyon operatörü de daha önce literatürde bulunmayan yeni bir operatördür.

Geliştirilen algoritmanın başarısını göstermek için literatürde yer alan farklı test problemleri üzerinde algoritma uygulanmıştır. Bunu için öncelikle KFA'da kullanılan parametrelerin en iyi değerleri Taguchi L32 tasarımı kullanılarak belirlenmiştir. Algoritma belirlenen parametre seti kullanılarak yedi farklı test probleminde uygulanmış ve oldukça iyi sonuçlar elde edilmiştir.veri seti için deneyler gerçekleştirilmiştir. Test problemlerinde şehir sayısı 75 olan problemler için en iyi ve en iyiye yakın çözümler elde edilmiştir. Fakat şehir sayısı 101 olan problem için bulunan çözüm en iyiden uzaklaşmıştır. Veri seti boyutunun artması problemin çözüm süresini de arttıran bir unsur olarak tespit edilmiştir. Ayrıca geliştirilen mutasyon operatörünün etkili bir öperatör olduğu görülmüştür. Yapılan deneylerde bulunan çözümlerin %82'sinde bu operatör uygulanarak daha iyi bir çözüm elde edilmiştir.

Bu algoritma geliştirmeye açık bazı konular içermektedir. Bu çalışmada sinyal kayıp katsayısı statik olarak ele alınmıştır. Fakat düğümler arasındaki mesafe dikkate alınarak sinyal kayıp katsayısı dinamik bir şekilde belirlenirse daha iyi sonuçlar elde edilebilir. Düğüm çifti seçiminde kullanılan strateji iyileştirilerek daha kısa sürede çözümler elde edilebilir.

Yapılan bu çalışmada kullanılan düğüm çifti seçimi, en yakın komşu sayısı parametresine göre arama, yeni mutasyon operatörü diğer sezgisellere entegre edilebilir veya yöntem diğer sezgisellerle hibrid yapıda tekrar tasarlanabilir.

SGSP için geliştirilen bu yöntem farklı tiplerdeki GSP problemleri için uyarlanabilir. Farklı konulardaki atama, sıralama, çizelgeleme veya araç rotalama gibi optimizasyon problemlerinde yöntemin etkinliği araştırılabilir.

# 6. **KAYNAKLAR**

[1] Rego, C., Gamboa, D., Glover, F., and Osterman C., "Traveling salesman problem heuristics: Leading methods, implementations and latest advances", *European Journal of Operational Research*, 211, 427–441, (2011).

[2] Davendra, D., *Travelling Salesman Problem, Theory and Applications,* Croatia: Intech, (2010).

[3] Laporte, G., "The Travelling Salesman Problem: An overview of exact and approximate algorithms", *European Journal of Operational Research,* 59, 231-247, (1992).

[4] Greco, F., *Travelling Salesman Problem,* Austria: In-teh, (2008).

[5] Çolak, S., "Genetik Algoritmalar Yardımı ile Gezgin Satıcı Probleminin Çözümü Üzerine Bir Uygulama", *Ç.Ü. Sosyal Bilimler Enstitüsü Dergisi,* 19 (3), 423-438, (2010).

[6] Bahaabadi, M., Mohaymany, A. S., and Babaei, M., "An Efficient Crossover Operator For Travelling Salesman Problem", *International Journal Of Optimization In Civil Engineering*, 2 (4), 607-619, (2012).

[7] Ahmadvand, M., Yousefikhoshbakt, M., and Darani, M. N., "Solving the Travelling Salesman Problem by an Efficient Hybrid Metaheustic Algorithm", *Journal of Advance in Computer Research,* 3 (3), 75-84, (2012).

[8] Alpdoğan, K. A., "A New Heuristic Approach to TSP", PhD Thesis, TC Marmara University, Istanbul, (2008).

[9] Liao, Y. F, Yau, D. H., and Chen C. L., "Evolutionary Algorithm to Travelling Salesman problems", *Computers and Mathematics with Applicaitons,* 64, 788-797, (2012).

[10] Jiao, L., and Wang, L., "A Novel Genetic Algorithm Based on Immunity", *IEEE Transaction on Systems, Mani, and Cybernetics- Part A:Systems and Humans,* 30 (5), 552-561, (2000).

[11] Feng, X., Lau, F.C.M., and Yu H., "A novel bio-inspired approach based on the behaviour of mosquitoes", *Information Sciences,* 233, 87-108, (2013).

[12] Shah-Hosseini, H., "The intelligent water drops Algorithm: a nature-insoired swarm- based Optimization algorithm", *Int.J.Bio-Inspired Computation,* 1 (1/2), 71-79, (2009).

[13] Applagate, D.L., Bixby, R.E., Chvatal, V., and Cook, W. C., *The Travelling Salesman Problem,* UK: Princeton Universtity Press, (2006).

[14] Johnson, D.S., and McGeoch, L.A., "The Travelling Salesman Problem: A Case Study in Local Optimization", (eds: Aarts E.H.L., and Lenstra J.K.), *Local Search in Combinational Optimization,* London: John Wiley nd Sons, (1997).

[15] Jarvis, U.M:, and Sherman, P. W., "Heterocephalus glaber", *Mamalian Species,* 706, 1-9, (2002).

[16] Kimchi, T., Reshef, M., and Terkel, J., "Evidence fort he use of reflected self-generated seismic waves for spatial orientation in a Blind subterranean mammal", *J.Exp. Biol.*, 208, 647-659, (2005).

[17] Kimchi, T., and Terkel, J., "Mole rats (*Spalax ehrenbergi*) select bypass burrowing strategies in accordance with abstacle size", *Naturwissenschaften,* 90, 36-39, (2003).

[18] "Biologists Identify the Chemical Behind Cancer REsistance in Naked Mole Rats", (Erişim Tarihi, 2013), http://www.rochester.edu/news/show.php?id=6572, (2013).

[19] Çelik, Y., and Ulker, E., "A Marriage in Honey Bee Optimization Approach to the Asymmetric Travelling Salesman Problem", *International*

*Journal of Innovative Computing Information and Control,* 8 (6), 4123-4132, (2012).

[20] Helsgaun, K., "An effective implementation of the Lin-Kernighan traveling Salesman heuristic", *European Journal of Operational Research,* 126, 106-130, (2000).

[21] Sallabi, O. M., and El-Haddad, Y., "An improved genetic Algorithm to solve the traveling Salesman problem", *World Academy of Science, Engineering and Technology,* 28, 472-474, (2009).

[22] Ölüç, M. A., "Kurbağa Sıçrama Algoritması ve Gezgin Satıcı Problemine Uygulanması", Yüksek Lisans Tezi, İstanbul Üniversitesi Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilimdalı, İstanbul, (2013).

[23] Cevre, U., "Çoklu Gezgin Satıcı Probleminin Çözümü için Bir Eniyileme Kütüphanesinin Tasarımı ve Görsel Geliştirme Ortamı ile Birlikte Gerçekleştirimi", Yüksek Lisans Tezi, Ege Üniversitesi Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, İzmir, (2008).

[24] Bayzan, Ş., "Araç Rotalarının En Kısa Yol Algoritmaları Kullanılarak Belirlenmesi ve .Net Ortamında Simülasyonu", Yüksek Lisans Tezi, Pamukkale Üniversitesi Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Denizli, (2005).

[25] Filip, E., and Otakar, M., "The Travelling Salesman Problem and its Application in Logistic Practice", *Wseas Transactions on Business and Economics,* 4 (8), 163-173, (2011).

[26] Bakhouya, M., and Gaber, J., "An Immune Inspired-Based Optimization Algorithm: Application to the Travelling Salesman Problem", *AMO,* 9 (1), 105-116, (2007).

[27] Lenstra, J.K., and Rinnoy Kan, A.H.G., "Some Simple Applications of the Travelling Salesman Problem", *Opt. Res.Q., Pergeunon Press,* 26 (4), 717-733, (1975).

[28] Grötschel, M., and Holland, O., "Solution of Large-scale symetric Travelling Salesman problems", *Mathematical Programming,* 51, 141-202, (1991).

[29] Plante, R.D., Lowe, T. J., and Chandrasekaran, R., "The Product Matrix Travelling Salesman Problem: An Applicaiton and Solution Heuristic", *Operation Research,* 35 (5), (1987).

[30] Bland, R. G., and Shallcross, D.F., "Large Travelling Salesman problems arising from experiments in X-ray crystallography: A preliminary report on computation", *Operations Research Letters,* 8 (3), 125-128, (1989).

[31] Grafinkel, R.S., "Minimizing Wallpaper Waste, Part 1: A Class of Travelling Salesman Problems", *Operations Research,* 25 (5), (1977).

[32] Ratliff, H.D. and Rosenthal, A.S. "Order-Picking in a Rectangular Warehouse: A Solvable Case for the Travelling Salesman Problem", *Operations Research,* 31, 507-521, (1983).

[33] "Theory of computation", (erişim zamanı 29.10.201, 22:00) http://en.wikipedia.org/wiki/Theory_of_computation, (2007).

[34] "P Versus NP Problem", (erişim zamanı:29.10.2013 22:40), http://en.wikipedia.org/wiki/P_%3D_NP_problem, (2013).

[35] Lim, Y.F., Hong, P.Y., Ramli, R., and Khalid, R., "An Improved Tabu Search for Solving Symmetric Travelling Salesman Problems", *2011IEEE Colloquium Humanities, Science and Engineering Research (CHUSER 2011),* Penang, 851-854, (2011).

[36] Gönülol, S., "Gezgin Satıcı Problemi için Veri Madenciliği Tabanlı Sezgisel Bir Yaklaşım", Yüksek Lisans Tezi, Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı, Kocaeli, (2009).

[37] Mondal, R.N., Hossain, S.K., and Saha S.K., "An Approach for Solving Travelling Salesman Problem", *International Journal of Applied Operational Research,* 3 (22), 15-26, (2013).

[38] “The Icosian Game”, (erişim tarihi 0.10.2013 23:50), http://puzzlemuseum.com/month/picm02/200207icosian.htm, (2002).

[39] “Icosian Game”, (erişim tarihi 30.10.2013 23:45), http://en.wikipedia.org/wiki/Icosian_game, (2008).

[40] Dantzig, G., Fulkerson, R., and Johnson, S., “Solution of a Large-Scale Travelling-Salesman Problem”, *Journal of the Operations Research Society of America*, 2 (4), 393-410, (1954).

[41] Flood, M. M., “The Travelling-Salesman Problem”, *Operations Research*, 4 (1), (1956).

[42] Miller, C. Tucker A., and Zemlin R., “Integer Programming formulations and travelling salesman problems”, *Journal of ACM*, 7, 326-329, (1960).

[43] Held, M., and Karp, R.M., “A Dynamic Programming Approach to Sequencing Problems”, *Journal of the Society for Industrial and Applied Mathematics*, 10 (1), 196-210, (1962).

[44] Little, J. D. C., Murty, K. G., Sweeney, D. W. and Karel, C., "An algorithm for the traveling salesman problem", *Operations Research*,11 (6), 972-989 (1963).

[45] Karg, R.L. and Thompson, G.L., "A heuristic approach to solving travelling salesman problems", *Management Science*, 10, 225-248, (1964).

[46] Lin, S., “Computer Solutions of the Travelling Salesman Problem”, *The Bell System Technical Journal*, 2245-2269, (1965).

[47] Held, M., and Karp, R.M., “The traveling-salesman problem and minimum spanning trees”, *Operations Research*, 18, 1138-1162, (1970).

[48] Christofides, N., “ Technical Note-Bounds for the Travelling-Salesman Problem”, *Operations Research*, 20 (5), 1044-1056, (1972).

[49] Lin, S. and Kernighan, B. W., “An Effective Heuristic Algorithm for the Traveling-Salesman Problem”, *Operations Research*, 21 (2), 498-516, (1973).

[50] Clarke, G., and Wright, J. W., “Scheduling of Vehicles from a Central Depot to a Number of Delivery Points”, *Operations Research*, 12 (4), 568-581, (1964).

[51] Rosenkrantz, D. J., Stearns, R. E., and Lewis, P. M., “An Analysis of Several Heuristics for the Traveling Salesman Problem”, *Society for Industrial and Applied Mathematics*, http://epubs.siam.org/doi/abs/10.1137/0206041, (1977).

[52] Karp, R.M., “Probabilistic Analysis of Partitioning Algorithms fort he Travelling-Salesman Problem in the Plane”, *Mathemetics of Operations Research*, 2 (3), 209-224, (1977).

[53] Grötschel M., and Padberg M.W., “On the Symmetric Travelling Salesman Problem: Theory and Coputation”, *Optimization and Operations Research Lecture Notes in Economics and Mathematical Systems*, 157, 105-115, (1978).

[54] Crowder, H., and Padberg M.W., “Solving Large-Scale Symmetric Travelling Salesman Problems to Optimality”, *Management Science*, (1980).

[55] Padberg, M. and Rinaldi, G., "Optimization of a 532-city symmetric traveling salesman problem by branch and cut", *Operations Research Letters*, 6, 1-7, (1987).

[56] Golden, B., Bodin, L., Doyle, T., and Stewart Jr, W., “Approximate Traveling Salesman Algorithms”, *Operations Research*, 28 (3), 694-711, (1980).

[57] Carpaneto, G., Fischetti, M., and Toth, P., “New lower bounds for the Symmetric Travelling Salesman Problem”, *Mathematical Programming*, 45 (1-3), 233-254, (1989).

[58] Smith, T.H.C., and Meyer, T.W.S., “Lower Bounds fort he Symmetric Travelling Salesman Problem from Lagrangean Relaxation”, *Discrete Applied Mathematics,* 26, 209-217, (1990).

[59] Grötschel, M., and Holland, O., “Solution of Large-Scale Symmetric Travelling Salesman Problems”, *Mathematical Programming,* 51, 141-202, (1991).

[60] Terzi, Ü., “Gezgin Satıcı Problemi için Diferansiyel Gelişim Algoritması Tabanlı Bir Metasezgisel Öneri”, Doktora Tezi, Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı, Kocaeli, (2009).

[61] Eastman, W.L. "Linear programming with pattern constraints", Ph.D. Thesis, Harvard University, Cambridge, (1958).

[62] Murty, K. G., “Letter to the Editor- An Algorithm for Ranking all the Assignments in Order of Increasing Cost”, *Operations Research,* 16 (3), 682-687, (1968).

[63] Bellmore, M., and Malone, J.C., "Pathology of traveling-salesman subtour-elimination algorithms", *Operations Research,* 19, 278-307, (1971).

[64] Garfinkel, R. S., “On Partitioning the Feasible Set in a Branch-And-Bound Algorithm for the Asymmetric Traveling-Salesman Problem”, *Operations Research, Mathematical Programming and Its Applications,* 21 (1), 340-343, (1973).

[65] Smith, T.H.C., Srinivasan, V., and Thompson, G.L., "Computational performance of three subtour elimination algorithms for solving asymmetric traveling salesman problems", *Annals of Discrete Mathematics,* 1, 495-506, (1977).

[66] Carpaneto, G., and Toth, P., "Some new branching and bounding criteria for the asymmetric travelling Salesman problem", *Management Science,* 26, 736-743, (1980).

[67] Balas E., and Christofides, N. “A restricted Lagrangean Approach to the Travelling Salesman Problem”, *Mathematical Programming,* 21, 19-46, (1981).

[68] Miller, D.L., and Pekny, J.F., “Exact solution of large asymmetric traveling salesman problems", *Science*, 251, 754-761, (1991).

[69] Christofides, N. "The shortest Hamiltonian chain of a graph", *SIAM Journal on Applied Mathematics,* 19, 689-696, (1970).

[70] Helbig Hansen, K., and Krarup, J. "Improvements of the Held-Karp algorithm for the symmetric traveling-salesman problem", *Mathematical Programming*, 7, 87-96, (1974).

[71] Smith, T.H.C., and Thompson, G.L., "A LIFO implicit enumeration search algorithm for the symmetric traveling salesman problem using Held and Karp's 1-tree relaxation", *Annals of Discrete Mathematics,* 1, 479-493, (1977).

[72] Volgenant, T., and Jonker, R. "A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation", *European Journal of Operational Research*, 9, 83-89, (1982).

[73] Gavish, B., and Srikanth, K.N., "An optimal solution method for large-scale multiple traveling salesman problems", *Operations Research*, 34, 698-717, (1986).

[74] Carpaneto, G., Fischetti, M., and Toth, P., "'New lower bounds for the symmetric travelling salesman problem*", Mathematical Programming,* 45,233-254, (1989).

[75] Demircioğlu, M., “Araç Rotalama Probleminin Sezgisel bir Yaklaşım ile Çözümlemesi Üzerine Bir Uygulama”, Doktora Tezi, TC Çukurova Üniversitesi Sosyal Bilimler Enstitüsü, İşletme Anabilim Dalı, Adana, (2009).

[76] Padberg, M. W., Hong, S., “On the symmetric traveling salesman problem: A computational study”, *Mathematical Programming Study*, 12, 78-107, (1980).

[77] Lawler E.L., Lenstra J.K., Rinnoy Kan A.H.G., Shmoys D.B., *The Traveling Salesman Problem*, John Wiley & Sons, 361-377, (1986).

[78] Reeves C. R., *Modern Heuristic techniques for Combinatorial Problems*, London: McGraw-Hill, (1995).

[79] Marinakis, Y., Marinaki, M., and Dounias, G., "Honey bees mating Optimization fort he Euclidean Travelling Salesman problem", *Information Sciences,* 181, 4684-4698, (2011).

[80] Jünger, M., Reinelt, G., and Rinaldi, G., "The Travelling Salesman Problem", (eds: Ball, M.O. et all.), *Handbooks in OR&MS,* 7, Elsevier Science, (1995).

[81] Gülsün, B., Tuzkaya, G., and Bildik, E., "Reverse Logistics Network Design: A Simulated Annealing Approach", *Journal of Engineering and Natural Sciences,* 26 (1), 68-80, (2008).

[82] Kirpatrick, S., Gelatt, C.D., and Vecchi, M.P., "Optimization by Simulated Annealing", *Science,* 220 (4598), 671-680, (1983).

[83] Siphaioğlu, A., "Gezgin Satıcı ve Araç Turu Belirleme Problemleri için Yeni Alt Tur Engelleme Kısıtları", Doktora Tezi, Osmangazi Üniversitesi Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı, Eskişehir, (1996).

[84] Cerny, V., "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm", *Journal of Optimization Theory and Appliations,* 45 (1), 41-51, (1985).

[85] Dueck, G., and Scheuber, T., "Threshold accepting: A general Purpose Optimization Algorithm appearing superior to simulated annealing", *Journal of Computational Physics*, 90 (1), 161-175, (1990).

[86]Karaboğa, D.,*Yapay Zeka Optimizasyon Algoritmaları,*Ankara: NobelYayın Dağıtım, (2011).

[87] Glover, F., "Tabu Search-Part 1", *ORSA Journal on Computing,* 1 (3), 196-204, (1989).

[88] Knox, J., and F. Glover, "Comparative Testing of Traveling Salesman Heuristics Derived from Tabu Search, Genetic Algorithms and Simulated Annealing", *Center for Applied Artificial Intelligence,* Univ. of Colorado, (1989).

[89] Fiechter, C-N., "A paralel tabu search Algorithm for large traveling salesman problems", *Discrete Applied Mathematics,* 51, 2443-267, (1994).

[90] Holland, J.H., *Adaptation in Natural and Artifical Systems,* Ann Arbor: University of Michigan Press, (1975).

[91] Brady, R. M., "Optimization strategies gleaned from biological evolution", *Nature*, 317, 804 – 806, (1985).

[92] Taherdangkoo, M., Shirzadi, M.,H., and Bagheri, M., H., "A novel meta-heuristic algorithm for numerical function optimization: Blind, naked mole-rats (BNMR) algorithm", *Scientific Research and Essays*, 741, 3566-3583, (2012).

[93] Koç, I.O., "Gezgin Satıcı Problemi için Çok Popülasyonlu Paralel Bir Genetik Algoritma Tasarımı, Geliştirilmesi ve Analizi", Doktora Tezi, Eskişehir Osmangazi Üniversitesi Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı, Eskişehir, (2007).

[94] Larranaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I., Dizdarevic, S., "Genetic algorithms for the traveling salesman problem: A review of representations and operators", *Articial Intelligence Review*, 13: 129 -70, (1999).

[95] Hopfield, J.J., and Tank, D.,W., "Neural Computation of Decisions in Optimization Problems", Biological Cybernetics, 52, 141-152, (1985).

[96] Bout, D. E. & Miller, T. K., *A Traveling Salesman Objective Function That Works.*, ICNN-88, 299-303, (1988).

[97] Beale, R. & Jackson, T., *Neural Computing: An Introduction*, Bristol, U.K: IOP Publishing Ltd., (1990).

[98] Cichock, A. & Unbehaun, R., *Neural Networks for Optimization and Signal Processing*, New York: John-Wiley & Sons, (1993).

[99] Fritzke, B., and Wilke, P., "FLEXMAP - A neural network for the traveling salesman problem with linear time and spaee complexity", *Research Report,Universität*Erlangen-Nürnberg, (1991).

[100] Bhide, S., John, N., and Kabuka, M.R., "A Boolean Neural Network Approach fort he Travelling Salesman Problem", *IEEE Transactions on Computers,* 42 (10), 1271-1278, (1993).

[101] Dorigo, M., Maniezzo, V., and Coloni, A., "Positive Feedback as a Search Strategy", *Technical Report 91-016,* Italy, (1991).

[102] Söyler, H. Ve Keskintürk, T., "Karınca Kolonisi Algoritması ile Gezen Satıcı Probleminin Çözümü", *8. Türkiye Ekonometri ve İstatistik Kongresi 24-25 Mayıs 2007-İnönü Üniversitesi,* Malatya, (2007).

[103] Dorigo, M.; & Gambardella, L.M., "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", *IEEE Transactions on Evolutionary Computation*, 1 (1), (1997), ISSN 1089–778X/97.

[104] Dorigo, M., and Stützle, T., *Ant Colony Optimization,* UK: The MIT Press, (2004).

[105] Özsağlam, M.Y, "Parçacık Sürü Optimizasyonu Algoritmasının Gezgin Satıcı Problemine Uygulanması ve Performasının İncelenmesi", Yüksek Lisans Tezi, TC Selçuk Üniversitesi Fen Bilimleri Enstitüsü, Elektronik ve Bilgisayar Sistemleri Eğitimi Anabilim Dalı, Konya, (2009).

[106] Wang K.P. , Huang L., Zhou C. G., Pang W., "Particle Swarm Optimization for Travelling Salesman Problem", *Machine Learning and Cybernetics*, 4, 1583-1585, (2003).

[107] Pang, W., Wang, K., Zhou, C., Dong, L., "Fuzzy Discrete Particle Swarm Optimization for Solving Traveling Salesman Problem", *Proceedings of*

*the Fourth International Conference on Computer and Information Technology* (CIT'04), (2004).

[108] Goldbarg, E.F.G; Souza, G.R. & Goldbarg, M.C., "Particle swarm for the traveling salesman problem", *Proceedings of the EvoCOP 2006,(*Eds: Gottlieb, J. & Raidl, G.R.), Lecture Notes in Computer Science, 3906, 99-110, Budapest, Hungary, Springer, Berlin, ISBN: 3540331786, (2006).

[109] Shi, X.H., Liang, Y.C., Lee, H.P., Lu, C., and Wang, Q.X., "Particle swarm Optimization-based algorithms for TSP and Generalized TSP", *Information Processing Letters,* 103, 169-176, (2007).

[110] Sato, T., and Hagiwara, M., "Bee system: Finding Solution by a Concentrated Search", *IEEE,* 3954-3959, (1997).

[111] Akça, R.M., "Yapay Arı Kolonisi Algoritması Kullanılarak Gezgin Satıc Probleminin Türkiye'deki İl ve İlçe Merkezlerine Uygulanması", Yüksek Lisans Tezi, TC Selçuk Üniversitesi Fen Bilimleri Enstitüsü, Elektronik ve Bilgisayar Sistemleri Eğitimi Anabilim dalı, Konya, (2011).

[112] Karaboğa, D., "An Idea Based on Honey Bee Swarm For Numerical Optimization", *Technical Report-TR006,* Kayseri, (2005).

[113] "Animal Echolocation", (erişim tarihi 23.11.2013 22:20 ), http://en.wikipedia.org/wiki/Animal_echolocation, (2012).

[114] Letchford, A.N., Nasiri, S.D., and Theis, D.O., "Compact Formulations of the Steiner Travelling Salesman Problem and Related Problems", *CET,* 1-23, (2012).

[115] Gerard, M., and Laporte, G., "Improvements and Extensions to the Miller-Tucker-Zemlin Subtour Elimination Constraints", *Journal Operations Research Letters,* 10 (1), 27-36, (1991).

[116] Kennedy, J. & Eberhart, R.C. "Particle swarm Optimization", *Proceedings of the IEEE International Conference on Neural Networks*, 4, 1942-1948, ISBN: 0780327683, Perth, Western Australia, (1995).

[117] Osman, I.H., and Laporte,G. “Metaheuristics: A bibliography”,*Ann. Oper. Res.*, 63, 513–623,(1996).

[118] Tseng, L-Y.,“Metaheuristic Methods and Their Applications”, (erişim tarihi 18.12.2013) http://www.inf.cyut.edu.tw/952/960428.pdf , (2013).

[119]Polat, O., “Montaj hattı işçi atama ve dengeleme problemlerinin genetik algoritmalarla çözülmesi”, Yüksek Lisans Tezi, Pamukkale Üniversitesi Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim dalı, Denizli, (2008).

[120] Bastı, M., “P-medyan Tesis Yeri Seçim Problemi ve Çözüm Yaklaşımları”, *AJIT-e Online Academic Journal of Information Technology*, 3(7), (2012).

[121] Eryavuz, M., ve Gencer, C., “Araç Rotalama Problemine Ait Bir Uygulama”, Süleyman Demirel Üniveristesi, İktisadi ve İdari Bilimler Fakültesi, 6(1), 139-155, (2001).

[122] “Königsberg’in yedi köprüsü”, (erişim tarihi 04.01.2014), http://tr.wikipedia.org/wiki/K%C3%B6nigsberg%27in_yedi_k%C3%B6pr%C3%BCs%C3%BC , (2014).

[123] Bauk, S., and Kovac, N., " The Comparative Analysis of two Neural Networks Models in The Function Of The Linear Ship’s Route Costs Minimization ", *JEL Classifiaction*, 92, 89 , (2006).

[124] Patır, S., “Tamsayılı Programlama ve Malatya Maksan Transformatör İşletmesinde Bir Uygulama”, (erişim tarih 04.01.2014), http://e-dergi.atauni.edu.tr/index.php/IIBD/article/viewFile/3815/3643 , (2014).

[125] Croes, G., A., “A method for solving traveling salesman problems” *Operations Res*,. 6, 791-812, (1958).

[126] Bock, F., “An algorithm for solving 'travelling salesman' and related netmork optimization problems”, 14. National meeting of the Operations Research Society of America, 897, (1958).

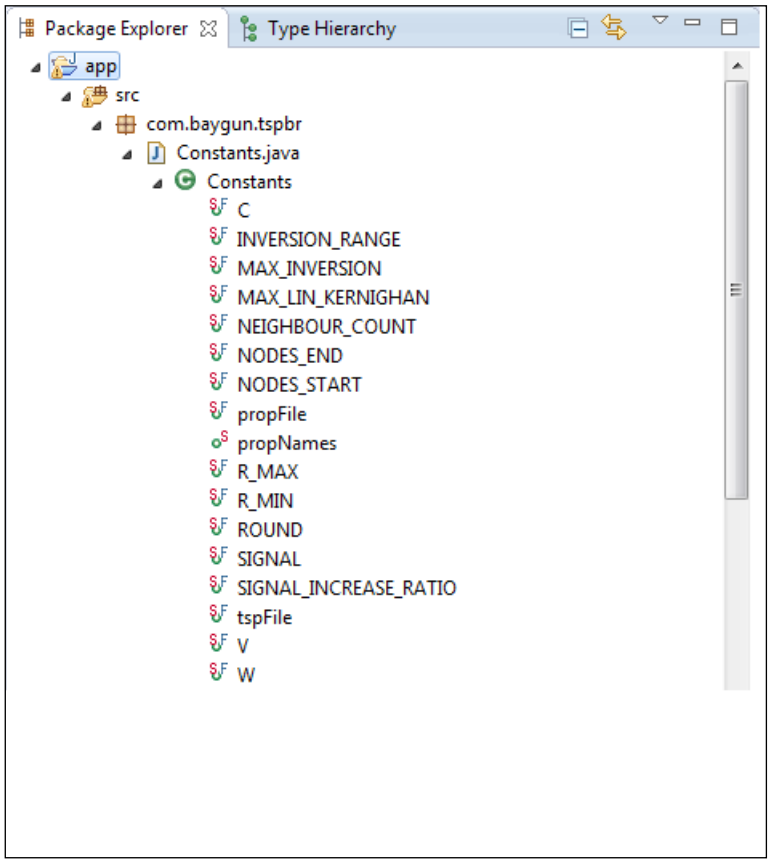
[127] “Turing Makinesi”, (erişim tarihi 18.02.2014), http://tr.wikipedia.org/wiki/Turing_makinesi, (2014).

[128] Evans, J.,R., and Minieka, E., “*Optimization Algorithms for Networks and Graphs*”, Marcel Dekker Inc., United States of America, (1992).

# EKLER

# 7. EKLER

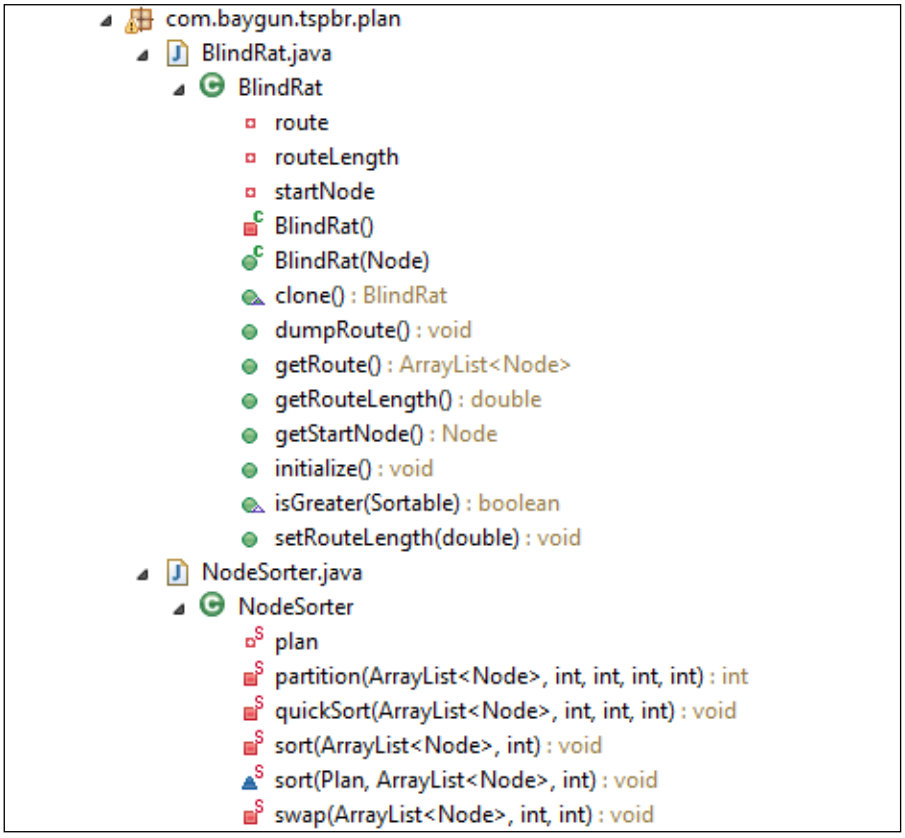
## EK A: Yazılımda kullanılan değişkenler ve fonksiyonlar



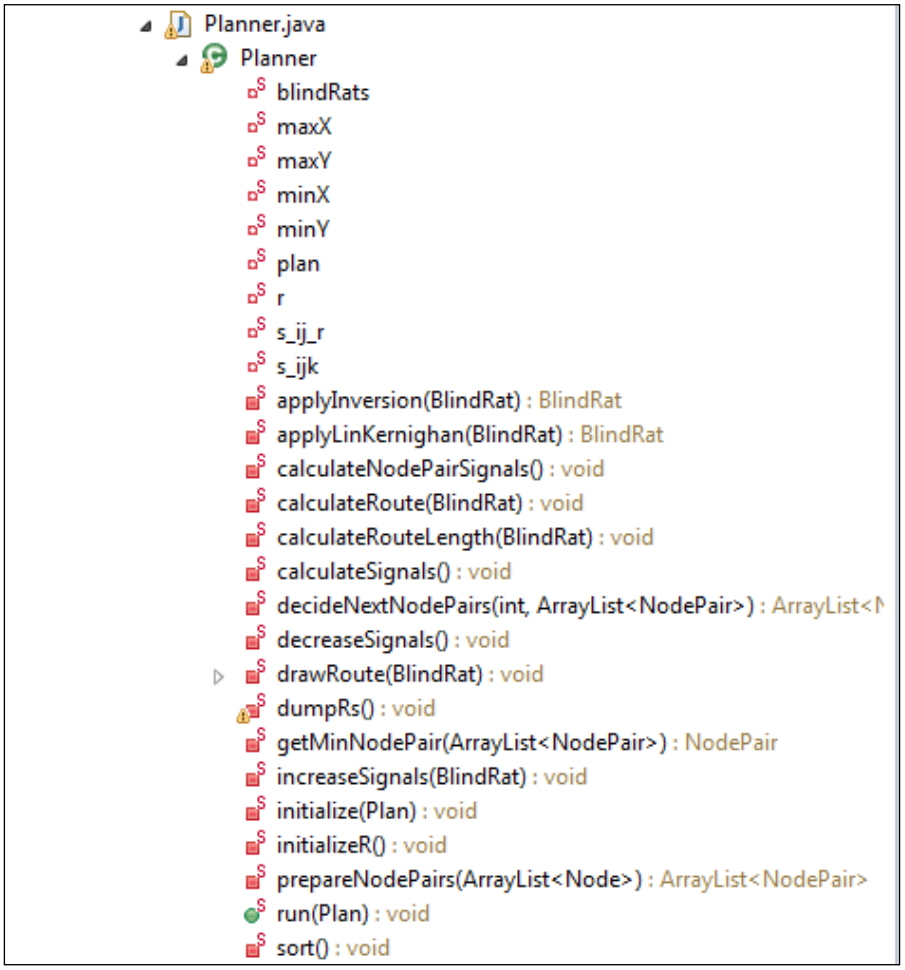
**EK A:Devam**


com.baygun.tspbr.model
Node.java
Node
number
x
y
Node()
Node(int, double, double)
Node(String)
equals(Object) : boolean
getNumber() : int
getX() : double
getY() : double
setNumber(int) : void
setX(double) : void
setY(double) : void
toString() : String
NodeManager.java
NodeManager
distance(Node, Node) : int
getNodes(FileReader) : ArrayList<Node>
NodePair.java
NodePair
firstNode
secondNode
NodePair(Node, Node)
getFirstNode() : Node
getSecondNode() : Node
setFirstNode(Node) : void
setSecondNode(Node) : void

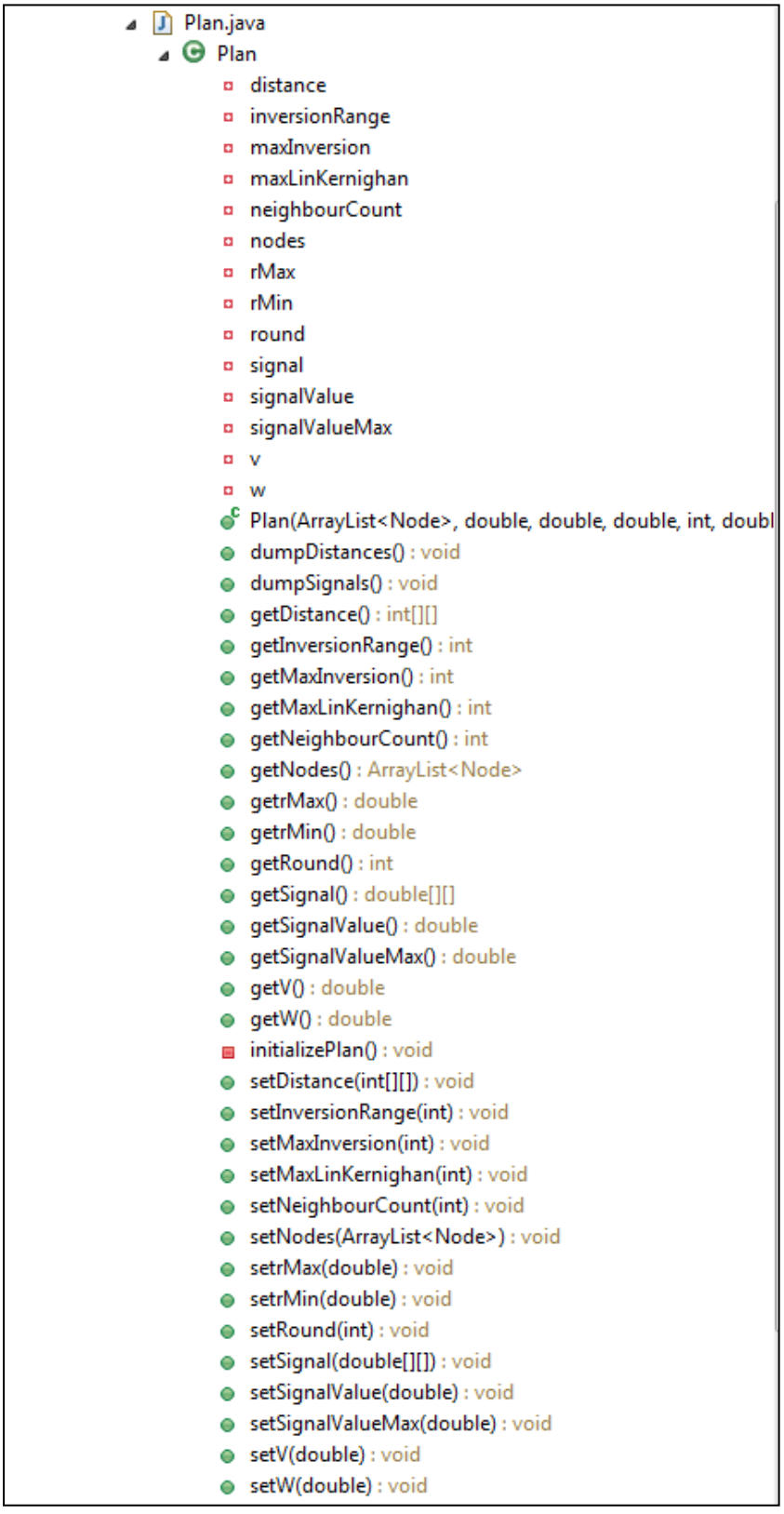
**EK A: Devam**


com.baygun.tspbr.plan
BlindRat.java
BlindRat
route
routeLength
startNode
BlindRat()
BlindRat(Node)
clone() : BlindRat
dumpRoute() : void
getRoute() : ArrayList<Node>
getRouteLength() : double
getStartNode() : Node
initialize() : void
isGreater(Sortable) : boolean
setRouteLength(double) : void
NodeSorter.java
NodeSorter
plan
partition(ArrayList<Node>, int, int, int, int) : int
quickSort(ArrayList<Node>, int, int, int) : void
sort(ArrayList<Node>, int) : void
sort(Plan, ArrayList<Node>, int) : void
swap(ArrayList<Node>, int, int) : void

**EK A: Devam**


Planner.java
Planner
blindRats
maxX
maxY
minX
minY
plan
r
s_ij_r
s_ijk
applyInversion(BlindRat) : BlindRat
applyLinKernighan(BlindRat) : BlindRat
calculateNodePairSignals() : void
calculateRoute(BlindRat) : void
calculateRouteLength(BlindRat) : void
calculateSignals() : void
decideNextNodePairs(int, ArrayList<NodePair>) : ArrayList<N
decreaseSignals() : void
drawRoute(BlindRat) : void
dumpRs() : void
getMinNodePair(ArrayList<NodePair>) : NodePair
increaseSignals(BlindRat) : void
initialize(Plan) : void
initializeR() : void
prepareNodePairs(ArrayList<Node>) : ArrayList<NodePair>
run(Plan) : void
sort() : void

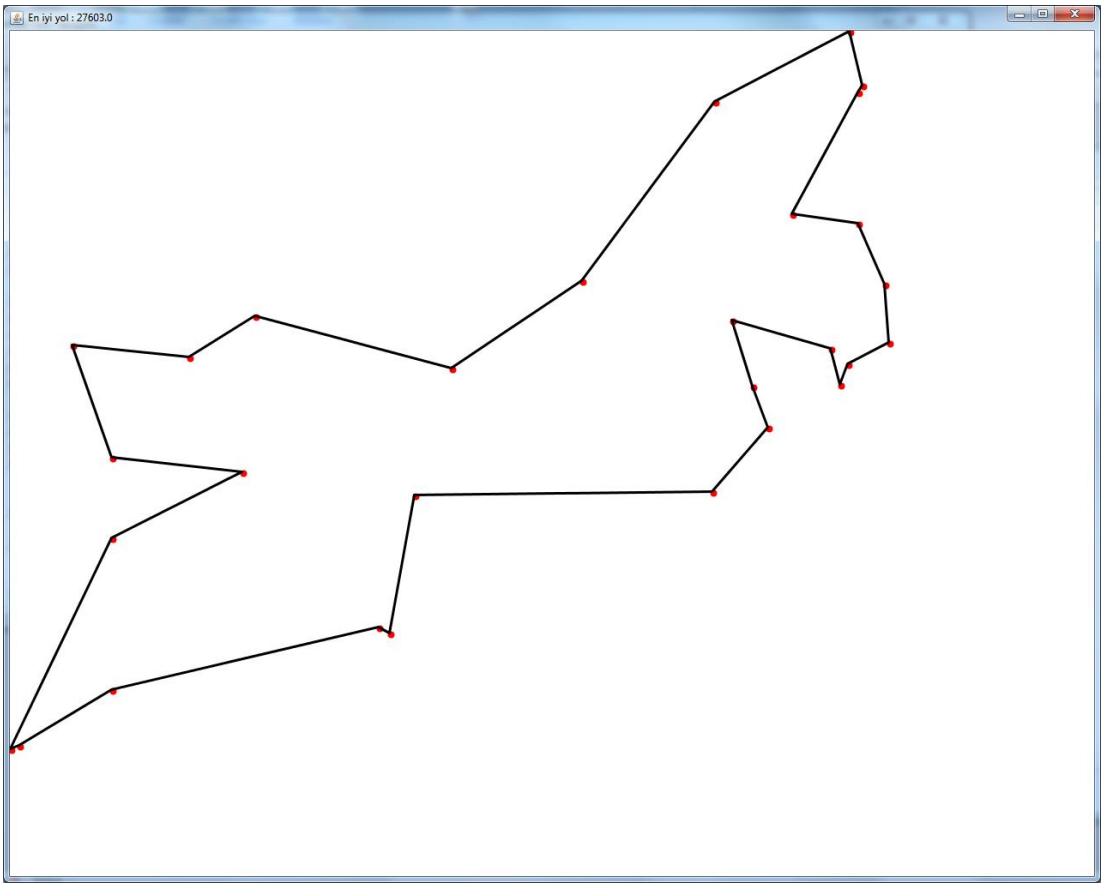
**EK A: Devam**

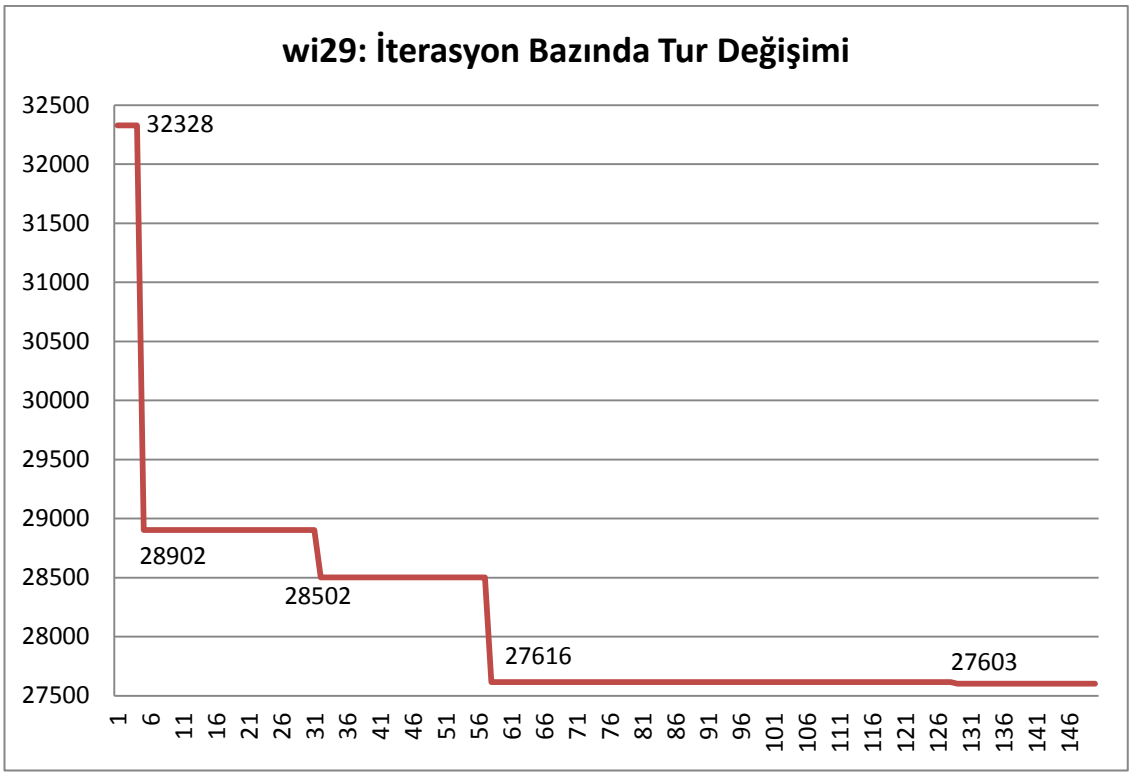

Plan.java
Plan
distance
inversionRange
maxInversion
maxLinKernighan
neighbourCount
nodes
rMax
rMin
round
signal
signalValue
signalValueMax
v
w
Plan(ArrayList<Node>, double, double, double, int, doubl
dumpDistances() : void
dumpSignals() : void
getDistance() : int[][]
getInversionRange() : int
getMaxInversion() : int
getMaxLinKernighan() : int
getNeighbourCount() : int
getNodes() : ArrayList<Node>
getrMax() : double
getrMin() : double
getRound() : int
getSignal() : double[][]
getSignalValue() : double
getSignalValueMax() : double
getV() : double
getW() : double
initializePlan() : void
setDistance(int[][]) : void
setInversionRange(int) : void
setMaxInversion(int) : void
setMaxLinKernighan(int) : void
setNeighbourCount(int) : void
setNodes(ArrayList<Node>) : void
setrMax(double) : void
setrMin(double) : void
setRound(int) : void
setSignal(double[][]) : void
setSignalValue(double) : void
setSignalValueMax(double) : void
setV(double) : void
setW(double) : void

## EK B: Taguchi L32 Tasarımı: Parametreler, ortalama tur uzunluğu ve S/N oranları

| Sinyal | w | v | İterasyon | Rmin | Rmax | SAO | q | Mut.Sayısı | Mut.Aralık | Ort.Tur Uzunluğu | S/N Oranları |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 100 | 20 | 2 | 100 | 0,3 | 1,05 | 0,33 | 4 | 50 | 3 | 8185,15625 | -78,25762055 |
| 100 | 20 | 2 | 100 | 0,95 | 1,05 | 1 | 8 | 200 | 6 | 8520,46875 | -78,61532443 |
| 100 | 20 | 2 | 300 | 0,3 | 1,75 | 0,33 | 8 | 200 | 6 | 8225,96875 | -78,29807715 |
| 100 | 20 | 2 | 300 | 0,95 | 1,75 | 1 | 4 | 50 | 3 | 8157,65625 | -78,22862793 |
| 100 | 20 | 12 | 100 | 0,3 | 1,75 | 1 | 4 | 200 | 6 | 8263,21875 | -78,34514177 |
| 100 | 20 | 12 | 100 | 0,95 | 1,75 | 0,33 | 8 | 50 | 3 | 8017,65625 | -78,07749349 |
| 100 | 20 | 12 | 300 | 0,3 | 1,05 | 1 | 8 | 50 | 3 | 8203,40625 | -78,27794599 |
| 100 | 20 | 12 | 300 | 0,95 | 1,05 | 0,33 | 4 | 200 | 6 | 8452,21875 | -78,54557472 |
| 100 | 300 | 2 | 100 | 0,3 | 1,75 | 1 | 8 | 50 | 6 | 8056,21875 | -78,11426937 |
| 100 | 1300 | 2 | 100 | 0,95 | 1,75 | 0,33 | 4 | 200 | 3 | 7961,90625 | -78,01615425 |
| 100 | 1300 | 2 | 300 | 0,3 | 1,05 | 1 | 4 | 200 | 3 | 8147,65625 | -78,21660675 |
| 100 | 1300 | 2 | 300 | 0,95 | 1,05 | 0,33 | 8 | 50 | 6 | 8245,21875 | -78,31470232 |
| 100 | 1300 | 12 | 100 | 0,3 | 1,05 | 0,33 | 8 | 200 | 3 | 8007,65625 | -78,06547231 |
| 100 | 1300 | 12 | 100 | 0,95 | 1,05 | 1 | 4 | 50 | 6 | 8282,46875 | -78,36176694 |
| 100 | 1300 | 12 | 300 | 0,3 | 1,75 | 0,33 | 4 | 50 | 6 | 7987,96875 | -78,04451966 |
| 100 | 1300 | 12 | 300 | 0,95 | 1,75 | 1 | 8 | 200 | 3 | 7980,15625 | -78,03647969 |
| 500 | 20 | 2 | 100 | 0,3 | 1,75 | 1 | 8 | 200 | 3 | 7945,21875 | -78,00181438 |
| 500 | 20 | 2 | 100 | 0,95 | 1,75 | 0,33 | 4 | 50 | 6 | 8088,15625 | -78,15397191 |
| 500 | 20 | 2 | 300 | 0,3 | 1,05 | 1 | 4 | 50 | 6 | 8273,90625 | -78,3544244 |
| 500 | 20 | 2 | 300 | 0,95 | 1,05 | 0,33 | 8 | 200 | 3 | 8134,21875 | -78,20224733 |
| 500 | 20 | 12 | 100 | 0,3 | 1,05 | 0,33 | 8 | 50 | 6 | 8133,90625 | -78,20328997 |
| 500 | 20 | 12 | 100 | 0,95 | 1,05 | 1 | 4 | 200 | 3 | 8171,46875 | -78,24931195 |
| 500 | 20 | 12 | 300 | 0,3 | 1,75 | 0,33 | 4 | 200 | 3 | 7876,96875 | -77,93206467 |
| 500 | 20 | 12 | 300 | 0,95 | 1,75 | 1 | 8 | 50 | 6 | 8106,40625 | -78,17429735 |
| 500 | 1300 | 2 | 100 | 0,3 | 1,05 | 0,33 | 4 | 200 | 6 | 8078,15625 | -78,14195073 |
| 500 | 1300 | 2 | 100 | 0,95 | 1,05 | 1 | 8 | 50 | 3 | 7964,46875 | -78,01843955 |
| 500 | 1300 | 2 | 300 | 0,3 | 1,75 | 0,33 | 8 | 50 | 3 | 7669,96875 | -77,70119227 |
| 500 | 1300 | 2 | 300 | 0,95 | 1,75 | 1 | 4 | 200 | 6 | 8050,65625 | -78,11295811 |
| 500 | 1300 | 12 | 100 | 0,3 | 1,75 | 1 | 4 | 50 | 3 | 7707,21875 | -77,74825689 |
| 500 | 1300 | 12 | 100 | 0,95 | 1,75 | 0,33 | 8 | 200 | 6 | 7910,65625 | -77,96182367 |
| 500 | 1300 | 12 | 300 | 0,3 | 1,05 | 1 | 8 | 200 | 6 | 8096,40625 | -78,16227617 |
| 500 | 1300 | 12 | 300 | 0,95 | 1,05 | 0,33 | 4 | 50 | 3 | 7896,21875 | -77,94868984 |

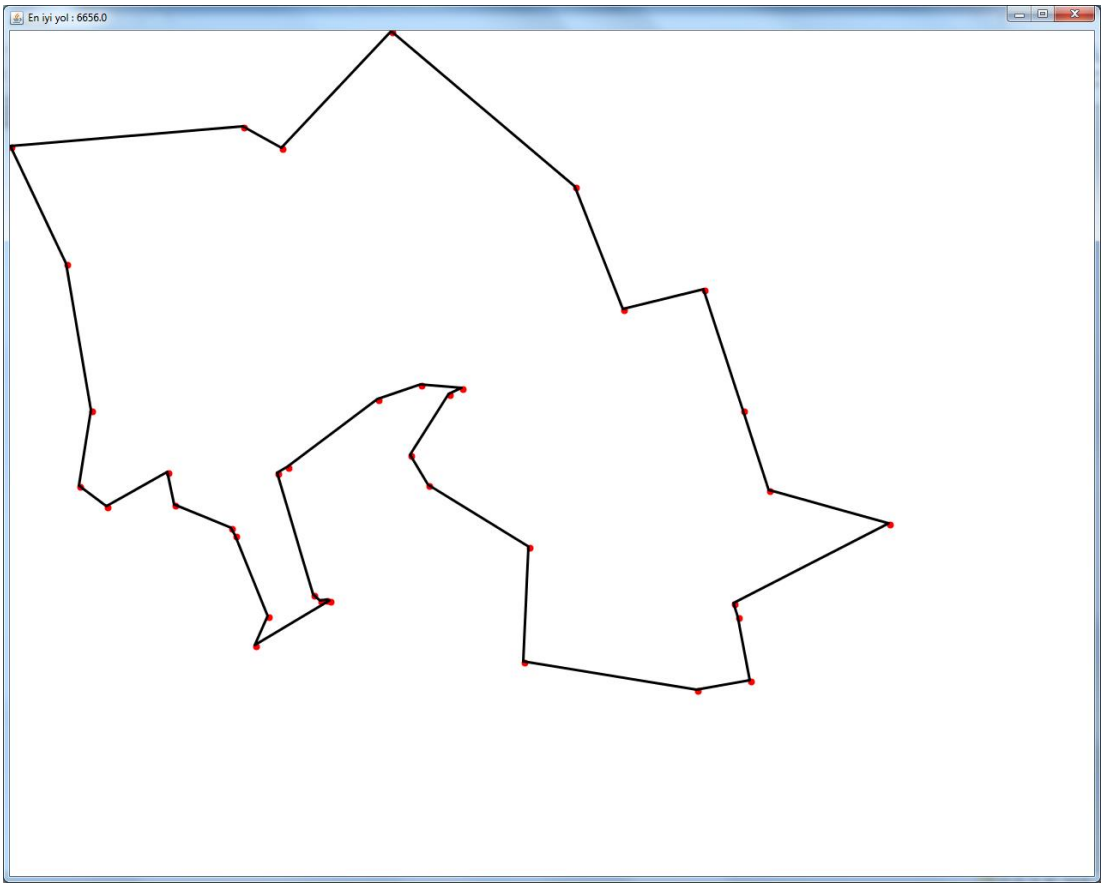
## EK C-1: wi29 Test Problemi İçin Bulunan Rota ve Çözüm Grafiği

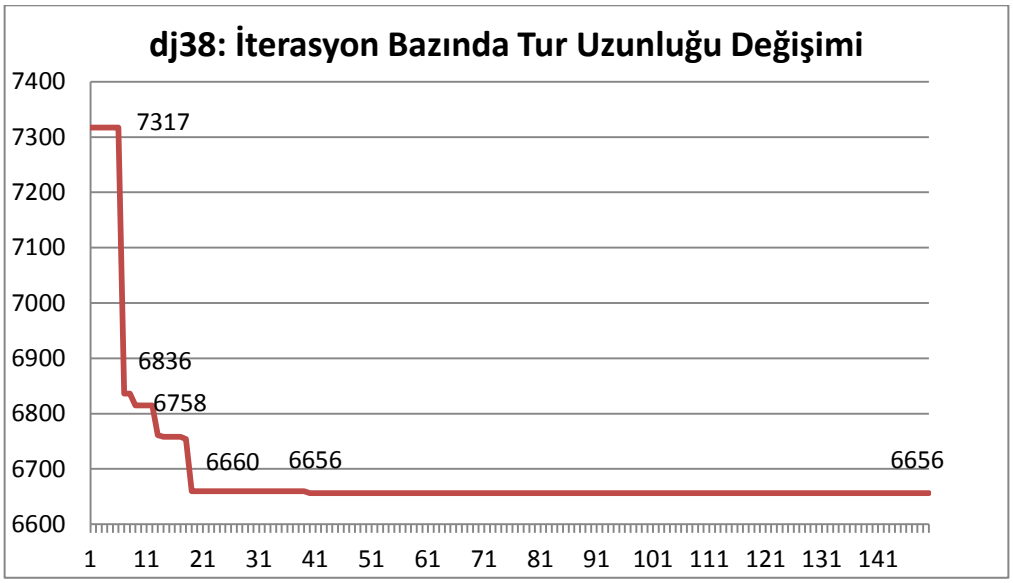



**Toplam Tur Uzunluğu = 27603**

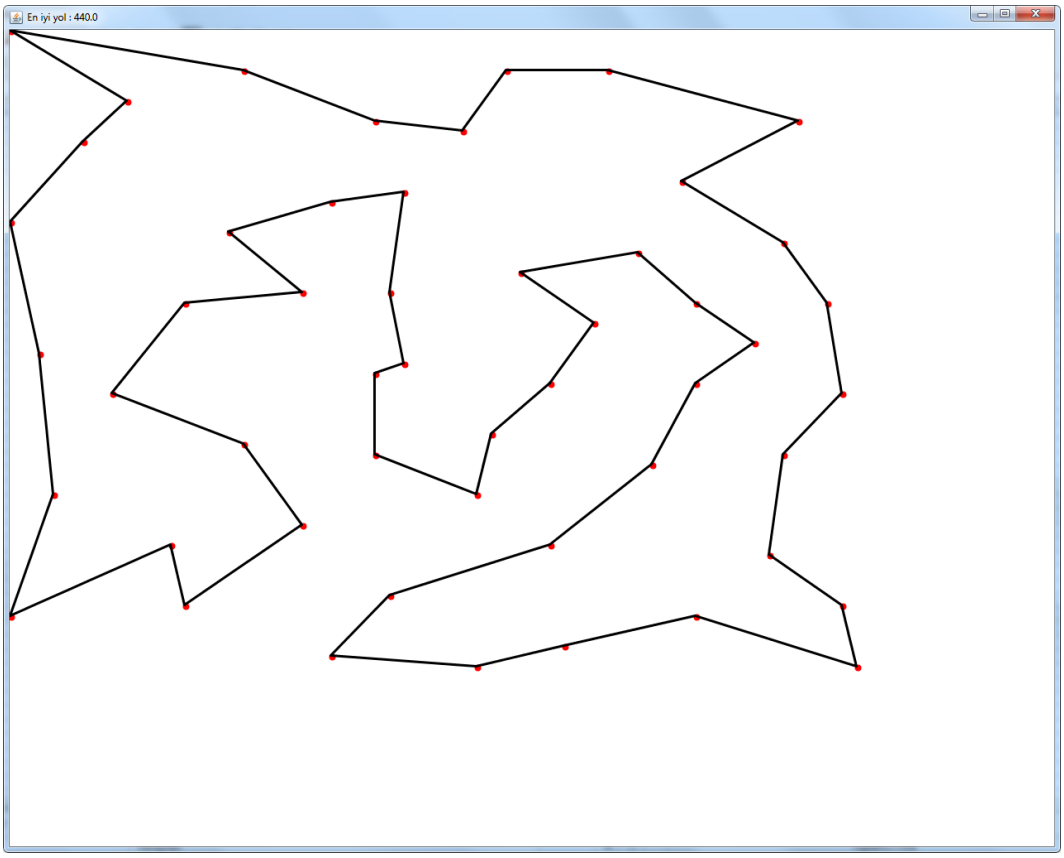
## EK C-2: dj38 Test Problemi İçin Bulunan Rota ve Çözüm Grafiği

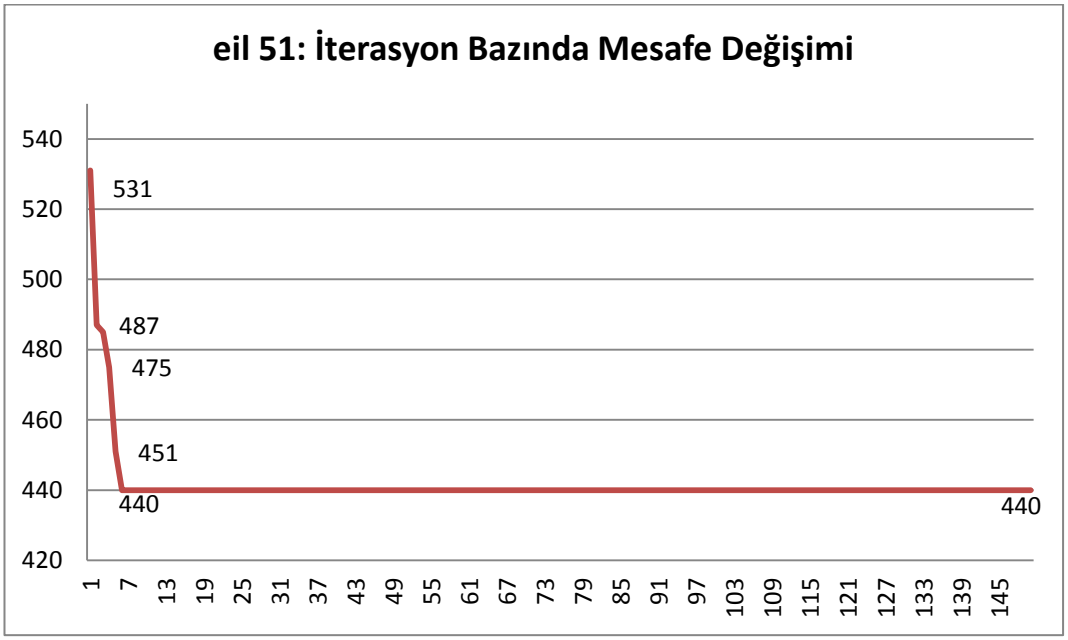



**Toplam Tur Uzunluğu = 6656**

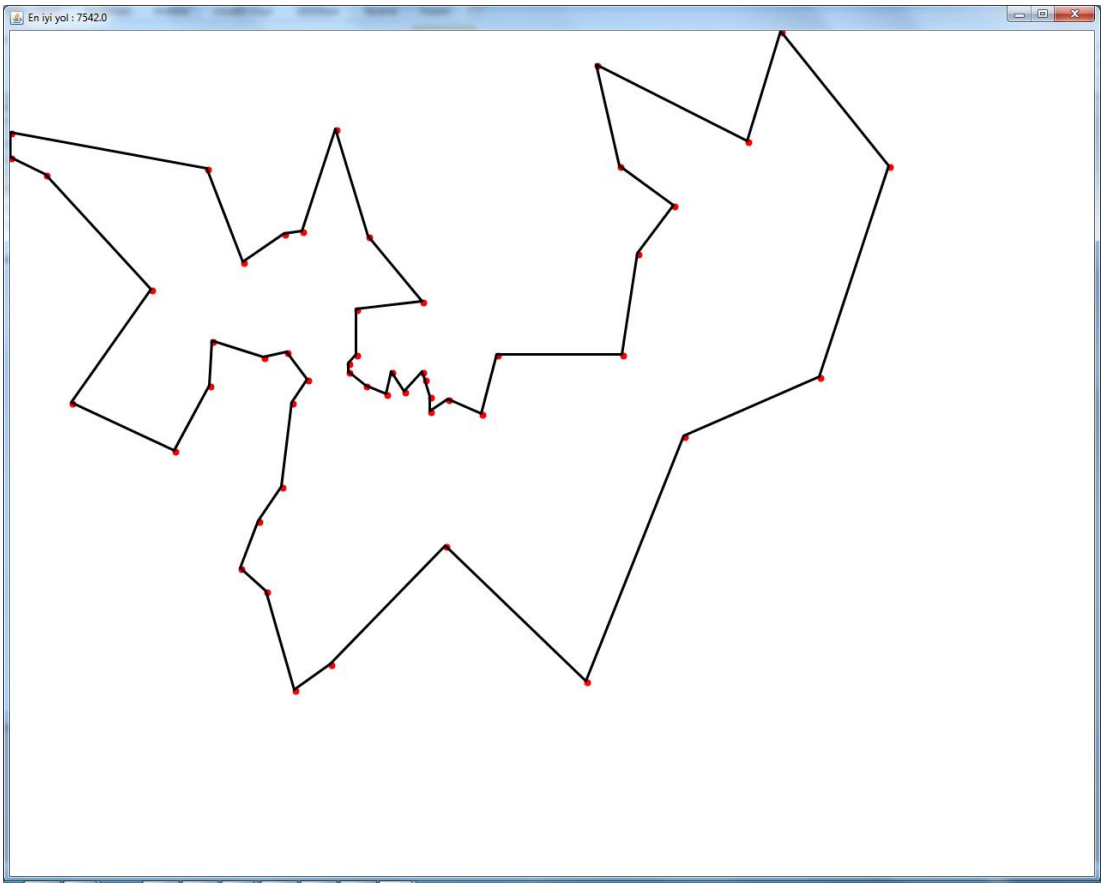
## EK C-3: eil51 Test Problemi İçin Bulunan Rota ve Çözüm Grafiği

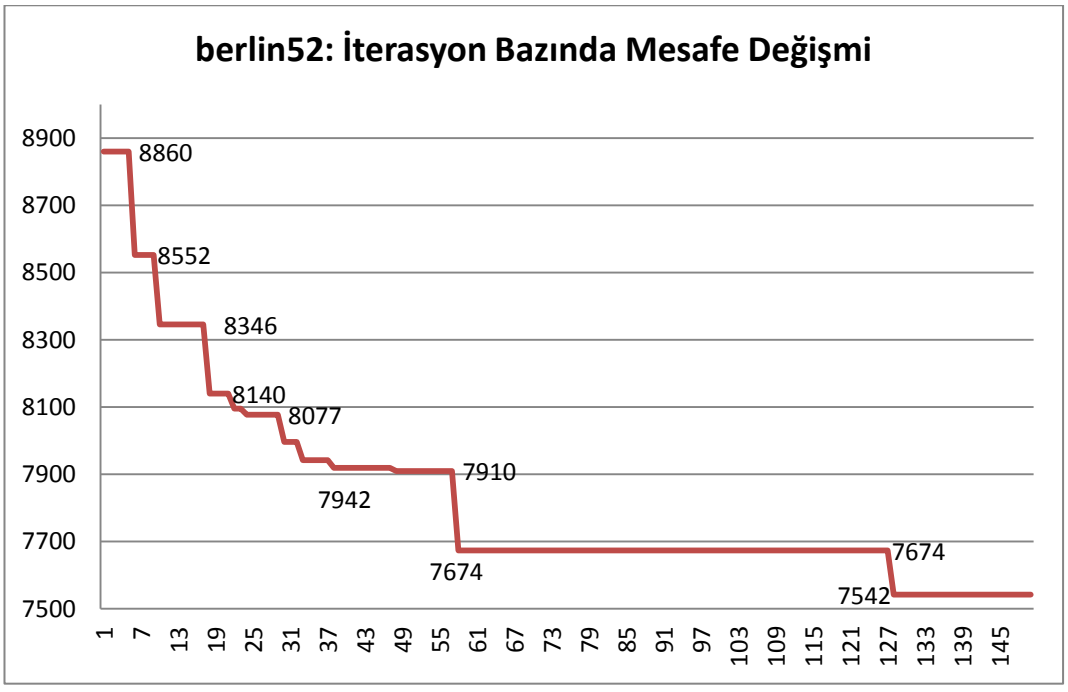



**Toplam Tur Uzunluğu = 440**

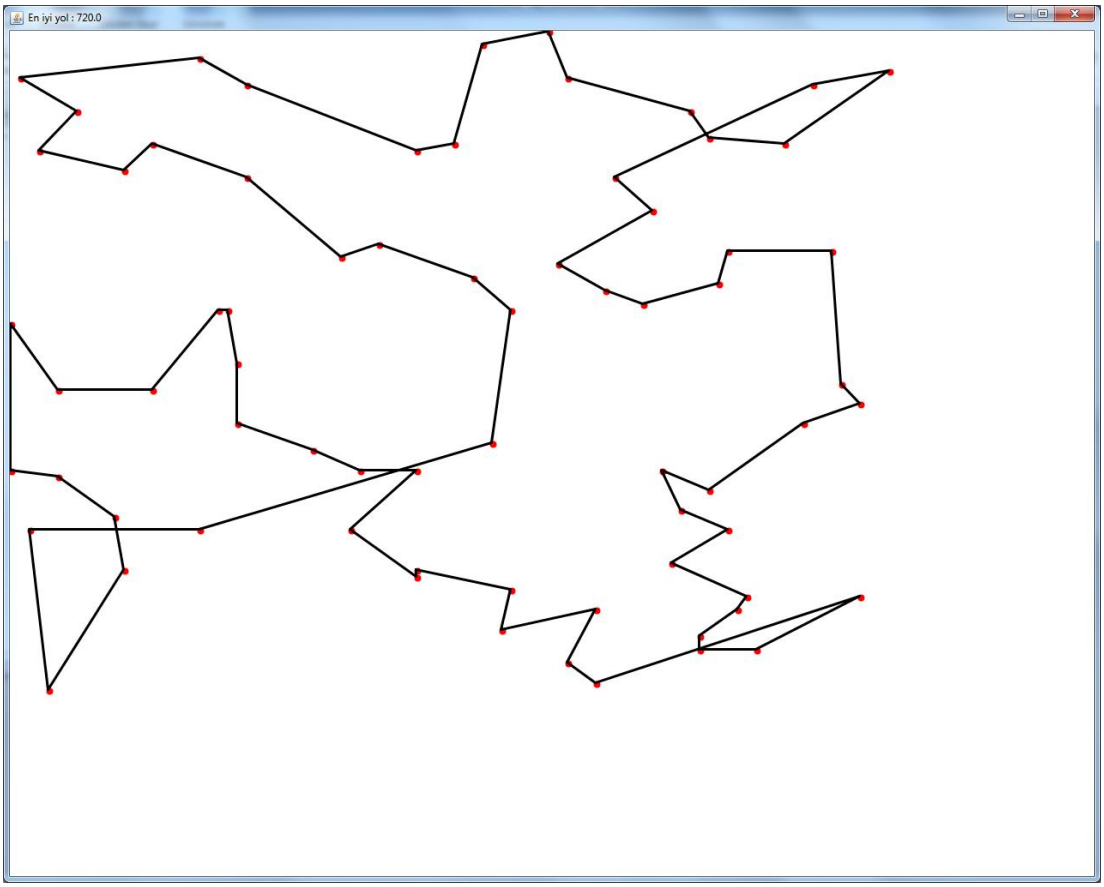
## EK C-4: berlin52 Test Problemi İçin Bulunan Rota ve Çözüm Grafiği

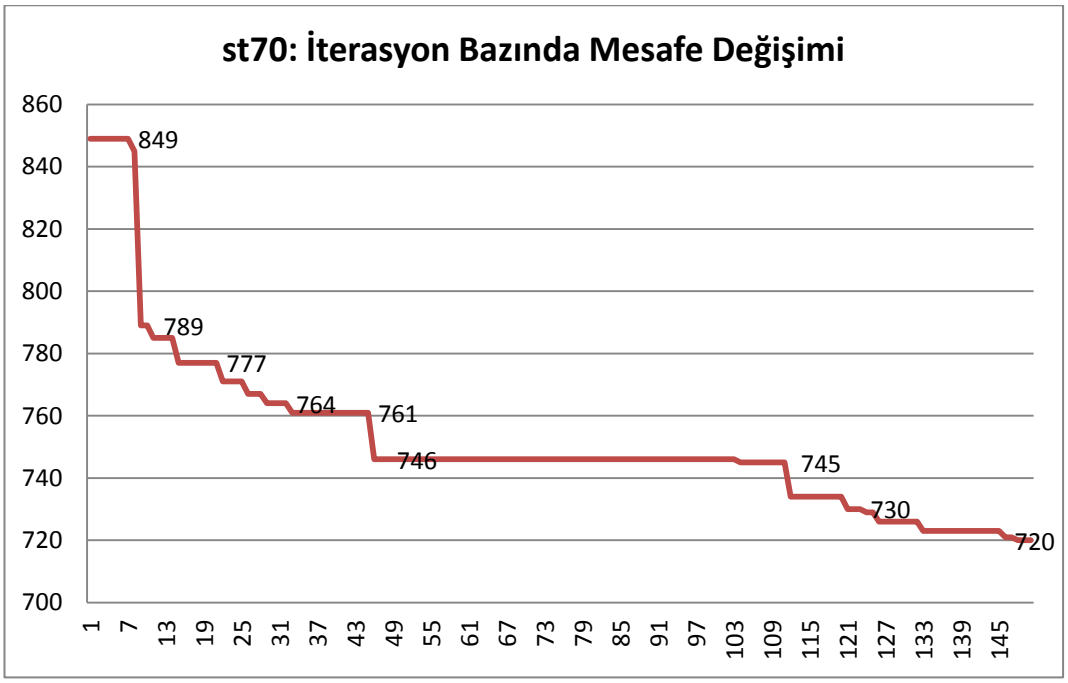



**Toplam Tur Uzunluğu = 7542**

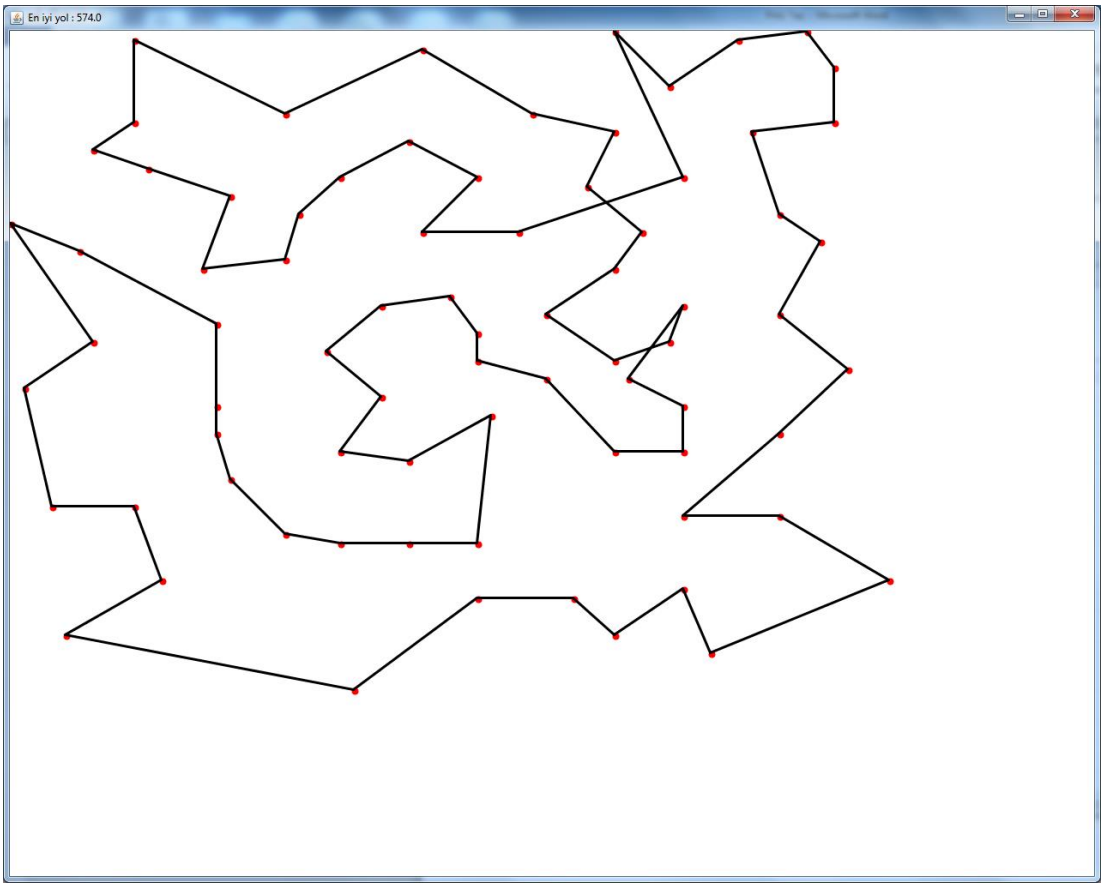
**EK C-5: st70 Test Problemi İçin Bulunan Rota ve Çözüm Grafiği**

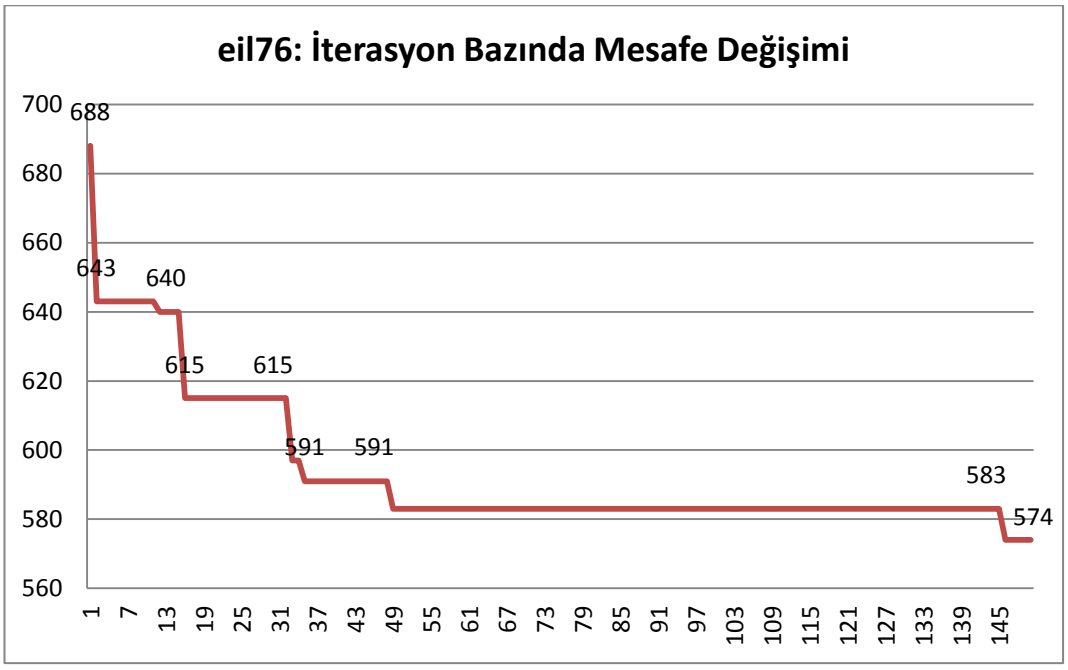



**Toplam Tur Uzunluğu = 720**

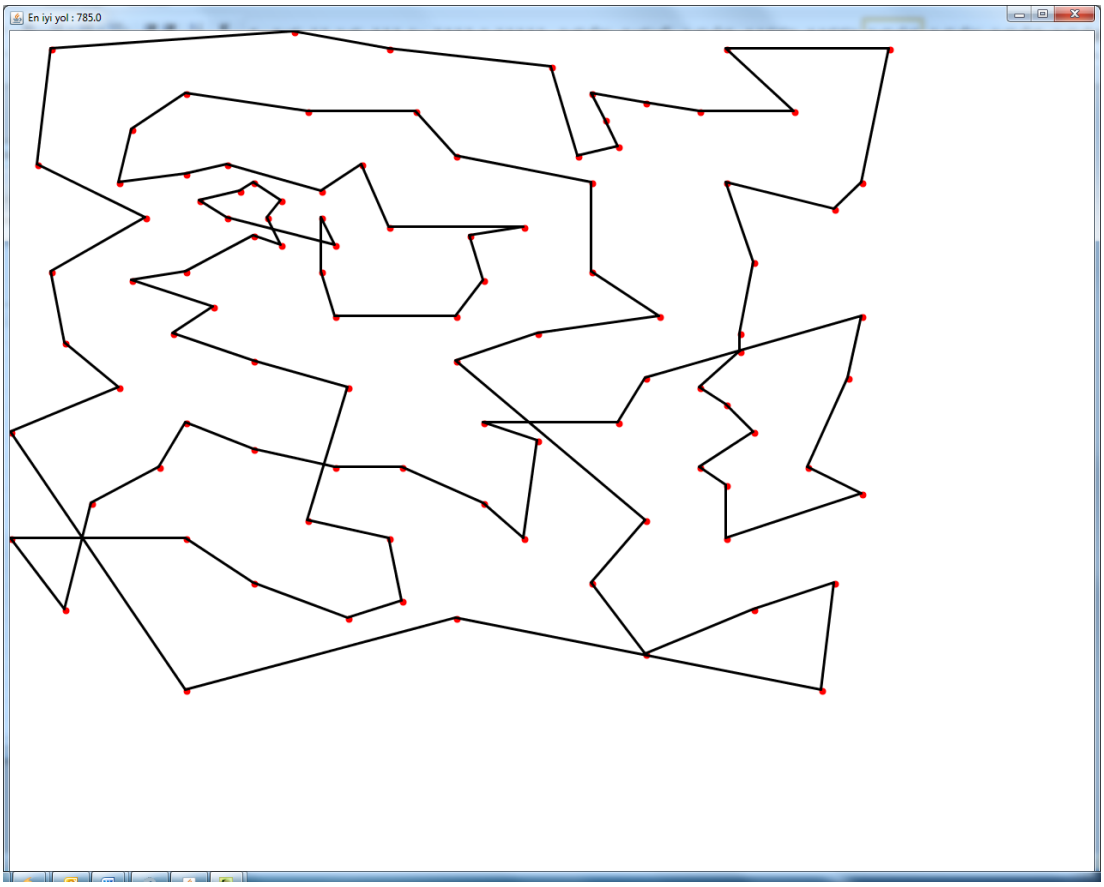
**EK C-6: eil76 Test Problemi İçin Bulunan Rota ve Çözüm Grafiği**

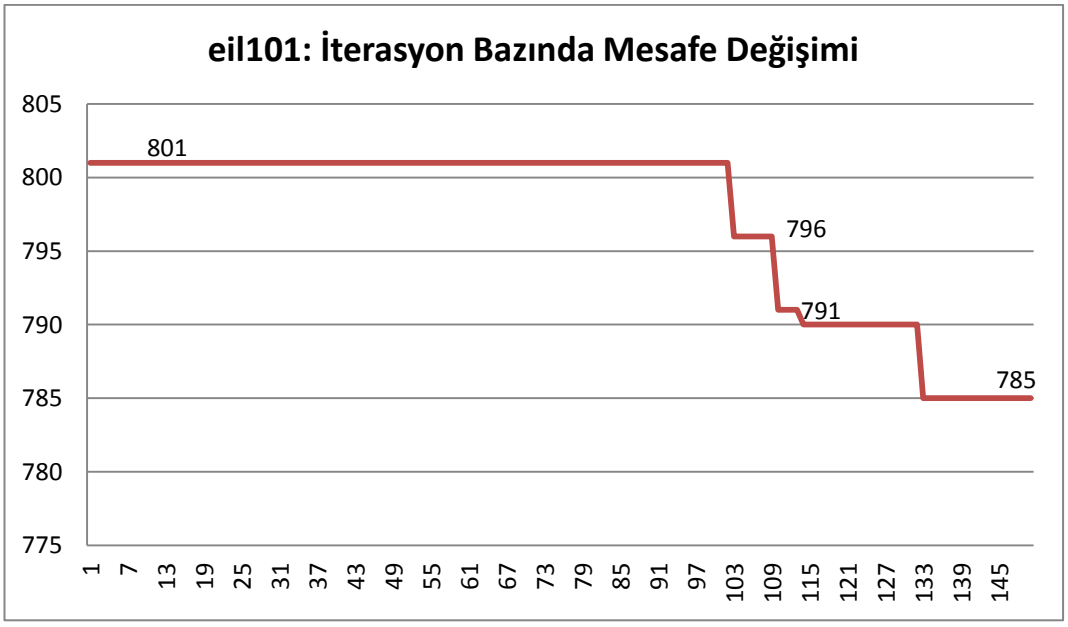



**Toplam Tur Uzunluğu = 574**

## EK C-7: eil101 Test Problemi İçin Bulunan Rota ve Çözüm Grafiği




**Toplam Tur Uzunluğu = 785**

# 8. **ÖZGEÇMİŞ**


| | |
|---|---|
| **Ad Soyad** | Tevfik Yıldırım |
| **Doğum Yeri ve Tarihi** | Denizli 30/05/1979 |
| **Lisans** | Eskişehir Osmangazi Üniversitesi |
| **Elektronik Posta** | tyildirim@pau.edu.tr |
| **İletişim Adresi** | Pamukkale Üniversitesi |
| | Bilgi İşlem Daire Başkanlığı |
| | Rektörlük Kınıklı Kampüsü Denizli |


**Yayın Listesi**

- **Yıldırım, T**., Mutlu, Ö., "Simetrik Gezgin Satıcı Problemi İçin Yeni Bir Meta-Sezgisel Yaklaşım", 33.Ulusal Yöneylem Araştırması ve Endüstri Mühendisliği ve IIE Kongresi, İstanbul, (2013).