T.C.
İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

## YÜKSEK LİSANS TEZİ

# ZAMAN PENCERELİ ARAÇ ROTALAMA PROBLEMLERİNİN POPÜLASYON TABANLI SEZGİSEL YÖNTEMLER İLE OPTİMİZE EDİLMESİ

**Çağrı KURAM**

**Endüstri Mühendisliği Anabilim Dalı**

**Endüstri Mühendisliği Programı**

**DANIŞMAN**
**Prof. Dr. Alp BARAY**

**Ekim, 2016**

**İSTANBUL**

Bu çalışma 19/10/2016 tarihinde aşağıdaki jüri tarafından Endüstri Mühendisliği Anabilim Dalı Endüstri Mühendisliği programında Yüksek Lisans Tezi olarak kabul edilmiştir.

**Tez Jürisi:**

Prof. Dr. Alp BARAY (Danışman)
İstanbul Üniversitesi
Mühendislik Fakültesi

Prof. Dr. Şakir ESNAF
İstanbul Üniversitesi
Mühendislik Fakültesi

Prof. Dr. Mehmet Mutlu YENİSEY
İstanbul Üniversitesi
Mühendislik Fakültesi

Prof. Dr. Selim ZAİM
İstanbul Teknik Üniversitesi
İşletme Fakültesi

Doç. Dr. Tarık KÜÇÜKDENİZ
İstanbul Üniversitesi
Mühendislik Fakültesi

20.04.2016 tarihli resmi gazetede yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, İstanbul Üniversitesi'nin abonesi olduğu intihal yazılım programı kullanılarak Fen Bilimleri Enstitüsü'nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

# ÖNSÖZ

Tüm tez çalışması boyunca değerli görüş, öneri ve zamanını benden esirgemeyen, her zaman en güzel yorumlarıyla tezime yardımcı olan ve beni yönlendiren, iş ahlâkına hayranlık ve saygı duyduğum danışman hocam sayın Prof. Dr. Alp BARAY'a teşekkür ederim. Tüm hayatım boyunca sabırla beni yetiştiren, güvenlerini, şefkatlerini her zaman üzerimde hissettiğim değerli aileme, maddi ve manevi her türlü desteği için teşekkürü borç bilirim. Tezimin literatür taramasına katkıda bulunan ablam Dr. Emel KURAM'a teşekkür ederim. Değerli görüşleri ile tezime katkıda bulunan Doç. Dr. Tarık KÜÇÜKDENİZ'e ve diğer jüri üyelerine teşekkür ederim. Son olarak çalışmam sırasında seferberlik ilan eden tüm arkadaşlarıma çok teşekkür ederim.

Ekim, 2016 Çağrı KURAM

# İÇİNDEKİLER

**Sayfa No**

# ŞEKİL LİSTESİ

**Sayfa No**

# TABLO LİSTESİ

**Sayfa No**

# SİMGE VE KISALTMA LİSTESİ

**Simgeler** **Açıklama**

$a_i$ : En erken varış zamanı
**A** : Ark kümesi
$b_i$ : En geç varış zamanı
$C^k$ : K aracının kapasitesi
$d_{ij}$ : i ve j noktaları arasında Öklit uzaklığı
**k** : Araç indisi
**N** : Müşteri sayısı
**M** : Araç sayısı
$s_i$ : Servis süresi
$t_{ij}$ : Seyahat süresi
**V** : Düğüm kümesi
$X_{ij}^k$ : İkili karar değişkeni
$\lambda_i$ : Lagrange çarpanı
$\tau_{ij}$ : Feromon miktarı
$\eta_{ij}$ : Sezgisel seçilebilirlik parametresi

**Kısaltmalar** **Açıklama**

**ARP** : Araç Rotalama Problemi
**AUARP** : Açık Uçlu Araç Rotalama Problemi
**ÇDARP** : Çok Depolu Araç Rotalama Problemi
**DARP** : Dinamik Araç Rotalama Problemi
**GA** : Genetik Algoritmalar
**GSP** : Gezgin Satıcı Problemi
**KKARP** : Kapasite Kısıtlı Araç Rotalama Problemi
**KKO** : Karınca Kolonisi Algoritması
**KUARP** : Kapalı Uçlu Araç Rotalama Problemi
**NP** : Non-deterministic polynomial-polinomiyal kararlı olmayan
**ÖDST-ARP** : Önce Dağıt Sonra Topla Araç Rotalama Problemi
**PARP** : Periyodik Araç Rotalama Problemi
**PSO** : Parçacık Sürü Optimizasyonu
**SARP** : Stokastik Araç Rotalama Problemi
**TA** : Tabu Arama
**TB** : Tavlama Benzetimi
**VNS** : Variable Neighborhood Search-Değişken Komşuluk Arama

# ÖZET


**YÜKSEK LİSANS TEZİ**

**ZAMAN PENCERELİ ARAÇ ROTALAMA PROBLEMLERİNİN POPÜLASYON TABANLI SEZGİSEL YÖNTEMLER İLE OPTİMİZE EDİLMESİ**

**Çağrı KURAM**

**İstanbul Üniversitesi**

**Fen Bilimleri Enstitüsü**

**Endüstri Mühendisliği Ana Bilim Dalı**

**Danışman: Prof. Dr. Alp BARAY**

Modern dünyada rota planlama lojistik süreçlerinde en dikkat çekici uygulama alanında yer almaktadır. Kötü yönetilen bu rota planlama şirketlere müşterilerine hizmet vermesi açısından sorun teşkil etmekte olup, diğer lojistik süreçleri için darboğaz oluşturabilmektedir. Hızla gelişen ve değişen küresel rekabet ortamı işletmeleri daha iyi süreç yönetimi için planlama yapmaya zorlamaktadır. Bu planlamanın doğru yapılması için hem müşteri memnuniyetine önem verilmeli hem de işletme maliyetleri düşürülmelidir. Araç rotalama problemleri türünden biri olan zaman pencereli araç rotalama problemi birçok araştırmacının çalışma alanını oluşturmaktadır. Bu tezde müşterilerine istekleri doğrultusunda hizmet vermek bunun yanında toplam araç taşıma maliyetlerini en aza indirmek için araç rotalama probleminin özel bir durumu olan zaman pencereli araç rotalama problemi çalışılacaktır. Bu rotalama sürecinde kesin çözüm yöntemleri hızlı cevap verememesinden dolayı en iyi çözümü garanti etmeyen fakat en iyi çözüme kısa sürede ve yakın çözümler veren popülasyon tabanlı sezgisel yöntemler kullanılacaktır.

Ekim, 2016, 160 sayfa



**Anahtar kelimeler:** Zaman penceresi, zaman pencereli araç rotalama problemi, genetik algoritmalar

# SUMMARY


**M.Sc. THESIS**

**OPTIMISING VEHICLE ROUTING PROBLEMS WITH TIME WINDOWS VIA POPULATION-BASED HEURISTICS**

**Çağrı KURAM**

**İstanbul University**

**Institute of Graduate Studies in Science and Engineering**

**Department of Industrial Engineering**

**Supervisor: Prof. Dr. Alp BARAY**

The routing planning takes place of logistics processes in the most remarkable implementation area in the modern world. This routing planning is managed in bad mood in case constitutes problems to companies to serve their customers, it makes bottleneck for the other logistics processes. Global competititon area changes and developes fast imposes planning better process management to entreprises. In order to be made this plan correct both the customer satisfaction should be paid attention and entreprises costs should be reduced. Vehicle routing problems with time windows is one kind of vehicle routing problems composes working area for most researchers. In order to serve the customers according to their requirement as well to minimize the total vehicle cost, it is studied on vehicle routing problems with time windows which is special kind of vehicle routing problems in this thesis. In this routing process the population-based heuristics does not ensure the best solutions but gives approximate solutions for the best solutions in short time are used because the exact solutions methods could not reply fast.

October, 2016, 160 Pages.



**Keywords:** Time windows, vehicle routing problem with time windows, genetic algorithms

# 1. GİRİŞ

Rekabet ortamında firmaların müşterilerine daha iyi hizmet vermesi ancak iyi bir planlama yapılarak ve güncel teknolojik gelişmeler takip edilerek gerçekleşmektedir. Giderek önem kazanan müşteri memnuniyeti firmaların tam zamanında müşteri ihtiyaçlarını karşılamaya itmektedir. Lojistik firmaları zaman ve maliyet öğelerini dağıtım sürecinde iyileştirme sağlayarak müşteri servis düzeyini yükseltmeyi amaçlamaktadır. Dağıtım maliyetleri lojistik süreçlerinde önemli bir yer tutmaktadır. Son zamanlarda lojistik firmaları müşterilerine en kısa sürede hizmet vermelerini dağıtım maliyetlerini düşürerek gerçekleştirmesi gerekmektedir. Bu nedenle dağıtım maliyetini azaltacak, aracın toplam hareket ettiği mesafeyi en küçükleyecek araç veya araçların ağ üzerinde en uygun rotaların bulunmasını sağlayacak amaçlar araç rotalama probleminin temel konusu olmaktadır.

Araç rotalama problemi bir veya birden fazla depodan farklı merkezlerde konumlanmış müşterilere hizmet vermek üzere birden fazla aracın optimum rotalara ulaşmasını planlama amacıyla çözüme ulaşılması gereken problemdir. Araç rotalama problemi (ARP), araç filosu ile minimum operasyon maliyetine ulaşmak için rota planlamasını kullanmaktadır. ARP literatürde ilk defa 1959 yılında Dantzig ve Ramser tarafından yerini almıştır. Yayınlanan bu çalışmada gezgin satıcı probleminden (GSP) yararlanılmıştır. Tek araç yerine aynı noktadan başlayıp aynı noktaya ulaşması gereken araç filosu kullanılmıştır. Benzin istasyonlarına benzin dağıtımı konusuyla ilgilenilmiştir. 1964 yılında Clarke ve Wright, Dantzig ve Ramser'in çalışmasına yeni algoritmalar ekleyerek geliştirmişlerdir. Araç rotalama problemlerinin temel özellikleri aşağıdaki gibidir:

- Aynı rotada bulunan her araç depodan yola çıkmalıdır.
- Her müşteri bir kez bir araç tarafından ziyaret edilmelidir.
- Araç kapasite kısıtı aşılmadan ve müşterinin talebi bölünmeden rotalama yapılmalıdır.
- Müşteri hizmetini yerine getiren araç tekrar depoya dönmelidir.

Araç rotalama problemine her bir müşteri için servise başlayacağı en erken ve en geç zaman kısıtlarının eklenmesiyle problem zaman pencereli araç rotalama problemine (ZPARP) dönüşmektedir. Zaman pencereli araç rotalama problemleri son zamanlarda literatürde geniş yer bulmaktadır. Zaman pencereli araç rotalama problemlerinde belirli talebe sahip müşterilerin belirli kapasiteli araçlarla araçların kapasitesi ve müşterilerin belirlediği en erken ve en geç zaman kısıtları aşılmadan hizmet verilmektedir. Bir araç ile rota kurulmakta olup, araç müşteri ziyaretini ve hizmetini gerçekleştirdikten sonra başlangıç noktası olan depoya dönmek zorundadır. Toplam seyahat mesafesi ve rota sayısının minimize edilmesi amaçlanmaktadır. Bu tür problemlerin çözümünde kesin, sezgisel ve meta-sezgisel çözüm yöntemleri kullanılmaktadır.

Zaman pencereli araç rotalama probleminin çözümünde kullanılan meta-sezgisel yöntemler bu çalışmada yer alacaktır. Bu çalışmada evrim teorisinden esinlenilerek geliştirilen genetik algoritmalar kullanılacaktır.

Bu tezin amacı,

- Klasik ve kesin yöntemlerle çözümü uzun ve zor olan ZPARP'ın popülasyon tabanlı sezgisel yöntemler ile kısa yoldan ve kısa zamanda optimuma yakın çözümler elde etmek.
- ZPARP çözümü için pek tercih edilmeyen popülasyon tabanlı sezgisel yöntemlerin hızını göstermek.
- Literatürde var olan çözüm yöntemlerini açıklamak.

Bu çalışma 5 ana bölümden oluşmaktadır. 1. Bölümden tezin konusu ve amacı özetle ifade edilmektedir. 2. Bölüm olan genel kısımlarda araç rotalama problemi hakkında genel bilgiler, önemi türleri hakkında genel bilgiler verildikten sonra, bu tez konusuna başlığı veren zaman pencereli araç rotalama problemleri hakkında bilgi verildikten sonra çözüm yöntemleri üzerinde durulmaktadır. 3. Bölümde genetik algoritmalar, karınca kolonisi algoritması ve parçacık sürü optimizasyonu hakkında detaylı bilgiler verilmektedir. 4. Bölüm bulgular kısmında 3. Bölümde teorik olarak ele alınan genetik algoritmalar ile literatürde yer alan bazı test problemleri çözülmekte olup, en iyi çözümlerden ne kadar sapma gösterdiği hakkında kıyaslama yapılmaktadır. Son bölümde ise sonuçlar yorumlanmakta ve geleceğe yönelik öneriler verilmektedir.

# 2. GENEL KISIMLAR

## 2.1. ARAÇ ROTALAMA PROBLEMİ

### 2.1.1. Genel Bilgiler

Lojistik sektöründe bulunan birçok nakliye firması Araç Rotalama Problemleri (ARP) ile günlük güzergâhlarını belirlemektedir. Lojistik firmaları maliyet etkinliğini sağlayarak müşterileri için her gün düzenli olarak sevkiyat planları oluşturmaktadır. Bu planları hazırlamak için müşteri ihtiyaçlarına sadık kalmak zorundadır. Bu yüzden müşterilerin de ihtiyaçlarına cevap verecek rotalama türü olan Zaman pencereli araç rotalama kullanılmaktadır. ARP, müşteri taleplerine hızlı cevap vermek için araç filosunun toplam sevkiyat maliyetini minimize etmeye çalışmaktadır. Böylelikle minimum maliyetli rotalar oluşturulmaktadır. Her bir müşteriyi tek bir araç ziyaret etmekte ve araç kapasitesi aşılmadan talepler dağıtılmaktadır. Araç için başlangıç noktası olan depo aynı zamanda bitiş noktasıdır. Her rota bir depodan başlamaktadır, araçlar talepleri önceden bilinen müşteri kümesine talepleri ulaştırıldıktan sonra aynı depoya varmak zorundadır. (Taş, ve diğ., 2014) ARP lojistik süreçlerinin içinde bulunan dağıtım alanında önemli bir problemdir. Tipik araç rotalama probleminde bir orijin (başlangıç noktası) bulunmakta, farklı veya aynı kapasiteye sahip araçlarla ürünlerin dağıtımı sağlanmaktadır. Toplam ulaştırma maliyeti araçların rota uzaklığına göre minimize edilmektedir. (Moghaddam, ve diğ., 2012)

Literatürde ARP ilk defa 1959 yılında Management Science dergisinde makale olarak yerini almıştır. Dantzig ve Ramser tarafından ortaya atılan bu çalışmaya göre Travelling-Salesman Problem-Gezgin Satıcı Probleminden (GSP) esinlenerek Truck Dispatching Problem- adlı makale ile araç rotalama probleminin ilk uygulamaları yapılmıştır. Çalışmada benzin istasyonuna benzin taşıyan tankerlerin rotaları doğrusal programlama yöntemi ile en iyiye yakın çözüm elde edilmiştir. Bu çalışmanın GSP'den farkı tek araç yerine araç filosunun rotalamada kullanılmasıdır. (Dantzig & Ramser, 1959) Bu çalışmanın yayınlanmasından 5 yıl sonra 1964 yılında Clarke ve Wright yeni bir yaklaşım olan tasarruf algoritması geliştirmişlerdir. Toplam taşıma maliyetini

minimize eden çalışma klasik ARP olarak adlandırılmaktadır. (Clarke & Wright, 1964) 1990'lar itibariyle ARP'deki müşteri ve depo sayılarının artması ve çözüm uzayının genişlemesi nedeniyle birçok metasezgisel yöntemler uygulanmaya başladı. 1993 yılında İbrahim Hassan Osman tarafından hem tabu arama algoritması hem de tavlama benzetimi algoritması kullanılarak ARP'ye yeni bir metasezgisel yaklaşım geliştirildi. Bu çalışma sayesinde hesaplama süresinde yaklaşık %50 iyileşme gözlenmiştir. (Osman, 1993) 1994 yılında Laporte ve arkadaşları tarafından tabu arama algoritması kullanılarak ARP'nin çözümüne yeni bir yaklaşım getirilmiştir. (Laporte, ve diğ., 1994)

Araç rotalama problemleri amaçları itibariyle gezgin satıcı problemine benzemektedir. Her iki problemde de amaçlar tüm dağıtım maliyetlerini minimize etme, müşteri servis düzeyini arttırma, hizmet verilen müşteri sayısını arttırmaktır. Araç rotalama probleminin gezgin satıcı probleminden farkları ARP'in birden fazla araca sahip olmasının yanı sıra, herhangi depo kullanması ve araçların belirli kapasitelerinin olmasıdır. (Macedo, ve diğ., 2011)

Klasik araç rotalama probleminde her müşteri için bir araç atanmalıdır. Müşteri talebinin araç kapasitesini aşmaması istenmektedir. (Kuşçu, 2009) Klasik araç rotalama problemlerinin amacı toplam rota uzunluğunu minimize ederek minimum maliyetli rotalar bulmaktır. (Reed, ve diğ., 2014) İlk şehir depodur, n adet şehir, i müşterisinden j müşterisine $d_{ij}$ ve Q kapasiteli m adet araç ile rotalama yapılmaktadır. Klasik araç rotalama probleminde amaçlar;

- Araçlar tarafından ziyaret edilen rotaların uzaklığı minimize edilmelidir.
- Müşterilere hizmet edecek toplam araç sayısı minimize edilmelidir.
- Minimum araç sayısı için toplam rotaların birbirine uzaklığı minimize edilmelidir.

Klasik araç rotalama probleminde kısıtlar;

- Araç kapasitesi aşılmamalıdır.
- Her bir aracın hizmeti depoda başlar, depoda biter.
- Araçların gidebileceği rota uzunluğu kapasitesini geçmemelidir.
- Her müşteriye bir kez hizmet verilmelidir.

- Her müşteriye bir araç hizmet verebilir.
- Zaman penceresi kısıtı aşılmamalıdır.

Yukarıdaki kısıtlara göre amaç fonksiyonuna ulaşılmaya çalışılır. (Ghoseiri & Ghannadpour, 2010)

ARP'de yol ağı, ürünlerin müşteriye ulaştırılmasında veya müşterilerden ürünlerin teslin alınmasında kullanılmaktadır. Araç rotası arklar ile temsil edilir. Köşeler ise müşteri yerleri ve depo olarak belirtilir. (Toth & Vigo, 2002)

Klasik ARP, G = (V,A) grafı ile ifade edilmektedir. V=$\{v_0, \dots, v_n\}$ düğüm kümesi olarak tanımlanmaktadır. Burada $v_1 \dots v_n$ düğümleri müşterileri temsil ederken, $v_0$ başlangıç düğümü olan depoyu temsil etmektedir. Bazen depo $n + 1$ ile temsil edilmektedir. A ark kümesini temsil etmektedir. A=$\{(i,j) : i,j \in V,\ i < j\}$ A ark kümesi pozitif $c_{ij}$ maliyeti ifade etmektedir. i düğümünden j düğümüne gitmek için harcanan seyahat süresini ile ilişkilidir. Her müşteri yine negatif olmayan $q_i$ talebine sahip olup, C kapasiteli m adet araçtan oluşan araç filosu depoda bulunmaktadır. Depoya ait talep $q_0=0$ ve $q_i \leq C$ olmalıdır. Her aracın bir rotası olabilir, minimum araç sayısı toplam talebin araç kapasitesiyle bölünmesinden bulunabilir. (Toth & Vigo, 2002)

Klasik ARP'de kullanılan semboller ve açıklamaları Tablo 2.1'de gösterilmektedir:

**Tablo 2.1**: Semboller ve Açıklamaları.

| SEMBOL | AÇIKLAMALAR |
|---|---|
| G=(V,A) | Şebeke kümesi |
| V | Düğüm kümesi |
| A | Ark kümesi |
| n | Müşteri sayısı (Düğüm sayısı) |
| (i,j) | Müşteri i ve j arasındaki bağlantı |
| $d_{ij}$ | i ve j müşterileri arası uzaklık |
| $q_i$ | i müşterisinin talebi |
| k | Araç indisi |
| $C^k$ | k aracının kapasitesi |
| m | Araç sayısı |
| $x_{ij}^k$ | İkili karar değişkeni |

Klasik ARP'nin doğrusal modeli aşağıdaki gibi formüle edilmektedir:

$$x_{ij}^k = \begin{cases} 1, & eğer\ k\ nolu\ araç\ i,j\ arasında\ hareket\ ederse, \\ 0, & diğer\ durumlar \end{cases}$$

Amaç fonksiyonu:

$$fs = min \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} \sum_{k=1}^{m} x_{ij}^k \tag{2.1}$$

Kısıtlar:

$$\sum_{i=0}^{n} q_i \ \sum_{j=0}^{n} x_{ij}^k \leq C^k;\ \forall\ k \in \{1, \dots, m\} \tag{2.2}$$

$$\sum_{i=0}^{n} x_{il}^{k} - \sum_{j=0}^{n} x_{lj}^{k} = 0;\ \forall\, k \in \{1, \dots, m\}, l \in \{1, \dots, n\} \quad (2.3)$$

$$\sum_{i=1}^{n} x_{i0}^{k} \leq 1;\ \forall\, k \in \{1, \dots, m\} \quad (2.4)$$

$$\sum_{j=1}^{n} x_{0j}^{k} \leq 1;\ \forall k \in \{1, \dots, m\} \quad (2.5)$$

$$\sum_{k=1}^{m} \sum_{j=1}^{n} x_{ij}^{k} = m; i = 0 \quad (2.6)$$

Yukarıda (2.1) ile ifade edilen denklem amaç fonksiyonudur. Buna göre toplam seyahat edilecek yolun diğer bir ifadeyle maliyetin minimize edilmesini göstermektedir. (2.2) nolu kısıt kapasite kısıdını ifade etmektedir. Hiçbir araca kapasitesinden fazla yükleme yapılamaz. (2.3) kısıt müşteriye giden aracın tekrar depoya (başlangıç noktasına) dönmesi gerekir. (2.4) ve (2.5) ile ifade edilen kısıtlar her bir aracın bir defa rotalanması gerektiğini ifade etmektedir. (2.6) kısıdı firmadan servise başlayacak araç sayısının m adet olması gerektiğidir.

Araç rotalama probleminde müşterilerin karakteristik özellikleri;

- Müşterilerin konumu bellidir.
- Müşteriden alınacak veya müşteriye sevk edilecek ürün miktarı veya ürün çeşidi farklıdır.
- Müşteriye hizmet verilecek süre müşterinin açık olduğu süredir. Müşterinin çalışma süresi içinde hizmet verilir.
- Müşteri önceliği uygulanır. (Toth & Vigo, 2002)

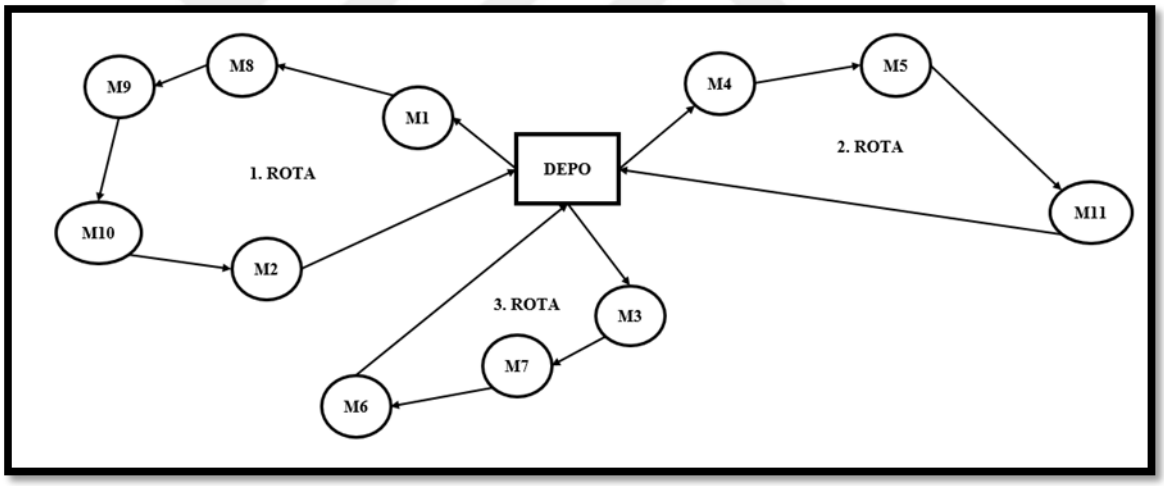
Araç rotalama probleminde araçların karakteristik özellikleri;

- Araç kapasitesi aşılamaz; araç kapasitesi, maksimum hacim, maksimum taşıdığı palet, maksimum ağırlık olarak ifade edilir.
- Aracın başlangıç noktası aynı zamanda bitiş noktasıdır.
- Taşınacak ürünlere göre araçlara yükleme yapılır.
- Araçların rota başına harcadığı bütün maliyetler bilinir. (Toth & Vigo, 2002)

Araç rotalama problemi (ARP) birden fazla müşteriye hizmet vermesi için birden fazla araçla ürün/hizmetlerin dağıtımını sağlamaya odaklanmaktadır. Bu sayede mevcut talepler minimum operasyon maliyeti ile müşterilere ulaştırılacaktır. (Wang & Chen, 2012)

ARP hizmet ve ürünlerin müşterilerine ulaştırılmasında kullanılmaktadır. Birçok ARP çeşidi bulunmaktadır. ARP, NP-zor problem olmasından dolayı çözümünde birçok yöntemler kullanılmaktadır. ARP önemli bir kombinatoryal optimizasyon problemidir. Bu yüzden literatürde çok geniş optimizasyon çözümleri yer almaktadır. ARP zengin sezgisel ve metasezgisel çözüm yöntemleri kullanılarak işletmeleri etkin stratejilere ulaştırmaktadır. (Kumar & Panneerselvam, 2012)

ARP dağıtım yönetiminde kullanılan önemli bir problemdir. Bu yüzden ARP çözümünde kullanılan yöntemler karar alma sürecini etkiler.




**Şekil 2.1:** Klasik ARP.

Şekil 2.1'de klasik ARP modelinin şekilsel gösterimi verilmiş olup, M ile belirtilen düğümler her bir müşteriyi (talep noktasını) göstermektedir. Bu şekle göre 11 adet M ile belirtilen müşteriler ile bir depo bulunmaktadır. Müşterilerin ihtiyaçlarını sağlamak için 3 farklı rota uygulanmaktadır. Düğümler müşterileri, oklar ise arkı göstermektedir.

Literatürde ARP'in çözümüne yardımcı olan performans ölçütleri toplam rota uzunluğu, müşterinin hizmet memnuniyeti, bir çevrimdeki rota sayısı, her bir çevrimdeki rota süresidir. Bu ölçütler hem etkin güzergâhların oluşturulmasını hem de müşteri memnuniyetini sağlamaktadır. (Tüfekçier, 2008)

### 2.1.2. Araç Rotalama Problemi Türleri

Araç rotalama problemleri en az maliyetle müşteri ihtiyaçlarını karşılamak için uygun rotaları belirlemek için birçok çözüm yöntemleri kullanmaktadır. (Vidal, ve diğ., 2014)

Araç rotalama problemlerinin çözümünün zorlaşması, her firmanın kendine göre kısıtlar koyması dağıtım yönetiminde etkin rol oynayan yöneticileri yeni yöntemler kullanmaya zorlamaktadır. Bu yöntemler ile ARP daha fazla çeşide sahip olurken aynı zamanda daha fazla kısıda sahip olan yeni bir ARP oluşmaktadır.

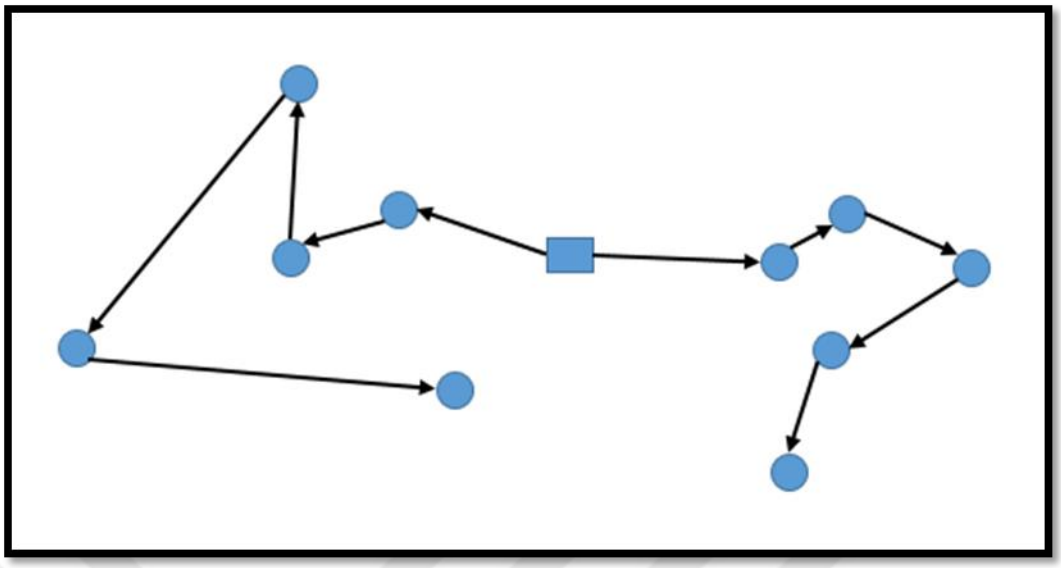
#### *2.1.2.1. Açık Uçlu Araç Rotalama Problemi*

Açık Uçlu Araç Rotalama Problemi-Open Vehicle Routing Problem (AUARP) klasik araç rotalama probleminden türetilmiş olup, aracın müşterilere hizmet verdikten sonra başlangıç noktasına geri dönmemesi durumunu ifade etmektedir. Bu tür araçların dönüş planı yapılmamaktadır. Firma sahipleri bu araçları firmalarında bulundurmadıkları için kiralama yoluna giderler. Bu tür uygulamalar literatürde 3. Parti lojistik problemi olarak da adlandırılmaktadır. (Marinakis & Marinaki, 2014) Önemli ürünlerini müşterilerine ulaştırmak için lojistik müdürleri özel araç veya dış kaynak kullanımı arasında seçim yapmak zorundadır. (Liu & Jiang, 2012)

Klasik araç rotalama problemine eklenecek kısıt aşağıdaki gibidir:

$$\sum_{i=1}^{n} x_{i0}^{k} = \sum_{j=1}^{n} x_{0j}^{k} = 1;\ \forall\, k \in \{1, \dots, m\} \qquad (2.7)$$

(2.7) nolu denklem ile başlangıç noktasına geri dönmesi engellenir. Klasik araç rotalama problemine göre $x_{ij}^{k}$ ile belirtilen ikili karar değişkenin toplam değeri 1 olmalıdır.



**Şekil 2.2:** Açık Uçlu Araç Rotalama Problemi.

Şekil 2.2'de Açık Uçlu Araç Rotalama Probleminin şekilsel gösterimi verilmektedir. 10 düğüm, bir başlangıç noktasından oluşan bu sistemde araç rota ziyaretini gerçekleştirdikten sonra başlangıç noktasına geri dönmemektedir.

#### *2.1.2.2. Kapalı Uçlu Araç Rotalama Problemi*

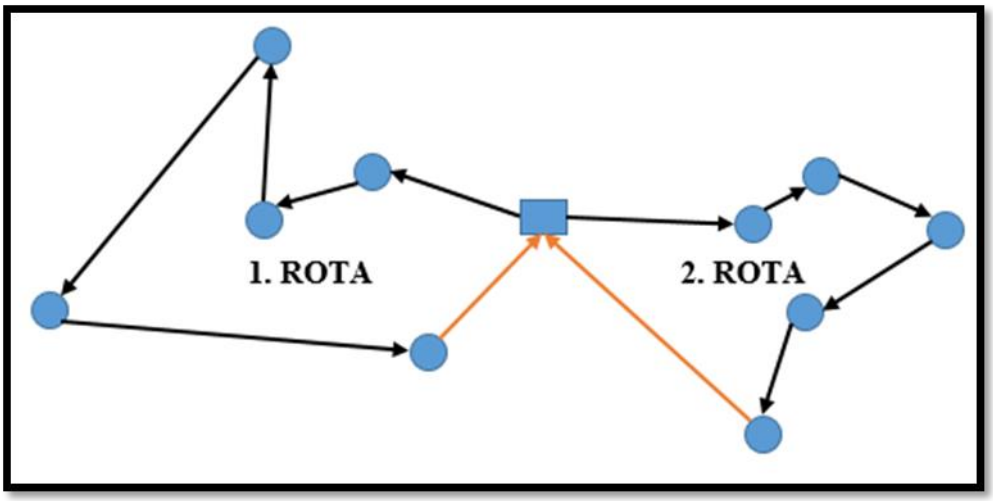
Kapalı Uçlu Araç Rotalama Problemi (KUARP), her rotanın başlangıç noktası olan işletmede başlayıp, tekrar işletmede son bulması olarak ifade edilmektedir. Klasik araç rotalama problemi KUARP olarak da bilinmektedir. (Liu & Jiang, 2012)

Gezgin Satıcı Problemleri (GSP) bu tür problemlerden türetilmektedir.

Klasik araç rotalama probleminde de bulunan aşağıdaki kısıtla sistem kapalı uçlu araç rotalama problemi olmaktadır:

$$\sum_{i=1}^{n} x_{i0}^{k} = \sum_{j=1}^{n} x_{0j}^{k} \leq 1;\ \forall\, k \in \{1, \ldots, m\} \qquad (2.8)$$

(2.8) nolu denkleme göre x ikili karar değişkeni değeri birbirine eşitlenerek başlangıç noktasına aracın geri dönmesi sağlanmaktadır. $x_0$ ile ifade edilen başlangıç noktası depodur.




**Şekil 2.3:** Kapalı Uçlu Araç Rotalama Problemi.

Şekil 2.3'te Kapalı Uçlu Araç Rotalama Probleminin şekilsel gösterimi verilmektedir. 10 düğüm, bir başlangıç noktasından oluşan bu sistemde araç rota ziyaretini gerçekleştirdikten sonra başlangıç noktasına geri dönmektedir. 2 farklı rota bulunmaktadır.

#### *2.1.2.3. Kapasite Kısıtlı Araç Rotalama Problemi*

Kapasite Kısıtlı Araç Rotalama Problemi (KKARP), klasik ARP problemi olarak da bilinmektedir. Araç filosunun homojen kapasiteli olarak müşterilerine hizmet vermesi için KKARP'den yararlanılmaktadır. Amaç toplam ziyaret edilen rotaların maliyetini minimize etmektir. Rota üzerinde bulunan müşterilerin talebi aracın kapasitesini aşmamalıdır. (Jin, ve diğ., 2014)

Bir veya daha fazla depo (başlangıç noktası) bulunan işletmelerin belirli talebe sahip müşterilerinin araç kapasitesini aşmadan yükleme yapılabilmesi problemi olarak da tanımlanırlar. KKARP, minimum toplam maliyet ile maksimum rota kümesine dayanmaktadır. Klasik ARP gibi depo veya depolar başlangıç noktası olup, araçlar tekrar başlangıç noktasına dönmektedir. Her müşteri bir araç ile ziyaret edilmelidir. Rotada bulunan müşterilerin toplam talebi araç kapasitesinden küçük olmak zorundadır. (Nazif & Lee, 2012)

(2.2) nolu formül ile kapasite kısıtı sağlanmaktadır. Kapasite kısıtlı ARP'nin bazı değişik durumları da bulunmaktadır. Bu değişiklikle gerçek hayattaki karmaşık ARP problemleri çözümlenmektedir. Bunlardan birisi geleneksel amaç fonksiyonundan farklı

olarak toplam uzaklık ve araçlarla ziyaret edilen rotaların zamanını minimize edilmesidir. Böylece en uzun turun boyutu minimize edilir. Bazı çeşitlerinde araçların sayısı da minimize edilmektedir. Yine KKARP için yapılan bazı çalışmalarda toplam müşteri varış zamanları minimize edilmektedir. (Lysgaard & Wøhlk, 2014)

#### *2.1.2.4. Önce Dağıt Sonra Topla Araç Rotalama Problemi*

ARP'nin özel bir türü olan Önce Dağıt Sonra Topla Araç Rotalama Problemi (ÖDST ARP) literatürde önemli bir yer tutmaktadır. 2 çeşit müşteri ilişkisine sahiptir. Müşteriden ürünleri alır ve bu ürünleri müşterilere ulaştırır. Aynı bölgede bulunan şirketler bitmiş ürünlerini müşterilerine ulaştırdıktan sonra tedarikçilerinden üretim merkezlerine hammadde tedariği yapmaktadır. Klasik ARP'ye ilave kısıt eklenerek ÖDST-ARP oluşturulmaktadır:

- Tüketicilere ulaştırılacak bitmiş ürünler ve tedarikçilerden alınacak hammaddeler araç kapasitesini aşmamalıdır.
- Her araç en az bir tüketiciyi (müşteriyi) ziyaret etmelidir.
- Tedarikçi ziyaretinden önce tüketicilere ürün teslimi yapılmalıdır.
- Tüketiciler ayrı bir grup, tedarikçiler ayrı bir grup olmalıdır. (Cuervo, ve diğ., 2014)

Firmalar bu yöntemle taşıma maliyetleri düşürmeyi amaçlamaktadır. Bazı çalışmalarda gerekli araç sayısı minimize edilmektedir. Günümüz çevresel sorunlara da çözüm bulunacak kısıtlar da bu probleme eklenmektedir. (Juan, ve diğ., 2014)

En genel ifadeyle klasik ARP'ye ek olarak her rotada öncelikli olarak tüketicilere ürünler teslim edilmektedir. (Keçeci, 2008)

#### *2.1.2.5. Periyodik Araç Rotalama Problemi*

Periyodik Araç Rotalama Problemi (PARP), her bir müşteri için aynı planlama periyoduna sadık kalarak ziyaretlerin gerçekleştirilmesidir. Müşteri için hizmet frekansı $f$ belirlenir ve müşteri bu hizmet frekansı ile ziyaret edilir. Diğer kısıtlar klasik ARP'ye göre uygulanmaktadır. Toplam ziyaret edilen süre minimize edilmeye çalışılır. Atık toplama, asansör bakımı ve tamiri, otomat makinesi yenilemesi belirli süreler ile yapılmaktadır. (Yu & Yang, 2011)

#### *2.1.2.6. Stokastik Araç Rotalama Problemi*

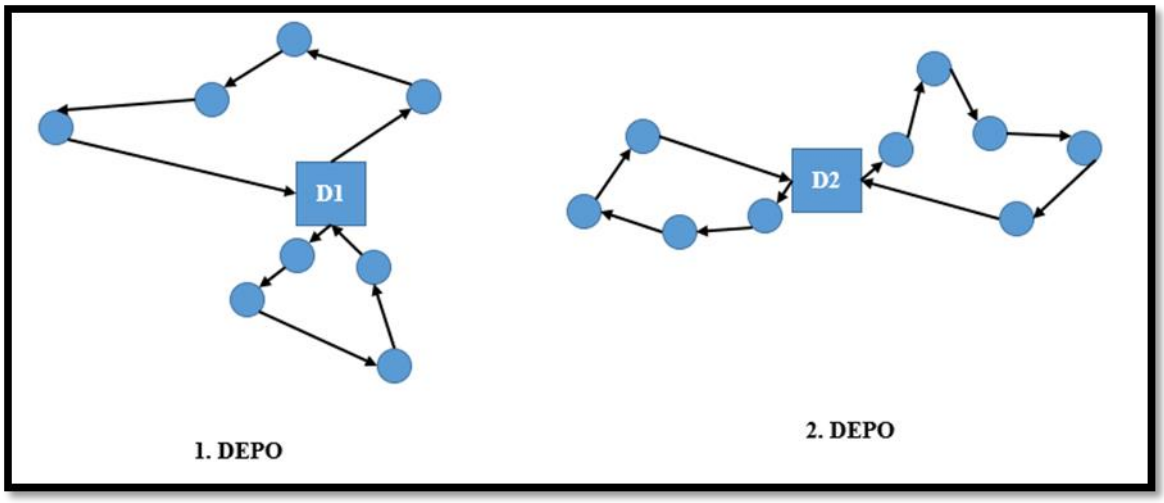
Klasik araç rotalama problemlerinde müşteri talepleri, müşteri sayısı, müşteri hizmet zamanları deterministiktir. Stokastik Araç Rotalama Problemi (SARP) Dinamik Araç Rotalama Problemi (DARP) olarak da ifade edilmektedir. En genel kullanım şekli müşteri taleplerinin daha önceden bilinmemesi durumudur. (Gauvin, ve diğ., 2014)

SARP beklenmeyen durumlarda ortaya çıkarak, firma yönetimini yeni karar almaya zorlamaktadır. Bu durumda yeni müşteri talebi veya hizmet süresinin değişmesi firmayı etkilememektedir. Kurye ile hizmet verilmesi işletmelerin son dönemde aldığı önemli bir uygulamadır. Toplam beklenen seyahat maliyetini minimize edilmesi amaçlanmaktadır. (Gauvin, ve diğ., 2014)

Amaç fonksiyonu optimize etmek için problem iki aşamada çözülmektedir. İlk aşamada rassal gerçekleşme değeri bilinmeden problem çözülür. İkinci adımda ilk aşamada çözülen değere düzeltici işlemler yapılır. Yeni müşteri talebi olduğu zaman kapasite aşımı da olabilecektir. Bu durumdan kaçınmak için probleme yeni kısıtlar eklenerek olasılıklı çözüm yöntemleri denenmektedir. (Mavrovouniotis & Yang, 2015)

#### *2.1.2.7. Çok Depolu Araç Rotalama Problemi*

Klasik ARP'de ve diğer tüm ARP çeşitlerinde başlangıç noktası olarak sadece bir depo bulunmaktadır. Çok Depolu Araç Rotalama Problemi (ÇDARP) birden fazla depo seçeneği ve araç filosu ile nihai müşteriye hizmet vermektedir. (Montoya-Torres, ve diğ., 2015)




**Şekil 2.4:** Çok Depolu Araç Rotalama Problemi.

Şekil 2.4 ile ÇDARP'in şekilsel gösterimi verilmektedir.

ÇDARP'ın amacı, toplam seyahat maliyetini minimize ederek tüm müşterilere talepleri ölçüsünde hizmet vermektir. Tedarikçi firmalar bu şekilde araç kapasitesini aşmadan birden fazla depo ile müşteri ihtiyaçlarını en iyi şekilde karşılamaya çalışmaktadır. (Rahimi-Vahed, ve diğ., 2015)

#### *2.1.2.8. Asimetrik Araç Rotalama Problemi*

Asimetrik Araç Rotalama Problemi (AARP), i düğümünden j düğümüne gidilmesi gereken mesafe ile j düğümünden i düğümüne gidilmesi gereken mesafenin birbirine eşit olmaması durumudur. (Almoustafa, ve diğ., 2013)

Düğümler arasını ifade eden maliyet c birbirine eşit değildir. ($c_{ij}$<>$c_{ji}$) Gerçek dünyada karşılaşılan durumlar bu rotalama türüne örnektir. (Subramanian, ve diğ., 2013)

#### *2.1.2.9. Zaman Pencereli Araç Rotalama Problemi*

Literatürde birçok çalışmada yer almış ve bu çalışmanın temelini oluşturulacak Zaman Pencereli Araç Rotalama Problemi (ZPARP) müşterilere hizmet verilmek üzere hizmet süresinin belirtilmesidir. ZPARP her müşteri için belirli bir zaman aralığı $[a_i , b_i]$ verilmektedir. Müşteri hizmeti en erken $a_i$ süresinde başlar, en geç $b_i$ süresinde biter. (Toth & Vigo, 2002)

### 2.1.3. Araç Rotalama Probleminin Önemi

Tedarik zinciri içinde yer alan işletmelerin rakip firmalar ile rekabet edebilmesi, toplam maliyetlerini azaltması ile müşteri istek ve ihtiyaçlarının en doğru şekilde karşılanması için dağıtım planlaması optimize edilmelidir. Lojistik faaliyetlerinin hem işletme hem de müşteri açısından özenle uygulanması gerekmektedir. Bir yandan lojistik sektöründe faaliyet gösteren işletmelerin rakiplerine göre daha düşük maliyetler ile rekabet etmelerinin sağlanması gerekirken, diğer yandan müşterilerin ihtiyaç ve isteklerine daha yüksek memnuniyet sağlanması, araç rotalama problemlerinin optimizasyonu ile mümkün olmaktadır. Lojistik yönetiminin en önemli maliyet kalemi olan dağıtım maliyetleri araç rotalanması en optimum şekilde yapılarak azaltılmaktadır.

Son yıllarda müşteri istek ve ihtiyaçlarının artması ve lojistik firmalarından daha iyi hizmet alma isteği dağıtım sürecini daha karmaşık duruma getirmektedir. Hizmet veya ürünün bir yerden başka bir yere transfer edilmesi bütün firmalar için önemli bir optimizasyon problemidir. Bu yüzden birçok firma lojistik sisteminde kombinatoryal optimal problemlere çözümler bulabilmek için sahip olduğu araçları rotalayarak veya çizelgeleyerek araç rotalama problemlerini optimize etmektedir. En etkin ve verimli rota kümesini bütün müşterilerin taleplerini karşılayarak bulan ARP, önemli bir araştırma konusudur. Her lojistik firması günlük, haftalık ve aylık rotalama planını hazırlamak için ARP'den faydalanmaktadır. Rotalamanın doğru yapılmaması müşterilerinin ihtiyaçlarını karşılayan lojistik firmaları için yüksek dağıtım maliyetlerine katlanmasına sebep olmaktadır.

Firmalar Pazar payını arttırmak için etkin araç rotaları oluşturarak hem maliyetlerini hem de kullanılan araç sayısını azaltmaktadır. Böylece müşteri hizmet düzeyi arttırılarak lojistik faaliyetleri düzgün yönetilmektedir. Bunun için karmaşık araç rotalama problemlerinin lojistik sektöründe çözümlenmesi gerekmektedir. Coğrafi koordinatları bakımından birbirinden ayrı bölgelerde bulunan müşterilere ürün veya hizmetin ulaştırılması araç rotalanması sayesinde mümkün olmaktadır. Müşterilerin taleplerini karşılamak için lojistik firmaları sahip oldukları araçlar ile minimum maliyet ile hizmet vermelidir. Minimum maliyeti sağlamak ve seyahat edilen müşterilerin toplam uzaklıklarını düşürmek için araçların doğru rotalanması önemlidir.

## 2.2. ZAMAN PENCERELİ ARAÇ ROTALAMA PROBLEMLERİ

Zaman pencereli araç rotalama problemi (ZPARP) araç rotalama problemlerinde sık sık kullanılan, ARP'ye zaman penceresi kısıtı eklenmiş çeşididir. Her bir müşteriye belirli süre içerisinde ulaşma zorunluluğu getirmektedir. Coğrafi olarak birbirinden bağımsız talebi ve hizmet süresi (zaman penceresi) önceden belli olan müşterilere bir depodan (başlangıç noktası) hizmet vermek amacıyla araç filosunun optimum dağıtımını tasarlama problemidir. ZPARP'ın amacı araç sayısını ve toplam araç seyahat uzaklığını minimize etmek, bu sayede toplam rotalama maliyetini en aza indirmektir. Ayrıca müşterilerin zaman penceresi kısıtı aşılmadan müşterilere en verimli şekilde hizmet verilmelidir. (Ghoseiri & Ghannadpour, 2010)

Her bir $i$ müşterisi için $[a_i, b_i]$ gösteriminde zaman penceresi adı verilen zaman aralığı verilmektedir. Belirli ve sabit bir sürede başlangıç noktasından hareket eden her bir araç her bir ark için seyahat süresi $[t_{ij}]$ geçirmektedir. Her bir müşteriye ulaştığı zaman müşteriye hizmet ettiği süre servis süresi $[s_i]$ olarak ifade edilmektedir. Her bir $i$ müşteriye müşterinin izin verdiği zaman penceresi süresi içinde hizmet verilmeye başlanmalıdır. Servis süresi $[s_i]$ kadar müşteri ihtiyaçları karşılanmalıdır. ZPARP'te araç erken müşteriye ulaşırsa zaman penceresinde ifade edilen başlangıç süresi $[a_i]$'ye kadar beklemek zorundadır. Hizmet verilen her müşteri için başlangıç ve bitiş sürelerinin yanı sıra servis ve seyahat süreleri de önceden belirlenmektedir. Başlangıçta rota sayısı belli değildir, çözüm sonucuna göre en uygun rota sayısı belirlenmektedir.

Zaman pencereli araç rotalama problemi ARP'in tüm kısıtlarını kullanmaktadır. Aracın başlangıç noktası olan depo aynı zamanda rotanın bitiş noktasıdır. Başlangıç saati sıfır kabul edilir. Zaman pencereli araç rotalama probleminin genel varsayımları aşağıdaki gibidir:

- Her rota sonunda başlangıç noktası-depoya ulaşılmalıdır.
- Tek bir başlangıç noktası-depo bulunmaktadır.
- Rota içinde bulunan her düğüm-müşteri bir defa ziyaret edilmelidir.
- Ziyaret edilen düğümlerin-müşterilerin toplam talebi araç kapasitesini aşmamalıdır.

- Her bir müşterinin talebi bölünmemelidir, daima sadece bir araçla hizmet verilmelidir.
- Rota içinde ziyaret edilen her müşteri hizmeti $[a_i, b_i]$ zaman aralığında başlamalı, $[s_i]$ servis süresini aşmayacak şekilde gerçekleştirilmelidir.
- Arklar simetrik ve asimetrik olabilir. (Toth & Vigo, 2002)

ZPARP, klasik ARP'de olduğu gibi (V,A) grafı üzerinde tanımlanır. V ile temsil edilen müşteri kümesidir. Müşteriler aynı zamanda düğümler ile temsil edilmektedir. Depo 0 düğümü ile gösterilirken, müşteriler düğüm noktasında 1,2…,n ile ifade edilir. Tüm rotaların başlangıç noktası 0'dır ve her rota n+1 düğümünde rotasını tamamlar. Klasik ARP'deki A ark kümesi olarak adlandırılan düğümler arası bağlantıları sağlayan rota yay setidir. M adet araç özdeştir ve klasik ARP'den farklı olarak maliyet $c_{ij}$, $[t_{ij}]$ seyahat süresi ile ilişkilidir. Seyahat süresi, i müşterisine hizmet verilen süreyi $[s_i]$ de kapsamaktadır. Her müşteri için $[a_i, b_i]$ ile ifade edilen zaman penceresi içerisinde hizmet verilmeye başlanmalıdır. Aracın zaman penceresi $a_i$'den önce varmasına izin verilirken, $b_i$ 'den sonra varmasına izin verilmemektedir.

Aşağıda ZPARP için matematiksel formülasyon bulunmaktadır.

Parametreler:

$x_{ij}^k$: İkili karar değişkeni

n: Toplam müşteri sayısı

m: Toplam araç sayısı

$d_{ij}$: i ve j düğümleri arası uzaklık

$a_i$: i müşterisine en erken varış zamanı

$b_i$: i müşterisine en geç varış zamanı

$C^k$: k aracının kapasitesi

$s_i$: i müşterisindeki servis zamanı

$q_i$: i müşterisinin talebi

$t_{ij}$: i'den j müşterisine gitmek için harcanan seyahat süresi

$y_i^k$: i düğümünde bekleme zamanı

$$x_{ij}^k = \begin{cases} 1, eğer\ k\ nolu\ araç\ i,j\ arasında\ hareket\ ederse, \\ 0, diğer\ durumlar \end{cases}$$

Amaç fonksiyonu:

$$fs = min \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} \sum_{k=1}^{m} x_{ij}^k \quad (2.9)$$

Kısıtlar:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} x_{0j}^k = 1;\ \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i0}^k = 1;\ \forall\ i,j\ \{1, \dots, n\} \quad (2.10)$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m} x_{i0} = \sum_{j=1}^{n} \sum_{r=1}^{m} x_{0j}\ ;\ \forall\ r \in n \quad (2.11)$$

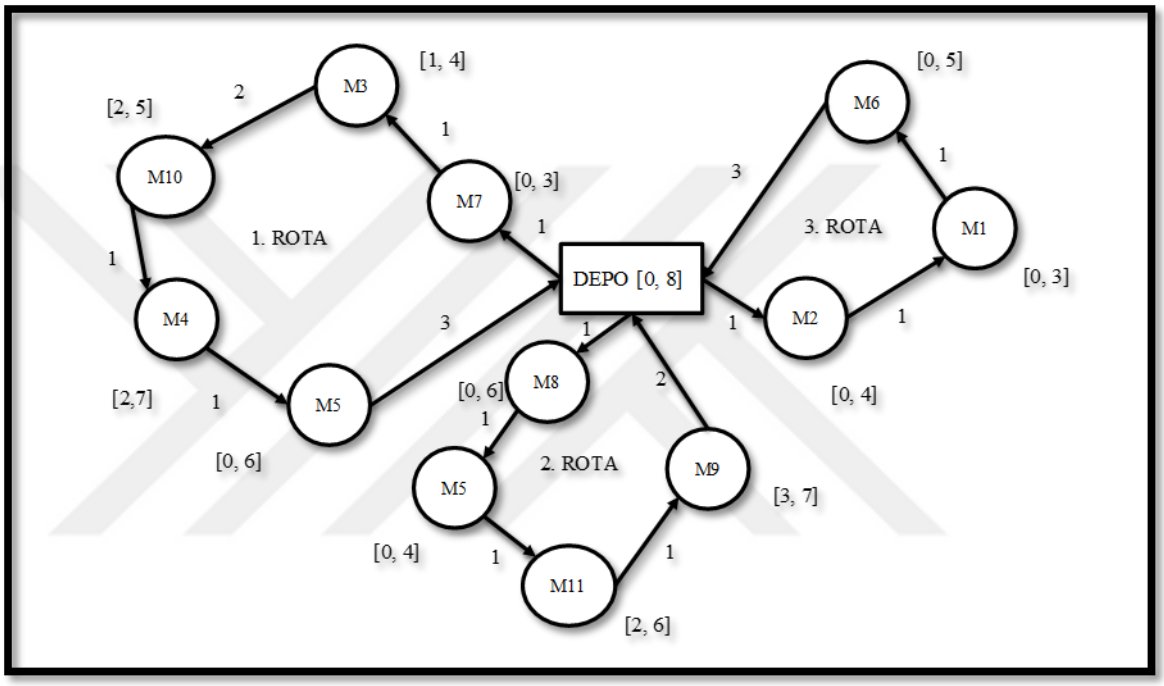
$$a_i \sum_{i=1;j=1}^{n} x_{ij}^k \leq y_i^k \leq b_i \sum_{i=1;j=1}^{n} x_{ij}^k\ ;\ \forall\ k \in m;\ \forall\ i \in n \quad (2.12)$$

$$y_i^k + s_i \geq T\left(1 - x_{ij}^k\right);\ \forall k = \{1, \dots, m\}; \forall\ (i,j) \in A \quad (2.13)$$

$$\sum_{i=0}^{n} q_i\ \sum_{j=0}^{n} x_{ij}^k \leq C^k;\ \forall\ k \in \{1, \dots, m\} \quad (2.14)$$

Yukarıda (2.9) nolu denklem ZPARP'ın en genel amaç fonksiyonu gösterimidir. (2.10) nolu kısıt her bir müşterinin bir defa ziyaret edilmesi gerektiğini göstermektedir. (2.11)

nolu kısıt ile müşteriden ayrılan aracın tekrar depoya dönmesi sağlanır. (2.12) nolu denklem zaman penceresi kısıdır. (2.13) nolu denklem k aracı ile hizmet edilen i müşteri için hizmet başlama zamanı $y_i^k$ ile servis süresi arasındaki ilişkiyi önleme kısıdır. (2.14) nolu denklem ile klasik araç rotalama probleminde de kullanılan toplam müşteri talebinin araç kapasitesini aşmaması gerektiğini gösteren kısıdır. (Toth & Vigo, 2002)




**Şekil 2.5:** Zaman Pencereli Araç Rotalama Problemi.

Şekil 2.5 ile ZPARP'in şekilsel gösterimi verilmektedir. Her müşteri için zaman penceresi ve seyahat süreleri önceden verilir. M1 müşterisi için zaman penceresi [0,3] olarak gösterilmiştir. M2 müşterisinin M1 müşterisine seyahat süresi 1 birimdir.

Literatürde yer alan ZPARP'in iki farklı kısıta sahip çeşidi bulunmaktadır. Bunlar Sıkı Zaman Pencereli Araç Rotalama Problemi (VRP with Hard Time Window) ve Esnek Zaman Pencereli Araç Rotalama Problemi (VRP with Soft Time Window) olarak isimlendirilmektedir.

Sıkı zaman pencereli araç rotalama probleminde araç $[a_i, b_i]$ zaman aralığı içinde gelmemişse hizmet verememektedir. $a_i$ Süresinden önce gelirse, hizmet verebilmesi için

bu süreye kadar beklemek zorundadır. $b_i$ Süresinden sonra gelmesi durumunda ise hizmet verememektedir.

Esnek zaman pencereli araç rotalama probleminde araç $[a_i, b_i]$ zaman aralığı içinde müşteriye ulaşmasa bile hizmet verebilmektedir. Fakat bu durumda ceza maliyeti uygulanmaktadır. (Drexl & Schneider, 2015)

Araçların servis sürelerini müşterilerine göre ayarlayan firmalar zaman pencereli araç rotalama problemi sayesinde toplam rota uzaklığını minimize etmenin yanı sıra, sürücülerin çalışma saatini iyileştirmeyi de amaçlamaktadır. (Melián-Batistaa, ve diğ., 2014)

### 2.2.1. Zaman Pencereli Araç Rotalama Problemi için Literatür Araştırması

Literatürde araç rotalama problemleri için birçok çözüm yöntemleri bulunmaktadır. Yöneylem araştırması için kullanılan kesin optimizasyon yöntemleri bu problemlerin çözümünde yetersiz kalmasından dolayı NP-zor problem için kesin olmayan bazı yöntemler geliştirilmiştir. Kesin olmayan bu yöntemler kesin çözüm yöntemlerine göre daha düşük hesaplama süresi sağlamaktadır. Sezgisel olarak ifade edilen yöntemler iyi kalitede çözümler geliştirmektedir. Bu çalışmada sadece zaman pencereli araç rotalama problemleri için literatür incelenmiş, sezgisel çözüm yöntemleri kullanılacaktır. 1950'lerin sonlarında araç rotalama problemi ilk olarak kesin çözüm veren tam sayılı programla optimize edilmiştir. 1990'larda araç rotalama problemleri çeşitlenerek daha karmaşık yapı halini aldığından sezgisel ve meta-sezgisel çözüm yöntemleri kullanılmaya başlanmıştır.

Literatür taraması kapsamında incelenen bütün zaman pencereli araç rotalama problemi makalelerinde Toth & Vigo (2002) 'nun araştırmalarına yer verilmektedir.

Ursani, ve diğ. (2011), zaman pencereli araç rotalama problemi için genetik algoritmaları kullanarak çözüm yöntemleri geliştirmiştir. Veri kümesi 1987'de Solomon tarafından sunulan çalışmadan alınmıştır. Toplam uzaklığın minimizasyonu hedeflenmiştir. Bu çalışmada genetik algoritmaların özel bir yöntemi kullanılmış olup, 2 fazda çözümler elde edilmiştir.

Yu ve Yang (2011), iyileştirilmiş karınca kolonisi algoritmasını kullanarak Solomon test problemlerini çözmeyi amaçladılar. Çaprazlama operatörü ile yerel optimum noktalara takılması engellendi, daha geniş arama uzayında optimum çözümler araştırıldı. Hem bir noktalı hem de iki noktalı çaprazlama operatörü ile alt rotalar geliştirildi.

Ding, ve diğ. (2012), yeni bir karınca kolonisi yaklaşımı önerdiler. Çalışmalarında iyileştirilen karınca kolonisi sayesinde düşük arama hızının önüne geçilmesi amaçlanmıştır. Solomon'un veri setleri ile kıyaslama yapılarak daha iyi sonuçların elde edildiği görülmektedir.

Xu, ve diğ. (2012), çok depolu zaman pencereli araç rotalama probleminin çözümü için melez değişken komşuluklu arama sezgiseli önerdiler. Başlangıç çözümü için m adet depodan n adet müşterilerine olan uzaklık bilgisi kullanılmıştır. Silkeleme aşamasında ekleme ve değiştirme operatörü kullanılmış olup, yerel minimumlara takılmayı önlemek için 2-opt ve tavlama benzetimi sürecine benzer kriterlere sahip olan Metropolis algoritması kullanılmıştır.

Vidal, ve diğ. (2014), zaman pencereli araç rotalama problemlerine yeni bir hibrit model önerdiler. Model hibrit genetik arama algoritması olarak ifade edilmektedir. Bu çalışmada yerel aramayı gerçekleştirebilmek için öğrenme mekanizması geliştirilmiştir. Çözüm kalitesini iyileştirme ve hesaplama etkinliğini artırma amacıyla geliştirilen bu yöntem ile test problemlerinde optimumluk elde edilmiştir.

Taş, ve diğ. (2014), esnek zaman pencereli araç rotalama problemi üzerine yeni bir çalışma önerdiler. Esnek zaman pencereli problemlerde kullanılan ceza maliyeti başlangıç çözümü olarak Solomon'un önerdiği en yakın komşuluk yöntemini kullanmıştır. Seyahat, aracın diğer maliyetleri ile erken ve geç hizmetten kaynaklanan ceza maliyetlerini içeren toplam maliyetinin minimizasyonu hedeflenmiştir. Tabu arama algoritması kullanılarak, toplam maliyet %12 düşürülmüştür, aynı zamanda hesaplama süresi de azaltılmıştır. Toplam maliyet ve rotalamada kullanılan araç sayısı düşürülmüştür.

Tan, ve diğ. (2015), ZPARP için bakteriyel besin arama algoritmasını geliştirdiler. Bakteriyel besin arama algoritmalarının yakınsama hızının zayıf olmasından dolayı

yerel minimum noktalarda takıldığı görülmektedir. Bunu önlemek için uyarlanabilen arama algoritması ve kapsamlı öğrenme stratejileri ile ZPARP'a uygulamaktadırlar. Çalışmada 8 müşterili 4 homojen araç kullanılmaktadır.

Armas ve Melian-Batista (2015), çalışmalarında değişken komşuluk arama algoritmasını kullanarak Solomon problemlerinde test ettiler. Gerçek dünya problemlerine uyarlayarak literatürdeki çözümlere yakınsamayı sağladılar. Müşteri hizmete başlama süresini azaltarak zaman pencereli problemlerde hizmeti geciktirme seviyesi iyileştirildi. Değişken komşuluk arama meta-sezgiseli kullanılarak gerçek dünyadaki problemlerdeki müşteri hizmet düzeyi optimuma ulaşıldı.

Miranda ve Conceição, (2016), stokastik hizmet ve servis süresine sahip zaman pencereli araç rotalama problemlerinin çözümüne odaklandılar. Araçların daha kısa sürede hizmete başlaması için müşteri bekleme süresini azaltmayı amaçladılar. Solomon test problemleri veri seti kullanılarak istatiksel sonuçlar metasezgisel yöntemler ile değerlendirildi.

Schneider (2016), zaman pencereli araç rotalama probleminin bir türü olan sürücüye-özgün zaman pencereli araç rotalama (vehicle-routing problem with time windows and driver-specific times) problemine odaklanarak sürücü seyahat ve çalışma süresini minimize etmeyi amaçladı. Böylece sürücü performans yönetimi de problemin çözümünde kullanıldı. Solomon (1987) veri setleri, farklı gruplama yöntemi ile hazırlanarak 4 farklı zaman penceresi yoğunluğu oluşturuldu. Bu gruplar yakınlık faktörleri sayesinde aynı tip problemlerin aynı grup içinde değerlendirmesi ile hazırlandı. Tabu arama algoritma meta-sezgiseli ile hem sürücülerin verimliliği hem de araçların toplam seyahat süresinde minimizasyon sağlandı.

Keskin ve Çatay (2016), çalışmalarında elektrikli araçların hem bateri şarjını sağlayacak istasyonlara rotalanması hem de müşterilerine zaman aralığı içinde hizmet verme sürecinin yönetilmesine odaklandılar. Uyarlanabilen büyük komşuluk algoritması ( adaptive large neighborhood search) meta-sezgisel yöntemi ile müşteri noktalarına müşterilerin istediği zaman aralığında en uygun hizmeti verebilmek için bateri şarj istasyonlarının ziyaretlerini de etkinliği amaçlandı. Başlangıç çözümü olarak depoya en yakın müşteri ilk hizmet noktası olarak atandı. Rotaya müşteri ekleme süreci, en uygun

noktalar çiftinin maliyeti hesaplanarak en düşük maliyete göre devam eder. Çalışma verileri olarak Solomon (1987)'un önerdiği ile Scheineder ve ark. (2014) tarafından oluşturulan 100 müşterili, 21 şarj istasyonlu veri seti geliştirildi.

### 2.2.2. Zaman Pencereli Araç Rotalama Problemi için Çözüm Yöntemleri

Araç rotalama probleminin tanımlandığı gibi zaman pencereli araç rotalama problemi de karma tam sayılı modeller ile ifade edilmektedir. Çözümünde kesin ve yaklaşık çözümler veren birçok yöntemler kullanılmaktadır. Kesin çözüm yöntemleri müşteri sayısının az, zaman penceresi olarak ifade edilen müşteri hizmet süre aralığının dar olduğu problemlerde en iyi sonuçlar vermektedir. Kesin çözüm yöntemleri müşteri sayısı artıkça optimale yakın çözüm bulma olasılığı azalmaktadır. Bu yüzden kesin çözüm yöntemleri büyük ölçekli problemlerin çözümünde optimal çözümü çok uzun süre içinde bulmaktadır.

Son yıllarda artan rekabet ile müşterilere en kısa ve en verimli hizmeti sağlama amacı içinde olan dağıtım firmaları rotalama problemlerine birçok kısıtı dâhil etmek zorunda kalmaktadır. Zaman pencereli araç rotalama problemleri için müşterinin ihtiyaçlarına göre çözüm yöntemleri geliştirilmektedir.

Zaman pencereli araç rotalama problemlerinin çözüm yöntemlerinde 2000'li yılların başlarına kadar kesin çözüm yöntemleri sıklıkla kullanılmaktadır. Son yıllarda gerçek hayatta zaman pencereli araç rotalama problemleri için sezgisel yöntemler geliştirilmektedir. Zaman pencereli araç rotalama problemi çözümü için kullanılan yöntemler kesin ve sezgisel çözüm yöntemleri olmak üzere ikiye ayrılmaktadır.

### 2.2.3. Kesin Çözüm Yöntemleri

Kesin çözüm yöntemleri ZPARP için optimum sonuç üretirler. ZPARP, karma tam sayılı model olarak ifade edildiği için çözümünde doğrusal programlamada kullanılan yöntemler uygulanmaktadır. Kesin çözüm yöntemleri optimum sonuç vermesine rağmen, yüksek boyutlu problemlerde çözüm süresi ve bilgisayarda kaplayacağı alan yönünden yetersiz kalmaktadır. Son yıllarda problemlerin karmaşık olması nedeniyle ZPARP literatüründe kesin çözüm yöntemlerinin kullanımı azalmaktadır.

Kesin çözüm yöntemleri matematiksel programlama metotlarını kullanmaktadır. ZPARP için kullanılan bu yöntemler 3 kısımda toplanmaktadır:

- Lagrange tabanlı kesin çözüm yaklaşımı (Lagragian relaxion)
- Sütun yaratma tabanlı kesin çözüm yaklaşımı (Column generation)
- Dinamik programlama tabanlı kesin çözüm yaklaşımı

#### *2.2.3.1. Lagrange Tabanlı Kesin Çözüm Yaklaşımı*

Lagrange tabanlı kesin çözüm yaklaşım sıkı zaman pencereli araç rotama problemleri için çözümler sunmaktadır. Her müşteriye sadece bir kez hizmet verilmesi kısıdı sağlanarak, amaç fonksiyonuna ceza kısıdı eklenir:

$$min \sum_{k\in m} \sum_{i\in N} \sum_{j\in N} (c_{ij} - \lambda_j) x_{ij}^k + \sum_{j\in m} \lambda_j \qquad (2.15)$$

(2.15) nolu denklem ile amaç fonksiyonunda ceza maliyeti $\lambda_j$ Lagrange çarpanı ile gösterilmektedir. Böylece j nolu müşteriye hizmet verilmesi sağlanır. (Karimi & Seifi, 2012)

Zaman pencereli araç rotalama problemlerinde zaman aralıkları kısıdı için Lagrange tabanlı kesin çözüm yöntemleri kullanılır. Bu kısıtların çözümü zor olduğu için Lagrange çarpanı ile problem alt problemlere ayrılarak kolaylıkla çözülür. (Desrosiers, ve diğ., 1995)

#### *2.2.3.2. Sütun Yaratma Tabanlı Kesin Çözüm Yaklaşımı (Dal-Değer Algoritmaları)*

ZPARP literatünde bulunan ilk çalışmada (Desroisers, ve diğ., 1984) 6 okul servisinin rotalamasında sütun yaratma (column generation) yaklaşımı kullanılarak bir taşıma problemi çözülmüştür. Eğer doğrusal programlama çok fazla değişken içeriyorsa, problem indirgenerek alt problemler oluşturulur. ZPARP için kullanılan bu yaklaşım kesme düzlemi (cutting plane) metoduna benzemektedir. Kesme adı verilen özel kısıtlar hazırlanarak, çözüm uzayı oluşturulur. Küme bölümlendirme problemine dayanarak ana problem formüle edilir. Ana problem çok fazla değişken içerdiğinden dolayı alt problemler yardımıyla çözülmektedir. (Qureshi, ve diğ., 2009)

ZPARP her biri araç kapasitesi ve zaman penceresi kısıtları içeren en kısa yol problemine ayrıştırılmaktadır. Böylece karmaşık kısıtlara ve zaman penceresine bağlı ceza maliyetli ZPARP için en iyi sonuçlar vermektedir. (El-Sherbeny, 2010)

#### *2.2.3.3. Dinamik Programlama Tabanlı Kesin Çözüm Yaklaşımı*

Dinamik Programlama çok değişkenli ve kompleks yapıdaki problemleri alt problemlere ayrıştırıp, bu alt problemlerin çözümüyle global optimum çözümü adım adım tekrarlayarak optimum zaman pencereli araç rotalama problemi oluşturmaktadır. (Ursani, ve diğ., 2011)

Dinamik programlamanın kullanım amacı toplam ceza maliyetlerini minimize etmektir. En yaygın kullanım alanı ise müşteriler için optimal hizmet başlama süresine karar vermektir. (Ibaraki, ve diğ., 2008)

### 2.2.4. Sezgisel Çözüm Yöntemleri

Sezgisel yöntemler, her hangi bir amacı gerçekleştirmek için geniş çözüm uzayında oldukça sınırlı alanda tarama işlemini yapan ve kabul edilebilir hesaplama zamanı içerisinde araç rotalama problemine uygun kaliteli çözümler üretmeye çalışan kriterler veya bilgisayar metotlarıdır. (Karaboğa, 2014)

Zaman pencereli araç rotalama problemi NP-Zor tipi problemdir. Klasik ARP gibi değişken sayısı arttıkça optimum sonuca ulaşmak için harcanacak zaman da artmaktadır. ZPARP çözümü için kesin çözüm yöntemleri optimum sonuç vermesine rağmen yetersiz kalmaktadır. Gerçek hayattaki ZPARP'in çözümünde sezgisel olarak ifade edilen çözüm yöntemleri ile iyi olarak nitelendirilen sonuçlar elde edilmektedir. Sezgisel çözüm yöntemleri ile optimuma yakınsama yapılmaktadır. Literatürde ZPARP için önerilen sezgisel çözüm yöntemleri ile kesin çözüm garantisi verilmemektedir.

ZPARP'ta kullanılan sezgisel çözüm yaklaşımı ile ilk olarak araç sayısı daha sonra uzaklıklar minimize edilmektedir. (Vidal, ve diğ., 2014)

Sezgisel yöntemler klasik ve meta-sezgisel yöntemler iki gruba ayrılmaktadır. Klasik sezgisel yöntemler bir ilk çözüm ile algoritmaya başlar. Klasik sezgisel yöntemler arama uzayında oldukça kısıtlı bir alanda çözüm bulmaya çalıştığı için daha kısa sürede iyiye yakın çözümler üretmektedir. Sezgisel yöntemler yakınsama özelliğine sahip olduğu için kesin çözümü garanti etmezler. Sezgisel yöntemlerin genel uygulama adımları aşağıdaki gibidir:

- Adım 1: Eldeki veriler ile olası durumlardan herhangi birini başlangıç çözümü olarak atama
- Adım 2: Başlangıç durumuna mümkün testler uygulayarak durumu değiştirme, yoksa Adım 2.'yi tekrar etme
- Adım 3: Durum değerlendirme
- Adım 4: Gereksiz durumların çözüm kümesinden atma
- Adım 5: Bölgesel optimum çözümün bulunması, aksi halde yeni değerler ele alınarak Adım 1'den işlemlerin tekrar etme

Klasik sezgisel yöntemler çoğunlukla zaman pencereli araç rotalama problemine özgü olduğu için farklı bir problemin çözümünde kullanılamaz. Klasik sezgisel yöntemler ZPARP uygulamasına göre ikiye ayrılmaktadır:

- Rota oluşturma (route-building) sezgiselleri
- Rota iyileştirme (route-improving) sezgiselleri

#### *2.2.4.1. Rota Oluşturma Sezgiselleri*

Rota oluşturma sezgiseli, uygun bir çözüm bulmak için rotalanmamış her bir müşteriyi mevcut rotaya ekleyerek oluşturulan yöntemdir. Araç kapasitesi ve zaman aralıkları kısıtları kontrol edilerek rotaya eklenecek aday müşteriler kümesi içinden seçim işlemi maliyet fonksiyonu kullanılarak gerçekleştirilir. Uygun olmayan atamalar ile başlayan çözüm adımı, her defasında düğüm arasına yeni bir yay ekleyerek uygun çözüme ulaşılır. ZPARP için en çok tercih edilen rota kurucu sezgisel Clark ve Wright (1964) tarafından geliştirilen tasarruf algoritmasıdır. Algoritma başlangıç noktası olarak deponun alınması ile başlar. Bu algoritma bütün olası müşteri-depo-müşteri olarak ifade edilen tekli rota ile başlar. Her bir müşteri düğümü için depodan ziyaret gerçekleştirildiği varsayılır. Her adımda maksimum tasarruf ile birleştirilen iki rota hesaplanır. Tasarruf sezgiselinin çözüm adımları aşağıdaki gibidir:

- Adım 1: Tüm müşteri düğümlerine başlangıç noktası depodan bir aracın gideceği varsayılarak, n adet araç turu oluşturulur.
- Adım 2: Zaman penceresi kısıtları ve talep yerlerinin birbirlerinden uzaklıkları (d) belirlenerek mesafe ve servis süresi matrisi oluşturulur.

- Adım 3: Adım 2'de oluşturulan mesafe matrisinden yararlanılarak kazançlar matrisi hazırlanır. Tasarruf değerleri aşağıdaki denklem ile hesaplanır:

$$s_{ij} = d_{i0} + d_{0j} - d_{ij}; \forall\, i,j\{1,\ldots.,n\}\; ve\; i \neq j \qquad (2.16)$$

(2.16) nolu denkleme göre kazanç değerleri bulunur. i ve j müşterilerini içeren rotaların birleştirilmesi durumuna bakılır.

- Adım 4: Tasarruf değerleri büyükten küçüğe doğru sıralanır.

Tasarruf algoritması dışında Solomon tarafından ZPARP'a uygulanan diğer rota kurucu yöntem en yakın komşuluk (nearest neighbourhood) sezgiselidir. En yakın komşuluk sezgiseli zaman odaklı yaklaşımdır. Başlangıç noktası olarak belirtilen depoya en yakın düğüm noktası ile problem çözümü başlar. Her rota için daha önce rota atanmamış müşteri düğümü araç kapasitesi ve zaman aralık kısıtı göz önüne alınarak mevcut müşteri düğümüne en yakın olanı rotaya ekleyerek çözüm oluşturmaktadır. her seferinde son eklenen düğüm noktasına göre en yakın nokta seçilir.

#### *2.2.4.2. Rota İyileştirme Sezgiselleri*

Rota iyileştirme sezgiseli daha iyi komşuluk çözümleri bulmak için uygun başlangıç çözümü üzerinde tekrarlı arama yapan sezgisellerdir. Mevcut durumu yerel arama yaparak iyileştirir. Rota oluşturma sezgisellerinden farkı uygun bir çözümle aramaya başlamasıdır. Her iterasyonla mevcut çözümden daha iyi amaç fonksiyonuna yeni bir çözüm elde edilmeye çalışılır, eğer çözüm bulunursa mevcut çözüm küçük değişiklikler yapılarak iyileştirilmiş olur.

Zaman pencereli araç rotalama problemlerinde en çok geliştirilen yöntem Lin'in λ-opt yöntemidir. Bu yöntemde λ rotadan çıkarılan köşelerin sayısını ifade etmektedir. Çıkarılan köşeler rotaya tekrardan olası bir noktaya eklenerek mevcut çözümü iyileştirecek sonuçlar elde etmeye çalışır. Mevcut çözümden daha iyi sonuç veren bir bağlantı bulunursa, yeni rota bu şekilde geliştirilir. 2-opt mekanizması iki köşenin çıkarılmasını ifade eder. Yeni bir rota elde etmek için açıkta kalan 2 rota tekrar birleştirilmektedir. 3-opt mekanizması ZPARP'ta kullanılan 2-opt mekanizmasının geliştirilmiş halidir. 3 köşenin 2-opt mekanizmasındaki çıkarılması ve açıkta kalan 3 rotanın birleşmesini sağlar.

Rota iyileştirme sezgiselleri uygun olan çözümün düğümleri araç rotası içinde veya rotalar arasında birbirleri arasında değişim uygulayarak uygun bir çözüm geliştirir. Bu sezgiseller zaman pencereli araç rotalama problemlerinde uygulanarak daha gerçekçi problem çözümleri sunar.

### 2.2.5. Meta-Sezgisel Çözüm Yöntemleri

90'lı yıllarda bilgisayar teknolojilerinin gelişmesi zaman pencereli araç rotalama problemleri gibi kombinatoryal optimizasyon problemlerinde yüksek programlama zekası kullanan yeni bir metot olan meta-sezgiselin kullanılmasını sağlamıştır. Bu problemlerde arama uzayında elde edilen çözüm kümesindeki yerel minimum noktalarının fazla olması ve en uygun çözümü ararken bu noktalardan çıkışın zor olması ARP için farklı çözüm yöntemlerinin uygulanmasını beraberinde getirmektedir. Meta, kelime olarak Yunancadan türetilmiş olup, tek başına daha büyük, daha üst seviye anlamına gelmektedir. Meta-sezgisel yöntemler optimuma yakın çözüm vermesinin yanı sıra diğer yöntemlere göre çok daha hızlı bir şekilde iyi kalitede çözüme ulaşır.

Sezgisel olarak ifade edilen yerel arama çözüm yöntemlerinin yerel minimum noktalarda takılmasını engellemek ve global optimuma daha yakın sonuçlar bulmak amacıyla zaman pencereli araç rotalama problemlerinin çözümünde meta-sezgisel yaklaşımlar kullanılmaktadır. Meta-sezgiseller sezgisel yöntemlere göre daha büyük problemlerde iyiye yakın çözümler veren güçlü tekniklerdir. Meta-sezgisel yöntemler adım adım çözüme ulaşmayı hedefler. Yinelemeli ileri sezgisel çözüm yöntemleri olarak da ifade edilir. Her adım bir önceki adımın geliştirilmesi ile elde edilir. İlk adımda elde edilen çözüm veya çözüm kümesi sonraki çözümlerin üretilmesine yardımcı olur. Meta-sezgisel yöntemlerin zaman pencereli araç rotalama problemlerinin çözümünde kullanılmalarının temel sebebi, sezgisel algoritmaların her probleme özgü olma özelliğini ortadan kaldırmaktır. Meta-sezgisel yöntemler sezgisel yöntemleri bir adım daha ileri seviyeye taşımaktadır. Meta-sezgisel yöntemler diğer klasik yaklaşımlara göre çözümün kötüleşmesine ve hatta uzaydaki arama sürecinde ortalama çözümlere izin verir. Yerel minimum noktalardan kurtulmak için daha kötü çözümlerinde kabul edilmesi ve algoritma içinde bir durdurma kriterinin bulunmayışı dezavantaj olarak ön plana çıkmaktadır. (Bräysy & Gendreau, 2005)

Klasik sezgisel arama algoritmaların aksine yerel optimum noktasına ulaşıldığında optimale yakın çözüm bulma umuduyla daha geniş çözüm alanında yeni çözümler araştırmaya devam ederler. Meta-sezgisel yöntemler diğer yaklaşımlara göre daha kısıtlı bölgede arama yaparlar. Meta-sezgisel yöntemler arama uzayında en elverişli çözüm noktalarında keşif yaparak yerel optimum noktalara takılmayı önler. Meta-sezgisel yöntemler, klasik sezgisel yöntemlere göre daha iyi yerel sonuçlar elde etmesine rağmen, iyi çözüm bulmak için sezgisel yöntemlere göre daha fazla zaman harcamaktadır. (Toth & Vigo, 2002)

Meta-sezgisel yöntemler, arama uzayında yaptığı araştırmalar ile kurum içi farklı seviyedeki kavramları birleştirmek için yapay zekâ uygulamalarından faydalanarak kendinden daha düşük seviyedeki sezgisel yöntemlere rehberlik etme stratejisini gösteren iteratif üretim ve geliştirme sürecidir. Meta-sezgisel yaklaşımın iteratif çözüm üretme biçimiyle her adımda önceki adımda elde edilen bir çözüm ve çözüm kümesinden yola çıkarak yeni çözümler üretirler. Meta-sezgisel yöntemlerin en önemli amacı zaman pencereli araç rotalama probleminde kapasite ve zaman penceresi kısıdını da hesaba katarak en iyiye yakın çözümleri elde etmek için arama uzayını hızlı bir şekilde araştırmaktır. Meta-sezgisel yöntemler daha önce bahsedildiği gibi probleme özgü değildir. Yaklaşık algoritma mantığına sahip olduğu için kararlı çözümler elde edilemez. Çözüm uzayında stokastik arama yaparak kararlı çözüm bulmaya çalışır. Yerel en iyi tuzaklardan kurtulmak için çeşitli mekanizmaları kullanırlar. İleri seviye meta-sezgisel aramaya rehberlik etmesi amacıyla arama sırasında elde edilen hafızaya bağlı olarak adım adım çözüme yaklaşırlar. Arama uzayında arama anlamında gelen çeşitlendirme (diversification) ve arama sürecinde elde edilen hafızanın işletilmesi anlamına gelen yoğunlaştırma (intensification) stratejilerini birlikte kullanırlar. Arama sürecine çözüm uzayında uygun bulunmayan alanların dâhil edilmesine izin verirler. Meta-sezgisel yöntemler, klasik sezgisel yöntemler gibi optimal çözüme veya çözüm kümesine ulaşılmasını sağlamazlar, daha hızlı ve güvenilir optimale yakın çözümler sunmaktadır.

Meta-sezgisel yöntemler klasik sezgisel yöntemlere göre daha kaliteli çözümler vermesine rağmen, daha fazla hafıza kullanımı, daha karmaşık tasarım ve iyi ayarlanmış parametrelere ihtiyaç duyar.

Zaman pencereli araç rotalama probleminde kullanılan meta-sezgisel yöntemler sezgisel yöntemlere rehberlik etmek için kullanılırlar. Meta-sezgisel yöntemleri sınıflandırmada birçok farklı yöntem kullanılmaktadır. ZPARP çözümünde kullanılan sınıflandırma popülasyon tabanlı (population-based search), yerel arama (local search) ve öğrenme mekanizma (learning mechanism) yaklaşımı şeklindedir. Meta-sezgisellerin popülasyon ve yerel arama meta-sezgisel yöntemlerine ayrılmasının nedeni aynı andaki çözüm sayısıdır. Bu gruplandırma yönteminin hibrit olarak kullanımı son yıllarda artarak devam etmektedir. (Cordeau, ve diğ., 2007)

Birçok meta-sezgisel yöntem ZPARP ‘e uygulanabilmektedir. Klasik sezgisellere göre daha iyi performans gösterir. Tablo 2.2’de ZPARP’de kullanılan meta-sezgisel yöntemler kronolojik sıraya göre verilmiştir:

**Tablo 2.2:** ZPARP'de Kullanılan Meta-sezgisel Yöntemler.

| YAZAR | YILI | KULLANILAN YÖNTEM | AMAÇ |
|---|---|---|---|
| **Amini; Javanshir; Moghaddam** | 2010 | Parçacık Sürü Optimizasyonu | Test problemlerini kullanarak gerçek dünyadaki ZPARP'daki Araç sayısını minimize etme |
| **Ghoseiri; Ghannadpour** | 2010 | Genetik Algoritma | Zaman penceresi ve kapasite kısıtları altında araç filo sayısı ve toplam seyahat süresini minimize etmek |
| **Ursani; Essam; Cornforth; Stocker** | 2011 | Genetik Algoritma | Toplam uzaklığı ve zaman kısıtlarının aşılmasını minimize etme, |
| **Yu; Yang** | 2011 | Karınca Kolonisi Algoritması | Zaman aralığı içerisinde her aracın müşteri talebini toplam ulaştırma süresini minimize etme |
| **Balseiro; Loiseau; Ramonet** | 2011 | Karınca Kolonisi Algoritması | Rota sayısını ve toplam hizmet süresini minimize etme |
| **Ding; Hu; Sun; Wang** | 2012 | Karınca Kolonisi Algoritması | Kapasite ve müşteri zaman kısıtları altında rota maliyetini minimize etme |
| **Wang; Chen** | 2012 | Genetik Algoritma | Toplam seyahat maliyetini ve araç sayısını minimize etme |
| **Vidal; Crainic; Gendreau; Prins** | 2013 | Genetik Algoritma | Toplam seyahat maliyetini minimize etme |
| **Taş; Jabali; Woensel** | 2014 | Tabu Arama Algoritması | Seyahat süresini minimize etme ve müşteri hizmet seviyesini maksimum yapma ( seyahat maliyeti, sabit maliyetler, araç hizmet maliyeti ve ceza maliyetlerini içeren toplam maliyeti minimize etme) |
| **Belhaiza; Hansen; Laporte** | 2014 | Tabu Arama Algoritması | Toplam seyahat süresini minimize etme |
| **Jiang; Ming; Poh; Teo** | 2014 | Tabu Arama Algoritması | Toplam ziyaret edilen müşteri sayısı maksimizasyonu, araç maliyeti ve toplam seyahat süresini minimize etme |
| **Tan; Lin; Wang** | 2015 | Bakteriyel besin arama optimizasyonu | Toplam seyahat maliyetini minimize etme |

| | | | |
|---|---|---|---|
| **Armas; Melián-Batista** | 2015 | Değişken komşuluk arama algoritması | Toplam araç sayısını minimize etme |
| **Brito; Martinez; Moreno; Verdegay** | 2015 | Değişken komşuluk arama algoritması, GRASP, KKO | Toplam seyahat süresimi minimize etme |

#### *2.2.5.1. Popülasyon Tabanlı Meta-Sezgisel Yöntemler*

Popülasyon genellikle biyoloji biliminde kullanılan ve aynı yerde belirli bir zamanda yer alan aynı türe ait bireyler topluluğu olarak ifade edilir. Genetik biliminde ise birden fazla genin belirli bir düzenle bir araya gelerek oluşturduğu kromozom topluluğudur. Kombinatoryal problemlerin çözümünde kullanılan popülasyon ise çözüm uzayında aynı zamanda belirli sayıda düğümlerin oluşturduğu çözüm kümesidir. Popülasyon tabanlı meta-sezgisel yöntemler ZPARP gibi kombinatoryal problemlerin çözümünde en çok kullanılan yöntemlerdendir. Evrim teorisine göre çözümü bulur, tek bir çözüm yerine birden fazla çözümden oluşan çözüm kümesi ile çalışır. Diğer bir ifade ile aynı zamanda birden fazla çözümü ele alarak, çözümler kümesinden faydalanarak yeni çözüm kümesi üretilmektedir. Çözüm uzayında aramayı daha önceden belirlenen kurallar çerçevesinde gerçekleştirerek çözümler kümesi bulan yöntemdir. Popülasyondaki her üye arama süreci boyunca kuşakların değişmesiyle yeni bir çözümü oluşturur. Popülasyon tabanlı algoritmalarda arama süreci birden fazla çözüm noktaları kümesinin aynı anda daha iyi çözümü bulmak için yapılan hareketler ile başlar.

Popülasyon tabanlı meta-sezgiseller çözüm havuzundaki adım adım iyileştirmeyi kapsar. İlk olarak popülasyon başlangıç durumuna hazırlanır, daha sonra yeni çözüm popülasyonu oluşturulur, bu yeni oluşturulan popülasyon bazı seçim prosedürlerine göre yeni çözüme katılır.

ZPARP'de en çok kullanılan popülasyon tabanlı meta-sezgisel yöntemler genetik algoritmalar (GA), uyarlanabilen bellek prosedürleri (adaptive memory) ve parçacık sürü optimizasyonudur (PSO).

Genetik Algoritmalar (GA), hem doğal seçilim hem de doğal genetikten esinlenmiş sayısal optimizasyon algoritmaları olarak ifade edilir. Pratik mühendislik optimizasyon problemlerinin çözümünde kullanılan popülasyon tabanlı yöntemlerdir. GA, zaman pencereli araç rotalama problemi gibi kararlı ve analitik kesin çözüm yöntemlerinin

yetersiz kaldığı problemlerin çözümünde kullanılmaktadır. GA, problemi incelemek için rasgele başlangıç çözümü ile değil çözüm kümesiyle başlar. Bu çözüm kümesi popülasyon olarak adlandırılır. Popülasyonu oluşturan her birey "kromozom" olarak ifade edilir. Popülasyon önceden tanımlanan uygunluk fonksiyonuna göre değerlendirilmektedir. Kromozomlar, jenerasyon olarak ifade edilen ardışık iterasyonlar ile geliştirilmektedir. Her bir jenerasyonda, verilen popülasyonun değiştirilmiş ve birleştirilmiş olarak yeni popülasyon oluşturulur. Birleşmiş kromozomlar "çaprazlama (cross over)" olarak ifade edilirken, var olanı değiştirme "mutasyon" olarak bilinmektedir. Çaprazlama yeni çift kromozom üretmede kullanılan rasgele değerlerle birleşmiş bir süreçtir. Mutasyon operatörü gen değerlerinde beklenmeyen değişiklikleri ortaya çıkarmada kullanılmaktadır. Çaprazlamaya ve mutasyona kromozom seçimi uygunluk fonksiyonuna bağlıdır. Her bir jenerasyonda, çaprazlama ve mutasyon operatörüne başvurarak yeni jenerasyon oluşturulmaktadır. Bu süreç durdurma şartları sağlanıncaya kadar devam eder. GA, tez çalışmasında kullanılan çözüm yöntemi olmasından dolayı, bölüm 3'te daha ayrıntılı bir şekilde anlatılacaktır.

Uyarlanabilen (adaptif) bellek prosedürleri, arama sırasında ziyaret edilen müşterilerin en iyi çözümlerinden alınan rotalar kümesini ifade eder. Bellek rastgele yerleştirilen rotalar ile oluşturulur. Her iterasyonda uyarlanabilen bellekteki rastgele seçilen rotalardan çözümlere ulaşılır. Bulunan rotalar hafıza dolu olmaması ve depolanan çözümün daha iyi olması şartıyla bellekte depolanır. Durdurma kriteri sağlanana kadar bu sürece devam edilir. (Toth & Vigo, 2002)

Uyarlanabilen bellek çözüm havuz içerisindeki en iyi nitelikli elemanları birleştirerek geçici bir çözüm geliştirir. Havuz dinamik olarak yerel arama sezgiselleri ile güncellenir. Havuzun amacı yeni arama adımları için iyi bir başlangıç çözümü sağlamaktır. Uyarlanabilen bellek iki fazda etkinliğini gerçekleştirmektedir. Başlangıç fazında yerel arama yöntemiyle elde edilen çözümler birleştirilirken, işleme fazında adaptif hafızadan bilgi seçimi yaparak yeni çözüm kümesi ile hafıza güncellenir. (Repoussis & Tarantilis, 2010)

Parçacık sürü optimizasyonu (PSO), zaman pencereli araç rotalama problemlerinde kullanılan popülasyon temelli ve sürü zekası esasına dayalı bir optimizasyon yöntemidir. Kuş ve balık gibi birlikte birbirleri veya çevreleri ile sosyal davranışlar

sergileyen hayvan sürülerinden esinlenerek geliştirilmiştir. PSO'da parçacık olarak isimlendirilen bireyler muhtemel çözüme ulaşmak için mevcut en iyi çözüm popülasyonunu takip ederek, araştırma uzayında akış içerisindedirler. PSO rastgele üretilen çözümler kümesi ile aramaya başlatılır ve parçacık olarak ifade edilen her bir birey bir çözüm sunar. Her bir parçacığın bir hızı vardır. Parçacığın konumu, mevcut pozisyonuna parçacığın hızı ilave edilerek değiştirilir. Bölüm 3'te daha ayrıntılı bir şekilde anlatılacaktır.

#### *2.2.5.2. Yerel Arama Meta-Sezgisel Yöntemler*

Yerel arama yöntemleri tek nokta algoritması olarak da kullanılan çözüm sayısına göre sınıflandırılan bir meta sezgisel metodudur. Yerel arama yöntemlerinde her iterasyonda yeni bir çözüm ele alınır, uygun komşu çözümler incelenir, arama uzayında belirli bir noktada tarama yapılır. Sistemli şekilde arama yapılan bir uzayda çözüm üzerinde değişiklikler yapar. Durdurma kriteri sağlanıncaya kadar arama uzayı komşu çözümlere hareket edilerek araştırılır. Yerel arama meta-sezgisel yöntemleri yinelemeli arama prosedürlerini de içinde barındırır. Komşuluk yapısı çözüm hızı ve kalitesi için belirleyici faktördür.

Yerel arama yöntemiyle arama süreci başlangıçta verilen komşuluk ile başlar ve yerel minimuma erişildikçe sıradaki komşuluğa hareket eder. Bu süreç adım adım devam eder. Yerel arama algoritmasıyla yerel optimuma ulaşılır. Her yeni adımda yerel optimum bozulması meydana gelir, optimizasyon problemi çözülerek yeni çözüm oluşturulur. Yeni çözümün olurlu olup olmadığına kabul kriteri karar verir. (Vansteenwegen & Mateo, 2014)

Yerel arama meta-sezgisel yöntemlerin genel özellikleri aşağıdaki gibidir (El-Sherbeny, 2010):

- Başlangıçta uygun bir çözüm ile başlanır.
- Adım adım ek çözümler üretilir.
- Aday çözümler yaklaşık veya doğru olarak değerlendirir.
- Elde edilen en iyi çözümün kaydını hafızada tutabilirler.

ZPARP'de en çok kullanılan yerel arama meta-sezgisel yöntemler tabu arama,tavlama benzetimi algoritması ve değişken komşuluk dır.

Tabu arama algoritması (TA), insan hafızasının çalışma mekanizmasından esinlenilen yerel arama sezgiselidir. Arama sırasında yerel optimum noktalara takılmaktan kurtulmak için daha önce arama sırasında elde edilen bilgileri kullanır. Bu bilgiler tabu olarak ifade edilen listeye eklenir, yeni çözümler bu listeye eklenirken, eski çözümler silinir. Bu yüzden kısa dönemli bellek kullanılır (short-term memory). Tabu arama algoritmasının adımları aşağıdaki gibidir:

- Adım 1: Rastgele başlangıç çözümü seçilerek, parametre değerleri atanır.
- Adım 2: Başlangıç çözümüne göre komşu çözümler değerlendirilir, çözümler arasından en iyi olanı seçilir, seçilen bu çözüm tabu arama listesinde yoksa elde edilen en iyi çözümle karşılaştırılır ve arama listesine eklenir.
- Adım 3: Mevcut çözüm en iyi çözümle yer değiştirir ve tabu listesinde ekleme-çıkarma işlemi yapılarak güncellenir.
- Adım 4: Önceden belirtilen durdurma kriteri veya en iyi çözüme göre adım 2 ve adım 3'e devam edilir.

ZPARP için iyi sonuçlar vermesinin yanı sıra uygun sürede çözümler vermektedir.

Tavlama benzetimi (TB), fiziksel olarak katı bir metalin ısıtılıp daha sonra yavaş yavaş soğutulması sürecinden esinlenilerek oluşturulan yerel bir arama meta-sezgiselidir. Tavlama benzetimi mevcut çözümün komşuluğunda arama yapar. En iyi çözümü aramak yerine komşuluktan rastgele bir çözüm elde eder. Komşuluktan elde edilen çözümün daha iyi bir çözüm olup olmadığına bakılır, daha iyi bir çözüme sahipse mevcut çözüm olarak kabul edilir, aksi durumda, yalnızca belirli bir olasılık değerine (kademeli olarak azalan sıcaklık değeri) göre çözüm kabul edilebilir. Tavlama benzetimi algoritması adımları aşağıdaki gibidir:

- Adım 1: Rassal olarak veya önceden seçilmiş bir başlangıç çözümü $S_0$ kabul edilir.
- Adım 2: Başlangıç çözümü $S_0$ en uygun çözüm $S^*$ olarak atanır.
- Adım 3: Başlangıç çözümünün fonksiyonu hesaplanır.
- Adım 4: Başlangıç sıcaklığı $T_0$ mümkün olduğunca en yüksek sıcaklık belirlenir.

- Adım 5: Sıcaklık düşürme fonksiyonu belirlenir. $T = a * T_0$ (Genellikle geometrik fonksiyon belirlenir. $a$ soğutma parametresidir. En iyi kristalleşme değerine ulaşmak için [0 1] Aralığında 1'e yakın bir değer olması istenir. )
- Adım 6: Sıcaklık en düşük değere ulaşana kadar devam edilir.
- Adım 7: Yeni bir komşu çözüm üretilir.
- Adım 8: Bir önceki komşu çözüm $S_0$ ile mevcut çözümün maliyet fonksiyonlarının farkı alınır. Eğer $\Delta = C(S_1) - C(S_0) < 0$ ise yeni çözüm olarak $S_1$ seçilir, değilse [0 1] Aralığında r rassal sayı üretilir.
- Adım 9: Eğer $r < (\frac{1}{e^{\frac{|\Delta C|}{t}}})$ ise yeni çözüm $S_1$ seçilir.
- Adım 10: En iyi $S^*$ değeri oluşturulur, durdurma şartı sağlanıncaya kadar aramaya devam edilir.

Değişken komşuluk arama algoritması (Variable neighborhood search), son zamanlarda ZPARP'ın çözümünde kullanılan kısa sürede iyiye yakın çözümler sunan yerel arama meta-sezgiselidir. Yerel arama fonksiyonu kullanılmasına ve aramada komşuluk değişimlerine düzenli olarak izin vermektedir. Kullanıcıya diğer yerel arama meta-sezgisellerine göre daha fazla kolaylık sağlamaktadır. Değişken komşuluk arama meta-sezgiseli başlangıç çözümü ile başlar, silkeleme (shaking) sürecinde komşuluktan rastgele müşteri seçer, yeni çözüm geliştirir. Çözüme iteratif iyileştirmeler yaparak yakınsar. Komşuluktan elde edilen çözümün daha iyi bir çözüm olup olmadığına bakar, daha iyi bir çözüme sahipse mevcut çözüm olarak kabul edilir, aksi durumda, diğer komşulukta arama yapmaya devam eder. Değişken komşulukta birden fazla komşuluk yapısı kullanılarak, yerel arama ile çözüm çeşitliliği sağlanır. (Aydoğmuş, 2011)

Değişken komşuluk arama algoritması adımları aşağıdaki gibidir:

- Adım 1: Rassal veya diğer sezgisel yöntemler ile mevcut ve kararlı müşterilerden oluşan başlangıç çözümü oluşturulur.
- Adım 2: Komşuluk yapısı belirlenir.
- Adım 3: Silkeleme sürecine göre rassal olarak belirlenen çözüm içinden t. Komşuluktan amaç fonksiyonuna göre yeni bir çözüm noktası (s') oluşturulur.

- Adım 4: Oluşturulan yeni başlangıç çözüm noktasına yerel arama metodu uygulanır. Yeni en iyi çözüm (s'') belirlenir.
- Adım 5: Hareket veya dur sürecine göre yeni elde edilen en iyi çözüm ile bir önceki iterasyondan elde edilen çözüm karşılaştırılır, Eğer s'' çözümü s' çözümünden daha iyi ise bu komşulukta daha iyi çözüm aramaya devam edilir. Eğer s'' çözümü s' çözümünden daha kötü ise bu komşuluktan çıkılarak başka komşulukta çözüm aramaya devam edilir. Böylece çeşitlilik uygulanır.
- Adım 6: Durdurma kriteri sağlanıncaya kadar arama işlemine devam edilir. (Armas & Melián-Batista, 2015)

#### *2.2.5.3. Öğrenme Mekanizmalı Meta-Sezgisel Yöntemler*

Bu yaklaşım önceki iterasyondan elde edilen deneyimlerden yarar sağlayan bütün algoritmaları içermektedir. Öğrenme mekanizması öğrenme süreci boyunca olasılıklı nokta atamaları sağlar. ZPARP'de kullanılan en yaygın türü karınca kolonisi algoritmasıdır (KKO).

Karınca kolonisi algoritması, gerçek karıncaların yiyecek arama sürecini taklit eden öğrenme mekanizmalı ve popülasyon tabanlı meta-sezgisel algoritmadır. Gerçek karıncalar yiyecek ararken birbirleri ile feromon adı verilen kimyasal aromatik maddeyle iletişim kurmaktadır. Görme yetileri zayıf olduğundan karıncalar hareket ettikleri zaman bulundukları yere belirli miktarda feromon bırakırlar. Ardından gelecek olan karıncaların yönünü bulmaları için bu maddenin miktarı önemli bir yer tutar. Karıncaların feromon maddesinin daha yoğun olduğu yönü seçme olasılığı daha yüksektir.

KKO, ZPARP çözümü için uygulamada bazı zayıf noktaları bulunmaktadır:

- Araştırma yerel optimum noktalara takılır.
- Optimal çözüme ulaşmak için çok fazla hesaplama zamana ihtiyaç duyulur.
- İyi performans elde edebilmek için parametre ayarlamasının da iyi yapılması gerekir.

KKO'da bu zayıflıklardan kaçınmak için birçok sistem önerilmektedir. ZPARP çözümü için melez KKO uygulamaları kullanılır. (Ding, ve diğ., 2012)

KKO'nun zaman pencereli araç rotalama problemi için uygulama adımları aşağıdaki gibidir:

- Adım 1: Bütün kontrol parametreleri belirlenir. M adet (büyük bir sayı) karınca rastgele seçilen şehirlerde serbest bırakılır. Başlangıç optimal çözüm atanır. Aday liste oluşturulur.
- Adım 2: Her karınca için bir sonraki talep noktası denklemdeki seçilme olasılığına göre seçilir. Bütün ziyaretler her bir karınca için tabu listesine kaydedilir.
- Adım 3: Her karınca için optimal çözüm hesaplanır. Rota yerel arama tablosuna kaydedilir.
- Adım 4: Eğer yeni rota araç kapasitesi ve zaman aralığı kısıdını karşılamıyorsa rota tablosuna kaydedilen yerel en iyi çözüm listeden çıkarılır. Rota tablosu güncellenir.
- Adım 5: Yerel feromon güncellemesi yapılır. Çözüm tekrar edilir.
- Adım 6: Yerel çözüm ile global çözüm karşılaştırılır ve rota tablosu güncellenir.
- Adım 7: Bütün karıncalar rotayı oluşturduktan sonra en çok feromon içeren noktalardan rotadaki en iyi sonuç elde edilir.
- Adım 8: Durdurma kriteri sağlanıncaya kadar algoritmaya devam edilir, aksi takdirde adım 2'den devam edilir.

Bölüm 3'te detaylı bir şekilde anlatılacaktır.

# 3. MALZEME VE YÖNTEM

## 3.1. GENETİK ALGORİTMALAR

### 3.1.1. Genel Bilgiler

Genetik Algoritmalar (GA), popülasyon tabanlı arama meta-sezgiseli olup, biyolojik bir süreç olarak Darwin'in Evrim kuramından esinlenilerek ortaya çıkarılmıştır. Popülasyonda yer alan bireyler rakiplerine göre daha iyi uyum sağladıkları sürece hayatta kalır ve üremeye devam ederler. Hayatta kalma şansını bulan bireyler genlerinde kodlanmış özellikleri gelecek kuşağa aktarılır, böylece iyi özellikteki bireyler daha fazla temsil özelliğini gösterirler. Pratik mühendislik optimizasyon problemlerinin çözümünde kullanılan popülasyon fikri 1950'li ve 1960'lı yıllarda ortaya atılmıştır. Genetik algoritmalar ilk olarak John Holland tarafından 1975 yılında "Doğal ve Yapay sistemlerin uyumu" adlı kitapta yer almaktadır. GA bir türün kendine özgü genetik davranışlarını taklit etmeye çalışmaktadır. GA ve diğer sezgisel arama (Tavlama benzetimi ve Tabu Arama vb.) arasındaki en belirgin fark, tek bir çözüm yerine popülasyon çözümü sağlamasıdır (Ghoseiri & Ghannadpour, 2010).

Genetik algoritmalar yapay bağışıklık sistemi (Artificial Immune System) büyük ve karmaşık tümleşik optimizasyon problemlerinde kullanılan sezgisel metottur. Stokastik arama yöntemi olan genetik algoritmalar kabul edilebilir hesaplama zamanı içerisinde başarılı sonuçlar üretmektedir. Genetik algoritmalar kararlı ve analitik durumlarda çözülemeyen problemler üzerine çalışmaktadır. Genetik Algoritmaların ana kullanım amacı, birleştirme, mutasyon ve değerlendirme aşamasındaki probleme sürekli değişik çözümler üretmeye devam etmektir. Böylece orijinal problem istenilen duruma çok çabuk yaklaşarak çözülmektedir (Straßburg, ve diğ., 2012).

Genetik algoritmalar başlangıç popülasyonu olarak ifade edilen kromozom setleri ile çözüm bulmaya başlar. Başlangıç popülasyonu bazı zaman pencereli araç rotalama problemlerinde rastgele elde edilirken, bazılarında ise sezgisel yöntemler kullanılarak oluşturulur. Çözüm uzayının geniş, süreksiz ve kesin çözüm yöntemleri ile

çözülemeyecek kadar karmaşık problemlerde optimale yakın çözümler sunar. Müşteri noktaları kümesinde arama yapar.

Genetik algoritmalar, problemi incelemek için rasgele başlangıç çözüm kümesi ile başlar. Bu çözüm kümesi populasyon olarak adlandırılır. Bu populasyonda bulunan her bir bireye "kromozom" olarak ifade edilir. Populasyon kromozomları önceden tanımlanan uygunluk fonksiyonuna göre değerlendirilmektedir. Kromozomlar "jenerasyon (generation)" olarak ardışık iterasyonlardan geçerek gelişirler. Her bir jenerasyonda, verilen populasyonun değiştirilmiş ve birleştirilmiş olarak yeni populasyon oluşturulur. Birleşmiş kromozomlar "çaprazlama (cross over)" olarak ifade edilirken, var olanı değiştirme "mutasyon" olarak bilinmektedir. Çaprazlama yeni çift kromozom üretmede kullanılan rasgele değerlerle birleşmiş bir süreçtir. Mutasyon operatörü gen değerlerinde beklenmeyen değişiklikleri ortaya çıkarmada kullanılmaktadır. Çaprazlamaya ve mutasyona kromozom seçimi uygunluk fonksiyonuna bağlıdır. Her bir jenerasyonda, çaprazlama ve mutasyon operatörüne başvurarak yeni jenerasyon oluşturulmaktadır. Bu süreç durdurma şartları sağlanıncaya kadar devam eder. Genetik Algoritmalar 4 adımda incelenmektedir. Genel işleyişi aşağıdaki gibidir:

- Adım 1: Başlangıç popülasyonu rastsal olarak üretilir ve en uygun parametrelerin ataması yapılır.
- Adım 2: Popülasyon içerisindeki her bir kromozom için amaç fonksiyonu (uygunluk değeri) hesaplanır.
- Adım 3: Uygunluk değeri değerlendirmesi yapılabilmesi için, hesaplanan uygunluk değeri ile şimdiki popülasyondan eşleme havuzu oluşturulur.
- Adım 4: Durdurma kriter kontrolü yapılır, durdurma kriteri karşılanmıyorsa Adım 5'e devam edilir.
- Adım 5: Yeniden üretim, çaprazlama ve mutasyon operatörleri uygulanır.
- Adım 6: Çaprazlama oranına göre eşleme havuzundan ebeveynler ve çaprazlama operatörü kullanılarak rastsal olarak çaprazlama için bireyler seçilir. Mutasyon operatörleri ile mutasyon için eşleme havuzundan bireyler seçilir.
- Adım 7: Oluşturulan her kromozom için uygunluk değeri hesaplanır.
- Adım 8: Uygunluk değerlerinde kötü olan kromozomlar popülasyondan çıkarılır.

- Adım 3 ve adım 8 arası işlemler tekrarlanır (Karaboğa, 2014).

Zaman pencereli araç rotalama problemine genetik algoritmaların ilk uygulaması Thangiah, (1995) tarafından uygulanmıştır. Önce kümele-sonra rotala metodu ile araç rotalama problemleri optimize edilmiştir.

### 3.1.2. Genetik Algoritmalar Temel Kavramlar

Evrim teorisinden esinlenmesinden dolayı kullanıldığı kavramlar genetik algoritmaların da temelini oluşturmaktadır. Genetik biliminde kullanılan bazı kavramlar algoritmayı modellemekte kullanılmaktadır. Başarılı çözüm elde edebilmek için algoritma yapısında kullanılan kavramların iyi tanımlanması gerekir. Aynı zamanda kullanılan kavramlara değer ataması yapılmalıdır.

#### *3.1.2.1. Gen*

Biyoloji bilimine göre gen, bireyde kalıtsal görevleri yerine getiren, bireyin karakter özelliklerini belirlemede rol oynayan en küçük birimdir. GA'da kullanılan gen bu tanımla benzerlik göstermektedir. Kromozom içerisindeki kendi başına genetik bilgi taşıyan en küçük yapı birimidir. Programcının tanımlamasına bağlı olarak, içerdiği bilgi ikili, onluk veya on altılık sayı değerleri içermektedir. Bazı durumlarda programcı A ve B gibi alfa numerik karakterler kullanmaktadır. Örneğin tesisin x ve y koordinatları gen olarak 00001 ve 10001 bit olarak gösterilir.

#### *3.1.2.2. Kromozom*

Genetik biliminde genlerin bir amacı gerçekleştirmek için bir araya gelerek oluşturulan diziye kromozom denir. Problemin çözümüne ait tüm bilgiyi içerisinde bulundururlar. Popülasyondaki her bir birey kromozom olarak adlandırılır. Kromozomlar üzerinde durulan problemin olası çözüm bilgilerini içerisinde barındırmaktadır. Kromozomlar en önemli birim olduğu için iyi ifade edilmeleri gerekmektedir. Fazla sayıda kromozom çözüm süresinin uzamasına neden olduğu gibi, az sayıdaki kromozom çözüme yaklaşmayı zorlaştırmaktadır.

#### *3.1.2.3. Popülasyon*

Kromozomların oluşturduğu topluluk popülasyon olarak adlandırılmakta ve GA ile çözülen problemlerde geçerli alternatif çözüm kümesi olarak ifade edilmektedir. Bu

nedenle çözüm bilgilerini içermektedir. Popülasyondaki kromozom sayısı sabit tutulup, problemin özelliğine göre programcı tarafından problem çözülmeye başlamadan öncede belirlenir. GA'nın işlem görmesi sırasında popülasyonda bazı kromozomlar yok olurken, yerlerine yeni kromozom yapıları eklenerek popülasyon büyüklüğü sabitlenir. Popülasyon büyüklüğü de kromozom uzunluğu gibi iyi ifade edilmelidir. Çözüm sürecini etkilediğinden, fazla sayıdaki kromozom popülasyonu problemin çözüm süresini uzatırken, az sayıda popülasyon çözüm değerlerine ulaşılmamasına neden olur.

#### *3.1.2.4. Uygunluk Fonksiyonu*

Genetik algoritmaların kullanıldığı problemlerde başarılı bir şekilde çözümler elde edildiğini göstermek için uygunluk fonksiyonu kullanılır. Hangi kromozomun bir sonraki nesle taşınacağı ve hangi kromozomların yok olacağı uygunluk değerlerinin büyüklüğüne göre karar verilir. Problemden probleme değişen değerlendirme kriteridir. Genel işleyişi başlangıç popülasyonu oluşturulduktan sonra kromozomlar hesaplanır ve bu kromozomun problem çözümünde iyi bir cevap olup olmadığının kararı uygunluk fonksiyonu ile bulunur. Diğer bir ifadeyle hangi kromozomun "en iyi" olduğuna karar verebilmek için uygunluk fonksiyonu kullanılır. Böylece yeni nesilde kullanılacak kromozomlar bulunur.

Uygunluk fonksiyonu problem için en optimal çözümü belirleme kriteri olmasından dolayı probleme göre kâr veya verimliliği maksimum yapacak, maliyet veya kaybı minimum yapacak değişkenlerin ölçülmesini sağlayacak bir fonksiyon olarak belirlenmelidir. Problem çözümünde genetik algoritmaların kullanılması durumunda, probleme uygun çözüm kümesinin bulunması için, başlangıçta uygunluk fonksiyonunun belirlenmesi gerekmektedir (Bakırlı, ve diğ., 2011).

### 3.1.3. Genetik Algoritmalar ve Metodolojisi

Genetik algoritmalar evrim teorilerinden etkilenerek kendine özgü metodolojileri içinde barındırmaktadır.

#### *3.1.3.1. Genetik Kodlama*

Genetik algoritmalarda popülasyondaki kromozomların nasıl temsil edileceğine karar verilmesinde kodlama önemli rol oynamaktadır. Böylece sistem hakkındaki bütün bilgilere ulaşılması sağlanır. Kromozomlara dayanan çözümlerin kodlanması genetik

algoritmalarda önemli bir sorundur. Bunun için çeşitli kodlama sistemleri oluşturulmuştur (Guillaume, ve diğ., 2010).

İkili (binary) kodlama, en çok kullanılan kodlama sistemidir. Genetik algoritmaların ilk uygulayıcılarının bu sistemi kullanmasından ve kolay olmasından dolayı en yaygın kullanılan kodlamadır. Her kromozom 0 ve 1 karakterleri ile temsil edilen bitlerden oluşan diziyle kodlanır.


**Şekil 3.1:** İkili Kodlama Gösterimi.

Şekil 3.1'de ikili kodlama örneği verilmektedir. Kromozom I, 84 değerinin ikili kodlama sistemindeki gösterimidir. Kromozom II, 137 değerinin ikili kodlama sistemindeki gösterimidir.

Gri (Gray) kodlama, ikili kodlama yapısı ile benzerlik göstermektedir. 0-1 kodlama sisteminden farkı, çözüm uzayında ardışık gelen iki noktanın kodlandığında birbirinden tamamen farklı bireylerin ortaya çıkmasıdır. Gri kodlamada Hamilton yolu kullanılır (Reeves & Rowe, 2003).

**Tablo 3.1:** İkili ve Gri Kodlama.

| Tamsayı | İkili Kodlama | Gri Kodlama |
|---|---|---|
| **0** | 000 | 000 |
| **1** | 001 | 001 |
| **2** | 010 | 011 |
| **3** | 011 | 010 |
| **4** | 100 | 110 |
| **5** | 101 | 111 |
| **6** | 110 | 101 |
| **7** | 111 | 100 |

Tablo 3.1'de ikili ve gri kodlama arasındaki ilişki gösterilmektedir.

Permütasyon kodlama, 1,2,..., n sıralamasına sahip n adet sayı dizisi için kromozomların gösterilmesinde kullanılmaktadır. Zaman pencereli araç rotalama için uygun kodlama

yapısına sahiptir. Her tam sayı değeri doğrudan müşteri talep noktasını gösterir. 7 müşteri noktası ve bir depodan oluşan kodlama V=[2 4 5 1 0 3 6 7] ile gösterilir. Depodan hareket eden araç önce 2 sonra sırasıyla 4, 5, 1 müşteri noktalarına hareket ederken, depodaki bulunan başka bir araç ise önce 3 daha sonra sırasıyla 6 ve 7 numaralı talep noktalarına hareket etmektedir (Reeves & Rowe, 2003).

Rassal sayılı kodlama, [0 1] Aralığında rassal sayılar kullanılır. Bu kodlamaların dışında günlük hayat problemlerinin çözümünde kullanılan değer kodlama ve ağaç kodlama yöntemleri bulunmaktadır.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **KROMOZOM I** | 2 | 3 | 1 | 6 | 4 | 5 | 7 | 10 | 9 | 8 |
| **KROMOZOM II** | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 |

**Şekil 3.2:** ARP Kodlama.

Şekil 3.2'de 10 müşterili 3 araçlı araç rotalama problemi kodlaması bulunmaktadır. Buna göre Kromozom II ile gösterilen 1 nolu araç sırasıyla 2, 3, 1 ve 6 nolu talep noktalarını ziyaret ederken, 2 nolu araç sırayla 4, 5, 7; 3 nolu araç ise 10, 9 ve 8 nolu talep noktalarına hizmet vermektedir.

#### *3.1.3.2. Başlangıç Popülasyonu Oluşturma*

Genetik algoritmalar yerel arama meta-sezgisellere göre önemli farkı çözümü tek bir noktadan değil, noktaları oluşturduğu popülasyon içinde aramasıdır. Popülasyon boyutu GA'nın en önemli parametrelerinden birisidir. Aday kromozom şeklinde N adet olası çözüm popülasyonu oluşturarak çalışmaktadır. Her bir kromozom hedef sisteminin bir gösterimini uygulayıcıya sunmaktadır. Potansiyel çözüm popülasyonu ile problemin çözümüne başlanır. Bu çözümler kromozomlarda kodlanmaktadır. Her bir potansiyel popülasyon çözümü hem olurlu hem de tek olmalıdır.

ZPARP için oluşturulacak başlangıç popülasyonu için diğer sezgisel yöntemlerden yararlanılır. (Nguyen, ve diğ., 2014) En çok kullanılan yöntem en yakın komşu (nearest neighbour) metodudur. Bu yöntemle önce ilk araca kapasite ve zaman aralığı kısıdına göre atama yapılır. Önceden hizmet almayan en yakın talep noktasına bu kısıtlar dahilinde araç yönlendirilir. Eğer müşterinin zaman aralığı uygun değil ve talebi araç kapasitesini aşıyorsa, 1. Araç başlangıç noktası olan depoya döner ve 2. Araç

rotalamaya devam eder. Bir diğer başlangıç populasyonu oluşturmak için kullanılan yöntem ise rastsal olarak rotalamaya başlamaktır. Rastgele oluşturulan rotadaki ilk müşteri zaman ve kapasite kısıtlarına bakılarak başlangıç noktası olarak seçilmektedir. Bu yöntemle oluşturulan başlangıç popülasyonunda bütün müşterilerin ilk rotaya atandığı varsayılmakta olup, daha sonra ZPARP'ın kısıtlarını sağlayamayan müşteriler bu başlagıç rotasından çıkarılmaktadır (Wang & Chen, 2012).

#### *3.1.3.3. Uygunluk Değeri Hesaplama*

Başlangıç popülasyonu oluşturulduktan sonra uygunluk değerinin hesaplanması gerekmektedir. Bu değer popülasyondaki her kromozom için hesaplanır. Uygunluk değeri ne kadar optimal olursa kromozomun performansı da bir o kadar iyidir. Başarılı performans göstermesi bu değerin verimliliğinden kaynaklanır. Her bir probleme özel çalışan tek bölüm bu değerdir.

Araç kapasite kısıdı, müşteri hizmet aralıkları ve depo çalışma saatleri dikkate alınarak toplam mesafe minimizasyonu ZPARP için uygunluk değerini oluşturur. ZPARP için en çok kullanılan uygunluk değeri her bir müşterinin depoya olan uzaklıkları ile rotada bulunan diğer müşterilere olan uzaklık değeri toplamı şeklinde hesaplanmaktadır (Wang & Chen, 2012).

#### *3.1.3.4. Yeniden Üretim*

Bu aşama uygunluk değerlerinin orantısal olarak olasılıklarına göre ebeveynlerin çoğalması sürecini oluşturur. Sonuçta, yüksek uygunluk değerlerine sahip ebeveynler bir sonraki nesilde yavrularını üretmede daha yüksek olasılıklara sahip olacaklardır.

Yeniden üretme süreci mevcut kromozom dizisinden gelecek kromozom dizisine aktarılacak dizilerin seçilme işlemini ifade etmektedir. Bu işlem sayesinde belirlenen uygunluk değerine sahip daha iyi bireylerin bir sonraki nesle aktarılmasına yardımcı olmaktadır. Yeniden üretme işlemi kodlama ve popülasyon oluşturulduktan sonra genetik algoritma metodolojisinde kullanılan bir yöntemdir. Popülasyondan rastgele oluşturulan bir grup dizi seçilir ve grup içindeki en iyi uygun değere sahip dizi, yeni popülasyona kopyalanmaktadır (Bakırlı, ve diğ., 2011).

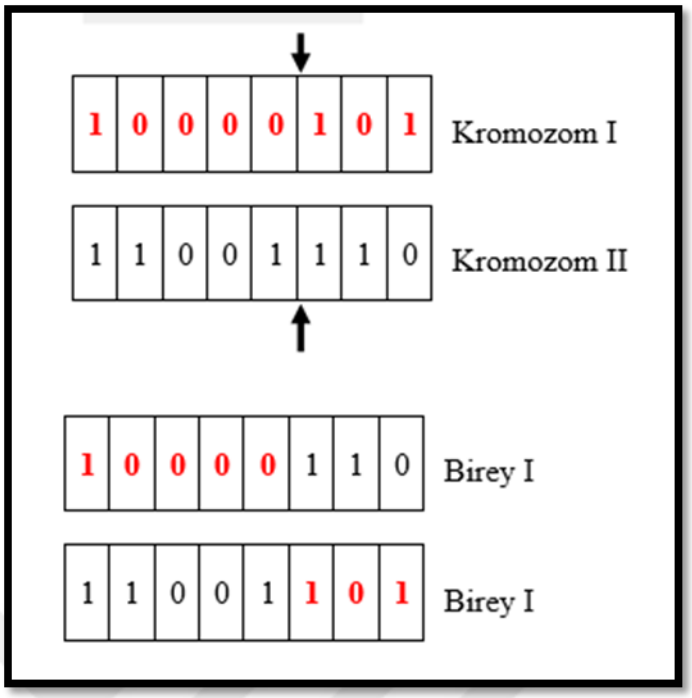
Turnuva yöntemi en sık kullanılan yöntemlerden bir tanesidir. Yığından rastgele oluşturulan bir grup dizi seçilir ve grup içindeki en iyi uygun değere sahip dizi, yeni yığına kopyalanmaktadır.

#### *3.1.3.5. Çaprazlama*

Çaprazlama, 2 ebeveyn kromozomundan gelen 2 yavru kromozomun oluşmasına yardımcı olmaktadır. Her bir yavru kromozom geni iki ebeveynin farklı özelliklerini içinde barındırmaktadır. Çaprazlama genetik algoritmalarda yeni jenerasyonun içeriğini oluşturan yeni iki kromozom üretmek için seçilmiş iki kromozomun birleştirilmesini sağlayan operatörler geliştirmektedir. Her bir bireyden en iyi özellikleri almayı ve yavrularda kalan özellikleri değiştirmek için bu metodoloji kullanılmaktadır. Çaprazlama en iyi birey oluşturmak için genetik algoritmalara yardımcı olmaktadır. Ebeveynlerin çaprazlama operatörü olasılığı çaprazlama oranı ile karar verilir (Cheng, ve diğ., 2013).

Çaprazlama popülasyonu adım adım çözüme ulaştıracaktır. Çaprazlamanın genetik algoritmalarda görevini yerine getirmek için kodlama ve problem çeşidine göre değişik uygulama yöntemleri bulunmaktadır.

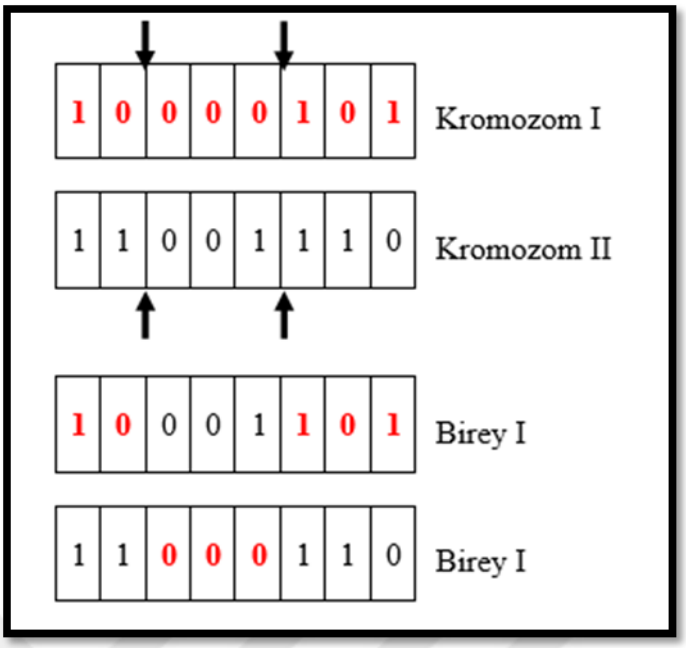
Tek noktalı çaprazlama, Bu yöntem her bir ebeveynin 1 ile rastgele üretilen sabit bir sayıya göre iki parçaya ayırmak için kullanılmaktadır. İlk ebeveynden gelen ilk parça ile ikinci ebeveynden gelen ikinci parça yeni birey oluşturmak için birleşir. İlk ebeveynden gelen ikinci parça ile ikinci ebeveynin ilk parçası ikinci yeni bireyi oluşturmak için birleştirilir.




**Şekil 3.3:** Tek Noktalı Çaprazlama.

Şekil 3.3'te tek noktalı çaprazlama örneği bulunmaktadır.

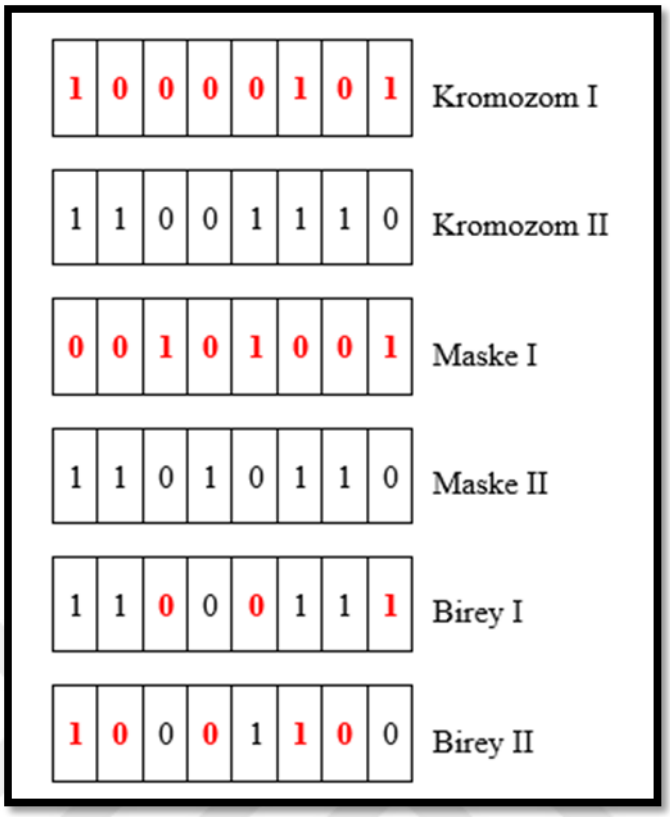
Çift noktalı çaprazlama, iki sayıya göre her bir ebeveyni üç parçaya ayırmaktadır. İlk ve üçüncü parça ilk ebeveynden, ikinci parça ikinci ebeveynden gelerek yeni birey oluşturulmaktadır. İlk ebeveynden ikinci parça ile ikinci ebeveynden ilk ve üçüncü parçalar birleşerek ikinci yeni bireyi oluşturmaktadır.




**Şekil 3.4:** Çift Noktalı Çaprazlama.

Şekil 3.4'te çift noktalı çaprazlama örneği bulunmaktadır.

Düzgün çaprazlama metodunda, dizi maskesi öncelikle tanımlanmaktadır. Bu çaprazlama maskesi hangi bitlerin ilk ebeveynden ve diğer ebeveynden kopyalanacağına karar vermek için rastgele olarak üretilmektedir. İlk bireyi oluşturmak için ilk ebeveynden "1" ile belirtilen maske bitler kopyalanır. "0" ikinci ebeveynden kopyalanacaktır. (Bakırlı, Yani birinci yeni birey için üretilen maske, ikinci çocuk için tam tersi üretilerek kullanılmaktadır.




**Şekil 3.5**: Düzgün Çaprazlama

Şekil 3.5'te düzgün çaprazlama örneği bulunmaktadır.

ZPARP için geliştirilen bazı çaprazlama algoritmalarında herhangi bir çaprazlama operatörü kullanılmaz. Ebeveynden gelecek iyi özellikler uygunluk değerine göre bir sonraki kuşağa aktarılmaktadır. Böylece ebeveynler bir sonraki kuşağa çok fazla özellik bırakmaktadır. Öncelikle rastgele oluşturulan rotalar seçildikten sonra, yavru rotalar ve diğer tüm rotalanmamış müşteriler bu rotalara eklenmekte veya yeni bir rotaya geçilmektedir. Tablo 3.2'de bu çaprazlama türünün algoritması gösterilmektedir (Wang & Chen, 2012).

**Tablo 3.2**: Çaprazlama Algoritması.(Wang & Chen)

| |
|---|
| **Fonksiyon** çaprazlama |
| **Başla** |
| **Tekrarla** |
| Rastgele Rotadan 1. Ebeveyni çocuğa aktar |
| Rastgele Rotadan 2. Ebeveyni çocuğa aktar |
| **Devam et** (bütün çözümler uygun olana kadar) |
| Rotalara dahil edilmemiş müşterileri 1*1 matrisli rotaya ekle |
| Rotalanan müşterilerin sayısını azalt |
| **Bitir** |

Diğer çaprazlama yöntemleri çok noktalı çaprazlama, ölçekli çaprazlama olarak ifade edilmektedir.

#### *3.1.3.6. Mutasyon*

Çaprazlama ile birlikte genetik algoritmalarda kullanılan önemli operatörleri içinde bulundurmaktadır. GA'da sistem belli döngü değerine ulaştıktan sonra kromozomlar birbirlerine benzemeye başlamaktadır. Mutasyonun kullanım amacı popülasyonu oluşturan bireylerde çeşitliliği artırmaktır. Mutasyon genlerde küçük bir değişiklik yaparak yeni kromozom üretir. Mutasyon, genetik değişimi korumak ve yerel optimumdan sakınmak için belli bir olasılık ile rasgele bit deformasyonunu sağlayan operatördür. Mutasyon işlemi pm olasılık değeri ile kromozomda bulunan genleri geliştirmektedir. Mutasyon olasılığı genellikle çok küçüktür. (pm<0,01) Mutasyon olasılığı aynı çaprazlama olasılığı gibidir. Eğer rastgele üretilen sayı mutasyon olasılığından küçük ise, dizinin belirlenmiş bitleri değiştirilir.

Mutasyon bireylere seçim aşamasında veya çaprazlamadan sonra uygulanmaktadır. Mutasyonun diğer önemli görevi ise iyi özelliklere sahip genlerin tekrar kromozoma kazandırılmasıdır.

| Mutasyon Olasılıkları | 0,002 | 0,859 | 0,765 | 0,945 | 0,32 | 0,075 | 0,363 | 0,452 |
|---|---|---|---|---|---|---|---|---|
| Mutasyondan önce | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| Mutasyondan sonra | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

**Şekil 3.6:** İkili Kodlama İçin Mutasyon.

Şekil 3.6'da ikili kodlama için mutasyon olasılığı 0,003 seçildiğindeki kromozomdaki sadece ilk genin (0,002<0,003) değiştiği görülmektedir.

Genetik Algoritmalarda mutasyon süreci algoritması şu şekildedir:

- Adım 1: 0 ile 1 arasında rastgele bir sayı üret. (n) Eğer n<mutasyon_olasilik Adım 2'ye değilse Adım 1'e git
- Adım 2: 1 ile 3 arasında rastgele bir sayı üret. (m)
  For i=0:i<m
  0 ile kromozom_uzunlugu arasında x sayısı üret.
  x. biti "0 ise 1'e", "1 ise 0'a" dönüştür.
  Döngüden çık
- Adım 3: Mutasyon değeri döndür.

Mutasyon sonucu arama havuzunda yeni genler oluşmaktadır. Komşu değişim mutasyonunda (Adjacent Exchange Mutation) kromozomdaki iki komşu gen rastsal olarak yer değiştirmektedir.

Kaydırma (insert) mutasyonunda düzgün dağılmış kromozomdan rastsal olarak bir gen seçilmekte ve bu gen rastsal olarak araya girerek kromozom dizisinde yer almaktadır.

#### *3.1.3.7. Optimallik Ölçütü*

Basit genetik algoritmalarda, uygunluk değeri hesaplama, seçim, çaprazlama ve mutasyon operatörleri optimallik şartı sağlanıncaya kadar devam etmektedir. Optimallik ölçütü genellikle jenerasyon sayısı ve uygunluk değeri olarak belirtilmektedir. Optimallik ölçütü diğer bir ifade ile sonlandırma kriteri bu iki kriterle değerlendirilmektedir. Çoğu uygulamada önceden belirlenen jenerasyon sayısı sağlandığında algoritma sona ermektedir. Jenerasyon sayısı kullanıcı tanımlı bir

parametredir. Değerlendirmeler maksimum sayıya ulaştığında sonlandırılır. Yani kaç tane jenerasyonun oluşturulacağı kararına göre algoritma sonlandırılmaktadır. Bazı durumlarda optimum çözüme ulaşılmadan sonlandırılması genetik algoritmanın performansını etkilemektedir. Uygunluk değeri metodunda kullanıcı tanımlı uygunluk eşiğine ulaşıldığında değerlendirme sona erdirilmektedir. Yanlış uygunluk değeri sonsuz döngü oluşmasına sebep olmakta ve verilen eşik değere asla ulaşılamamaktadır. Çalışmada kullanılan Genetik Algoritmalar, popülasyon tabanlı sezgisel olup, başlangıç çözüm kümesi ile ve yardımcı amaç fonksiyonu ile aramaya başlanmaktadır.

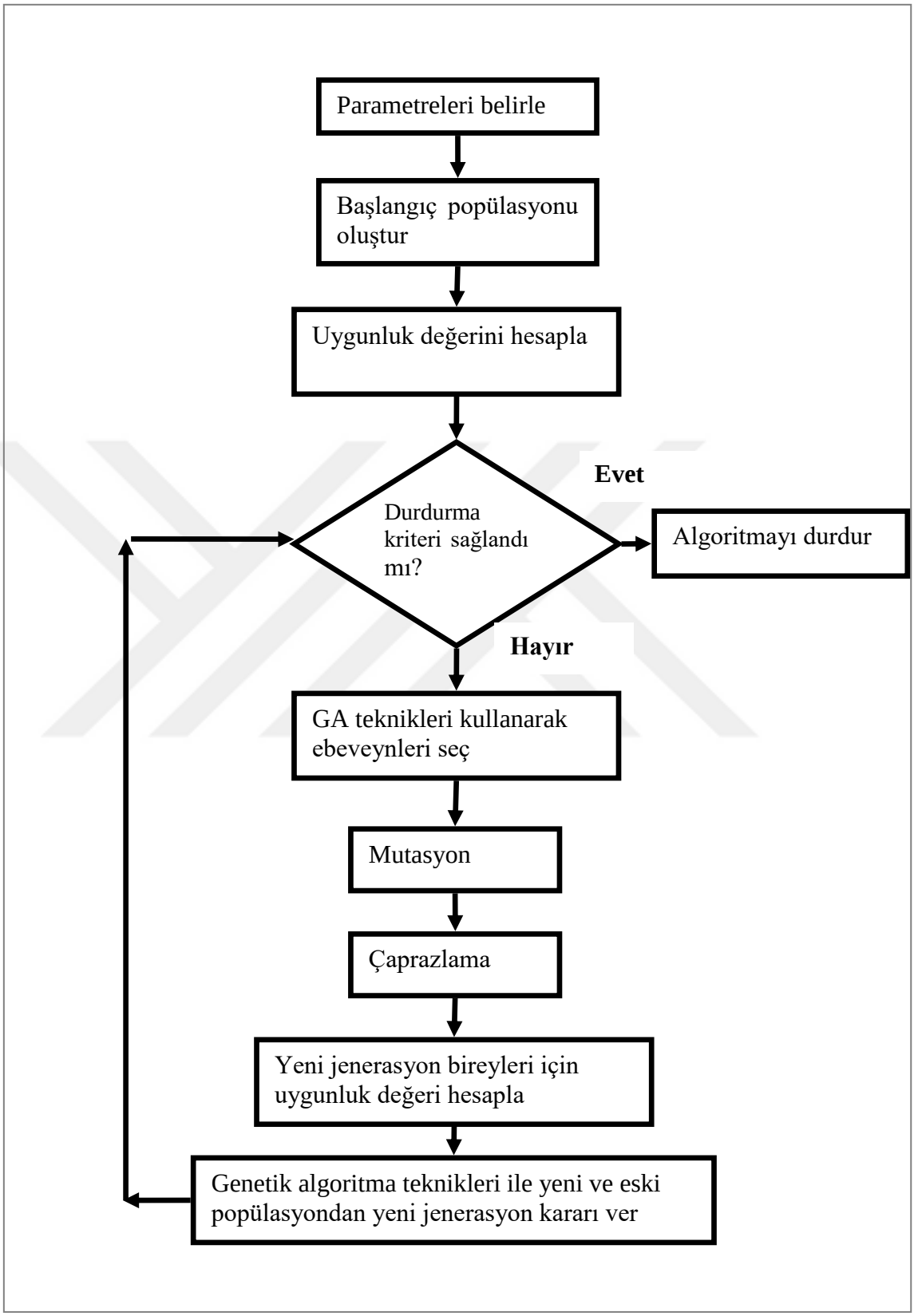
Bu çalışmada çaprazlama aşaması için Wang & Chen (2012)'in önerdiği algoritma kullanılacaktır.

- Çözüm uzayında diğer çaprazlama metotlarına göre daha hızlı arama yapmaktadır.
- Her çözüm kümesinde mutasyona uğrayacak bireyler bulunduğundan, iyi olan bireyler hayatta kalmaktadır.

Bu algoritmaya göre müşterinin sahip olduğu zaman ve kapasite kısıtına göre 2 müşteri arasında uygunluk fonksiyonuna göre rastgele çaprazlama yapılacaktır. Böylece iyi özelliklere sahip çocuk bireyler bir sonraki nesile aktarılarak, daha uygun rota oluşturulacaktır. Kullanılan bu metod Random Seeds Cheapest Insertion Method (RSCIM) olarak da ifade edilmektedir. GA'nın yerel optimum noktalara takılmasını önler. Rastgele sıradaki müşterilerin uygunluk fonksiyonuna göre yine rastgele çaprazlamaya tabi tutulmaktadır. Bu yüzden sabit bir çaprazlama operatörü kullanılmamaktadır. 1. Ebeveynden çocuğa rastgele rota kopyalanır. 2. Ebeveynden çocuğa rastgele rota kopyalanır. Zaman ve kapasite kısıtları hafızaya alınır. Uygunluk değerleri hesaplanır. Uygun değerler bulunana kadar bu döngü tekrar edilir. Rotalanmamış müşteriler için bu süreç devam eder.

Rotalamada popülasyon içi çeşitlilik için bazı bireylerin mutasyona uğrayarak hayatta kalması gerekmektedir. Her bir iterasyonda birkaç kromozom için rastgele değişimler uygulanacaktır. Rastgele bir araç veya rota seçilip, 2 rastgele seçilen müşterinin aynı rotada yer değiştirmesi yöntemi kullanılacaktır. (Wang & Chen, 2012)

Şekil 3.7'de genetik algoritmaların genel akış diyagramı verilmektedir:




**Şekil 3.7:** Genetik Algoritmalar Akış Diyagramı **(Wang & Chen, 2012)**.

### 3.1.4. Genetik Algoritmaların Üstün ve Zayıf Yönleri

#### *3.1.4.1. Üstün Yönleri*

Kesin ve klasik sezgisel yöntemlere göre genetik algoritmalar kullanıcılarına oldukça avantaj sağlamaktadır. Genetik algoritmalar araç rotalama gibi NP-zor problemlere tek bir çözüm üretmek yerine farklı çözümlerden oluşan çözüm kümesi sağlamaktadır. Bu yüzden arama yaptığı uzayda birden fazla nokta kısa süre içerisinde değerlendirilmektedir. Yüksek kaliteli çözümler sunar. Sürekli ve kesikli değişkenler ile optimizasyonu gerçekleştirir. Çok sayıda değişkenlere sahip analitik, deneysel veya sayısal fonksiyonları kolaylıkla çözümleyebilmektedir. Ek bilgiye gerek duymamaktadır. Karmaşık ve büyük çözüm uzayına sahip problemlerde optimal çözüm bulmada çok etkilidir. Geliştirilmesi düşük maliyetli olup, çok karmaşık yapıdaki fonksiyon örneklerinde arama yapmaya izin vermektedir. Çözüm uzayında aynı anda birden fazla çözüm sunduğu için potansiyel çözüm listesi ile alternatif çözüm yollarını araştırır. Diğer klasik ve meta-sezgiseller ile birlikte melez kullanıma uygundur. Uygunluk fonksiyonu için türev alınmasına gerek duymaz. Bu yüzden geniş bir alanda uygunluk fonksiyonunu araştırır. Genetik algoritmaların önemli bir özelliği parametrelerin kendileri yerine parametre kümesinin kodlanmasına izin vermesidir. Diğer avantajları aşağıdaki gibidir:

- Çözüm uzayında herhangi bilgiye ihtiyaç duymazlar.
- Yerel optimumda takılmazlar.
- Büyük ölçekli optimizasyon problemlerin çözümünde iyi sonuç vermektedir.
- Diğer bilgisayar programlama dillerine rahatlıkla uyarlanabilirler.

#### *3.1.4.2. Zayıf Yönleri*

Her ne kadar genetik algoritmalar kullanıcıya birçok avantaj sağlasa da bazı durumlarda problemlere yetersiz çözümler sunmaktadır. Her bir bireyin üretimini tanımlamak için zaman harcanması genetik algoritmaların zayıf yönlerinden bir tanesidir. Genetik algoritmalar global optimumun bulunmasını garanti etmez. Klasik sezgisel yöntemlere göre çözümde daha fazla zaman harcarlar. (Guillaume, ve diğ., 2010)

Kullanıcı tarafından tanımlanan jenerasyon sayısı ve uygunluk değeri optimum çözümü bulamadan bitirilebilmektedir. Diğer zayıf yönleri aşağıdaki gibidir:

- Çok fazla uygunluk fonksiyonu hesaplamaktadır.
- Kesin optimum noktasını bulamaz.
- Probleme özgün yapı oluşturulması nedeniyle kullanıcıdan her defasında farklı konfigürasyon ister.
- Nesiller arasındaki iyi özellikler bazı durumlarda kaybolmaktadır. (Kaya, 2012)

## 3.2. PARÇACIK SÜRÜ OPTİMİZASYONU

### 3.2.1. Genel Bilgiler

1995 yılında Eberhart ve Kennedy tarafından optimizasyon problemlerin çözümünde kullanılmak üzere tasarlanmış kuş ve balık sürülerinin sosyal davranışlarını taklit eden popülasyon tabanlı meta-sezgisel arama algoritmasıdır. Sürü zekâsı esasına göre kuş ve balık türlerinin davranışlarına göre doğal sistem hareketleri incelenmiş, yiyecek arama ve tehlike anında birbirleri ile iletişim kurduğu görülmüştür. Kuşlar yiyecek ararken yiyeceğe en yakın kuşu takip ederek sürüyü oluştururlar. Genetik algoritmalar gibi evrim teorisinden esinlenerek oluşturulmuştur. Genetik algoritmaların sahip olduğu bazı özellikleri kullanır. Fakat genetik algoritmalarda kullanılan çaprazlama ve mutasyon gibi evrimsel parametreler kullanılmaz. Rastgele oluşturulan başlangıç popülasyonu ile optimum çözüm kümesini aramaya başlar, popülasyonu değerlendirmek için uyum fonksiyonundan yararlanarak, popülasyonu günceller. (Önüt, ve diğ., 2008)

Evrimsel hesaplama parametrelerinde olduğu gibi algoritmada her bir birey parçacık olarak ifade edilir, parçacıklardan oluşan popülasyon ise sürü adını almaktadır. Parçacık olarak ifade edilen bireyler araştırma yapılan uzayda hareket halindedir, her bir parçacığın konumu kendisi ve yakınında bulunan parçacıkların tecrübelerine göre arama yapmaktadırlar. Her parçacık potansiyel çözümü taşırken, en iyi durumu hafızaya alır. Potansiyel çözümler mevcut hafızadaki en iyi çözüm kümesini takip ederek problem uzayında arama yaparlar. N parçacıklı D boyutlu arama uzayında hareket sürü parçacık sürü optimizasyonunun temelini oluşturmaktadır. (Karaboğa, 2014)

PSO'da bireylerin birbirleri ile etkileşimi sürü zekâsı olarak ifade edilir. Parçacıklar arama uzayında hareket etmesini sağlayan hız ve hız vektöründen elde edilecek konum bilgisine sahiptir. Bütün parçacıklar bireysel uygunluk değerine sahiptir. Her bir

parçacık kendine özgü hız ile daha iyi uygunluk değerine sahip olmak için arama uzayında hareket eder. Her bir parçacık uygunluk fonksiyonuna göre elde edilen değer ile kıyaslanır. Rastgele üretilen çözüm kümesiyle aramaya başlar ve parçacıkların hızı ve konumu güncellenerek en uygun çözüm değerleri araştırılır. Parçacıkların her birinde kendi en iyi değeri (pbest-kişisel en iyi) ve tüm parçacıkların en iyi (gbest-global en iyi) değerine göre değerlendirmeye alınır. Bireylerin kendi önceki deneyimlerden yararlanarak pbest yerel aramada en iyi bulma amacı ile popülasyondaki diğer bireylerin önceki deneyimlerinde yararlanarak gbest global aramada en iyi bulma amacı bulunmaktadır. Güncellemeler bu değerlerin hafızada tutulması ile yapılır.

$$v_i(t+1) = wv_i(t) + c_1 rand_1 * \left(p_{best} - p_i(t)\right) + c_2 rand_2 * \left(g_{best} - p_i(t)\right) \quad (3.1)$$

$$p_i(t+1) = p_i(t) + v_i(t+1) \quad (3.2)$$

Denklem (3.1) ile her bir parçacığın hızı, denklem (3.2) ile konumu hesaplanır. t iterasyon sayısını, v hızı, w eylemsizlik ağırlık vektörünü, $c_1$ ve $c_2$ parametreleri öğrenme katsayılarını göstermektedir. $rand_1$ ve $rand_2$ [0 1] aralığında üretilen düzgün dağılan rastgele sayılardır. Parçacığın hızı formülü 3 bölümden oluşur. $wv_i(t)$ iterasyondaki bir önceki hızın mevcut hıza atalet gücü etkisini, $c_1 rand_1 * \left(p_{best} - p_i(t)\right)$ olarak ifade edilen 2. Bölüm kavramsal gücü, $c_2 rand_2 * \left(g_{best} - p_i(t)\right)$ olarak ifade edilen 3. Bölüm sosyal düşünceyi bireyler arasındaki uyumu ortaya koyar. (Norouzi, ve diğ., 2015)

### 3.2.2. Parçacık Sürü Optimizasyonu Genel Kavramlar

PSO algoritması genetik algoritmalar gibi evrim teorisinden esinlenerek oluşturulmuş olmasına rağmen bazı parametrelere sahip olmaması nedeniyle basit yapılı olduğu ifade edilir.

#### *3.2.2.1. Parçacık*

Sürüdeki her bir birey parçacık olarak adlandırılır. Her parçacık kendine ait uygunluk değerine sahip olduğu için bir çözüm elde edilmesine yardımcı olurlar. Aynı zamanda parçacığın $p_i$ ve $v_i$ olarak ifade edilen konum ve hızı bulunur. Parçacıklar potansiyel çözümler

olarak adlandırılırlar, bu sayede mevcut en iyi çözümleri takip ederek problem uzayında hareket ederler. Parçacık sayısı problemlerin boyutuna göre önceden belirlenir.

#### *3.2.2.2. Parçacık Hızı*

Hız vektörel bir büyüklük olduğu için PSO'da kullanılan parçacık hızı pozitif ve negatif değerler almaktadır. Her bir parçacığın kendine ait hızı vardır. Bu hız vektörü belirli değer aralığındadır. Böylece parçacığın belirli çözüm uzayı dışına çıkılmasına izin verilmemektedir. Hız parametresi bu algoritmanın temelini oluşturmaktadır. Bulunduğu adım ile bir sonraki adımın hız ve konum bilgilerinin güncellenmesinde kullanılır. Arama uzayında mevcut konum ile hedeflenen konuma ulaşılmasına yardımcı olur. v sembolü ile gösterilir.

#### *3.2.2.3. Parçacık Konumu*

Her bir parçacık için çözüm uzayında bulunduğu pozisyon olarak kullanılır. $p_i$ ile ifade edilir. Hız denklemine göre elde edilir. Üst ve alt limitleri iyi tanımlanmalıdır.

#### *3.2.2.4. Popülasyon Büyüklüğü*

PSO'da popülasyon sürü olarak adlandırılmaktadır. Sürü büyüklüğünün fazla olması hızlı arama yapılmasını sağlamasının yanı sıra daha hızlı çözümler bulunmasına yardımcı olacaktır.

#### *3.2.2.5. En İyi Değerler*

Parçacığa ait en iyi değer ($p_{best}$), arama uzayında her bir parçacığın bir önceki iterasyonda en iyi değeri hafızasında tutmasıyla elde edilen yerel en iyi değerdir. Sonraki iterasyonlarda hafızasında tuttuğu yerel değerden daha iyi bir değer elde ederse, hafızasında sakladığı değeri yeni en iyi ile günceller.

Sürüye ait en iyi değer ($g_{best}$), arama uzayında her bir sürünün bir önceki iterasyonda en iyi değeri hafızasında tutması ile elde edilen global en iyi değerdir. Sonraki iterasyonlarda hafızasında tuttuğu global değerden daha iyi bir değer elde ederse, hafızasında sakladığı değeri yeni en iyi ile günceller.

#### *3.2.2.6. Atalet Ağırlığı*

Parçacığın bir mevcut hızının bir sonraki iterasyonda elde edilecek hızına ne derecede etkileyeceğini belirleyen parametredir. Çözüm uzayında parçacıkların ani hız

değişimlerini engellemek için kullanılır. Parçacıkların bir önceki hızına bağlı kalarak uygun arama yapılmasını sağlar. $w$ ile gösterilir.

#### *3.2.2.7. Öğrenme Faktörleri*

$c_1$ ve $c_2$ olarak hız denkleminde kullanılan bu parametreler öğrenme faktörü olarak adlandırılırlar. $c_1$ parçacığa güvenme, $c_2$ sürüye güvenme parametresi olarak ifade edilir. Algoritmanın performansı üzerinde önemli etkisi bulunur. Diğer bir ifadeyle, $c_1$ kavramsal parametre olarak tanımlanırken, $c_2$ sosyal parametre olarak tanımlanır. $c_1$ ve $c_2$ parametreleri [0,4] aralığında farklı bir değer alırlar.

#### *3.2.2.8. Durdurma Kriteri*

PSO'da durdurma kriterleri iterasyonun sona ermesini sağlamak için iyi seçilmelidir. Yerel arama noktalarına takılmasını engellemek için durdurma kriteri erken yakınsama olmamalıdır. Durdurma kriteri kullanıcının optimal çözüme erken ulaşmasını sağlamalıdır. Önceden belirlenen maksimum iterasyon sayısına ulaşmışsa, uygun bir çözüm bulunmuşsa, amaç fonksiyonu eğimi 0'a yaklaşıyorsa, belirli sayıda iterasyon süresince herhangi bir gelişme gösterilmiyorsa ($g_{best}$ değerinin değişmemesi) arama durdurulur. (Karaboğa, 2014)

### 3.2.3. Parçacık Sürü Optimizasyonu Metodolojisi

Parçacık sürü optimizasyonu sürekli optimizasyon problemlerinde kullanılırlar. PSO'da bulunan parçacıklar için başlangıç çözümleri rastsal olarak üretilirler. Her parçacığın d-boyutlu arama uzayında hız vektörü, mevcut konum vektörü ve önceki en iyi konum vektörü olmak üzere üç bileşeni bulunmaktadır. PSO başlangıç parametreleri belirlendikten sonra rastgele parçacık sürüsü ile çözüme başlar. Her bir iterasyonda güncelleme yaparak en iyi çözüm kümesi bulunmaya çalışır. Her iterasyonda parçacık hızı ve konumu güncellenir. Parçacık konumu o ana kadar parçacığın elde ettiği en iyi çözümü sağlayan pozisyon olan $p_{best}$ değerini hafızasında tutar. Popülasyonda bulunan parçacık sürüsünün global en iyi pozisyonu olan $g_{best}$ değerine ulaşır.

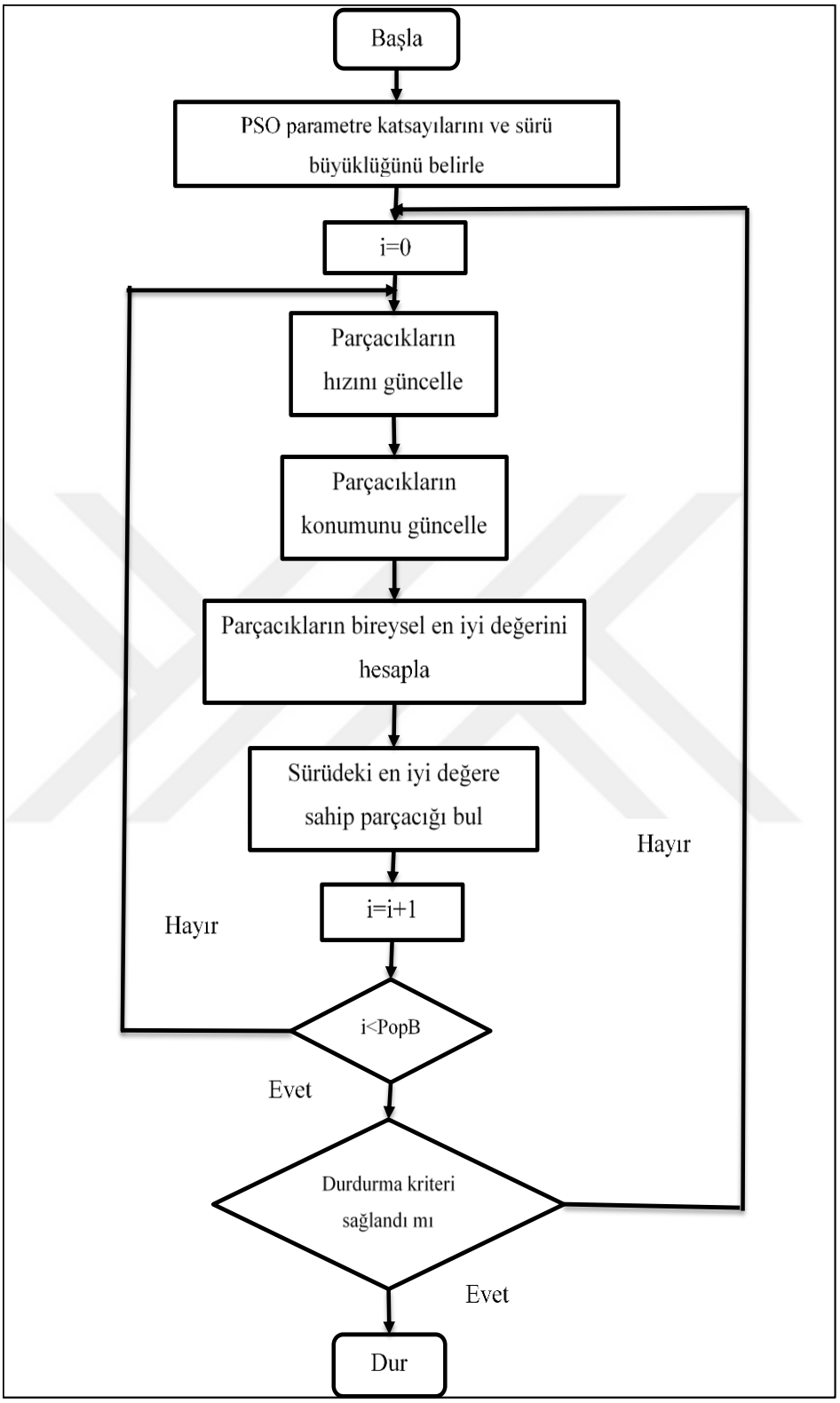
Zaman pencereli araç rotalama probleminde n müşteri sayısı k araç sayısı olmak üzere n+k-1 boyutlu vektör oluşturulur. Her bir parçacık zaman pencereli araç rotalama probleminde rotayı ifade eder. Her bir parçacık kendine ait hız vektörü ile optimum çözüme doğru hareket eder. Her bir iterasyonda parçacığın sahip olduğu mevcut hız

önceki en iyi sonuçlardan da faydalanarak yeniden hesaplanır. Sürüde bulunan bireyler daha iyi konuma ulaşırlar.

Parçacıklar bir önceki iterasyonda elde ettikleri arama tecrübesine göre hızlarını ayarlayarak arama uzayında hareket ederler. Parçacıklar arama uzayında her hareket ettiklerinde kendi koordinatları için uygunluk fonksiyonu hesaplanır. Her bir parçacık sahip olduğu hafıza sayesinde hareket edeceği koordinatları ve kendi koordinatlarını, hızını, güncel en iyi uygunluk değerini hatırlamaktadır. (Gong, ve diğ., 2012)

Algoritma adımları aşağıdaki gibidir:

- Adım 1: Popülasyon oluşturularak rastgele üretilen başlangıç pozisyonları ve hızları ile birlikte parçacıklar atanır.
- Adım 2: Sürü içerisindeki tüm parçacıklar için önceden belirlenen amaç fonksiyonuna göre uygunluk değerleri hesaplanır.
- Adım 3: Her nesildeki parçacıklar bir önceki neslin en iyisi ile karşılaştırılır ve daha iyi birey varsa bu $p_{best}$ değeri olarak atanır. Aynı şekilde nesildeki en iyi değer global en iyi değerden daha iyi ise $g_{best}$ değeri olarak atanır.
- Adım 4: Denklem (3.1)'e göre parçacık hızları, denklem (3.2)'ye göre parçacık konumu güncellenir.
- Adım 5: Başlangıçta durdurma kriteri sağlanıncaya kadar adım 2-3-4'e devam edilir.




**Şekil 3.8:** PSO Akış Diyagramı (Gong, ve diğ., 2012).

Şekil 3.8'de PSO'ya ait basit akış diyagramı gösterilmektedir.

### 3.2.4. Parçacık Sürü Optimizasyonu Üstün ve Zayıf Yönleri

#### *3.2.4.1. Üstün Yönleri*

Parçacık sürü optimizasyonu genetik algoritmalara göre ayarlanması gereken az sayıda parametreye ihtiyaç duyar. Bu sayede uygulamada kullanıcıya kolaylık sağlar. Diğer meta-sezgisel yöntemlerle melez uygulamalar oluşturmada uyum sağlayarak klasik PSO'nun performansını artırmaktadır. Mutasyon ve çaprazlama operatörlerine sahip olmadığı için araştırma parçacık hızı ile uygulanır bu yüzden arama hızı yüksektir. Daha büyük boyutlu optimizasyon problemlerinde bile kolaylıkla çözüm kümesine ulaşılır. Genetik algoritmalardan farklı olarak gerçek değerleri kullanır. (Selvi & Umarani, 2010)

PSO diğer evrimsel yöntemlere göre yüksek hesaplama kapasitesine sahiptir. Kontrol parametreleri algoritmanın başında tanımlandığı için kararlı bir çözüme daha çabuk ulaşır. Diğer üstün yönleri aşağıdaki gibidir:

- Parçacık sürü optimizasyonu diğer meta-sezgisel yöntemlere göre daha basit kurulum gösterir.
- Melez olarak diğer meta-sezgisel yöntemlere kolaylıkla uygulanabilir.
- Hızlı ve bilgisayar programlarına adapte edilmesi kolaydır.
- Çok fazla zaman ve hafızaya ihtiyaç duymazlar.
- Diğer meta-sezgisel yöntemlerde olduğu gibi yeni bir optimizasyon problemine kolaylıkla uygulanabilir, çok fazla ayarlama gerektirmez.
- Kesikli ve sürekli değişkenlere kolaylıkla uygulanabilir.

#### *3.2.4.2. Zayıf Yönleri*

Sürü içerisinde bulunan bireylerdeki farklı özellikler ( müşteri talepleri, araç kapasitesi ve zaman aralığı) optimum konum bulmayı zorlaştırmaktadır. Kesikli ve sürekli optimizasyon problemlerine uygulanmasına rağmen sürekli optimizasyon problemlerinde daha iyi sonuçlar vermektedir. Kontrol parametrelerinin iyi tanımlanmaması veya problemin kendisinin yetersiz olmasından dolayı bazı diğer meta-sezgisel yöntemlerde olduğu gibi yerel optimum noktalarda aramayı sonlandırır.

Parçacık sürü optimizasyonu sürekli arama uzayında daha iyi sonuçlar verir. Bu yüzden zaman pencereli araç rotalama problemi gibi kesikli durumlarda bazı melez yöntemlerin

uygulamaya alınması problemin çözümünü zorlaştırarak çözüme ulaşma süresini arttırmaktadır.

## 3.3. KARINCA KOLONİSİ OPTİMİZASYONU

### 3.3.1. Genel Bilgiler

Doğada koloni halinde yaşayan karıncalar karşılaştıkları problemleri kolonide bulunan diğer karıncalarla birlikte ortaklaşa çözmektedir. Karıncalar görme yetisine sahip olmadan yuvalarından yiyecek kaynağına veya yiyecek kaynağından yuvalarına ulaşmak için en kısa yolu bulmaya çalışmaktadır. Karıncalar çevrelerindeki değişime uyum sağlama yeteneğine sahiptir. (Karaboğa, 2014)

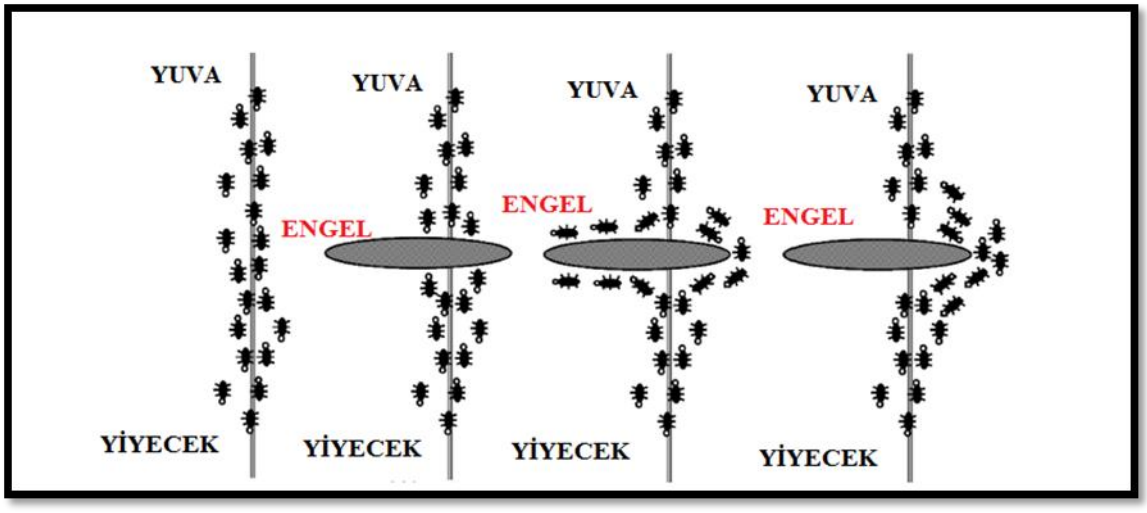
Karınca kolonisi optimizasyonu (KKO) 1990'ların başında Dorigo ve arkadaşları tarafından gezgin satıcı yöntemlerine uygulanan öğrenme mekanizmalı ve popülasyon tabanlı meta-sezgisel yöntemdir. Karmaşık yapıdaki kombinasyonel (NP-zor) problemlerin çözümünde kullanılır. Optimizasyon problemlerinin çözümü için gerçek karıncaların yiyecek arama davranışlarından esinlenerek matematiksel yeni bir algoritma geliştirilmiştir. Karıncalar sürü halinde yuva ve yiyecek kaynağı arasındaki en kısa yolu bulmak için sürü zekâsından yararlanmaktadır. Karınca sürüsü veya yuvalarından besin kaynağına doğru veya besin kaynaklarından yuvalarına yürürken yol üzerine iz bırakmak için feromon adı verilen aromatik kimyasal bir madde bırakırlar. Feromon, karınca gibi bazı canlıların kendi türündeki bireyler ile dolaylı olarak iletişim kurmalarını sağlar. Bıraktıkları feromon sayesinde sürüdeki karıncalar ile iletişim kurularak bu karıncaların da yeni besin kaynağına ulaşılmasına olanak sağlanmaktadır. Görme yetileri olmadan yönlerini bulmaları feromon sayesindedir. Eğer önlerine engel konulursa, feromon miktarının nerede fazla olduğunu bulamayacak olmalarından dolayı karıncalar gideceği alternatif yolları rastsal olarak seçerler. Kısa yoldan geçiş sayısı daha fazla olmasından dolayı bu yolda feromon miktarı da fazla olacaktır. Belli bir süre sonra kısa yolu tercih eden karınca sayısında artış olacaktır. Karıncaların kısa yolu tercih etmelerinin bir diğer sebebi sürüde bulunan bazı karıncaların uzun yoldan giden karıncaların feromon miktarı (Karaboğa, 2014)

KKO, gerçek karıncaların davranışlarından esinlenerek yapay karınca kolonisi optimizasyon aracı olarak kullanılmaktadır. Alternatif yönlü durumlarda yapay

karıncalar feromon maddesine göre yönlerini bulurlar. Yeni gelen yapay karıncaların kısa yolu daha yüksek olasılıkla seçme olasılığının bulunmasının nedeni buradaki feromon miktarının fazla olmasıdır. Bir yoldan geçen karınca sayısı arttıkça yoldaki feromon miktarı artmakta, buna bağlı olarak buradan geçen karınca sayısında da artış olmaktadır. Yapay karıncalar gerçek karıncalara göre tamamen kör olmayıp, optimizasyon problemine göre detaylara sahiptirler, zamanın kesikli olduğu çevrede yaşarlar, hafızaya sahiptirler, problemin çözümü için oluşturdukları bilgileri belirli bir süre hafızalarında tutarlar. Elde ettikleri çözüm kalitesine göre çözümü oluşturan arklar üzerine iz bırakırlar.

KKO popülasyon tabanlıdır. Optimizasyon problemlerinde aday çözümlere ulaşılmasını sağlayan bileşen kümesi mevcuttur. Bu küme problem boyunca değişmez. Popülasyonda feromon olarak adlandırılan uygunluk değeri kullanıcıya bağlı olarak değişir. KKO'da her bir iterasyonda karınca izleri değerlendirmeye alınır. (Luke, 2014)

KKO sayesinde yüksek kaliteli çözümler üretilir. İterasyonlu bir algoritma olduğu için her iterasyonda belirli sayıdaki yapay karınca çözüme katkı sağlar. Zaman pencereli araç rotalama probleminde bu algoritmanın kullanım amacı rota sayısını azaltarak toplam taşıma maliyetlerini en küçüklemektir. (Balseiro, ve diğ., 2011)




**Şekil 3.9:** Gerçek Karıncaların En Kısa Yolu Bulması.

Şekil 3.9'da Dorigo ve Gamberdella (1996) tarafından yapılan deneyle gerçek karıncaların yuvaları ile yiyecek kaynağı arasında izledikleri yol resmedilmektedir. Başlangıçtaki yol önceden gerçek karıncalar tarafından bulunan en optimum yoldur. 2.

Durumda yola engel konularak karıncaların hareketlerinin rastgele olduğu görülmektedir. 3. Durumda karıncaların sağdan ve soldan hareket eden karıncaların sayısı eşittir. 4. Durumda engel etrafındaki daha kısa yolu seçen karıncalar, diğer yolu tercih eden karıncalara göre bulundukları yere daha yoğun feromon bırakırlar. Böylece karıncalar kısa süre içerisinde daha kısa yolu tercih etmektedir. (Karaboğa, 2014)

### 3.3.2. Karınca Kolonisi Optimizasyonu Genel Kavramlar

#### *3.3.2.1. Gerçek Karıncalar*

Arjantin karıncalarının (Iridomyrmex humilis) yiyecek arama ve yuvaya taşıma süreci incelenerek gerçek karıncaların davranışları araştırılmıştır. Gerçek karıncalar, yüksek sosyal örgütlenmeye sahip canlı topluluğudur. Bireysel olarak zeki davranışlar sergilemezler. Koloni halinde yaşayan karıncalar belirli bir amacı gerçekleştirmek için daha zeki davranışlar sergilemektedirler. Koloni içerisinde bulunan karıncalar kendi yuvaları ile yiyecek kaynağı arasında en kısa yolu bulma yeteneğine sahiptir. Yiyecek aramaya karar veren karınca görme yetisini kullanmadığı için, başlangıçta bilmediği yolu rastgele olarak seçmektedir. Seçtikleri bu rastgele rotaya feromon bırakırlar. Bu feromonlar ile kendinden sonra gelen karıncalar ile iletişim kurularak koloninin en kısa yoldan yiyecek kaynağına ulaşması sağlanır. Yoldaki feromon yoğunluğunun fazlalığı bu rotayı diğer karıncalar arasında daha cazip hale getirir. Eğer karıncaların kullandığı rota üzerinde herhangi bir değişiklik olursa, yeni değişime göre kolaylıkla uyum sağlarlar.

#### *3.3.2.2. Yapay Karıncalar*

Karınca kolonisi algoritması arama uzayında gerçek karıncalar yerine yapay karıncaları kullanır. Yapay karıncalar, gerçek karıncaların yiyecek arama işlemini taklit ederek en kısa sürede yiyecek bulmayı hedeflemektedir. Gerçek karıncaların davranışlarından esinlenilerek oluşturulan yapay karıncalar da yiyecek ile yuva arasındaki iletişimini feromon adı verilen kimyasal madde ile sağlamaktadır. Yapay karıncaların salgıladığı yapay feromon yeni çözümler oluşturmak için karınca kolonisi algoritmasına yardımcı olmaktadır.

Feromon miktarının fazla olduğu yollar kendinden sonra gelecek bireylerin rota seçimine yardımcı olmaktadır. Yapay karıncaların sahip oldukları yetenekler aşağıdaki gibidir:

- Yapay karıncalar gerçek karıncaların aksine arama uzayında problemin olası çözümlerini hafızalarında tutabilme yeteneğine sahiptirler.
- Arama uzayında iterasyonlu çözümler üreterek yerel ve global çözümlere katkı sağlarlar.
- Gerçek karıncalar gibi tamamen kör değildirler. Problemlere özgü verilere dahili belleklerinden ulaşabilirler.
- Her bir yapay karınca için feromon güncellemesi probleme özgü olarak yapılmaktadır.
- Gerçek karıncaların aksine sürekli olmayan kesikli bir uzayda yaşarlar.
- Yapay karıncalar koloni içerisinde bulunan diğer karıncalar ile feromon maddesine göre stigmergy denilen iletişim mekanizmasını kullanırlar. Stigmergy, Türkçe karşılığı bulunmamakla birlikte, ortamdaki iki mekanizmanın birbirleriyle dolaylı iletişimini sağlamaktadır. Bu sayede aynı çevre içerisindeki bireylerin birbirlerinin hareketlerini yöneterek sistematik aktivite yapmalarına olanak sağlamaktadır.

#### *3.3.2.3. Geçiş Kuralı*

Geçiş kuralı her bir koloni için çözüm kümesi oluşturma aşamasıdır. Yeni çözüm kümesinin üretilmesini kapsamaktadır. Karınca kolonisi optimizasyonunda yapay karınca sayısı kadar rota kurulmaktadır. Öncelikle bütün yapay karıncalar depoda yer almakta, daha sonra rastgele müşteri noktalarına hareket etmektedir. Her bir yapay karınca bir aracı temsil etmektedir. Karınca popülasyonu harekete depodan başlamaktadır. Feromon ve bazı sezgisel parametreler kullanılarak olasılıklı rota oluşturulmakta, yapay karıncaların talep noktalarını adım adım ziyaret etmesi gerekmektedir. Daha sonra karıncaların tekrar başlangıç noktası olan depoya ulaşması amaçlanmaktadır. Kapasite ve zaman kısıtlarına göre yeni bir karınca yeni bir rota için harekete başlamalıdır. Bu adımlar hiçbir karınca ve müşteri noktası kalmayana kadar devam etmektedir.

Karınca kolonisi algoritmasında yol tercihi yaparken iki alternatif söz konusudur. İlk alternatif Denklem (3.3) ile feromon izi ve sezgisel parametreler kullanılarak her karıncanın bir sonraki adımda hangi müşteri noktasını seçeceğinin olasılığı hesaplanmaktadır.

$$p_{ij}^{k} = \begin{cases} \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{u \in N_i^k}[\tau_{iu}]^{\alpha}[\eta_{iu}]^{\beta}} , eğer\ j \in N_i^k \\ 0 , \quad diğer\ durumlarda \end{cases} \tag{3.3}$$

$p_{ij}^{k}$, k karıncasının i müşterisinin j müşterisini ziyaret etme olasılığını, $\tau_{ij}$, i ve j müşterileri arasındaki feromon miktarını, $\eta_{ij}$ sezgisel seçilebilirlik parametresini $(\eta_{ij} = \frac{1}{d_{ij}})$ $\alpha$ ve $\beta$ ise feromon izi ve sezgisel bilgilere ilişkin karar verilmesini sağlayan ayarlanabilir parametreleri, $N_i^k$ k karıncasının i şehrindeki uygun komşuluğu ifade etmektedir. feromon miktarı göz önüne alınarak olasılık dağılımına göre yollar seçilmektedir.

İkinci alternatif, yapay karıncanın gidebileceği rotalar içerisinden feromon miktarına bağlı olarak hesaplanan seçim değerlerinden en yüksek olanının seçilmesidir. i müşterisinde bulunan k karıncasının bir sonraki j müşterisini seçmek için Denklem (3.4) olasılık formülü aşağıdaki gibidir:

$$j = \max\{[\tau(i,u)]^{\alpha} * \eta(i,u)]^{\beta}\}\ u \in j_i^k \qquad eğer\ q \leq q_0 \tag{3.4}$$

$q_0$ olasılık parametresi genellikle 0,90 olarak belirlenir. $j_i^k$ i müşterisinde bulunan k karıncası tarafından henüz gidilmemiş müşterileri temsil etmektedir.

Zaman pencereli araç rotalama probleminde bütün karıncalar uygun çözümler oluşturdukları zaman çevrim tamamlanır.

#### *3.3.2.4. Feromon Güncellemesi*

Bütün karıncaların rotaları oluşturulduktan sonra, feromon güncellemesi yapılarak turlarını tamamlaması sağlanır.

Karınca kolonisi algoritmasında feromon güncellemesi yeni çözümler oluşturmada önemli bir yer tutmaktadır. Karıncalar çevrelerine feromon bırakarak diğer karıncalar ile dolaylı iletişim sağlamaktadır. Bir ortamda feromon miktarı veya feromon yoğunluğu fazlaysa diğer karıncaların da bu rotayı izleme olasılığı yüksektir. KKO'da feromon güncelleme optimal rotayı bulmaya çalışmaktadır. Yapay feromon sayesinde optimal çözüme ulaşma olasılığı artmaktadır. Çözüm uzayında araştırmanın en iyi şekilde yapılması için feromon güncellemesinin yapılması gerekmektedir.

Feromon güncellemesi yerel ve global olmak üzere iki yolla yapılmaktadır.

Yerel güncelleme, çözüme ulaşmak için rota oluşturma sürecince yapılmaktadır. Yerel güncellemenin amacı en optimal çözüme ulaşmak için her yapay karıncanın kendi rotasında hareket ederken dinamik olarak geçtiği bölgelerdeki feromonu azaltmaktır. Böylece geçtiği bölgelerde feromon miktarı azaltılarak kendinden sonra gelecek diğer yapay karıncaların başka rotalara yönlendirmesi sağlanmaktadır.

Yerel güncelleme işlemi Denklem (3.5)'deki gibidir:

$$\tau_{ij} = (1-p)\tau_{ij} + (p)\tau_0 \tag{3.5}$$

$p$ parametresi [0 1] aralığında feromon buharlaştırma oranını, $\tau_0$ parametresi başlangıç feromon miktarını ifade etmektedir. Herhangi bir karınca i noktasından j noktasına hareket ederken Denklem (3.4)'deki gibi feromon miktarını azaltmaktadır. Böylece kendinden sonra gelen karıncaların farklı rotaları kullanarak farklı çözümler bulunması sağlanır. Karıncalar henüz ziyaret etmedikleri rotaları kolaylıkla keşfedebilecektir.

Global güncelleme, rota kurulumu gerçekleştikten sonra yapılmaktadır. Global güncellemenin amacı en iyi çözümün komşuluğunda yoğunlaşmaktır.

Global güncelleme işlemi Denklem (3.6)'deki gibidir:

$$\tau_{ij} = (1-p)\tau_{ij} + (p)/J_{\psi}^{global} \qquad \forall\,(i,j) \in \psi^{global} \tag{3.6}$$

$J_{\psi}^{global}$, bulunan en uygun karıncalar tarafından bulunan en kısa rotanın uzunluğudur.

Her çevrim sonunda global feromon güncellemesi yapılarak en optimal çözüm bulunarak döngü tamamlanır.

### 3.3.3. Karınca Kolonisi Optimizasyonu Metodolojisi

Karıca Kolonisi Optimizasyonu (KKO), NP-zor problemler için makul bir zaman dilimi içerisinde iyi çözümler elde etmek için yaklaşık algoritmalar kullanmakta, karıncaların yiyecek ve yuva arasında rastgele hareketler ile en kısa yolu bulmaya çalışması esasına dayanmaktadır.

KKO, gerçek karınca kolonisi yiyecek arama sırasında ortaya koydukları davranışların matematiksel modeller ile ifade edilmesidir. Gerçek karınca davranışlarının geliştirilmesi ile yeni yapay karınca kolonisi oluşturularak NP-zor problemlerin çözümü kolaylaşır.

Her bir adımda, araç sayısı kadar karınca zaman pencereli araç rotalama probleminin kısıtlarına uyarak ve feromon güncellemesi yapılarak rastgele çözüm kümesi oluşturulmaktadır. Çözüm kümesine yerel arama uygulanır. Bir sonraki adımda feromon güncellemesini gerçekleştirmek için bazı çözümler kullanılır. Başlangıç feromonu 0'dan büyük olmalıdır.

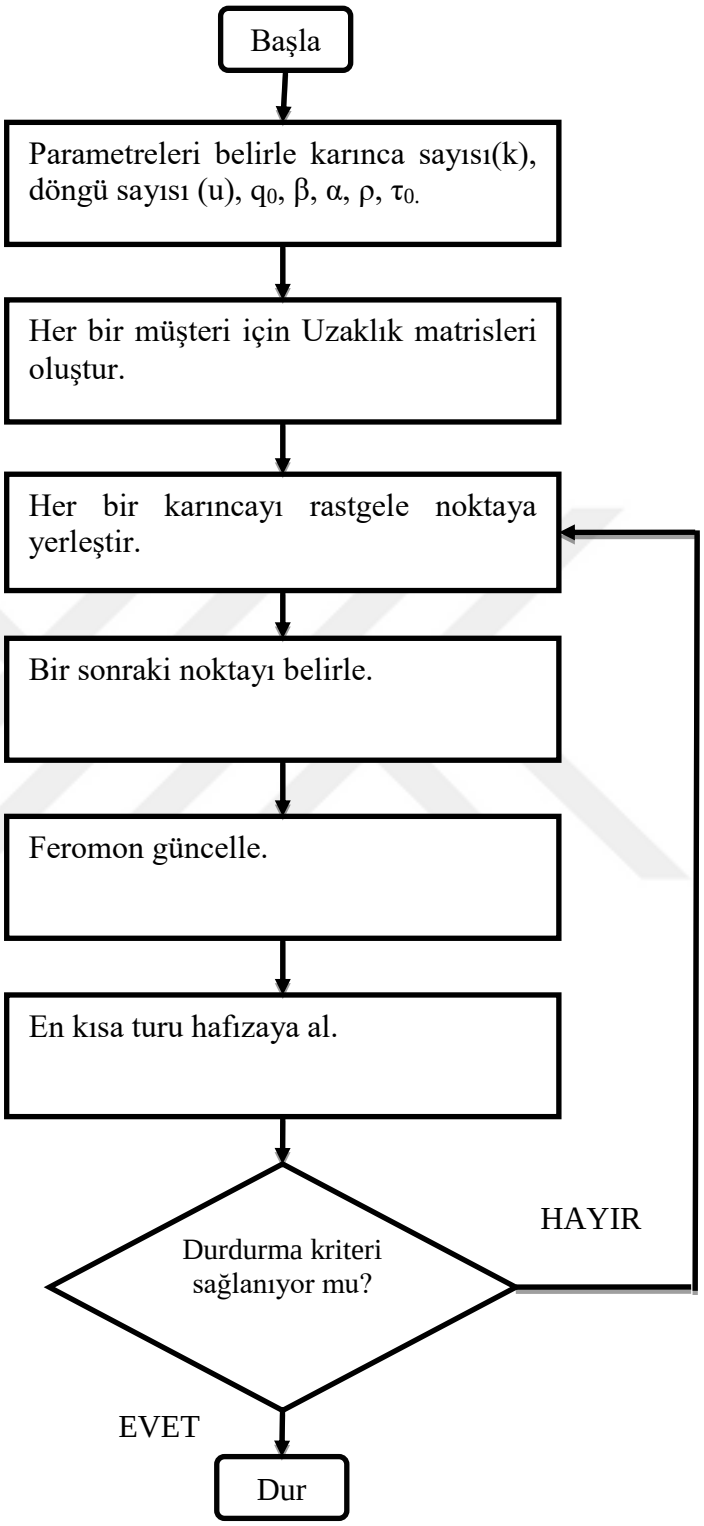
Zaman pencereli araç rotalama problemlerinde rota karınca hareketleri ile kurulur. Her bir karınca rota veya araçları temsil etmektedir. Her karıncanın başlangıç noktası depodur. Rotalama depoda başlar, bütün müşterilere hizmet verilerek tekrar başlangıç noktası olan depoda biter. Her müşteri noktası karınca tarafından bir defa ziyaret edilmelidir. Eğer karınca araç kapasitesinden fazla yüklenirse veya müşterinin zaman kısıtlarını aşar ise depoya geri dönmelidir. Böylece bir araç için rota tamamlanır. Karınca depoya geri döndüğünde, karıncanın yükü ve zamanı sıfırlanır. Yeni müşteri noktaları ile yeni rotalar bütün müşterilerin ihtiyaçları karşılanana kadar devam eder. KKO'da her bir adımda birbirinden bağımsız çözümler kurulmaktadır. Bu çözüm kümesi olasılıklı feromon güncellemesi ile çözüm kümesi içindeki uygun sayısal değerlerden oluşturulmaktadır. Çözümler bulunduktan sonra bazı değerler feromon değerlerine göre güncellenir. (Bluma, ve diğ., 2011)

Zaman pencereli araç rotalama problemlerinde çoklu-koloni (multi-colony) KKO metodolojisi kullanılmaktadır. İki koloni kullanılarak oluşturulan bu sistemde her bir koloni bir amacı gerçekleştirmektedir. Örneğin ilk oluşturulan koloni toplam hizmet süresini minimize ederken, ikinci koloni kullanılan araç sayısını minimize etmektedir. Bir kolonide amaca ulaşılırsa diğer koloninin çözümüne ihtiyaç duyulmaz. (Mavrovouniotis & Yang, 2015)

Aşağıda karınca kolonisi algoritmasının adımları bulunmaktadır:

- Adım 1: Karınca sayısı, iterasyon sayısı, α, β parametreleri belirlenir.
- Adım 2: Tüm talep noktalarına rastsal olarak karınca yerleştirilir. Karıncaların hafızaları yenilenerek her yolun feromon miktarı sıfırlanır.
- Adım 3: Her talep noktasındaki tüm karıncalar için geçiş kuralında hesaplanan olasılık değerine göre $\left(\frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{u \in N_i^k}[\tau_{iu}]^{\alpha}[\eta_{iu}]^{\beta}}\right)$ karıncanın hareket edeceği bir sonraki talep noktası belirlenir. Karınca k, j noktasına gittiğinden dolayı bu nokta k karıncasının hafızasına alınır.
- Adım 4: Her bir karınca için kat ettiği yolların toplam uzunluğu hesaplanır ve yerel feromon güncellemesi yapılır. Feromon miktarı hesaplanır.
- Adım 5: Şu ana kadar bulunan en kısa tur hafızaya alınır. Durdurma kriteri sağlanıyorsa, adım 6'ya gidilir. Aksi takdirde karıncanın hafızası boşaltılır. Tüm talep noktalarına belirli miktarda karınca yerleştirilir ve adım 3'e gidilir.
- Adım 6: En kısa tur yazılır ve algoritma tamamlanır.

Şekil 3.10'da Karınca kolonisi optimizasyonun akış diyagramı gösterilmektedir.




**Şekil 3.10:** Karınca Kolonisi Optimizasyonu Akış Diyagramı **(Karaboğa, 2014)**.

### 3.3.4. Karınca Kolonisi Optimizasyonu Üstün ve Zayıf Yönleri

#### *3.3.4.1. Üstün Yönleri*

Karınca kolonisi optimizasyonu zaman pencereli araç rotalama problemlerinde kullanıcıya büyük avantaj sağlamaktadır. Diğer meta-sezgisel yöntemlerin sağladığı avantajların yanı sıra değişen verilere göre kolaylıkla uyum sağlayabilmektedir. Esneklik özelliği sayesinde çözüm kümesine güçlü bir şekilde yakınsamaktadır. İlk uygulama gezgin satıcı problemlerinde kullanıldığı için araç rotalama problemleri türlerinde etkin çözümler bulmaktadır. İyiye yakın sonuçlar elde etmek için karınca kolonisi optimizasyonu güçlü meta-sezgisel yöntemdir.

Karınca kolonisi optimizasyonu sürekli problemlerde kullanılmakta olup, gerçek zamanlı değişikliklere kolaylıkla uyum sağlayabilmektedir. Karınca kolonisi optimizasyonu gerçek karınca davranışlarını taklit ederek karmaşık yapıdaki problemlerin çözümünde kullanılmaktadır. Basit iletişim mekanizmasıyla araç rotalama problemlerinde iki nokta arasındaki en kısa yoluyla kısa süre içerisinde bulma özelliğine sahiptir. Açgözlü algoritmaların çözümünde uygulanabilmektedir. Karınca kolonisi optimizasyonu parametreler ile çalışmaya başladığından arama yapılırken parametreler dinamik olarak ayarlanmaktadır. (Talbi, 2009)

KKO kullanılarak çözüm uzayındaki değişikliklere rağmen, yeni rota hareketleri ile karıncanın optimum yolu bulması kısa bir sürede gerçekleşmektedir. Rotalama süreci boyunca karıncalar bireysel kabiliyetleri ve feromon güncellemesi sayesinde çok yüksek örgütleme yapısına sahiptir. Diğer meta-sezgisel yöntemlere göre hesaplama süresi ve olurlu çözüm kümesi gibi avantajlara sahiptir. (Xie, ve diğ., 2014)

KKO açık ve zeki arama özellikleri sayesinde hesaplama sırasında doğruya yakın çözümler sunmaktadır. Araç rotalama problemlerinde daha verimli çözümler sağlamaktadır. Genetik algoritmalar gibi sadece önceki neslin bilgilerini hafızasında tutmaz, bütün koloni bilgilerini hafızasında tutmaktadır. Genetik algoritmalar ile birlikte kolaylıkla zaman pencereli araç rotalama gibi ayrık kısıtlı problemlerin çözümü için melez yapı oluşturulmaktadır.

#### *3.3.4.2. Zayıf Yönleri*

Karınca kolonisi optimizasyonu çok fazla parametre ayarlaması içerdiğinden teorik analizi ve anlaşılabilirlik açısından zorluk gösterebilmektedir. Belirsiz yakınsama zamanına sahiptir. Karınca kolonisi optimizasyonunda kullanılan α ve β parametreleri bazı problemlerde çok duyarlı olabilmektedir. KKO yerel optimum noktalara takılabilir. Bu durum arama verimliliğini de olumsuz etkilemektedir.

Zaman pencereli araç rotalama problemleri için daha fazla müşteri noktasına sahip olunan durumlarda çözüm kümesine yakınsamak zorlaşmaktadır. Diğer meta-sezgisel yöntemlere göre daha yavaş yakınsama riski bulunmaktadır. Müşteri sayısının arttığı ayrık problemlerde kaliteli çözüm kümesine ulaşmak için zayıf performans göstermektedir. ZPARP'nde kullanılan karıncalar henüz ziyaret edilmeyen müşteriler için uygun olmayan çözümler üretebilmektedir. Olurlu olmayan durumlar için bazı ilave prosedürler uygulanarak uygun çözümler elde edilmektedir. Böylece bütün müşteriler için rotalama yapılmakta, geçici uygulanan yöntemler kaldırılmaktadır. Genellikle diğer sezgisel veya meta-sezgisel yöntemler ile melez yapı oluşturularak olurlu ve etkin çözüm kümesine ulaşılmaktadır. (Balseiro, ve diğ., 2011)

KKO'nun diğer sezgisel yöntemler ile birlikte melez yapı oluşturması büyük ölçekli problemlerde uygulama açısından karmaşıklık göstermektedir. Bu durum aynı zamanda büyük ölçekli problemlerde çözüm kalitesini düşürmektedir.

# 4. BULGULAR

## 4.1. PROBLEM TANIMI

Malzeme ve yöntemler bölümünde bahsedilen genetik algoritmalar bu bölümde problem çözümünde kullanılacaktır. Başlangıç çözüm rastsal olarak üretilecek, permütasyon tipi sıralama kullanılacaktır. Bu çalışmada problem verisi olarak Solomon'un (1987) ve Gehring & Homberger (2001)'in veri setleri ele alınmıştır. Solomon'un veri setleri http://w.cba.neu.edu/~msolomon/problems.htm adresinden alınmıştır. Gehring ve Homberger veri setleri https://www.sintef.no/projectweb/top/vrptw/homberger-benchmark/ adresinden alınmıştır.

Solomon veri setleri ZPARP testi için hâlâ sıklıkla kullanılan verileri içermektedir. Toplam 56 farklı uygulama verisi bulundurmaktadır. R1, R2, C1, C2, RC1 ve RC2 olmak üzere 6 farklı problem tipi tanımlanmaktadır. Bu problemlerin genel özellikleri aşağıdaki gibidir:

- R1 tipi problemlerde müşteriler rastsal olarak coğrafi bölgelere dağılmıştır. Müşterilere hizmet verilmesi gereken zaman aralığı sıkı olup, her rotada çok fazla müşteriye hizmet verilememektedir. Küçük sevkiyat kapasitesi bulunmaktadır.
- R2 tipi problemlerde müşteriler rastsal olarak coğrafi bölgelere dağılmıştır. Müşterilere hizmet verilmesi gereken zaman aralığı geniş olup, her rotada çok fazla müşteriye hizmet verilmektedir.
- C1 tipi problemlerde müşteriler kümelenmiş olarak coğrafi bölgelere dağılmıştır. Müşterilere hizmet verilmesi gereken zaman aralığı sıkı olup, her rotada çok fazla müşteriye hizmet verilememektedir. Küçük sevkiyat kapasitesi bulunmaktadır.
- C2 tipi problemlerde müşteriler kümelenmiş olarak coğrafi bölgelere dağılmıştır. Müşterilere hizmet verilmesi gereken zaman aralığı geniş olup, her rotada çok fazla müşteriye hizmet verilmektedir.

- RC1 tipi problemlerde müşteriler hem kümelenmiş, hem de rastsal olarak coğrafi bölgelere dağılmıştır. Müşterilere hizmet verilmesi gereken zaman aralığı sıkı olup, her rotada çok fazla müşteriye hizmet verilememektedir. Küçük sevkiyat kapasitesi bulunmaktadır.
- RC2 tipi problemlerde müşteriler hem kümelenmiş, hem de rastsal olarak coğrafi bölgelere dağılmıştır. Müşterilere hizmet verilmesi gereken zaman aralığı geniş olup, her rotada çok fazla müşteriye hizmet verilmektedir.

Her veri seti birbirinden ayrılmış coğrafi bölgelerde bulunan 100 müşteriden ve 1 depodan oluşmaktadır. R ve RC tipi problemlerde müşteri hizmet süresi 10 birim olup, C tipi problemlerde bu süre 90 birimdir. Toplam 25 adet araç bulunmaktadır. Her aracın kapasitesi R1, C1 ve RC1 tipi problemlerde 200, R2 ve RC2 tipi problemler 1000, C2 tipi problemlerde ise 700 birimdir. Tüm araçlar homojendir. İki müşteri arasında uzaklıklar Öklid uzaklığı kullanılarak hesaplanmaktadır. C1, R1 ve RC1 tipi problemler daha sıkı zaman penceresine sahip olup, küçük sevkiyat kapasitesine sahiptir. Her bir müşterinin zaman penceresi bulunmaktadır. C ve R tipi problemlerden 6 adet problem rastsal olarak seçilmiş, tez kapsamında C101, C104, C205, C206, R105, R108 problemleri ele alınmıştır.

Gehring ve Homberger veri setleri, Solomon testlerinden geliştirilerek oluşturulmuştur. 200, 400, 600, 800 ve 1000 müşteriden oluşan C1, C2, R1, R2, RC1, RC2 tip olmak üzere 6 farklı problem seti, 300 farklı uygulama verisi bulunmaktadır. 200 müşterili problem tipleri C1_2_1, C1_2_2 olarak ifade edilmektedir. Her problem tipinde 10 farklı veri bulunmaktadır. Bu çalışma kapsamında C1_2_1, C1_2_3, C1_2_5, C2_2_6, C2_2_8, C2_2_10 problemleri ele alınacaktır.

Gehring ve Homberger veri setleri, Solomon veri setleri gibi kümelenmiş, rastsal olarak coğrafi bölgelere dağılmış, hem kümelenmiş hem de rastsal olarak coğrafi bölgelere ayrılmıştır. Tablo 4.1'de Gehring ve Homberger'in problem veri setlerinin özellikleri bulunmaktadır.

**Tablo 4.1:** Homberger & Gehring Problem Tipi ve Özellikleri.

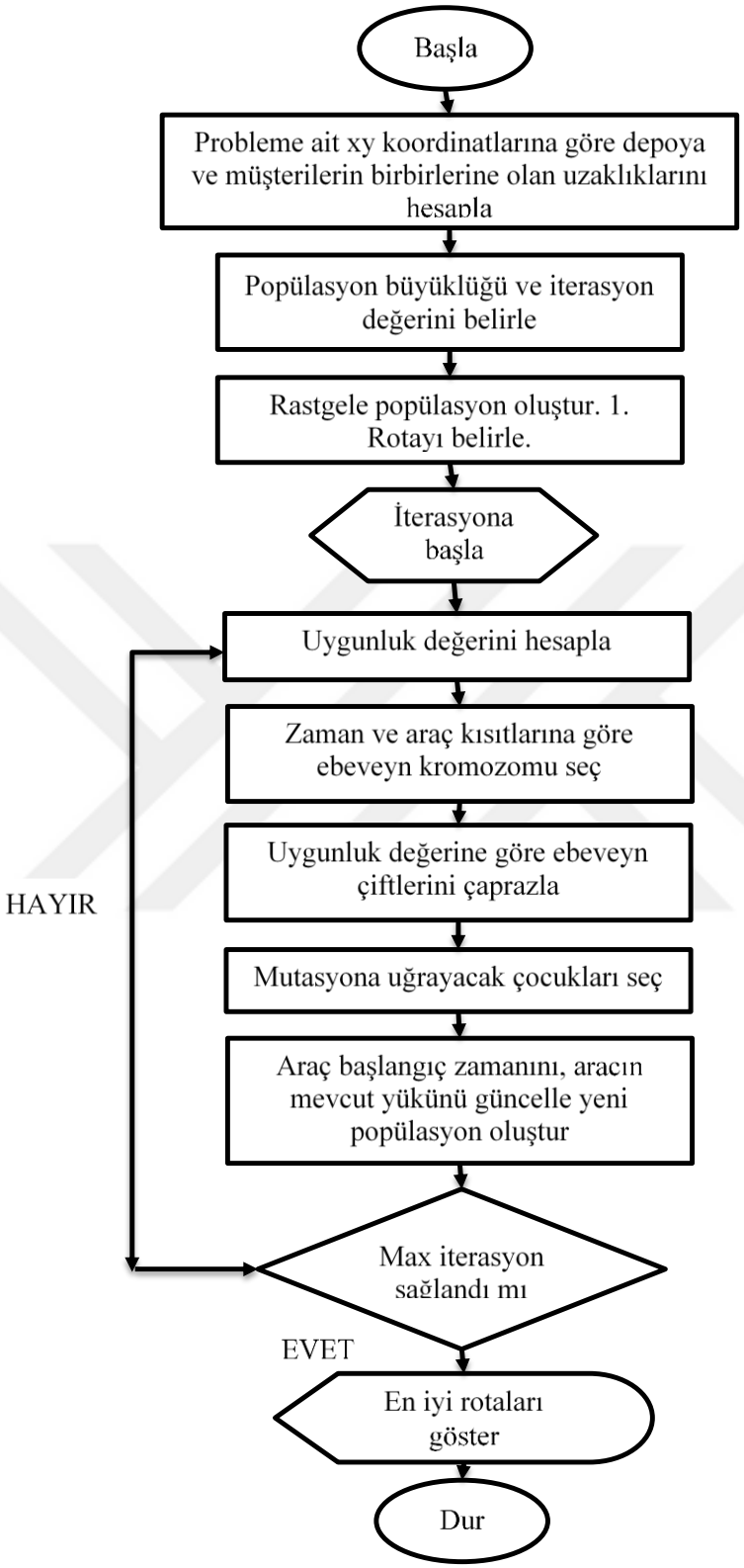
| Problem Tipi | Müşteri Sayısı | Araç Sayısı | Araç Kapasitesi | Hizmet Süresi |
|---|---|---|---|---|
| C1 | 200 | 50 | 200 | 90 |
| C2 | 200 | 50 | 700 | 90 |
| R1 | 200 | 50 | 200 | 10 |
| R2 | 200 | 50 | 1000 | 10 |
| RC1 | 200 | 50 | 200 | 10 |
| RC2 | 200 | 50 | 1000 | 10 |
| C1 | 400 | 100 | 200 | 90 |
| C2 | 400 | 100 | 700 | 90 |
| R1 | 400 | 100 | 200 | 10 |
| R2 | 400 | 100 | 1000 | 10 |
| RC1 | 400 | 100 | 200 | 10 |
| RC2 | 400 | 100 | 1000 | 10 |
| C1 | 600 | 150 | 200 | 90 |
| C2 | 600 | 150 | 700 | 90 |
| R1 | 600 | 150 | 200 | 10 |
| R2 | 600 | 150 | 1000 | 10 |
| RC1 | 600 | 150 | 200 | 10 |
| RC2 | 600 | 150 | 1000 | 10 |
| C1 | 800 | 200 | 200 | 90 |
| C2 | 800 | 200 | 700 | 90 |
| R1 | 800 | 200 | 200 | 10 |
| R2 | 800 | 200 | 1000 | 10 |
| RC1 | 800 | 200 | 200 | 10 |
| RC2 | 800 | 200 | 1000 | 10 |
| C1 | 1000 | 250 | 200 | 90 |
| C2 | 1000 | 250 | 700 | 90 |
| R1 | 1000 | 250 | 200 | 10 |
| R2 | 1000 | 250 | 1000 | 10 |
| RC1 | 1000 | 250 | 200 | 10 |
| RC2 | 1000 | 250 | 1000 | 10 |

Problemlerin çözüm amacı, müşterilerin zaman kısıtı ve araçların kapasite kısıtı göz önüne alınarak toplam mesafeyi en azlamaktır. Algoritma Matlab 8.1'de kodlanmış, 8 Gb Ram, I7 2.50 Ghz işlemcili bilgisayarda test edilmiştir. Belirlenen bu problemler Genetik Algoritma ile farklı popülasyon sayısı denenerek çözülmüştür.

## 4.2. GENETİK ALGORİTMALAR İLE TEST PROBLEMLERİNİN OPTİMİZE EDİLMESİ

Zaman pencereli araç rotalama problemlerinin çözümünde kullanılan genetik algoritmalar popülasyon tabanlı sezgisel yöntem olduğu için başlangıç popülasyon büyüklüğü seçimi önemlidir. Başlangıç popüslasyonu için iki ebeveyn belirlenir. Bu çalışmada iki ebeveyn de aynı alınacaktır. i ve j müşterilerine konumlandırılarak, uygunluk değerine göre çaprazlama işlemleri uygulanacaktır. En olası uygunluk değeri ile yeni nesil oluşturmak için ebeveynler çaprazlama işlemine tabi tutulacaktır. Çaprazlama işlemi için müşteriler rastsal olarak yer değiştirme yaparak, yeni nesil bireyler oluşturacaktır. Mutasyon operatörü aynı rotada bulunan rastsal olarak seçilen 2 müşteri arasında yer değiştirme şeklinde olacaktır. Literatürde ZPARP için önerilen genetik algoritmalarda mutasyon oranı %1-%5 arasıdır. Bu çalışmada her rotada iki müşteri seçilerek yer değiştirme mutasyonu uygulanacaktır. Bu yüzden algoritma süresince %1 alınarak sabit tutulacaktır.

Çalışmanın bu bölümünde test problemlerinin çözümü için kullanılacak yöntem için akış diyagramı Şekil 4.1'deki gibidir:



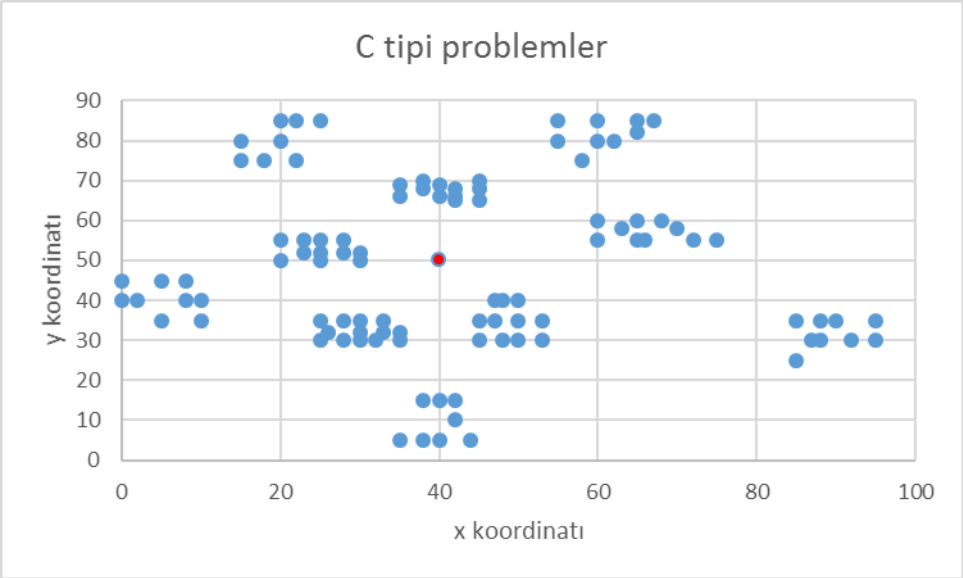

**Şekil 4.1:** ZPARP için Genetik Algoritmalar Akış Diyagramı **(Wang & Chen, 2012)**.

Uygunluk değeri n adet müşteri ve K adet araç ile toplam uzaklığı minimize edecek formülle aşağıdaki gibi hesaplanacaktır:

$$UygunlukDeğeri\,(fx) = \sum_{i=1,j=1}^{n+K} d_{ij} \quad (4.1)$$

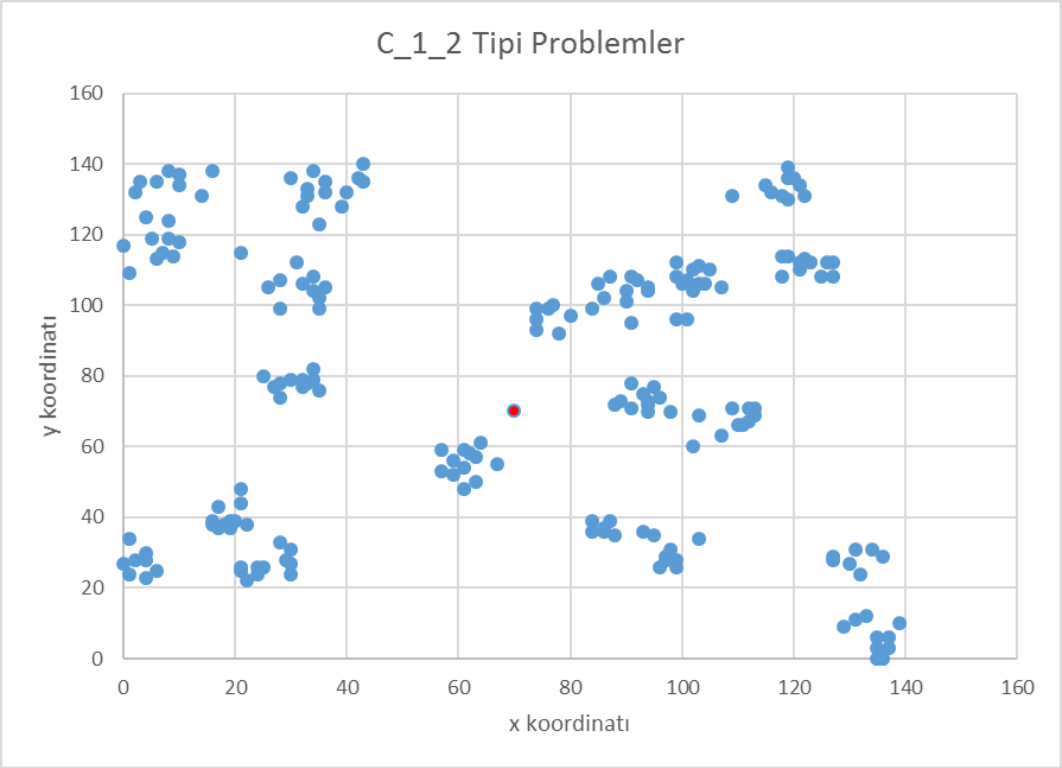
$d_{ij}$ i ve j müşterilerinin x ve y Öklit cinsinden birbirlerine uzaklıklarıdır. Aşağıdaki formülle hesaplanacaktır:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (4.2)$$



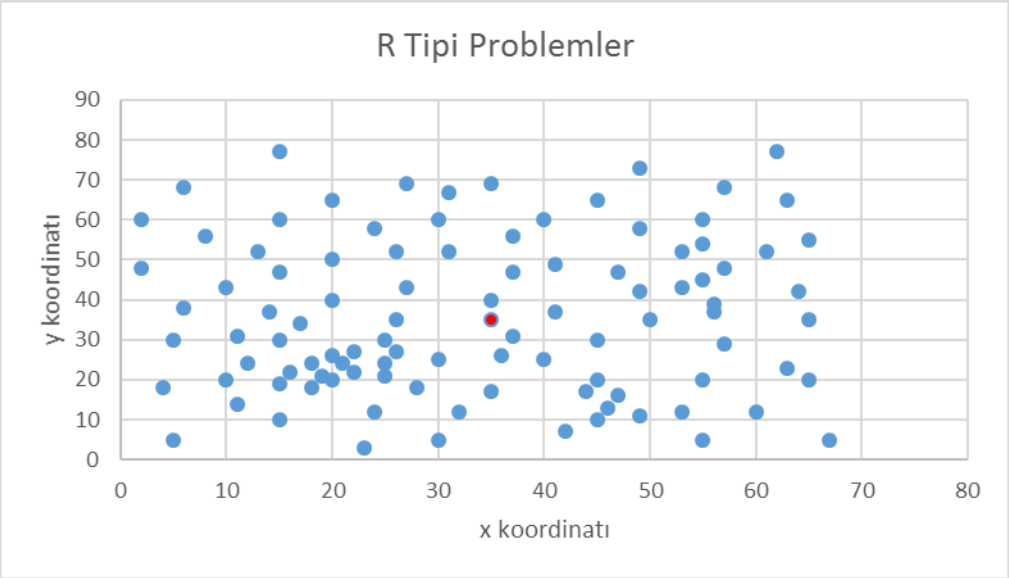

**Şekil 4.2:** Solomon C Tipi Problem Müşteri Koordinatları.

Şekil 4.2'de mavi ile işaretlenen noktalar C tipi müşterilerin x ve y düzleminde koordinatları verilmekte olup, kırmızı ile işaretli nokta deponun x ve y koordinatını göstermektedir.



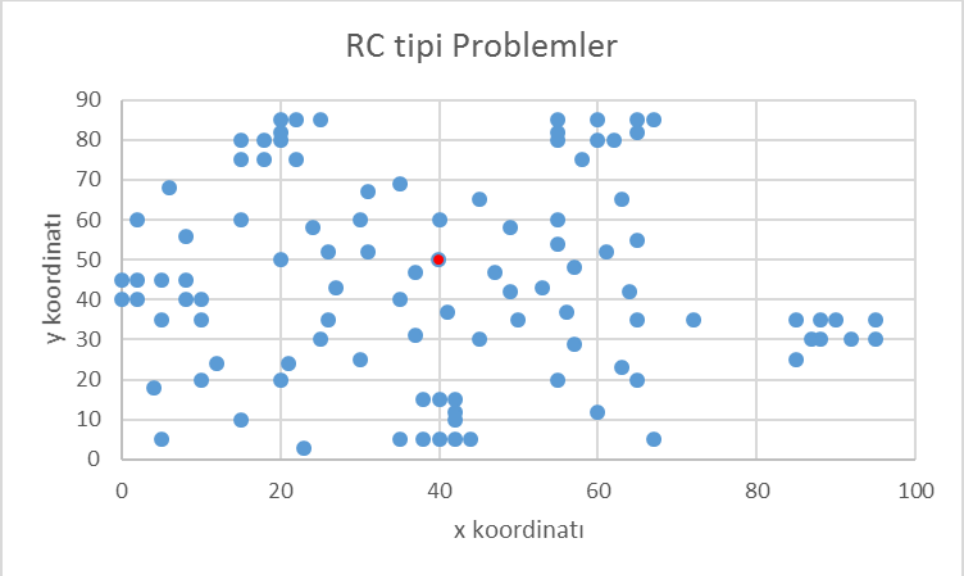

**Şekil 4.3:** Gehring & Homberger C_1_2 Tipi Problemler.

Şekil 4.3'te C_1_2 tipi problem için kümelenmiş müşteri koordinatları bulunmaktadır. Solomon veri setlerinden esinlenerek Gehring & Homberger problemleri oluşturulmuştur.




**Şekil 4.4:** Solomon R Tipi Problem Müşteri Koordinatları.

Şekil 4.4'de R tipi problem için rastgele konumlanan müşteri koordinatları bulunmaktadır.




**Şekil 4.5:** Solomon RC Tipi Problem Müşteri Koordinatları.

Şekil 4.5'te RC tipi problem için rastgele ve kümelenmiş olarak konumlanan müşteri koordinatları bulunmaktadır.

Her rotanın uygunluk değeri hesaplanırken hem depodan çıkış için harcadığı süre hem de en son müşteriden depoya gelişi için harcadığı süre hesaba katılmaktadır. Bu süre Solomon test problemleri için uzaklık birim süreleridir.

Başlangıç popülasyon sayısı (NC) problemlerin çözümü için belirleyici kriter olacaktır. Program süresince popülasyon büyüklüğü değiştirilerek optimum rotaya ulaşılmaya çalışılacaktır. Oluşturulan bu popülasyon NC adet kromozomdan oluşan n adet müşteriyi kapsamaktadır. Her bir kromozom 1'den n'e kadar rastgele dizilen genden oluşmaktadır. Çocuk oluşturmak için 2*NC adet kromozom rastgele oluşturulacaktır. Tablo 4.2'de bu çalışmada kullanılacak olan örnek başlangıç popülasyonu verilmektedir.

**Tablo 4.2:** Başlangıç Popülasyonu Oluşturma.

| MÜŞTERİ NO | 1 | 2 | 3 | … | … | … | … |
|---|---|---|---|---|---|---|---|
| KROMOZOM | 99 | 32 | 40 | … | 57 | 10 | 12 |
| KROMOZOM | 23 | 75 | 74 | … | 68 | 22 | 18 |
| … | … | … | … | … | … | … | … |
| NC. KROMOZOM | 94 | 44 | 81 | … | 97 | 26 | 60 |

Başlangıç popülasyonunda yer alan her kromozomun uygunluk değeri hesaplanır. C101 problemi için 1. Müşteri olan depodan ziyaret edilen 99. Müşteriye olan uzaklık hesaplanır. Son müşteri olan 12. Müşterinin de depoya dönüşü için geçen süre birim uzaklık cinsinden hesaplanır. NC. kromozom dahil popülasyonda bulunan bütün kromozomların uygunluk değerleri hesaplanır. Hesaplanan değerler küçükten büyüğe doğru sıralanır. Yüksek değerlere sahip kromozomlar populasyondan çıkarılır.

Bu çalışmada, problemde yer alan her genin (müşterinin) zaman kısıtı, her kromozomun (rotanın) zaman kısıtı ve kapasite kısıtı göz önüne alarak uygunluk değeri minimum yapılacaktır. Her araç deponun en geç zamandan önce depoya geri dönmelidir. Çeşitli denemeler sonunda rotada ilk müşteri rastgele seçilecektir. Uygunluk değerine göre ikinci ve diğer müşteriler seçilmeye devam edilecektir. Rotadaki her araç hareketlerinde rotada bulunan mevcut müşterinin talebi toplanarak rotanın araç kapasitesini (200 birim)

aşıp aşmadığı kontrol edilecektir. Toplam belirli sayıda araç olduğu için ve bu sayı aşılmaması için gerekli durumlarda tekrar rastgele kromozom oluşturulacaktır.

### 4.2.1. Solomon Problemlerinin Sayısal Sonuçları

Problemlerde kullanılan uygunluk değeri için aşağıdaki kısıtlar göz önüne alınmaktadır:

- Depo son tarih kısıtı: Bu kısıta göre müşteriye seyahat süresi, müşteride harcadığı servis süresi (C tipi problemlerde 90 birim, R tipi problemlerde 10 birim) toplamı aracın depoya gideceği son tarihi geçmemelidir. Araç deponun belirlenen son tarihinden önce görevini yerine getirmelidir.
- Araç kapasite kısıtı: Bu kısıta göre araç depodan çıktıktan sonra uğrayacağı müşterilerin talepleri toplamı araç kapasite kısıtını aşmamalıdır.
- Müşteri zaman aralığı kısıtı: Aracın şimdiki zamanına bir sonraki müşteriye seyahat süresi eklendiğinde bu müşterinin son tarihini geçmemesi gerekmektedir.

Bu kısıtlar altında mevcut rotaya müşteri ataması yapılır, aracın bilgileri (talebin eklenmesi, müşteriden ayrıldığı zamanın başlangıç olarak değişmesi) güncellenir. Rotaya bu müşteri eklenir ve bu müşterinin konumu aracın yeni konumu olarak güncellenir. Bu döngü, ziyaret edilmeyen müşteri kalmayana kadar veya başlangıçta programa tanımlanan araç sayısına ulaşılana kadar devam eder.

Her problem için farklı popülasyon büyüklüğü ve iterasyon sayısı ile çözümlerin iyileştirilmesi amaçlanarak, literatürdeki en iyi değerlerden sapmaları hesaplanmıştır. Sapma değeri (D) denklemi 4.3 formülle hesaplanmıştır:

$$D = \left[\frac{|f_{eniyi} - f_{çözüm}|}{f_{eniyi}}\right] x100\% \qquad (4.3)$$

#### *4.2.1.1. C101 Problemi için Sayısal Sonuçlar*

C101 probleminin çözümünde 100 müşterinin maksimum 25 araç ile ziyaret edilmesi hedeflenmektedir. Her aracın 200 birimlik kapasitesi bulunmakta olup, araç kapasitesi eşittir. Algoritma Matlab 8.1'de kodlanmış, 8 Gb Ram, I7 2.50 Ghz işlemcili bilgisayarda test edilmiştir. Bu problem için sezgisel yöntemlerle bulunan en iyi çözüm

Rochat ve Taillard, (1995)'in çalışmalarında yer almaktadır. Buna göre en iyi çözüm 10 araç ve 828,94 birimdir.

Farklı popülasyon büyüklüğü ve iterasyon ile literatürde bulunan en iyi değerlere ulaşılması hedeflenmektedir.

C101 probleminin çözümü için ilk olarak başlangıç popülasyon büyüklüğü olarak 20 kromozom alınmış, 1000 iterasyon ile problem 5 kez çalıştırılmıştır.

Tüm problem türlerinde mutasyon oranı %1 olarak alınmıştır. Böylece her rotada toplam uzaklığı azaltmak için bir veya iki müşteri kendi aralarında yer değiştirmektedir.

Çalışmaya ait elde edilen çözümler Tablo 4.3'teki gibidir:

**Tablo 4.3:** 20 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 13 | 11 | 13 | **12** | 12 | 12,2 |
| **TOPLAM UZAKLIK** | 1183,204 | 968,353 | 1020,166 | **934,489** | 966,791 | 1014,600 |
| **CPU (sn)** | 0,564 | 0,527 | 0,443 | **0,528** | 0,486 | 0,510 |
| **SAPMA** | 43% | 17% | 23% | **13%** | 17% | 22% |

20 kromozom ve 1000 iterasyon ile 0,528 CPU zamanı (sn) ile en iyi uzaklık 12 araç ile 934,489 birim olarak bulunmuştur. En iyi çözüm olan 828,94 birimden %13 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 12,2 olarak hesaplanmış, ortalama süre 0,510 saniyedir. Ortalama toplam uzaklık 1014,600 birimdir, en iyi çözümden %22 oranında sapma göstermiştir.

20 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.4'te yer alan sonuçlar bulunmuştur:

**Tablo 4.4:** 20 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 11 | 13 | 11 | 11 | **11** | 11,4 |
| **TOPLAM UZAKLIK** | 961,0038 | 1003,241 | 914,309 | 898,664 | **893,556** | 934,155 |
| **CPU (sn)** | 0,531 | 0,557 | 0,546 | 0,517 | **0,511** | 0,532 |
| **SAPMA** | 16% | 21% | 10% | 8% | **8%** | 13% |

20 kromozom ve 10000 iterasyon ile 0,511 CPU zamanı (sn) ile en iyi uzaklık 11 araç ile 893,556 birim olarak bulunmuştur. En iyi çözüm olan 828,94 birimden %8 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 11,4 olarak hesaplanmış, ortalama süre 0,532 saniyedir. Ortalama toplam uzaklık 934,155 birimdir, en iyi çözümden %13 oranında sapma göstermiştir.

Daha iyi çözüme ulaşmak için problem 25 kromozom ve 1000 iterasyon ile problem yeniden 5 defa çalıştırılmış, Tablo 4.5'te yer alan sonuçlar bulunmuştur:

**Tablo 4.5:** 25 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 11 | **11** | 11 | 11 | 11 | 11 |
| **TOPLAM UZAKLIK** | 965,237 | **889,461** | 994,100 | 889,829 | 951,573 | 938,04 |
| **CPU** | 0,615 | **0,885** | 0,667 | 0,356 | 0,64 | 0,633 |
| **SAPMA** | 16% | **7%** | 20% | 7% | 15% | 13% |

25 kromozom ve 1000 iterasyon ile 0,885 CPU zamanı (sn) ile en iyi uzaklık 11 araç ile 889,461 birim olarak bulunmuştur. En iyi çözüm olan 828,94 birimden %7 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 11 olarak hesaplanmış, ortalama süre 0,633 saniyedir. Ortalama toplam uzaklık 938,04 birimdir, en iyi çözümden %13 oranında sapma göstermiştir.

25 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.6'te yer alan sonuçlar bulunmuştur:

**Tablo 4.6:** 25 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 11 | 12 | 12 | 11 | **11** | 11,4 |
| TOPLAM UZAKLIK | 972,187 | 945,912 | 922,660 | 1001,932 | **887,723** | 946,083 |
| CPU | 0,994 | 0,935 | 2,519 | 3,591 | **0,856** | 1,779 |
| SAPMA | 17% | 14% | 11% | 21% | **7%** | 14% |

25 kromozom ve 10000 iterasyon ile 0,856 CPU zamanı (sn) ile en iyi uzaklık 11 araç ile 887,723 birim olarak bulunmuştur. En iyi çözüm olan 828,94 birimden %7 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 11,4 olarak hesaplanmış, ortalama süre 1,779 saniyedir. Ortalama toplam uzaklık 946,083 birimdir, en iyi çözümden %14 oranında sapma göstermiştir.

30 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.7'te yer alan sonuçlar bulunmuştur:

**Tablo 4.7:** 30 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | **10** | 10 | 10 | 12 | 10 | 10,4 |
| TOPLAM UZAKLIK | **893,813** | 897,736 | 898,98 | 910,789 | 912,768 | 902,817 |
| CPU | **0,885** | 0,508 | 0,247 | 0,597 | 0,594 | 0,566 |
| SAPMA | **8%** | 8% | 8% | 10% | 10% | 9% |

30 kromozom ve 1000 iterasyon ile 0,885 CPU zamanı (sn) ile en iyi uzaklık 10 araç ile 893,813 birim olarak bulunmuştur. En iyi çözüm olan 828,94 birimden %8 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 10,4 olarak hesaplanmış, ortalama süre 0,566 saniyedir. Ortalama toplam uzaklık 902,817 birimdir, en iyi çözümden %9 oranında sapma göstermiştir.

30 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.8'te yer alan sonuçlar bulunmuştur:

**Tablo 4.8:** 30 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 10 | 11 | **10** | 11 | 12 | 10,8 |
| TOPLAM UZAKLIK | 909,907 | 882,45 | **828,94** | 898,989 | 836,5 | 871,357 |
| CPU | 0,655 | 0,562 | **0,604** | 0,567 | 0,533 | 0,584 |
| SAPMA | 10% | 6% | **0%** | 8% | 1% | 5% |

30 kromozom ve 10000 iterasyon ile 0,604 CPU zamanı (sn) ile en iyi uzaklık 10 araç ile 828,94 birim olarak bulunmuştur. En iyi çözüm olan 828,94 birimden %0 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 10,8 olarak hesaplanmış, ortalama süre 0,584 saniyedir. Ortalama toplam uzaklık 871,357 birimdir, en iyi çözümden %5 oranında sapma göstermiştir.

Popülasyon büyüklüğü 30 alınarak 10000 iterasyonda en iyi çözüme ulaşılmıştır. Buna göre bulunan rotalar Tablo 4.9'teki gibidir:

**Tablo 4.9:** GA ile Bulunan Optimum Araç Rotaları.

| ROTALAR | MÜŞTERİ NO |
|---|---|
| ROTA 1 | 0 5 3 7 8 10 11 9 6 4 2 1 75 0 |
| ROTA 2 | 0 32 33 31 35 37 38 39 36 34 0 |
| ROTA 3 | 0 43 42 41 40 44 46 45 48 51 50 52 49 47 0 |
| ROTA 4 | 0 67 65 63 62 74 72 61 64 68 66 69 0 |
| ROTA 5 | 0 20 24 25 27 29 30 28 26 23 22 21 0 |
| ROTA 6 | 0 90 87 86 83 82 84 85 88 89 91 0 |
| ROTA 7 | 0 13 17 18 19 15 16 14 12 0 |
| ROTA 8 | 0 57 55 54 53 56 58 60 59 0 |
| ROTA 9 | 0 81 78 76 71 70 73 77 79 80 0 |
| ROTA 10 | 0 98 96 95 94 92 93 97 100 99 0 |

#### *4.2.1.2. C104 Problemi için Sayısal Sonuçlar*

C104 probleminin çözümünde 100 müşterinin maksimum 25 araç ile ziyaret edilmesi hedeflenmektedir. Her aracın 200 birimlik kapasitesi bulunmakta olup, araç kapasitesi

eşittir. Algoritma Matlab 8.1'de kodlanmış, 8 Gb Ram, I7 2.50 Ghz işlemcili bilgisayarda test edilmiştir.

Bu problem için sezgisel yöntemlerle bulunan en iyi çözüm Rochat ve Taillard, (1995)'in çalışmalarında yer almaktadır. Buna göre en iyi çözüm 10 araç ve 824,78 birimdir.

Farklı popülasyon büyüklüğü ve iterasyon ile literatürde bulunan en iyi değerlere ulaşılması hedeflenmektedir.

C104 probleminin çözümü için ilk olarak başlangıç popülasyon büyüklüğü olarak 20 kromozom alınmış, 1000 iterasyon ile problem 5 kez çalıştırılmıştır.

Tüm problem türlerinde mutasyon oranı %1 olarak alınmıştır. Böylece her rotada toplam uzaklığı azaltmak için bir veya iki müşteri kendi aralarında yer değiştirmektedir.

Çalışmaya ait elde edilen çözümler Tablo 4.10'daki gibidir:

**Tablo 4.10:** 20 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | **10** | 10 | 10 | 11 | 11 | 10,4 |
| **CPU (sn)** | **1,58** | 1,419 | 1,373 | 1,183 | 1,207 | 1,352 |
| **TOPLAM UZAKLIK** | **990,355** | 1.021,291 | 1.000,922 | 1.017,223 | 1.016,592 | 1.009,277 |
| **SAPMA** | **20%** | 24% | 21% | 23% | 23% | 22% |

20 kromozom ve 1000 iterasyon ile 1,58 CPU zamanı (sn) ile en iyi uzaklık 10 araç ile 990,355 birim olarak bulunmuştur. En iyi çözüm olan 824,78 birimden %20 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 10,4 olarak hesaplanmış, ortalama süre 1,352 saniyedir. Ortalama toplam uzaklık 1.009,277 birimdir, en iyi çözümden %22 oranında sapma göstermiştir.

20 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.11'de yer alan sonuçlar bulunmuştur:

**Tablo 4.11:** 20 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 10 | **10** | 10 | 11 | 10 | 10,2 |
| **CPU (sn)** | 1,433 | **1,93** | 1,89 | 2,583 | 2,32 | 2,031 |
| **TOPLAM UZAKLIK** | 1.017,95 | **980,873** | 1.003,448 | 1.003,635 | 1.010,132 | 1.003,209 |
| **SAPMA** | 23% | **19%** | 22% | 22% | 22% | 22% |

20 kromozom ve 10000 iterasyon ile 1,93 CPU zamanı (sn) ile en iyi uzaklık 10 araç ile 980,873 birim olarak bulunmuştur. En iyi çözüm olan 824,78 birimden %19 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 10,2 olarak hesaplanmış, ortalama süre 2,031 saniyedir. Ortalama toplam uzaklık 1.003,209 birimdir, en iyi çözümden %22 oranında sapma göstermiştir.

25 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.12'de yer alan sonuçlar bulunmuştur:

**Tablo 4.12:** 25 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 10 | 11 | 10 | **10** | 10 | 10,2 |
| **CPU (sn)** | 1,986 | 1,222 | 1,47 | **1,47** | 1,065 | 1,443 |
| **TOPLAM UZAKLIK** | 982,803 | 981,192 | 931,551 | **922,487** | 959,678 | 955,542 |
| **SAPMA** | 19% | 19% | 13% | **12%** | 16% | 16% |

25 kromozom ve 1000 iterasyon ile 1,47 CPU zamanı (sn) ile en iyi uzaklık 10 araç ile 922,487 birim olarak bulunmuştur. En iyi çözüm olan 824,78 birimden %12 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 10,2 olarak hesaplanmış, ortalama süre 1,443 saniyedir. Ortalama toplam uzaklık 955,542 birimdir, en iyi çözümden %16 oranında sapma göstermiştir.

25 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.13'te yer alan sonuçlar bulunmuştur:

**Tablo 4.13:** 25 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | **10** | 10 | 10 | 11 | 11 | 10,4 |
| CPU (sn) | **1,465** | 1,475 | 1,67 | 1,54 | 1,135 | 1,457 |
| TOPLAM UZAKLIK | **962,005** | 962,639 | 974,97 | 976,827 | 986,595 | 972,607 |
| SAPMA | **17%** | 17% | 18% | 18% | 20% | 18% |

25 kromozom ve 10000 iterasyon ile 1,465 CPU zamanı (sn) ile en iyi uzaklık 10 araç ile 962,005 birim olarak bulunmuştur. En iyi çözüm olan 824,78 birimden %17 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 10,4 olarak hesaplanmış, ortalama süre 1,457 saniyedir. Ortalama toplam uzaklık 972,607 birimdir, en iyi çözümden %17 oranında sapma göstermiştir.

30 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.14'te yer alan sonuçlar bulunmuştur:

**Tablo 4.14:** 30 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 11 | 10 | 10 | 10 | **10** | 10,2 |
| CPU (sn) | 0,929 | 0,968 | 0,699 | 0,706 | **0,694** | 0,799 |
| TOPLAM UZAKLIK | 900,323 | 893,191 | 880,673 | 944,950 | **856,206** | 895,069 |
| SAPMA | 9% | 8% | 7% | 15% | **4%** | 9% |

30 kromozom ve 1000 iterasyon ile 0,694 CPU zamanı (sn) ile en iyi uzaklık 10 araç ile 856,206 birim olarak bulunmuştur. En iyi çözüm olan 824,78 birimden %4 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 10,2 olarak hesaplanmış, ortalama süre 0,799 saniyedir. Ortalama toplam uzaklık 895,069 birimdir, en iyi çözümden %9 oranında sapma göstermiştir.

30 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.15'te yer alan sonuçlar bulunmuştur:

**Tablo 4.15:** 30 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 10 | 10 | 11 | **10** | 10 | 10,2 |
| **CPU (sn)** | 1,196 | 1,007 | 1,125 | **1,006** | 1,159 | 1,0986 |
| **TOPLAM UZAKLIK** | 866,531 | 901,56 | 886,490 | **826,556** | 886,001 | 873,428 |
| **SAPMA** | 5% | 9% | 7% | **0%** | 7% | 6% |

30 kromozom ve 10000 iterasyon ile 1,006 CPU zamanı (sn) ile en iyi uzaklık 10 araç ile 826,556 birim olarak bulunmuştur. En iyi çözüm olan 824,78 birimden %0 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 10,2 olarak hesaplanmış, ortalama süre 1,006 saniyedir. Ortalama toplam uzaklık 873,428 birimdir, en iyi çözümden %6 oranında sapma göstermiştir.

Popülasyon büyüklüğü 30 alınarak 10000 iterasyonda en iyi çözüme ulaşılmıştır. Buna göre bulunan rotalar Tablo 4.16'daki gibidir:

**Tablo 4.16:** GA ile Bulunan Optimum Araç Rotaları.

| ROTALAR | MÜŞTERİ NO |
|---|---|
| **ROTA 1** | 0 5 3 7 8 11 9 6 4 1 2 75 0 |
| **ROTA 2** | 0 13 17 18 19 15 16 14 12 10 0 |
| **ROTA 3** | 0 20 24 25 27 29 30 28 26 23 22 21 0 |
| **ROTA 4** | 0 32 33 31 35 37 38 39 36 34 0 |
| **ROTA 5** | 0 81 78 76 71 70 73 77 79 80 63 0 |
| **ROTA 6** | 0 57 55 54 53 56 58 60 59 0 |
| **ROTA 7** | 0 67 65 62 74 72 61 64 68 66 69 0 |
| **ROTA 8** | 0 43 42 41 40 44 46 45 48 51 50 52 49 47 0 |
| **ROTA 9** | 0 98 96 95 94 92 93 97 100 99 0 |
| **ROTA 10** | 0 90 87 86 83 82 84 85 88 89 91 0 |

#### *4.2.1.3. C205 Problemi için Sayısal Sonuçlar*

C205 probleminin çözümünde 100 müşterinin maksimum 25 araç ile ziyaret edilmesi hedeflenmektedir. Her aracın 700 birimlik kapasitesi bulunmakta olup, araç kapasitesi eşittir. Algoritma Matlab 8.1'de kodlanmış, 8 Gb Ram, I7 2.50 Ghz işlemcili bilgisayarda test edilmiştir.

Bu problem için sezgisel yöntemlerle bulunan en iyi çözüm Rochat ve Taillard, (1995)'in çalışmalarında yer almaktadır. Buna göre en iyi çözüm 3 araç ve 588,88 birimdir.

Farklı popülasyon büyüklüğü ve iterasyon ile literatürde bulunan en iyi değerlere ulaşılması hedeflenmektedir.

C205 probleminin çözümü için ilk olarak başlangıç popülasyon büyüklüğü olarak 25 kromozom alınmış, 1000 iterasyon ile problem 5 kez çalıştırılmıştır.

Tüm problem türlerinde mutasyon oranı %1 olarak alınmıştır. Böylece her rotada toplam uzaklığı azaltmak için bir veya iki müşteri kendi aralarında yer değiştirmektedir.

Çalışmaya ait elde edilen çözümler Tablo 4.17'deki gibidir:

**Tablo 4.17:** 25 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 3 | 3 | 4 | 4 | **3** | 3,4 |
| **TOPLAM MESAFE** | 716,994 | 760,533 | 775,672 | 724,529 | **699,649** | 735,476 |
| **CPU (sn)** | 0,294 | 0,293 | 0,289 | 0,294 | **0,289** | 0,292 |
| **SAPMA** | 22% | 29% | 32% | 23% | **19%** | 25% |

25 kromozom ve 1000 iterasyon ile 0,289 CPU zamanı (sn) ile en iyi uzaklık 3 araç ile 699,649 birim olarak bulunmuştur. En iyi çözüm olan 588,88 birimden %19 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 3,4 olarak hesaplanmış, ortalama süre 0,292 saniyedir. Ortalama toplam uzaklık 735,476 birimdir, en iyi çözümden %25 oranında sapma göstermiştir.

25 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.18'de yer alan sonuçlar bulunmuştur:

**Tablo 4.18:** 25 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 3 | **3** | 3 | 3 | 3 | 3 |
| TOPLAM MESAFE | 696,471 | **675,203** | 716,601 | 700,614 | 693,109 | 696,399 |
| CPU (sn) | 0,306 | **0,295** | 0,295 | 0,301 | 0,291 | 0,298 |
| SAPMA | 18% | **15%** | 22% | 19% | 18% | 18% |

25 kromozom ve 10000 iterasyon ile 0,295 CPU zamanı (sn) ile en iyi uzaklık 3 araç ile 675,203 birim olarak bulunmuştur. En iyi çözüm olan 588,88 birimden %15 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 3 olarak hesaplanmış, ortalama süre 0,298 saniyedir. Ortalama toplam uzaklık 696,399 birimdir, en iyi çözümden %18 oranında sapma göstermiştir.

30 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.19'de yer alan sonuçlar bulunmuştur:

**Tablo 4.19:** 30 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 4 | 4 | **3** | 3 | 3 | 3,4 |
| TOPLAM MESAFE | 735,594 | 755,288 | **724,300** | 738,545 | 754,577 | 741,661 |
| CPU (sn) | 0,353 | 0,357 | **0,366** | 0,362 | 0,354 | 0,358 |
| SAPMA | 25% | 28% | **23%** | 25% | 28% | 26% |

30 kromozom ve 1000 iterasyon ile 0,366 CPU zamanı (sn) ile en iyi uzaklık 3 araç ile 724,300 birim olarak bulunmuştur. En iyi çözüm olan 588,88 birimden %23 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 3 olarak hesaplanmış, ortalama süre 0,38 saniyedir. Ortalama toplam uzaklık 741,661 birimdir, en iyi çözümden %26 oranında sapma göstermiştir.

30 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.20'de yer alan sonuçlar bulunmuştur:

**Tablo 4.20:** 30 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 3 | 3 | 3 | **3** | 3 | 3 |
| **TOPLAM MESAFE** | 709,779 | 707,146 | 709,982 | **706,928** | 716,736 | 710,114 |
| **CPU (sn)** | 0,355 | 0,347 | 0,355 | **0,352** | 0,358 | 0,353 |
| **SAPMA** | 21% | 20% | 21% | **20%** | 22% | 21% |

30 kromozom ve 10000 iterasyon ile 0,352 CPU zamanı (sn) ile en iyi uzaklık 3 araç ile 706,928 birim olarak bulunmuştur. En iyi çözüm olan 588,88 birimden %20 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 3 olarak hesaplanmış, ortalama süre 0,353 saniyedir. Ortalama toplam uzaklık 710,114 birimdir, en iyi çözümden %21 oranında sapma göstermiştir.

50 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.21'de yer alan sonuçlar bulunmuştur:

**Tablo 4.21:** 50 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 3 | 3 | **3** | 3 | 3 | 3 |
| **TOPLAM MESAFE** | 735,011 | 733,670 | **677,01** | 702,492 | 731,025 | 715,842 |
| **CPU (sn)** | 0,589 | 0,5953 | **0,583** | 0,586 | 0,576 | 0,586 |
| **SAPMA** | 25% | 25% | **15%** | 19% | 24% | 22% |

50 kromozom ve 1000 iterasyon ile 0,583 CPU zamanı (sn) ile en iyi uzaklık 3 araç ile 677,01 birim olarak bulunmuştur. En iyi çözüm olan 588,88 birimden %15 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 3 olarak hesaplanmış, ortalama süre 0,586 saniyedir. Ortalama toplam uzaklık 715,842 birimdir, en iyi çözümden %22 oranında sapma göstermiştir.

50 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.22'de yer alan sonuçlar bulunmuştur:

**Tablo 4.22:** 50 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 3 | 3 | 3 | 3 | **3** | 3 |
| **TOPLAM MESAFE** | 692,464 | 716,740 | 725,22 | 704,162 | **666,207** | 700,959 |
| **CPU (sn)** | 0,584 | 0,584 | 0,585 | 0,581 | **0,578** | 0,582 |
| **SAPMA** | 18% | 22% | 23% | 20% | **13%** | 19% |

50 kromozom ve 10000 iterasyon ile 0,578 CPU zamanı (sn) ile en iyi uzaklık 3 araç ile 666,207 birim olarak bulunmuştur. En iyi çözüm olan 588,88 birimden %13 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 3 olarak hesaplanmış, ortalama süre 0,582 saniyedir. Ortalama toplam uzaklık 700,959 birimdir, en iyi çözümden %19 oranında sapma göstermiştir.

100 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.23'te yer alan sonuçlar bulunmuştur:

**Tablo 4.23:** 100 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 3 | 3 | 3 | 3 | **3** | 3 |
| **TOPLAM MESAFE** | 675,203 | 669,99 | 677,369 | 672,105 | **668,974** | 672,728 |
| **CPU (sn)** | 1,1 | 1,078 | 1,09 | 1,098 | **1,767** | 1,227 |
| **SAPMA** | 15% | 14% | 15% | 14% | **14%** | 14% |

100 kromozom ve 1000 iterasyon ile 1,767 CPU zamanı (sn) ile en iyi uzaklık 3 araç ile 668,974 birim olarak bulunmuştur. En iyi çözüm olan 588,88 birimden %13 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 3 olarak hesaplanmış, ortalama süre 1,227 saniyedir. Ortalama toplam uzaklık 672,728 birimdir, en iyi çözümden %14 oranında sapma göstermiştir.

100 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.24'te yer alan sonuçlar bulunmuştur:

**Tablo 4.24:** 100 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 3 | **3** | 3 | 3 | 3 | 3 |
| TOPLAM MESAFE | 674,872 | **655,924** | 667,52 | 659,485 | 662,251 | 664,010 |
| CPU (sn) | 1,08 | **1,073** | 1,74 | 1,075 | 1,71 | 1,336 |
| SAPMA | 15% | **11%** | 13% | 12% | 12% | 13% |

Popülasyon büyüklüğü 100 alınarak 10000 iterasyonda en iyi çözüme ulaşılmıştır. Buna göre bulunan rotalar Tablo 4.25'teki gibidir:

**Tablo 4.25:** GA ile Bulunan Optimum Araç Rotaları.

| ROTALAR | MÜŞTERİ NO |
|---|---|
| ROTA 1 | 0 67 63 62 74 72 61 64 66 69 68 65 49 55 54 53 56 58 60 59 57 40 44 46 45 50 51 52 47 43 42 41 48 21 8 10 11 9 25 0 |
| ROTA 2 | 0 93 5 75 2 1 99 100 97 95 94 92 98 7 3 4 89 91 88 86 84 83 82 85 76 71 70 73 79 80 81 78 77 87 96 90 0 |
| ROTA 3 | 0 20 22 24 27 29 30 6 32 33 31 35 37 38 39 36 34 28 26 23 18 19 15 14 16 12 17 0 |

#### *4.2.1.4. C206 Problemi için Sayısal Sonuçlar*

C206 probleminin çözümünde 100 müşterinin maksimum 25 araç ile ziyaret edilmesi hedeflenmektedir. Her aracın 700 birimlik kapasitesi bulunmakta olup, araç kapasitesi eşittir. Algoritma Matlab 8.1'de kodlanmış, 8 Gb Ram, I7 2.50 Ghz işlemcili bilgisayarda test edilmiştir.

Bu problem için sezgisel yöntemlerle bulunan en iyi çözüm Rochat ve Taillard, (1995)'in çalışmalarında yer almaktadır. Buna göre en iyi çözüm 3 araç ve 588,49 birimdir.

Farklı popülasyon büyüklüğü ve iterasyon ile literatürde bulunan en iyi değerlere ulaşılması hedeflenmektedir.

C206 probleminin çözümü için ilk olarak başlangıç popülasyon büyüklüğü olarak 25 kromozom alınmış, 1000 iterasyon ile problem 5 kez çalıştırılmıştır.

Tüm problem türlerinde mutasyon oranı %1 olarak alınmıştır. Böylece her rotada toplam uzaklığı azaltmak için bir veya iki müşteri kendi aralarında yer değiştirmektedir.

Çalışmaya ait elde edilen çözümler Tablo 4.26'daki gibidir:

**Tablo 4.26:** 25 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 4 | 3 | 4 | **3** | 3 | 3,4 |
| TOPLAM MESAFE | 0,414 | 0,373 | 0,358 | **0,353** | 0,351 | 0,370 |
| CPU (sn) | 789,242 | 784,786 | 856,453 | **781,124** | 821,953 | 806,711 |
| SAPMA | 34% | 33% | 46% | **33%** | 40% | 37% |

25 kromozom ve 1000 iterasyon ile 0,353 CPU zamanı (sn) ile en iyi uzaklık 3 araç ile 781,124 birim olarak bulunmuştur. En iyi çözüm olan 588,49 birimden %33 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 3,4 olarak hesaplanmış, ortalama süre 0,370 saniyedir. Ortalama toplam uzaklık 806,711 birimdir, en iyi çözümden %37 oranında sapma göstermiştir.

25 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.27'de yer alan sonuçlar bulunmuştur:

**Tablo 4.27:** 25 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | **3** | 4 | 3 | 4 | 3 | 3,4 |
| TOPLAM MESAFE | **0,353** | 0,363 | 0,355 | 0,343 | 0,353 | 0,353 |
| CPU (sn) | **733,293** | 791,281 | 781,768 | 784,609 | 808,95 | 779,980 |
| SAPMA | **25%** | 34% | 33% | 33% | 37% | 33% |

25 kromozom ve 10000 iterasyon ile 0,353 CPU zamanı (sn) ile en iyi uzaklık 3 araç ile 733,293 birim olarak bulunmuştur. En iyi çözüm olan 588,49 birimden %25 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 3,4 olarak

hesaplanmış, ortalama süre 0,353 saniyedir. Ortalama toplam uzaklık 779,980 birimdir, en iyi çözümden %33 oranında sapma göstermiştir.

30 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.28'de yer alan sonuçlar bulunmuştur:

**Tablo 4.28:** 30 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 3 | **3** | 3 | 3 | 3 | 3 |
| TOPLAM MESAFE | 0,438 | **0,418** | 0,42 | 0,409 | 0,406 | 0,418 |
| CPU (sn) | 771,849 | **749,509** | 773,182 | 797,691 | 752,411 | 768,928 |
| SAPMA | 31% | **27%** | 31% | 36% | 28% | 31% |

30 kromozom ve 1000 iterasyon ile 0,418 CPU zamanı (sn) ile en iyi uzaklık 3 araç ile 749,509 birim olarak bulunmuştur. En iyi çözüm olan 588,49 birimden %27 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 3 olarak hesaplanmış, ortalama süre 0,418 saniyedir. Ortalama toplam uzaklık 768,928 birimdir, en iyi çözümden %31 oranında sapma göstermiştir.

30 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.29'de yer alan sonuçlar bulunmuştur:

**Tablo 4.29:** 30 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 3 | 3 | **3** | 3 | 3 | 3 |
| TOPLAM MESAFE | 0,415 | 0,42 | **0,418** | 0,421 | 0,42 | 0,419 |
| CPU (sn) | 781,810 | 796,993 | **737,878** | 778,704 | 783,065 | 775,690 |
| SAPMA | 33% | 35% | **25%** | 32% | 33% | 32% |

30 kromozom ve 10000 iterasyon ile 0,418 CPU zamanı (sn) ile en iyi uzaklık 3 araç ile 737,878 birim olarak bulunmuştur. En iyi çözüm olan 588,49 birimden %25 birimlik

sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 3 olarak hesaplanmış, ortalama süre 0,419 saniyedir. Ortalama toplam uzaklık 775,690 birimdir, en iyi çözümden %32 oranında sapma göstermiştir.

50 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.30'de yer alan sonuçlar bulunmuştur:

**Tablo 4.30:** 50 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | **3** | 3 | 3 | 3 | 3 | 3 |
| TOPLAM MESAFE | **0,685** | 2,576 | 0,648 | 0,646 | 0,641 | 1,039 |
| CPU (sn) | **749,294** | 774,835 | 764,033 | 733,293 | 788,578 | 762,007 |
| SAPMA | **27%** | 32% | 30% | 25% | 34% | 29% |

50 kromozom ve 1000 iterasyon ile 0,685 CPU zamanı (sn) ile en iyi uzaklık 3 araç ile 749,294 birim olarak bulunmuştur. En iyi çözüm olan 588,49 birimden %27 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 3 olarak hesaplanmış, ortalama süre 1,039 saniyedir. Ortalama toplam uzaklık 762,007 birimdir, en iyi çözümden %29 oranında sapma göstermiştir.

50 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.31'de yer alan sonuçlar bulunmuştur:

**Tablo 4.31:** 50 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 3 | 3 | 3 | **3** | 3 | 3 |
| TOPLAM MESAFE | 0,644 | 0,649 | 0,643 | **0,65** | 0,641 | 0,645 |
| CPU (sn) | 810,934 | 794,152 | 768,796 | **755,681** | 791,675 | 784,247 |
| SAPMA | 38% | 35% | 31% | **28%** | 35% | 33% |

50 kromozom ve 10000 iterasyon ile 0,65 CPU zamanı (sn) ile en iyi uzaklık 3 araç ile 755,681 birim olarak bulunmuştur. En iyi çözüm olan 588,49 birimden %28 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 3 olarak hesaplanmış, ortalama süre 0,645 saniyedir. Ortalama toplam uzaklık 784,247 birimdir, en iyi çözümden %33 oranında sapma göstermiştir.

100 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.32'de yer alan sonuçlar bulunmuştur:

**Tablo 4.32:** 100 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 3 | 3 | 3 | 3 | **3** | 3 |
| **TOPLAM MESAFE** | 1,27 | 1,31 | 1,27 | 1,33 | **1,517** | 1,339 |
| **CPU (sn)** | 759 | 737,815 | 736,306 | 730,292 | **721,297** | 736,942 |
| **SAPMA** | 29% | 25% | 25% | 24% | **23%** | 25% |

100 kromozom ve 1000 iterasyon ile 1,517 CPU zamanı (sn) ile en iyi uzaklık 3 araç ile 721,297 birim olarak bulunmuştur. En iyi çözüm olan 588,49 birimden %23 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 3 olarak hesaplanmış, ortalama süre 1,339 saniyedir. Ortalama toplam uzaklık 736,942 birimdir, en iyi çözümden %25 oranında sapma göstermiştir.

100 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.33'te yer alan sonuçlar bulunmuştur:

**Tablo 4.33:** 100 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | **3** | 3 | 3 | 3 | 3 | 3 |
| **TOPLAM MESAFE** | **1,347** | 1,27 | 1,41 | 1,3 | 1,29 | 1,323 |
| **CPU (sn)** | **721,691** | 737,878 | 734,252 | 733,2931 | 734,252 | 732,273 |
| **SAPMA** | **23%** | 25% | 25% | 25% | 25% | 24% |

100 kromozom ve 10000 iterasyon ile 1,347 CPU zamanı (sn) ile en iyi uzaklık 3 araç ile 721,691 birim olarak bulunmuştur. En iyi çözüm olan 588,49 birimden %23 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 3 olarak hesaplanmış, ortalama süre 1,323 saniyedir. Ortalama toplam uzaklık 732,273 birimdir, en iyi çözümden %24 oranında sapma göstermiştir.

Popülasyon büyüklüğü 100 alınarak 1000 iterasyonda en iyi çözüme ulaşılmıştır. Buna göre bulunan rotalar Tablo 4.34'deki gibidir:

**Tablo 4.34:** GA ile Bulunan Optimum Araç Rotaları.

| ROTALAR | MÜŞTERİ NO |
|---|---|
| ROTA 1 | 0 60 58 56 53 54 59 57 40 44 46 45 42 41 43 47 50 51 52 48 21 8 10 11 9 25 13 17 15 16 14 12 19 18 23 26 28 36 0 |
| ROTA 2 | 0 93 5 75 2 1 99 100 97 95 94 92 98 7 3 4 89 91 88 86 84 83 82 85 76 71 70 73 79 80 81 78 77 87 96 90 0 |
| ROTA 3 | 0 30 27 24 22 20 29 6 32 33 31 35 37 38 39 34 55 49 65 68 69 66 64 61 72 74 62 63 67 0 |

#### *4.2.1.5. R105 Problemi için Sayısal Sonuçlar*

R105 probleminin çözümünde 100 müşterinin maksimum 25 araç ile ziyaret edilmesi hedeflenmektedir. Her aracın 200 birimlik kapasitesi bulunmakta olup, araç kapasitesi eşittir. Algoritma Matlab 8.1'de kodlanmış, 8 Gb Ram, I7 2.50 Ghz işlemcili bilgisayarda test edilmiştir.

Bu problem için sezgisel yöntemlerle bulunan en iyi çözüm Rochat ve Taillard, (1995)'in çalışmalarında yer almaktadır. Buna göre en iyi çözüm 14 araç ve 1377,11 birimdir.

Farklı popülasyon büyüklüğü ve iterasyon ile literatürde bulunan en iyi değerlere ulaşılması hedeflenmektedir.

R105 probleminin çözümü için ilk olarak başlangıç popülasyon büyüklüğü olarak 25 kromozom alınmış, 1000 iterasyon ile problem 5 kez çalıştırılmıştır.

Tüm problem türlerinde mutasyon oranı %1 olarak alınmıştır. Böylece her rotada toplam uzaklığı azaltmak için bir veya iki müşteri kendi aralarında yer değiştirmektedir.

Çalışmaya ait elde edilen çözümler Tablo 4.35'daki gibidir:

**Tablo 4.35:** 25 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 17 | 17 | **17** | 18 | 17 | 17,20 |
| **CPU (sn)** | 0,604 | 0,584 | **0,576** | 0,559 | 0,593 | 0,58 |
| **TOPLAM MESAFE** | 1.814,5 | 1.842,67 | **1.774,67** | 1.795,53 | 1.804,62 | 1.806,40 |
| **% SAPMA** | 32% | 34% | **29%** | 30% | 31% | 31% |

25 kromozom ve 1000 iterasyon ile 0,576 CPU zamanı (sn) ile en iyi uzaklık 17 araç ile 1.774,67 birim olarak bulunmuştur. En iyi çözüm olan 1.377,11 birimden %29 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 17,2 olarak hesaplanmış, ortalama süre 0,58 saniyedir. Ortalama toplam uzaklık 1.806,40 birimdir, en iyi çözümden %31oranında sapma göstermiştir.

25 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.36'da yer alan sonuçlar bulunmuştur:

**Tablo 4.36:** 25 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 19 | 17 | 15 | 16 | **14** | 16,20 |
| **CPU (sn)** | 0,611 | 0,579 | 0,602 | 0,565 | **0,571** | 0,59 |
| **TOPLAM MESAFE** | 1.423 | 1.415 | 1.392,5 | 1.412,5 | **1.378** | 1.404,20 |
| **% SAPMA** | 3% | 3% | 1% | 3% | **0%** | 2% |

25 kromozom ve 10000 iterasyon ile 0,571 CPU zamanı (sn) ile en iyi uzaklık 14 araç ile 1.378 birim olarak bulunmuştur. En iyi çözüm olan 1.377,11 birimden %0,01 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 16,2 olarak hesaplanmış, ortalama süre 0,59 saniyedir. Ortalama toplam uzaklık 1.404,00 birimdir, en iyi çözümden %2 oranında sapma göstermiştir.

Popülasyon büyüklüğü 25 alınarak 10000 iterasyonda en iyi çözüme ulaşılmıştır. Buna göre bulunan rotalar Tablo 4.37'deki gibidir:

**Tablo 4.37:** GA ile Bulunan Optimum Araç Rotaları.

| ROTALAR | MÜŞTERİ NO |
|---|---|
| **ROTA 1** | 0 2 15 57 87 43 100 91 93 0 |
| **ROTA 2** | 0 31 30 51 81 78 34 35 77 0 |
| **ROTA 3** | 0 42 14 44 38 86 37 97 13 0 |
| **ROTA 4** | 0 27 69 76 40 53 26 0 |
| **ROTA 5** | 0 28 12 29 79 3 50 68 24 80 0 |
| **ROTA 6** | 0 21 73 75 22 41 56 74 58 0 |
| **ROTA 7** | 0 72 39 23 67 55 54 4 25 0 |
| **ROTA 8** | 0 33 65 71 9 66 20 1 0 |
| **ROTA 9** | 0 63 64 11 90 10 32 70 0 |
| **ROTA 10** | 0 47 36 19 49 46 48 0 |
| **ROTA 11** | 0 52 62 88 7 0 |
| **ROTA 12** | 0 95 92 98 99 94 6 96 0 |
| **ROTA 13** | 0 45 83 82 8 18 0 |
| **ROTA 14** | 0 59 5 61 16 85 84 17 60 89 0 |

#### *4.2.1.6. R208 Problemi için Sayısal Sonuçlar*

R208 probleminin çözümünde 100 müşterinin maksimum 25 araç ile ziyaret edilmesi hedeflenmektedir. Her aracın 700 birimlik kapasitesi bulunmakta olup, araç kapasitesi eşittir. Algoritma Matlab 8.1'de kodlanmış, 8 Gb Ram, I7 2.50 Ghz işlemcili bilgisayarda test edilmiştir.

Bu problem için sezgisel yöntemlerle bulunan en iyi çözüm Rochat ve Taillard, (1995)'in çalışmalarında yer almaktadır. Buna göre en iyi çözüm 2 araç ve 726,82 birimdir.

Farklı popülasyon büyüklüğü ve iterasyon ile literatürde bulunan en iyi değerlere ulaşılması hedeflenmektedir.

R208 probleminin çözümü için ilk olarak başlangıç popülasyon büyüklüğü olarak 25 kromozom alınmış, 1000 iterasyon ile problem 5 kez çalıştırılmıştır.

Tüm problem türlerinde mutasyon oranı %1 olarak alınmıştır. Böylece her rotada toplam uzaklığı azaltmak için bir veya iki müşteri kendi aralarında yer değiştirmektedir.

Çalışmaya ait elde edilen çözümler Tablo 4.38'daki gibidir:

**Tablo 4.38:** 25 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 2 | 2 | 2 | **2** | 2 | 2 |
| CPU (sn) | 794,973 | 827,961 | 800,401 | **790,932** | 829,789 | 808,811 |
| TOPLAM MESAFE | 0,53 | 0,536 | 0,542 | **0,538** | 0,552 | 0,540 |
| % SAPMA | 9% | 14% | 10% | **9%** | 14% | 11% |

25 kromozom ve 1000 iterasyon ile 0,538 CPU zamanı (sn) ile en iyi uzaklık 14 araç ile 790,932 birim olarak bulunmuştur. En iyi çözüm olan 726,82 birimden %9 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 2 olarak hesaplanmış, ortalama süre 0,54 saniyedir. Ortalama toplam uzaklık 808,811 birimdir, en iyi çözümden %11 oranında sapma göstermiştir.

25 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.39'da yer alan sonuçlar bulunmuştur:

**Tablo 4.39:** 25 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| Araç sayısı | 2 | 2 | 2 | 2 | **2** | 2 |
| Toplam uzaklık | 796,925 | 816,122 | 825,910 | 797,600 | **778,433** | 802,998 |
| CPU | 0,551 | 0,539 | 0,56 | 0,54 | **0,55** | 0,548 |
| sapma | 10% | 12% | 14% | 10% | **7%** | 10% |

25 kromozom ve 10000 iterasyon ile 0,55 CPU zamanı (sn) ile en iyi uzaklık 2 araç ile 778,433 birim olarak bulunmuştur. En iyi çözüm olan 726,82 birimden %7 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 2 olarak hesaplanmış,

ortalama süre 0,548 saniyedir. Ortalama toplam uzaklık 802,998 birimdir, en iyi çözümden %10 oranında sapma göstermiştir.

30 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.40'da yer alan sonuçlar bulunmuştur:

**Tablo 4.40:** 30 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **Araç sayısı** | **2** | 2 | 2 | 2 | 2 | 2 |
| **Toplam uzaklık** | **797,793** | 814,738 | 802,559 | 817,568 | 825,135 | 811,559 |
| **CPU** | **0,657** | 0,655 | 0,654 | 0,652 | 0,659 | 0,655 |
| **sapma** | **10%** | 12% | 10% | 12% | 14% | 12% |

30 kromozom ve 1000 iterasyon ile 0,657 CPU zamanı (sn) ile en iyi uzaklık 2 araç ile 797,793 birim olarak bulunmuştur. En iyi çözüm olan 726,82 birimden %10 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 2 olarak hesaplanmış, ortalama süre 0,655 saniyedir. Ortalama toplam uzaklık 811,559 birimdir, en iyi çözümden %12 oranında sapma göstermiştir.

30 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.41'da yer alan sonuçlar bulunmuştur:

**Tablo 4.41:** 30 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **Araç sayısı** | 2 | 2 | 2 | 2 | **2** | 2 |
| **Toplam uzaklık** | 779,074 | 819,282 | 790,881 | 811,755 | **773,8** | 794,959 |
| **CPU** | 0,657 | 0,646 | 0,635 | 0,662 | **0,649** | 0,65 |
| **sapma** | 7% | 13% | 9% | 12% | **6%** | 9% |

30 kromozom ve 10000 iterasyon ile 0,649 CPU zamanı (sn) ile en iyi uzaklık 2 araç ile 773,80 birim olarak bulunmuştur. En iyi çözüm olan 726,82 birimden %6 birimlik

sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 2 olarak hesaplanmış, ortalama süre 0,65 saniyedir. Ortalama toplam uzaklık 794,959 birimdir, en iyi çözümden %9 oranında sapma göstermiştir.

Popülasyon büyüklüğü 30 alınarak 10000 iterasyonda en iyi çözüme yaklaşılmıştır. Buna göre bulunan rotalar Tablo 4.42'deki gibidir:

**Tablo 4.42:** GA ile Bulunan Optimum Araç Rotaları.

| ROTALAR | MÜŞTERİ NO |
|---|---|
| ROTA 1 | 0 54 80 68 77 3 79 33 81 9 51 20 30 70 31 88 7 82 48 47 36 49 19 11 62 10 90 32 63 64 46 8 45 17 84 5 60 83 18 52 89 6 94 95 97 92 59 99 96 93 85 91 100 37 98 61 16 44 14 42 87 2 57 15 43 38 86 13 0 |
| ROTA 2 | 0 58 40 53 28 27 69 1 50 76 12 26 21 73 72 74 22 75 56 39 23 67 41 4 55 25 24 29 78 34 35 71 66 65 0 |

### 4.2.2. Gehring & Homberger Problemlerinin Sayısal Sonuçları

Problemlerde kullanılan uygunluk değeri için aşağıdaki kısıtlar göz önüne alınmaktadır:

- Depo son tarih kısıtı: Bu kısıta göre müşteriye seyahat süresi, müşteride harcadığı servis süresi (C tipi problemlerde 90 birim, R tipi problemlerde 10 birim) toplamı aracın depoya gideceği son tarihi geçmemelidir. Araç deponun belirlenen son tarihinden önce görevini yerine getirmelidir.
- Araç kapasite kısıtı: Bu kısıta göre araç depodan çıktıktan sonra uğrayacağı müşterilerin talepleri toplamı araç kapasite kısıtını aşmamalıdır.
- Müşteri zaman aralığı kısıtı: Aracın şimdiki zamanına bir sonraki müşteriye seyahat süresi eklendiğinde bu müşterinin son tarihini geçmemesi gerekmektedir.

Bu kısıtlar altında mevcut rotaya müşteri ataması yapılır, aracın bilgileri (talebin eklenmesi, müşteriden ayrıldığı zamanın başlangıç olarak değişmesi) güncellenir. Rotaya bu müşteri eklenir ve bu müşterinin konumu aracın yeni konumu olarak güncellenir. Bu döngü, ziyaret edilmeyen müşteri kalmayana kadar veya başlangıçta programa tanımlanan araç sayısına ulaşılana kadar devam eder.

Her problem için farklı popülasyon büyüklüğü ve iterasyon sayısı ile çözümlerin iyileştirilmesi amaçlanarak, literatürdeki en iyi değerlere ulaşılması hedeflenmektedir.

#### *4.2.2.1. C1_2_1 Problemi için Sayısal Sonuçlar*

C1_2_1 probleminin çözümünde 200 müşterinin maksimum 50 araç ile ziyaret edilmesi hedeflenmektedir. Her aracın 200 birimlik kapasitesi bulunmakta olup, araç kapasitesi eşittir. Algoritma Matlab 8.1'de kodlanmış, 8 Gb Ram, I7 2.50 Ghz işlemcili bilgisayarda test edilmiştir.

Bu problem için sezgisel yöntemlerle bulunan en iyi çözüm Gehring & Homberger (2001)'in çalışmalarında yer almaktadır. Buna göre en iyi çözüm 20 araç ve 2.704,57 birimdir.

Farklı popülasyon büyüklüğü ve iterasyon ile literatürde bulunan en iyi değerlere ulaşılması hedeflenmektedir.

C1_2_1 probleminin çözümü için ilk olarak başlangıç popülasyon büyüklüğü olarak 40 kromozom alınmış, 1000 iterasyon ile problem 5 kez çalıştırılmıştır.

Tüm problem türlerinde mutasyon oranı %1 olarak alınmıştır. Böylece her rotada toplam uzaklığı azaltmak için bir veya iki müşteri kendi aralarında yer değiştirmektedir.

Çalışmaya ait elde edilen çözümler Tablo 4.43'daki gibidir:

**Tablo 4.43:** 40 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 31 | 27 | 27 | 27 | **27** | 27,80 |
| **CPU (sn)** | 2,33 | 2,126 | 2,35 | 2,056 | **2,086** | 2,19 |
| **TOPLAM MESAFE** | 4.317,86 | 3.720,714 | 3.580,64 | 3.852,784 | **3.500,147** | 3.794,43 |
| **% SAPMA** | 60% | 38% | 32% | 42% | **29%** | 40% |

40 kromozom ve 1000 iterasyon ile 2,086 CPU zamanı (sn) ile en iyi uzaklık 27 araç ile 3.500,147 birim olarak bulunmuştur. En iyi çözüm olan 2.704,57 birimden %29 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 27,80 olarak hesaplanmış, ortalama süre 2,19 saniyedir. Ortalama toplam uzaklık 3.794,43 birimdir, en iyi çözümden %40 oranında sapma göstermiştir.

40 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.44'de yer alan sonuçlar bulunmuştur:

**Tablo 4.44:** 40 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 28 | **27** | 28 | 28 | 29 | 28,00 |
| CPU (sn) | 2,113 | **2,35** | 2,151 | 2,586 | 2,549 | 2,35 |
| TOPLAM MESAFE | 3.991,953 | **3.700,529** | 3.910,606 | 3.943,460 | 4.007,255 | 3.910,76 |
| % SAPMA | 48% | **37%** | 45% | 46% | 48% | 45% |

40 kromozom ve 10000 iterasyon ile 2,35 CPU zamanı (sn) ile en iyi uzaklık 27 araç ile 3.700,529 birim olarak bulunmuştur. En iyi çözüm olan 2.704,57 birimden %37 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 28 olarak hesaplanmış, ortalama süre 2,35 saniyedir. Ortalama toplam uzaklık 3.910,76 birimdir, en iyi çözümden %45 oranında sapma göstermiştir.

50 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.45'te yer alan sonuçlar bulunmuştur:

**Tablo 4.45:** 50 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | **21** | 28 | 27 | 26 | 27 | 25,80 |
| CPU (sn) | **12,787** | 2,636 | 3,301 | 3,074 | 3,224 | 5,00 |
| TOPLAM MESAFE | **2.918,941** | 3.889,707 | 3.590,271 | 3.850,612 | 3.513,633 | 3.552,63 |
| % SAPMA | **8%** | 44% | 33% | 42% | 30% | 31% |

50 kromozom ve 1000 iterasyon ile 12,787 CPU zamanı (sn) ile en iyi uzaklık 21 araç ile 2.918,941 birim olarak bulunmuştur. En iyi çözüm olan 2.704,57 birimden %8 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 25,80 olarak

hesaplanmış, ortalama süre 5 saniyedir. Ortalama toplam uzaklık 3.552,63 birimdir, en iyi çözümden %31 oranında sapma göstermiştir.

50 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.46'da yer alan sonuçlar bulunmuştur:

**Tablo 4.46:** 50 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 27 | 27 | 25 | 27 | **26** | 26,40 |
| **CPU (sn)** | 3,301 | 3,291 | 3,455 | 3,338 | **3,446** | 3,37 |
| **TOPLAM MESAFE** | 3.593,553 | 3.700,34 | 3.606,128 | 3.772,385 | **3.569,373** | 3.648,36 |
| **% SAPMA** | 33% | 37% | 33% | 39% | **32%** | 35% |

50 kromozom ve 10000 iterasyon ile 3,446 CPU zamanı (sn) ile en iyi uzaklık 26 araç ile 3.569,373 birim olarak bulunmuştur. En iyi çözüm olan 2.704,57 birimden %32 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 26,40 olarak hesaplanmış, ortalama süre 5 saniyedir. Ortalama toplam uzaklık 3.648,36 birimdir, en iyi çözümden %35 oranında sapma göstermiştir.

70 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.47'de yer alan sonuçlar bulunmuştur:

**Tablo 4.47:** 70 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 20 | 20 | 21 | 20 | **20** | 20,20 |
| **CPU (sn)** | 4,158 | 4,43 | 4,818 | 1,477 | **3,534** | 3,68 |
| **TOPLAM MESAFE** | 2.900,56 | 2.827,57 | 2.835 | 2.862,99 | **2.716** | 2.828,42 |
| **% SAPMA** | 7% | 5% | 5% | 6% | **0%** | 5% |

70 kromozom ve 1000 iterasyon ile 3,534 CPU zamanı (sn) ile en iyi uzaklık 20 araç ile 2.716 birim olarak bulunmuştur. En iyi çözüm olan 2.704,57 birimden %0,004 birimlik

sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 20,20 olarak hesaplanmış, ortalama süre 5 saniyedir. Ortalama toplam uzaklık 2.828,42 birimdir, en iyi çözümden %5 oranında sapma göstermiştir.

70 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.48'de yer alan sonuçlar bulunmuştur:

**Tablo 4.48:** 70 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 20 | 20 | **20** | 20 | 20 | 20,00 |
| **CPU (sn)** | 4,066 | 5,562 | **2,289** | 4,8 | 3,846 | 4,11 |
| **TOPLAM MESAFE** | 2.940,28 | 2.985,4 | **2.817,94** | 2.877,13 | 2.897,48 | 2.903,65 |
| **% SAPMA** | 9% | 10% | **4%** | 6% | 7% | 7% |

70 kromozom ve 10000 iterasyon ile 2,289 CPU zamanı (sn) ile en iyi uzaklık 20 araç ile 2.817,64 birim olarak bulunmuştur. En iyi çözüm olan 2.704,57 birimden %4 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 20 olarak hesaplanmış, ortalama süre 5 saniyedir. Ortalama toplam uzaklık 2.903,65 birimdir, en iyi çözümden %7 oranında sapma göstermiştir.

Popülasyon büyüklüğü 70 alınarak 1000 iterasyonda en iyi çözüme ulaşılmıştır. Buna göre bulunan rotalar Tablo 4.49'daki gibidir:

**Tablo 4.49:** GA ile Bulunan Optimum Araç Rotaları.

| ROTALAR | MÜŞTERİ NO |
|---|---|
| **ROTA 1** | 0 20 41 85 80 31 25 172 77 110 162 0 |
| **ROTA 2** | 0 21 23 182 75 163 194 145 195 52 92 0 |
| **ROTA 3** | 0 32 171 65 86 115 94 51 174 136 189 0 |
| **ROTA 4** | 0 177 3 88 8 186 127 98 157 137 183 0 |
| **ROTA 5** | 0 161 104 18 54 185 132 7 181 117 49 0 |
| **ROTA 6** | 0 113 155 78 175 13 43 2 90 67 39 107 0 |
| **ROTA 7** | 0 148 103 197 124 141 69 200 0 |
| **ROTA 8** | 0 114 159 38 150 22 151 16 140 187 142 111 63 56 0 |
| **ROTA 9** | 0 170 134 50 156 112 168 79 29 87 42 123 0 |
| **ROTA 10** | 0 190 5 10 193 46 128 106 167 34 95 158 0 |
| **ROTA 11** | 0 93 55 135 58 184 199 37 81 138 0 |
| **ROTA 12** | 0 57 118 83 143 176 36 33 121 165 188 108 0 |
| **ROTA 13** | 0 133 48 26 152 40 153 169 89 105 15 59 198 0 |
| **ROTA 14** | 0 30 120 19 192 196 97 14 96 130 28 74 149 0 |
| **ROTA 15** | 0 101 144 119 166 35 126 71 9 1 99 53 0 |
| **ROTA 16** | 0 164 66 147 160 47 91 70 0 |
| **ROTA 17** | 0 73 116 12 129 11 6 122 139 0 |
| **ROTA 18** | 0 62 131 44 102 146 68 76 0 |
| **ROTA 19** | 0 45 178 27 173 154 24 61 100 64 179 109 0 |
| **ROTA 20** | 0 60 82 180 84 191 125 4 72 17 0 |

#### *4.2.2.2. C1_2_3 Problemi için Sayısal Sonuçlar*

C1_2_3 probleminin çözümünde 200 müşterinin maksimum 50 araç ile ziyaret edilmesi hedeflenmektedir. Her aracın 200 birimlik kapasitesi bulunmakta olup, araç kapasitesi eşittir. Algoritma Matlab 8.1'de kodlanmış, 8 Gb Ram, I7 2.50 Ghz işlemcili bilgisayarda test edilmiştir.

Bu problem için sezgisel yöntemlerle bulunan en iyi çözüm Gehring & Homberger (2001)'in çalışmalarında yer almaktadır. Buna göre en iyi çözüm 18 araç ve 2.707,35 birimdir.

Farklı popülasyon büyüklüğü ve iterasyon ile literatürde bulunan en iyi değerlere ulaşılması hedeflenmektedir.

C1_2_3 probleminin çözümü için ilk olarak başlangıç popülasyon büyüklüğü olarak 40 kromozom alınmış, 1000 iterasyon ile problem 5 kez çalıştırılmıştır.

Tüm problem türlerinde mutasyon oranı %1 olarak alınmıştır. Böylece her rotada toplam uzaklığı azaltmak için bir veya iki müşteri kendi aralarında yer değiştirmektedir.

Çalışmaya ait elde edilen çözümler Tablo 4.50'deki gibidir:

**Tablo 4.50:** 40 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 24 | 22 | 23 | **22** | 23 | 22,80 |
| **CPU (sn)** | 1,868 | 1,819 | 1,724 | **1,926** | 1,815 | 1,83 |
| **TOPLAM MESAFE** | 3.912,086 | 3.724,655 | 3.675,096 | **3.563,286** | 3.857,563 | 3.746,54 |
| **% SAPMA** | 44% | 38% | 36% | **32%** | 42% | 38% |

40 kromozom ve 1000 iterasyon ile 1,926 CPU zamanı (sn) ile en iyi uzaklık 22 araç ile 3.563,286 birim olarak bulunmuştur. En iyi çözüm olan 2.707,35 birimden %32 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 22,80 olarak hesaplanmış, ortalama süre 1,83 saniyedir. Ortalama toplam uzaklık 3.746,54 birimdir, en iyi çözümden %38 oranında sapma göstermiştir.

40 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.51'de yer alan sonuçlar bulunmuştur:

**Tablo 4.51:** 40 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 23 | 21 | **21** | 22 | 20 | 21,40 |
| CPU (sn) | 3,05 | 2,947 | **3,213** | 3,131 | 3,108 | 3,09 |
| TOPLAM MESAFE | 3.842,228 | 3.524,461 | **3.309,334** | 3.366,245 | 3.432,105 | 3.494,87 |
| % SAPMA | 42% | 30% | **22%** | 24% | 27% | 29% |

40 kromozom ve 10000 iterasyon ile 3,213 CPU zamanı (sn) ile en iyi uzaklık 21 araç ile 3.309,334 birim olarak bulunmuştur. En iyi çözüm olan 2.707,35 birimden %22 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 21,40 olarak hesaplanmış, ortalama süre 3,09 saniyedir. Ortalama toplam uzaklık 3.494,87 birimdir, en iyi çözümden %29 oranında sapma göstermiştir.

50 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.52'de yer alan sonuçlar bulunmuştur:

**Tablo 4.52:** 50 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 21 | 22 | **21** | 20 | 23 | 21,40 |
| CPU (sn) | 4,94 | 3,971 | **3,898** | 3,72 | 3,707 | 4,05 |
| TOPLAM MESAFE | 3.503,147 | 3.818,141 | **3.449,108** | 3.521,807 | 3.518,192 | 3.562,08 |
| % SAPMA | 29% | 41% | **27%** | 30% | 30% | 32% |

50 kromozom ve 1000 iterasyon ile 3,898 CPU zamanı (sn) ile en iyi uzaklık 21 araç ile 3.449,108 birim olarak bulunmuştur. En iyi çözüm olan 2.707,35 birimden %27 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 21,40 olarak hesaplanmış, ortalama süre 4, 05 aniyedir. Ortalama toplam uzaklık 3.562,08 birimdir, en iyi çözümden %32 oranında sapma göstermiştir.

50 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.53'de yer alan sonuçlar bulunmuştur:

**Tablo 4.53:** 50 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 23 | 24 | 22 | 22 | **21** | 22,40 |
| **CPU (sn)** | 10,83 | 3,756 | 3,658 | 3,794 | **3,77** | 5,16 |
| **TOPLAM MESAFE** | 3.800,438 | 3.722,882 | 3.470,022 | 3.614,855 | **3.435,242** | 3.608,69 |
| **% SAPMA** | 40% | 38% | 28% | 34% | **27%** | 33% |

50 kromozom ve 1000 iterasyon ile 3,77 CPU zamanı (sn) ile en iyi uzaklık 21 araç ile 3.435,242 birim olarak bulunmuştur. En iyi çözüm olan 2.707,35 birimden %27 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 22,40 olarak hesaplanmış, ortalama süre 5,16 saniyedir. Ortalama toplam uzaklık 3.608,69 birimdir, en iyi çözümden %33 oranında sapma göstermiştir.

70 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.54'te yer alan sonuçlar bulunmuştur:

**Tablo 4.54:** 70 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 22 | 22 | 22 | 21 | **20** | 21,40 |
| **CPU (sn)** | 5,35 | 5,4 | 5,215 | 5,508 | **5,433** | 5,38 |
| **TOPLAM MESAFE** | 3.647,249 | 3.666,011 | 3.546,555 | 3.577,603 | **3.319,045** | 3.551,29 |
| **% SAPMA** | 35% | 35% | 31% | 32% | **23%** | 31% |

70 kromozom ve 1000 iterasyon ile 5,433 CPU zamanı (sn) ile en iyi uzaklık 20 araç ile 3.319,045 birim olarak bulunmuştur. En iyi çözüm olan 2.707,35 birimden %23 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 21,40 olarak hesaplanmış, ortalama süre 5,38 saniyedir. Ortalama toplam uzaklık 3.551,29 birimdir, en iyi çözümden %31 oranında sapma göstermiştir.

70 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.55'te yer alan sonuçlar bulunmuştur:

**Tablo 4.55:** 70 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 20 | 22 | **21** | 21 | 21 | 21,00 |
| CPU (sn) | 5,6 | 5,24 | **5,24** | 5,29 | 5,286 | 5,33 |
| TOPLAM MESAFE | 3.406,313 | 3.457,761 | **3.388,009** | 3.470,267 | 3.470,92 | 3.438,65 |
| % SAPMA | 26% | 28% | **25%** | 28% | 28% | 27% |

70 kromozom ve 10000 iterasyon ile 5,24 CPU zamanı (sn) ile en iyi uzaklık 21 araç ile 3.388,009 birim olarak bulunmuştur. En iyi çözüm olan 2.707,35 birimden %25 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 21 olarak hesaplanmış, ortalama süre 5,33 saniyedir. Ortalama toplam uzaklık 3.438,65 birimdir, en iyi çözümden %27 oranında sapma göstermiştir.

200 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.56'da yer alan sonuçlar bulunmuştur:

**Tablo 4.56:** 200 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 22 | 21 | 22 | **21** | 21 | 21,40 |
| CPU (sn) | 14,72 | 14,827 | 14,46 | **14,47** | 14,44 | 14,58 |
| TOPLAM MESAFE | 3.774,62 | 3.610,250 | 3.594,779 | **3.354,67** | 3.379,348 | 3.542,73 |
| % SAPMA | 39% | 33% | 33% | **24%** | 25% | 31% |

200 kromozom ve 1000 iterasyon ile 14,47 CPU zamanı (sn) ile en iyi uzaklık 21 araç ile 3.354,67 birim olarak bulunmuştur. En iyi çözüm olan 2.707,35 birimden %24 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 21,40 olarak hesaplanmış, ortalama süre 14,58 saniyedir. Ortalama toplam uzaklık 3.542,73 birimdir, en iyi çözümden %31 oranında sapma göstermiştir.

200 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.57'de yer alan sonuçlar bulunmuştur:

**Tablo 4.57:** 200 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 21 | 21 | **21** | 20 | 21 | 20,80 |
| **CPU (sn)** | 14,806 | 14,47 | **14,57** | 14,27 | 14,37 | 14,50 |
| **TOPLAM MESAFE** | 3.429,891 | 3.553,818 | **3.380,685** | 3.508,614 | 3.458,543 | 3.466,31 |
| **% SAPMA** | 27% | 31% | **25%** | 30% | 28% | 28% |

200 kromozom ve 10000 iterasyon ile 14,57 CPU zamanı (sn) ile en iyi uzaklık 21 araç ile 3.380,685 birim olarak bulunmuştur. En iyi çözüm olan 2.707,35 birimden %25 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 20,80 olarak hesaplanmış, ortalama süre 14,50 saniyedir. Ortalama toplam uzaklık 3.466,31 birimdir, en iyi çözümden %28 oranında sapma göstermiştir.

Popülasyon büyüklüğü 40 alınarak 10000 iterasyonda en iyi çözüme ulaşılmıştır. Buna göre bulunan rotalar Tablo 4.58'deki gibidir:

**Tablo 4.58:** GA ile Bulunan Optimum Araç Rotaları.

| ROTALAR | MÜŞTERİ NO |
|---|---|
| **ROTA 1** | 0 177 3 88 8 186 127 98 157 18 54 0 |
| **ROTA 2** | 0 103 197 124 141 69 200 148 189 136 171 51 0 |
| **ROTA 3** | 0 155 78 39 67 175 2 90 17 107 13 43 122 11 0 |
| **ROTA 4** | 0 40 153 169 89 105 198 59 14 96 130 28 74 120 30 0 |
| **ROTA 5** | 0 123 42 50 156 112 168 79 29 87 195 52 0 |
| **ROTA 6** | 0 150 38 159 114 151 16 140 187 142 111 63 56 0 |
| **ROTA 7** | 0 199 37 81 137 183 135 55 184 104 161 117 49 0 |
| **ROTA 8** | 0 9 1 99 144 71 126 166 119 35 53 101 138 0 |
| **ROTA 9** | 0 160 47 66 164 91 12 116 129 0 |
| **ROTA 10** | 0 77 172 25 85 80 31 41 20 24 61 100 64 0 |
| **ROTA 11** | 0 185 132 181 7 58 22 146 102 44 131 0 |
| **ROTA 12** | 0 133 48 26 152 196 97 19 192 68 76 0 |
| **ROTA 13** | 0 57 118 83 143 176 36 33 121 165 188 108 0 |
| **ROTA 14** | 0 147 70 73 6 179 45 109 27 173 154 178 0 |
| **ROTA 15** | 0 84 180 82 4 72 60 125 191 46 193 10 0 |
| **ROTA 16** | 0 182 75 163 194 145 92 23 21 170 134 0 |
| **ROTA 17** | 0 174 94 65 86 115 32 149 15 0 |
| **ROTA 18** | 0 95 34 167 5 106 128 158 190 139 110 162 0 |
| **ROTA 19** | 0 62 0 |
| **ROTA 20** | 0 93 0 |
| **ROTA 21** | 0 113 0 |

#### *4.2.2.3. C1_2_5 Problemi için Sayısal Sonuçlar*

C1_2_5 probleminin çözümünde 200 müşterinin maksimum 50 araç ile ziyaret edilmesi hedeflenmektedir. Her aracın 200 birimlik kapasitesi bulunmakta olup, araç kapasitesi eşittir. Algoritma Matlab 8.1'de kodlanmış, 8 Gb Ram, I7 2.50 Ghz işlemcili bilgisayarda test edilmiştir.

Bu problem için sezgisel yöntemlerle bulunan en iyi çözüm Gehring & Homberger (2001)'in çalışmalarında yer almaktadır. Buna göre en iyi çözüm 20 araç ve 2.702,05 birimdir.

Farklı popülasyon büyüklüğü ve iterasyon ile literatürde bulunan en iyi değerlere ulaşılması hedeflenmektedir.

C1_2_5 probleminin çözümü için ilk olarak başlangıç popülasyon büyüklüğü olarak 40 kromozom alınmış, 1000 iterasyon ile problem 5 kez çalıştırılmıştır.

Tüm problem türlerinde mutasyon oranı %1 olarak alınmıştır. Böylece her rotada toplam uzaklığı azaltmak için bir veya iki müşteri kendi aralarında yer değiştirmektedir.

Çalışmaya ait elde edilen çözümler Tablo 4.59'deki gibidir:

**Tablo 4.59:** 40 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 32 | 32 | 30 | 30 | **28** | 30,4 |
| **CPU (sn)** | 2,683 | 2,637 | 2,766 | 2,805 | **2,72** | 2,722 |
| **TOPLAM MESAFE** | 3.709,08 | 3.358,08 | 3.793,08 | 3.342,32 | **3.110,5** | 3.462,612 |
| **% SAPMA** | 37% | 24% | 40% | 24% | **15%** | 28% |

40 kromozom ve 1000 iterasyon ile 2,72 CPU zamanı (sn) ile en iyi uzaklık 28 araç ile 3.110,5 birim olarak bulunmuştur. En iyi çözüm olan 2.702,05 birimden %15 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 30,4 olarak hesaplanmış, ortalama süre 2,722 saniyedir. Ortalama toplam uzaklık 3.462,612 birimdir, en iyi çözümden %28 oranında sapma göstermiştir.

40 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.60'da yer alan sonuçlar bulunmuştur:

**Tablo 4.60:** 40 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 28 | 31 | **29** | 35 | 33 | 31,2 |
| **CPU (sn)** | 2,46 | 2,354 | **2,269** | 2,37 | 2,351 | 2,361 |
| **TOPLAM MESAFE** | 3.118,52 | 3.718,25 | **3.014,73** | 3.869,287 | 3.653,8 | 3.474,917 |
| **% SAPMA** | 15% | 38% | **12%** | 43% | 35% | 29% |

40 kromozom ve 10000 iterasyon ile 2,269 CPU zamanı (sn) ile en iyi uzaklık 29 araç ile 3.014,73 birim olarak bulunmuştur. En iyi çözüm olan 2.702,05 birimden %12 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 31,2 olarak

hesaplanmış, ortalama süre 2,361 saniyedir. Ortalama toplam uzaklık 3.474,917 birimdir, en iyi çözümden %29 oranında sapma göstermiştir.

50 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.61'da yer alan sonuçlar bulunmuştur:

**Tablo 4.61:** 50 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 26 | 25 | 25 | **22** | 23 | 24,2 |
| CPU (sn) | 2,78 | 7,17 | 3,06 | **3,05** | 3,079 | 3,828 |
| TOPLAM MESAFE | 3.738 | 3.833,88 | 3.700,54 | **2.822,25** | 2.898,56 | 3.398,646 |
| % SAPMA | 38% | 42% | 37% | **4%** | 7% | 26% |

50 kromozom ve 1000 iterasyon ile 3,05 CPU zamanı (sn) ile en iyi uzaklık 22 araç ile 2.822,25 birim olarak bulunmuştur. En iyi çözüm olan 2.702,05 birimden %4 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 24,2 olarak hesaplanmış, ortalama süre 3,828 saniyedir. Ortalama toplam uzaklık 3.398,646 birimdir, en iyi çözümden %26 oranında sapma göstermiştir.

50 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.62'de yer alan sonuçlar bulunmuştur:

**Tablo 4.62:** 50 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 22 | 21 | 22 | 20 | **20** | 21 |
| CPU (sn) | 3,175 | 3,16 | 3,091 | 3,099 | **3,158** | 3,137 |
| TOPLAM MESAFE | 3.123,5 | 3.155,75 | 2.884,03 | 2.880,4 | **2.828,53** | 2.974,442 |
| % SAPMA | 16% | 17% | 7% | 7% | **5%** | 10% |

50 kromozom ve 10000 iterasyon ile 3,158 CPU zamanı (sn) ile en iyi uzaklık 20 araç ile 2.828,53 birim olarak bulunmuştur. En iyi çözüm olan 2.702,05 birimden %5 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 21 olarak

hesaplanmış, ortalama süre 3,137 saniyedir. Ortalama toplam uzaklık 2.974,442 birimdir, en iyi çözümden %10 oranında sapma göstermiştir.

70 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.63'te yer alan sonuçlar bulunmuştur:

**Tablo 4.63:** 70 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 20 | 21 | **20** | 20 | 20 | 20,2 |
| **CPU (sn)** | 3,57 | 3,75 | **3,95** | 4,299 | 4,308 | 3,975 |
| **TOPLAM MESAFE** | 2.876,4 | 2.769,5 | **2.718,25** | 2.874,65 | 2.777,21 | 2.803,202 |
| **% SAPMA** | 6% | 2% | **1%** | 6% | 3% | 4% |

70 kromozom ve 1000 iterasyon ile 3,95 CPU zamanı (sn) ile en iyi uzaklık 20 araç ile 2.718,25 birim olarak bulunmuştur. En iyi çözüm olan 2.702,05 birimden %1 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 20,2 olarak hesaplanmış, ortalama süre 3,975 saniyedir. Ortalama toplam uzaklık 2.803,202 birimdir, en iyi çözümden %4 oranında sapma göstermiştir.

70 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.64'te yer alan sonuçlar bulunmuştur:

**Tablo 4.64:** 70 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 20 | 21 | 20 | **20** | 20 | 20,2 |
| **CPU (sn)** | 4,241 | 4,234 | 4,147 | **4,207** | 4,235 | 4,213 |
| **TOPLAM MESAFE** | 2.825,17 | 2.895,5 | 2.981,63 | **2.705,28** | 2.798,56 | 2.841,228 |
| **% SAPMA** | 5% | 7% | 10% | **0%** | 4% | 5% |

70 kromozom ve 10000 iterasyon ile 4,207 CPU zamanı (sn) ile en iyi uzaklık 20 araç ile 2.705,28 birim olarak bulunmuştur. En iyi çözüm olan 2.702,05 birimden %0,001

birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 20,2 olarak hesaplanmış, ortalama süre 4,213 saniyedir. Ortalama toplam uzaklık 2.841,228 birimdir, en iyi çözümden %5 oranında sapma göstermiştir.

Popülasyon büyüklüğü 70 alınarak 10000 iterasyonda en iyi çözüme ulaşılmıştır. Buna göre bulunan rotalar Tablo 4.65'teki gibidir:

**Tablo 4.65:** GA ile Bulunan Optimum Araç Rotaları.

| ROTALAR | MÜŞTERİ NO |
|---|---|
| ROTA 1 | 0 20 41 85 80 31 25 172 77 110 162 0 |
| ROTA 2 | 0 21 23 182 75 163 194 145 195 52 92 0 |
| ROTA 3 | 0 32 171 65 86 115 94 51 174 136 189 0 |
| ROTA 4 | 0 30 120 19 192 196 97 14 96 130 28 74 149 0 |
| ROTA 5 | 0 161 104 18 54 185 132 7 181 117 49 183 0 |
| ROTA 6 | 0 113 155 78 175 13 43 2 90 67 39 107 0 |
| ROTA 7 | 0 133 48 26 152 40 153 169 89 105 15 59 198 0 |
| ROTA 8 | 0 114 159 38 150 22 151 16 140 187 142 111 63 56 0 |
| ROTA 9 | 0 170 134 50 156 112 168 79 29 87 42 123 0 |
| ROTA 10 | 0 190 5 10 193 46 128 106 167 34 95 158 0 |
| ROTA 11 | 0 93 55 135 58 184 199 37 81 137 0 |
| ROTA 12 | 0 57 118 83 143 176 36 33 121 165 188 108 0 |
| ROTA 13 | 0 164 66 147 160 47 91 70 0 |
| ROTA 14 | 0 101 144 119 166 35 126 71 9 1 99 53 0 |
| ROTA 15 | 0 62 131 44 102 146 68 76 0 |
| ROTA 16 | 0 73 116 12 129 11 6 122 139 0 |
| ROTA 17 | 0 45 178 27 173 154 24 61 100 64 179 109 0 |
| ROTA 18 | 0 177 3 88 8 186 127 98 157 138 0 |
| ROTA 19 | 0 60 82 180 84 191 125 4 72 17 0 |
| ROTA 20 | 0 148 103 197 124 141 69 200 0 |

#### *4.2.2.4. C2_2_6 Problemi için Sayısal Sonuçlar*

C2_2_6 probleminin çözümünde 200 müşterinin maksimum 50 araç ile ziyaret edilmesi hedeflenmektedir. Her aracın 700 birimlik kapasitesi bulunmakta olup, araç kapasitesi eşittir. Algoritma Matlab 8.1'de kodlanmış, 8 Gb Ram, I7 2.50 Ghz işlemcili bilgisayarda test edilmiştir.

Bu problem için sezgisel yöntemlerle bulunan en iyi çözüm Bräysy (2001)'in çalışmasında yer almaktadır. Buna göre en iyi çözüm 6 araç ve 1.857,35 birimdir.

Farklı popülasyon büyüklüğü ve iterasyon ile literatürde bulunan en iyi değerlere ulaşılması hedeflenmektedir.

C2_2_6 probleminin çözümü için ilk olarak başlangıç popülasyon büyüklüğü olarak 40 kromozom alınmış, 1000 iterasyon ile problem 5 kez çalıştırılmıştır.

Tüm problem türlerinde mutasyon oranı %1 olarak alınmıştır. Böylece her rotada toplam uzaklığı azaltmak için bir veya iki müşteri kendi aralarında yer değiştirmektedir.

Çalışmaya ait elde edilen çözümler Tablo 4.66'daki gibidir:

**Tablo 4.66:** 40 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 6 | 6 | **6** | 6 | 6 | 6 |
| **CPU (sn)** | 1,397 | 1,342 | **1,364** | 1,346 | 1,363 | 1,362 |
| **TOPLAM MESAFE** | 2.383,468 | 2.555,625 | **2.382,806** | 2.468,992 | 2.491,262 | 2.456,431 |
| **% SAPMA** | 28% | 38% | **28%** | 33% | 34% | 32% |

40 kromozom ve 1000 iterasyon ile 1,364 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.382,806 birim olarak bulunmuştur. En iyi çözüm olan 1.857,35 birimden %28 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 1,362 saniyedir. Ortalama toplam uzaklık 2.456,431 birimdir, en iyi çözümden %32 oranında sapma göstermiştir.

40 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.67'de yer alan sonuçlar bulunmuştur:

**Tablo 4.67:** 40 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 6 | **6** | 6 | 7 | 6 | 6,2 |
| CPU (sn) | 1,356 | **1,348** | 1,355 | 1,347 | 1,362 | 1,354 |
| TOPLAM MESAFE | 2.519,575 | **2.373,537** | 2.532,260 | 2.548,179 | 2.466,333 | 2.487,977 |
| % SAPMA | 36% | **28%** | 36% | 37% | 33% | 34% |

40 kromozom ve 10000 iterasyon ile 1,348 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.373,537 birim olarak bulunmuştur. En iyi çözüm olan 1.857,35 birimden %28 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6,20 olarak hesaplanmış, ortalama süre 1,354 saniyedir. Ortalama toplam uzaklık 2.487,977 birimdir, en iyi çözümden %34 oranında sapma göstermiştir.

50 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.68'de yer alan sonuçlar bulunmuştur:

**Tablo 4.68:** 50 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 6 | 6 | 6 | **6** | 7 | 6,2 |
| CPU (sn) | 1,713 | 1,691 | 1,694 | **1,707** | 1,681 | 1,697 |
| TOPLAM MESAFE | 2.462,620 | 2.554,842 | 2.429,280 | **2.393,638** | 2.414,697 | 2.451,016 |
| % SAPMA | 33% | 38% | 31% | **29%** | 30% | 32% |

50 kromozom ve 1000 iterasyon ile 1,707 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.393,638 birim olarak bulunmuştur. En iyi çözüm olan 1.857,35 birimden %29 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6,20 olarak hesaplanmış, ortalama süre 1,697 saniyedir. Ortalama toplam uzaklık 2.451,016 birimdir, en iyi çözümden %32 oranında sapma göstermiştir.

50 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.69'da yer alan sonuçlar bulunmuştur:

**Tablo 4.69:** 50 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 6 | 6 | 6 | **6** | 6 | 6 |
| CPU (sn) | 1,686 | 1,685 | 1,695 | **1,712** | 1,706 | 1,697 |
| TOPLAM MESAFE | 2.518,169 | 2.525,605 | 2.499,446 | **2.385,209** | 2.493,806 | 2.484,447 |
| % SAPMA | 36% | 36% | 35% | **28%** | 34% | 34% |

50 kromozom ve 10000 iterasyon ile 1,712 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.385,209 birim olarak bulunmuştur. En iyi çözüm olan 1.857,35 birimden %28 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 1,697 saniyedir. Ortalama toplam uzaklık 2.484,447 birimdir, en iyi çözümden %34 oranında sapma göstermiştir.

70 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.70'da yer alan sonuçlar bulunmuştur:

**Tablo 4.70:** 70 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 6 | 6 | 6 | **6** | 6 | 6 |
| CPU (sn) | 2,328 | 2,353 | 2,367 | **2,355** | 2,351 | 2,351 |
| TOPLAM MESAFE | 2.109,123 | 1.937,866 | 1.931,484 | **1.892,773** | 1.903,394 | 1.954,928 |
| % SAPMA | 14% | 4% | 4% | **2%** | 2% | 5% |

70 kromozom ve 1000 iterasyon ile 2,355 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 1.892,773 birim olarak bulunmuştur. En iyi çözüm olan 1.857,35 birimden %2 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 2,351 saniyedir. Ortalama toplam uzaklık 1.954,928 birimdir, en iyi çözümden %5 oranında sapma göstermiştir.

70 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.71'de yer alan sonuçlar bulunmuştur:

**Tablo 4.71:** 70 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 6 | 6 | 6 | 6 | **6** | 6 |
| CPU (sn) | 2,35 | 2,373 | 2,355 | 2,377 | **2,39** | 2,369 |
| TOPLAM MESAFE | 2.432,901 | 1.900,5 | 1.890,36 | 1.915,3 | **1.860,88** | 1.999,988 |
| % SAPMA | 31% | 2% | 2% | 3% | **0%** | 8% |

70 kromozom ve 10000 iterasyon ile 2,39 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 1.860,88 birim olarak bulunmuştur. En iyi çözüm olan 1.857,35 birimden %0,2 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 2,369 saniyedir. Ortalama toplam uzaklık 1.999,988 birimdir, en iyi çözümden %8 oranında sapma göstermiştir.

Popülasyon büyüklüğü 70 alınarak 10000 iterasyonda en iyi çözüme ulaşılmıştır. Buna göre bulunan rotalar Tablo 4.72'deki gibidir:

**Tablo 4.72:** GA ile Bulunan Optimum Araç Rotaları.

| ROTALAR | MÜŞTERİ NO |
|---|---|
| ROTA 1 | 0 28 188 196 142 107 200 41 65 183 35 116 15 3 199 189 47 175 155 179 89 9 19 186 146 161 197 54 147 195 156 79 4 23 11 0 |
| ROTA 2 | 0 144 42 6 110 184 148 29 2 138 143 128 126 25 90 48 145 127 20 193 71 5 194 190 80 158 101 91 106 159 44 83 14 36 0 |
| ROTA 3 | 0 178 167 40 39 119 38 166 102 98 76 8 78 22 134 177 43 191 150 151 97 31 69 181 172 75 163 87 168 154 21 51 0 |
| ROTA 4 | 0 104 64 198 164 137 174 26 140 115 18 57 131 113 169 192 84 27 53 56 185 180 37 123 121 124 55 85 88 135 1 111 157 34 176 114 68 0 |
| ROTA 5 | 0 136 67 132 10 153 59 86 93 118 182 45 77 33 13 12 73 32 50 96 139 103 141 74 100 81 130 112 120 125 109 149 152 95 129 52 105 0 |
| ROTA 6 | 0 171 63 173 94 162 99 70 46 187 60 30 108 133 16 61 72 62 122 24 92 49 160 117 82 17 7 170 165 58 66 0 |

#### *4.2.2.5. C2_2_8 Problemi için Sayısal Sonuçlar*

C2_2_8 probleminin çözümünde 200 müşterinin maksimum 50 araç ile ziyaret edilmesi hedeflenmektedir. Her aracın 700 birimlik kapasitesi bulunmakta olup, araç kapasitesi

eşittir. Algoritma Matlab 8.1'de kodlanmış, 8 Gb Ram, I7 2.50 Ghz işlemcili bilgisayarda test edilmiştir.

Bu problem için sezgisel yöntemlerle bulunan en iyi çözüm Ropke & Psinger (2005)'in çalışmalarında yer almaktadır. Buna göre en iyi çözüm 6 araç ve 1.820,53 birimdir.

Farklı popülasyon büyüklüğü ve iterasyon ile literatürde bulunan en iyi değerlere ulaşılması hedeflenmektedir.

C2_2_8 probleminin çözümü için ilk olarak başlangıç popülasyon büyüklüğü olarak 40 kromozom alınmış, 1000 iterasyon ile problem 5 kez çalıştırılmıştır.

Tüm problem türlerinde mutasyon oranı %1 olarak alınmıştır. Böylece her rotada toplam uzaklığı azaltmak için bir veya iki müşteri kendi aralarında yer değiştirmektedir.

Çalışmaya ait elde edilen çözümler Tablo 4.73'teki gibidir:

**Tablo 4.73:** 40 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 7 | **6** | 8 | 6 | 7 | 6,8 |
| CPU (sn) | 1,487 | **1,384** | 1,392 | 1,383 | 1,384 | 1,406 |
| TOPLAM MESAFE | 2.429,921 | **2.366,209** | 2.544,805 | 2.420,442 | 2.405,017 | 2.433,279 |
| % SAPMA | 33% | **30%** | 40% | 33% | 32% | 34% |

40 kromozom ve 1000 iterasyon ile 1,384 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.366,209 birim olarak bulunmuştur. En iyi çözüm olan 1.820,53 birimden %30 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6,8 olarak hesaplanmış, ortalama süre 1,406 saniyedir. Ortalama toplam uzaklık 2.433,279 birimdir, en iyi çözümden %34 oranında sapma göstermiştir.

40 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.74'te yer alan sonuçlar bulunmuştur:

**Tablo 4.74:** 40 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 7 | **6** | 6 | 6 | 6 | 6,2 |
| CPU (sn) | 1,39 | **1,397** | 1,393 | 1,393 | 1,371 | 1,389 |
| TOPLAM MESAFE | 2.545,455 | **2.420,400** | 2.474,233 | 2.423,023 | 2.483,177 | 2.469,258 |
| % SAPMA | 40% | **33%** | 36% | 33% | 36% | 36% |

40 kromozom ve 10000 iterasyon ile 1,397 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.420,400 birim olarak bulunmuştur. En iyi çözüm olan 1.820,53 birimden %33 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6,2 olarak hesaplanmış, ortalama süre 1,389 saniyedir. Ortalama toplam uzaklık 2.469,258 birimdir, en iyi çözümden %36 oranında sapma göstermiştir.

50 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.75'te yer alan sonuçlar bulunmuştur:

**Tablo 4.75:** 50 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 7 | 6 | 6 | **6** | 6 | 6,2 |
| CPU (sn) | 7,028 | 1,772 | 1,759 | **1,792** | 1,799 | 2,83 |
| TOPLAM MESAFE | 2.452,647 | 2.501,187 | 2.511,901 | **2.448,895** | 2.488,359 | 2.480,598 |
| % SAPMA | 35% | 37% | 38% | **35%** | 37% | 36% |

50 kromozom ve 1000 iterasyon ile 1,792 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.448,895 birim olarak bulunmuştur. En iyi çözüm olan 1.820,53 birimden %35 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6,2 olarak hesaplanmış, ortalama süre 2,83 saniyedir. Ortalama toplam uzaklık 2.480,598 birimdir, en iyi çözümden %36 oranında sapma göstermiştir.

50 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.76'da yer alan sonuçlar bulunmuştur:

**Tablo 4.76:** 50 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 6 | 6 | **6** | 6 | 6 | 6 |
| CPU (sn) | 1,779 | 1,769 | **1,813** | 1,798 | 1,78 | 1,788 |
| TOPLAM MESAFE | 2.473,233 | 2.433,573 | **2.391,269** | 2.448,185 | 2.490,497 | 2.447,351 |
| % SAPMA | 36% | 34% | **31%** | 34% | 37% | 34% |

50 kromozom ve 10000 iterasyon ile 1,813 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.391,269 birim olarak bulunmuştur. En iyi çözüm olan 1.820,53 birimden %31 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 1,788 saniyedir. Ortalama toplam uzaklık 2.447,351 birimdir, en iyi çözümden %34 oranında sapma göstermiştir.

70 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.77'de yer alan sonuçlar bulunmuştur:

**Tablo 4.77:** 70 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 6 | **6** | 6 | 6 | 6 | 6 |
| CPU (sn) | 2,455 | **2,485** | 2,481 | 2,483 | 2,5 | 2,480 |
| TOPLAM MESAFE | 2.441,255 | **2.345,182** | 2.450,066 | 2.439,812 | 2.388,250 | 2.412,913 |
| % SAPMA | 34% | **29%** | 35% | 34% | 31% | 33% |

70 kromozom ve 1000 iterasyon ile 2,485 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.345,182 birim olarak bulunmuştur. En iyi çözüm olan 1.820,53 birimden %29 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 2,480 saniyedir. Ortalama toplam uzaklık 2.412,913 birimdir, en iyi çözümden %33 oranında sapma göstermiştir.

70 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.78'de yer alan sonuçlar bulunmuştur:

**Tablo 4.78:** 70 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 6 | **6** | 6 | 6 | 6 | 6 |
| CPU (sn) | 2,593 | **2,453** | 2,519 | 2,52 | 2,478 | 2,513 |
| TOPLAM MESAFE | 2.402,680 | **2.373,303** | 2.457,743 | 2.441,986 | 2.448,284 | 2.424,799 |
| % SAPMA | 32% | **30%** | 35% | 34% | 34% | 33% |

70 kromozom ve 10000 iterasyon ile 2,453 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.373,303 birim olarak bulunmuştur. En iyi çözüm olan 1.820,53 birimden %30 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 2,513 saniyedir. Ortalama toplam uzaklık 2.424,799 birimdir, en iyi çözümden %33 oranında sapma göstermiştir.

200 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.79'de yer alan sonuçlar bulunmuştur:

**Tablo 4.79:** 200 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 6 | **6** | 6 | 6 | 6 | 6 |
| CPU (sn) | 5,047 | **5,17** | 5,84 | 5,98 | 5,4 | 5,487 |
| TOPLAM MESAFE | 2.381,414 | **2.290,170** | 2.334, 433 | 2.336,940 | 2.483,622 | 2.365,315 |
| % SAPMA | 31% | **26%** | 28% | 28% | 36% | 30% |

200 kromozom ve 1000 iterasyon ile 5,17 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.290,170 birim olarak bulunmuştur. En iyi çözüm olan 1.820,53 birimden %26 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 5,487 saniyedir. Ortalama toplam uzaklık 2.365,315 birimdir, en iyi çözümden %30 oranında sapma göstermiştir.

200 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.80'de yer alan sonuçlar bulunmuştur:

**Tablo 4.80:** 200 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 6 | 6 | 6 | **6** | 6 | 6 |
| CPU (sn) | 6,953 | 6,88 | 6,71 | **6,89** | 6,91 | 6,869 |
| TOPLAM MESAFE | 2.376,974 | 2.368,211 | 2.296,906 | **2.291,935** | 2.382,550 | 2.343,315 |
| % SAPMA | 31% | 30% | 26% | **26%** | 31% | 29% |

200 kromozom ve 10000 iterasyon ile 6,89 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.291,935 birim olarak bulunmuştur. En iyi çözüm olan 1.820,53 birimden %26 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 6,869 saniyedir. Ortalama toplam uzaklık 2.343,315 birimdir, en iyi çözümden %29 oranında sapma göstermiştir.

Popülasyon büyüklüğü 200 ve 1000 iterasyon ile en iyi çözüme yakınsanmıştır. Buna göre bulunan rotalar 4.81'deki gibidir:

**Tablo 4.81:** GA ile Bulunan Optimum Araç Rotaları.

| ROTALAR | MÜŞTERİ NO |
|---|---|
| ROTA 1 | 0 83 14 159 106 91 101 158 80 190 194 5 71 127 25 90 126 128 143 2 138 29 148 110 6 184 144 42 132 67 10 153 59 33 13 0 |
| ROTA 2 | 0 99 70 162 94 173 60 30 108 133 16 61 72 24 122 62 19 9 89 179 155 175 15 116 35 107 142 196 188 136 28 171 63 178 0 |
| ROTA 3 | 0 96 139 103 141 74 100 81 130 112 120 125 109 152 149 95 129 68 114 176 34 157 111 1 135 85 55 37 180 27 53 84 192 169 0 |
| ROTA 4 | 0 54 147 23 11 4 79 156 195 44 197 161 146 186 20 193 145 48 47 189 199 3 183 65 41 200 167 40 119 38 166 78 22 134 191 150 43 151 0 |
| ROTA 5 | 0 51 21 154 168 163 87 75 172 181 69 97 31 88 17 7 82 117 165 170 58 66 105 52 36 0 |
| ROTA 6 | 0 46 39 102 98 76 8 177 187 73 32 50 12 77 182 45 86 93 118 104 198 164 113 131 185 56 123 121 124 49 160 92 0 |

#### *4.2.2.6. C2_2_10 Problemi için Sayısal Sonuçlar*

C2_2_10 probleminin çözümünde 200 müşterinin maksimum 50 araç ile ziyaret edilmesi hedeflenmektedir. Her aracın 700 birimlik kapasitesi bulunmakta olup, araç

kapasitesi eşittir. Algoritma Matlab 8.1'de kodlanmış, 8 Gb Ram, I7 2.50 Ghz işlemcili bilgisayarda test edilmiştir.

Bu problem için sezgisel yöntemlerle bulunan en iyi çözüm Nagata, ve diğ. (2010)'in çalışmasında yer almaktadır. Buna göre en iyi çözüm 6 araç ve 1.806,58 birimdir.

Farklı popülasyon büyüklüğü ve iterasyon ile literatürde bulunan en iyi değerlere ulaşılması hedeflenmektedir.

C2_2_10 probleminin çözümü için ilk olarak başlangıç popülasyon büyüklüğü olarak 40 kromozom alınmış, 1000 iterasyon ile problem 5 kez çalıştırılmıştır.

Tüm problem türlerinde mutasyon oranı %1 olarak alınmıştır. Böylece her rotada toplam uzaklığı azaltmak için bir veya iki müşteri kendi aralarında yer değiştirmektedir.

Çalışmaya ait elde edilen çözümler Tablo 4.82'deki gibidir:

**Tablo 4.82:** 40 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | **6** | 6 | 6 | 6 | 6 | 6 |
| **CPU (sn)** | **1,743** | 1,591 | 1,61 | 1,598 | 1,606 | 1,63 |
| **TOPLAM MESAFE** | **2.303,961** | 2.375,196 | 2.344,584 | 2.325,860 | 2.360,509 | 2.342,022 |
| **% SAPMA** | **28%** | 31% | 30% | 29% | 31% | 30% |

40 kromozom ve 1000 iterasyon ile 1,743 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.303,961 birim olarak bulunmuştur. En iyi çözüm olan 1.806,58 birimden %28 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 1,63 saniyedir. Ortalama toplam uzaklık 2.342,022 birimdir, en iyi çözümden %30 oranında sapma göstermiştir.

40 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.83'te yer alan sonuçlar bulunmuştur:

**Tablo 4.83:** 40 Kromozom 10000 İterasyon ile Elde Edilen Çözümler.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | **6** | 6 | 6 | 6 | 6 | 6 |
| CPU (sn) | **1,592** | 1,605 | 1,611 | 1,586 | 1,6 | 1,599 |
| TOPLAM MESAFE | **2.299,642** | 2.314,758 | 2.360,142 | 2.336,325 | 2.342,871 | 2.330,747 |
| % SAPMA | **27%** | 28% | 31% | 29% | 30% | 29% |

40 kromozom ve 10000 iterasyon ile 1,592 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.299,642 birim olarak bulunmuştur. En iyi çözüm olan 1.806,58 birimden %27 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 1,599 saniyedir. Ortalama toplam uzaklık 2.330,747 birimdir, en iyi çözümden %29 oranında sapma göstermiştir.

50 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.84'te yer alan sonuçlar bulunmuştur:

**Tablo 4.84:** 50 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 6 | 6 | **6** | 6 | 6 | 6 |
| CPU (sn) | 1,976 | 1,976 | **1,973** | 1,989 | 1,96 | 1,975 |
| TOPLAM MESAFE | 2.250,379 | 2.335,812 | **2.246,714** | 2.354,437 | 2.279,651 | 2.293,398 |
| % SAPMA | 25% | 29% | **24%** | 30% | 26% | 27% |

50 kromozom ve 1000 iterasyon ile 1,973 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.246,714 birim olarak bulunmuştur. En iyi çözüm olan 1.806,58 birimden %24 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 1,975 saniyedir. Ortalama toplam uzaklık 2.293,398 birimdir, en iyi çözümden %27 oranında sapma göstermiştir.

50 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.85'te yer alan sonuçlar bulunmuştur:

**Tablo 4.85:** 50 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 6 | **6** | 6 | 6 | 6 | 6 |
| CPU (sn) | 1,981 | **1,998** | 1,982 | 1,993 | 1,976 | 1,986 |
| TOPLAM MESAFE | 2.378,453 | **2.269,606** | 2.274,561 | 2.311,542 | 2.302,388 | 2.307,310 |
| % SAPMA | 32% | **26%** | 26% | 28% | 27% | 28% |

50 kromozom ve 10000 iterasyon ile 1,998 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.269,606 birim olarak bulunmuştur. En iyi çözüm olan 1.806,58 birimden %26 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 1,986 saniyedir. Ortalama toplam uzaklık 2.307,310 birimdir, en iyi çözümden %28 oranında sapma göstermiştir.

70 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.86'da yer alan sonuçlar bulunmuştur:

**Tablo 4.86:** 70 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 6 | 6 | 6 | **6** | 6 | 6 |
| CPU (sn) | 2,737 | 2,753 | 2,744 | **2,768** | 2,733 | 2,747 |
| TOPLAM MESAFE | 2.333,701 | 2.271,029 | 2.300,631 | **2.246,486** | 2.292,901 | 2.288,95 |
| % SAPMA | 29% | 26% | 27% | **24%** | 27% | 27% |

70 kromozom ve 1000 iterasyon ile 2,768 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.246,486 birim olarak bulunmuştur. En iyi çözüm olan 1.806,58 birimden %24 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 2,747 saniyedir. Ortalama toplam uzaklık 2.288,95 birimdir, en iyi çözümden %27 oranında sapma göstermiştir.

70 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.87'de yer alan sonuçlar bulunmuştur:

**Tablo 4.87:** 70 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 6 | 6 | 6 | **6** | 6 | 6 |
| CPU (sn) | 2,751 | 2,774 | 2,74 | **2,755** | 2,76 | 2,756 |
| TOPLAM MESAFE | 2.375,941 | 2.389,671 | 2.320,274 | **2.174,289** | 2.228,346 | 2.297,704 |
| % SAPMA | 32% | 32% | 28% | **20%** | 23% | 27% |

70 kromozom ve 10000 iterasyon ile 2,755 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.174,289 birim olarak bulunmuştur. En iyi çözüm olan 1.806,58 birimden %20 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 2,756 saniyedir. Ortalama toplam uzaklık 2.297,704 birimdir, en iyi çözümden %27 oranında sapma göstermiştir.

200 kromozom ve 1000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.88'de yer alan sonuçlar bulunmuştur:

**Tablo 4.88:** 200 Kromozom 1000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| ARAÇ SAYISI | 6 | 6 | 6 | **6** | 6 | 6 |
| CPU (sn) | 7,375 | 7,41 | 7,486 | **7,461** | 7,44 | 7,434 |
| TOPLAM MESAFE | 2.199,214 | 2.303,01 | 2.303,717 | **2.190,728** | 2.237,396 | 2.246,813 |
| % SAPMA | 22% | 27% | 28% | **21%** | 24% | 24% |

200 kromozom ve 1000 iterasyon ile 7,461 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.190,728 birim olarak bulunmuştur. En iyi çözüm olan 1.806,58 birimden %21 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 7,434 saniyedir. Ortalama toplam uzaklık 2.246,813 birimdir, en iyi çözümden %24 oranında sapma göstermiştir.

200 kromozom ve 10000 iterasyon ile problem 5 defa çalıştırılmış, Tablo 4.89'da yer alan sonuçlar bulunmuştur:

**Tablo 4.89:** 200 Kromozom 10000 İterasyon ile Elde Edilen Sonuçlar.

| | 1. deneme | 2. deneme | 3. deneme | 4. deneme | 5. deneme | ORTALAMA |
|---|---|---|---|---|---|---|
| **ARAÇ SAYISI** | 6 | 6 | 6 | 6 | **6** | 6 |
| **CPU (sn)** | 7,373 | 7,74 | 7,67 | 7,82 | **7,58** | 7,637 |
| **TOPLAM MESAFE** | 2.210,843 | 2.218,213 | 2.214,288 | 2.201,163 | **2.151,693** | 2.199,240 |
| **% SAPMA** | 22% | 23% | 23% | 22% | **19%** | 22% |

200 kromozom ve 10000 iterasyon ile 7,58 CPU zamanı (sn) ile en iyi uzaklık 6 araç ile 2.151,693 birim olarak bulunmuştur. En iyi çözüm olan 1.806,58 birimden %19 birimlik sapma göstermektedir. 5 deneme sonucunda ortalama araç sayısı 6 olarak hesaplanmış, ortalama süre 7,637 saniyedir. Ortalama toplam uzaklık 2.199,240 birimdir, en iyi çözümden %22 oranında sapma göstermiştir.

Popülasyon büyüklüğü 200 alınarak 10000 iterasyonda en iyi çözüme ulaşılmıştır. Buna göre bulunan rotalar Tablo 4.90'daki gibidir:

**Tablo 4.90:** GA ile Bulunan Optimum Araç Rotaları.

| ROTALAR | MÜŞTERİ NO |
|---|---|
| **ROTA 1** | 0 96 139 103 141 74 100 81 130 112 120 125 109 152 149 95 129 68 117 82 7 170 165 58 66 105 52 160 49 92 24 72 61 108 133 16 30 173 94 0 |
| **ROTA 2** | 0 5 71 127 145 48 25 90 126 128 143 2 138 29 148 110 6 184 144 42 132 67 12 13 33 86 93 118 77 182 45 59 153 10 32 73 50 0 |
| **ROTA 3** | 0 15 116 107 35 65 183 3 199 189 47 175 9 89 179 155 147 54 19 146 83 14 159 106 91 101 158 80 190 194 20 193 161 197 44 186 62 122 0 |
| **ROTA 4** | 0 114 176 34 157 111 121 123 124 1 135 85 55 37 180 27 53 84 192 169 57 164 198 64 137 174 115 26 140 18 113 131 104 187 60 162 70 99 63 171 136 0 |
| **ROTA 5** | 0 156 79 195 4 11 23 36 51 21 154 168 163 87 43 150 191 22 134 78 8 102 98 76 151 97 69 181 172 75 31 88 17 185 56 0 |
| **ROTA 6** | 0 28 188 196 142 200 41 177 40 167 119 38 166 39 46 178 0 |

# 5. TARTIŞMA VE SONUÇ

Bu çalışma kapsamında ARP kavramı ve özel bir türü olan zaman pencereli araç rotalama problemi için kullanılan yöntemler detaylı bir şekilde araştırılarak, genetik algoritmalar ile literatürde daha önce yer alan test problemleri çözülmüştür. Genetik algoritmalar diğer problemlerde olduğu gibi ZPARP'ın çözümünde çözüm kümesiyle aramaya başladığı için olurlu çözüme daha hızlı yakınsamaktadır. Kesin çözüm yöntemleri ZPARP'ın çözümü için daha fazla çözüm süresi kullandığından, bu çalışmanın amacı ZPARP için daha kısa sürede çözüme yaklaştıran popülasyon tabanlı sezgisel oluşturmaktır. Birçok sezgisel yöntemlerde başlangıç en iyi çözüm oluşturularak, en iyi çözüme ulaşılması sağlanmaktadır. Genetik algoritmaların buna benzer NP-zor problemlerin çözümündeki kuvvetli rastsal gücü popülasyon tabanlı sezgisel yöntemlerin de iddialı olduğunu göstermektedir. Melez meta-sezgisel yöntemlerin aksine bu yöntemle en optimum çözüme yakın çözüm elde edilebilmektedir. Bu yöntemle birlikte klasik sezgisel ve kesin çözüm yöntemlerine göre daha hızlı çözümlere ulaşılmaktadır. Bu çalışmada permütasyon tipi rastgele oluşturulan başlangıç popülasyonu dışında da birçok yöntem kullanılmaktadır. Aramanın daha geniş bir alanda yapılabilmesi ve bir önceki nesilde kalan minimum uzaklıktaki müşteriyi bir sonraki nesile aktarmak için mutasyon oranı seçimi önemli olmaktadır. Bu tez çalışmasında Wang ve Chen (2012)'in de kullandığı ve her problem çeşidinde sabit ve 0.01 olarak yer almaktadır.

Literatürde yer alan ve sezgisel yöntemlerle elde edilen en iyi sonuçlar ve bu çalışmada elde edilen en iyi sonuçlar tablo 5.1'de verilmektedir:

**Tablo 5.1:** Sonuç Değerleri.

| | EN İYİ DEĞERLER | | ÇALIŞMAYA AİT DEĞERLER | | |
|---|---|---|---|---|---|
| **Problem No** | **Toplam Uzaklık** | **Toplam Araç (Rota) Sayısı** | **Toplam Uzaklık** | **Toplam Araç (Rota) Sayısı** | **Sapma (%)** |
| **C101** | **828,940** | **10** | **828,940** | **10** | **%0** |
| **C104** | **824,780** | **10** | **826,556** | **10** | **%0** |
| **C205** | 588,880 | 3 | 655,924 | 3 | %11 |
| **C206** | 588,490 | 3 | 721,297 | 3 | %23 |
| **R105** | **1.377,110** | **14** | **1.378,000** | **14** | **%0** |
| **R208** | 726,820 | 2 | 773,800 | 2 | %6 |
| **C1_2_1** | **2.704,570** | **20** | **2.716,000** | **20** | **%0** |
| **C1_2_3** | 2.707,350 | 18 | 3.309,334 | 21 | %22 |
| **C1_2_5** | **2.702,050** | **20** | **2.705,280** | **20** | **%0** |
| **C2_2_6** | **1.857,350** | **6** | **1.860,880** | **6** | **%0** |
| **C2_2_8** | 1.820,530 | 6 | 2.290,170 | 6 | %26 |
| **C2_2_10** | 1.806,580 | 6 | 2.151,693 | 6 | %19 |

Başlangıç popülasyon büyüklüğünün doğru seçilmesi, çözüm kalitesini olumlu yönde etkilemiştir. Başlangıç popülasyon büyüklüğünün küçük seçilmesi çözüm uzayında daha kısıtlı arama yapılmasına neden olmuştur. Bu yüzden bazı problemlerde kromozom sayısı müşteri sayısı kadar belirlenmiştir. Mutasyon oranı sabit alınarak çözüm kümesinin yerel optimuma takılması önlenmiştir. Tezde kullanılan test problemlerinin çözümü için iterasyon sayısının artması olurlu çözüm kümesine ulaştırmayı garantilemiş olmasına rağmen, işlem süresini de arttırmış böylece düşük CPU performansına neden olmuştur.

C1_2_3 problemi dışında bütün test problemlerinde rota sayısı değişmemiştir.

Bu tez çalışmasında kullanılan genetik algoritmalar ile C101, C104, R105, C1_2_1, C1_2_5, C2_2_6 test problemlerinden elde edilen değerler, literatürde sezgisel yöntemlerle bulunan değerlerden yaklaşık %1 sapma göstermiştir. C205 test probleminde %11, C206 test probleminde, %23, R208 test probleminde %6, C1_2_3 test probleminde %22, C2_2_8 test probleminde %26, C2_2_10 probleminde %19'luk sapmalar elde edilmiştir. Sadece genetik algoritmalar kullanılarak optimum çözüme yakınsamak için başlangıç popülasyon büyüklüğü ve rastgele seçim önemli kriterler olduğundan dolayı bu konuda daha fazla deneme yapılmalıdır. Popülasyon

büyüklüğünün artmasıyla iterasyon sayısının da arttırılması olurlu çözüme daha uzun sürede ulaşılmasına olanak sağlar.

Bu çalışmada kullanılan yöntem, gerçek hayattaki daha büyük ve daha fazla müşteri odaklı araç rotalama problemlerinin çözümünde popülasyon tabanlı sezgisel yöntemlerin kullanılmasına yardımcı olacaktır. sonuçlara Daha iyi çözümlere daha kısa sürelerde ulaşabilmek için hem kesin çözüm yöntemleri hem de sezgisel ve metasezgisel yöntemleri kullanılarak hibrit sezgisel - metasezgisel yöntemlerin geliştirilmesi gerekmektedir. Başlangıç popülasyonu olarak rastsal sıralama kullanılarak yeni genetik algoritmalar yöntemi araç rotalama problemlerinin çözümünde kullanılabilir.

Kullanılan yöntemle daha yeni sezgisel yöntemler geliştirilerek daha iyi sonuçlara ulaşılması hedeflenmektedir. Fakat tek başına popülasyon tabanlı sezgisel yöntemlerin ZPARP'in çözümü için yeterli değildir. Zaman pencereli araç rotalama problemleri için daha basit ve uyarlanabilir hibrit meta-sezgisel yöntemlerin kullanılması hem CPU performansını hem de çözüme olumlu yönde katkı sağlayacaktır.

# KAYNAKLAR


Almoustafa, S., Hanafi, S. & Mladenovic´, N., 2013. New exact method for large asymmetric distance-constrained vehicle routing problem. *European Journal of Operational Research,* p. 386–394.

Bakırlı, G., Birant, D. & Kut, A., 2011. An incremental genetic algorithm for classification and sensitivity analysis of its parameters. *Expert Systems with Applications,* p. 2609–2620.

Balseiro, S., Loiseau, I. & J.Ramonet, 2011. An Ant Colony algorithm hybridized with insertion heuristics for the Time Dependent Vehicle Routing Problem with Time Windows. *Computers & Operations Research,* Issue 38, p. 954–966.

Bluma, C., Puchinger, J., Raidl, G. R. & Roli, A., 2011. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing,* Issue 11, p. 4135–4151.

Bräysy, O. & Gendreau, M., 2005. Vehicle Routing Problem with Time Windows, Part II: Metaheuristics. *TRANSPORTATION SCIENCE,* 39(1), p. 119–139.

Cheng, H., Yang, S. & Cao, J., 2013. Dynamic genetic algorithms for the dynamic load balanced clustering problem in mobile ad hoc networks. Issue 1381–1392.

Clarke, G. & Wright, J., 1964. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research,* pp. 568-581.

Cordeau, J.-F., Laporte, G., Savelsbergh, M. W. & Vigo, D., 2007. *Handbook in OR & MS.* 14 dü. basım yeri bilinmiyor:Elsevier B.V.

Cuervo, D. P., Goos, P., Sörensen, K. & Arráiz, E., 2014. An iterated local search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research,* p. 454–464.

Dantzig, G. B. & Ramser, J. H., 1959. The Truck Dispatching Problem. *Management Science,* pp. 80-91.

Desroisers, J., Soumis, F. & Desrochers, M., 1984. Routing with Time Windows by Column Generation.

Desrosiers, J., Dumas, Y., Solomon, M. M. & Soumis, F., 1995. *Handbooks in OR & MS.* 8 dü. basım yeri bilinmiyor:Elsevier Science B.V.

Ding, Q., Hu, X., Sun, L. & Wang, Y., 2012. An improved ant colony optimization and its application to vehicle routing problem with time windows. *Neurocomputing,* Issue 98, pp. 101-107.

Dorigo, M. & Gamberdella, L. M., 1996. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation,* Issue 53-66.

Drexl, M. & Schneider, M., 2015. A survey of variants and extensions of the location-routing problem. *European Journal of Operational Research,* p. 283–308.

El-Sherbeny, N. A., 2010. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University,* p. 123–131.

El-Sherbeny, N. A., 2010. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University,* Issue 22, p. 123–131.

Gauvin, C., Desaulniers, G. & Gendreau, M., 2014. A branch-cut-and-price algorithm for the vehicle routing problem with stochastic demands. *Computers & Operations Research,* p. 141–153.

Ghoseiri, K. & Ghannadpour, S. F., 2010. Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. *Applied Soft Computing,* p. 1096–1107.

Gong, Y.-J.ve diğerleri, 2012. Optimizing the Vehicle Routing Problem With Time Windows: A Discrete Particle Swarm Optimization Approach. *IEEE Transactions On Systems, Man, And Cybernetics,* Issue 2, p. 42.

Guillaume, C., Reynald, G. & Antoine, T., 2010. Genetic Algorithms and Finite Element Coupling For Mechanical Optimization. *Advance in Engineering Software,* pp. 422-426.

Ibaraki, T. ve diğerleri, 2008. An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. *Discrete Applied Mathematics,* p. 2050 – 2069.

Jin, J., Crainic, T. G. & Løkketangen, A., 2014. A cooperative parallel metaheuristic for the capacitated vehicle routing problem. *Computers & Operations Research,* p. 33–41.

Juan, A. A., Faulin, J., Pêrez-Bernabeu, E. & Jozefowiez, N., 2014. Horizontal Cooperation in Vehicle Routing Problems with Backhauling and Environmental Criteria. *Social and Behavioral Sciences,* p. 1133 – 1141.

Karaboğa, D., 2014. *Yapay Zeka Optimizasyon Algoritmaları.* 3. dü. Ankara: Nobel .

Karimi, H. & Seifi, A., 2012. Acceleration of Lagrangian Method for the Vehicle Routing Problem with Time Windows. *International Journal of Industrial Engineering & Production Research,* pp. 309-315.

Kaya, İ., 2012. Genetik Algoritmaların Optimal Güzergah Belirlenmesine Uygulanması. *Yüksek Lisans Tezi,* p. 31.

Keçeci, B., 2008. *Yüksek Lisans Tezi,* Ankara: Başkent Üniversitesi.

Kumar, S. N. & Panneerselvam, R., 2012. A Survey on the Vehicle Routing Problem and Its Variants. *Intelligent Information Management,* pp. 66-74.

Kuşçu, Ö., 2009. *Araç Rotalama Sistemlerinde Sezgisel Yöntemler.* Isparta:

Laporte, G., Hertz, A. & Gendreau, M., 1994. A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science,* pp. 1276-1290.

Liu, R. & Jiang, Z., 2012. The close–open mixed vehicle routing problem. *European Journal of Operational Research,* p. 349–360.

Luke, S., 2014. *Figure 0 The Mona Lisa, estimated.* 2 dü. San Francisco: George Mason University.

Lysgaard, J. & Wøhlk, S., 2014. A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. *European Journal of Operational Research,* p. 800–810.

Macedo, R. ve diğerleri, 2011. Solving the vehicle routing problem with time windows and multiple routes exactly using a pseudo-polynomial model. *European Journal of Operational Research,* p. 536–545.

Marinakis, Y. & Marinaki, M., 2014. A Bumble Bees Mating Optimization algorithm for the Open Vehicle. *Swarm and Evolutionary Computation,* Issue 15, pp. 80-94.

Mavrovouniotis, M. & Yang, S., 2015. Ant algorithms with immigrants schemes for the dynamic vehicle routing problem. *Information Sciences,* p. 456–477.

Melián-Batistaa, B., Santiagoa, A. D., AngelBello, F. & Alvarez, A., 2014. A bi-objective vehicle routing problem with time windows: A real case in Tenerife. *Applied Soft Computing,* p. 140–152.

Moghaddam, B. F., Ruiz, R. & Sadjadi, S. J., 2012. Vehicle routing problem with uncertain demands: An advanced particle swarm algorithm. *Computers & Industrial Engineering,* p. 306–317.

Montoya-Torres, J. R., Franco, J. L., Isaza, S. N. & Jiménez, H. F., 2015. A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering,* p. 115–129.

Nazif, H. & Lee, L. S., 2012. Optimised crossover genetic algorithm for capacitated vehicle routing problem. *Applied Mathematical Modelling,* p. 2110–2117.

Nguyen, P. K., Crainic, T. G. & Toulouse, M., 2014. A hybrid generational genetic algorithm for the periodic vehicle routing problem with time windows. *J Heuristics,* Issue 20, p. 383–416.

Norouzi, N., Sadegh-Amalnick, M. & Alinaghiyan, M., 2015. Evaluating of the particle swarm optimization in a periodic vehicle routing problem. *Measurement,* Issue 62, p. 162–169.

Osman, I. H., 1993. Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem. *Annals of Operations Research,* pp. 421-451 .

Önüt, S., Tuzkaya, U. R. & Doğaç, B., 2008. A particle swarm optimization algorithm for the multiple-level warehouse layout design problem. *Computers & Industrial Engineering,* Issue 54, p. 783–799.

Qureshi, A., Taniguchi, E. & Yamada, T., 2009. An exact solution approach for vehicle routing and scheduling problems with soft time windows. *Transportation Research ,* p. 960–977.

Rahimi-Vahed, A., Crainic, T. G., Gendreau, M. & Rei, W., 2015. Fleet-sizing for multi-depot and periodic vehicle routing problems using a modular heuristic algorithm. *using a modular heuristic algorithm,* p. 9–23.

Reed, M., Yiannakou, A. & Evering, R., 2014. An Ant Colony Algorithm for the Multi-Compartment Vehicle Routing Problem. *Applied Soft Computing,* p. 169–176.

Reeves, C. & Rowe, J., 2003. *Genetic Algorithms Principles and Perspectives A Guide to GA Theory.* 1st dü. New York: Kluwer Academic Publishers.

Repoussis, P. & Tarantilis, C., 2010. Solving the Fleet Size and Mix Vehicle Routing Problem with Time Windows via Adaptive Memory Programming. *Transportation Research Part C,* Issue 18, p. 695–712.

Selvi, V. & Umarani, R., 2010. Comparative Analysis of Ant Colony and Particle Swarm Optimization Techniques. *International Journal of Computer Applications,* Issue 4, p. 0975 – 8887.

Solomon, M. M., 1987. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research,* Issue 2, pp. 254-265.

Solomon, M. M., 2005. *VRPTW BENCHMARK PROBLEMS.* [Çevrimiçi] Available at: http://w.cba.neu.edu/~msolomon/problems.htm [Erişildi: 13 Haziran 2015].

Straßburg, J., Gonz´alez-Martel, C. & Alexandrov, V., 2012. Parallel genetic algorithms for stock market trading rules. *Procedia Computer Science ,* p. 1306 – 1313 .

Subramanian, A., Uchoa, E. & Ochi, L. S., 2013. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research,* p. 2519–2531.

Talbi, E.-G., 2009. *Metaheuristics From Design To Implementation.* New Jersey: John Wiley & Sons, Inc.

Tan, L., Lin, F. & Wang, H., 2015. Adaptive comprehensive learning bacterial foraging optimization and its application on vehicle routing problem with time windows. *Neurocomputing,* Issue 151, p. 1208–1215.

Taş, D., Jabali, O., Woelsen & Van, T., 2014. A Vehicle Routing Problem with Flexible Time Windows. *Computers & Operations Research,* p. 39–54.

Thangiah, S. R., 1995. *Applications Handbook of Genetic Algorithms: New Frontiers.* 2 dü. U.S.A.: CRC Press.

Toth, P. & Vigo, D., 2002. *The Vehicle Routing Problem.* 1. dü. Philadelphia: Society for Industrial and Applied Mathematics.

Tüfekçier, H., 2008. *İki Amaçlı Açık Araç Rotalama Problemi İçin Bir Çözüm Yaklaşımı.* Eskişehir.

Ursani, Z., Essam, D., Cornforth, D. & Stocker, R., 2011. Localized genetic algorithm for vehicle routing problem with time windows. *Applied Soft Computing,* p. 5375–5390.

Vansteenwegen, P. & Mateo, M., 2014. An iterated local search algorithm for the single-vehicle cyclic inventory routing problem. *European Journal of Operational Research,* Issue 237, p. 802–813.

Vidal, T., Crainic, T. G., Gendreau, M. & Prins, C., 2014. Implicit depot assignments and rotations in vehicle routing heuristics. *European Journal of Operational Research,* p. 15–28.

Wang, H.-F. & Chen, Y.-Y., 2012. A genetic algorithm for the simultaneous delivery and pickup problems with time window. *Computers & Industrial Engineering,* pp. 84-95.

Xie, W.-C., Chen, C.-M., Fan, S.-S. & Li, L.-L., 2014. A Vehicle Routing Optimization Method with Constraints Condition based on Max-Min Ant Colony Algorithm. *Applied Mathematics & Information Sciences,* Issue 8, pp. 239-243.

Yu, B. & Yang, Z. Z., 2011. An ant colony optimization model: The period vehicle routing problem with time windows. *Transportation Research,* p. 166–181.

# EKLER

## EK 1. GENETİK ALGORİTMA İÇİN MATLAB KODU***

```
function[routes;distance_Cus;fitnessAll]= geneticVRPTW()
tic
clear;

load Problem;                % Problem türüne göre veriler
çağrılacaktır.
[n m]=size(Problem);
M=n;                         % müşteri sayısı
K=25;                        % araç sayısı
capacity=200;                % her aracın kapasitesi
Pop=30;                      % popülasyon büyüklüğü
xy=Problem(2:M,2:3);         % koordinatlar
timLim=Problem(2:M,5:6);     % zaman aralığı
demand=Problem(2:M,4);       % talep bilgileri
serTime=Problem(2,7);        % 90 veya 10 birim hizmet süresi
Total_cost=0;
veCap=0;
count=0;
% bütün müşterilerin depoya olan uzaklıklarını hesaplama
disDepot=zeros(1,M-1);
for ii=1:M-1
    disDepot(ii)=sqrt((Problem(ii+1,2)-
Problem(1,2))^2+(Problem(ii+1,3)-Problem(1,3))^2);
end
% Rota oluşturmaya başlama
routes={};
route_number=zeros(1,M-1);
distance_Cus=ones(M)*Inf;
for ii=1:M-2
    for j=ii+1:M-1
        arrival_travel_time=sqrt((xy(ii,1)-
xy(j,1))^2+(xy(ii,2)-xy(j,2))^2)+serTime;
dDistance=arrival_travel_time-serTime;% aynı rotada bulunan
iki noktanın birbirine uzaklığı
        if (timLim(ii,2)+arrival_travel_time>timLim(j,1) &&
timLim(ii,1)+arrival_travel_time<timLim(j,2))
            route_number(ii)=route_number(ii)+1;
            routes{ii}(route_number(ii))=j;
            distance_Cus(ii, j)=arrival_travel_time;
```

```
            veCap=veCap+demand(ii);
            Total_cost=Total_cost+dDistance;
        end
        if (timLim(j,2)+arrival_travel_time>timLim(ii,1) &&
timLim(j,1)+arrival_travel_time<timLim(ii,2))&&
veCap<capacity
            route_number(j)=route_number(j)+1;
            routes{j}(route_number(j))=ii;
            distance_Cus(j, ii)=arrival_travel_time;
            veCap=veCap+demand(ii);
        end
    end
end
chrom={};
numVehi=zeros(1,Pop*2);
fit=ones(1,Pop*2);
for ii=1:Pop*2
        [chrom{ii},numCus]=randomSelect(demand,
route_number,distance_Cus,capacity);
        if (numCus==M)
            break;
        end
end

chromChild={};
d={};
iter=1000;
fitnessAll=zeros(iter,Pop);
firstMinimum=0;
nextIndex=0;
for j=1:iter
    random_select_child=randperm(Pop);
for ii=1:Pop

chromChild{ii}=crossOver(chrom{random_select_child(ii)},
capacity,demand);
end
for ii=1:Pop
    chrom{ii}=chromChild{ii};
             if (numCus==M)
            break;
        end
    fit(ii)=fitnessdiss(chrom{ii},dDistance,disDepot);
  end
[minChrom, minIndex]=min(fit);
if (minChrom==firstMinimum)
 break;
end
firstMinimum=minChrom;
```

```
nextIndex=minIndex;
lend=length(chrom{nextIndex});
d{j}=[];
for k=1:lend
    d{j}=[d{j},0,chrom{nextIndex}{k}];
Route_VRP=Routes{1,k};
        r=length(Route_VRP);
        SSum=0;
        for t=1:r-1
         fitD=fitD + disDepot (Route_VRP
(t,1))+disDepot(Route_VRP(t,end);
    subRoute=SSum+fitD +
(distance(Route_VRP(t),Route_VRP(t+1))) ;
            SSum=subRoute;
    end
    fitAll(j,:)=fit;
  TotalDistance=0;
  DistanceSets(k,:)=[SSum];
  TotalDistance=sum(DistanceSets);
    end
end
resultChrom=chrom{lastIndex};
TotalDistance=TotalDistance
Toc
```

*** Bu tezde yer alan Matlab kodları mevcut durumda çalışmamaktadır. Bazı satırlar ve fonksiyonlar yazar tarafından silinmiştir.

# ÖZGEÇMİŞ

| Kişisel Bilgiler | |
|---|---|
| Adı Soyadı | Çağrı KURAM |
| Doğum Yeri | GEBZE |
| Doğum Tarihi | 10.01.1988 |
| Uyruğu | ☑ T.C. ❒ Diğer: |

| Eğitim Bilgileri | |
|---|---|
| **Lisans** | |
| Üniversite | Sakarya Üniversitesi |
| Fakülte | Mühendislik Fakültesi |
| Bölümü | Endüstri Mühendisliği |
| Mezuniyet Yılı | 2010 |

| Yüksek Lisans | |
|---|---|
| Üniversite | İstanbul Üniversitesi |
| Enstitü Adı | Fen Bilimleri Enstitüsü |
| Anabilim Dalı | Endüstri Mühendisliği Anabilim Dalı |
| Programı | Endüstri Mühendisliği Programı |
| Mezuniyet Tarihi | 2016 |